UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI FISICA "E.R. CAIANIELLO"

DOTTORATO DI RICERCA IN SCIENZE E TECNOLOGIE
DELL'INFORMAZIONE, DEI SISTEMI COMPLESSI E DELL'AMBIENTE
XIII CICLO – NUOVA SERIE

TESI DI DOTTORATO IN INFORMATICA

METODI DELLA COMPLESSITÀ E ALGORITMI PER IL DECISION MAKING NEL CONTESTO DEL TRADING FINANZIARIO

Antonino Amorosia

Tutor: *Ch.mo Prof. Gerardo Iovane*

Coordinatore: *Ch.mo Prof. Roberto Scarpa*

A. A. 2015/2016

Sommario

Il lavoro di ricerca si colloca nell'ambito dei sistemi di supporto alle decisioni, nello specifico nella loro evoluzione in Sistemi Decisionali Automatizzati (ADS) per il Financial Computing. Riprendendo il lavoro di tesi magistrale, il quale ha visto come argomento la teoria delle decisioni e i sistemi di supporto alle decisioni per la sicurezza del territorio, si è deciso di spostare il focus su argomenti attuali di ricerca, quali appunto la finanza computazionale e il trading automatico, calando il problema nell'ambito dei sistemi di supporto alle decisioni. Tale scelta è frutto delle molteplici variabili in gioco in tali contesti, che è necessario valutare al fine di permettere al decisore, in questo caso il trader, di attuare le proprie strategie operative. Inoltre il forte impatto che ha la componente umana ed emotiva su tali sistemi e la necessità di automatizzare specifici processi propri delle tecniche di High Frequency Trading, ha condotto la ricerca verso soluzioni quanto più possibile automatizzate, ovvero soluzioni software in grado di acquisire dati dall'ambiente circostante, manipolarli secondo la modellazione matematica su cui si basa l'infrastruttura software, valutare quali sono le possibili strategie operative per il particolare stato del sistema e, infine, porle in essere secondo i vincoli imposti dal decisore.

Questa dissertazione si sviluppa dapprima con una introduzione riguardante i mercati finanziari, con un breve approfondimento sul mercato Forex, ovvero il dominio applicativo di questo lavoro di ricerca.

Il capitolo due riguarda una analisi dal punto di vista teorico e metodologico delle varie classificazioni dei Sistemi di Supporto alle Decisioni e dei Sistemi Decisionali Automatizzati, con un breve cenno sull'High Frequency Trading, ramo di tale settore di ricerca particolarmente florido dal punto di vista della ricerca.

Il capitolo tre descrive l'infrastruttura della piattaforma di trading utilizzata e le necessarie tecnologie di basso livello, sviluppate per permettere agli applicativi di alto livello di comunicare con tale piattaforma, e quindi di realizzare l'operatività;

Nel capitolo quattro, essendo l'infrastruttura software particolarmente complessa, ovvero integrando non soltanto le componenti algoritmiche per l'attività di trading vera e propria, ma fornendo anche dei servizi accessori, si descrivono due componenti del sistema complesso, ovvero un software per l'analisi di serie storiche estrapolate dalla piattaforma e un calendario automatico per le news finanziarie, entrambi utilizzati dagli algoritmi di trading.

Dovendo rendere disponibile per il trader un sistema di gestione multipla di account con la necessità di profilazione del rischio finanziario, nel capitolo cinque si descrive la struttura di un sistema client/server via web server, atto a permettere il cloning dell'operatività di un account master da parte di utenti slave.

Infine, nei capitoli sei e sette vengono ampiamente sviscerati i due applicativi di trading realizzati, ovvero il primo modellato per operare in momenti di spiccata volatilità del mercato, l'altro invece destinato all'apertura di posizioni OCO secondo particolari condizioni dello stesso. Questo secondo algoritmo è inoltre dotato di un modulo di auto apprendimento, atto ad addestrare sulle serie storiche il suo comportamento, adattando quindi i propri vincoli alla condizione in cui si trova l'ambiente circostante;

L'ultimo capitolo mostra la metodologia di test degli algoritmi sviluppati, con annessa analisi dei risultati ottenuti.

1 Introduzione al Financial Computing ed ai Mercati Finanziari

Il termine "mercato finanziario" si riferisce a qualunque mercato nel quale acquirenti e venditori partecipano ad uno scambio mediante contrattazione di beni quali azioni, obbligazioni, fondi comuni di investimento, coppie di valute, futures, options, ecc.

Una classificazione generale dei mercati finanziari può essere la seguente:

- Mercati principali
    - Mercato azionario
    - Mercato obbligazionario e titoli di stato
    - Cambi valutari
- Mercati per prodotti derivati

Ulteriori classificazioni possono essere realizzate in base alla tipologia e alla durata temporale degli scambi. Fondamentalmente, i mercati finanziari sono caratterizzati da regole di scambio definite da autorità di tutela e controllo della trasparenza sulle quotazioni degli strumenti finanziari. Un esempio di questi enti di regolamentazione è l'FSA (Financial Services Authority), ovvero un organismo indipendente non governativo atto a regolamentare i servizi finanziari nel Regno Unito, ed è considerato uno dei maggiori enti di regolamentazione dell'area euro.

I mercati per i quali invece non esiste un ente di regolamentazione delle contrattazioni, sono definiti ***OTC*** (Over-The-Counter). I mercati di questa tipologia portano spesso alla creazione di diversi sottomercati indipendenti con quotazioni che possono essere anche molto diverse in momenti specifici. A tale tipologia di mercato appartiene il FOREX.

1.1 Il Mercato FOREX

Il *FOREX*, ***FOR****eign* ***EX****change* ovvero cambio tra valute estere, è un immenso mercato finanziario esteso a livello planetario, che consente di acquistare e/o vendere valute internazionali.

Il mercato del Forex viene utilizzato da differenti categorie di utenti; in particolare troviamo coloro che utilizzano tale mercato motivati dall'esigenza di avere scambi commerciali tra paesi che usano monete differenti, coloro i quali hanno reso questa tipologia di mercato fonte di guadagno, detti retail o speculatori, e le istituzioni, come banche, Fondi di investimento ed assicurazioni.

Il Forex, differentemente da altre tipologie di mercati valutari, è non regolamentato ovvero non vi sono delle quotazioni ufficiali attraverso le quali poter effettuare acquisti o vendite di cross valutari, bensì le quotazioni vengono fornite agli attori sul mercato da un fornitore di tale servizio, il Broker, ovvero una entità giuridica che agisce come ente di intermediazione, mettendo insieme compratori e venditori in cambio di una commissione. Fornisce inoltre la piattaforma ed un conto di trading, sul quale il trader deposita i fondi utilizzati per le azioni di compravendita.

Il mercato del FOREX solitamente si divide in due macro-categorie di trader, di seguito definite [1]:

- **Mercato Interbancario**: ovvero la parte di mercato FOREX relativo alle transazioni tra istituti appartenenti alle seguenti categorie, i quali hanno accesso diretto al mercato:
    - Banche centrali (Fed Americana, BCE)
    - Banche commerciali (Deutsche Bank, Barclays)
    - Istituzioni Finanziarie (Fondi pensione, fondi d'Investimento)

- **Mercato al Dettaglio**: ovvero riguarda la parte degli investimenti del mercato FOREX realizzati da piccoli speculatori o investitori. Le transazioni avvengono tramite l'intermediazione di un broker.
    - Fondi speculativi (Macro Hedge Funds)
    - Corporazioni (Compagnie impegnate nell'Import/Export)
    - Privati

I vantaggi nel preferire il mercato del FOREX rispetto, ad esempio, al mercato azionario, sono molteplici. Forse la più importante riguarda l'impossibilità di "drogare" tale mercato. Infatti a differenza dell'azionario, nel quale il ruolo svolto dall'attività di grandi attori (ad esempio fondi di investimento o banche) può influenzare l'andamento dello strumento finanziario, il mercato FOREX, vista l'enorme liquidità che lo caratterizza (ovvero circa 5300 Miliardi di dollari al giorno, secondo i dati aggiornati ad aprile 2013), non può essere influenzato da immissioni di capitali anche cospicui da parte di tali attori.

Inoltre l'assenza di intermediari permette di avere dei costi molto bassi per poter operare sul mercato. Difatti il FOREX, a differenza dell'azionario, è un mercato decentralizzato, ovvero la compravendita avviene attraverso il broker che, al fine di essere concorrenziale e quindi trarre a sé nuovi traders, pone delle commissioni esigue alle operazioni poste in essere.

La possibilità di operare 24 ore su 24, escluso il week end, è un altro fattore aggiunto che slega l'attività di trading dall'attività delle "piazze" dei mercati azionari. Il minor numero di strumenti da analizzare, a differenza degli indici azionari, solitamente composti da centinaia di titoli, permette di effettuare una analisi del mercato razionale.

Da non dimenticare la possibilità data dai Broker di aprire conti sfruttando la leva finanziaria, un concetto che permette di operare sul mercato amplificando la portata delle proprie transazioni, con un effetto moltiplicatore. Tale concetto è molto apprezzato dai traders medio-piccoli in quanto, con somme relativamente piccole di capitale inziale possono ottenere, se ben commisurate al rischio, profitti considerevoli. Tale moltiplicatore, ovviamente, impatta sia sui profitti, che saranno incrementati di un fatto moltiplicativo pari alla leva, sia sulle perdite. Ad esempio, supponendo di avere un conto con leva 1:100 e di aprire una posizione di 10€, la posizione equivalente sarà di 1000€, con conto di trading in nozionale.

Tutti questi fattori hanno permesso al mercato del FOREX di espandersi, soprattutto negli ultimi anni, in maniera capillare tra piccoli e medi trader, a tal punto che anche le principali banche, (vedi ad esempio Fineco), mettono a disposizione del correntista una propria piattaforma di trading.

1.1.1 Analisi Fondamentale

L'analisi fondamentale è una tecnica di valutazione degli strumenti finanziari di interesse, basata principalmente sull'analisi dei fattori qualitativi, quantitativi e macro economici che possono influenzare l'andamento del suddetto strumento. L'analisi fondamentale ha lo scopo di suggerire al trader una eventuale sopravvalutazione o svalutazione corrente dello strumento finanziario, così da poter stabilire un conveniente ingresso a mercato. Solitamente l'analisi fondamentale viene utilizzata per investimenti a lungo termine, ed è considerata in contrapposizione all'analisi tecnica. Il ruolo svolto dall'analista fondamentale, che è colui che realmente effettua l'analisi, definita *Expert Opinion,* è alquanto complesso, difatti è necessario disporre di spiccate doti analitiche nonché essere in grado di esprimere un giudizio oggettivo sullo strumento finanziario che si è scelto di analizzare.

Hanno influenza sui mercati e quindi sono di interesse per l'analisi fondamentale la pubblicazione di dati economici dei paesi cui è correlato lo strumento finanziario in esame o eventi politici di rilievo.

Nel FOREX, data la natura del mercato nel quale il ruolo o le azioni di una singola azienda o investitore non possono influenzare l'andamento del mercato stesso visto l'enorme numero di volumi scambiati, si preferisce affiancare ad essa l'analisi tecnica.

1.1.2 Analisi Tecnica

Nel forex l'analisi tecnica è lo studio dell'andamento futuro del mercato basato sull'analisi dei livelli di prezzo storici, i prezzi correnti e il volume, al fine di prevedere i futuri movimenti del mercato. A tali informazioni l'analista tecnico affianca indicatori statistico/matematici, utili nell'analisi dell'andamento dello strumento finanziario.

L'analisi tecnica si basa principalmente sulle idee di **Charles Dow**, primo editore del Wall Street Journal ed ideatore dei primi indici di borsa (tra cui il Dow-Jones), ovvero:

1. ***Il mercato è formato da tre tendenze (trend)****: trend primario (nel lungo periodo), secondario (nel medio periodo) e minori (nel breve periodo).*
2. ***Le tendenze si suddividono in tre fasi****: accumulo (solo le "mani forti" investono), partecipazione del pubblico (ritorno della fiducia nel mercato) e distribuzione (tutti investono, momento di euforia del mercato)*
3. ***Il mercato azionario sconta tutte le notizie****: i prezzi delle azioni incorporano e riflettono velocemente tutte le informazioni che le riguardano*
4. ***Le medie del mercato azionario devono confermarsi a vicenda****: occorre che il trend di un settore sia confermato anche dall'andamento dei settori a lui collegati*
5. ***Le tendenze devono essere confermate dal volume****: il volume in genere segue i prezzi, però se l'andamento è caratterizzato anche da forti volumi, allora è veritiero*
6. ***Le tendenze esistono fino a che dei segnali definitivi dimostrano che sono terminate****: brevi periodi di movimenti in controtendenza non costituiscono inversione di rotta del mercato salvo un persistere degli stessi*

L'analisi tecnica, basandosi sulla teoria di Dow suddivide in tre tipologie le fasi nelle quali il mercato può trovarsi, ovvero

- *Fase di accumulazione*: è la fase durante la quale i mercati vedono come attori i grandi investitori, le banche, i fondi immobiliari, ovvero coloro che hanno una visione globale più completa del mercato stesso. Tali attori solitamente entrano in contro tendenza. Durante questa fase il mercato si "carica", ma i prezzi non subiscono variazioni.
- *Fase di speculazione*: è la fase successiva a quella di accumulazione, nella quale la restante parte dei trader iniziano ad immettere liquidità nel mercato. È una fase caratterizzata da forti oscillazioni del prezzo (quindi forte volatilità), il quale alla fine di questa fase raggiungerà potenzialmente un nuovo massimo, assestandosi.
- *Fase di distribuzione*: è l'ultima fase del mercato, durante la quale gli attori della prima fase decidono di invertire la tendenza del mercato, alleggerendo le proprie posizioni. In questa fase, gli attori medio-piccoli, spaventati dal movimento in contrapposizione del mercato, decidono di vendere le proprie posizioni, creando nuovamente un deprezzamento dello strumento e comportando l'inizio di un nuovo ciclo di accumulazione-speculazione-distribuzione.

L'analisi tecnica viene effettuata analizzando i dati storici presenti sulle piattaforme di trading, rappresentati principalmente tramite tre tipologie di grafici, ovvero

- Grafico a Linea
- Grafico a Barre
- Grafico a Candele Giapponesi.

Il grafico a Candele Giapponesi, il più utilizzato nella visualizzazione degli andamenti degli strumenti finanziari, è anche sfruttato nell'analisi tecnica in quanto racchiude molteplici informazioni in una rappresentazione compatta e di facile interpretazione. Ogni candela giapponese, in base al selezionato intervallo temporale, detto *timeframe*, è costituita dai seguenti quattro livelli di prezzo:

- *Open*: Prezzo di apertura dell'intervallo temporale in esame;
- *High*: Prezzo massimo raggiunto nell'intervallo temporale;
- *Low*: Prezzo minimo raggiunto nell'intervallo temporale;
- *Close*: Prezzo di chiusura dell'intervallo temporale

Una ulteriore informazione disponibile nei grafici a candele giapponesi è il *Volume*, ovvero il numero dei contratti scambiati nell'intervallo temporale.

Le candele giapponesi assumono anche una colorazione, in base all'andamento dello strumento finanziario nell'unità temporale scelta. Nello specifico:

- Se il prezzo di apertura è minore del prezzo di chiusura, la candela si definisce rialzista, ed assume solitamente la colorazione verde;
- Se il prezzo di apertura è maggiore del prezzo di chiusura, la candela si definisce ribassista, ed assume solitamente la colorazione rossa.

La distanza tra il prezzo di apertura e di chiusura costituiscono il corpo della candela, la distanza tra il massimo e il prezzo di apertura/chiusura è definita Ombra Superiore, mentre la distanza tra il minimo e il prezzo di apertura/chiusura è definita Ombra Inferiore (Vedi Figura 1).

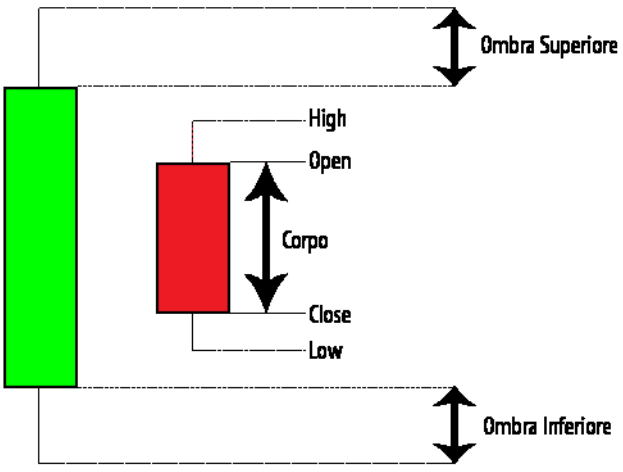


Figura 1 - Esempio di Candele Giapponesi

L'analisi dei dati storici permette all'analista tecnico, il quale crede fortemente nell'assunzione della ciclicità del mercato, che si possa prevedere, ma non in maniera scientifica, un ipotetico livello di prezzo che lo strumento finanziario in esame potrebbe raggiungere nell'immediato futuro.

L'analisi tecnica prevede l'individuazione sul grafico dei prezzi, analizzato a differenti timeframe, di eventuali elementi che possano mostrare l'inizio di una nuova fase di mercato.

Di seguito si descrivono i principali indicatori statistico/matematici presenti sulle principali piattaforme di trading, ed utilizzati per l'analisi tecnica.

1.2 Principali Indicatori del forex

Gli indicatori tecnici sono degli strumenti statistici che interpretano il comportamento dei mercati valutari in base all'andamento dei prezzi pregressi delle valute. Tali indicatori possono essere suddivisi in due tipologie principali:

- *Leading*: indicatori che, interpretando le informazioni sull'andamento del cross, forniscono un segnale prima che si verifichi una inversione del trend in esame;
- *Lagging*: a differenza dei precedenti, essi forniscono informazioni in ritardo, ovvero soltanto dopo che il nuovo trend si è formato

Entrambe le tipologie di indicatori soffrono di problematiche relative al rischio su perdite e profitti, difatti utilizzando soltanto indicatori Leading, la percentuale di rischio cresce proprio perché tali indicatori cercano di prevedere l'andamento del trend; d'altra parte utilizzando solamente indicatori Lagging, si apriranno posizioni solamente quando il nuovo trend si è sviluppato o, al massimo, quando è ancora in fase di sviluppo. Ciò provoca una perdita di quella parte di profitto non considerata, proprio perché l'indicatore ha fornito una informazione in ritardo rispetto lo sviluppo del trend stesso.

Di seguito si elencano le tipologie e gli indicatori più comunemente utilizzati nell'ambito del Forex.

Le descrizioni degli indicatori di seguito riportati e le figure di esempio degli stessi, tratte dalla piattaforma, fanno riferimento principalmente alla documentazione ufficiale della MetaQuotes Software Corp. [2]. Inoltre per i grafici a candele giapponesi, le stesse avranno colore bianco se rialziste, nere se ribassiste.

1.2.1 Panoramica sugli indicatori

Indicatori di Trend: A tale categoria appartengono, ad esempio, il MACD, il SAR parabolico e le varie medie mobili. Oltre a questi ovviamente ne esistono di svariati che fanno parte dello stesso gruppo, ovvero quelli che vengono utilizzati per l'identificazione della tipologia di trend che caratterizza il mercato osservato. Questi indicatori possono aiutare il trader a prendere la decisione corretta relativamente alla tipologia di pozione da aprire (Buy o Sell), dato che la base per molti traders è quella di aprire posizioni solamente in presenza di trend prevalenti, permettendo di individuare quale sia il trend dominante in un cross di valute e scegliere quindi in quale direzione operare [3].

Indicatori di Momentum: A questa categoria di indicatori appartengono il Relative Strenght Index (RSI), l'oscillatore stocastico ed il Commodity Channel Index, ovvero raggruppa sostanzialmente gli oscillatori che vengono utilizzati, principalmente, per determinare le situazioni di ipercomprato (Over-Bought) e di ipervenduto (OverSold) di uno strumento finanziario. In tale direzione, questi oscillatori possono aiutare i traders a confermare un segnale fornito da un altro indicatore, e quindi risultano utili per individuare l'inizio di un nuovo trend oppure l'esaurimento di un movimento del mercato [3].

Indicatori di Volume: Intuitivamente, questa categoria di indicatori permette di valutare il volume degli scambi che vengono effettuati su di uno specifico strumento finanziario. Esempi di indicatori di volume sono il Money Flow Index e il Force Index. Il vantaggio dato da questi indicatori è quello di fornire un'informazione riguardo la forza di un segnale. Un cross supportato da un alto volume di scambi,

rappresenta un segnale molto più forte di un movimento basato su un volume basso. Inoltre molto spesso i volumi sono maggiori durante le ore diurne [3].

Indicatori di Volatilità: Gli indicatori appartenenti a tale categoria sono utilizzati per mostrare il range di variazione di uno strumento finanziario, in combinazione alla forza della fluttuazione dei prezzi. Le informazioni fornite da questi indicatori, definibili come indicatori di range, risultano molto utili al trader in quanto permettono di individuare quali sono i momenti più indicati per operare sul mercato. A tale categoria appartengono, ad esempio, l'Average True Range e le Bande di Bollinger [3].

1.2.1.1 *Moving Average*

Le medie mobili (Moving Average) inizialmente sviluppate da R. Colby e T. Meyers [4], semplicemente misurano l'andamento medio del cross durante un determinato arco temporale, definito periodo. Tali medie effettuano lo smooth dei dati su cui sono calcolate, permettendo di analizzare il trend di un cross nel periodo prefissato.

Tale indicatore, all'arrivo di una nuova candela, esegue lo shift della finestra di analisi, definito appunto periodo della media, permettendo così di includere la nuova informazione nel calcolo, rimuovendo l'elemento più vecchio.

Gran parte degli indicatori descritti nei paragrafi successivi fanno uso delle varie medie mobili per poter effettuare lo smoothing del segnale ottenuto, così da permettere una analisi più accurata, eliminandone il rumore. Esistono principalmente quattro tipologie di medie mobili, descritte di seguito nel dettaglio.

SMA - *Simple Moving Average*: mostra il prezzo medio per un dato periodo di tempo. Calcolata secondo la seguente formulazione, su un arco temporale di periodo P:

$$SMA_j = \frac{1}{P} \sum_{j=i-P}^{i} Close_j$$

Il nuovo valore della media viene calcolato tramite uno shift del periodo considerato, ovvero facendo rientrare il prezzo della nuova candela nella sommatoria sui *Close*, ed escludendo il *Close* della candela più vecchia.

EMA - *Exponential Moving Average*: Tale media, tramite un fattore di smorzamento, da maggiore importanza ai dati recenti, così da reagire più velocemente al cambiamento dei prezzi, rispetto alla SMA. Di seguito la formulazione calcolata su periodo P:

$$s_1 = x_0$$

$$s_i = \alpha x_i + (1-\alpha)\, s_{i-1} \qquad con\ \boldsymbol{\alpha}\ fattore\ di\ smorzamento$$

$$\alpha = \frac{2}{P+1}$$

$$EMA_j = Close_j \cdot \alpha + (1-\alpha) \cdot EMA_{j-1}$$

SMMA - *Smoothed Moving Average*: La SMMA è una sorta di fusione tra la SMA e la EMA, con la differenza di avere un periodo P su cui viene calcolata più lungo rispetto alle altre. Inoltre nella SMMA, il dato più vecchio non viene mai rimosso dal calcolo, ma bensì esso avrà un peso molto piccolo sul calcolo del nuovo valore di SMMA.

Il primo valore della media è calcolato come una media semplice SMA:

$$SMMA_1 = \frac{1}{P} \sum_{j=i-P}^{i} Close_j$$

I successivi sono calcolati secondo la formula:

$$PrevSum = SMMA_{i-1} \cdot P$$

$$SMMA_i = \frac{(PrevSum - SMMA_{i-1}) + Close_i}{P}$$

LWMA - *Linear Weighted Moving Average*: Assegna dei pesi ai dati interessati dal calcolo della media, dando maggior importanza ai dati più recenti rispetto che a quelli meno recenti.
La formulazione teorica della LWMA è:

$$p_j = componente\ j - esima$$

$$w_j = peso\ della\ componente\ j - esima$$

Al passo j-esimo si avrà quindi

$$weight_J = \sum_{j}^{1} w_j$$

Da cui

$$WMA_J = \frac{1}{weight_j} \sum_{j}^{1} p_j w_j$$

Per il passo successivo si avrà:

$$lsum_i = lsum_{i-1} - Close_{i+P} + Close_{i+1}$$

$$LWMA_{i+1} = sum_i - lsum_i + Close_{i+1} \cdot P$$

In Figura 2 è possibile vedere un esempio delle quattro moving average appena descritte, nello specifico *Simple Moving Average* in rosso, *Exponential Moving Average* in azzurro, *Smoothed Moving Average* in verde ed infine *Linear Weighted Moving Average* in blu.

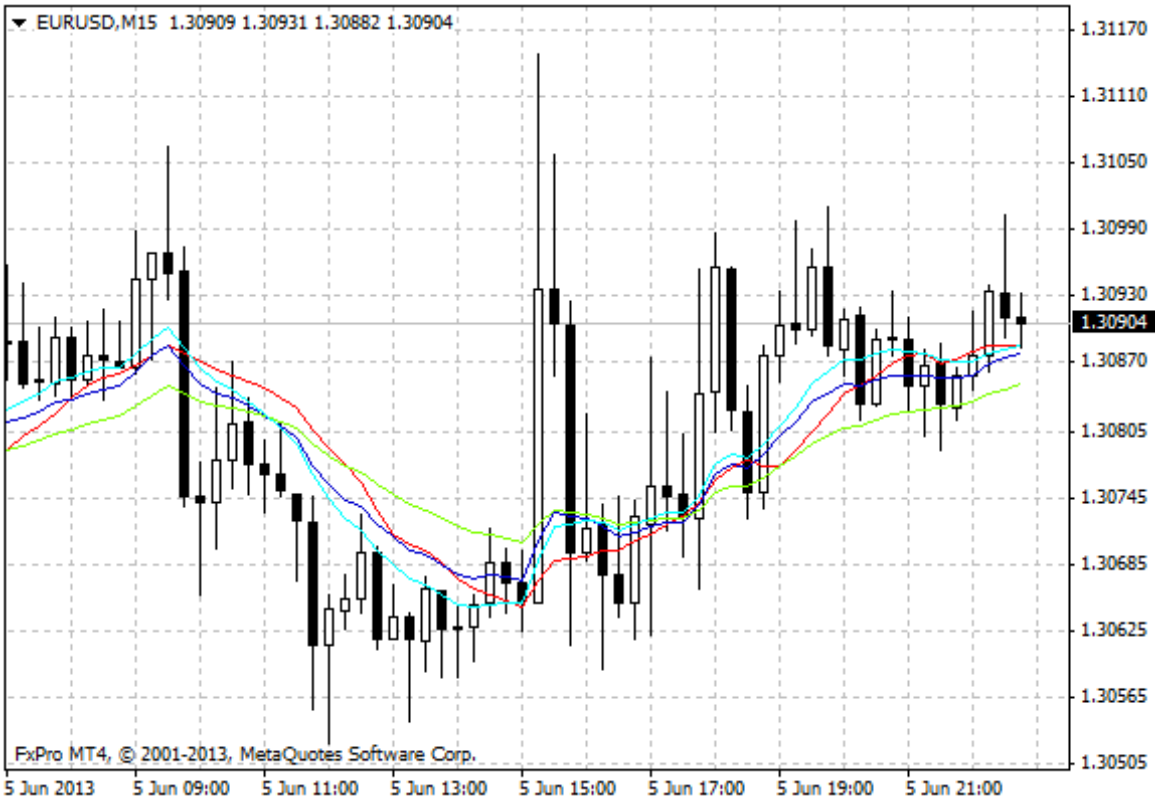


Figura 2 - Moving Average

1.2.1.2 *Accelerator/Decelerator Oscillator*

L'indicatore, ideato da Bill Williams [5], [6], è rappresentato in un istogramma separato dal grafico dei prezzi. Il calcolo di questo indicatore si basa su una differenza tra una media mobile sul breve periodo (periodo = 5) ed una media mobile sul lungo (periodo = 34).

Quando il valore di Accelerator Oscillator (AC) della barra attuale è maggiore del valore della precedente barra, la barra dell'istogramma viene colorata di *verde*. Viceversa quando la barra AC attuale è minore rispetto alla barra AC precedente la barra viene colorata di *rosso*.

Quando il valore di AC è vicino alla linea dello zero, significa che la forza trainante del mercato è bilanciata, ovvero il numero di operatori in buy bilancia quello in sell.

Se il valore dell'indicatore si trova sopra la linea dello zero, solitamente tale andamento prosegue, dunque possiamo aspettarci un aumento dei prezzi di mercato.

Normalmente una barra verde di AC ci suggerisce di aprire una posizione di tipo buy, al contrario una barra rossa può essere considerata come un'indicazione di aprire una posizione in sell (v. Figura 3).

Figura 3 - Accelerator/Decelerator Oscillator

Si ottiene quindi la seguente formulazione teorica:

$$MA' = SMA_i = \frac{1}{P}\sum_{j=i-P}^{i} Close_j \qquad con\ P = 5$$

$$MA'' = SMA_i = \frac{1}{P}\sum_{j=i-P}^{i} Close_j \qquad con\ P = 34$$

$$AC_i = MA'_i - MA''_i$$

1.2.1.3 *Accumulation/Distribution Index*

Si parla di "Fase di Accumulazione" quando, nell'ottica di un ragionamento intra-day (cioè in una singola giornata di operatività), si registra un momento di pausa delle contrattazioni, segnale che a breve si avrà un movimento al rialzo del titolo da cogliere al volo con un entrata/uscita rapida dal mercato. ADI, Sviluppato da Marc Chaikin [7], è appunto l'oscillatore che ci mostra indicazioni sull'imminente verificarsi di tale fase. Un valore alto di tale rapporto indica una fase di accumulazione, viceversa, un valore basso di tale rapporto indica una fase di distribuzione.

La formulazione è la seguente:

$$\text{MoneyFlow}Volume_i = \frac{(Close_i - Low_i) - (High_i - Close_i)}{High_i - Low_i} \times Volume_i$$

$$ADI_i = MoneyFlowVolume_i + MonewFlowVolume_{i-1}$$

Un esempio di tale indicatore è mostrato in Figura 4.

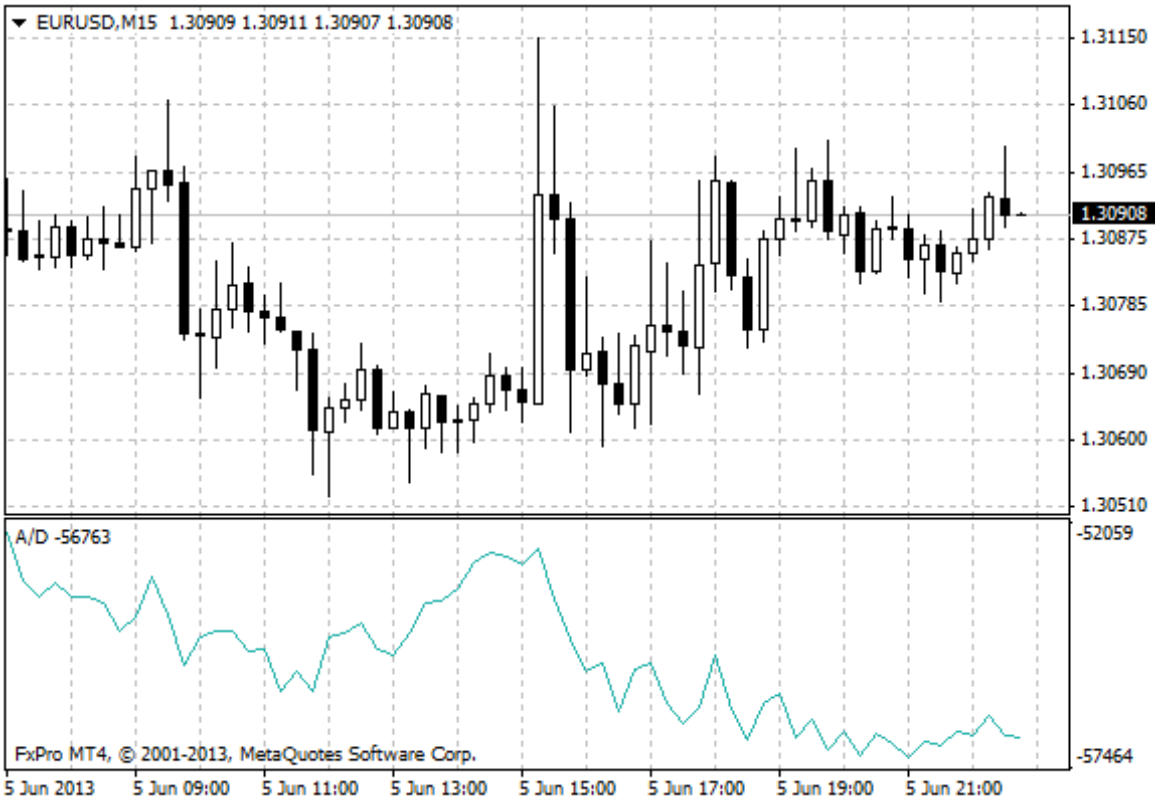


Figura 4 - Accumulation/Distribution Index

1.2.1.4 *Average Directional Movement Index (ADX)*

ADX è un indicatore usato in analisi tecnica per identificare la forza di un trend. L'ADX non è direzionale, ovvero quantifica la forza di un trend senza tener conto del fatto che questo sia rialzista o ribassista [8].

Questo indicatore è composto da due componenti: +DI e -DI, ovvero il *Directional Indicator positivo* ed il *Directional Indicator negativo*.

Quando DI+ ha pendenza positiva, significa che il trend positivo si sta rafforzando. Molti traders aspettano che il +DI incroci il –DI verso l'alto e interpretano questo segnale come inizio di un trend rialzista. Viceversa in caso di trend ribassista.

Solitamente quando i valori di ADX sono in crescita, ovvero quando l'indicatore supera in crescita la soglia dei 20 punti, si ritiene che il trend in esame si stia rafforzando, mentre quando diminuisce e si porta al di sotto della soglia dei 40 punti si ritiene che il trend si stia esaurendo e che probabilmente sta per verificarsi un'inversione di tendenza.

Nel grafico sottostante possiamo notare come gli incroci delle due linee +DI (verde) e –DI (marrone) determinino prima un trend negativo e poi uno positivo, e come ADX (celeste) sia utile per capire se dopo l'incrocio ci sia un trend abbastanza forte da giustificare l'apertura di un'operazione. Un esempio di tale indicatore è mostrato in Figura 5.

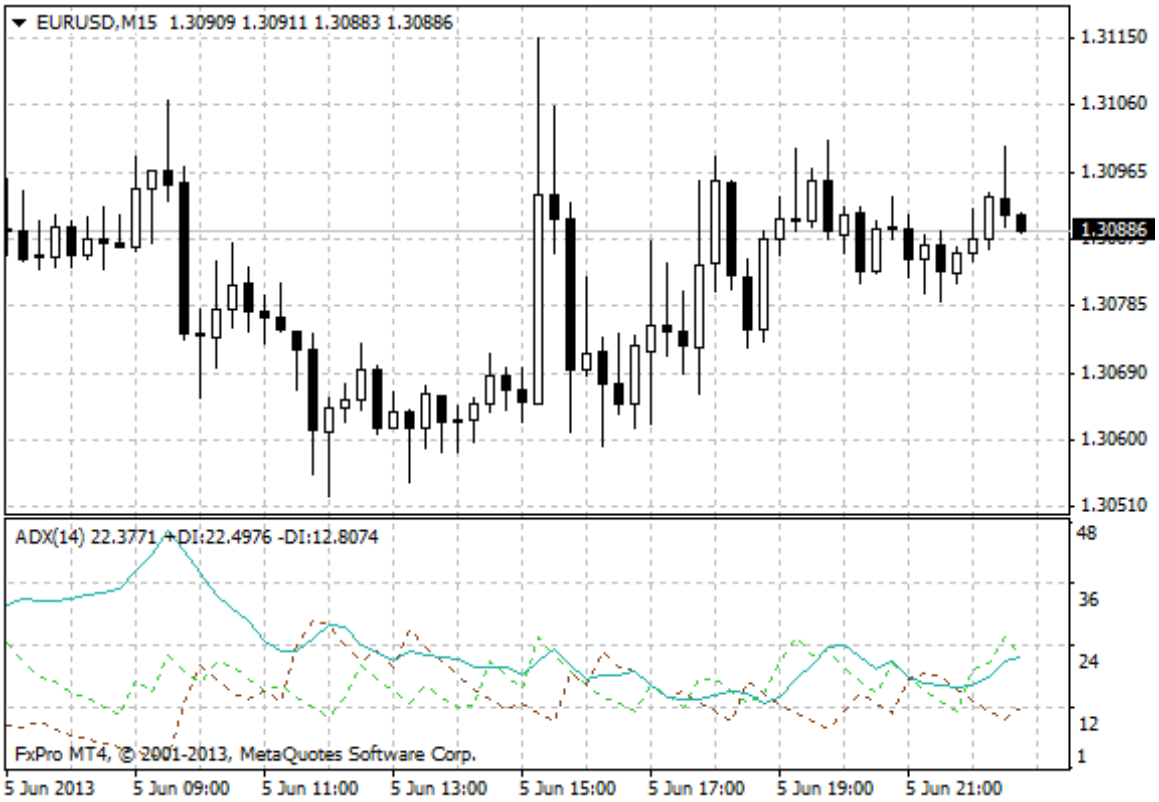


Figura 5 - Average Directional Movement Index

L'ADX nasce dell'unione degli indicatori, +DI e -DI. Una volta combinati questi due valori vengono poi smussati da una media mobile esponenziale EMA di periodo 14.

Il calcolo di +DI e -DI necessita di prezzo di Chiusura, Massimo e Minimo di ciascun periodo. La formulazione teorica di questo indicatore è la seguente:

$$ADX = \frac{1}{P}\sum_{i}^{P}\frac{PosDI - NegDI}{PosDI + NegDI}$$

Per l'algoritmo prevede intanto il calcolo di due termini definiti *Directional Movement* (*PosDM* e *NegDM*):

$$UpMove = Max_i - Max_{i-1}; \qquad DownMove = Min_i - Min_{i-1}$$

$$PosDM = \begin{cases} UpMove\ se\ UpMove > DownMove, UpMove \neq 0 \\ 0\ Altrimenti \end{cases}$$

$$NegDM = \begin{cases} DownMove\ se\ UpMove < DownMove, DownMove \neq 0 \\ 0\ Altrimenti \end{cases}$$

Si calcola quindi il Directional Index Positivo e Negativo

$$PosDI = \frac{EMA(PosDM) \cdot 100}{ATR}; \qquad NegDI = \frac{EMA(NegDM) \cdot 100}{ATR}$$

Infine Il valore dell'ADX vale

$$ADX_i = \frac{EMA(|PosDL - NegDL|)}{PosDL - NegDL} \cdot 100$$

L'EMA è calcolata sul numero di periodi selezionati (24 ore di default);

L'ATR è una media mobile esponenziale del true range con periodo di default di 14 giorni [8].

1.2.1.5 *Average True Range*

L'ampiezza di un cross è la differenza tra il massimo ed il minimo in un dato intervallo temporale. Questa informazione mostra quanto sia volatile il prezzo stesso. Più grande è l'ampiezza e più alta è la volatilità e viceversa.

Il True Range è stato sviluppato per migliorare la misurazione della volatilità, usando un sistema di tre equazioni anch'esso basato sui massimi e sui minimi.

Questo indicatore viene restituito nel seguente modo:

- **Max**: la differenza tra il massimo attuale e il prezzo di chiusura precedente;
- **Min**: la differenza tra il minimo attuale e il prezzo di chiusura precedente;
- **TR**: Max – Min.

l'ATR, infine, viene calcolato come una media mobile esponenziale (EMA) del True Range.

Il periodo standard è di 14, ma può essere modificato. I periodi più lunghi saranno più lenti ed indicheranno meno segnali di trading, al contrario i più corti porteranno il trader ad intensificare le operazioni. La maggior parte dei traders ritengono che la volatilità sia ciclica e, in base a questo, usano l'ATR come segnale di entrata, ovvero:

- livelli alti e crescenti della ATR segnalano il proseguimento della forza del trend;
- livelli bassi o decrescenti ne segnalano la diminuzione di intensità, indicando l'avvio di una fase di stabilizzazione prima di un possibile break-out.

Un esempio di tale indicatore è mostrato in Figura 6 [8].

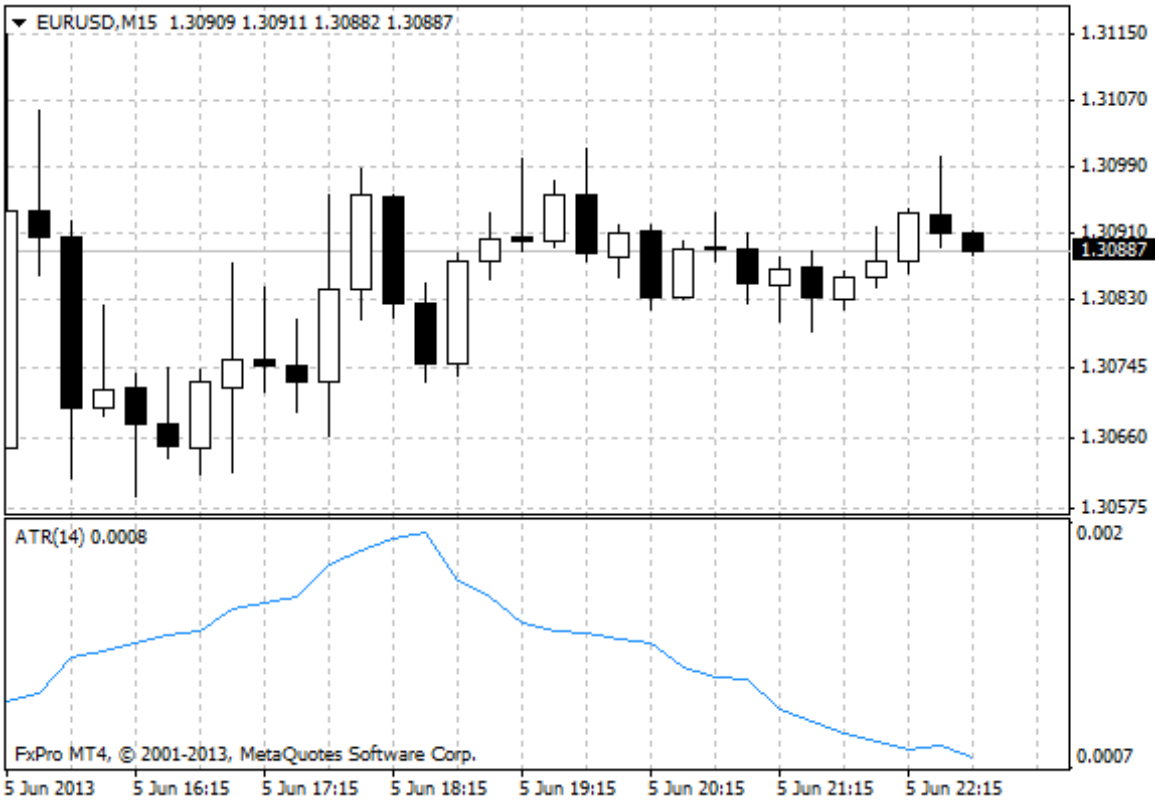


Figura 6 – Average True Range

1.2.1.6 *Awesome Oscillator (AO)*

L'indicatore Awesome Oscillator (AO), ideato da Bill Williams [5], è utilizzato per rappresentare l'attuale momentum del mercato, utilizzando una rappresentazione tramite istogramma.

Per tale oscillatore si utilizza la differenza tra due medie mobili, una a periodo corto, ovvero **5** candele, ed una di periodo lungo, ovvero **34** candele.

Il calcolo delle medie si basa non sugli ultimi close, ma utilizzando il prezzo medio della barra, ovvero

$$\left[\frac{(Max\ -Min)}{2}\right]$$

In Figura 7 è possibile vedere un esempio sul grafico di un cross di tale indicatore.

La formulazione teorica è la seguente:

$$MA' = SMA_i = \frac{1}{P}\sum_{j=i-P}^{i}\frac{(Max_j - Min_j)}{2} \qquad con\ P = 5$$

$$MA'' = SMA_i = \frac{1}{P}\sum_{j=i-P}^{i}\frac{(Max_j - Min_j)}{2} \qquad con\ P = 34$$

Da cui si ricava:

$$AO_i = MA'_i - MA''_i$$

Figura 7 - Awesome Oscillator

Il valore dell'indicatore AO si muove lungo lo zero, alternando intervalli di tempo positivi ad intervalli negativi. Nello specifico:

- Quando l'AO risulta essere maggiore di zero, si ha un segnale di acquisto
- Quando l'AO risulta essere minore di zero, si ha un segnale di vendita

L'interpretazione dei segnali di acquisto/vendita è la seguente, ed avviene esaminando tre barre consecutive:

Segnale d'acquisto:

- Prima barra di colore rosso;
- Seconda barra di colore rosso, ma di valore inferiore alla precedente;
- Terza barra di colore verde, ma sempre di valore maggiore della precedente;

In questo caso è consigliata l'apertura di un ordine d'acquisto sopra la prima candela in corrispondenza della terza barra verde dell'istogramma AO.

Segnale di vendita:

- Prima barra verde;
- Seconda barra verde, ma di valore inferiore alla precedente;
- Terza barra rossa, ma sempre di valore maggiore della precedente.

In questo caso è consigliata l'apertura di un ordine di vendita sotto la prima candela in corrispondenza della terza barra rossa dell'istogramma.

1.2.1.7 *Bollinger Bands*

Misurano la volatilità del mercato. Consiste di tre bande:

- Banda Centrale calcolata come SMA di periodo 20.
- Banda Inferiore calcolata come SMA di periodo 20 meno 2 volte la Deviazione Standard;
- Banda Superiore calcolata come SMA di periodo 20 più 2 volte la Deviazione Standard.

Quando il mercato tende ad aumentare la sua volatilità, le bande superiori ed inferiori si allontanano dalla media centrale, quando il mercato diventa meno volatile, le bande torneranno ad avvicinarsi [9].

La formulazione teorica delle Bollinger Bands è la seguente

Posto $P = Periodo$

$$StdDev = \frac{1}{P}\sqrt{\frac{1}{P}\sum_{i}^{P}(Close - SMA(Close, P))^2}$$

Un esempio di tale indicatore è mostrato in Figura 8.

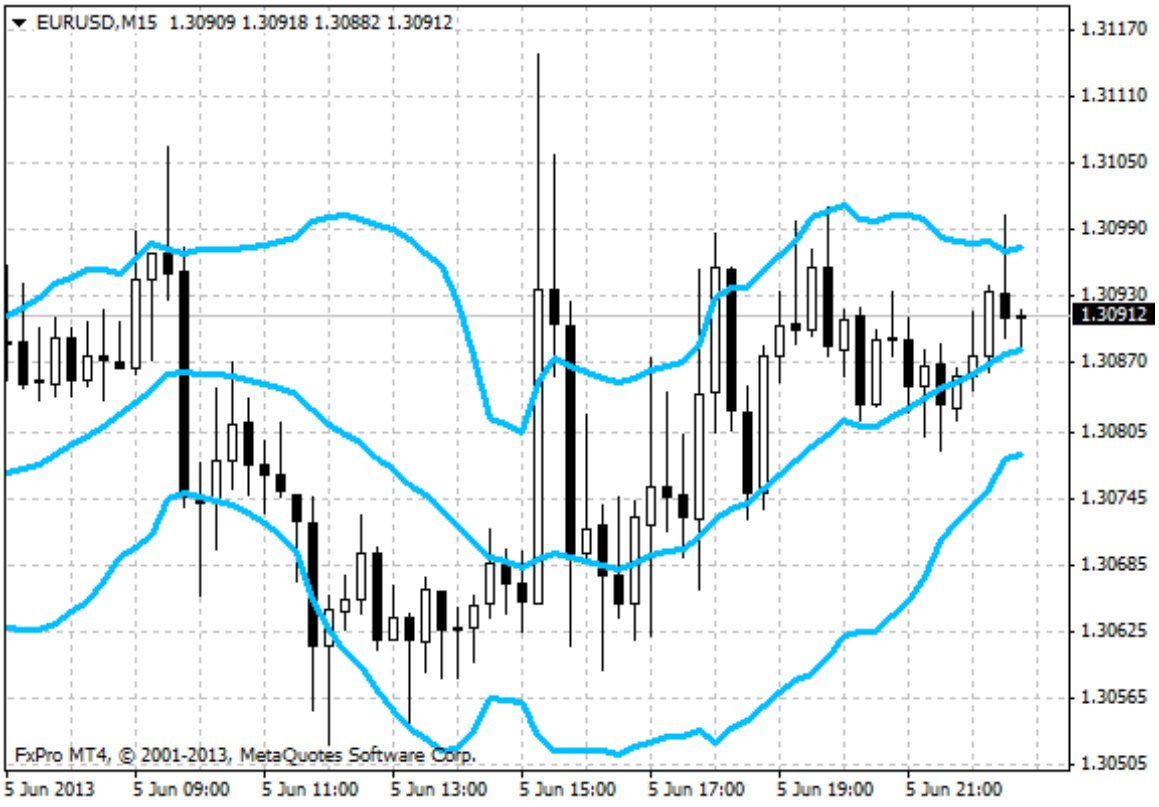


Figura 8 - Bollinger Bands

1.2.1.8 *MACD – Moving Average Convergence/Divergence*

Questo indicatore, ideato da Gerald Appel [10], e sostanzialmente realizzato tramite due medie mobili di periodo differente. Tali medie, in base alle variazioni dello strumento finanziario, si muovono con esso, allontanandosi (divergendo) quando il mercato è in una fase di trend ben definita e avvicinandosi (convergendo) quando il mercato rallenta, ovvero quando vi è la possibilità di una inversione. È calcolato come segue:

$$MACD_i = EMA'_i - EMA''_i$$

$$EMA' = Exponential\ Moving\ Average\ di\ periodo\ P = 12$$

$$EMA'' = Exponential\ Moving\ Average\ di\ periodo\ P = 26$$

Inoltre tale indicatore è fornito di una signal line così calcolata:

$$Signal = Simple\ Moving\ Average\ di\ periodo\ P = 9\ dei\ valori\ di\ MACD$$

Un esempio di tale indicatore è mostrato in Figura 9.

Si dimostra essere molto efficace in mercati con grosse oscillazioni. Esistono tre casi per utilizzare l'MACD:

- **Crossover**: La regola di base di trading MACD è quello di vendere quando il MACD scende al di sotto della sua signal line. Allo stesso modo, un segnale di acquisto si verifica quando il MACD sale al di sopra della sua signal line.
- **Ipercomprato / Ipervenduto**: Il MACD è anche utile come indicatore di ipercomprato / ipervenduto. Difatti la posizione dell'istogramma al di sopra o al di sotto dello zero indicano una fase di trend rialzista o ribassista. Quando la media mobile corta si allontana drasticamente dalla media mobile lunga (cioè, il MACD sale), è probabile che ci si trova appunto in una condizione di ipercomprato, viceversa in ipervenduto. Ovviamente è lecito aspettarsi presto un rientro a livelli più realistici.
- **Divergenze:** Una indicazione che la fine del trend corrente può essere vicino, si verifica quando il MACD, ovvero l'istogramma, diverge rispetto all'andamento dei prezzi.
    - Una *divergenza rialzista* si verifica quando il MACD sta compiendo nuovi massimi mentre i prezzi non raggiungono nuovi massimi.
    - Una *divergenza ribassista* si verifica quando il MACD sta compiendo nuovi minimi mentre i prezzi non raggiungono nuovi minimi.

Entrambe queste divergenze sono più significative quando si verificano in relazione a livelli di ipercomprato/ipervenduto ed indicano spesso la possibile inversione del trend.

Figura 9 - MACD

1.2.1.9 *Parabolic SAR*

Il SAR parabolico è usato per determinare la direzione del momentum di un cross e l'istante in cui vi è una maggiore probabilità di inversione del cross stesso [8].

L'indicatore si presenta come una serie di punti disposti sopra o sotto della linea del prezzo. La posizione del punto determina una indicazione precisa per il trader, (v. Figura 10) ovvero:

- Se un punto è posizionato sotto il livello di prezzo, allora esso indica un segnale di trend crescente, quindi presuppone che il trader confidi in un consolidamento dell'andamento del cross stesso;
- Se il punto è posizionato sopra il livello di prezzo, allora indica un segnale di debolezza, di conseguenza il trader dovrà aspettarsi una direzione negativa del trend, una inversione.

Il SAR è molto utile per indicare i punti di entrata. Infatti se la linea del prezzo attraversa il più recente punto di massimo, il SAR viene posizionato nel più recente punto di minimo. Avviene il viceversa in caso contrario.

Quando il prezzo di un cross sale, i punti posizionati sotto saliranno prima lentamente per poi prendere velocità ed avvicinarsi alla linea del prezzo.

Questa accelerazione permette di controllare lo sviluppo del trend, fino al consolidamento dello stesso. Il SAR infatti si muove sempre più velocemente (ovvero i punti si avvicinano sempre di più ai prezzi) mano a mano che un trend si sviluppa.

Figura 10 - Parabolic Sar

$$SAR_i = SAR_{i-1} + \alpha \cdot (EPrice_{i-1} - SAR_{i-1})$$

Con:

- SAR_{i-1} è valore dell'indicatore sulla candela precedente;
- α è fattore di accelerazione;
- $EPrice_{i-1}$ è il più alto (o più basso) prezzo analizzato nel precedente periodo.

1.2.1.10 *Stochastic Oscillator*

Tale oscillatore, ideato dal Dr. George Lane [11], si occupa di confrontare il prezzo di un determinato strumento finanziario rispetto ad una determinata fascia di prezzo entro cui dovrebbe trovarsi, in un determinato periodo.

L'oscillatore stocastico viene rappresentato tramite due curve. La curva principale è chiamata **%K**. La seconda curva, chiamata **%D**, è una media mobile dei valori della curva %K. Il valore di tale oscillatore è compreso tra 0 e 100.

Vi sono tre modi comuni di utilizzare questo oscillatore:

- Comprare quando il suo valore scende al di sotto di 20, e vendere quando supera il valore di 80;
- Acquistare quando la curva %K incrocia ed oltrepassa superiormente la curva %D e, viceversa, vendere;
- Considerando le divergenze.

La formulazione è la seguente:

$$Lowest_P = Il\ piu\ basso\ tra\ i\ prezzi\ nel\ periodo\ P$$

$$Highest_P = Il\ piu\ alto\ tra\ i\ prezzi\ nel\ periodo\ P$$

$$\%K_i = \frac{Close_i - Lowest_P}{Highest_P - Lowest_P} \times 100$$

$$\%D_i = \frac{1}{P} \sum_{j=i-P}^{i} \%K_j$$

La media applicata per determinare il valore dell'oscillatore stocastico può essere una tra le possibili medie disponibili. In Figura 11 è mostrato un esempio di tale indicatore tratto dalla piattaforma MetaTrader, in un grafico separato da quello dei prezzi. La curva dell'oscillatore stocastico è quella in azzurro, la rossa in stile tratteggiato, è la signal line.

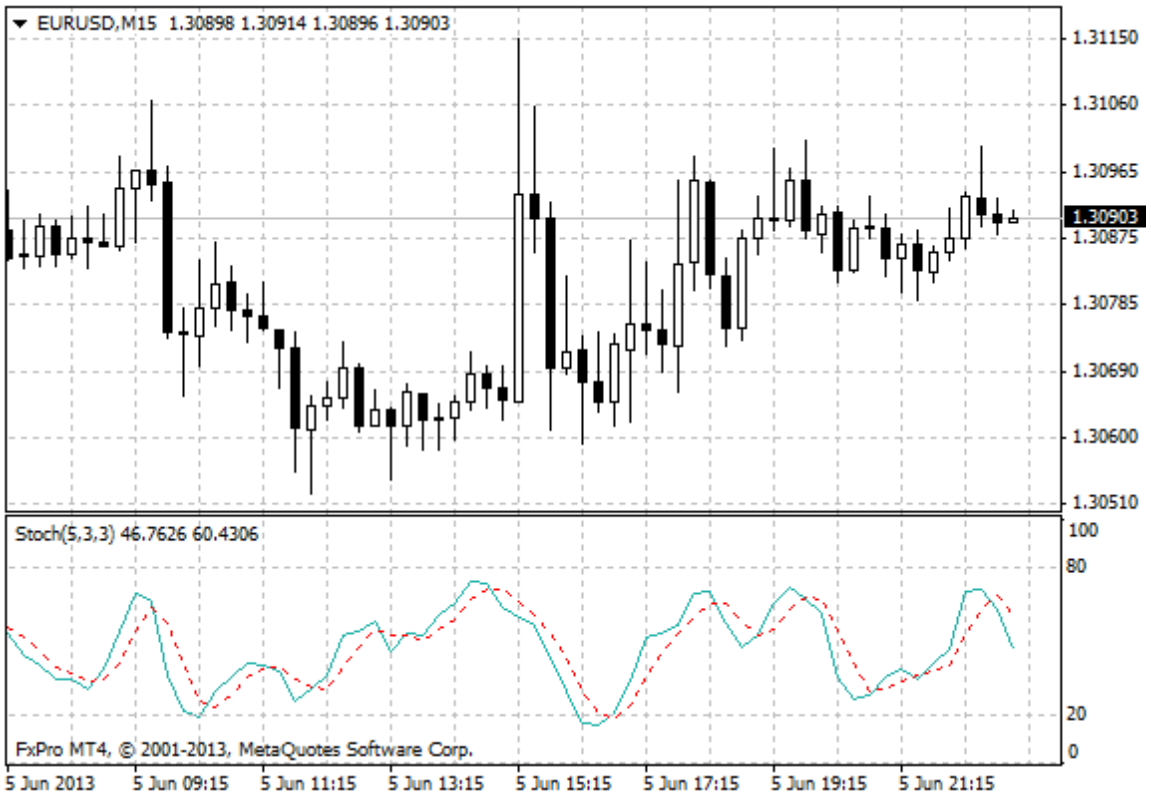


Figura 11 - Stochastic Oscillator

1.2.1.11 *Volumes Indicator*

Questo indicatore è una semplice rappresentazione dei volumi sotto forma di istogramma. I volumi indicano il numero di contratti scambiati nell'unità temporale predefinita, per cui a barre dell'istogramma di maggiore entità corrispondono maggiori volumi, e di conseguenza maggiore volatilità del mercato. Una barra verde indica che il volume riferito alla candela attuale è maggiore della precedente. Viceversa, una barra rossa indica che il volume per la candela attuale è minore paragonato alla precedente Figura 12.

Figura 12 - Volumes Indicator

2 Decision Support System (DSS) e Automated Decision System (ADS)

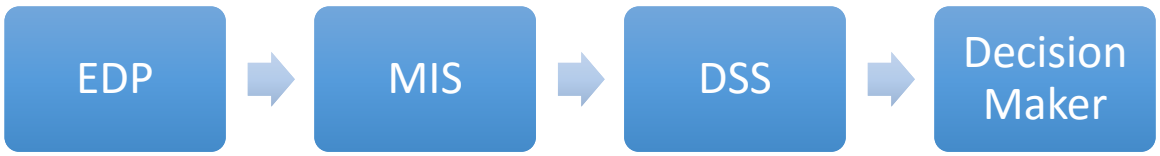
Un processo decisionale è una successione di attività elementari che hanno luogo nel momento in cui un individuo o un'organizzazione prende una decisione. Ogni attività produce dei risultati che saranno l'input per le attività successive. Naturalmente i processi decisionali dipendono dal soggetto preposto a prendere la decisione.

I processi decisionali sono guidati da tre paradigmi, il primo, *razionale*, mira a massimizzare il valore atteso della decisione in termini di introito, costi e rischi. Il secondo si riferisce ad un processo decisionale che ha lo scopo di isolare la prima alternativa efficiente in termini di costo, utilizzando semplici *euristiche* piuttosto che tecniche di ottimizzazione. Il terzo traduce il processo decisionale in una serie di *successivi confronti* per raggiungere la scelta di una alternativa. Tali modalità di scelta possono essere utilizzate anche in combinazione. Tutte sono supportate, però, da sistemi di memorizzazione: database, aggregazioni di dati, sistemi di rappresentazione, librerie, collegamenti, profili.

Un *DSS* (Decision Support System) può essere definito come un sistema informatico interattivo rivolto a migliorare i processi decisionali non completamente strutturati, consentendo quindi al decisore di effettuare precise scelte in contesti particolarmente complessi. Un DSS è, infatti, un sistema software che mette a disposizione dell'utente una serie di funzionalità:

- *Decision*: sta ad indicare l'attenzione rivolta ad attività decisionali e a problemi direzionali;
- *Support*: indica che le tecnologie informatiche sono di aiuto nel prendere le decisioni, ma non si sostituiscono al decisore, il quale rimane il vero protagonista;
- *System*: evidenzia che questi strumenti mirano all'integrazione tra utenti, macchine e metodologie di analisi.

I DSS differiscono dai tradizionali sistemi di trattamento delle informazioni perché richiedono la simbiosi fra utente, ossia il *decision maker*, e il sistema. Affinché tutto ciò sia possibile è necessario da un lato che il decision maker comprenda cosa il DSS può fare e dall'altro gli sviluppatori di sistema sappiano come integrare le tecnologie nella presa della decisione. Il DSS si pone nel processo decisionale come il sistema più vicino alla decisione e ingloba due sistemi, *EDP* e *MIS*.



L'EDP (*Electronic Data Processing*) è il sistema con cui si processano i dati per ottenere informazioni; il MIS (*Management Information Systems*) invece elabora le informazioni per fornire dei riferimenti, delle scelte possibili, delle opzioni nella decisione. Il DSS utilizza EDP e MIS per elaborare in modo interattivo una possibile decisione insieme al decision maker. La differenza rispetto all'approccio classico della ricerca operativa è che il sistema di supporto alla decisione non cerca di fornire una soluzione ottima e quindi una risposta definitiva. I DSS combinano l'uso di modelli o di tecniche analitiche con tradizionali funzioni di elaborazione dei dati, consentendone, tramite un'elevata interattività e semplicità, l'utilizzo a qualunque tipologia di utente.

Il processo decisionale, ancora oggi, può essere rappresentato con il modello proposto da Simon [12] negli anni '60. Tale modello suddivide in tre fasi principali il percorso da seguire per giungere alla decisione finale; inoltre da ognuna delle fasi è possibile tornare alle precedenti:

- *Intelligence*: è la fase in cui si raccolgono informazioni sia dall'ambiente esterno che interno per individuare e circoscrivere un problema da affrontare;
- *Design*: questa fase consiste nel comprendere il problema, generare soluzioni possibili ed analizzarle. È in questa fase che intervengono le capacità e l'esperienza del decisore, nonché la sua creatività, soprattutto nel generare le alternative;
- *Choice*: in questa fase si procede alla valutazione e quindi alla scelta delle alternative formulate nella fase precedente. Si definiscono a tal fine dei parametri e degli indicatori che permettano di fare sia confronti fra i piani d'azione che previsioni, sui quali saranno effettuate delle scelte.

Nel 1972 Simon e Newell, hanno esteso le fasi riguardanti il processo decisionale, schematizzandolo come segue [13]:

- Analisi e definizione del problema;
- Ricerca delle possibili soluzioni;
- Valutazione e scelta della soluzione tra le alternative individuate;
- Implementazione della decisione;
- Controllo e monitoraggio dei risultati ottenuti.

2.1 Tassonomie dei DSS

In questo paragrafo si descrivono brevemente alcune tra le principali tassonomie che nel corso degli anni sono state descritte al fine di classificare i DSS.

Donovan e Madnick nel 1977 [14], classificano i DSS in due categorie:

1. *Istituzionali*: quei DSS pensati per supportare decisioni ricorrenti;
2. *Ad Hoc*: atti a supportare decisioni con interrogazioni sui dati una tantum.

Alter, nel 1980, [15], dopo numerosi studi sui sistemi di supporto alle decisioni, concluse che i DSS potevano essere classificati in termini della tipologia di generiche operazioni svolte da tali sistemi. Queste operazioni generiche, estese lungo una singola dimensione, variano tra i due estremi data-oriented e model-oriented. Nello specifico:

1. *File drawer systems*: sistemi che forniscono accesso ai dati;
2. *Data analysis systems*: sistemi per la manipolazione automatica dei dati o per specifiche applicazioni o per tool e operatori di più alto livello;
3. *Analysis information systems*: sistemi per l'accesso a basi di dati e modelli orientati alle decisioni;
4. *Accounting and financial models*: per la stima delle conseguenze derivanti da possibili azioni;
5. *Representational models*: per la stima delle conseguenze derivanti da azioni, ottenute tramite modelli di simulazione;
6. *Optimization models*: forniscono linee guida generando una soluzione ottimale al problema, corredata da una serie di vincoli;

7. *Suggestion models*: eseguono il processing logico, al fine di condurre ad una specifica possibile decisione, per task sufficientemente strutturati.

Una ulteriore classificazione fu fatta da Hackathorn e Keen nel 1981 [16], i quali hanno suddiviso i DSS in base alle differenze dei potenziali utilizzatori del sistema:

- Personal DSS;
- Group DSS;
- Organizational DSS.

Una delle pietre miliari dei testi riguardanti i DSS è il libro di Sprague e Carlson nel 1982 [17], nel quale in prima istanza si descrive il framework che lo stesso Sprague ha decritto in [18]. Inoltre fornisce una chiara e pratica spiegazione di come una organizzazione potrebbe e dovrebbe strutturare il proprio DSS. Essi definiscono un DSS come

"*Una classe di sistema informativo che ricorre a sistemi di elaborazione delle transazioni e interagisce con le altre parti del sistema informativo a supporto delle attività decisionali dei manager e altri lavoratori della conoscenza nelle organizzazioni."*

Nel 1999, Haettenschwiler [19] differenzia i DSS nelle seguenti tre categorie:

- *Passive DSS*: ovvero un sistema che contribuisce al processo decisionale, ma non fornisce suggerimenti espliciti sulla decisione o una eventuale soluzione al problema;
- *Active DSS*: i quali, a differenza dei precedenti, sono in grado di fornire un effettivo suggerimento sul processo decisionale**;**
- *Cooperative DSS*: i quali permettono a decisore di intervenire in maniera attiva al processo decisionale, completando, modificando o raffinando la decisione suggerita dal DSS, prima di poter essere validata dal sistema complesso, in maniera iterativa, fino a giungere ad una soluzione consolidata.

A livello concettuale, Power nel 2002 [20] differenzia i DSS in cinque differenti tipologie, ovvero:

- *Communication-Driven DSS*, i quali forniscono supporto a decisori con l'obiettivo della risoluzione di un problema. Tali sistemi sfruttano le reti e le tecnologie classiche per permettere la comunicazione tra i decisori, quali ad esempio sistemi di video conferenza;
- *Data-Driven DSS*, sfruttano l'accesso e la manipolazione serie temporali interni ed esterni alle aziende. Le funzionalità di base sono fornite da tool di data retrivial. Per fornire maggiori funzionalità al sistema, talvolta si utilizzano tool specifici per l'accesso a database o data warehouse;
- *Document-Driven DSS*, i quali recuperano, gestiscono e manipolano informazioni non strutturate, provenienti da svariate tipologie di formati elettronici. Tali dati sono memorizzati all'interno di grandi database, e possono essere sotto forma di file di testo, immagini, file audio, cataloghi, ecc. Il livello più elementare di supporto alle decisioni utilizza un motore di ricerca per individuare, in base ricerche con specifiche parole chiave, i documenti di interesse;

- *Knowledge-Driven DSS*, i quali forniscono soluzioni di problem-solving specifiche, immagazzinate come fatti, regole e procedure. Sono sistemi ibridi uomo-macchina, e la competenza di tali sistemi riguarda la conoscenza su uno specifico dominio d'azione, la capacità di analizzare tale problema nel dominio e l'abilità a risolverlo;
- *Model-Driven DSS*, sfrutta l'accesso e la manipolazione di svariate tipologie di modelli, quali modelli statistici, finanziari, di ottimizzazione o di simulazione, per fornire un supporto al decisore. Tali modelli utilizzano parametri e dati di piccola entità forniti dal decisore per supportare lo stesso nell'analisi del processo decisionale. Tali tipologie di DSS non necessitano di una grande mole di dati.

Un modo comune per effettuare una classificazione dei DSS si fonda proprio sulle fasi del processo decisionale in cui essi possono intervenire. Ne deriva l'individuazione delle seguenti tre categorie di DSS:

1. *DSS data oriented*. Si basano sul trattamento dei dati contenuti nei database, con il fine di analizzare il problema, ricercarne la soluzione e controllare i risultati ottenuti a seguito delle decisioni prese. Assistono pertanto il processo decisionale, reperendo dati e correlandoli secondo criteri che consentono di ricavare le informazioni di interesse del decisore. Questi viene posto in grado di ricercare autonomamente e con la massima facilità operativa le informazioni di cui ha bisogno, affinando in corso d'opera il processo di ricerca in conseguenza delle maggiori conoscenze acquisite.

   I DSS data oriented possono, a loro volta, essere distinti in:

   - *DSS data retrieval*, attraverso i quali chi interroga il database accede ad una o più informazioni, fornite separatamente senza alcuna elaborazione di correlazione;
   - *DSS data analysis*, che producono le informazioni richieste attraverso la ricerca e la successiva correlazione di più informazioni prelevate da uno o più database.
2. *DSS model oriented*. Consentono di simulare le possibili soluzioni al problema, ricercando l'ottimizzazione della scelta da implementare. Tali DSS si prefiggono di riprodurre, in termini logico – matematico, su di un elaboratore elettronico, il modello di decisioni assunte in un determinato centro di responsabilità e, quindi, di effettuare una serie di analisi variabili, modificando sia la decisione che lo scenario in cui questa deve venire assunta.
3. *DSS esperti***.** Basati sull'intelligenza artificiale, sono utili sia nelle fasi di ricerca e scelta della soluzione che nella fase di controllo a posteriori delle decisioni adottate. I DSS basati sull'intelligenza artificiale tendono all'emulazione del comportamento di un decisore esperto. In particolare, essi prevedono di introdurre nella memoria dell'elaboratore, sia la struttura degli elementi che caratterizzano un certo fenomeno (i cosiddetti "fatti") sia le regole che rappresentano ciò che normalmente si intende per conoscenza di un certo fenomeno. Se a questa base di conoscenza si aggiunge una logica per la ricerca della soluzione più valida ad un problema, si ottiene quello che tecnicamente viene chiamato "sistema esperto".

2.2 Utilità di un DSS

Nell'ambito di un processo decisionale è possibile suddividere i tipi di decisione in:

- *Strutturate;*
- *Semi strutturate;*
- *Non strutturate.*

Una decisione è strutturata se si possono specificare per essa una serie di regole deterministiche e predefinite (ovvero un algoritmo) per poter giungere ad una soluzione; tali decisioni sono dette anche programmabili. Quindi è sufficiente porre in input all'algoritmo i dati in possesso del decisore, per ottenere l'esito della decisione.

All'opposto, una decisione non strutturata non ha una procedura decisionale prestabilita in quanto si riferiscono spesso ad avvenimenti nuovi, mai visti prima, oppure ad avvenimenti saltuari per cui definire una routine risulta poco conveniente. Per questo motivo tale decisione è detta anche non programmabile. Inoltre non esistono criteri prestabiliti per identificare la soluzione, e tutto ciò su cui il decisore può basarsi è l'esperienza.

Le decisioni semi strutturate presentano una struttura mista, cioè presentano contemporaneamente elementi programmabili e non programmabili. Per tali decisioni le regole sono parzialmente definite e necessitano di un certo grado di intuitività da parte del decisore per giungere alla soluzione.

Vediamo alcuni esempi:

- *Strutturate*: determinare l'interesse da applicare ad un prestito;
- *Non strutturate*: la decisione sull'assunzione di un dirigente dipende principalmente dalla percezione che le persone delegate all'assunzione hanno del soggetto in esame.
- *Semi strutturate*: la definizione del budget di una divisione dipende da indicatori economici interni ed esterni, ma richiede anche l'intervento del responsabile che consideri gli obiettivi della divisione.

I maggiori benefici da un DSS si ottengono per quei problemi in cui i parametri e le informazioni da considerare sono numerosi e difficili da controllare. È per questo che in genere si progettano DSS per supportare attività semi – strutturate o non strutturate.

Lo scopo di un DSS non è trovare una struttura o un algoritmo per poi automatizzarlo, ma dare un effettivo supporto a processi decisionali, poco o per niente strutturati, tramite un dialogo continuo con cui l'utente guida le operazioni del sistema.

2.3 Componenti di un DSS

In questo paragrafo andremo ad esaminare quali sono le componenti principali di un sistema di supporto alle decisioni.

2.3.1 Base di dati

La base dati contiene dati e informazioni che, direttamente o indirettamente, interessano l'utente. In genere egli è interessato solo ad alcuni tipi di dati, a certe opportune aggregazioni, non a tutti o almeno non ad ogni dettaglio. Un DSS deve quindi avere una base dati indipendente rispetto alle basi dati gestionali, inoltre tale base di dati è spesso integrata con informazioni esterne (ad esempio tassi d'interesse, quotazioni, ecc.). Va precisato che si tratta di una base dati *relazionale* con memorizzazione secondo viste diverse in base a ciò che è necessario analizzare. Il DBMS (*Data Base Management Software*) è il software che permette di definire schematicamente l'organizzazione dei dati, memorizzarli, modificarli e gestirli, permettendo una interrogazione semplice della base dati. Altri modelli di database che si

possono utilizzare sono quelli *gerarchici*, *a rete*, *a regole*. Il modello a rete sta diventando prevalente grazie anche alla capacità di supportare gli altri modelli e alla integrazione di reti di calcolatori. In questo caso la delocalizzazione dei dati riduce i costi di gestione con una minima perdita di efficienza dovuta ai tempi di accesso più lenti.

Da quanto appena detto è chiaro che la realizzazione di un DSS dipende sia dagli utenti che lo dovranno utilizzare sia dalle caratteristiche dei problemi che si intendono affrontare.

2.3.2 Base di modelli

L'altra risorsa informativa di notevole importanza, oltre ai dati, sono i modelli decisionali. Proprio partendo dall'idea che i modelli sono una fonte importantissima di informazione si è evoluta negli ultimi venti anni un'area della ricerca nell'ambito dei DSS: il Model Management.

Finora il termine "modello" è stato usato in maniera intuitiva, si deve, quindi, precisare cosa si intende con questo termine. Nella letteratura del Model Management non c'è una definizione univoca, ma, come per i DSS, ne esistono molte, o meglio esistono diverse concezioni di modello. Quella predominante definisce un modello come una procedura automatizzata che analizza dati in risposta ad un determinato problema. Una base di modelli contiene tutti i modelli, cioè tutte le procedure necessarie per risolvere i problemi dell'utente.

2.3.3 Software

Il tipo di software predominante per la realizzazione pratica di un DSS è quello costituito da pacchetti di routine. Le routine possono essere algoritmi di ottimizzazione, euristiche, sistemi esperti o quanto altro sia a disposizione.

Per favorire l'accesso trasparente ai dati è necessario un linguaggio di programmazione di alto livello. Infatti, tramite un tale linguaggio, selezionando parole chiavi, non viene richiamata la procedura, ma formulato un quesito tramite un linguaggio maggiormente orientato all'utente piuttosto che al sistema. È poi il sistema che interpreta il quesito ed eventualmente ponendo ulteriori quesiti, individua la routine adatta agli scopi. Sempre più, oggi, con lo sviluppo della intelligenza artificiale si cerca da un lato di sviluppare meta linguaggi naturali che rendano semplice la formulazione dei quesiti e dall'altro di sintetizzare nel sistema una "intelligenza". Sono state sviluppate nell'ultimo decennio delle regole comportamentali del sistema, ossia dei ragionamenti di tipo standard che gli consentano un dialogo adeguato al processo decisionale dell'utente.

Altri componenti che stanno diventando fondamentali sono l'interfaccia grafica e la multimedialità del sistema. Per la prima è chiaro che operare scelte su un supporto grafico che visualizza dati e diagrammi è molto più semplice che leggere tabulati di dati e scrivere comandi. Il DGMS (*Dialog Generation Management Software*) è il software che realizza l'interfaccia utente. Essa definisce, quindi, il tipo di interazione con esso. Inoltre determina le richieste che l'utente può fare, quali risposte può ottenere e in che modo, e lo guida nell'uso del DSS. Questa parte del sistema software è fondamentale per il successo di un DSS, poiché l'utente è fortemente interessato alle capacità di comunicazione del sistema e a come si fornisce l'interazione uomo – macchina oltre che alle sue capacità di elaborazione.

Dal punto di vista dell'architettura tecnologica si possono identificare 3 livelli di base.

Il primo livello è quello dei sistemi informativi "alimentanti". Sono sistemi informativi di tipo operativo, possono essere sistemi integrati di tipo *ERP* o *non* ERP (*Enterprise Resource Planning*), correnti o storici, sistemi dedicati al Customer Support o varie applicazioni delle tecnologie Web (eCommerce, Portali, eSupply Chain, ecc.).

Tra il primo e il secondo livello si trovano alcuni strumenti software specialistici dedicati alla mappatura, pulizia e trasferimento dei dati elementari nelle basi di dati fisiche del secondo livello.

Il secondo livello è quello delle basi dati direzionali, realizzate con approcci logici di *Datawarehousing* o di *Datamarting*. Queste basi dati direzionali disaccoppiano l'ambiente operativo e transazionale

dall'ambiente del controllo, delle analisi e delle decisioni manageriali soprastante, e integrano molteplici fonti di dati. A questo livello i dati direzionali possono essere archiviati sia con le tecnologie dei database relazionali, sia con le tecnologie dei database multidimensionali che consentono analisi specifiche di tipo OLAP (On-Line Analytical Processing), nonché di ottenere coerenti prestazioni tecniche nell'analisi e nella navigazione interattiva dei dati finalizzate al supporto decisionale. Queste basi dati alimentano a loro volta il livello tre dell'architettura.

Il terzo livello è quello dei sistemi di Business Intelligence, costituiti da svariate tipologie di strumenti software, pacchetti applicativi (Analytic Applications, Decision Support System), e software tools (Executive Information System toolkit, analisi multidimensionale, reporting) [21] [22].

2.4 Elementi di Teoria delle Decisioni

La teoria delle decisioni [23] studia il processo decisionale analizzando il comportamento degli attori da esso coinvolti ed esaminando, quindi, come questi prendono o dovrebbero prendere delle decisioni nel modo migliore possibile.
Il campo applicativo di tale teoria è assai vasto e comprende sia lo studio di situazioni assolutamente astratte che coinvolgono decisori razionali, sia problemi della vita reale fornendo un contributo pratico alla loro risoluzione. Quel che si fa è indagare su quali siano le conseguenze delle differenti decisioni, descrivendo con strumenti logico – matematici i comportamenti supposti razionali degli attori coinvolti nella decisione. In questo capitolo si darà enfasi maggiore a come le decisioni dovrebbero essere prese per massimizzare il proprio benessere, non di come esse siano prese.

Una prima distinzione da fare nella teoria delle decisioni è quella che contrappone le strategie *individuali* a quelle di gruppo. È importante sottolineare che per strategia individuale si intende una strategia attuata non solo da un singolo individuo, ma anche da aziende, enti pubblici, nazioni e così via, purché alla base vi sia il conseguimento di un unico obiettivo condiviso da tutti gli individui del gruppo. In questo tipo di strategie ci si concentra su come i vari individui agiscano per favorire i propri interessi, in modo egoistico senza dare rilevanza ad aspetti etici e morali. In contrapposizione, invece, le decisioni di gruppo sono quelle decisioni prese da individui che appartengono alla stessa organizzazione, ma manifestano opinioni diverse riguardo le scelte che dovrebbero essere attuate e riguardo agli obiettivi che si vorrebbero raggiungere. Ad ogni modo non importa quale decisione si stia prendendo in considerazione, sia essa individuale o di gruppo, ma quel che importa è studiare le azioni alternative perché ognuna produrrà delle conseguenze dipendenti dal contesto nel quale si sta prendendo la decisione. Le decisioni sono quindi costituite da azioni stati e conseguenze, con queste ultime che dipendono solitamente dall'azione e dallo stato in cui si trova il decisore al momento della decisione. L'analista (che potrebbe essere il decisore stesso) deve quindi individuare l'insieme delle azioni, degli stati e delle conseguenze per poter caratterizzare in modo adeguato il problema e poter prendere la decisione migliore possibile. Una volta individuati questi tre elementi fondamentali si può procedere alla specificazione del problema che solitamente avviene attraverso la costruzione di tavole o alberi di decisione.
La specificazione di un problema decisionale implica alcune questioni importanti che vanno tenute in considerazione durante il processo decisionale. La prima riguarda la descrizione appropriata degli stati di natura: ogni problema decisionale implica una serie di conseguenze e il decisore per ognuna di esse potrà dire se essa è migliore delle altre. Ma non è sempre facile per il decisore scegliere quale sia quella ideale e in questo ambito assume particolare rilievo il concetto di dominanza.

Nel dettaglio, in alcuni semplici casi, gli esiti conseguenti ad una decisione sono, comunque, non peggiori e, almeno in un caso, strettamente migliori a quelli di un'altra alternativa scelta, qualunque sia la modalità con cui si manifesta l'evento incerto. Si dice, in tal caso, che la prima decisione domina (in senso assoluto) l'altra e che quest'ultima può, pertanto, essere chiaramente eliminata dal contesto decisionale [24]. A questo punto, però, non vi è più dominanza tra le due situazioni ed il criterio decisionale si blocca. L'uso della dominanza, che può sembrare "banale" o "sciocco", una volta che il problema sia ben definito (cioè, di fatto già depurato di tutte le alternative "dominanti") è, invece, utilissimo in fase

di analisi del problema, in quanto consente di non perdere tempo ad esaminare innumerevoli alternative che, anche ad un primo esame superficiale, risultano chiaramente non interessanti, in quanto dominate.

Solo grazie a ciò ed all'accettazione di "semplificazioni" ed "approssimazioni", un problema può avere dimensioni gestibili (poche alternative).
Si sottolinea però che la dominanza non è un vero e proprio criterio decisionale (salvo il caso limite di un'alternativa che domini tutte le altre) ma, piuttosto, un criterio complementare utile per ridurre l'ambito del problema a dimensioni più trattabili.

Un'altra questione rilevante è quella relativa alla distinzione tra

- *Decisione giusta:* Si dice giusta una decisione che porta al raggiungimento di risultati (conseguenze) ottimale. Appare ovvio che in condizioni di completa conoscenza del futuro basterebbe semplicemente affidarsi alla decisione giusta che sicuramente sarà quella che porterà un vantaggio maggiore.
- *Decisione razionale*: Si definisce decisione razionale quella che il decisore prende, essendo a conoscenza di informazioni parziali, quindi risulta impossibilitato a prendere una decisione giusta, facendo una valutazione completa di tutte le informazioni parziali a disposizione.

Da quanto detto si evince che le situazioni decisionali vanno diversificate tra di loro ed è per questo che usualmente si utilizza distinguerle come:

- Decisioni in situazioni di certezza
- Decisioni in situazioni di rischio
- Decisioni in situazioni di incertezza.

Indichiamo con

$$A = \{ a_1, a_2, a_3, \dots , a_i \dots , a_m \}$$

l'insieme delle decisioni (azioni) possibili.

Inoltre indichiamo con

$$\theta = \{ \theta_1, \theta_2, \theta_3, \dots , \theta_j \dots , \theta_n \}$$

l'insieme dei possibili stati di natura e con

$$C = \{ c_{11}, c_{12}, c_{13}, \dots , c_{ij} \dots , c_{mn} \}$$

l'insieme delle conseguenze, dove le conseguenze sono funzione dell'azione a_i e dello stato θ_j

$$c_{ij} = f(a_i , \theta_j) \qquad per\ i = 1, 2, \dots , m; \quad j = 1, 2, \dots , n.$$

Si può rappresentare il processo decisionale (dove è stato ipotizzato un numero discreto di alternative ed un numero discreto di stati di natura) in modo appropriato facendo ricorso alla tavola delle decisioni (v. Tabella 1) o all'albero di decisione (v. Figura 13).

Tabella 1 - Tavola delle decisioni

Probabilità	$P(\theta_1)$	$P(\theta_2)$	…	$P(\theta_j)$	…	$P(\theta_n)$
Azioni						
a_1	c_{11}	c_{12}	…	c_{1j}	…	c_{1n}
a_2	c_{21}	c_{22}	…	c_{2j}	…	c_{2n}
.	.	.	…	.	…	.
.	.	.	…	.	…	.
a_i	c_{i1}	c_{i2}	…	c_{ij}	…	c_{in}
.	.	.	…	.	…	.
.	.	.	…	.	…	.
a_m	c_{m1}	c_{m2}	…	c_{mj}	…	c_{mn}

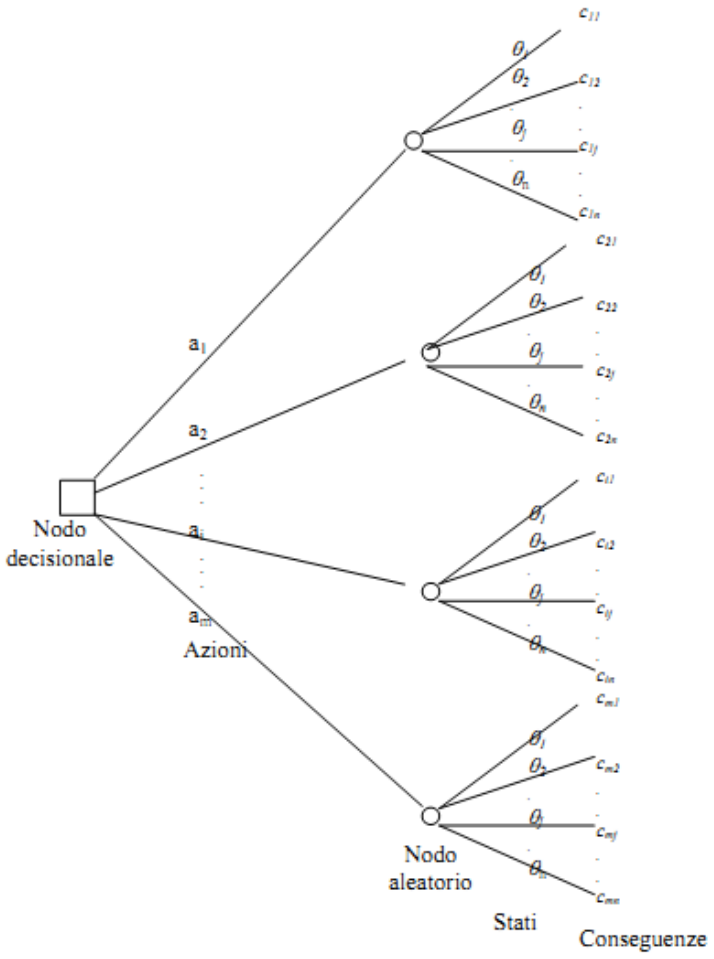


Figura 13 - Albero di decisione

Nella struttura ad albero è possibile distinguere i nodi che la compongono:

- *Nodo decisionale*: nodi di scelta delle possibili alternative da parte del decisore;
- *Nodo aleatorio*: si verifica uno stato di natura con relative probabilità di occorrenza;
- *Nodo terminale*: determinato dalla catena di decisione ed eventi verificatisi.

Se il decisore conoscesse lo stato di natura, ad esempio θ_j, il problema di scelta si ridurrebbe al confronto tra m conseguenze, e la scelta razionale equivarrebbe alla scelta giusta, sempre che siano note le conseguenze ed il decisore sia in grado di esprimere, in modo razionale, le sue preferenze riguardo alle conseguenze stesse.

Il comportamento razionale consente, in altre parole, l'individuazione dell'alternativa ottimale che comporta il conseguimento del massimo beneficio. Se lo stato di natura non è noto ma si dispone di una misura della probabilità dei vari stati di natura, si parla di decisioni in situazioni di rischio. Se non si dispone di alcuna informazione sulla probabilità dei vari stati di natura, si parla di decisioni in situazioni di estrema incertezza.

2.5 Applicazione dei Sistemi di Supporto alle Decisioni ai Mercati Finanziari

Come descritto nel capitolo 1.1, l'uso dell'analisi tecnica e dell'analisi fondamentale ricopre un ruolo di primo attore nell'estrazione delle informazioni per le attività di trading. Negli ultimi anni inoltre i sistemi informatici dedicati all'analisi dei mercati finanziari, hanno acquisito una importanza sempre maggiore, diventando strumento quasi indispensabile ai trader professionisti, nonché agli attori di maggiore impatto sul mercato, per acquisire suggerimenti scaturiti dall'analisi del mercato stesso, in relazione alle operazioni di trading da porre in essere. L'obiettivo finale nell'utilizzare un sistema decisionale automatizzato è quello di eliminare dalla fase di analisi le caratteristiche intrinseche del decisore, ovvero la componente umana, principalmente l'emotività, che possono influenzare di conseguenza il risultato.

Riassumendo, un Sistema di Supporto alle Decisioni per il trading finanziario non è altro che un sistema informatico atto a fornire ai trader delle indicazioni sul mercato in esame, utili alla definizione delle prossime attività di trading.

Un *Automatic Trading System* è invece un sistema informatico che, anziché suggerire una possibile strategia di trading o confermarne una sottoposta dall'utente, a seguito di una fase di analisi del mercato, automaticamente effettua il dimensionamento della posizione secondo regole prestabilite, e pone in essere l'operazione di trading. Il tutto avviene senza alcuna interazione del trader con il sistema, se non quello attuato durante la fase di inizializzazione e configurazione del sistema. Ovviamente il tutto prevede che lo strumento informatico sia stato dotato di modelli e algoritmi di analisi del mercato e gestione del portafoglio che lo rendano profittevole nel medio/lungo termine.

Che si parli di un Automatic Trading System o di un Decision Support System, la soluzione deve essere in grado di analizzare qualunque fase del mercato, ovvero deve saper riconoscere ed avere al suo interno modellati i comportamenti per le seguenti macro fasi di mercato:

- sistemi trend follower;
- sistemi contrarian
- sistemi per il break-out della volatilità

In tal senso deve essere basato su una forte base scientifica e deve avere delle regole operative che siano quanto più possibile oggettive. La scelta dei parametri caratterizzanti il modello e la definizione del loro campo di esistenza è un fattore fondamentale, non trascurabile e che va realizzato analizzando il comportamento generale della strategia nel tempo. Anche la modellazione della fase di money management

e il rischio/rendimento della strategia sono fondamentali per la definizione di un Automatic Trading System. Difatti la gestione del rischio definisce il dimensionamento delle posizioni in base al profilo di rischio che il trader vuole assumersi e impone una perdita massima atta ad evitare al trader perdite non sopportabili. Inoltre, utilizzando ad esempio modelli basati sul classico *Markowitz* o sul *Capital Asset Pricing Model,* è possibile stabilire un portafoglio di strumenti sul quale operare per mitigare eventuali ingressi errati a mercato. Infine il setup di un target di profitto, ad esempio il raggiungimento di supporti o resistenze statici importanti, permette di accontentarsi del guadagno previsto e la de-allocazione di risorse per ulteriori nuove operazioni.

2.5.1 Vantaggi e svantaggi nell'automazione delle strategie di trading

Come qualunque tecnologia, strumento, modello, esistono vantaggi e svantaggi che caratterizzano un sistema di trading automatizzato, in quanto pur modellando alla perfezione il sistema e considerando tutti i possibili eventi, è impossibile prevedere l'andamento del mercato stesso. In ogni caso, un sistema automatico di trading ha pregi e difetti.

Il sistema automatizzato di trading, appunto, decide in autonomia l'apertura e la chiusura delle posizioni, eliminando in questo modo ogni componente psicologica di pregiudizio o di emotività che invece è caratteristica del trader, soprattutto di quelli poco esperti. Ad esempio il sistema potrebbe decidere, dopo una attenta analisi del mercato basata sui modelli definiti in fase di programmazione dello stesso, di aprire una posizione al rialzo in una fase fortemente ribassista del mercato. Potrebbe decidere di chiudere una posizione attualmente vincente in quanto tutti i segnali inducono una imminente inversione del trend principale.

Un ulteriore vantaggio è la capacità che ha il sistema automatico di trading di chiudere le posizioni, principalmente quelle perdenti, nel caso in cui siano verificate le condizioni di massima perdita. In altre parole, l'accettazione della perdita da parte del sistema automatizzato non subisce l'influsso dell'emotività che invece pervade il trader nel momento in cui si trova a dover chiudere una posizione fortemente perdente.

Inoltre la capacità di stabilire una strategia di trading ordinata e definita su solide basi di analisi permette al trading system di ottenere risultati più regolari rispetto a quelli basati su decisioni maturate soggettivamente. Difatti così come risulta difficile poter accettare la perdita, allo stesso modo è difficile per il trader lasciar "*correre*" i profitti. Questo porta alla chiusura di operazioni in profitto in modo prematuro, decisione maturata magari per ovviare ad operazioni errate eseguite nell'immediato passato.

Lo svantaggio sostanziale di un trading system, se pur ben progettato, risiede nell'impossibilità di poter essere profittevole in ogni condizione di mercato. Certamente vi saranno periodi durante i quali il sistema non archivierà risultati positivi. Questo è dovuto principalmente all'impossibilità da parte del trader/analista/programmatore di modellare tutti i possibili andamenti del mercato.

Altro fattore di fondamentale importanza è la modalità di verifica dei risultati del trading system, ovvero la fase di ottimizzazione. Spesso tale fase, al fine di prevedere tutti i possibili casi del mercato, viene estesa per diversi mesi, potenzialmente anni, apportando modifiche e aggiungendo complessità alla strategia, a volte in maniera poco proficua. Inoltre i test per evidenziare il comportamento della strategia vengono eseguiti su serie storiche, che per poter essere utili allo scopo devono essere il più estese possibili. Se tale accorgimento non viene tenuto in considerazione, testando quindi ripetutamente il trading system su intervalli temporali troppo brevi o mantenendo invariato lo strumento finanziario, si rischia di specializzarne il comportamento solo a quella specifica tipologia o fase di mercato.

2.5.2 High Frequency Trading (HFT)

L'High Frequency Trading (HFT) è una tipologia di trading realizzata mediante l'utilizzo di trading system che usufruiscono di architetture hardware dalle elevate prestazioni in termini di tempo di risposta alle variazioni del mercato sottostante. Tali sistemi sono in grado di analizzare tali variazioni e giungere ad una decisione riguardante l'apertura di nuove posizioni in tempi dell'ordine delle decine di millisecondi. Solitamente l'utilizzo di strategie di trading ad alta frequenza comporta la generazione di un elevato numero di operazioni giornaliere, utilizzando connessioni a banda larga a bassissima latenza.

Ciò che differenzia il classico trading algoritmo dall'HFT è la rapidità con la quale le posizioni vengono inserite e rimosse dal mercato, al fine di ottenere guadagni anche dai più piccoli trend che si formano durante la normale price action dello strumento finanziario nella giornata di trading.

Principalmente vi sono tre tipologie di società di trading che operano mediante l'HFT:

- Grandi aziende, ovvero aziende che hanno a disposizione varie tipologie di strategie per l'HFT e grossi capitali. Solitamente queste aziende fungono da market maker e operano a mercato giornalmente;
- I broker dealing desk (ovvero quei broker che, anziché trasmettere l'ordine al mercato interbancario, si occupano di fare da controparte alle posizioni dei clienti), che hanno un settore separato da quello dei propri clienti dedicato all'HFT. Un esempio sono le banche maggiormente affermate;
- I fondi d'investimento.

L'HFT ha avuto un impatto profondo sui mercati finanziari, nello specifico, in positivo si ha [25]:

- *Aumento della liquidità*, ovvero l'elevato numero di transazioni comporta l'immissione di maggiore liquidità nei mercati;
- *Riduzione degli spread*, ovvero l'utilizzo di infrastrutture hardware/software ad elevate prestazioni permette un aggiornamento molto più frequente dei livelli di prezzo, con variazioni minime tra due aggiornamenti consecutivi. Questo ha portato ad una maggiore accuratezza nei livelli di prezzo, e di conseguenza anche una riduzione della differenza tra prezzo di acquisto e vendita, riducendo quindi lo spread;
- *Aumento dell'efficienza dei mercati*, ovvero l'aumento della frequenza delle contrattazioni comporta una migliore rappresentazione del mercato attraverso i prezzi, riducendo inoltre i costi di servizio.

La maggior parte degli articoli scientifici in letteratura non evidenziano gli effetti negativi relativi all'uso delle tecniche di HFT sulla qualità del mercato. Al contrario, come appena descritto, la maggioranza sostiene che HFT contribuisce a migliorare la costruzione dei livelli di prezzo e di conseguenza a migliorare qualità del mercato. In realtà vi sono alcuni fattori negativi che impattano sui mercati, di seguito elencati brevemente:

- *Impatto sugli attori istituzionali*, ovvero si pensa che spesso gli algoritmi per l'HFT ricercano pattern di trading ripetitivi, in maniera tale da avvantaggiarsi rispetto agli attori istituzionali. In tal modo riescono ad anticipare l'apertura di posizioni nella stessa direzione di tali trader, ai

quali rivendono subito dopo ad un prezzo leggermente superiore tale da garantire un profitto, a discapito proprio degli stessi attori istituzionali che ne pagano le conseguenze;

- *Aumento della volatilità dei mercati*, a causa della rapidità con la quale le operazioni vengono aperte e chiuse (da pochi secondi fino ad un massimo di pochi minuti), generando così volatilità a breve termine. Dato che al giorno d'oggi la maggior parte delle operazioni di trading e quindi della liquidità di mercato dipende da questa tipologia di algoritmi, si intuisce come l'aumento della volatilità dovuta all'HFT impatta sulla volatilità generale dei mercati. Inoltre vi è un dibattito etico basato sulla pratica degli algoritmi di HFT di optare sull'apertura di operazioni che istantaneamente vengono chiuse, utilizzate solamente per aumentare la volatilità dei mercati e fungere da trigger per le operazioni pendenti di altri attori sul mercato;
- *Impatto sui piccoli investitori*, in quanto solitamente i sistemi di HFT sfruttano tecnologie di *colocation* [26], ovvero sale disponibili solo a determinati operatori ed attrezzate con dispositivi ad alte prestazioni adatte all'HFT, o *Raw Data Feeds*, come servizi di distribuzione dei livelli storici di prezzo. Queste tecnologie non sono a disposizione ad esempio degli investitori retail o le piccole aziende, i quali non sono in grado di attuare le proprie strategie di trading.

Nel mondo esistono svariati centri dedicati allo sviluppo di algoritmi per il financial computing e tecnologie per l'HFT. Ad esempio uno dei centri maggiormente conosciuti al mondo è sito in Inghilterra è il *Centre for Doctoral Training in Financial Computing & Analytics*, presso il quale vengono trattati argomenti di Finanza Computazionale, Ingegneria Finanziaria e Finanza IT. I principali ambiti di ricerca riguardano Il Machine Learning, il Trading Algoritmico, la Teoria dei Portafogli, le Applicazioni dell'intelligenza artificiale al mondo della finanza, ecc.

Questo centro vanta partner accademici quali l'University College of London (UCL), la London School of Economics (LSE) e L'Imperial College of London.

Maggiormente noti e di grande rilevanza sono i partner aziendali, tra i quali ad esempio troviamo Barclays Capital, City Bank, Credit Suisse, HSBC, Royal Bank of Scotland, Thomson Reuters ecc.

2.6 Automated Decision System

Per poter definire una soluzione software come un sistema di supporto alle decisioni, è necessario che questo possieda tutte le componenti che gli permettano di rappresentare il modello di un DSS, secondo quanto indicato nel capitolo 2.

L'infrastruttura software discussa nei capitoli successivi rientra nella categoria degli Automated Decision System (ADS). Tali algoritmi differiscono dai Sistemi di Supporto alle Decisioni in quanto non si occupano soltanto di definire l'insieme delle possibili scelte da fornire poi al decisore (in base alle condizioni del sistema e all'analisi di dati provenienti da sensori esterni, ovvero i livelli di prezzo relativi allo strumento finanziario in esame), bensì sono anche in grado di valutare quale sia la migliore possibile strategia, la quale viene automaticamente posta in essere.

Quindi non sarà il decisore a scegliere autonomamente la strategia più adatta, bensì il software decisionale, il quale avrà a disposizione una serie di procedure decisionali preventivamente tradotte. Questo permette di evitare la selezione di strategie operative che possano essere influenzate dall'emotività o dal fattore psicologico del decisore.

Tali applicativi sono realizzati per limitare quindi il coinvolgimento del decisore al minimo indispensabile per il funzionamento dell'infrastruttura, ovvero in taluni casi eccezionali per cui l'intervento del decisore è essenziale per ovviare ad eventuali mancanze del sistema. Se le regole decisionali sono ben definite e di alta qualità i dati provenienti dai sensori di acquisizione, allora sarà semplice rendere completamente automatizzato il sistema.

Inoltre il vantaggio ottenuto dalla rapidità di trasformazione delle decisioni in azioni, permette l'uso di tali sistemi in situazioni in cui è fondamentale la rapidità di esecuzione, quali ad esempio il trading finanziario [27].

Nei successivi capitoli sarà ampiamente sviscerata una infrastruttura software atta a fungere da sistema automatizzato di trading, oltre che da supporto alle decisioni per lo stesso.

2.7 Modelli e approcci presenti in letteratura

Nell'ambito dell'analisi dei mercati finanziari, e nello specifico quello delle valute, vi sono vari studi riguardanti approcci o modelli atti ad analizzare gli andamenti degli strumenti finanziari, orientati ad individuare eventuali variazioni significative nella direzione dello strumento. Nello specifico, ci si riferisce all'individuare comportamenti dello strumento finanziario che possano indicare una imminente variazione della volatilità, sfruttabile da una eventuale strategia di trading.

Uno dei primi approcci, da cui poi è scaturita tutta una serie di varianti al modello iniziale, è il modello ARCH (AutoRegressive Conditional Heteroskedasticity), introdotto da Engle [28]. Tale modello è legato alle intuizioni di Mandelbrot [29] e Fama [30], riguardo la tipologia di dati in esame, ovvero le serie storiche finanziarie. Secondo i due studiosi, tali serie non risultano essere caratterizzate da distribuzioni normocurtiche, bensì da distribuzioni leptocurtiche, ovvero assegnano una maggiore probabilità ad eventi distanti dalla media della distribuzione, in raffronto alle probabilità che verrebbero loro assegnate nel caso di una distribuzione normale. Inoltre è spesso evidente il fenomeno di clusterizzazione della volatilità, ovvero a causa dell'inerzia o stabilità dei valori osservati, ogni valore è influenzato da quello precedente e determina in maniera rilevante il successivo. In tale modello media e varianza sono dipendenti dalle informazioni sulle serie storiche sino all'istante precedente.

Engle quindi basa il suo modello sul fatto che la volatilità mostra dinamiche auto regressive nel tempo, ovvero la varianza condizionale è in relazione con i valori da essa assunti nel passato. Una generalizzazione del modello appena descritto è il GARCH (Bollerslev) [31]. A partire da questo approccio sono stati sviluppati numerosi modelli derivati, ad esempio il modello IGARCH, FIGARCH [32] [33], ecc.

In contrapposizione ai modelli basati su GARCH vi sono i modelli a cambiamento di regime, detti Regime Switching, e più specificatamente il modello basato su catene di Markov. In tale modello vengono considerate le informazioni sui valori passati del processo che descrive i regimi, permettendo di spiegare meglio le dinamiche delle serie storiche. Lo studio dei modelli di tipo Regime Switching ha avuto storicamente inizio con Quandt [34], il quale, per primo, ha introdotto tale sistema in un modello regressivo. Successivamente Golfred e Quandt [35] hanno esteso il modello utilizzando le catena di Markov per definire lo switching tra un regime ed un altro.

Nel 1997, Calvet et al. [33] hanno proposto un modello previsionale della volatilità MSM (Markov Switching Multifractal), in cui si introduce il concetto di multifrattalità, riferito alla capacità di adattarsi alle variazioni del fattore di scala e la possibilità di integrare serie storiche apparentemente inusuali.

Le reti neurali artificiali (ANN) [36] [37] rappresentano un altro modello largamente utilizzato per l'analisi dei mercati finanziari. Ispirate dalla struttura del cervello e dalla rete neurologica biologica, gli approcci tramite reti neurali rappresentano una soluzione a problemi di decision making, forecasting e, più in generale, a problemi presenti nel mondo reale, come in contesti industriali [38]. Nell'ultimo decennio le ANN hanno trovato larga applicazione nei mercati finanziari [39], soprattutto negli approcci orientati alla simulazione dei mercati, alla predizione del comportamento degli investitori, risk assessment, portfolio management, forecasting, ecc [40] [41]. In [42] troviamo un esempio di applicazione di una rete neurale feed forward a singolo livello atta a predire i movimenti di strumenti finanziari. I risultati dei test hanno dimostrato come l'addestramento della rete su di uno specifico arco temporale causa l'overfitting del sistema.

In [43] è stato realizzato un modello basato su reti neurali al fine di effettuare una previsione sul valore del rapporto di uno strumento finanziario su un intervallo temporale pari ad una settimana, confrontando le performance del modello con altri modelli predittivi basati su random walk e auto-regressione lineare. In tutti i test effettuati, il modello ha mostrato una superiore efficacia del modello basato su reti neurali.

Altri approcci largamente studiati ed analizzati sono basati sugli Expert Systems (ES). Un sistema esperto è un sistema con una sufficiente base di conoscenza, in grado di imitare l'esperto durante la fase di problem solving. Nel dettaglio, è un sistema informatico in grado di sfruttare le informazioni presenti all'interno della propria knowledge base, tramite l'uso di regole di inferenza, per risolvere problemi

onerosi per l'esperto. Il vantaggio nell'uso di un sistema esperto, rispetto ai modelli precedenti risiede nella possibilità di non essere vincolato a rigidi modelli matematici, il che evita la necessità di dover riscrivere il modello per poterne modificare il comportamento, e di espandere la knowledge base al fine di rendere più efficiente il sistema [44] [45]. Anche questi modelli trovano applicazione nella predizione dei mercati finanziari, ad esempio in [46] è stato realizzato un ES atto a predire il trend di strumenti finanziari, con l'obiettivo di massimizzare i profitti e minimizzare le perdite. Il modello utilizza regole di inferenza basate sul confronto tra dati storici e attuali degli strumenti finanziari, accoppiati all'uso di un indice di analisi tecnica, per derivare decisioni riguardo le operazioni di trading da porre in essere nella giornata.

Infine, ma non ultimi per importanza, vi sono gli approcci basati su algoritmi genetici. Un Algoritmo Genetico (GA) è un algoritmo basato sulla teoria dell'evoluzione naturale. Tramite un pool iniziale di possibili soluzioni al problema (generate in modo random), si applica una metrica di valutazione della bontà della soluzione, definita funzione di fitting. Scelta la prima soluzione, questa viene successivamente ricombinata con le altre, introducendo un certo numero di elementi di disordine che variano il risultato della funzione di fitting, ottenendo una nuova possibile soluzione al problema. Il processo è iterato fino alla convergenza verso una soluzione ottima. Tali algoritmi appartengono al campo dell'intelligenza artificiale, e trovano svariate applicazioni, anche nell'ambito della finanza computazionale. Ad esempio, in [47] si trova un approccio ibrido tra una rete neurale e un algoritmo genetico per il trading. Più nel dettaglio, l'input è rappresentato dai valori di un set di indici di analisi tecnica, mentre i pesi del livello nascosto della rete neurale, utilizzato per la fase predittiva del modello, vengono ottimizzati dall'algoritmo genetico. In [48] si propone un approccio misto per la predizione del trend di stock market, basato sulla fusione tra un algoritmo genetico e l'indice di concordanza.

3 Infrastruttura della piattaforma di trading e ambiente di sviluppo

In questo capitolo verranno descritti gli ambienti di sviluppo e le tecnologie di comunicazione utilizzati nel lavoro di tesi. Nello specifico, nel paragrafo 3.1 si illustra la piattaforma di trading e l'ambiente di sviluppo utilizzato per la creazione dello strumento di gestione dell'operatività, mentre nel paragrafo 3.2 si descriverà l'ambiente di sviluppo LabVIEW by National Instruments utilizzato per lo sviluppo degli algoritmi di trading. Infine nel paragrafo 3.3 e successivi si descrive la soluzione software modellata per permettere l'interfacciamento tra due entità distinte, ovvero gli applicativi di trading sviluppati in LabVIEW e lo strumento di gestione dell'operatività, realizzata tramite la piattaforma MetaTrader4.

3.1 MetaQuotes Software Corp. e piattaforma MetaTrader4

La *MetaQuotes Software Corp.* [49] fu fondata nel 2000 in concomitanza del lancio della piattaforma *FX Charts*, un strumento per il trading online caratterizzato da ricche funzionalità software, una elevata velocità di operatività e la caratteristica di fornire ai trader i grafici relativi all'andamento dello strumento finanziario. Nel 2001 il software venne aggiornato alla nuova versione, denominata *MetaQuotes*, la quale permetteva all'utente di creare le proprie strategie di trading personalizzate, tramite l'uso del *MetaQuotes Language (MQL)*. Nel 2002 venne rilasciata la nuova versione del software, denominata *MetaTrader,* ed in concomitanza con tale release, venne resa disponibile una nuova versione del linguaggio di programmazione, l'*MQL II*. Molte software house iniziarono a sviluppare le proprie piattaforme di trading usufruendo del nuovo linguaggio MetaQuotes Language II.

Nel 2005 fu rilasciato un aggiornamento importante della piattaforma client, denominata *MetaTrader 4,* per la quale era disponibile il nuovo linguaggio di programmazione, *MetaQuotes Language 4 (MQL 4)*, un linguaggio di programmazione C-like, insieme all'MQL4-IDE (*Trading strategy development environment*) [50]. Viste le potenzialità del software e del linguaggio, l'azienda non rese disponibile il codice sorgente, rendendo così la propria piattaforma la più utilizzata per le operazioni di trading online.

Nel corso del 2010 MetaQuotes ha rilasciato una ulteriore nuova versione, *MetaTrader 5*, con annesso linguaggio di programmazione *MQL 5*, supportando la programmazione orientata agli oggetti.

Attualmente la piattaforma di trading, giunta alla versione 690, risulta essere in continuo aggiornamento. Molti broker forniscono per i propri clienti una versione personalizzata di tale piattaforma, che resta comunque fedele alla versione fornita dalla MetaQuotes, tramite la quale effettuare il login al proprio conto di trading ed iniziare subito ad operare sui mercati finanziari.

Ulteriore punto a favore della piattaforma di trading MetaTrader 4 fa riferimento alla disponibilità di un avanzato ambiente di test per le strategie sviluppate attraverso l'ambiente di sviluppo *MetaEditor*, ovvero lo *Strategy Tester*. Tale ambiente permette di effettuare backtest su serie storiche della propria strategia, con la possibilità di simulare in maniera dettagliata l'andamento del mercato. L'ambiente di simulazione difatti permette di testare le strategie utilizzando algoritmi specifici per interpolare nel più preciso modo possibile le serie storiche, al fine di ottenere un modello il più fedele possibile all'andamento reale del prezzo nell'intervallo temporale scelto per il test.

3.1.1 Linguaggio MQL4, Indicatori, Script ed Expert Advisor

MetaQuotes Language 4 è il linguaggio built-in per la programmazione di strategie di trading. Permette di creare i propri Expert Advisor per l'automatizzazione delle operazioni di trading. Inoltre si può usare il linguaggio per la creazione di propri Indicatori, Script, e Librerie. *MetaEditor 4* è invece l'ambiente di sviluppo per i propri sorgenti *MQL4*. L'ambiente di sviluppo fornisce indentazione, text-highlight delle parole chiave del linguaggio. Ha un dizionario (MetaQuotes Language Dictionary) per le parole chiave e una guida sintetica descrittiva riguardo le funzioni a disposizione.

Tramite l'ambiente di sviluppo è possibile implementare le proprie strategie di trading automatico, oppure i propri indicatori. Principalmente tramite il linguaggio di programmazione è possibile sviluppare le seguenti tipologie di algoritmi:

- **Expert Advisor (EA)**: È un *Mechanical Trading System* (MTS), ovvero un programma in grado di automatizzare la negoziazione. È composto principalmente da tre metodi:
    - *OnInit( ):* eseguita nel momento in cui l'EA viene agganciato ad un grafico dei prezzi. È utilizzata principalmente per l'inizializzazione eventuale delle variabili e delle strutture dati necessarie per il funzionamento della strategia di trading automatico;
    - *OnTick( ):* eseguita ogni volta in cui la piattaforma riceve dal broker un aggiornamento del livello di prezzo per il determinato strumento finanziario. All'interno di tale funzione solitamente trova collocazione il cuore della strategia di trading, con tutte le funzionalità previste dallo sviluppatore per l'apertura e la chiusura delle posizioni;
    - *OnDeinit( )*: eseguita nel momento in cui l'EA viene scollegato dal grafico dei prezzi, utilizzata per de allocare le risorse occupate dalla strategia e per eventuale esecuzione di operazioni di analisi dei risultati;
- **Custom Indicator**: È una funzionalità che permette principalmente di sviluppare i propri indicatori tecnici, le cui operazioni fanno riferimento ai dati presenti sui grafici dei prezzi cui vengono collegati. Tale programma consente inoltre di utilizzare le funzioni degli indicatori tecnici già esistenti in piattaforma per l'analisi dei prezzi. A differenza degli EA, gli Indicatori personalizzati non hanno la facoltà di automatizzare le operazioni di trading;
- **Script**: Questa categoria di programmi è pensata per eseguire operazioni in singola esecuzione; A differenza degli Expert Advisor, gli script possono essere eseguiti solo una volta (quando vengono agganciati ad un grafico dei prezzi), e ogni script si rimuove automaticamente quando l'ultima riga del suo codice viene eseguito.

Infine è possibile realizzare Librerie e file di Inclusione, al fine di separare dai propri EA, Custom Indicator e Script quelle funzionalità che vengono comunemente utilizzare dallo sviluppatore. In tal modo è possibile trasformare tale set di funzionalità in delle librerie, da includere all'interno dei propri algoritmi.

3.2 LabVIEW by National Instruments

LabVIEW by National Instruments è un linguaggio di programmazione ad oggetti grafici ed implementa il G-Language, il quale permette di velocizzare lo sviluppo di soluzioni software anche di elevata complessità, fornendo allo sviluppatore un set vastissimo di oggetti per la programmazione, atti a svolgere delle operazioni che altrimenti andrebbero, nei più comuni linguaggi testuali, sviluppati da zero. L'ambiente di sviluppo è suddiviso in due aree, un *Block Diagram* nel quale prende vita la struttura del software, ovvero vengono posizionati ed interconnessi i vari oggetti grafici al fine di eseguire le operazioni che lo sviluppatore ha previsto. Inoltre è disponibile un ulteriore pannello, denominato *Front Panel*, sul quale posizionare i controlli e gli indicatori che lo sviluppatore vuole rendere disponibili all'utente finale per controllare input e output dell'applicazione. Tale front panel andrà a comporre l'interfaccia grafica del software. LabVIEW Permette inoltre l'interfacciamento con dispositivi hardware e la possibilità di richiamare librerie esterne sviluppate in altri linguaggi di programmazione, funzionalità che ha permesso l'utilizzo di tale linguaggio di programmazione per lo sviluppo degli applicativi descritti nei capitoli 4, 6, 7.

3.3 Tecnologia di interfacciamento piattaforma - applicativi

In questo capitolo si descrive la metodologia sviluppata con lo scopo di permettere la comunicazione e l'interazione tra la piattaforma di trading MetaTrader4 e gli algoritmi sviluppati, che verranno descritti nei successivi capitoli.

Tale modellazione e sviluppo si è resa necessaria in quanto vi è l'impossibilità di avere accesso diretto ai dati ricevuti sulla nostra piattaforma di trading, ovvero ci si riferisce agli aggiornamenti dei livelli di prezzo degli strumenti finanziari inviati dal broker presso cui si è sottoscritto l'account.

In prima istanza l'idea era stata quella di realizzare una comunicazione uno-ad-uno tra l'applicativo di trading automatico e la piattaforma stessa, utilizzando come struttura di interscambio delle informazioni dei file ASCII, sui quali scrivere i dati relativi all'apertura e chiusura di posizioni e le informazioni di contorno necessarie al funzionamento della strategia.

Ovviamente, l'applicativo per il trading automatico ed un Expert Advisor ad-hoc si occupavano di eseguire le operazioni di lettura/scrittura sui suddetti file, con le conseguenze che ciò comporta, ovvero accesso simultaneo ai file con possibilità di inconsistenza delle informazioni in essi memorizzate. Inoltre la latenza dovuta alla scrittura dei dati su disco rendeva impossibile la realizzazione di applicativi per l'High Frequency Trading.

Viste le difficoltà appena descritte, si è deciso di seguire un altro approccio, certamente a livello realizzativo più complesso, ma di certo maggiormente conveniente in termini di performance e usabilità. Tale approccio ha visto la realizzazione di una *Dynamic-Link Library* (*DLL*), ovvero di una libreria a collegamento dinamico.

I vantaggi dati dall'utilizzo di una DLL sono molteplici:

- I dati sono scritti in memoria, per cui la velocità di scrittura/lettura è notevolmente maggiore rispetto a quella su file ASCII;
- La singola libreria, caricata in memoria all'esecuzione dell'Expert Advisor cui si riferisce, permette l'accesso ai dati a più programmi contemporaneamente, permettendo di risparmiare risorse di sistema;
- L'aggiunta di ulteriori funzionalità al software aggiornando la libreria ad una nuova versione aumenta la manutenibilità del software stesso.

Di seguito verrà descritta l'implementazione della DLL, con un esempio di utilizzo in LabView, e la struttura dell'Expert Advisor, necessario per l'interfacciamento tra la libreria e la piattaforma di trading.

3.3.1 Dynamic-Link Library MT4-LV

La DLL, realizzata in C++ tramite l'uso di Microsoft Visual Studio, è un insieme di funzioni Set/Get, atte alla lettura/scrittura delle informazioni in memoria. La libreria è composta principalmente da due sorgenti e da un file delle definizioni:

1. Dllmain.cpp;
2. MetaLabView.cpp;
3. MetaLabView.def.

3.3.2 Dllmain.cpp

Il sorgente *dllmain.cpp* contiene la funzione entry-point della libreria. Al momento del caricamento della libreria da parte di un applicativo, viene richiamato il metodo contenuto all'interno di questo sorgente,

denominato appunto *dllmain*, il quale ha lo scopo di effettuare il link dinamico della libreria all'applicativo. Tale funzione viene richiamata anche quando si termina l'esecuzione dell'eseguibile, al fine di eseguire l'un-link della libreria stessa.

3.3.3 MetaLabView.cpp

Il sorgente MetaLabView è il codice sorgente relativo alle funzioni contenute nella DLL. Il sorgente è suddiviso in tre macro aree:

1. Direttive per la creazione del segmento dati condiviso e inizializzazione delle variabili;
2. Signature di tutte le funzioni Set/Get contenute all'interno della DLL;
3. Corpo delle funzioni.

Per la prima macro area, sono previste due modalità di condivisione delle informazioni all'interno della DLL, ovvero tramite file mappati in memoria oppure tramite segmenti di dati condivisi. Si è scelto di utilizzare la seconda opzione, ovvero l'uso di un segmento dati condiviso. Tale struttura si realizza utilizzando da direttiva

```
#pragma data_seg(".shared")
```

Tale direttiva definisce una nuova area dati condivisa all'interno della DLL denominata "*.myseg*", e tutto ciò che verrà definito all'interno della direttiva "*data_seg*", sarà reso condiviso e quindi accessibile dalle funzioni all'interno della libreria. Infine È quindi necessario specificare gli attributi di condivisione corretti per questa nuova sezione di dati con l'opzione del linker

```
#pragma comment(linker, "/section:.shared,RWS")
```

Gli attributi impostati per il segmento dati "shared" sono

- **R**: *Read*, lettura;
- **W**: *Write*, scrittura;
- **S**: *Shared*, condivisa.

Ad esempio

```
#pragma data_seg(".shared")
       char name[15] = "";
       int  index=0;
#pragma data_seg()
```

È necessario tenere conto di alcune restrizioni o considerazioni relative all'uso delle sezioni shared, ovvero

- Tutte le variabili in un segmento di dati con nome devono essere inizializzate staticamente. Nell'esempio precedente la variabile intera *index* è inizializzata su 0, mentre la stringa *name*, di dimensione 15 caratteri, è inizializzata con una stringa vuota.
- Tutte le variabili condivise vengono inserite nella DLL compilata nel segmento di dati specificato. Variabili molto grandi, come ad esempio matrici di grandi dimensioni possono generare DLL molto grandi.

Per la seconda macro area, ovvero quella relativa alla signature delle funzioni contenute nella DLL, è necessaria in quanto permette di indicare al compilatore di aggiungere la direttiva di esportazione a quella funzione cui la signature si riferisce. Ad esempio:

```
__declspec(dllexport) void __stdcall WriteInMemory(double value);
```

La terza ed ultima macro area contiene la stesura del corpo di tutte le funzioni contenute all'interno della DLL. Tali funzioni ovviamente agiscono sulle variabili definite all'interno del segmento *shared*.

3.3.4 MetaLabView.def

Il file MetaLabView.def è un file contiene una o più istruzioni di modulo, necessari a descrivere alcuni attributi della DLL. Nello specifico sono necessari due attributi:

1. *LIBRARY*: Questa istruzione identifica il file *.def* come appartenente alla libreria, il cui nome segue l'istruzione stessa
2. *EXPORTS*: elenca i nomi delle funzioni esportate dalla DLL.

Di seguito, su due colonne, il contenuto del file *.def* descrittivo delle funzioni contenute nella DLL.

```
// Bid price e timestamp
WriteInMemory
ReadFromMemory
ReadTSFromMemory
// Ask price e timestamp
WriteAskInMemory
ReadAskFromMemory
// Numero di cifre decimali
SetDigit
GetDigit
// Margine libero
SetFreeMargin
GetFreeMargin
// Account number
SetAccNumber
GetAccNumber
// Leva del conto
SetLeverage
GetLeverage
// Nome dello strumento
SetSymbolName
GetSymbolName
// Media dei prezzi su 100 Giorni
SetMeanDay
GetMeanDay
// Std Dev dei prezzi su 100 giorni
SetSigmaMeanDay
GetSigmaMeanDay
// Massimo (Max-min) su 100 giorni
SetMaxMmDay
GetMaxMmDay
// Stop Loss
SetStopLevel
GetStopLevel
// Media degli ultimi X tick
SetMeanTick
GetMeanTick
// Pip compiuti nella giornata
SetCurrentPIPDay
GetCurrentPIPDay
```

```
// Strategia di Exist 3
SetExist3
GetExist3
// Strategia di Exist 2
SetExist2
GetExist2
// Info sull'ultimo ordine aperto
SetOrderInfo
GetOrderInfo
// Open-Close ultima candela chiusa
SetLastOC
GetLastOC
// Open dell'ultima candela
SetLastOpen
GetLastOpen
// Close dell'ultima candela
SetLastClose
GetLastClose
// High dell'ultima candela
SetLastHigh
GetLastHigh
// Low dell'ultima candela
SetLastLow
GetLastLow
// BollingerBands SupBand
SetLastBB_Sup
GetLastBB_Sup
// BollingerBands InfBand
SetLastBB_Inf
GetLastBB_Inf
//Open-Close dell'evento di volatilità
SetOC_Event
GetOC_Event
// Timestamp di apertura dell'ordine
SetOpenTS
GetOpenTS
```

3.4 Utilizzo di librerie esterne in LabView

Di seguito vedremo un breve esempio di come è possibile integrale l'uso delle funzioni implementate all'interno della DLL tramite LabView.

LabView, tramite l'utilizzo dell'oggetto "*Call Library Function Node*" permette di linkare una libreria, statica o dinamica che sia, semplicemente mappando ingressi ed uscite del nodo esattamente nella stessa sequenza in cui sono presenti i parametri nella signature della funzione, specificandone il tipo.

Figura 14 - Utilizzo del Call Library Function Node

La Figura 14 mostra un esempio di utilizzo del *Call Library Function Node*, usato in questo caso per la lettura di dati dalla DLL, ovvero richiamando una funzione di tipo Get.

Nello specifico, la Figura 15 mostra il pannello di configurazione del *Call Library Function Node*, nel quale si specificano i parametri:

- Percorso alla libreria assoluto, o semplicemente il nome della stessa se questa si trova all'interno della working directory dell'applicativo;
- La Funzione da richiamare;
- La tipologia di chiamata alla funzione contenuta nella libreria, la quale deve corrispondere alla specifica convenzione definita in fase di sviluppo della libreria, e data come direttiva di esportazione;

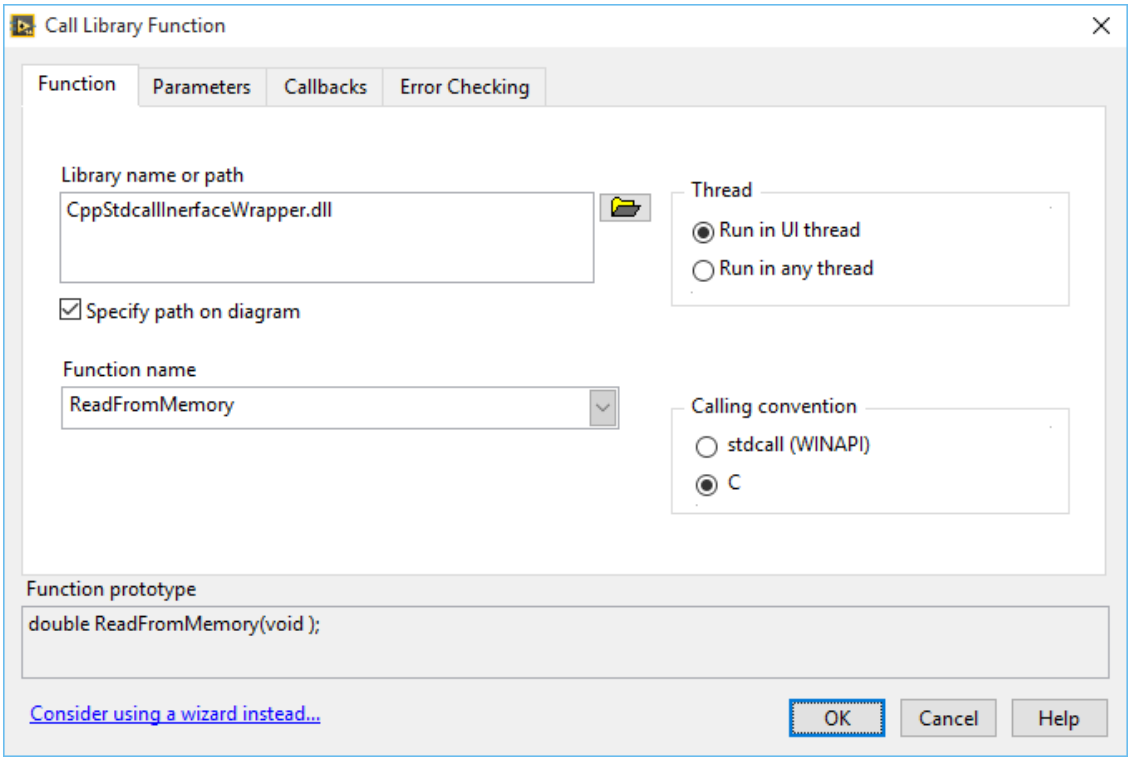


Figura 15 - Call Library Function Node – Scheda Function

La Figura 16 invece mostra la scheda della finestra di configurazione relativa ai parametri di input/output richiesti dalla funzione di libreria che si sta configurando:

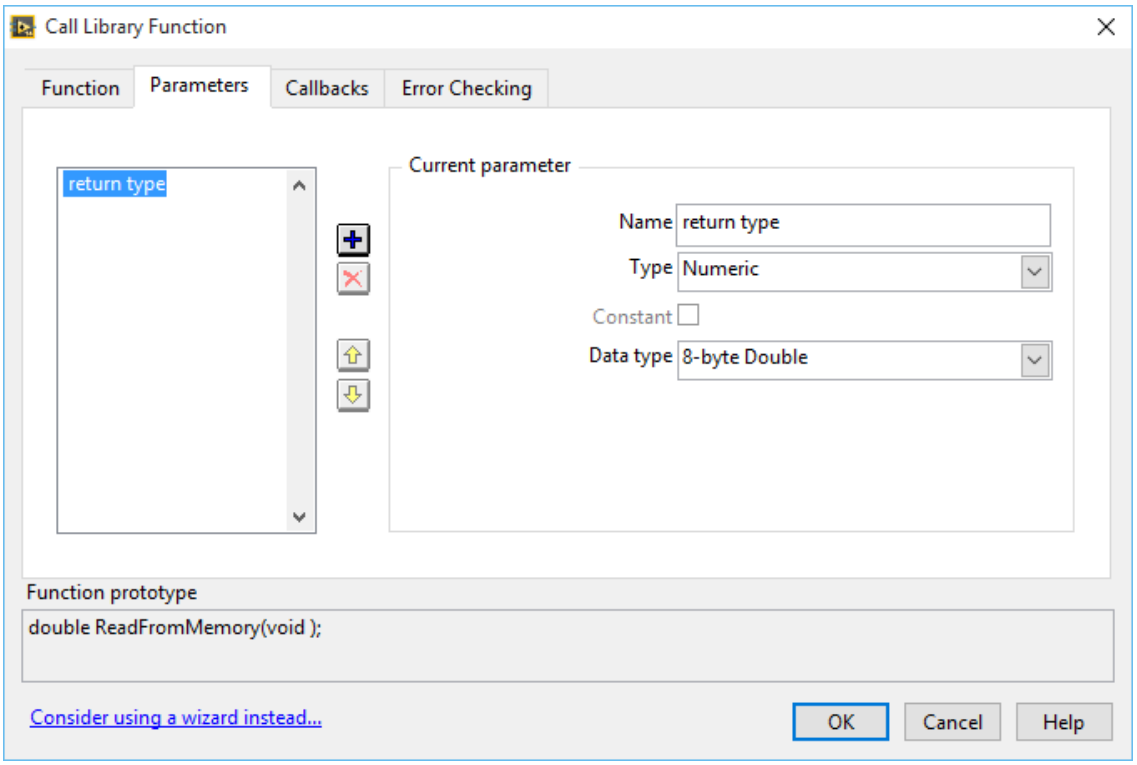


Figura 16 - Call Library Function Node – Scheda Parameters

Nello specifico, per la funzione selezionata, la quale non prevede parametri di input, ma soltanto un tipo di ritorno, si specifica che tale tipo è un *Double* ad *8 byte*, di tipo *Numeric*.

Qualora vi fossero stati dei parametri di input alla funzione, sarebbe stato necessario aggiungerli sotto al parametro relativo al tipo di ritorno, nell'ordine richiesto, specificandone tipo e dimensione.

Infine nella sezione denominata "*Function prototype*" è possibile vedere la costruzione del prototipo della funzione, man mano che si aggiungono parametri di input, così da avere conferma che il prototipo qui costruito rispecchi quello definito all'interno della DLL.

3.5 Expert Advisor MT4-LV

In questo paragrafo si descrivono i punti salienti riguardanti la realizzazione dell'Expert Advisor necessario alla comunicazione tra la piattaforma di trading MetaTrader4 e una applicazione esterna (nel caso di questo lavoro di tesi realizzate in LabView), tramite l'utilizzo della libreria descritta in precedenza.

L'Expert Advisor, realizzato in linguaggio MQL4, ha una doppia funzionalità: la prima è quella di scrivere all'interno della porzione di memoria condivisa resa disponibile dalla DLL, le informazioni relative allo strumento finanziario su cui è caricato, rendendole fruibili all'esterno, ovvero alle strategie di trading in esecuzione su quello strumento finanziario che condividono l'uso della DLL. La seconda funzionalità è quella di gestire l'operatività predisposta dalla strategia di trading.

La chiamata alla DLL e il link delle funzioni all'interno dell'Expert Advisor è immediato, prevede infatti l'utilizzo della direttiva #import comprensivo di prototipo delle funzioni che si vogliono integrare all'interno dell'Expert Advisor:

```
#import "MT4-LV.dll"

void WriteInMemory(double value, char &ts[]);
void SetDigit(int d);
```

Durante la fase di caricamento dell'Expert Advisor sullo strumento finanziario (fase di inizializzazione dell'EA), vengono scritte in memoria condivisa le informazioni relative allo strumento necessarie per l'operatività della strategia di trading. Tali informazioni sono:

- Identificativo dello strumento finanziario;
- Timeframe di operatività;
- Numero di cifre significative dello strumento (informazione necessaria per effettuare le operazioni in PIP);
- ID del conto di trading;
- Leva finanziaria;
- Margine Libero del conto;
- Stop Level, ovvero percentuale del margine libero al di sotto della quale il broker inizia a chiudere le posizioni più perdenti;
- Valore medio del movimento dello strumento, calcolato sugli ultimi 100 giorni di trading, in PIPs;
- Deviazione Standard del suddetto arco temporale;
- Escursione massima effettuata dallo strumento negli ultimi 100 giorni di trading.

Successivamente alla fase di inizializzazione, ad ogni aggiornamento ricevuto dal Broker (ovvero ogni qual volta si riceve un nuovo livello di prezzo) l'EA aggiorna le seguenti informazioni in memoria:

- Prezzo di vendita (Bid);
- Prezzo di acquisto (Ask);
- Timestamp (in millisecondi), relativo all'aggiornamento in questione;
- Margine Libero aggiornato;
- PIPs compiuti dallo strumento nella giornata odierna

Inoltre si rendono disponibili i dati relativi all'ultima candela chiusa. Tali informazioni si riferiscono a candele appartenenti a due differenti timeframe, ovvero è l'ultima candela chiusa del timeframe di esecuzione dell'Expert Advisor e l'ultima candela chiusa di timeframe 1 minuto.

Infine, in relazione ai timeframe precedentemente indicati, si rendono disponibili dei buffer unidimensionali di 20 elementi, di cui l'elemento in posizione *n-esima* è il più recente, contenenti:

- Bande di Bollinger;
- Media Mobile Esponenziale 21 Periodi;
- Media Mobile Esponenziale 63 Periodi;
- Media Mobile Esponenziale 100 Periodi.

Le ulteriori funzionalità messe a disposizione dall'Expert Advisor riguardano le singole strategie operative di trading, ovvero sono legate alle necessità degli applicativi esterni. Tali funzionalità si riferiscono all'apertura delle posizioni e all'analisi di alcune informazioni dello strumento utili alle strategie di uscita.

4 Servizi di Supporto

In questo capitolo si introdurranno dei servizi accessori realizzati a supporto degli applicativi per il trading automatico descritti nei capitoli successivi. Nello specifico verrà sviscerato un sistema software per l'estrazione ed elaborazione di file di dump e un sistema di alert per le news economiche.

4.1 Dump Reader

In questo capitolo si descrive lo sviluppo di uno strumento di estrazione dati, realizzato per potere interpretare il contenuto di specifici file di dump, ottenuti dalla piattaforma di trading MetaTrader4. La realizzazione di tale software nasce dalla necessità di poter elaborare i dati esistenti sulla piattaforma, ad esempio per effettuare analisi statistiche, oppure per testare le funzionalità implementate secondo una particolare strategia di trading, evitando di dover mantenere la strategia di trading in comunicazione con la piattaforma, permettendo dunque una elaborazione offline dei dati. Inoltre questa modalità di utilizzo dei dati riduce il tempo di elaborazione, al fine di poter testare le strategie di trading su dati appartenenti ad un esteso arco temporale, in tempi ridotti.

I file di dump sono creati mediante un Expert Advisor il quale, trascorsa una unità temporale dipendente dal timeframe cui fa riferimento, scrive un set di informazioni su un file di testo (nello specifico un file .csv), ovvero i livelli di prezzo della candela formatasi nell'intervallo temporale appena trascorso e i valori dei principali indicatori utilizzati nel trading.

Nel dettaglio, per ogni record del file di dump, si memorizzano le informazioni seguenti:

- Timestamp;
- Open;
- High;
- Low;
- Close;
- Volume;
- Banda di Bollinger Superiore e Banda di Bollinger Inferiore;
- Media mobile Esponenziale di periodo 7, 21, 63, 100, 270, 540, 810;
- Oscillatore Stocastico e sua Signal Line;
- ADX, composto da ADX Signal, Positive Directional Index e Negative Directional Index;
- Accumulation Distribution;
- Awesome Oscillator;
- Parabolic SAR.

4.1.1 Suddivisione in Aree Funzionali

Il software è stato realizzando cercando di suddividere al meglio il codice in aree funzionali, al fine di renderne la lettura e l'interpretazione da parte di terzi il più semplice e veloce possibile, oltre che meglio manutenibile. La Figura 17 illustra uno schema concettuale di come tale suddivisione è stata realizzata. Il file .csv utilizza come separatore di campo la virgola. Tale struttura dati permette di ottenere un vettore colonna per ognuno dei campi che sono stati specificati in precedenza.

Esaminiamo quindi le varie aree indicate in Figura 17:

Area 1: Area dedicata alla selezione del file di dump da cui estrapolare le informazioni e conversione in matrice di stringhe mediante l'uso del separatore;

Area 2: Area dedicata alla conversione del contenuto dei singoli vettori colonna da stringa al tipo di dato specifico, ad esempio oggetti in formato data/ora per i Timestamp e double per i valori in virgola mobile;

Area 3: Suddivisione in cluster dei dati convertiti.

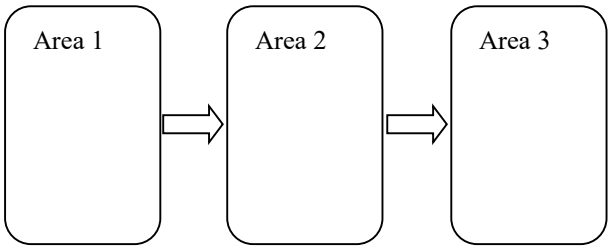


Figura 17 - Suddivisione in aree funzionali.

4.1.2 VI di uso comune o di supporto

Alcuni *Virtual Instrument* sviluppati, da adesso indicati per brevità *VI*, svolgono funzionalità generiche, non specifiche per un singolo obiettivo. Al fine di evitare ripetizioni, si descrivono in questa sezione tali vi, così da poterli menzionare senza descrizioni ripetute nelle pagine successive.

4.1.2.1 *GetColumn.vi*

Questo VI permette di estrapolare una singola colonna da una matrice. Riceve in input la matrice contenente i dati e l'indice della colonna che si vuole estrarre. Restituisce la colonna selezionata, se esiste. La Figura 18 mostra l'input/output del VI, mentre la Figura 19 il block diagram.

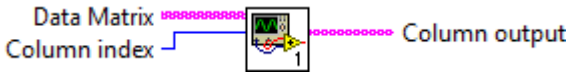


Figura 18 - Input/Output del VI GetColumn.

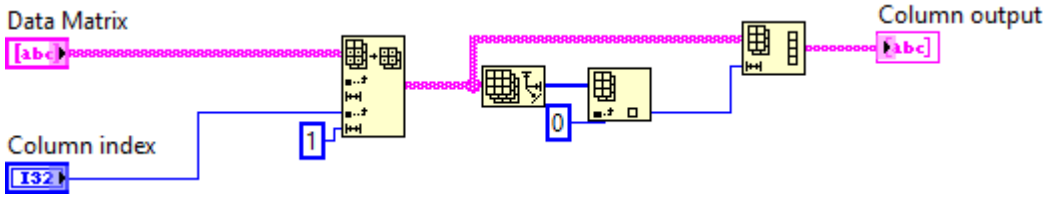


Figura 19 - Block Diagram del VI GetColumn.

4.1.2.2 *Convert_String_To_Timestamp.vi*

Questo VI permette di trasformare un Timestamp sotto forma di stringa, in un oggetto Timestamp di LabVIEW. La stringa di input deve essere nel formato seguente, per essere riconosciuta e convertita dal vi:

AAAA.MM.GG hh:mm:ss
2013.01.24 12:09:00

La Figura 20 mostra l'input/output del VI, mentre la Figura 21 il block diagram.

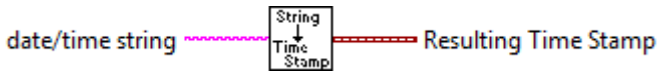


Figura 20 - Input/Output del VI Convert_String_To_Timestamp.

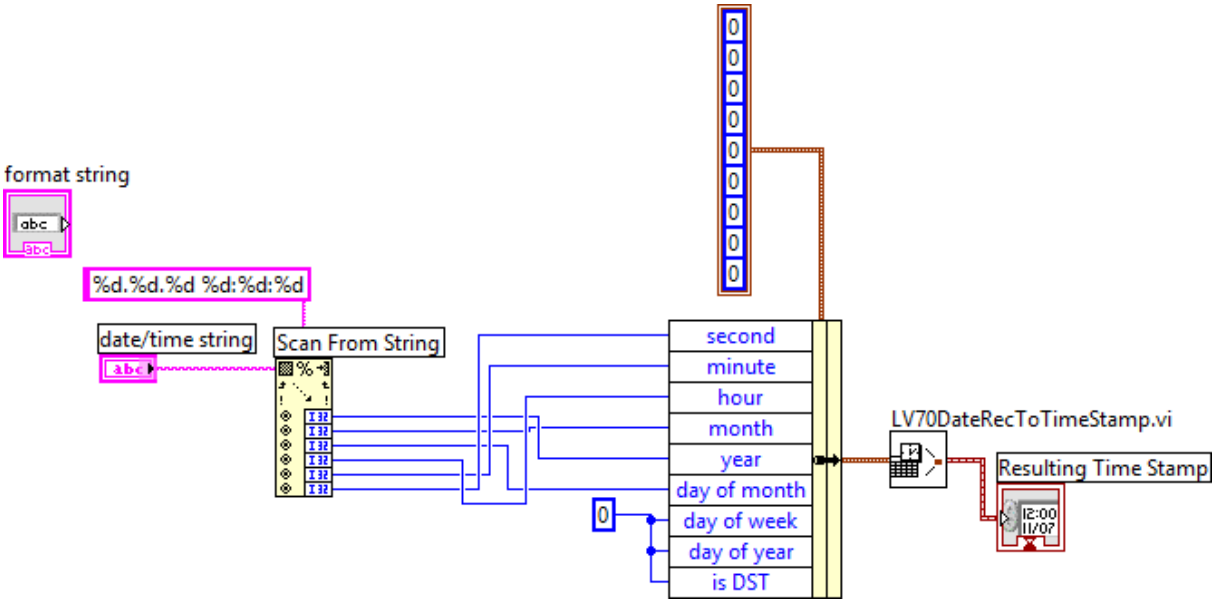


Figura 21 - Block Diagram del VI Convert_String_To_Timestamp.

4.1.3 Strutture dati principali

Prima di intraprendere la descrizione delle varie aree, si preferisce definire le strutture dati necessarie al funzionamento del software. Come anticipato, i dati estratti dal file di dump, una volta convertiti nelle varie tipologie secondo le specifiche, vengono suddivisi in cluster per mantenere la struttura logica dei dati stessi. A questo scopo sono stati realizzati cinque cluster contenenti ciascuno i vettori colonna risultanti dalla fase di estrazione e conversione di cui il dettaglio al paragrafo 4.1.

4.1.3.1 *Cluster Cross*

Questo cluster contiene le informazioni principali relative allo strumento finanziario oggetto del dump. I vettori al suo interno sono:

- Timestamp;
- Open;
- High;
- Low;
- Close;
- Volume.

Il vettore Timestamp è appunto della tipologia data / ora riconosciuta da LabVIEW, mentre tutti gli altri vettori sono di tipo double. A titolo esemplificativo e valido anche per le strutture cluster successivamente descritte, in Figura 22 si mostra uno screenshot del block diagram del cluster e la guida dettagliata dello stesso. Tutti gli altri cluster rispecchiano la medesima struttura.

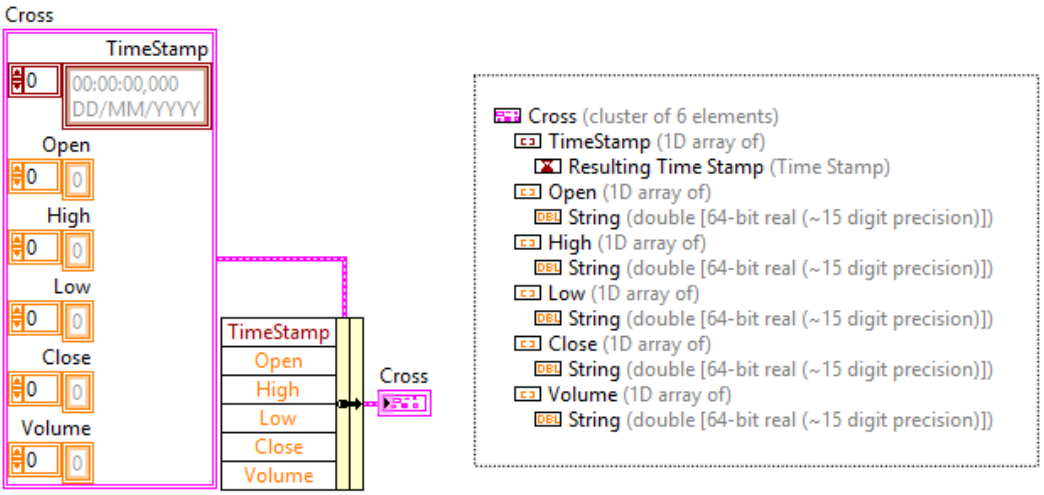


Figura 22 - Struttura dati Cross.

4.1.3.2 *Bollinger*

Questo cluster, formato da due vettori di double, memorizza i vettori colonna relativi alle Bande di Bollinger memorizzate all'interno del file di dump. I vettori al suo interno sono:

- B_UpBand;
- B_LowBand.

4.1.3.3 *EMA*

Il cluster *EMA* contiene al suo interno sette vettori di tipo double, per la memorizzazione delle medie mobili esponenziali (EMA). I vettori al suo interno sono:

- EMA_7;
- EMA_21;
- EMA_63;
- EMA_100;
- EMA_270;
- EMA_540;
- EMA_810.

4.1.3.4 *Stocastico*

Il cluster *Stocastico* contiene al suo interno due vettori di tipo double, per la memorizzazione dell'indicatore stocastico. I vettori al suo interno sono:

- StochMain;
- StochSignal.

4.1.3.5 *ADX*

Il cluster *ADX* contiene al suo interno tre vettori di tipo double, per la memorizzazione dell'indicatore ADX. I vettori al suo interno sono:

- ADX_Main;
- ADX_+DI;

- ADX_-DI.

4.1.3.6 *Altre strutture dati*

Infine, i rimanenti vettori colonna vengono posti in output al VI così come vengono estratti dal file .csv..

- AD: Accumulation Distribution
- Awesome: Awesome Oscillator
- Parabolic_SAR

Ultimi due elementi posti in output sono il nome del file, ovvero l'output *Filename* è il *digit*, ovvero il numero di cifre significative dopo la virgola del valore di prezzo dello strumento. Tale valore è necessario per poter effettuare la conversione in PIPs di una qualunque quantità riferita allo strumento, espressa come differenza di prezzi.

4.1.4 Aree funzionali

Di seguito verranno descritte nel dettaglio le tre aree funzionali in cui è stato suddiviso il codice dell'applicativo. La Figura 23 mostra l'input /output del VI *GetValueFromFile.*

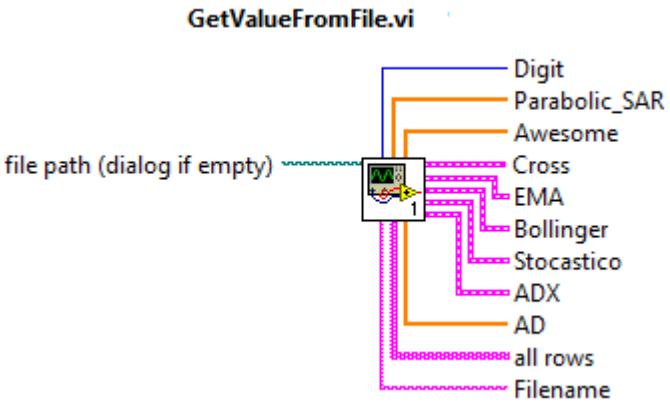


Figura 23 - Input / Output del VI GetValueFromFile.

4.1.4.1 *Area 1*

L'area 1 si occupa di effettuare l'apertura del file .csv contenente il dump dello strumento finanziario. In Figura 24 è possibile vedere che l'uso del VI *Read From SpreadSheet File* permette, specificato il separatore e il formato di dato contenuto del file, di suddividere il file in una matrice di *NxM* elementi.

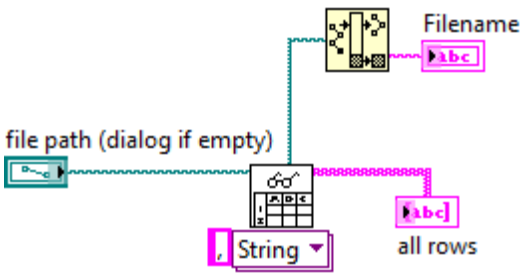


Figura 24 - Area 1: Estrazione del contenuto del file di dump

4.1.4.2 *Area 2*

L'area 2 si occupa, ottenuta la matrice contenente l'intero file di dump suddiviso in colonne, di estrarre le singole colonne e convertirle secondo il tipo di dato cui la colonna si riferisce.

La Figura 25 mostra un estratto del block diagram, riferito alla conversione della colonna dei Timestamp e alla colonna degli *Open* delle candele, che verranno poi inglobati nel cluster *Cross*.

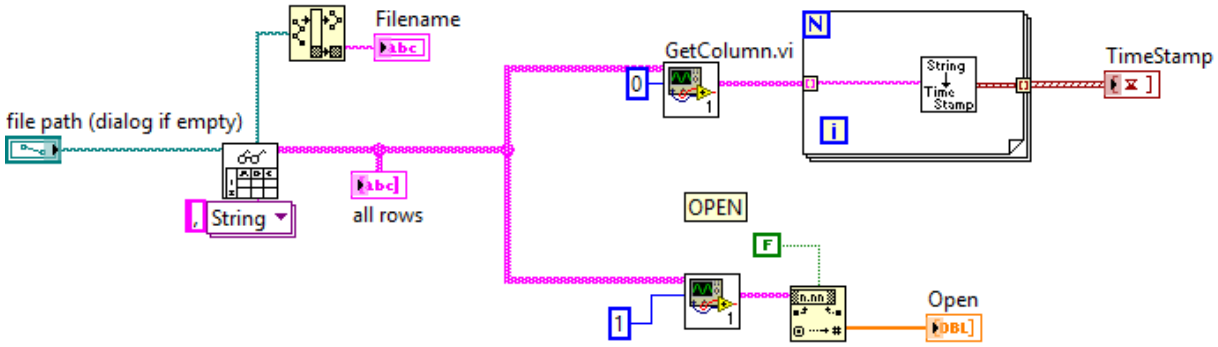


Figura 25 - Area 2: Conversione dei vettori colonna.

In quest'area viene richiamato il VI *GetColumn* e il VI *Convert_String_To_Timestamp*.

4.1.4.3 *Area 3*

L'Area 3 è l'area dedicata alla costruzione dei cluster e dei vettori di output contenenti i dati estratti dal file di dump. Nello specifico la Figura 26 mostra come viene composto uno dei cluster di output, nello specifico quello riferito alle Bande di Bollinger.

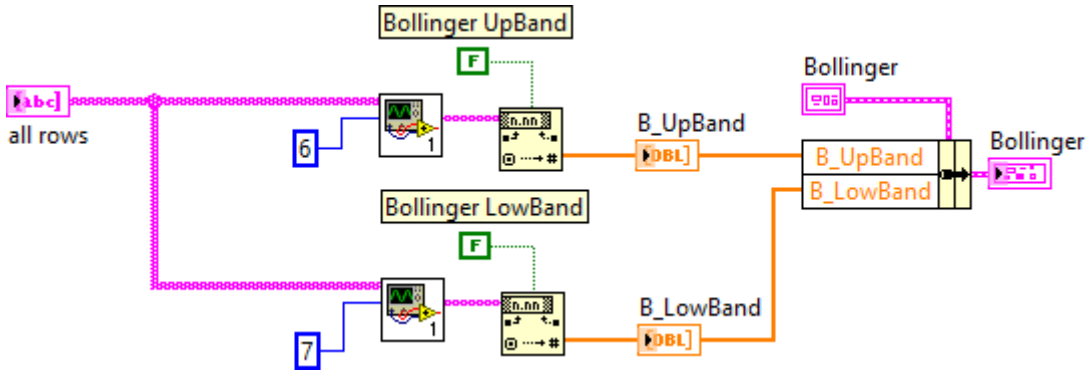


Figura 26 - Area 3: Vettori di Output e costruzione dei cluster.

4.2 CalendarAlert - Introduzione

Considerato il gran numero di news finanziarie che giornalmente che riguardano il mercato del FOREX (ovvero notizie macro economiche e sociali), e la necessità che il trader possa essere avvisato in concomitanza delle stesse al fine di porre l'attenzione sugli strumenti finanziari interessati dalle stesse per trarne eventuali movimenti favorevoli, si è deciso di sviluppare il software di seguito descritto. L'applicativo, utilizzato come generatore di allarmi temporizzati, permette di usufruire di un countdown fino all'ora esatta di avvenimento della news, mostrando all'utilizzatore un conto alla rovescia che viene automaticamente avviato un certo numero di minuti (personalizzabili dall'utente, 15 per impostazione predefinita) antecedenti alla news. Inoltre, l'applicativo emette un messaggio sonoro 5, 3 ed 1 minuti prima del verificarsi della notizia.

L'applicativo costruisce l'elenco aggiornato delle news della giornata effettuando la lettura di tre file .csv, contenenti rispettivamente le news di importanza 1, 2 e 3. Per importanza di una news, ci si riferisce all'impatto possibile che la stessa potrebbe avere sulla valuta interessata. L'importanza è crescente, ovvero una news di livello 3 può impattare sui mercati con una influenza sull'andamento dello strumento tre volte maggiore rispetto ad una news di livello 1.

Tramite una procedura di auto update, i file contenenti le news vengono aggiornati alla prima esecuzione dell'applicativo ed alla mezzanotte del nuovo giorno.

L'aggiornamento automatico, estraendo le news, le suddivide per importanza all'interno dei rispettivi files, memorizzati all'interno della directory di esecuzione dell'estrattore.

Ad esempio:

- C:\Calendar\ Importanza 1.csv;
- C:\Calendar \Importanza 2.csv;
- C:\Calendar \Importanza 3.csv.

Nella prima parte del capitolo si esaminerà in maniera dettagliata il modo in cui è stato strutturato il software. In una prima analisi si descrive la suddivisione in settori del sorgente, andando poi a sviscerare nel dettaglio le varie sezioni. Nella seconda parte si mostrerà l'interfaccia utente e se ne descriverà l'uso.

Il software è stato realizzato utilizzando l'ambiente di programmazione visuale *LabVIEW*, della National Instruments, nello specifico è stata utilizzata la versione 2014 Professional Development System.

Inoltre, a supporto del software, per l'ottenimento delle news aggiornate dal sito *http://it.investing.com*, è stato sviluppato un applicativo in linguaggio Microsoft Visual C#, il quale si occupa di eseguire il parsing della pagina web contenente le news economiche, salvando il contenuto relativo alle sole news su tre file .csv, suddividendo le news per importanza. L'applicativo di parsing è richiamato direttamente dall'applicativo News Calendar.

L'estrazione dei dati relative alle news è stata effettuata utilizzando la libreria *HTML Agility Pack v1.4.6*[1], una libreria costruita in .NET atta appunto ad effettuare il parsing del codice html presente nelle pagine web. I vantaggi che hanno portato alla scelta di utilizzare tale libreria si riferiscono alla semplicità di estrazione delle informazioni e il vantaggio di non avere dipendenze da altre DLL esterne, quali MSHTML o la W3C HTML, o ancora oggetti COM/ACTIVEX.

La libreria riceve in input l'intera pagina HTML, e da questa, tramite la chiamata ad appositi metodi, è in grado di individuare ed estrarre i nodi HTML relativi ai contenuti necessari agli scopi di questo applicativo. Il contenuto utile della pagina viene poi memorizzato su file .csv, e successivamente dato in input all'algoritmo di seguito presentato.

[1] HTML Agility Pack - *http://htmlagilitypack.codeplex.com*

4.2.1 Suddivisione in Aree Funzionali

Il software è stato realizzando cercando di suddividere al meglio il codice in aree funzionali, al fine di renderne la lettura e l'interpretazione da parte di terzi il più semplice e veloce possibile, oltre che meglio manutenibile. La Figura 27 illustra uno schema concettuale di come tale suddivisione è stata realizzata.

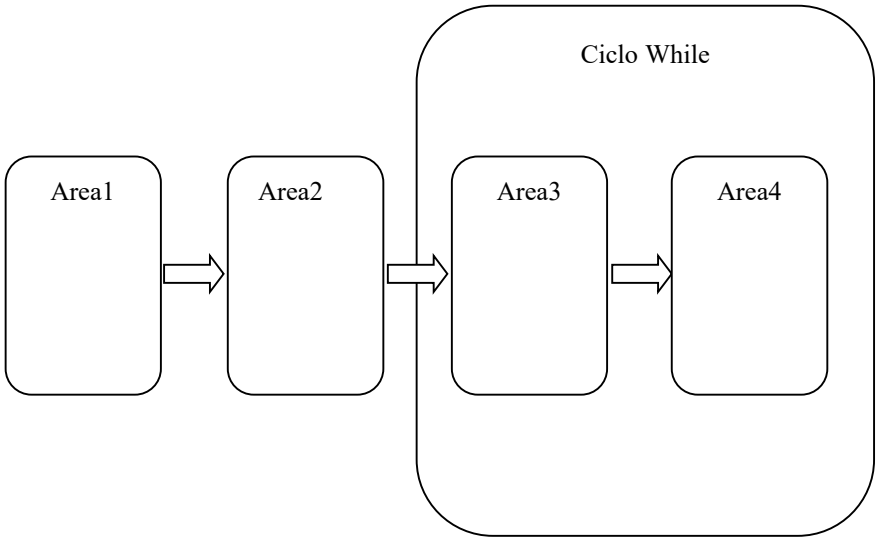


Figura 27 - Suddivisione in aree funzionali.

Esaminiamo quindi le varie aree indicate in figura:

Area 1: Porzione di codice per l'inizializzazione dell'interfaccia grafica ed esecuzione dell'applicazione di download delle news su file, con corrispondente aggiornamento dei file .csv relativi alla singola categoria di news. Questa area si trova al di fuori del ciclo While di esecuzione indicato in figura, proprio perché riguarda delle operazioni di inizializzazione del software da eseguire una sola volta;

Area 2: Tale area si occupa di effettuare il parsing dei file .csv contenenti le singole news suddivise per importanza. Al termine della fase di parsing dei file, i dati vengono importati all'interno dell'applicativo, filtrando le news già trascorse rispetto alla corrente esecuzione; Dopo aver superato queste due sezioni di codice, l'applicativo si evolve con un ciclo while che si occupa, oltre a mantenere attiva l'interfaccia grafica e quindi gestirne l'interazione con l'utente, di analizzare i cluster contenenti le news, eventualmente di avviare i timer, ed aggiornare il giorno successivo il file delle news;

Area 3: Quest'area si occupa di verificare se è trascorso un giorno dall'ultimo update, ed in tal caso di aggiornare i relativi tre file .csv;

Area 4: Lettura delle informazioni relative alle news contenute nei cluster ed eventuale esecuzione del timer, se verificate le condizioni.

4.2.2 VI di uso comune o di supporto

Alcuni *Virtual Instrument* sviluppati, da adesso indicati per brevità *VI*, svolgono funzionalità generiche, non specifiche per un singolo obiettivo. Al fine di evitare ripetizioni, si descrivono in questa sezione tali vi, così da poterli menzionare senza descrizioni ripetute nelle pagine successive.

4.2.2.1 *GetNextDay.vi*

Questo vi riceve in input un oggetto di tipo Timestamp, destruttura il cluster ad esso relativo e incrementa il campo *DayOfMonth*. Fornisce in output il Timestamp di origine aggiornato alle ore 0.00 del giorno successivo. Il VI è utilizzato per ottenere il Timestamp del prossimo aggiornamento dei files delle news.

In Figura 28 è possibile visualizzare l'input / output del VI, mentre in Figura 29 si mostra il block diagram relativo.

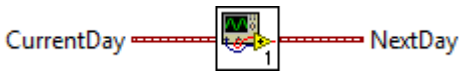


Figura 28 - Input / Output del VI GetNextDay

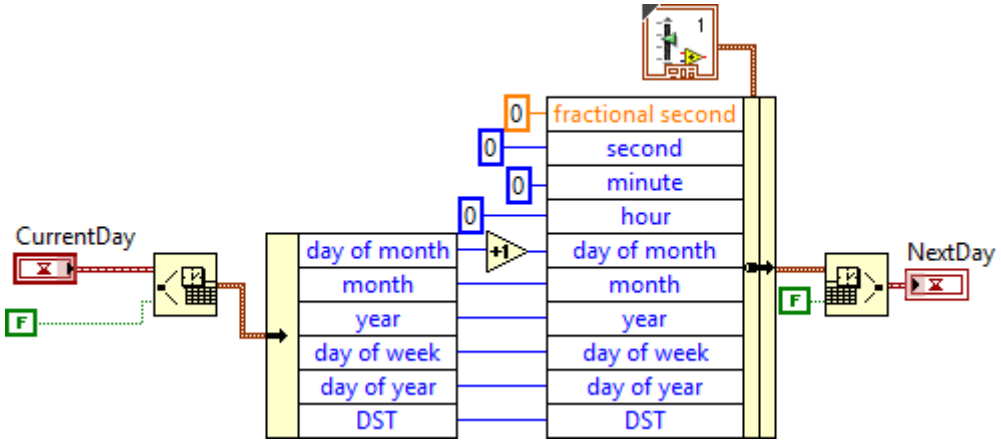


Figura 29 - Block Diagram del VI GetNextDay.

4.2.2.2 *GetUsefulTS.vi*

Questo VI riceve in input un oggetto di tipo Timestamp (nello specifico quello proveniente dal VI *GetNextDay*), e verifica se il giorno cui il Timestamp si riferisce è un giorno ufficiale di trading, oppure un festivo, verificando che il campo *DayOfMonth* del cluster del Timestamp non valga 7 o 1 (ovvero sabato o domenica). In questi casi il VI modifica il Timestamp aggiornandolo al primo lunedì utile. La

Figura 30 mostra l'input / output del VI, mentre la Figura 31 mostra il block diagram dello stesso.

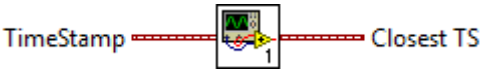


Figura 30 - Input / output del VI GetUsefulTS.

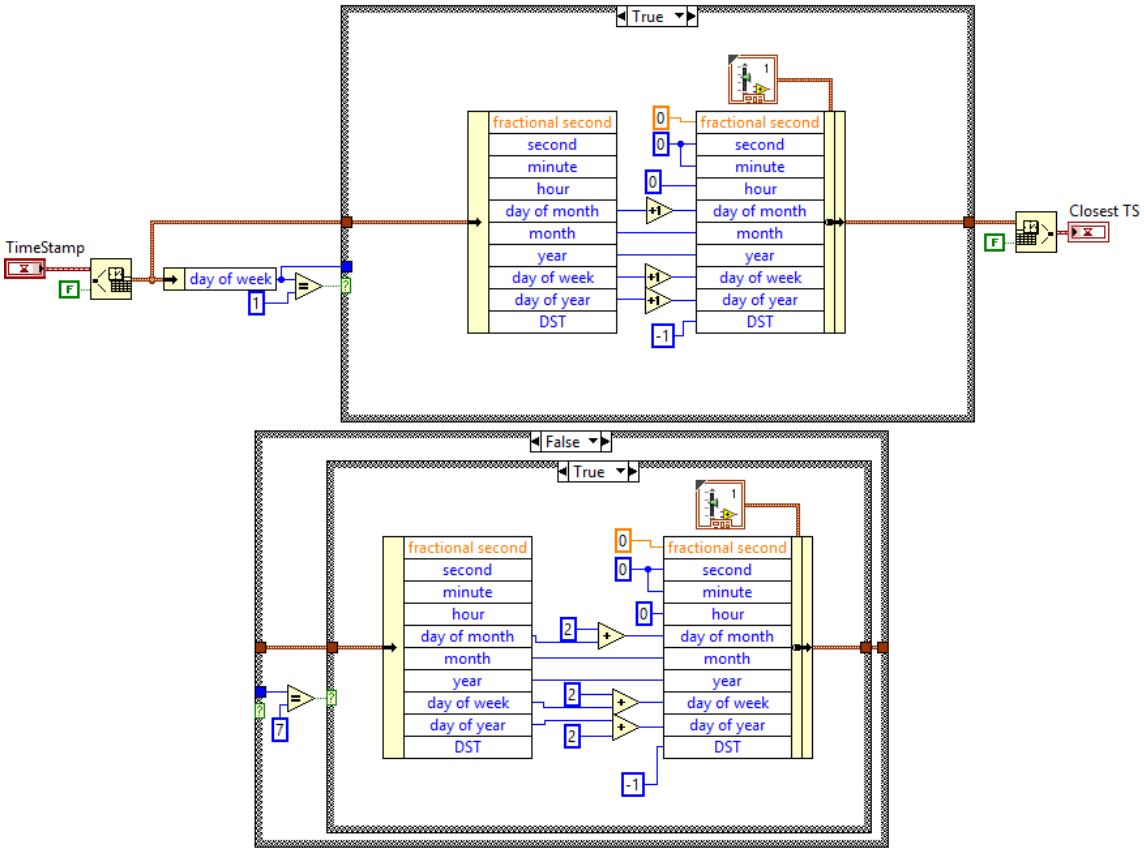


Figura 31 - Block Diagram del VI GetUsefulTS

4.2.3 Strutture dati principali

Prima di intraprendere la descrizione delle varie aree, si preferisce definire le strutture dati necessarie al funzionamento del software.

4.2.3.1 *News_Data*

Tale struttura dati è un Array di cluster, e rappresenta la struttura dati che viene generata dal VI che si occupa di estrarre le news dai file .csv. Nello specifico, l'array conterrà tre cluster, uno per ogni categorizzazione delle news. Nel dettaglio ogni cluster è così strutturato:

- *Intestazione:* Array unidimensionale di Stringhe, contiene l'intestazione delle varie colonne estratte dai file .csv delle news.
- *Timestamp News:* Array unidimensionale di Timestamp, contiene appunto i Timestamp relativi alle news estratte, ordinati per orario.
- *NewsInfo:* Array bidimensionale di Stringhe, in cui la singola riga si riferisce alla singola news, mentre le colonne corrispondono ai campi:
    - Timestamp;
    - Valuta;
    - Evento;
    - Valore Attuale di impatto sulla valuta interessata;
    - Valore Previsto di impatto sulla valuta interessata;
    - Valore Precedente di impatto sulla valuta interessata.

La Figura 32 mostra la struttura di tale array di cluster.

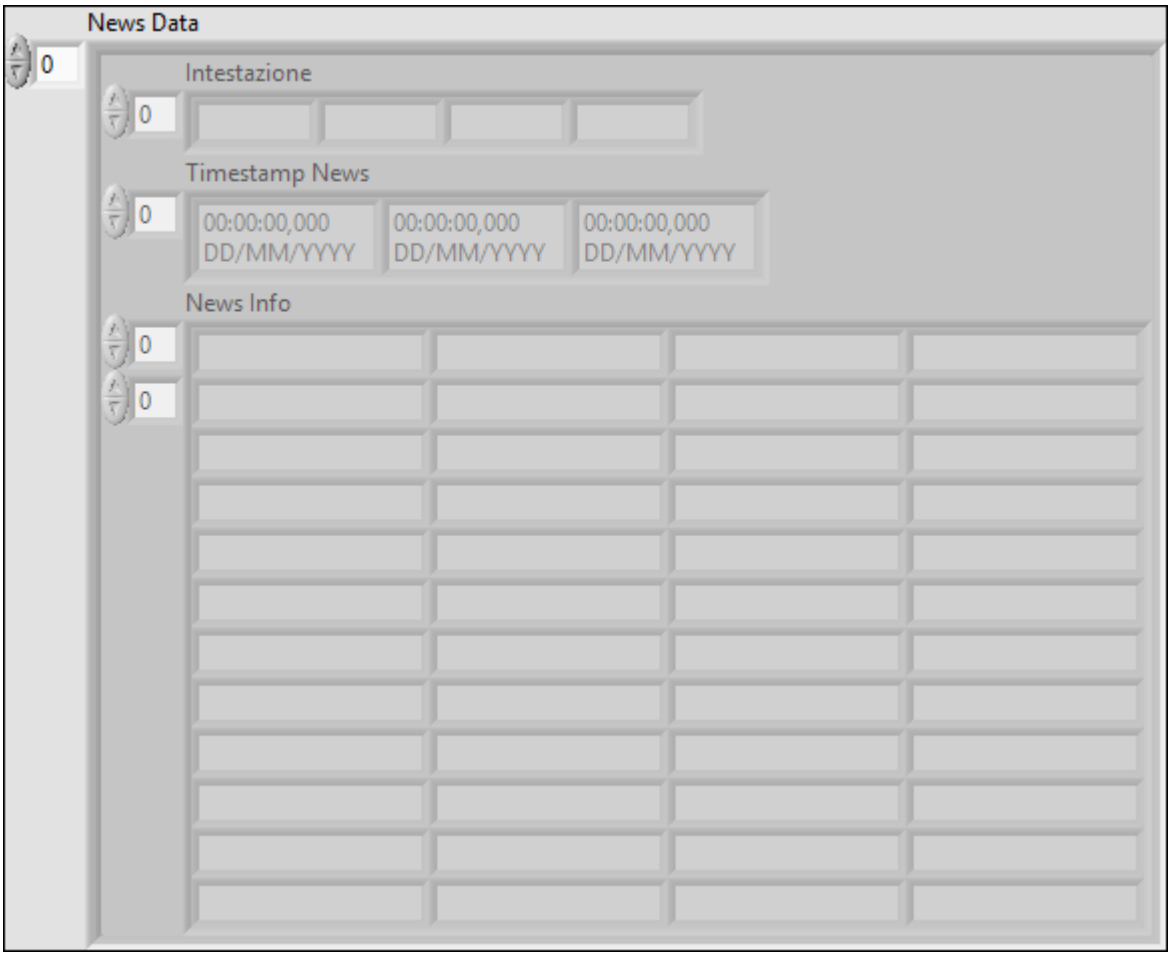


Figura 32 - Struttura dati atta a memorizzare le news estratte dai file .csv.

4.2.4 Aree funzionali

Di seguito verranno descritte nel dettaglio le quattro aree funzionali in cui è stato suddiviso il codice dell'applicativo News Calendar.

4.2.4.1 *Area 1*

L'area 1 si occupa di inizializzare l'interfaccia grafica ed eseguire l'aggiornamento dei file .csv contenente le news di interesse, e suddivise per categoria. Questo compito è demandato all'applicativo realizzato in Visual C#, brevemente descritto nell'introduzione a questo capitolo.

In Figura 33 si mostra, nella parte superiore, il dettaglio degli input necessaria alla funzione per l'esecuzione di un eseguibile esterno all'ambiente LabVIEW, mentre nella metà inferiore la porzione di codice che esegue il suddetto applicativo per l'aggiornamento dei file delle news.

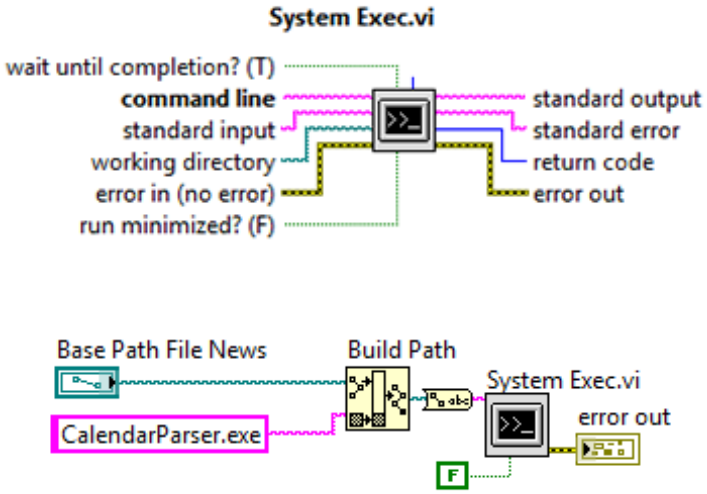


Figura 33 - Sezione dedita all'update del file delle news finanziarie.

4.2.4.2 *Area 2*

L'area 2 si occupa di effettuare il parsing dei file .csv contenenti le news. Quest'area esamina il contenuto della directory radice, costruendo i path ai tre *news-file*. Tali path vengono passati al VI che si occupa della effettiva estrazione ed inizializzazione dell'array di cluster *News-Data*.

Al termine della fase di parsing dei file, l'output (ovvero l'array delle news), verrà posto in input all'Area 3 che si occuperà di avviare i timer per le news imminenti, ovvero per quelle news che rientrano in un intervallo temporale dall'evento di un tempo inferiore o uguale a quello prestabilito per avviare il timer.

In quest'area, di cui si mostra un estratto del block diagram in Figura 35, viene richiamato il VI *Get_NewsInfo*, di cui di seguito si elencano, dettagliandone l'utilità, gli input e gli output (v. Figura 34):

- *Get_NewsInfo.vi* riceve in input:
    - *file_path*: percorso assoluto al file .csv;
    - *Cross*: stringa di testo contenente il nome dello strumento finanziario in esame, utilizzato all'interno del vi per poter effettuare l'estrazione delle sole news di interesse;
    - *FilterByCurrency*: booleano per attivare il filtraggio delle news secondo lo strumento specificato;
    - *CurrentTS*: Timestamp per filtrare, dalla totalità delle news estratte, quelle già verificatesi rispetto all'ora di esecuzione del vi.

Output:

- *Intestazione*: Array di stringhe contenente l'intestazione estratta dal file .csv;
- *Timestamp News*: Array ordinato di Timestamp relativi alle news future;
- *NewsInfo*: Matrice di stringhe contenente tutte le informazioni estratte dal file .csv.

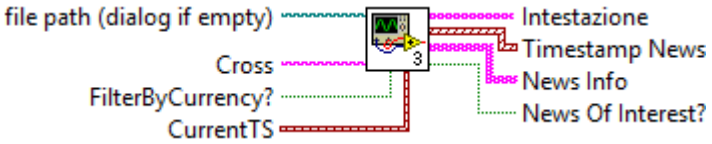


Figura 34 - Input / output del VI Get_NewsInfo

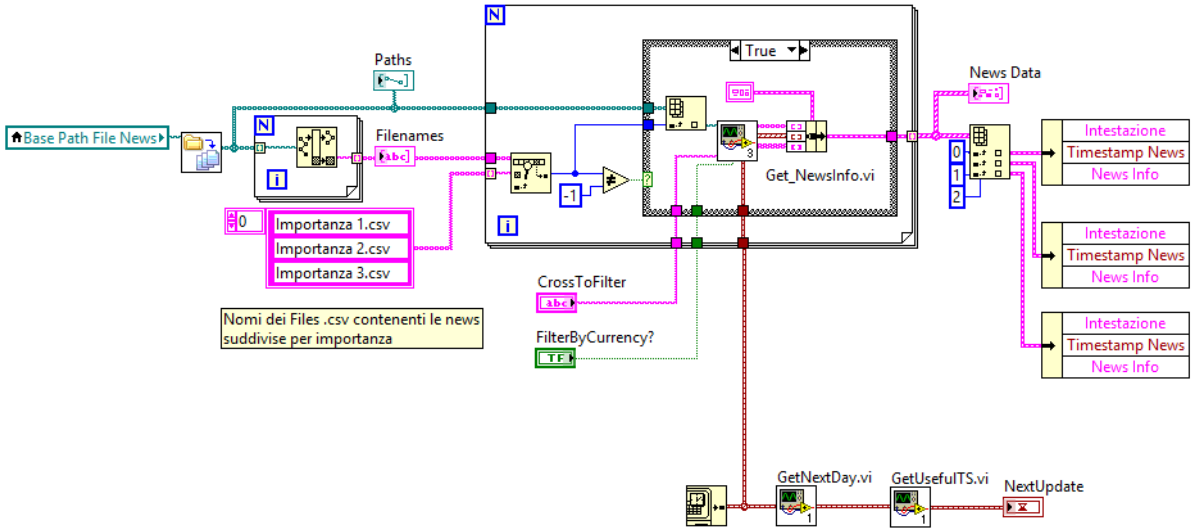


Figura 35 - Block diagram di utilizzo del VI GetNewsInfo.

Il parsing delle news avviene tramite esecuzione di particolari espressioni regolari sul contenuto dei files. Una volta effettuata tale fase, ovvero dopo aver ottenuto dal file i campi Intestazione, vettore contenente i Timestamp delle news e la matrice con i dati ad esse relativi, se selezionata l'opzione di filtraggio sullo strumento finanziario, si esegue il VI *FilterNewsByCurrencyFromFile* (v. Figura 36).

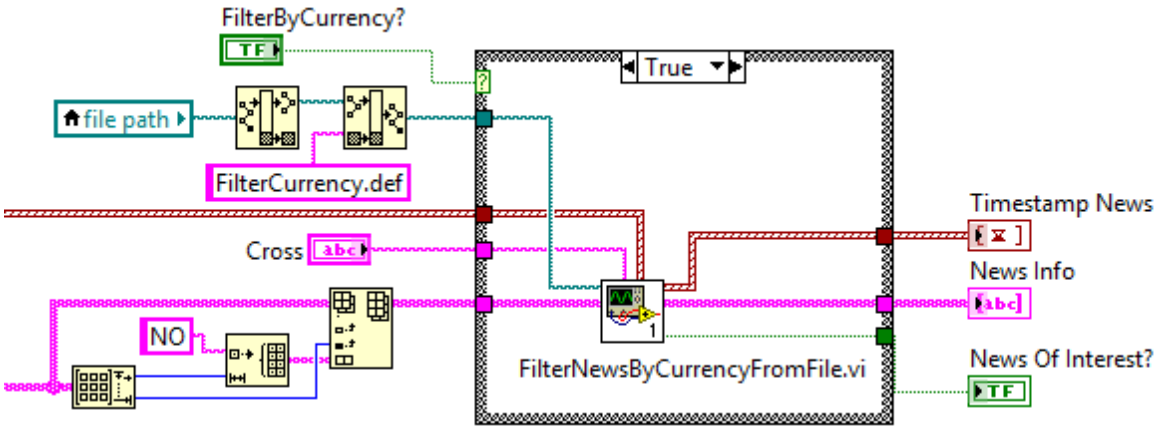


Figura 36 - Block diagram di utilizzo del VI FilterNewsByCurrencyFromFile.

Tale VI, al fine di poter effettuare il filtraggio delle news, deve fare riferimento ad una hashmap, scritta su di un file di configurazione e denominato *FilterCurrency.def*. Tale hashmap si rende necessaria in quanto, per ogni cross, è necessario conoscere le valute che possono influenzare l'andamento dello strumento, ad esempio per il GOLD Spot l'Euro, il Dollaro, la Sterlina e lo Yen.

Il formato di ogni record del file di filtraggio è del tipo

- *Chiave* = Cross;
- *Valore* = Espressione Regolare.

Il VI estrae l'espressione regolare opportuna in base al cross in esame e rimuove tutte le occorrenze delle news dal vettore in input che non rispettano l'espressione regolare.

Nel dettaglio:

- FilterNewsByCurrencyFromFile.vi:
    - *Input Timestamp News*: Array di Timestamp delle news da filtrare;
    - *Cross*: Stringa contenente la currency per individuare la RegExp (all'interno del file) da applicare per il filtraggio;
    - *CurrencyFilter FilePath*: path assoluto al file contenente l'hashmap;
    - *NewsInfo*: Matrice di stringhe contenente tutte le informazioni estratte dal file .csv sulla news, sul quale applicare il filtraggio.

  Output:

    - *Timestamp News*: array dei Timestamp delle news future filtrate secondo lo strumento finanziario;
    - *NewsInfo*: matrice dei dati delle news filtrate secondo lo strumento finanziario;
    - *NewsOfInterest*: booleano indicante se vi sono news di interesse post-filtraggio.

La Figura 37 mostra gli input del VI, mentre la Figura 38 mostra il block diagram dello stesso.

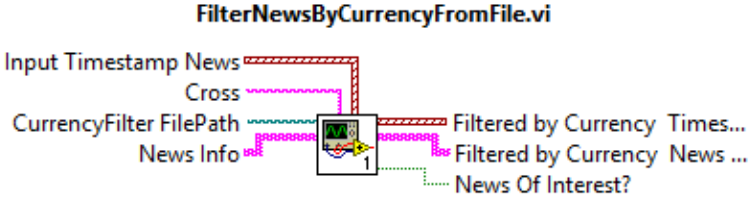


Figura 37 - Input / output del VI FilterNewsByCurrencyFromFile

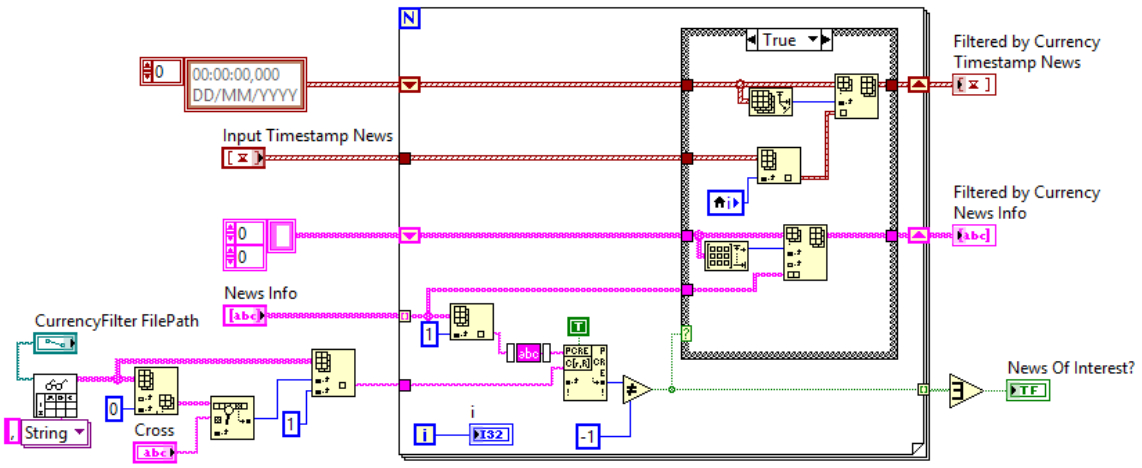


Figura 38 - Block Diagram del VI FilterNewsByCurrencyFromFile.

Infine in tale area viene individuato, a partire dall'istante corrente, giorno e ora in cui dovrà avvenire il nuovo update del file delle news. Tale operazione viene effettuata tramite i VI già documentati *GetNext-Day.vi* e *GetUsefulTS.vi.*

4.2.4.3 *Area 3*

Dopo la fase di inizializzazione tramite le due aree precedentemente dettagliate, l'applicativo si evolve con il ciclo while il quale, una volta al secondo, confronta dapprima il Timestamp corrente con quello del successivo update del file delle news.

In tal caso si ripete la fase di update del calendario economico descritta nei paragrafi 3.1 e 3.2.

Il ciclo while gestisce l'esecuzione dell'applicativo fintanto che l'esecuzione dello stesso non viene interrotta dall'utente. La Figura 39 mostra tale sezione del block diagram.

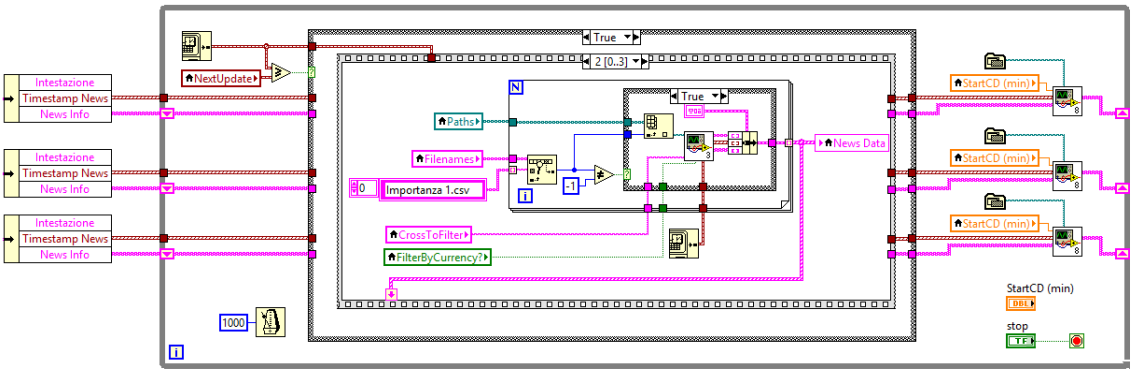


Figura 39 - Block Diagram dell'area di gestione dell'interfaccia grafica e di esecuzione dei countdown timer all'avvento di una nuova news

4.2.4.4 *Area 4*

Dopo aver eventualmente aggiornato il file delle news e i rispettivi file .csv, ad ogni iterazione del ciclo while viene richiamato il VI *ExecuteTimer*. Tale vi, di cui ne viene eseguita una istanza per ogni tipologia di news (Importanza 1, 2 e 3), si occupa di esaminare ogni elemento dell'array dei Timestamp delle news alla ricerca della prossima news appartenente all'intervallo definito dal parametro *StartCD*. In tal caso si esegue, tramite l'uso della funzione built-in del LabVIEW per l'esecuzione di System Command, la versione compilata del CountDownTimer.

Tale VI, riceve in input le informazioni relative alla news in esame, attraverso la System Command, ovvero:

- Ore, minuti e secondi rimanenti alla news;
- Ora corrente;
- Valuta Interessata dalla news;
- Importanza (volatilità attesa);
- Descrizione della news;
- Indicatori On/Off indicanti i minuti rimanenti alla news (5minuti, 3minuti, 1minuto);
- Pulsante di arresto del Timer.

ExecuteTimer.vi (v. Figura 40 e Figura 41), riceve in input:

- *CurrentDirectory*: Directory all'interno della quale si trova la versione eseguibile tramite SystemExec del *CountDownTimer*;
- *Timestamp News In*: Array dei Timestamp delle news non ancora trascorse;
- *NewsInfo In*: Matrice di Stringhe contenenti i dati della news da mostrare negli opportuni campi del *CountDownTimer;*
- *StartCD*: Tempo in Minuti antecedenti la news di esecuzione del timer.

Output:

- *NewsInfo Out*: Matrice di stringhe contenente i dati delle news, in cui il campo "*Trascorsa*" relativo alla news imminente appena avvenuta viene settato su "*SI*". Tale campo viene utilizzato per evitare che la news venga nuovamente considerata dall'applicativo.

Figura 40 - Input / output del VI Execute Timer

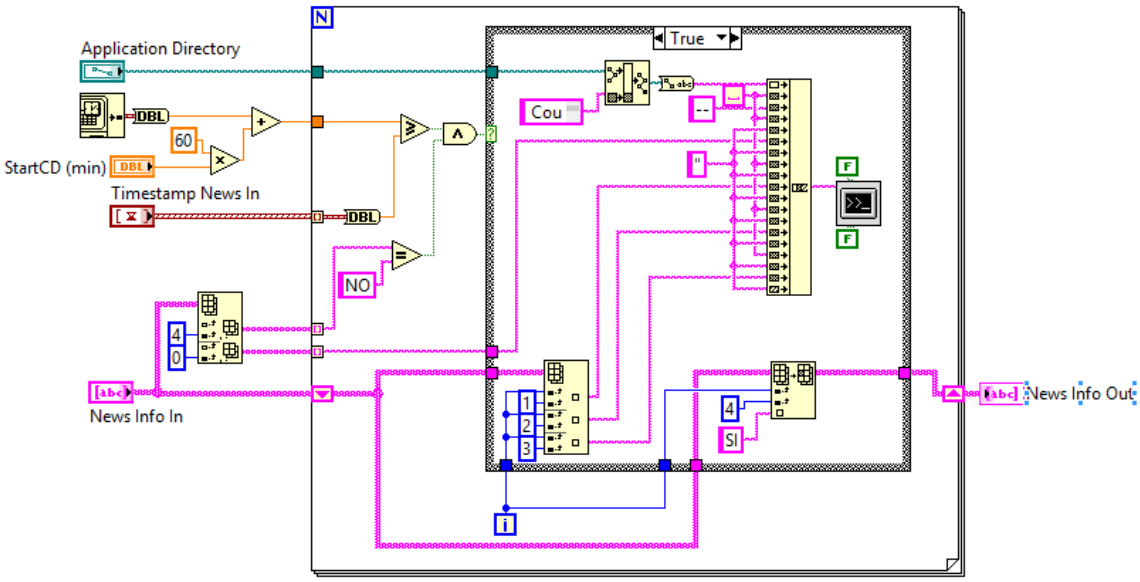


Figura 41 - Block diagram del VI ExecuteTimer

4.2.5 KeyNotes

Si rende noto che, all'interno della directory di esecuzione dell'applicativo del calendario, devono essere presenti i seguenti files:

- CalendarAlert.exe
- CalendarAlert.aliases
- CalendarAlert.ini
- CountDownTimer.exe
- CountDownTimer.aliases
- CountDownTimer.ini
- CalendarParser.exe
- Directory TempNews

Nello specifico, il file *CountDownTimer.ini* deve contenere, come ultimo record, la seguente voce:

- allowmultipleinstances=True

Tale dicitura permette di eseguire multiple istanze dell'applicativo contemporaneamente. Tale condizione si verifica quando vi sono più news previste con lo stesso istante temporale, siano queste di importanza uguale che differente.

4.2.6 Utilizzo dell'applicativo

L'applicativo consta di una semplice interfaccia grafica, all'interno della quale è possibile visualizzare principalmente la struttura dati *News Data*, contenente le news estrapolate dai file csv. L'applicativo si mostra all'utente con l'interfaccia grafica di Figura 42:

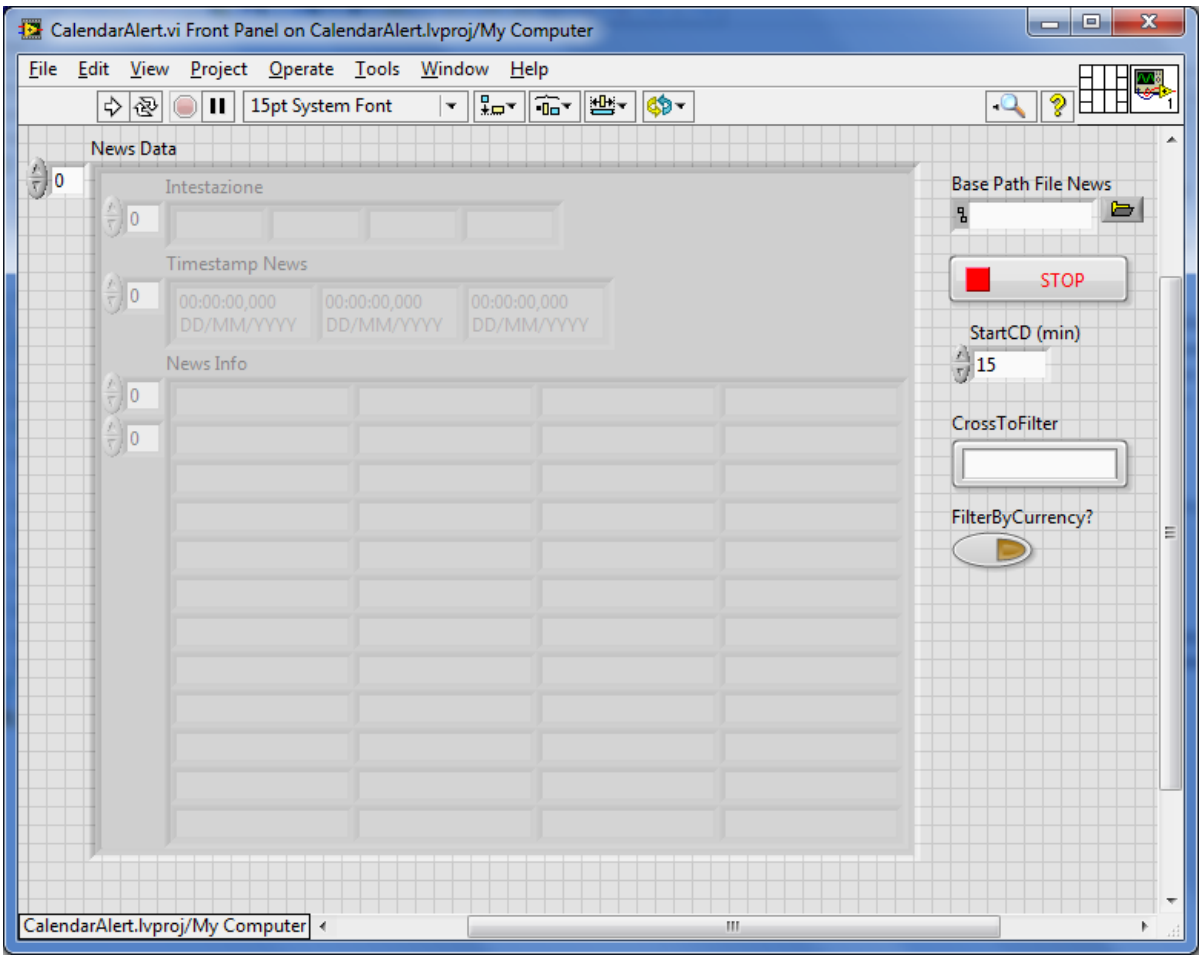


Figura 42 - Interfaccia grafica dell'applicativo Calendar Alert

I pochi parametri di input necessari al funzionamento dell'applicativo sono:

- *Base Path File News*: Percorso alla directory che contiene il file "Dati-News.xlsm";
- *StartCD (min)*: Numero di minuti antecedenti la news da cui avviare il count-down;
- *CrossToFilter*: Campo di testo in cui è possibile specificare lo strumento finanziario di interesse per effettuare un filtraggio sulle news;
- *FilterByCurrency*: Attiva il filtraggio delle news:
- *Stop*: Arresta l'esecuzione dell'applicativo

4.2.7 Count-Down Timer

Nel momento in cui si raggiunge l'istante temporale definito per l'esecuzione del timer di una news, il timer si mostra all'utente con l'interfaccia grafica di Figura 43:

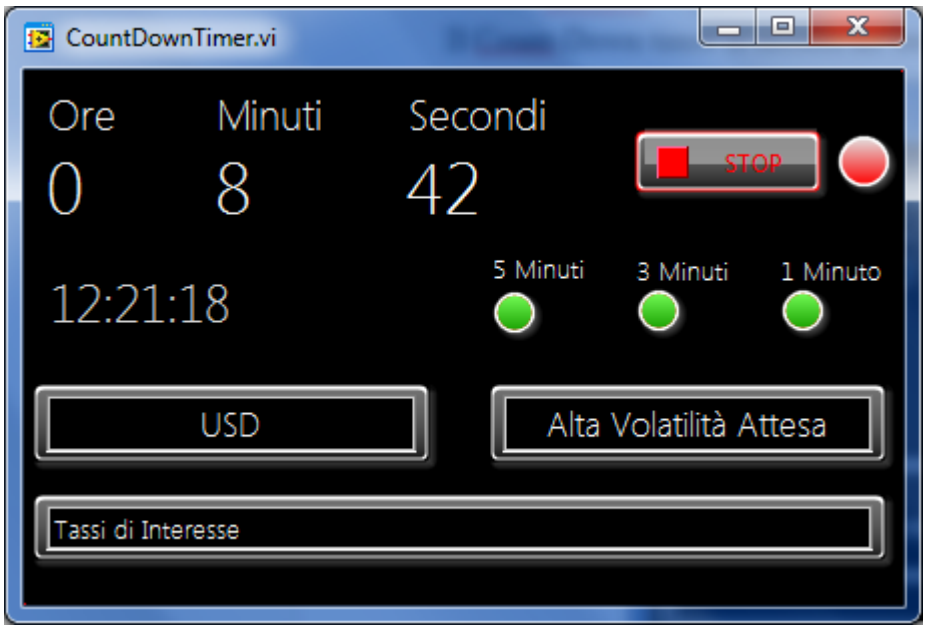


Figura 43 - Interfaccia grafica del VI CountDownTimer

Il timer mostra all'utente varie informazioni, nel dettaglio:

- Tempo alla news, ovvero ore, minuti e secondi rimanenti alla news;
- Ora Corrente;
- Indicatori "5 Minuti", "3 Minuti" ed "1 Minuto", i quali vengono spenti a 5, 3 ed 1 minuti dalla news. Contestualmente alla disattivazione dell'indicatore, viene emesso un segnale di notifica.
- Valuta interessata dalla news;
- Volatilità attesa per la news in esame (Bassa, Media, Alta);
- Informazioni dettagliate riguardanti la news.
- Tasto "Stop", per la chiusura anticipata del timer.

5 Controllo decentralizzato di account multipli con marginazione del rischio

In questo capitolo si descrive una parte della soluzione software sviluppata nell'ambito del lavoro di ricerca, nata dall'esigenza di poter gestire l'operatività di più utenti, legandone il comportamento ad un account master. Si è quindi deciso di realizzare un set di applicativi in grado di replicare l'operatività dell'account master su un numero potenzialmente infinito di account slave. Inoltre si è pensato alla necessità di poter gestire tali account non in locale, bensì in remoto, ovvero di dare la possibilità all'utente con account slave di visualizzare ed eventualmente e replicare l'operatività del account master senza dover essere direttamente connesso a tale account.

A tal fine la soluzione software realizzata consta di due applicativi, realizzati in Microsoft Visual C#, uno per la gestione dell'account master ed uno per la replica dell'operatività sull'account slave, di due Expert Advisor atti all'interfacciamento con la piattaforma per le modalità master/slave ed infine di una pagina web.

La pagina web, nello specifico, è stata realizzata per un duplice scopo: la prima è quella di poter permettere la gestione decentralizzata del cloning dell'operatività, in quanto i due applicativi vi accedono da remoto, previa autenticazione al sistema; la seconda è quella di dare la possibilità ad un qualunque trader o operatore di mercato, di visualizzare ed ottenere informazioni riguardo l'operatività degli utenti registrati al servizio.

I due applicativi per il cloning dell'operatività sono dotati di una interfaccia grafica di semplice utilizzo, nella quale si rende disponibile l'accesso al sistema tramite inserimento di username e password utilizzate in fase di registrazione. L'applicativo, una volta attivata la modalità cloning, permane in esecuzione in background, permettendo così all'utente di continuare le sue attività sulla macchina sulla quale è in esecuzione.

È stato necessario realizzare infine due Expert Advisor, il primo per l'interfacciamento con l'applicativo di gestione del conto master, il secondo atto a memorizzare l'apertura e la chiusura di nuove operazioni, per l'applicazione di gestione degli account slave.

Uno dei vantaggi dato dall'utilizzo del sistema di account cloning, dettagliato di seguito, risiede nella possibilità data al nuovo utente durante la fase di registrazione al sito web, di selezionare il proprio profilo di rischio, in relazione a quello dell'account master. Di seguito si mostra un diagramma concettuale di come è strutturato il sistema di cloning dell'operatività.

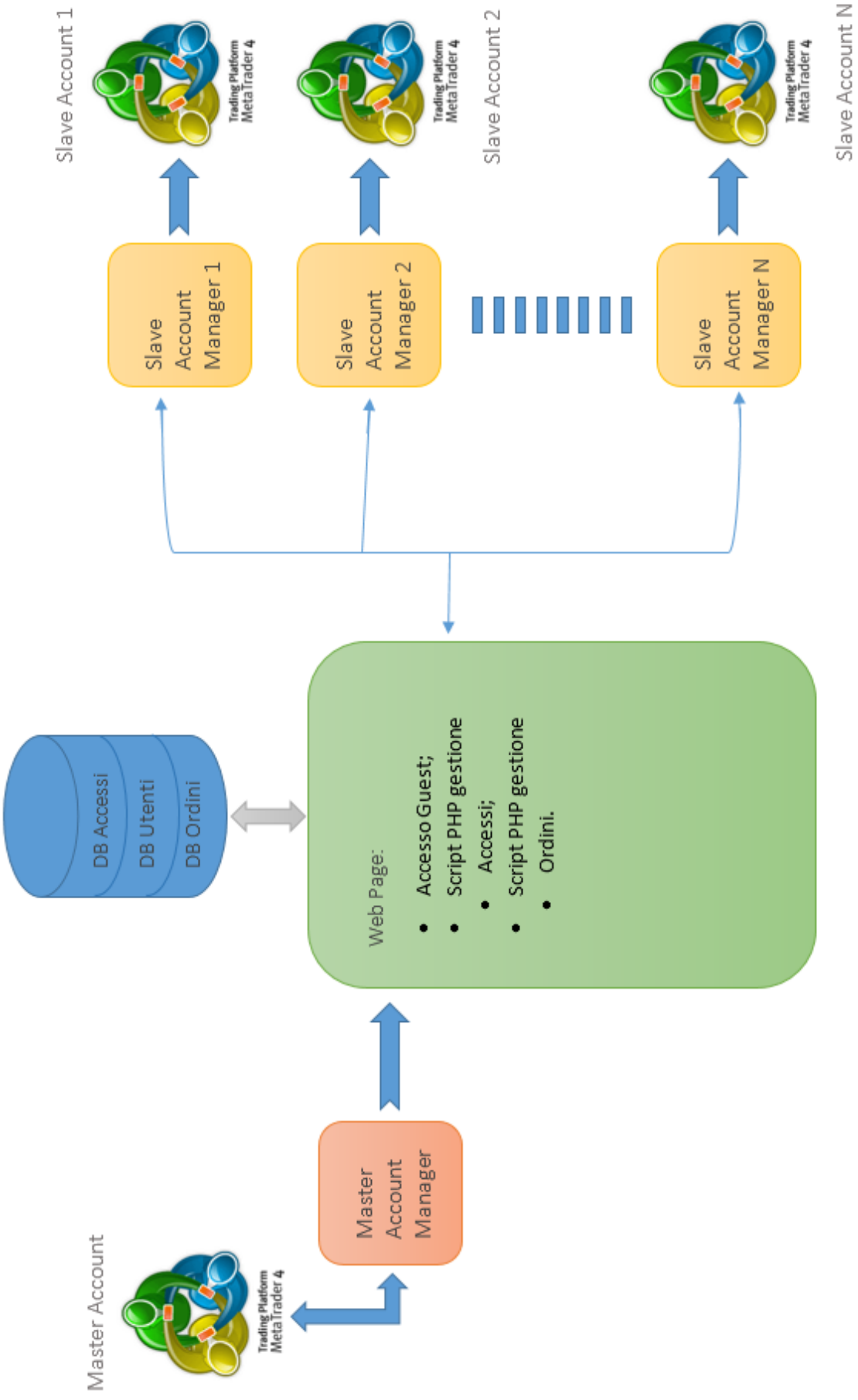


Figura 44 - Diagramma concettuale della soluzione di cloning

5.1 La Pagina web

La pagina web indicata nell'introduzione funge da nodo di comunicazione tra l'account master e gli account slave. Lo scopo primario è la gestione dell'account master, ovvero la memorizzazione delle operazioni che il trader o l'algoritmo di trading automatico pongono in essere sull'account master, e la visualizzazione di tali informazioni. Difatti un qualunque utente ha la possibilità di registrarsi alla pagina web e visualizzare gli ordini aperti o chiusi da parte dell'utente master e degli ulteriori *N* utenti registrati. Inoltre, durante la fase di registrazione, viene richiesto all'utente di specificare il profilo di rischio, ovvero la percentuale dell'allocazione dell'account master che l'utente ha intenzione di replicare sul proprio account per ogni operazione. La Figura 45 mostra l'home page della pagina web, mentre la Tabella 2 riassume i possibili profili di rischio disponibili in fase di registrazione.

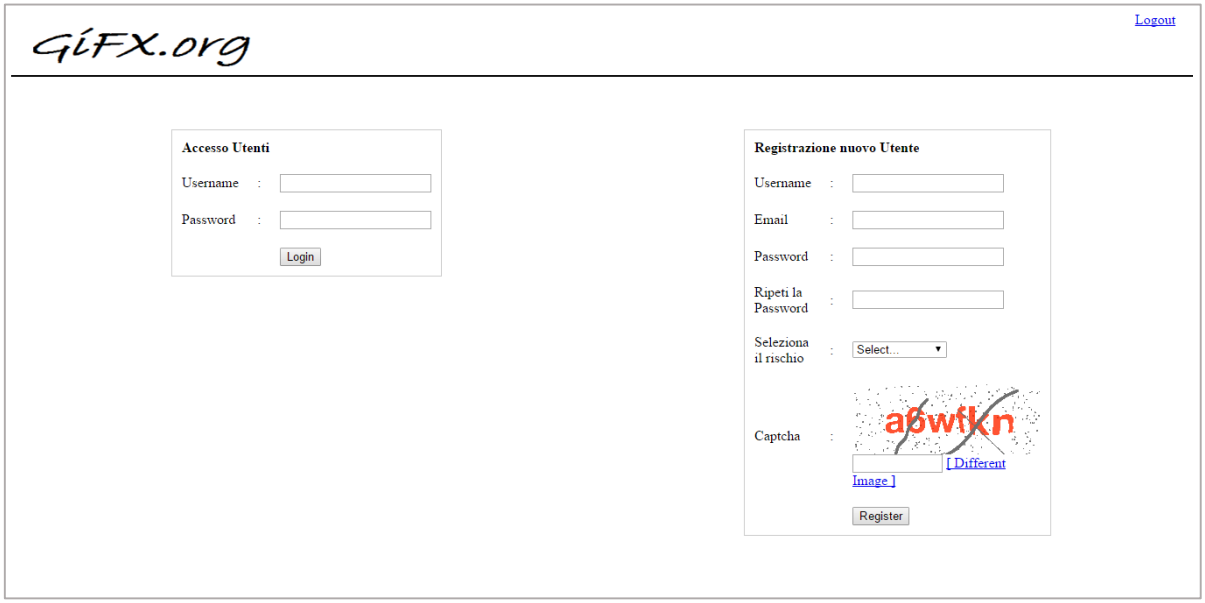


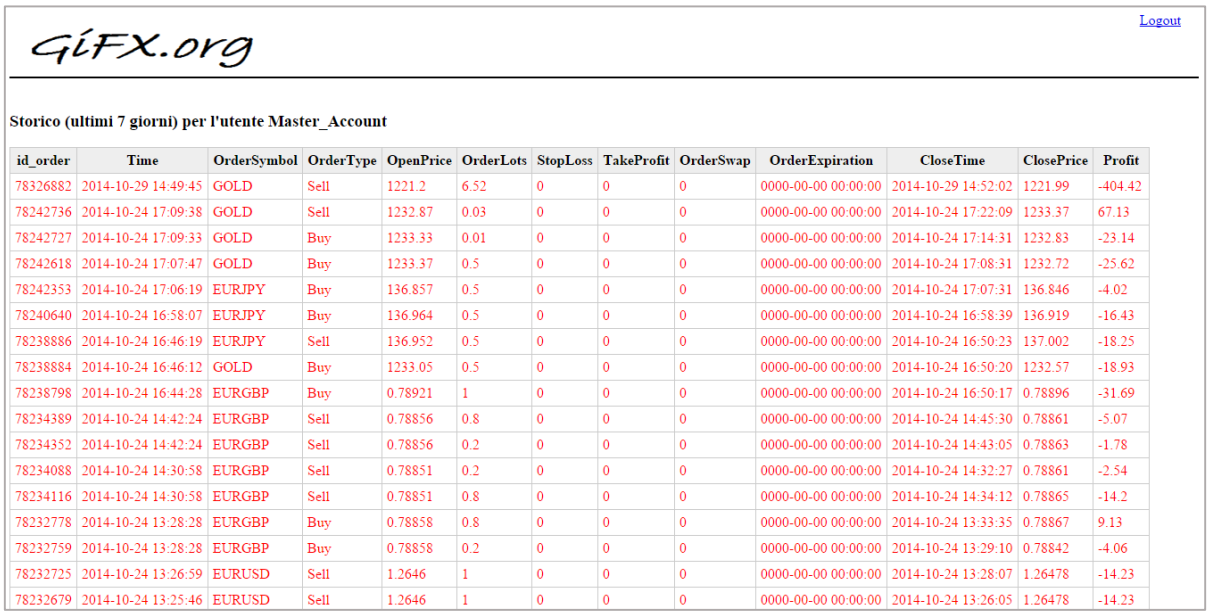
Figura 45 - Home Page

Profili di Rischio (% dell'account master)
100 % del Rischio
80 % del Rischio
60 % del Rischio
40 % del Rischio
20 % del Rischio
10 % del Rischio
4 % del Rischio
2 % del Rischio
1 % del Rischio
0,1 % del Rischio

Tabella 2 - Profili di rischio selezionabili (%)

A seguito della fase di registrazione del nuovo utente, il sistema associa il profilo di rischio selezionato, in maniera tale da permettere il dimensionamento corretto delle posizioni in fase di cloning. Dopo aver effettuato l'accesso, viene mostrato un elenco degli utenti registrati al sistema, dei quali è possibile

visualizzare, se ne esistono di disponibili, tutti gli ordini nell'ultima settimana di trading, evidenziati in verde se ancora a mercato, in rosso se chiusi (v. Figura 46).



Logout

GiFX.org

Storico (ultimi 7 giorni) per l'utente Master_Account

id_order	Time	OrderSymbol	OrderType	OpenPrice	OrderLots	StopLoss	TakeProfit	OrderSwap	OrderExpiration	CloseTime	ClosePrice	Profit
78326882	2014-10-29 14:49:45	GOLD	Sell	1221.2	6.52	0	0	0	0000-00-00 00:00:00	2014-10-29 14:52:02	1221.99	-404.42
78242736	2014-10-24 17:09:38	GOLD	Sell	1232.87	0.03	0	0	0	0000-00-00 00:00:00	2014-10-24 17:22:09	1233.37	67.13
78242727	2014-10-24 17:09:33	GOLD	Buy	1233.33	0.01	0	0	0	0000-00-00 00:00:00	2014-10-24 17:14:31	1232.83	-23.14
78242618	2014-10-24 17:07:47	GOLD	Buy	1233.37	0.5	0	0	0	0000-00-00 00:00:00	2014-10-24 17:08:31	1232.72	-25.62
78242353	2014-10-24 17:06:19	EURJPY	Buy	136.857	0.5	0	0	0	0000-00-00 00:00:00	2014-10-24 17:07:31	136.846	-4.02
78240640	2014-10-24 16:58:07	EURJPY	Buy	136.964	0.5	0	0	0	0000-00-00 00:00:00	2014-10-24 16:58:39	136.919	-16.43
78238886	2014-10-24 16:46:19	EURJPY	Sell	136.952	0.5	0	0	0	0000-00-00 00:00:00	2014-10-24 16:50:23	137.002	-18.25
78238884	2014-10-24 16:46:12	GOLD	Buy	1233.05	0.5	0	0	0	0000-00-00 00:00:00	2014-10-24 16:50:20	1232.57	-18.93
78238798	2014-10-24 16:44:28	EURGBP	Buy	0.78921	1	0	0	0	0000-00-00 00:00:00	2014-10-24 16:50:17	0.78896	-31.69
78234389	2014-10-24 14:42:24	EURGBP	Sell	0.78856	0.8	0	0	0	0000-00-00 00:00:00	2014-10-24 14:45:30	0.78861	-5.07
78234352	2014-10-24 14:42:24	EURGBP	Sell	0.78856	0.2	0	0	0	0000-00-00 00:00:00	2014-10-24 14:43:05	0.78863	-1.78
78234088	2014-10-24 14:30:58	EURGBP	Sell	0.78851	0.2	0	0	0	0000-00-00 00:00:00	2014-10-24 14:32:27	0.78861	-2.54
78234116	2014-10-24 14:30:58	EURGBP	Sell	0.78851	0.8	0	0	0	0000-00-00 00:00:00	2014-10-24 14:34:12	0.78865	-14.2
78232778	2014-10-24 13:28:28	EURGBP	Buy	0.78858	0.8	0	0	0	0000-00-00 00:00:00	2014-10-24 13:33:35	0.78867	9.13
78232759	2014-10-24 13:28:28	EURGBP	Buy	0.78858	0.2	0	0	0	0000-00-00 00:00:00	2014-10-24 13:29:10	0.78842	-4.06
78232725	2014-10-24 13:26:59	EURUSD	Sell	1.2646	1	0	0	0	0000-00-00 00:00:00	2014-10-24 13:28:07	1.26478	-14.23
78232679	2014-10-24 13:25:46	EURUSD	Sell	1.2646	1	0	0	0	0000-00-00 00:00:00	2014-10-24 13:26:05	1.26478	-14.23

Figura 46 - Ordini per l'utente Master_Account

5.2 Il Database

I database legato alla pagina web è stato realizzato con le seguenti finalità

- Registrazione degli utenti al sistema;
- Memorizzazione dell'operatività di ciascun utente:
- Cloning dell'operatività.

Il database consta di una tabella "User" atta alla memorizzazione dei dati personali e di accesso per ogni utente, e di una tabella che ne memorizza l'operatività, creata durante la stessa fase di registrazione al sistema.

La Figura 47 mostra un esempio di tali tabelle e i rispettivi campi.

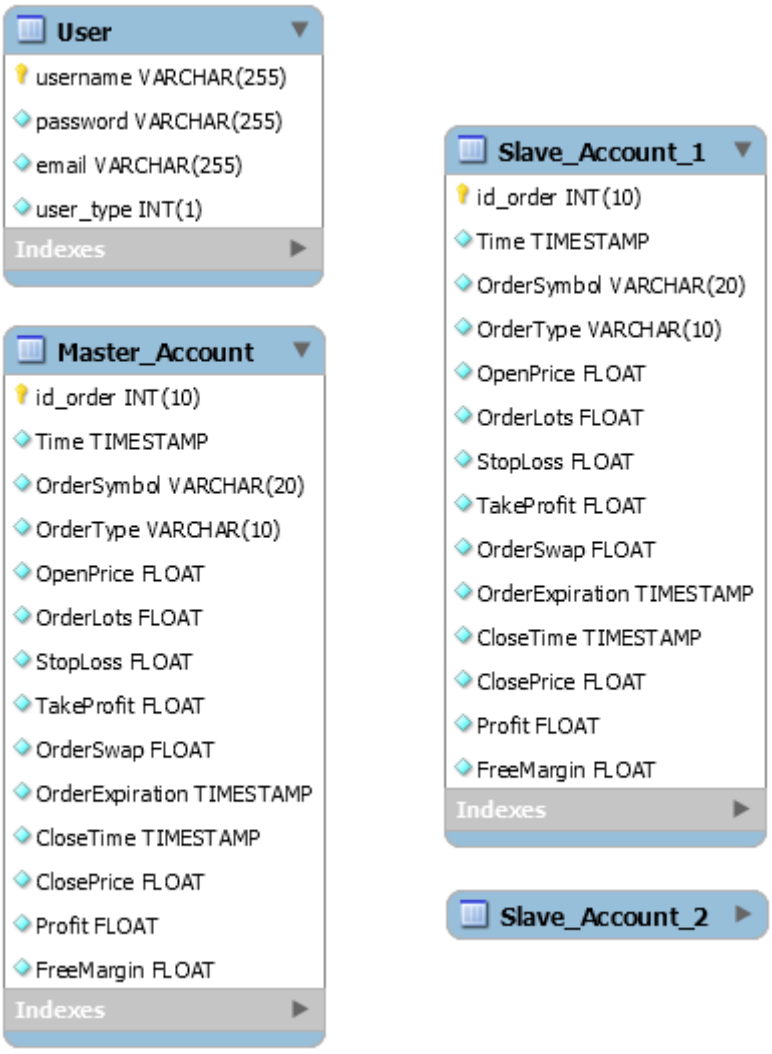


Figura 47 - Struttura e campi delle tabelle nel database.

Ovviamente gli attributi per le tabelle degli utenti sono indipendenti dalla tipologia dello stesso, si specifica però che gli utenti che con profilo Master, devono selezionare un profilo di rischio percentuale pari al 100% in fase di registrazione al sistema.

L'utente che desidera clonare l'operatività dovrà specificare il nome utente dell'operatore di riferimento. Il sistema utilizzerà quindi tale username per identificare all'interno del database la tabella dell'utente master.

Per poter interrogare il database da remoto utilizzando applicazioni esterne, attività non direttamente consentita per motivi di sicurezza dai web service provider, è stato necessario realizzare degli script in PHP di interfaccia tra il database e gli applicativi di cloning. Nello specifico, tali script permettono di interrogare il database, previo login allo stesso, ricevendo una query SQL inviata come web request utilizzando il metodo POST. Gli script estraggono la query dalla richiesta, la eseguono, e inviano il risultato alla sorgente (ovvero agli applicativi di gestione dei conti) come web response.

5.3 Sistema di cloning dell'operatività

Il sistema di cloning sviluppato, ovvero la parte di comunicazione con la piattaforma, consta come descritto nell'introduzione a questo capitolo, di due applicativi. Il primo, denominato *ForexDataSender*, permette di eseguire il login alla pagina web ed effettuare inserimento/aggiornamento degli ordini gestiti dall'utente dotato di account master. Il secondo, denominato *ForexOperativityCloning*, invece permette di clonare appunto l'operatività di un utente qualunque presente nel database.

I due applicativi si interfacciano alla piattaforma utilizzando dei file .csv, condivisi con i due Expert Advisor, realizzati appositamente per l'operatività. Difatti all'interno di una directory comune a piattaforma e software, il cui percorso va specificato in fase di inizializzazione degli applicativi e degli Expert Advisor, viene continuamente aggiornato uno o più file .csv il cui contenuto varia a seconda se ci stiamo riferendo ad un utente master o slave.

I due applicativi hanno in comune alcune classi, che verranno descritte brevemente nei paragrafi seguenti.

5.3.1 CsvFileReader.cs

La classe *CsvFileReader.cs* ha lo scopo di permettere l'interpretazione di file .csv. Gli applicativi utilizzano file .csv per effettuare l'update degli ordini in piattaforma e/o i record degli ordini memorizzati all'interno del database sul server. Una istanza della classe viene richiamata all'interno di un ciclo, tramite il quale è possibile leggere ed effettuare il parsing di ogni singola riga del file .csv.

Viene principalmente richiamato da un metodo che trasforma un file .csv in un Dictionary, ovvero una struttura ordinata di coppie chiavi-valore, utilizzato per indicizzare gli ordini a mercato, prendendo come riferimento il numero di ticket univoco rilasciato dal broker al momento dell'apertura dell'operazione.

5.3.2 DataUploader.cs

La classe *DataUploader.cs* è la classe che si occupa di effettuare l'interrogazione al database, tramite gli script PHP indicati precedentemente, per aggiornarne il contenuto. Nello specifico il metodo *queryDB* si occupa, costruendo ed inviando le richieste HTTP via metodo POST contenenti le varie query rivolte al database, di effettuare l'inserimento di nuovi record e l'aggiornamento di quelli già presenti.

5.3.3 HTTPResponseParser.cs

La classe *HTTPResponseParser.cs* è la classe atta a effettuare il parsing della risposta proveniente dalla pagina web, riferita alla query inviata dall'applicativo. Ovviamente la chiamata al metodo *ParseResponse* della classe avviene sempre dopo l'esecuzione di una query al database stesso. Riceve in input una stringa contenente l'intera risposta e, tramite un metodo di parsing appositamente modellato, ritorna un Dictionary contenente il risultato della query.

5.4 Gli applicativi di cloning dell'operatività

I due applicativi di cloning constano di una struttura simile, avendo in comune le classi precedentemente descritte. Nel dettaglio, il *ForexDataSender* prevede che all'avvio venga specificato il nome utente indicato in fase di registrazione al sistema con account master per l'aggiornamento delle informazioni ad esso riferite. È necessario inoltre indicare la directory tramite la quale si interfaccia tra la piattaforma di trading (ovvero la directory che conterà il file .csv degli ordini), e l'intervallo di aggiornamento delle informazioni. Una consolle infine mostra eventuali informazioni aggiuntive ed eventuali errori nel caso in cui non venga effettuato l'aggiornamento delle informazioni riguardanti le operazioni a mercato. La Figura 48 mostra l'applicativo in una sessione di log del trading.

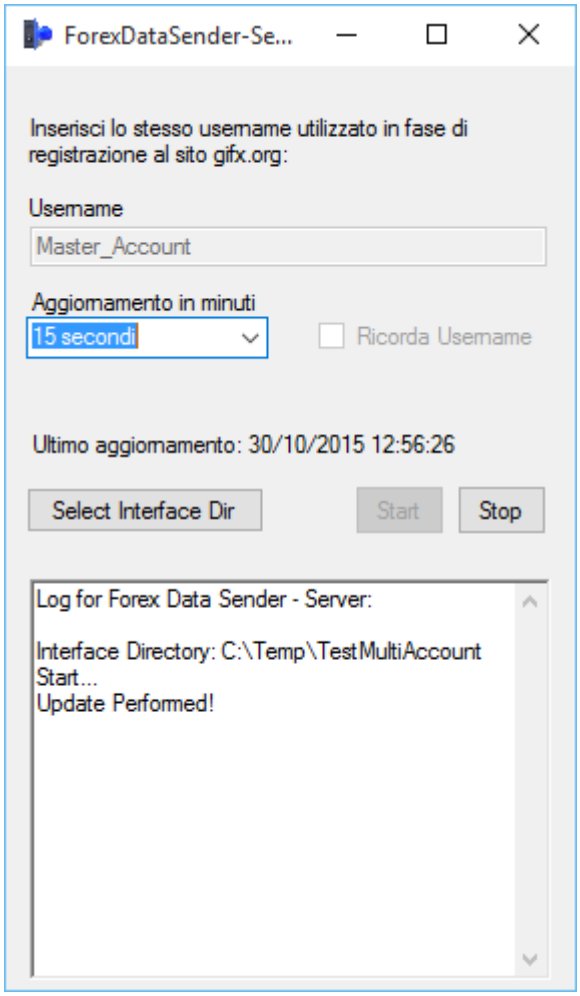


Figura 48 - Applicazione ForexDataSender per l'Account Master

Il secondo applicativo realizzato, ovvero il *ForexOperativityCloning*, è strutturato suddividendo l'interfaccia grafica in due schede. Nella prima è presente la sezione per l'accesso tramite l'immissione di nome utente e password, in maniera tale che si debba effettuare il login prima di poter accedere alla sezione di cloning dell'operatività (v. Figura 49).

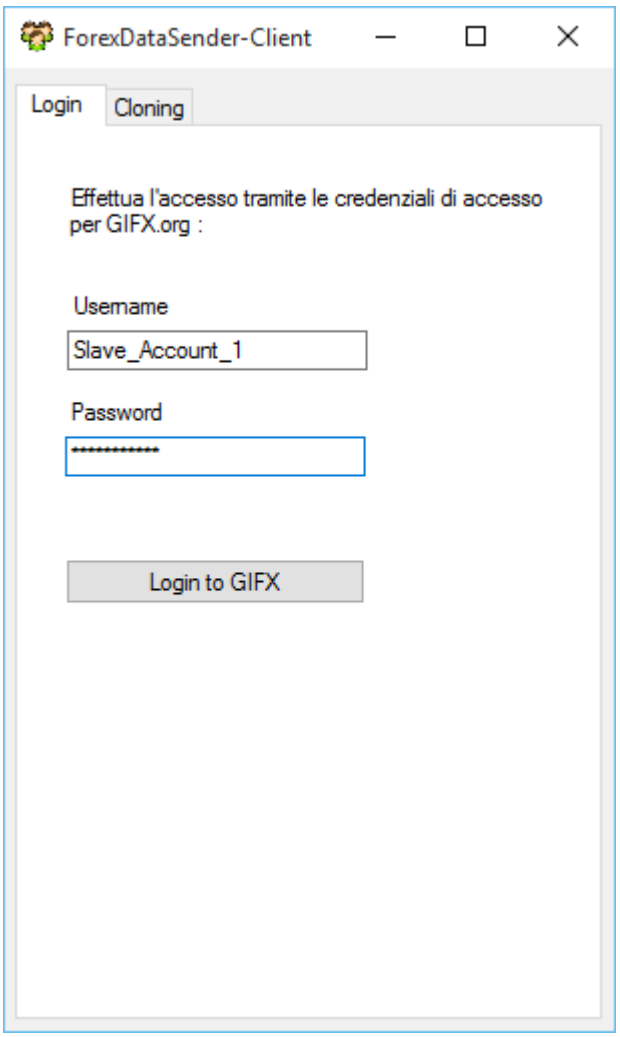


Figura 49 - Form di accesso al sistema.

Per ciò che concerne la seconda scheda, sono previsti gli stessi contenuti già descritti per l'applicativo *ForexDataSender*, con la specifica che nel campo *Username* va indicato l'identificativo dell'utente del quale si vuole clonare l'operatività (v. Figura 50).

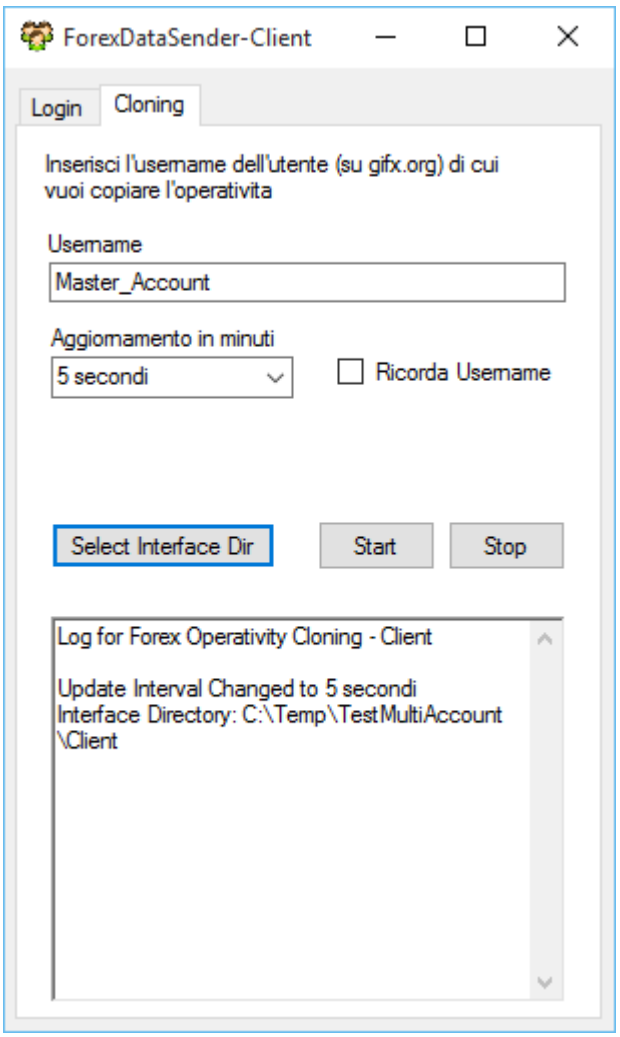


Figura 50 - Applicazione ForexOperativityCloning per l'account slave.

Per gestire l'operatività dell'utente slave, il client ad esso destinato costruisce ed aggiorna ogni qual volta viene inserito un ordine a mercato (replicato dall'account master), una LUT (*LookUp Table*), atta appunto a memorizzare una corrispondenza tra gli ID degli ordini dell'account master e quelli dell'account slave. In fase di inizializzazione inoltre viene applicato un filtro per escludere dalla lista gli ordini aperti precedentemente rispetto all'esecuzione corrente dell'applicativo.

Il software inoltre prevede che vengano gestiti tutti i casi previsti per un qualunque ordine a mercato, che sia questo un ordine semplice, pendente (ovvero un ordine messo in coda di esecuzione per una determinata data/ora), o ancora un ordine per il quale l'utente dotato di Account Master prevede una chiusura parziale (ovvero si prevede di ridurre il numero di lotti per quella posizione). Tali operazioni sono gestite tramite ulteriori file .csv presenti ed aggiornati nella directory di interfaccia.

Infine si descrivono di seguito, per ragioni di completezza dell'architettura di cloning dell'operatività, i due Expert Advisor atti alle funzionalità di log del trading dell'operatore detentore dell'account master e di cloning dell'operatività per conto dell'operatore con account slave.

L'Expert Advisor destinato a tenere traccia ed aggiornare le operazioni sulla pagina web per l'account master, denominato *OrderLogger.mql4*, opera su di un file .csv denominato *orderLog.csv*, al quale accede in lettura anche l'applicativo *ForexDataSender*.

Durante la fase di inizializzazione dell'Expert Advisor, viene effettuato il match tra gli ordini presenti a mercato e quelli contenuti sul file, al fine di mantenerne la sincronizzazione. Inoltre la gestione per

mezzo di un timer permette all'Expert Advisor di registrare sul file gli ordini che vengono di volta in volta posti in essere e di aggiornare quelli già esistenti, nei campi *profit* e *FreeMargin*, oltre che registrandone l'eventuale chiusura fissando definitivamente il valore di questi ultimi due campi.

Il secondo Expert Advisor, denominato *OpenClosePosition.mt4*, invece si occupa di gestire l'operatività sulla piattaforma di trading con account slave, acquisendo le informazioni sulle posizioni dalla pagina web, mediante l'applicativo *ForexOperativityCloning*. Anch'esso, così come il precedente Expert Advisor, si interfaccia all'applicativo mediante l'uso di file .csv. Nello specifico sfrutta la stessa LUT utilizzata dall'applicativo in C# per avere corrispondenza tra gli ordini dell'account master e quelli dell'account slave. In più ulteriori files specificano gli ordini da aprire/chiuder/modificare. Anch'esso, come il precedente, è provvisto di una procedura che, in fase di inizializzazione, verifica e mantiene la sincronizzazione relativamente ad ordini legati ad una precedente esecuzione.

6 Volatility Calendar

In questo capitolo si esaminano metodologie matematiche e tecnologie informatiche utilizzate per lo sviluppo dell'applicativo *Volatility Calendar.* In una prima analisi si descrive la suddivisione in settori del sorgente, andando poi a dettagliare le varie sezioni.

Il software è stato realizzato utilizzando l'ambiente di programmazione visuale LabView, della National Instruments, nello specifico è stata utilizzata la versione 2014 Professional Development System, Service Pack 1.

Inoltre a supporto del software, per la comunicazione con la piattaforma MetaTrader4, si è utilizzata la DLL di interfacciamento descritta nel Paragrafo 3.3.

I VI core della strategia Volatility Calendar sono stati implementati separatamente rispetto a quello principale. Tali VI, denominati *MOVA* e *MOAA*, sono stati convertiti in librerie LabVIEW (*MOVA.llb* e *MOAA.llb*), dai quali vengono richiamati. Si è reso necessario l'utilizzo di questa tecnologia per rendere modulare il progetto, e soprattutto per poter semplicemente duplicare i VI all'interno dei due progetti e richiamarli senza incorrere in conflitti ed errori di sovrascrittura dei dati.

Subito dopo il capitolo in cui si elenca la suddivisione in aree funzionali del progetto, prima delle specifiche implementative dell'applicativo, verrà descritto teoricamente il funzionamento dei due core MOVA e MOAA, con relativa specifica dei sub-vi utilizzati.

6.1 Modalità Calendar

Tale versione dell'applicativo, in presenza di una news da calendario relativa allo strumento finanziario in esame, prevede che vengano impostati dei parametri di apertura della posizione meno restrittivi, in maniera tale da accertarsi di riuscire ad entrare a mercato non appena la variazione di volatilità diventa significativa, e non soltanto quando questa è ormai nel pieno della sua evoluzione o addirittura quasi terminata, a causa di parametri troppo vincolanti, utilizzati per la normale attività della strategia.

A tal scopo è stato integrato l'applicativo *CalendarAlert*. Tale software, di cui sono disponibili i dettagli nel Paragrafo 4.2, non appena verificate le condizioni di "*attesa*" della news, ovvero non appena ci si trova nell'imminenza della stessa, inibisce l'apertura alla versione classica di Volatility, mantenendone attive solo le condizioni di uscita, e imposta i vincoli di ingresso in modo tale da permettere un ingresso tempestivo a mercato nella direzione corretta dettata dalla news. Difatti, con i parametri "*News*" impostati, l'applicativo permette una eventuale apertura di nuove posizioni solo in corrispondenza temporale della news.

Trascorso il lasso temporale previsto (configurabile da interfaccia grafica) in cui vanno utilizzati i suddetti parametri, l'applicativo automaticamente ricarica la configurazione standard e riprende la sua normale attività.

Lo schema di Figura 51 definisce l'operatività nell'intervallo temporale precedente e successivo ad una eventuale News. Nello specifico è possibile specificare quindi un range di ±T minuti dalla news, nel quale si attiva l'impostazione dei parametri come su descritto

S = Strategia Simple; **C**=Strategia Calendar

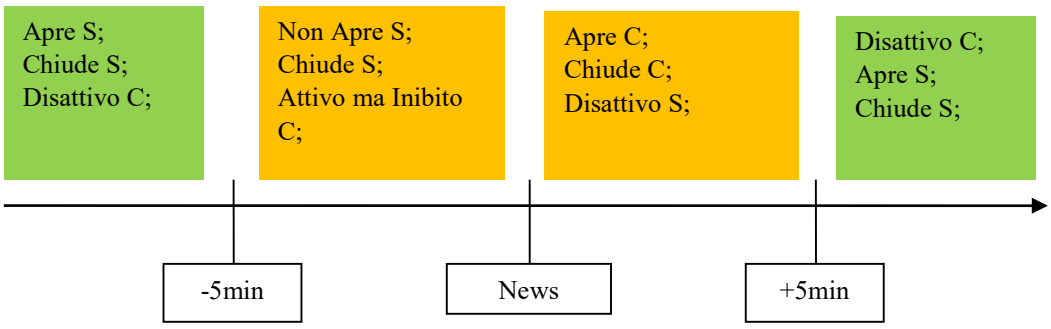


Figura 51 - Timing di ottimizzazione degli algoritmi in versione Calendar / No-Calendar

6.2 Suddivisione in Aree Funzionali

Il software è stato realizzato cercando di suddividere al meglio il codice in aree funzionali, al fine di renderne la lettura e l'interpretazione da parte di terzi il più semplice e veloce possibile, oltre che meglio manutenibile.

Di seguito si illustra uno schema concettuale di come tale suddivisione è stata realizzata (v. Figura 52).

Esaminiamo quindi le varie aree indicate in figura:

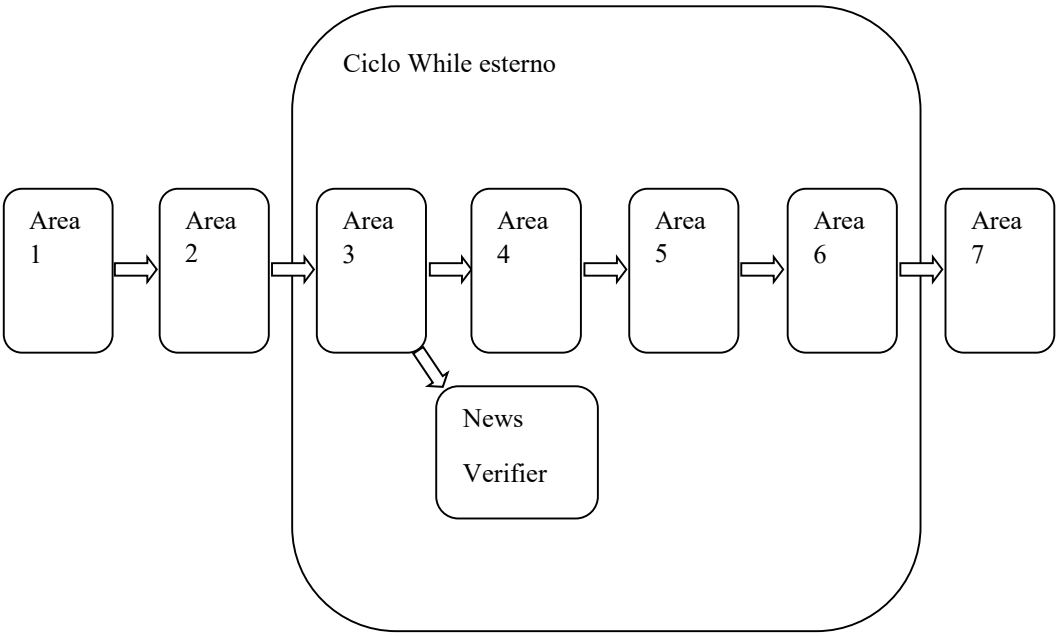


Figura 52 - Suddivisione in aree funzionali

Area 1: Porzione di codice per l'inizializzazione dell'interfaccia grafica, strutture dati necessarie al funzionamento del software, delle variabili globali e lettura di eventuali ordini rimasti aperti dall'esecuzione precedente.

Questa area si trova al di fuori del ciclo while di esecuzione indicato in figura, proprio perché riguarda delle operazioni di inizializzazione del software da eseguire una sola volta;

Area 2: Tale area si occupa di effettuare il buffering della struttura dati contenente i valori di prezzo ed i relativi Timestamp necessari per la prima iterazione del ciclo while;

Dopo aver superato queste due sezioni di codice, l'applicativo si evolve con un ciclo while che si occupa, oltre a mantenere attiva l'interfaccia grafica e quindi gestirne l'interazione con l'utente, ad ogni nuovo aggiornamento del livello di prezzo proveniente dalla piattaforma, di valutare secondo i criteri degli algoritmi di apertura e chiusura delle posizioni se richiedere appunto l'apertura di una nuova posizione (con il relativo dimensionamento) o la chiusura di una o più posizioni aperte. Inoltre, in presenza della news, vengono sostituiti i parametri di apertura classici con quelli ottimizzati per aperture rapide.

Area 3: Algoritmi di money management e verifica dei criteri di apertura della posizione; da tale area si dirama il nodo *News Verifier*, modulo di verifica della presenza di una imminente news di carattere macro economico tramite l'applicativo CalendarAlert ed in tal caso impostazione dei vincoli di ingresso a mercato relativi in presenza di notizie macro economiche.

Area 4: Lettura delle informazioni provenienti dalla piattaforma relativamente all'eventuale nuovo ordine aperto e la memorizzazione dello stesso nel cluster degli ordini;

Area 5: Strategie di *EXIST* (EXIt STrategy) per la chiusura delle posizioni;

Area 6: Backup degli ordini aperti su file per eventuale recupero alla successiva esecuzione;

Dopo l'Area 6, se non avviene l'interruzione dell'esecuzione da parte dell'utente, il ciclo while itera nuovamente le operazioni brevemente descritte nelle Aree 3, 4, 5, 6. Nel caso invece di una interruzione dell'esecuzione, viene dapprima eseguito il backup dei file di Log sui quali gli algoritmi di apertura della posizione hanno registrato i dati sui quali sono state valutate le condizioni di apertura di nuove posizioni.

Area 7: Memorizzazione dei file di Log degli algoritmi.

6.3 VI di uso comune o di supporto

Alcuni vi svolgono funzionalità generiche, non specifiche per un singolo obiettivo. Al fine di evitare ripetizioni, si descrivono in questa sezione tali vi, così da poterli menzionare senza descrizioni ripetute nelle pagine successive.

- *readfromDLL*: Tale VI si interfaccia alla DLL realizzata per la comunicazione tra applicativo e Piattaforma MT4. Riceve in input il percorso assoluto alla DLL, ed il suo scopo è quello di leggere e restituire i valori correnti di prezzo e relativo istante temporale scritti nella sezione shared di memoria della DLL dall'Expert Advisor in piattaforma.
- *ConvertStringToTimeStamp*: questo VI riceve in input un Timestamp sotto forma di stringa di testo nel formato *YYYY.MM.DD HH:MM:SS* e lo converte in un oggetto Timestamp di LabView.
- *DateRecToTimeStamp*: riceve in input un oggetto di tipo Time Rec (ovvero un cluster di dati temporali), e restituisce in output un oggetto strutturato Timestamp di LabVIEW, incluse le frazioni di secondo

- *ConvertStringToTimeStamp_msec*: questo VI riceve in input un Timestamp sotto forma di stringa di testo nel formato *YYYY.MM.DD HH:MM:SS:UUU* e lo converte in un oggetto Timestamp di LabView, riportando quindi in output anche i decimi di secondo.
- *DateRecToTimeStamp_msec*: riceve in input un oggetto di tipo Time Rec (ovvero un cluster di dati temporali), e restituisce in output un oggetto strutturato Timestamp di LabVIEW, incluse le frazioni di secondo.
- *ShiftArrayElement*: questo VI riceve in input un array di double ed un nuovo valore. Esegue un inserimento in posizione N-esima rispetto alla dimensione dell'array, eliminando quindi l'elemento in posizione zero (il più vecchio in termini temporali).
- *ConvertPriceToInt*: Riceve in input un valore di prezzo ed il numero di cifre significative dello strumento, fornisce in output il valore in PIPs dello strumento. Esistono due versioni di questo VI in quanto uno è utilizzato per il calcolo della velocità ed un altro per l'accelerazione.
- *GetMeanVelAcc*: questo VI riceve in input il buffer contenente gli ultimi 10 valori di prezzo, il numero di cifre significative dello strumento finanziario in esame e il numero di elementi su cui calcolare la media. Restituisce il valor medio di velocità ed accelerazione in PIP e decimi di PIP dello strumento calcolato appunto sul numero di elementi indicato in input.
- *RemoveDuplicateFromArray*: questo VI riceve in input un array di valori numerici, restituendo l'array dai quali sono stati rimossi i valori duplicati
- *GetNextDay*: questo vi riceve in input un oggetto di tipo Timestamp, destruttura il cluster ad esso relativo e incrementa il campo "*DayOfMonth*". Ritorna il Timestamp di origine aggiornato all'1 a.m. del giorno successivo. Utilizzato per ottenere il Timestamp del prossimo aggiornamento del file delle news.
- *GetUsefulTS*: questo vi riceve in input un oggetto di tipo Timestamp (nello specifico quello proveniente dal VI *GetNextDay*), e verifica se il giorno cui il Timestamp si riferisce è un festivo (ovvero sabato o domenica). In questi casi il vi porta il Timestamp al primo lunedì utile.
- *GetParameterFromFile*: Questo vi si occupa di leggere i file (con estensione *.def*) contenenti le configurazioni dei parametri di apertura da utilizzare in modalità *Simple* o in modalità *News*. Ogni record corrisponde ad uno strumento finanziario, ed i campi sono separati da virgola. Il separatore dei decimali è il "punto".

6.4 Core Engine MOVA & MOAA

In questa sezione, come detto nel capitolo introduttivo, verrà descritta la modellazione e progettazione del core della strategia. Questi due algoritmi si basano sull'analisi della velocità e dell'accelerazione dello strumento finanziario al fine di stabilire se vi è una variazione di volatilità significativa, tale da rendere l'apertura di una eventuale posizione a mercato profittevole, stabilendo il momento più opportuno. L'applicativo permette di poter scegliere se valutare l'apertura di una nuova posizione utilizzando la sola strategia su variazione significativa di velocità, la sola strategia su variazione significativa di

accelerazione, oppure le due strategie in OR oppure, quarta possibilità, in AND. La scelta di una delle quattro combinazioni degli algoritmi ovviamente condiziona i tempi di ingresso a mercato e la correttezza delle posizioni aperte.

Di seguito si descrive l'algoritmo MOVA di analisi delle variazioni di velocità. L'algoritmo MOAA verifica le stesse condizioni per l'apertura di nuove posizioni, basandosi però sulle variazioni di accelerazione.

6.4.1 MOVA & MOAA Engine

L'algoritmo riceve in input il buffer contenente gli ultimi dieci valori di prezzo "tick" provenienti dalla piattaforma. Alla presenza di un nuovo valore di prezzo, l'algoritmo calcola le ultime X variazioni di velocità dello strumento (in PIPs) nei corrispondenti istanti di tempo.

Considerati i valori di prezzo P, istante per istante, si avrà:

$$v(t) = P(t) - P(t-1)$$

Mentre l'accelerazione, espressa come differenza tra le velocità al tempo *t* ed al tempo *t-1*, ha la forma

$$a(t) = P(t) - 2P(t-1) + P(t-2)$$

Tali valori vengono utilizzati per costruire la retta dei minimi quadrati che meglio approssima tali dati. L'apertura di una nuova posizione è legata alla verifica di diversi fattori:

a) Lo strumento non deve aver compiuto, nella giornata odierna, una variazione di prezzo in PIP pari alla media calcolata su 100 giorni più 3 volte la sua deviazione standard;

$$CurrentPIPMeanDay \leq \sum_{i=1}^{100} \left| Max_{daily}[i] - Min_{daily}[i] \right| + 3\,\sigma$$

b) Il modulo dell'ultima variazione di prezzo, in PIPs, deve essere maggiore di una quantità minima, soglia sotto la quale è inibito l'ingresso; in formule:

$$\left| \sum PIP \right| > PIP\_Limit$$

c) Inoltre, in AND/OR con la b), deve essere verificato che, l'ultima variazione in PIPs, misurata considerando l'oscillazione del prezzo con segno, deve essere maggiore di una frazione della cumulata dei PIPs compiuti dallo strumento; in formule

$$\left| \sum PIP \right| > \frac{1}{\alpha} \sum |PIP|$$

d) Devono esserci a mercato meno di cinque operazioni aperte della strategia;

e) Devono essere trascorsi dall'apertura dell'ultima posizione almeno *T* secondi per poter entrare nuovamente a mercato, al fine di limitare il numero di operazioni da aprire sullo stesso movimento;

f) La retta dei minimi quadrati, appena costruita deve trovarsi all'interno dell'area di interesse indicata in Figura 1 (area positiva se la variazione di velocità è positiva, area negativa se la variazione di velocità è negativa).

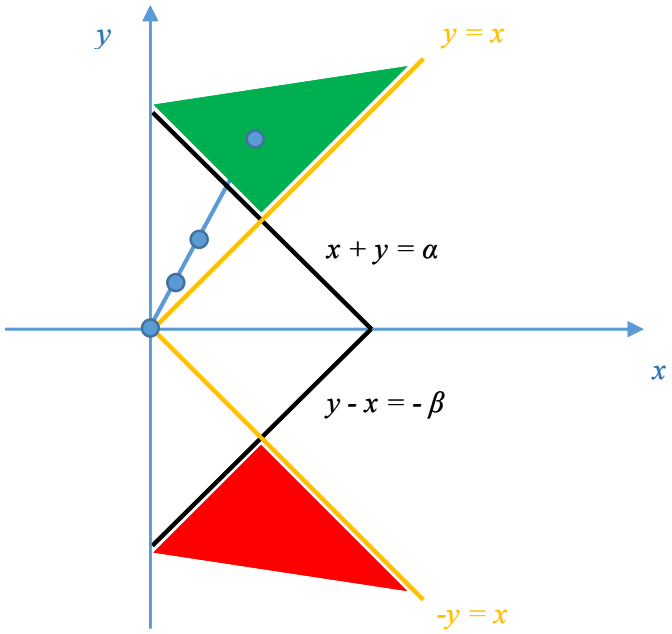


Figura 53 - Aree di interesse per la valutazione dell'apertura di una nuova posizione

Nel dettaglio, la Figura 53 mostra due aree che risultano essere quelle per cui l'algoritmo ritiene opportuno l'ingresso a mercato, sempre che siano verificati i vincoli ulteriori sopra indicati. Nello specifico, se vi è un aumento della velocità (o accelerazione nel caso di MOAA), si deve verificare, per l'ultimo punto della retta dei minimi quadrati, che:

$$\begin{cases} x + y > \alpha \\ y > x \end{cases}$$

Inoltre

$$SLOPE > 1$$

Ovvero la retta dei minimi quadrati dovrà essere all'interno dell'area verde.

Nel caso invece in cui la variazione di velocità (o accelerazione nel caso di MOAA) sia negativa, si deve verificare, per l'ultimo punto della retta dei minimi quadrati, che:

$$\begin{cases} y - x < -\beta \\ y < -x \end{cases}$$

Inoltre

$$SLOPE < -1$$

Ovvero la retta dei minimi quadrati dovrà essere all'interno dell'area rossa.

La motivazione per cui si è scelto di considerare solo i movimenti per i quali l'ultimo punto della retta dei minimi quadrati deve trovarsi all'interno delle aree su indicate deriva dalla relazione che intercorre tra variazione di prezzo per ogni aggiornamento ricevuto e tempo che intercorre tra una variazione e l'altra. Infatti minore sarà il tempo che intercorre tra due singole variazioni di prezzo e tanto saranno intense (in termini di PIPs) tali variazioni, maggiore sarà l'inclinazione della retta approssimante il movimento. Inoltre più la variazione è intensa, maggiore sarà la distanza tra i punti rappresentati dalla retta interpolatrice, portando così l'ultimo punto all'interno dell'area di interesse del grafico. Tale rappresentazione, supportata dai vincoli indicati nei punti precedenti, mostrano una variazione significativa di

volatilità, e possono indicare l'inizio di un possibile movimento sfruttabile in termini di profitto. La direzione di ingresso a mercato, ovvero la scelta di entrare Buy o Sell, deriva dal valore assunto dalla pendenza della retta approssimante la curva di velocità e/o accelerazione. Specificando meglio il concetto, è la pendenza a guidare la direzione e, nel caso di utilizzo di entrambi gli algoritmi MOVA a MOAA, se entrambi attivi si preferisce l'indicazione fornita dalla pendenza dell'algoritmo basato sulle accelerazioni.

Di seguito si discuteranno i VI costituenti le due strategie MOVA e MOAA.

6.4.1.1 *MOVA*

L'algoritmo *MOVA* è stato integrato all'interno di Volatility sotto forma di progetto compilato (LLB). Esso si occupa di effettuare l'analisi di tutti i vincoli descritti precedentemente, e in caso di esito positivo crea il file per l'apertura di un nuovo ordine.

Di seguito si dettaglia l'elenco degli input / output di cui è possibile vedere lo schema in Figura 54:

- *PathDLL*: percorso assoluto alla DLL di interfaccia;
- *Input Array TS*: Buffer contenente gli ultimi 10 Timestamp;
- *Input Array Tick*: Buffer contenente gli ultimi 10 livelli di prezzo tick;
- *Digit*: numero di cifre significative dello strumento finanziario in esame;
- *#tick*: indica il numero di punti su cui costruire la retta dei minimi quadrati;
- *x+y>?*: limite superiore che verifica la condizione del punto f);
- *y-x<?*: limite inferiore che verifica la condizione del punto f);
- *PIP Control OR/AND*: controllo per la selezione della modalità di verifica delle condizioni b) e c);
- *MaxPos*: indica il numero massimo di posizioni che è possibile aprire;
- *Percent*: valore del parametro α;
- *SecondToWait*: tempo di attesa tra l'apertura di due operazioni;
- *PIP Limit*: soglia in PIPs misurata a partire dalla verifica dell'evento, al di sotto della quale l'apertura non è permessa.
- *OpenByNews*: booleano per l'inibizione temporanea delle aperture in concomitanza dell'avvento di una news finanziaria.

Restituisce in output:

- *MOVA output array TS*: il buffer dei Timestamp in cui è stato inserito il nuovo elemento;
- *MOVA output array Tick*: il buffer dei valori di prezzo in cui è stato inserito il nuovo elemento;
- *Update*: Booleano che indica l'avvenuto aggiornamento dei buffer;
- *OpenPosMOVA*: Booleano TRUE/FALSE che indica se tutte le condizioni sono state verificate ed è quindi possibile entrare a mercato;
- *Vel_Event*: indica il valore corrente di Velocità dello strumento finanziario. Congiuntamente all'apertura di una nuova posizione, tale valore di velocità viene memorizzato nel relativo cluster del nuovo ordine.

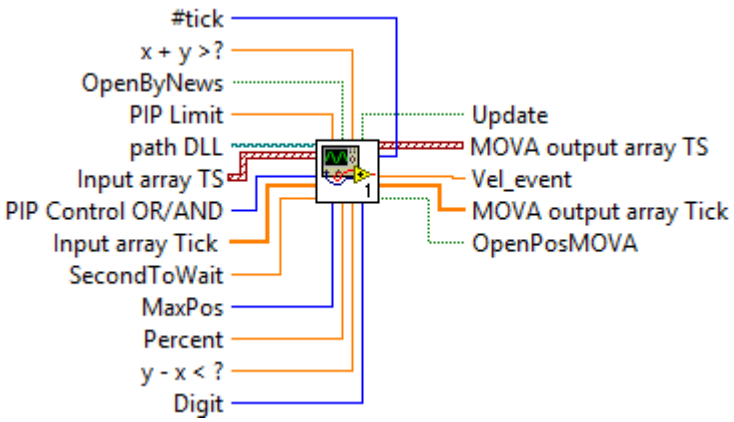


Figura 54 - Input / Output del VI MOVA

Il VI è stato sviluppato tenendo in considerazione che il calcolo della retta dei minimi quadrati costruita sui punti derivanti dal calcolo della velocità, e di conseguenza la verifica dei successivi vincoli propedeutici all'ingresso, deve avvenire solamente quando si riceve un nuovo aggiornamento del livello di prezzo da broker. A tal fine viene dapprima verificato se il prezzo memorizzato nella variabile in memoria condivisa della DLL (tramite il VI *readfromDLL*) è differente dall'ultimo memorizzato nel buffer, facendo riferimento al suo Timestamp. In caso di esito positivo, il nuovo valore viene inserito in cima

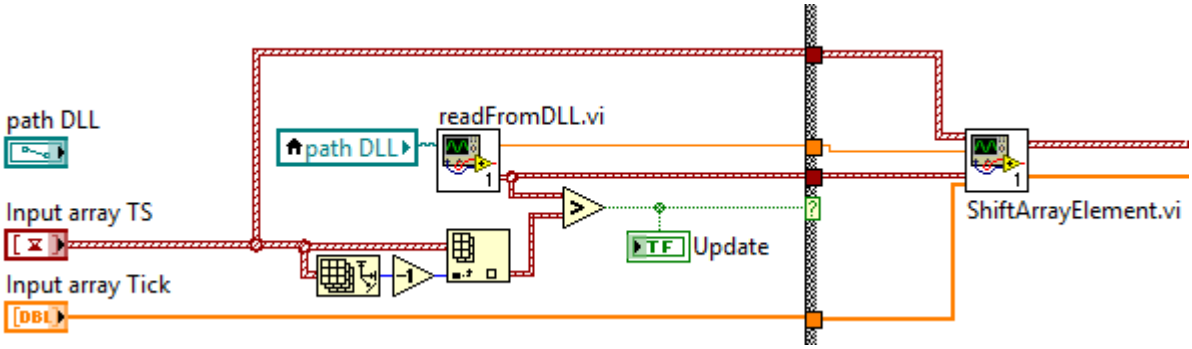


Figura 55 - Verifica della presenza di un nuovo elemento ed eventuale inserimento nel buffer

al buffer dei prezzi, utilizzando il VI *ShiftArrayElement,* comportando così la cancellazione dell'elemento più vecchio (v. Figura 55).

Si genera quindi il vettore contenente i valori di velocità necessari ed i corrispondenti delta temporali tra un livello di prezzo e l'altro, calcolati come differenza dei Timestamp.

I due vettori appena creati vengono quindi posti in input al VI che computa la retta dei minimi quadrati e la pendenza della stessa.

Il vettore dei punti viene dato quindi in pasto al VI che verifica le disequazioni indicate nel paragrafo 4.1, restituendone un valore true/false (v Figura 56).

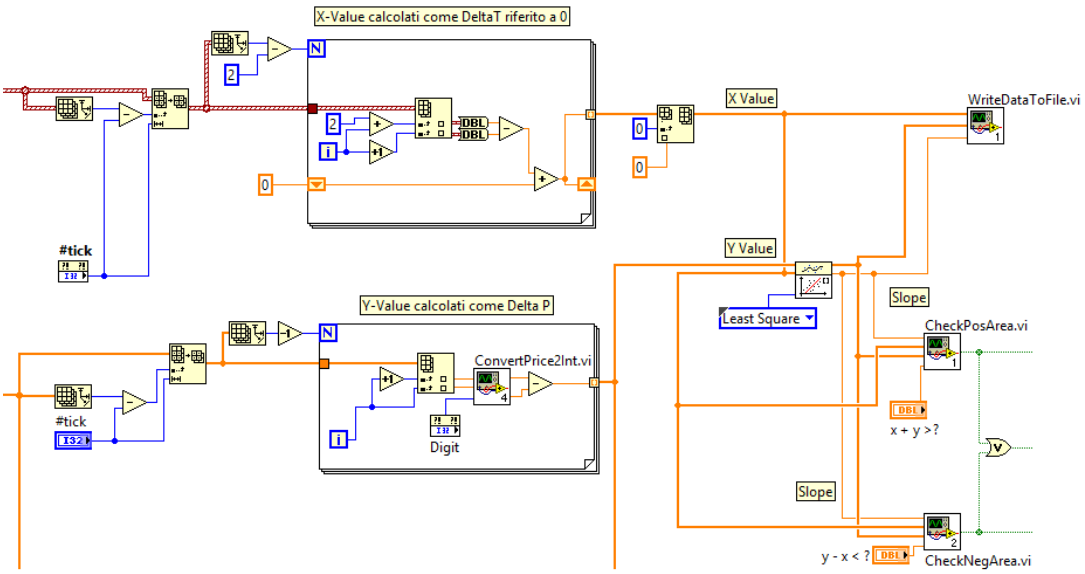


Figura 56 - Computazione dei punti per costruzione della retta dei minimi quadrati e verifica dei vincoli.

MOVA utilizza degli altri VI specializzati ad eseguire operazioni specifiche:

- *CheckPosArea*: Questo VI si occupa di verificare che la retta dei minimi quadrati ed il suo ultimo punto si trovino all'intero dell'area evidenziata in verde della Figura 1. Riceve in input la slope, gli array contenenti le X e le Y della retta e il parametro α indicato nel paragrafo 6.4.1. Restituisce TRUE se le condizioni sono verificate, FALSE altrimenti;
- *CheckNegArea*: Questo vi riprende i concetti del precedente, verificando però le disequazioni relative all'area negativa della Figura 2;
- *WriteDataToFile*: Questo VI, ad ogni aggiornamento ottenuto dalla piattaforma, riceve in input i valori X e Y della retta dei minimi quadrati costruita sugli ultimi valori di prezzo e la pendenza ad essa riferita, e scrive tali valori su di un file di log;

Verificato che la retta si trovi all'interno di una delle due aree di interesse, l'algoritmo memorizza in quale area si trova la retta dei minimi quadrati e si predispone a verificare le ulteriori condizioni precedentemente specificate (v. Figura 57 e Figura 58).

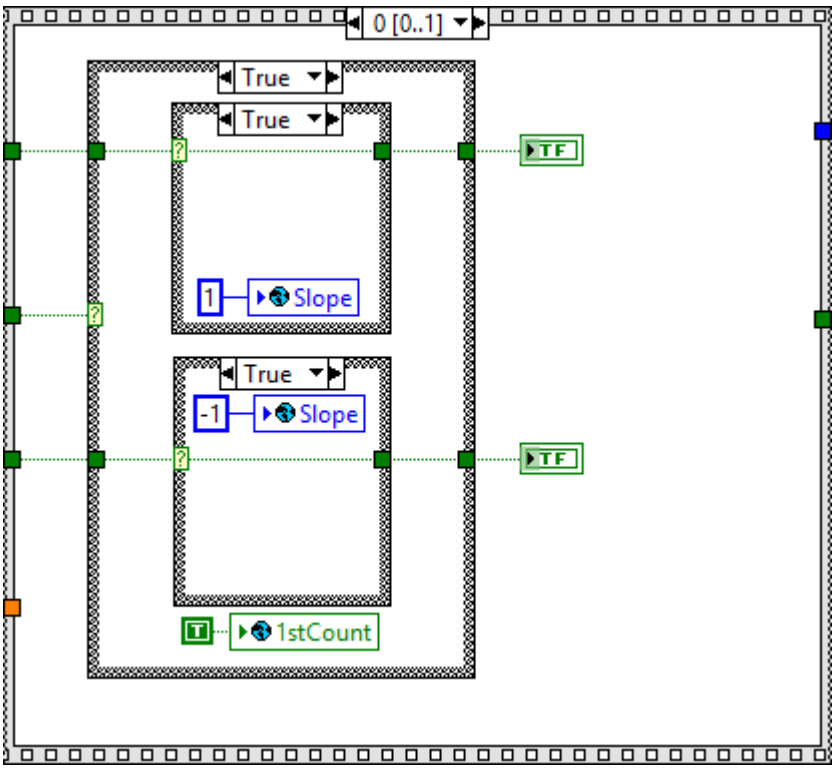


Figura 57 - Memorizzazione della direzione

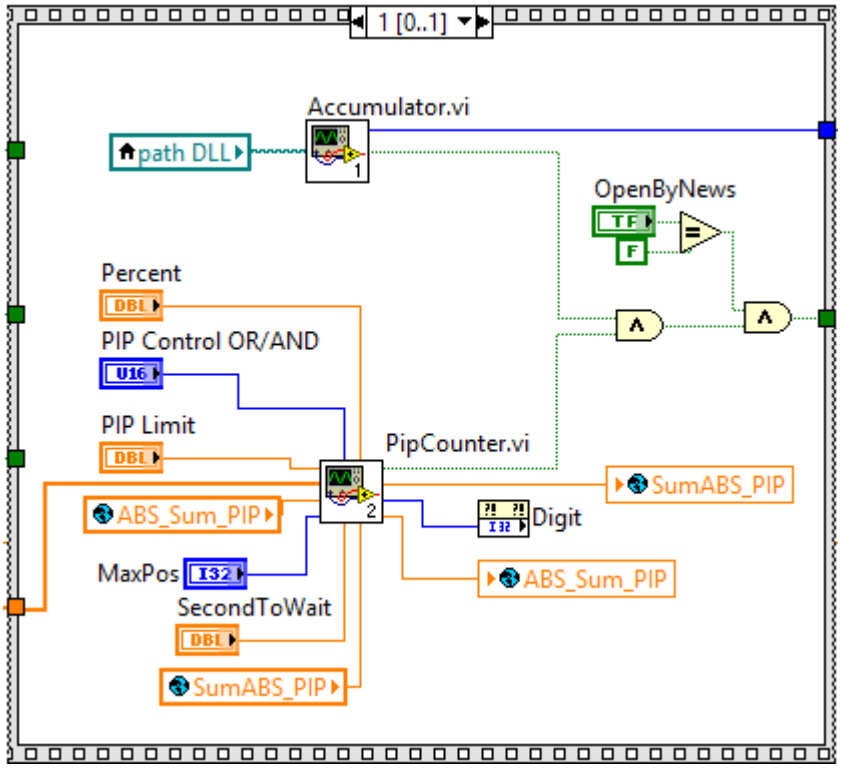


Figura 58 - Verifica degli ulteriori vincoli propedeutici all'ingresso.

Come si evince dalla Figura 58, vengono richiamati due ulteriori VI:

- *Accumulator*: Tale VI riceve in input il percorso alla DLL di interfaccia tra la piattaforma e LabVIEW, dalla quale legge il valore corrente di PIPs compiuti dallo strumento nella giornata odierna e la Deviazione Standard relativa. In base a tale valore e rispetto alla media giornaliera su 100 giorni, imposta lo Stop Loss Statico per l'ordine da aprire. I valori possibili di S/L statico sono i seguenti:
    - 20 PIPs, se CurrentPIPDay < MeanDay(100) + 1σ*MeanDay;
    - 15 PIPs, se CurrentPIPDay < MeanDay(100) + 2σ*MeanDay;
    - 10 PIPs, se CurrentPIPDay < MeanDay(100) + 3σ*MeanDay.

  Nel caso in cui nessuna delle tre disequazioni appena indicate è verificata, è evidente che lo strumento ha già compiuto tutto il movimento giornaliero, superando inoltre il valore medio su 100 giorni. Tale condizione inibisce l'apertura di nuove posizioni e pone in output (oltre ad un valore di S/L Statico negativo), un booleano indicante l'individuazione di un nuovo massimo nel periodo. In Figura 59 si mostra lo schema di input / output mentre, in Figura 60 parte del block diagram dell'implementazione.

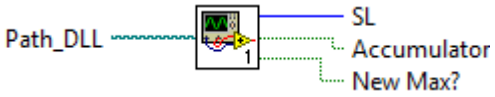


Figura 59 - Input / output del VI Accumulator

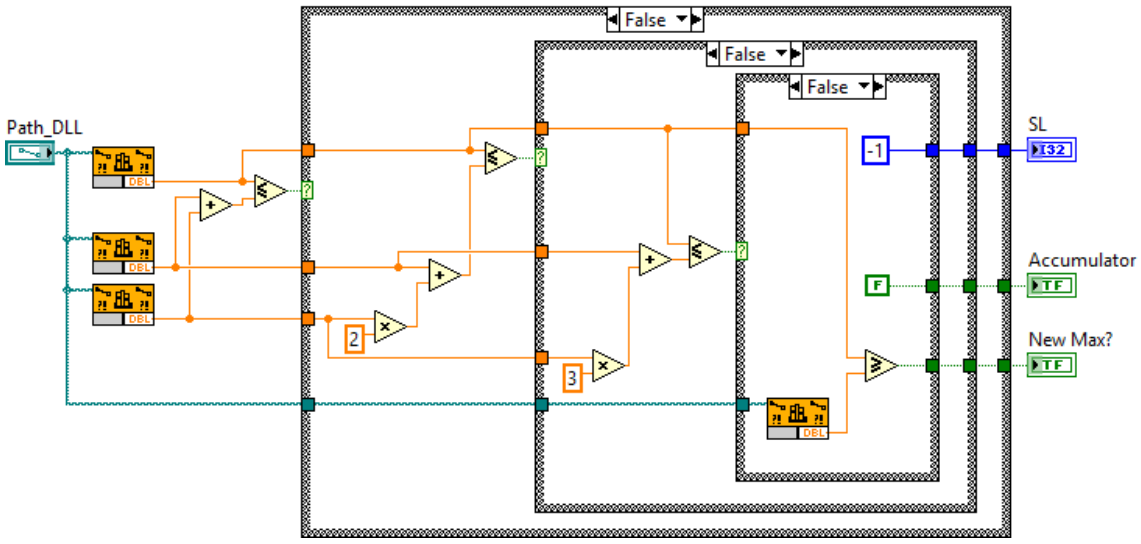


Figura 60 - Parte del Block diagram del vi Accumuator

- *PIPCounter*: Questo VI si occupa di verificare, per l'apertura di una nuova posizione, le condizioni indicate nei punti b), c), d), ed e) indicate nel paragrafo 6.4.1. Se tali condizioni, rispetto ai parametri impostati dall'utente, sono tutte verificate, permette l'apertura di una nuova posizione. La Figura 61 mostra gli input del suddetto VI:

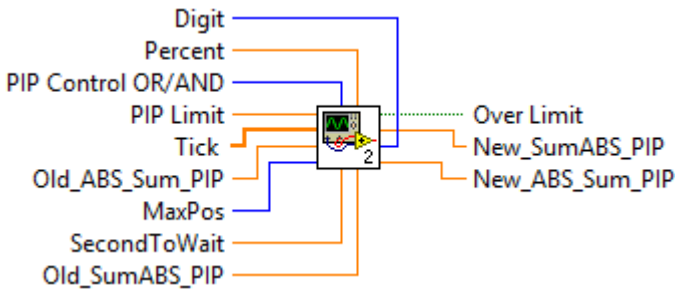


Figura 61 - Input / output per il VI PIPCounter.

Richiama:

- *TimeElapsed*: questo VI riceve in input il Timestamp relativo all'ultima apertura di una posizione e il numero di secondi da attendere prima di permettere l'apertura di una nuova posizione. Effettua il confronto tra il Timestamp in input, cui somma i secondi di delay tra una operazione e l'altra, e l'orario corrente. Ritorna un valore booleano TRUE/FALSE risultato della verifica di tale condizione.

A supporto dell'applicativo vi è una variabile globale utilizzata per memorizzare informazioni di uso comune all'interno dell'applicativo.

- *MOVA_Global*: Ha al suo interno i seguenti controlli:
  - *Price:* memorizza il valore del prezzo corrente;
  - *Slope:* memorizza il valore attuale della Slope;
  - *OldSlope:* memorizza il valore calcolato all'iterazione precedente della slope;
  - *SymbolName:* memorizza il nome dello strumento finanziario in esame:
  - *#Pos:* memorizza il numero di posizioni aperte fino all'iterazione attuale;
  - *LastOpen:* Memorizza il Timestamp relativo all'ultima operazione aperta;
  - *S/L_Statico:* Valore corrente di S/L statico da assegnare ad un ordine successivo;
  - *FirstCount:* Booleano per indicare che è la prima iterazione in cui si contano i PIPs
  - *ABS_Sum_PIP:* Memorizza il valore assoluto della sommatoria dei PIP calcolata fino a quell'istante;
  - *SumABS_PIP:* Memorizza la sommatoria del valore assoluto dei PIP calcolata fino a quell'istante.

Gli ultimi due contatori (*ABS_Sum_PIP* e *SumABS_PIP*), vengono azzerati e inizializzati al nuovo valore quando la pendenza (confrontata rispetto all'iterazione precedente) risulta opposta, il che indica il passaggio della retta dei minimi quadrati da un'area di interesse all'altra della Figura 53 (esempio di finta di corpo in casi di variazione di volatilità significative).

6.4.1.2 *MOAA*

Come detto, la struttura del VI che verifica le condizioni di apertura sull'accelerazione rispecchia esattamente quella del VI MOVA, per questo motivo si riportano solo gli input e gli output del VI (v. Figura 62). In questo senso, eredita la struttura implementativa concettuale, le variabili globali, input e output.

MOAA è stato integrato all'interno di Volatility sotto forma di progetto compilato (LLB).

Riceve in input i seguenti controlli:

- *PathDLL*: percorso assoluto alla DLL di interfaccia;
- *Input Array TS*: Buffer contenente gli ultimi 10 Timestamp;
- *Input Array Tick*: Buffer contenente gli ultimi 10 livelli di prezzo tick;
- *Digit*: numero di cifre significative dello strumento finanziario in esame;
- *#tick*: indica il numero di punti su cui costruire la retta dei minimi quadrati;
- *x+y>?*: limite superiore che verifica la condizione del punto f);
- *y-x<?*: limite inferiore che verifica la condizione del punto f);
- *PIP Control OR/AND*: controllo per la selezione della verifica delle condizioni b) e c);
- *MaxPos*: indica il numero massimo di posizioni che è possibile avere a mercato contemporaneamente;
- *Percent*: valore del parametro α;
- *SecondToWait*: tempo di attesa tra l'apertura di due operazioni;
- *PIP Limit*: soglia in PIPs misurata a partire dalla verifica dell'evento, al di sotto della quale l'apertura non è permessa;
- *OpenByNews*: booleano per l'inibizione temporanea delle aperture in concomitanza dell'avvento di una news finanziaria.

Restituisce in output:

- *MOAA output array TS*: il buffer dei Timestamp in cui è stato inserito il nuovo elemento;
- *MOAA output array Tick*: il buffer dei valori di prezzo in cui è stato inserito il nuovo elemento;
- *Update*: Booleano che indica l'avvenuto aggiornamento dei buffer di prezzo e Timestamp;
- *OpenPosMOAA*: Booleano TRUE/FALSE che indica se tutte le condizioni sono state verificate ed è quindi possibile entrare a mercato.
- *Acc_Event*: indica il valore corrente di Accelerazione dello strumento finanziario. Congiuntamente all'apertura di una nuova posizione, tale valore di velocità viene memorizzato nel relativo cluster del nuovo ordine.

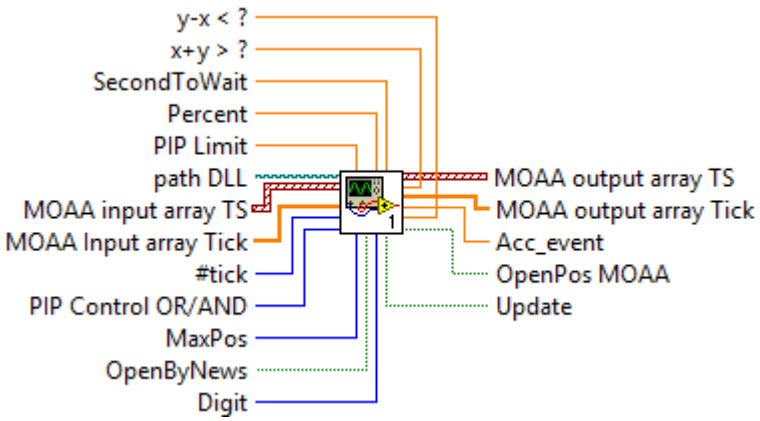


Figura 62 - Input / output del VI MOAA

6.5 Variabili Globali e Strutture dati principali

Prima di intraprendere la descrizione delle varie aree, si preferisce definire le strutture dati utilizzate e i flag di controllo, (sia contenuti all'interno di variabili globali che localmente all'interno del vi principale), necessarie al funzionamento del software. L'utilizzo di una variabile globale permette di rendere controlli, indicatori e strutture dati accessibili, in lettura e scrittura, da qualunque VI richiami la variabile stessa, evitando di dover porre in input al VI un innumerevole numero di oggetti.

6.5.1 Global_Volatility

Tale variabile globale, relativa ai controlli contenuti all'interno del vi principale, contiene:

- *Curr_V*: controllo per lo storage del valore corrente di velocità del cross in esame;
- *Curr_A*: controllo per lo storage del valore corrente di accelerazione del cross in esame;
- *Tick_Cnt*: contatore del numero di tick giunti all'applicativo trascorsi i quali verificare gli stop-loss per le operazioni aperte;
- *New_Order*: variabile booleana indicante la presenza della richiesta di una nuova posizione da parte degli algoritmi di valutazione delle condizioni di apertura. Tale booleano rimane settato a TRUE fintanto che la piattaforma MT4 non registra l'ordine, e indica che le informazioni sono state acquisite ed è quindi possibile valutare nuove condizioni per l'apertura di posizioni;
- *CF_Pos*: indica il valore del cut-off positivo registrato fino a quell'istante temporale dall'algoritmo di calcolo automatico del cut-off;
- *CF_Neg*: indica il valore del cut-off negativo registrato fino a quell'istante temporale dall'algoritmo di calcolo automatico del cut-off;
- *Time-TVE*: contiene l'istante temporale corrente per l'algoritmo Tick-Volatility-Engine;
- *NewsParam*: Indica che sono stati settati i parametri di apertura previsti per tale modalità;
- *IncomingNews:* Indica che ci si trova nell'intervallo "–*T*" minuti dalla news;
- *OpenByNews:* Disattiva l'operatività dell'applicativo in versione semplice nei *T* minuti prima della news;
- *TimeNewsUpdate:* Oggetto di tipo Timestamp contenente data e ora del prossimo aggiornamento dei file delle news;
- *NextNews_TS:* Oggetto di tipo Timestamp contenente data e ora della prossima news;
- *NewsData:* Cluster contenente i dati delle news future.

6.5.2 Order Cluster Array

Tale Array di cluster è utilizzato all'interno dell'applicativo per memorizzare gli ordini, che siano questi aperti o chiusi. Ad ogni iterazione dell'algoritmo il suo contenuto viene letto ed eventualmente modificato dalle strategie di uscita per l'eventuale chiusura delle posizioni.

La struttura del singolo cluster di ordini è la seguente:

- *Ticket*: campo numerico contenente l'ID dell'operazione all'interno della piattaforma
- *Type*: campo numerico relativo alla tipologia di ordine,
    - 0 = Buy Operation;
    - 1 = Sell Operation.

- *Status*: campo numerico utilizzato in fase di acquisizione dei dati dalla piattaforma, pari ad 1 quando l'ordine è ancora in fase di apertura, 0 quando completo;
- *OpenPrice*: campo numerico relativo al prezzo di apertura della posizione, ottenuto dalla piattaforma dopo che l'ordine è stato inviato;
- *S/L Static*: campo numerico, rappresenta lo Stop Loss statico per l'ordine;
- *S/L Trailing*: campo numerico, rappresenta lo Stop Loss dinamico (di inseguimento) per l'ordine;
- *S/L*: campo numerico, rappresenta lo Stop Loss effettivamente utilizzato in fase di valutazione della chiusura dell'ordine (per maggiori dettagli si rimanda alla descrizione della strategia di EXIST per stop-loss);
- *Closed*: campo booleano, indica se l'ordine è stato chiuso oppure no;
- *Counter*: campo numerico, utilizzato dalla strategia di EXIST_2, per tenere traccia del numero di candele in squeeze, a partire dalla candela di apertura.
- *OpenTS*: campo di tipo Timestamp, indica data e ora di apertura della posizione.
- *DeEvent*: campo numerico, indica l'estensione in PIPs della candela di apertura della posizione.
- *V_Open*: campo numerico, indica la velocità dello strumento al momento dell'apertura della posizione;
- *A_Open*: campo numerico, indica l'accelerazione dello strumento al momento dell'apertura della posizione;
- *EXIST2_TS*: campo Timestamp, memorizza data e ora utilizzati dalla strategia EXIST_2 per conoscere l'intervallo temporale trascorso il quale verificare la squeeze.

La Figura 63 mostra il cluster che memorizza tutti gli elementi necessari alla strategia per un ordine.

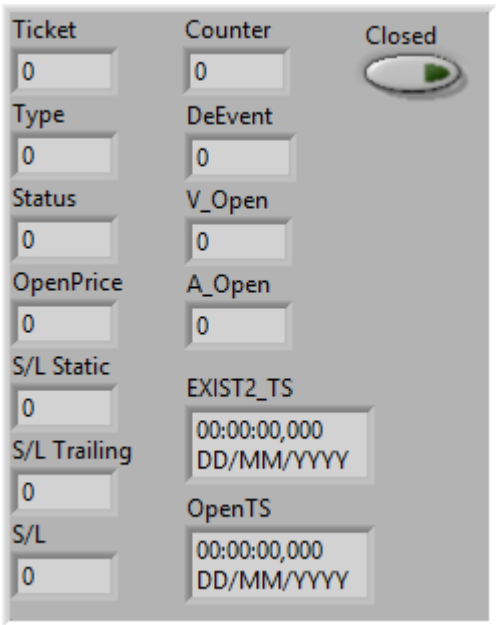


Figura 63 - Struttura dati Cluster degli ordini

6.6 Aree funzionali

Di seguito verranno descritte nei loro dettagli implementativi le sette aree funzionali indicate nel capitolo 2.

6.6.1 Area 1

L'area 1, come detto, si occupa di gestire l'inizializzazione del software, ad esempio riportare allo stato iniziale le strutture dati, le variabili globali della GUI e degli algoritmi per la valutazione dell'apertura della posizione, i vari controlli numerici e i percorsi alle directory di interfaccia.

All'interno di tale area inoltre viene richiamato il VI che si occupa di recuperare le informazioni relative agli ordini ancora aperti in piattaforma ma che derivano dall'esecuzione precedente, terminata per i motivi più svariati (ad esempio l'arresto improvviso dell'applicazione o della macchina). Nello specifico, il VI si occupa di leggere un file di testo all'interno del quale, ogni qual volta viene aperto un nuovo ordine, vengono scritte tutte le informazioni relative all'ordine stesso. Terminata la lettura, per ogni ordine individuato nel file (uno per ogni riga), viene effettuato il parsing del contenuto e strutturato un cluster della tipologia descritta nel paragrafo 5.2. Creato il cluster relativo all'operazione, lo stesso viene inserito all'interno dell'array contenente gli ordini. Nello specifico, il VI richiamato:

- *RestoreOpenOrder*: Riceve in input il nome dello strumento finanziario, così da individuare all'interno della directory di interfaccia tra MetaTrader4 e l'applicativo, il file relativo agli ordini per quello strumento. L'output è l'array di cluster contenente gli eventuali ordini contenuti nel file. Ritorna invece un array vuoto se non vi sono ordini aperti dall'esecuzione precedente dell'applicativo (v. Figura 65 e Figura 64).

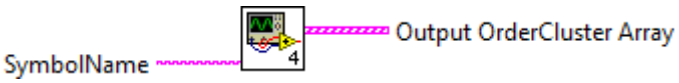


Figura 65 - Input / output del VI RestoreOpenOrder

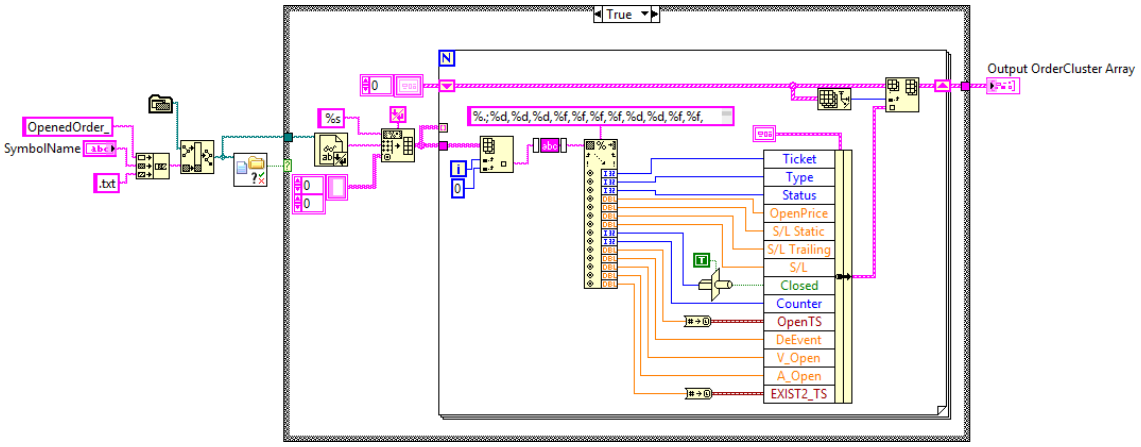


Figura 64 - Block Diagram del VI RestoreOpenOrder

Inoltre viene eseguita la porzione di codice che si occupa di aggiornare i file delle news e dell'estrazione dei dati stessi. La Figura 66 mostra la sezione per l'aggiornamento del file delle news, mentre la Figura 67 descrive la creazione del cluster contenente le news e si calcola data e ora in cui verrà eseguito il nuovo update dei file

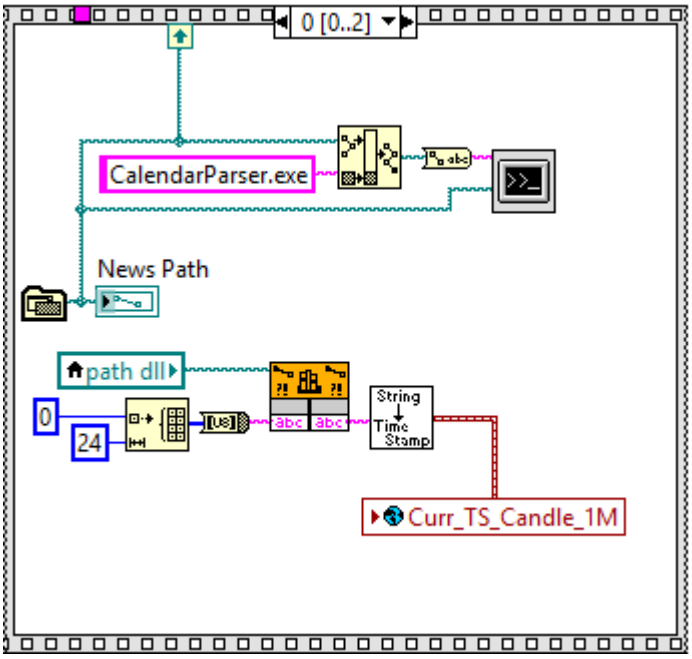


Figura 66 - Stralcio del block diagram relativo all'update del file delle news

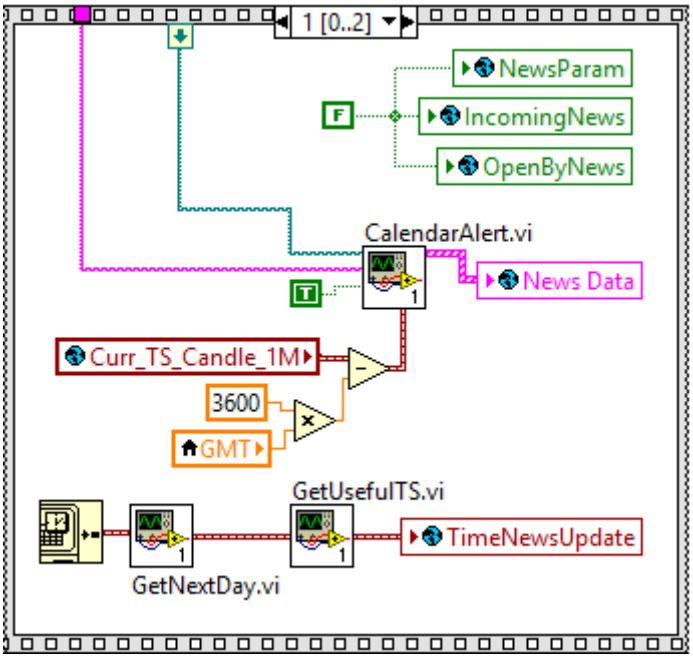


Figura 67 - Stralcio del block diagram relativo alla creazione dell'array di cluster delle news.

Infine la Figura 68 mostra l'estrazione e impostazione dei parametri iniziali di apertura in modalità "*Simple*", in quanto si è scelto che, all'avvio dell'applicativo, non ci si può mai trovare in regime "*News*", in quanto la strategia potrebbe decidere di entrare a mercato pur non avendo ancora acquisito tutte le informazioni provenienti dalla piattaforma ed in generale necessarie al corretto funzionamento della strategia stessa.

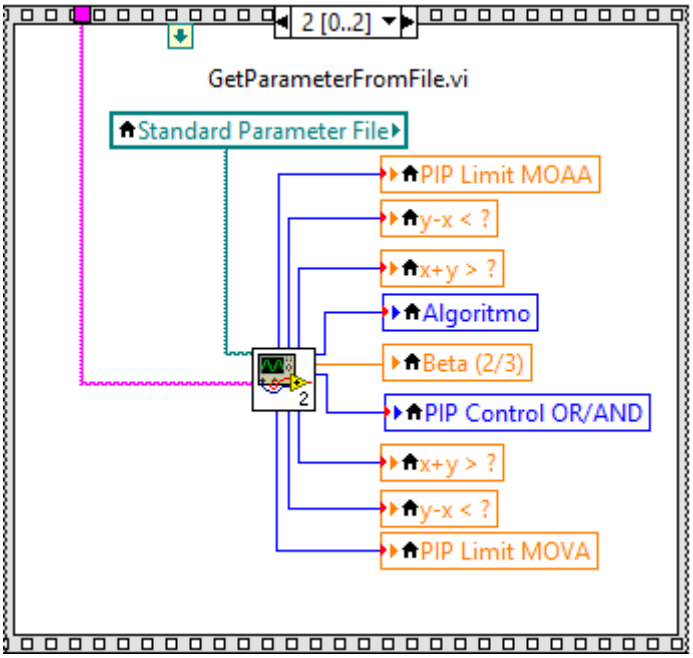


Figura 68 - Impostazioni parametri iniziali, strategia Simple.

In Quest'area si richiamano i VI:

- *UpdateEconomicCalendar*: si occupa di aggiornare i file delle news. Riceve in input:
  - *News_File*: path al file delle news "Dati-News.xlsm",
  - *Visible*: booleano, impostato a false di default, che permette di nascondere o mostrare le fasi di aggiornamento del file delle news.
  - *CalendarAlert*: si occupa di estrarre le informazioni dai file delle news, suddivisi per importanza, e costruire il cluster News Data, contenuto nella variabile globale. Riceve in input:
  - *Base path file News*: percorso al file delle news
  - *Currency*: Nome dello strumento finanziario in esame per il filtraggio delle news
  - *FilterByCurrency*: booleano per l'attivazione/disattivazione delle news secondo lo strumento finanziario

  Output:

  - *NewsData*: Cluster contenente i dati estratti dai file delle news.
- *GetNextDay* e *GetUsefulTS*: utilizzati per calcolare il Timestamp del successivo aggiornamento del file delle news.

6.6.2 Area 2

L'area 2 è adibita all'inizializzazione dei buffer, di prezzo e Timestamp, utilizzati dagli algoritmi MOAA e MOVA per eseguire le operazioni precedentemente descritte.

In quest'area viene richiamato:

- GetFirstArray: Questo VI esegue un ciclo while fintanto che non sono stati acquisiti dalla piattaforma un numero di nuovi valori di prezzo pari alla dimensione del buffer predisposto alla loro memorizzazione.

In Figura 69 si mostra l'input / output del VI, mentre Figura 70 il block diagram dello stesso.

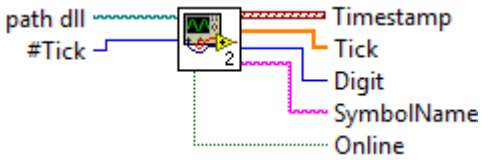


Figura 69 - Input / output del VI GetFirstArray

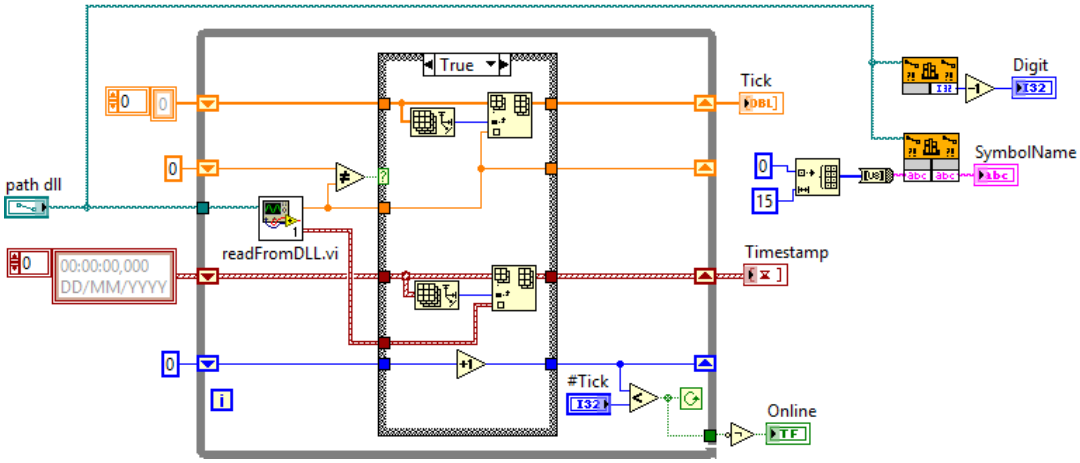


Figura 70 - Block Diagram del VI GetFirstArray

Il VI Riceve in input:

- *Path*: path alla DLL;
- *#Tick*: dimensione che deve avere ogni buffer (10 elementi di default).

L'output è invece:

- *Tick*: Buffer dei livelli di prezzo;
- *Timestamp*: Buffer degli istanti temporali riferiti ai livelli di prezzo memorizzati nel buffer *Tick*;
- *Online*: Boolean TRUE/FALSE che viene attivato al termine del ciclo while, indicando il termine della fase di buffering;
- *Digit*: numero di cifre significative dei livelli di prezzo dello strumento;
- *SymbolName*: Nome dello strumento finanziario in esame.

Richiama:

- readfromDLL.vi.

La Figura 71 mostra un estratto del block diagram dell'Area 2.

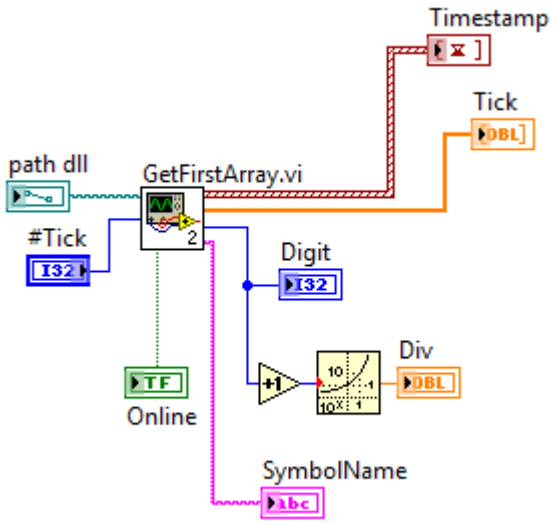


Figura 71 - Estratto del Block diagram dell'Area 2

6.6.3 Area 3

Dopo le fasi di inizializzazione tramite le due aree precedentemente dettagliate, l'applicativo si evolve con il ciclo while che, ricevuto un nuovo aggiornamento di prezzo da parte della piattaforma, richiama, in base alla combinazione selezionata, i due algoritmi MOAA e MOVA e, nel caso in cui vi siano le condizioni necessarie, viene generata la nuova posizione da inviare alla piattaforma MetaTrader4.

Le condizioni di apertura delle posizioni, oltre ai vincoli stabiliti dai due algoritmi MOAA e MOVA, prevedono che venga valutata anche la possibile presenza di una imminente news da calendario. Difatti, prima di effettuare i relativi controlli sulla volatilità dello strumento finanziario, l'applicativo dapprima verifica la necessità di effettuare l'update dei dati delle news, confrontando il Timestamp attuale con quello di prossimo update memorizzato all'interno della variabile globale e, in caso di esito positivo del confronto, esegue le operazioni per l'aggiornamento specificate nel paragrafo 6.6.1.

Successivamente si esamina, tramite il VI *CheckNewsEvent_V2*, la struttura di memorizzazione delle news per verificare se la successiva si trova nell'arco temporale di attivazione della modalità "*News*". In tal caso sarà necessario porre in essere quanto descritto nell'introduzione a questo capitolo riguardo al funzionamento in tale modalità. Nello specifico vengono richiamati i seguenti VI:

- **CheckNewsEvent_V2**: Tale vi si occupa di esaminare, in base alla tipologia di news di interesse dell'utente, il primo elemento dell'array dei Timestamp, che corrisponderà alla prima news in ordine temporale, essendo le news ordinate in tal senso. In questo caso viene impostato a true un booleano, indicante la presenza di una news, e vengono esaminati i rimanenti elementi all'interno dell'array al fine di rimuovere eventuali altre news che possano avere Timestamp coincidente con quello che ha appena generato l'evento "*News*". Nelle tre figure seguenti si mostrano tre estratti del block diagram di questo VI, nello specifico la Figura 72 mostra l'estrazione delle news di interesse per tipologia selezionata dall'utente; la Figura 73 mostra la selezione delle news che rientrano nell'intervallo di tempo prestabilito dall'utente per l'attivazione dei parame-

tri "*News*"; la Figura 74 mostra l'utilizzo del VI *FilterNewsClusterArray_OnNewsEvent* necessario a filtrare le news rimanenti, una volta individuata e selezionata una news appartenente alla categoria di interesse dell'utente.

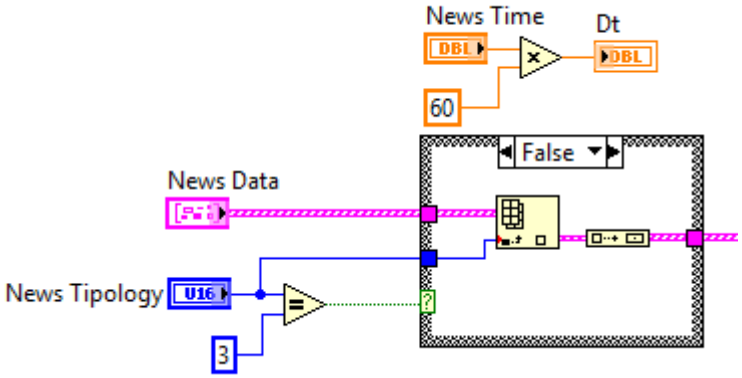


Figura 72 - Estrazione delle news di interesse per tipologia selezionata dall'utente.

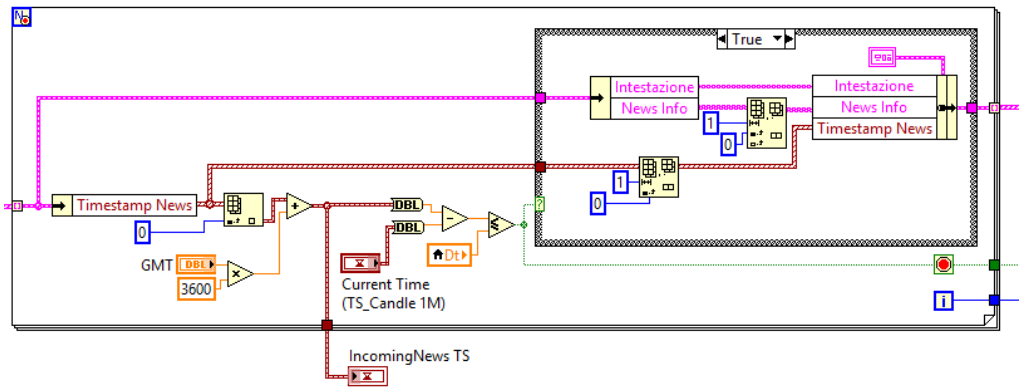


Figura 73 - Selezione delle news che rientrano nell'intervallo di tempo prestabilito dall'utente per l'attivazione dei parametri "News".

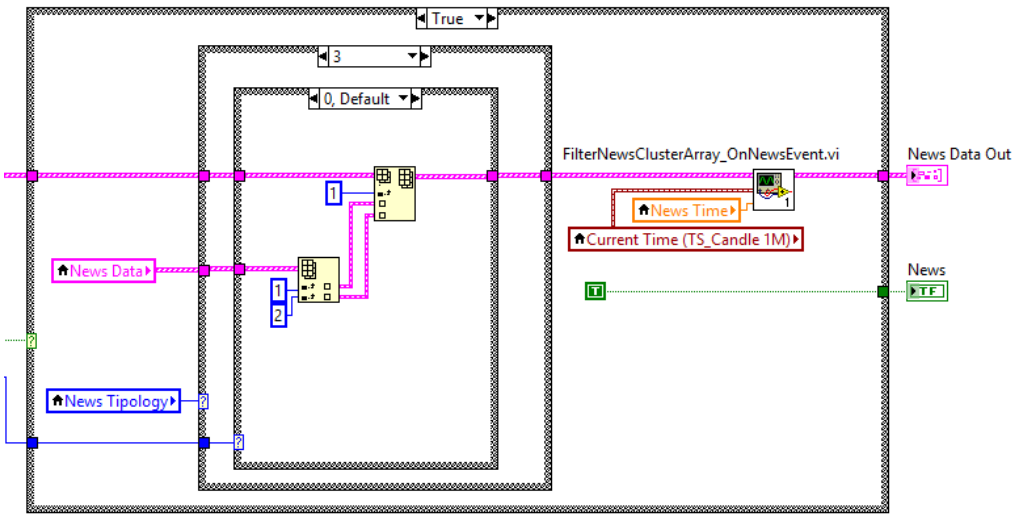


Figura 74 - Utilizzo del VI FilterNewsClusterArray_OnNewsEvent.

La Figura 75 mostra l'input / output del VI, descritti nel dettaglio di seguito:

- *NewsData:* Cluster contenente i dati delle news;
- *News Time*: tempo in minuti per l'attivazione della fase di "Allerta news"
- *GMT*: tempo in ore relative alla differenza di fuso orario tra le news e il broker
- *CurrentTS*: Timestamp Corrente prelevato dalla piattaforma.
- *News Tipology*: News di interesse dell'utente (1, 2, 3, Tutte)

Pone in output:

- *NextNewsTS*: Timestamp della prossima news
- *NewsData Out*: Cluster dei dati News Aggiornato
- *News:* Booleano indicante la presenza di una nuova news.

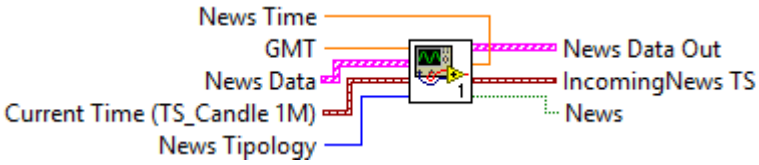


Figura 75 - Input / output del VI CheckNewsEvent_V2.

Il VI *CheckNewsEvent_V2* richiama a sua volta il vi *FilterNewsClusterArray_OnNewsEvent* (v. Figura 76) riceve in input:

- NewsData;
- CurrentTS;
- NewsTime.

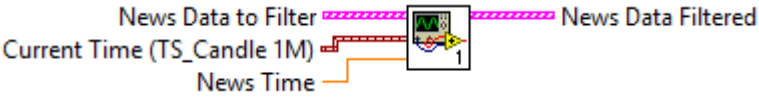


Figura 76 - Input / output del VI FilterNewsClusterArray_OnNewsEvent.

Restituisce in output:

- *NewsData_Out*: Cluster ripulito dai timestamp coincidenti delle news

Una volta attivato il booleano indicante la presenza di una news, il codice si evolve impostando i parametri di apertura in presenza di news, tramite il VI *GetParameterFromFile*, e confrontando, una volta al minuto, se è possibile abilitare l'apertura di nuove posizioni utilizzando tali parametri. Tale situazione si verificherà in corrispondenza dell'istante temporale della news stessa.

Il Vi di lettura dei parametri di apertura è strutturato come una hashmap, e contiene un record per ogni strumento finanziario possibile. Il campo "*chiave*" dell'hashmap è appunto il nome dello strumento, coincidente con quello letto dalla piattaforma, mentre il campo "*valore*" rappresenta una espressione regolare (ovvero le valute che possono influenzare lo specifico strumento, separate da pipe) che viene utilizzata dal vi per filtrare le news secondo lo specifico strumento. Ad esempio, in Figura 77, un esempio della struttura del file:

```
GOLD,EUR|USD|GBP
#EPM4,EUR|USD|GBP
GBPUSD,GBP|USD
```

Figura 77 - Struttura di esempio del file contenente l'hashmap.

A tal punto, trascorsi *T* minuti, l'applicativo ricaricherà il set di parametri *Simple* per lo strumento in esame e riprenderà con le normali attività.

Successivamente alla verifica della presenza di una news, l'area tre richiama prima i due algoritmi e, successivamente, richiama i due VI per il dimensionamento della posizione e la creazione dell'ordine di seguito descritti:

- *DimLots*: tale vi si occupa di eseguire il dimensionamento della posizione da aprire in base al margine libero, percentuale di rischio (in PIP) che il trader vuole assumersi, e alla percentuale di allocazione.

  Gli input di tale algoritmo sono:

  - Percentuale di allocazione;
  - Rischio (in PIP);
  - Leva;
  - SymbolName.

  Gli output sono:

  - *Lotti da Allocare*: numero di lotti calcolati dall'algoritmo
  - SymbolNameOut.

La Figura 78 mostra gli input / output del VI *DimLots*.

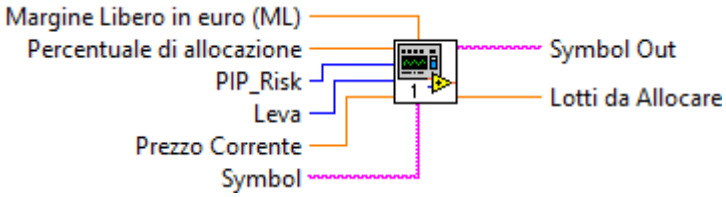


Figura 78 - Input / output del VI DimLots.

- OpenPos: si occupa di richiamare il VI che costruisce la stringa di testo per l'apertura della posizione. Tale stringa di testo contiene i parametri (separati da virgola) necessari all'Expert Advisor per richiedere al broker l'apertura di nuove posizioni.

  Riceve in input:

  - *InterfaceDir*: path alla directory di interfaccia
  - *Slope*: necessaria per individuare la tipologia della posizione:
    - Slope = 1 → Ordine in Buy (Long)

- Slope = -1 → Ordine in Sell (Short)

- Numero di lotti;
- SymbolName;

Richiama:

- *CreateOrderFile*: crea materialmente all'interno della directory di interfaccia tra i due applicativi, il file contenente l'ordine. Essendo un vi generico, prevede i parametri per l'apertura di tutte le possibili operazioni previste da Metatrader. Nel caso di Volatility, aprendo solo operazioni al mercato, soltanto alcuni di questi sono necessari, e sono di seguito elencati:
  - *InterfaceDir*: path alla directory di interfaccia;
  - *Numero di lotti*: double, dimensione della posizione;
  - *Tipo Operazione*: costante, "Esecuzione al Mercato"
  - *Buy / Sell*: Booleano;
  - *Cross*; stringa contenente il nome dello strumento finanziario

Restituisce un Booleano, il cui valore viene scritto all'interno della variabile globale *NewOrder*, che indica la presenza di un nuovo ordine pronto ad essere letto dall'Expert Advisor attivo in piattaforma ed inviato al broker. La

Figura 79 mostra lo schema di input / output del VI *OpenPos*¸ mentre la

Figura 80 il suo block diagram con la chiamata al VI *CreateOrderFile*. La Figura 81 mostra lo schema di input / output del VI *CreateOrderFile*.

Infine la Figura 82 mostra uno stralcio del block diagram di ciò che viene realizzato in tale area, nello specifico si è preso in esame il caso in cui vengono valutati entrambi gli algoritmi MOAA e MOVA.

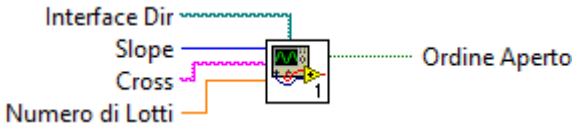


Figura 79 - Input / output del VI OpenPos.

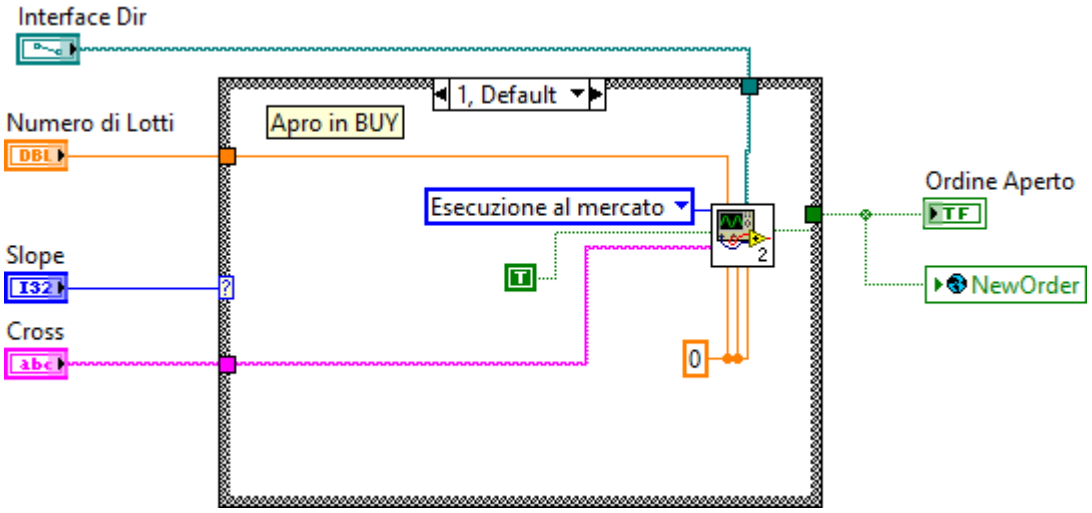


Figura 80 - Block diagram del VI OpenPos.

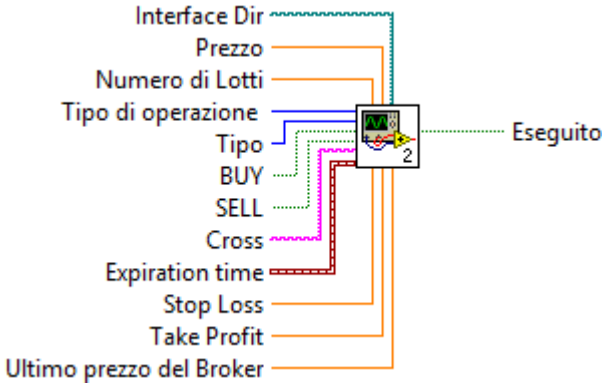


Figura 81 - Input / output del VI CreateOrderFile.

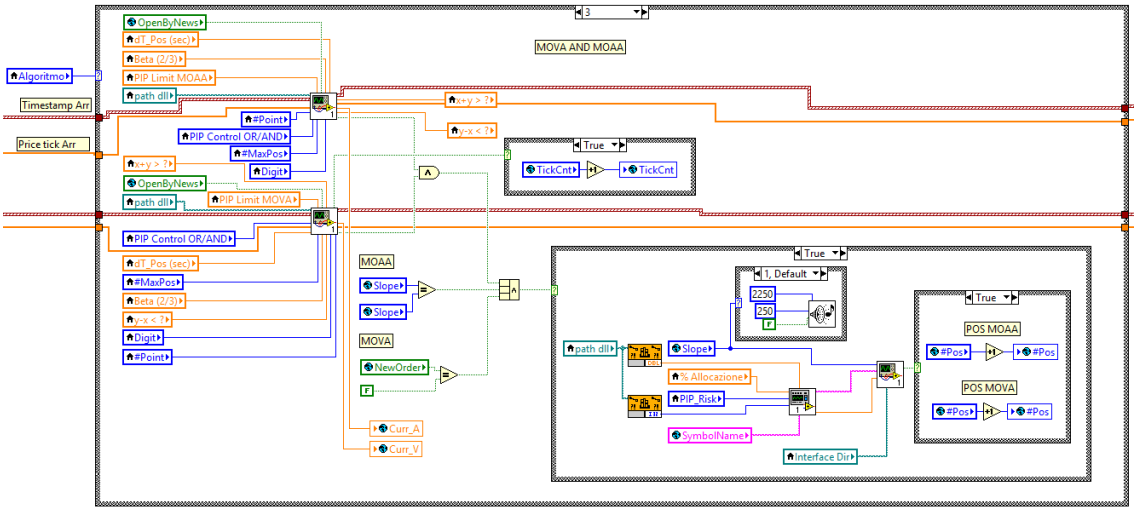


Figura 82 - Stralcio del block diagram rappresentativo della sezione di codice dedicata all'apertura posizioni.

6.6.4 Area 4

Dopo aver creato il file contenente le informazioni relative al nuovo ordine, è necessario acquisire le informazioni ad esso relative dalla piattaforma. Tale operazione è gestita dall'area 4 del software Volatility. In tale area si attende che il file relativo all'ordine sia stato cancellato dall'Expert Advisor in esecuzione sulla piattaforma, il che avviene soltanto quando l'ordine è stato preso in carico dal broker.

La cancellazione di tale file implica che l'Expert Advisor abbia scritto le informazioni relative al nuovo ordine (ovvero *ID*, *Open-Price* e *Open-Time*) nell'area di memoria predisposta all'interno della DLL a memorizzare tali informazioni.

Volatility accede quindi a queste informazioni, verifica che effettivamente i dati in memoria corrispondano ad un nuovo ordine (leggendo il flag **IsNew** settato a "*1*" dall'Expert Advisor), crea e riempie i campi del cluster relativo al nuovo ordine inserendolo all'interno dell'array contenente gli ordini aperti. Infine l'applicativo imposta il flag *IsNew* a *0* e la booleana *NewOrder* a *FALSE*. Tramite questa tecnica di sincronizzazione ci si assicura di leggere, non appena queste sono disponibili, i dati relativi al nuovo ordine.

All'interno di quest'area non vengono richiamati VI, in quanto essendo questa una operazione atomica, è necessario che stia all'interno del VI principale, al fine di mantenere la corretta sequenza logica di esecuzione. La Figura 83 mostra uno stralcio del block diagram di ciò che viene realizzato in tale area, nello specifico si è preso in esame il caso in cui vengono valutati entrambi gli algoritmi MOAA e MOVA.

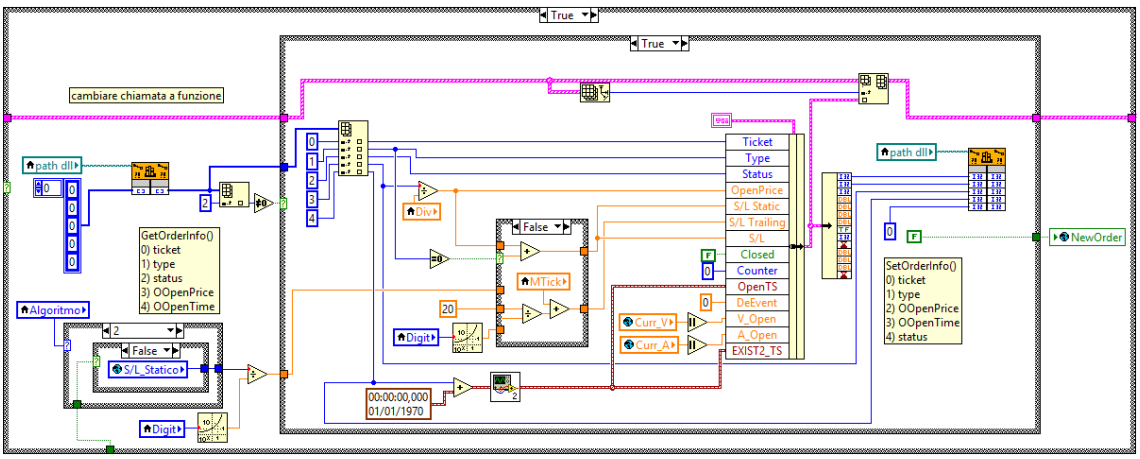


Figura 83 - Area dedicata alla lettura dei dati sull'ordine appena aperto in piattaforma MetaTrader4.

6.6.5 Area 5

All'interno dell'area 5 vengono testate, su tutti gli ordini ancora aperti, le cinque strategie di EXIST progettate per l'applicativo. Di seguito verranno singolarmente descritte le strategie e la loro implementazione. Si specifica che ogni strategia di EXIST, pone in output un array di cluster. Tali Array contengono gli ordini che devono essere posti in chiusura e l'identificativo corrispondente alla strategia per il quale tale ordine è stato chiuso.

L'array complessivo delle chiusure, risultato della concatenazione dell'output di tutte le strategie di uscita, viene poi esaminato per creare il file contenente i ticket degli ordini da chiudere. Le strategie di uscita sono poste in cascata, ovvero in serie, al fine di verificarle per ordine di importanza. Ad esempio la prima da verificare è la chiusura per Stop Loss, ovvero l'ordine va chiuso immediatamente se si è raggiunta la perdita prefissata, senza ulteriori verifiche di strategie di uscita.

La Figura 84 mostra una vista d'insieme dell'area che implementa le strategie di uscita e la creazione del file delle chiusure.

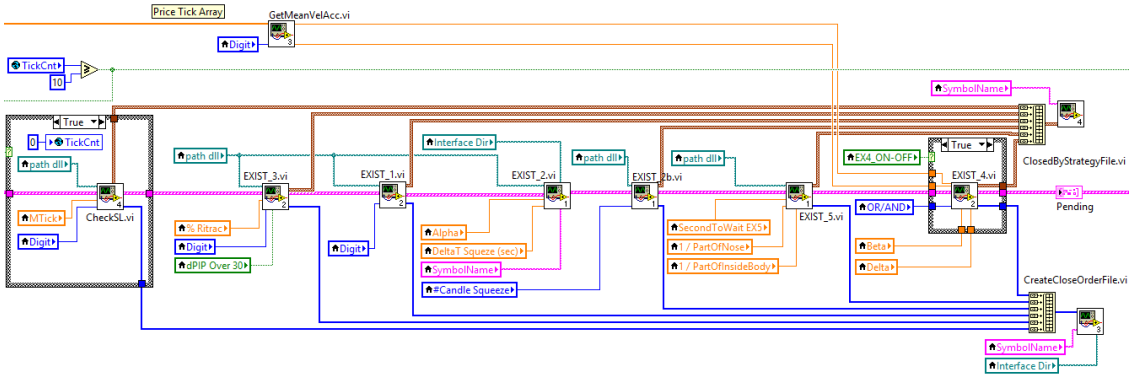


Figura 84 - Block diagram dell'Area 5, dedita alle chiusure degli ordini a mercato.

6.6.5.1 *Check_SL*

Questa strategia si occupa di calcolare runtime, ovvero ogni dieci aggiornamenti del livello di prezzo, lo Stop Loss per il singolo ordine. Ad ogni iterazione, viene calcolato uno Stop Loss definito "Trailing", ovvero di inseguimento del livello di prezzo, mediante confronto rispetto al valore medio degli ultimi tre livelli di prezzo noti. Se tale Stop Loss è minore di quello statico, allora questo sarà il nuovo valore di SL da verificare, altrimenti si mantiene lo SL statico. Da ciò si evince come lo stop loss non sarà mai maggiore, in valore assoluto, dello Stop Loss Statico calcolato in fase di apertura della posizione. In formule, si ha:

Ordini Buy:

$$SL_{Trailing} = MeanTick - 20\ PIP$$

$$\begin{cases} SL = SL_{Trailing} & se \quad SL_{Trailing} < SL_{Static} \\ SL = ST_{Static} & se \quad SL_{Trailing} \geq SL_{Static} \end{cases}$$

Ordini Sell:

$$SL_{Trailing} = MeanTick + 20\ PIP$$

$$\begin{cases} SL = SL_{Trailing} & se \quad SL_{Trailing} \geq SL_{Static} \\ SL = ST_{Static} & se \quad SL_{Trailing} < SL_{Static} \end{cases}$$

Ovviamente, dopo aver ricalcolato lo SL Trailing e scelto lo SL da utilizzare per l'ordine in esame, questo viene verificato rispetto al livello corrente di prezzo e, eventualmente, viene dato l'ordine di chiusura dell'ordine.

Riceve in input:

- *PathDLL*: path alla DLL di interfaccia
- *MTick*: valor medio calcolato sugli ultimi 3 tick;
- *Digit*: numero di cifre significative dello strumento.

Output:

- Array degli ordini aggiornato.

La Figura 85 mostra l'elenco degli input / output del VI *CheckSL*, mentre la Figura 86 ne mostra il block diagram dell'implementazione.

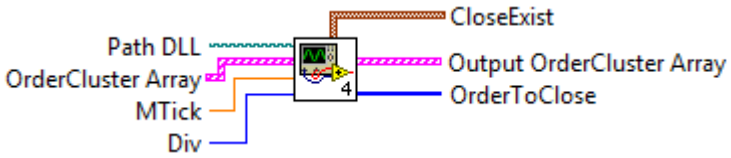


Figura 85 - Input / output del VI CheckSL.

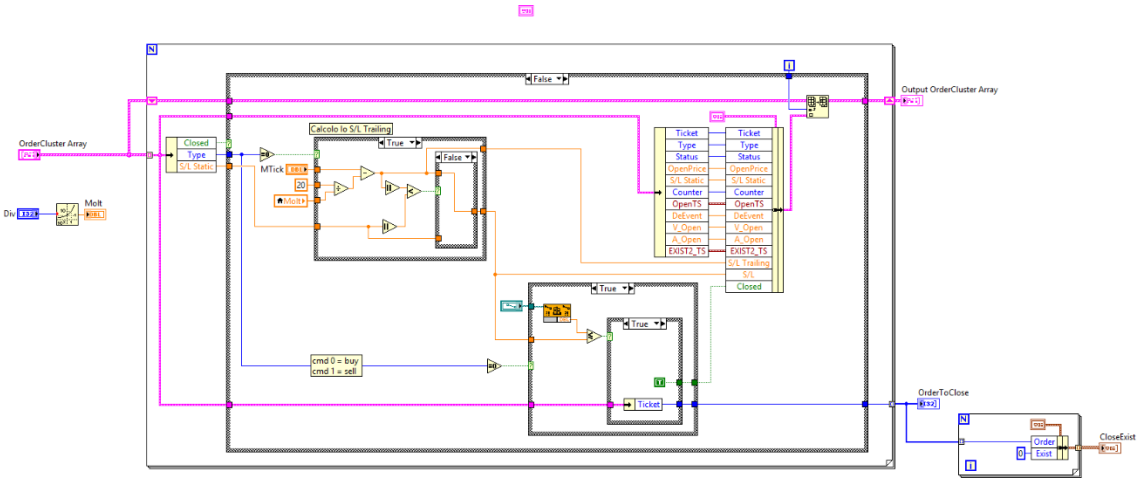


Figura 86 - Block diagram del VI CheckSL.

6.6.5.2 *EXIST_1*

Questa strategia verifica se, dal momento dell'apertura dell'ordine, lo strumento ha compiuto più PIP della media giornaliera, calcolata su 100 periodi. In tal caso, evidentemente, non vi sarà più margine di profitto su quello strumento, per cui l'ordine viene chiuso. La verifica viene compiuta calcolando il numero di PIP tra il prezzo di apertura ed il prezzo corrente. La Figura 87 mostra l'input / output del VI EXIST_1.

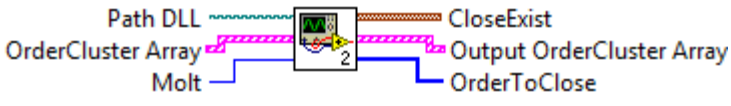


Figura 87 - Input / output del VI EXIST_1.

Riceve in input:

- PathDLL: path alla DLL di interfaccia;

Output:

- Array degli ordini aggiornato.

La Figura 88 mostra il block diagram della strategia di uscita appena descritta.

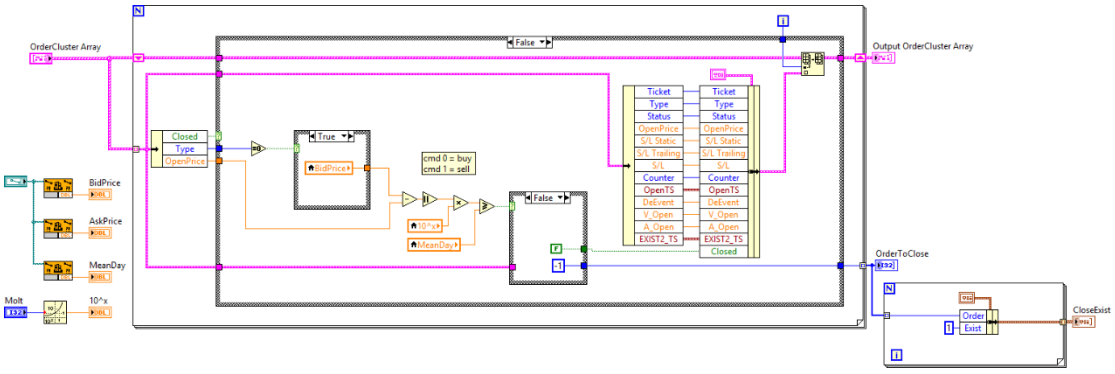


Figura 88 - Block diagram del VI EXIST_1.

6.6.5.3 *EXIST_2*

La Strategia EXIST_2 si occupa di verificare se lo strumento finanziario, dopo l'avvento dell'evento energetico, si trova in una fase di squeeze. Per considerare un ordine da chiudere per questa strategia, bisogna che siano state individuate, dopo l'evento, almeno 5 candele (anche non consecutive) la cui estensione sia 1/β della candela dell'evento. Tale strategia di uscita viene verificata eseguendo, per ogni ordine, un meccanismo di *request/response* tra la piattaforma e l'applicativo, tramite l'utilizzo di un file che funge da semaforo. Infatti, per utilizzare tale strategia, è necessario richiedere alla piattaforma quale sia l'estensione della candela dell'evento energetico. Per ogni ordine aperto di cui non si conosce tale parametro, la EXIST_2 crea un file contenente il Ticket dell'ordine da processare, e si pone in attesa di una risposta da parte della piattaforma. L'Expert Advisor, individuato il file di REQUEST, estrae l'ID dell'ordine, individua il minuto in cui è stato aperto, calcola l'estensione della corrispondente candela e scrive all'interno di un file RESPONSE tale valore. La EXIST_2, alla successiva iterazione, avrà a disposizione il file di risposta, ne leggerà il contenuto e potrà quindi verificare la presenza o meno di candele in squeeze, aggiornando, per l'ordine i-esimo, il valore del contatore.

Questa EXIST è in realtà suddivisa in due SubVi, *EXIST_2*, che semplicemente verifica la squeeze ed aggiorna il contatore, ed *EXIST_2b* che invece individua il miglior punto di uscita dal mercato. Tale scelta viene operata calcolando il centro geometrico dell'ultima candela chiusa, e considerando tale valore come limite (inferiore o superiore, in base al tipo di ordine), raggiunto il quale, confrontato con il corrente valore di prezzo, è possibile effettivamente dare l'ordine di chiusura. La Figura 89 mostra l'input / output del VI *EXIST_2*, mentre le Figura 90 e la Figura 91 mostrano le due aree del block diagram atte ad eseguire la fase di request e response rispettivamente.

EXIST_2 Riceve in input:

- *PathDLL*: path alla DLL di interfaccia;
- *InterfaceDir*: path alla directory di interfaccia per la scrittura dei file REQUEST/RESPONSE;
- *Alpha*: percentuale dell'estensione della candela dell'evento;

- *DeltaT Squeeze (sec)*: tempo trascorso il quale è possibile verificare la presenza di una nuova candela in squeeze (di default ogni 60 sec);
- *SymbolName*: nome dello strumento finanziario.

Output per *EXIST_2*:

- Array degli ordini aggiornato.

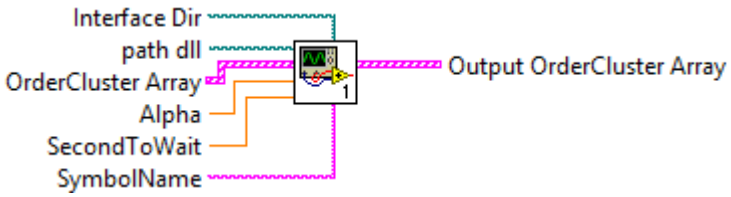


Figura 89 - Input / output per il VI EXIST_2.

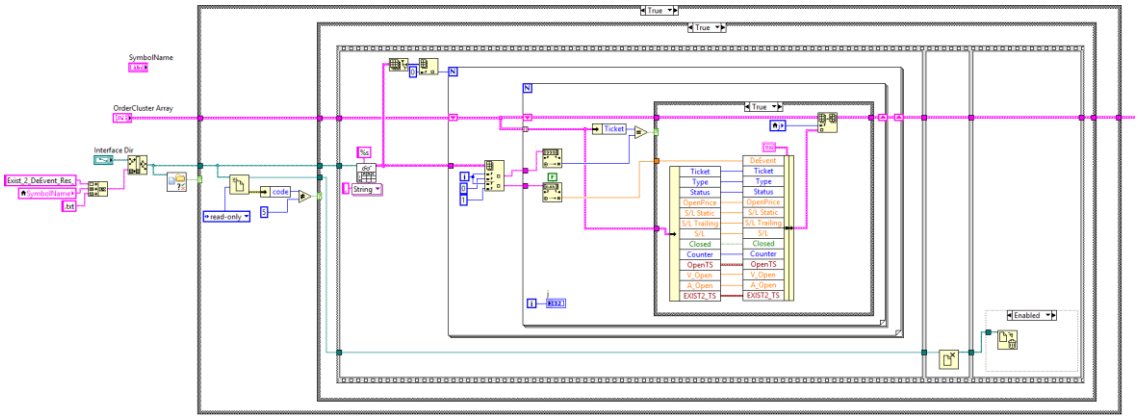


Figura 90 - Block diagram della fase di response per la EXIST_2.

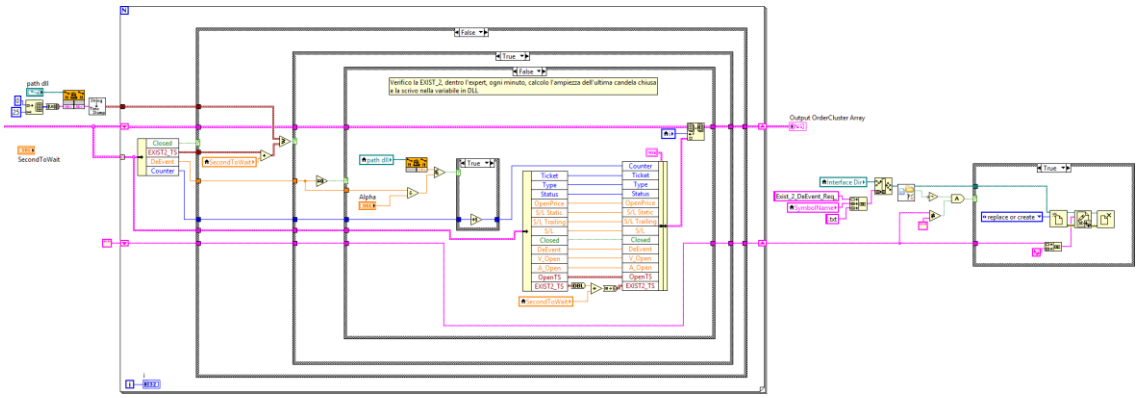


Figura 91 - Block diagram della fase di request per la EXIST_2.

EXIST_2b Riceve in input:

- *PathDLL*: path alla DLL di interfaccia;
- #CandleSqueeze

Output per *EXIST_2b*:

- Array degli ordini aggiornato.

La Figura 92 mostra lo schema di input / output del VI *EXIST_2b*.

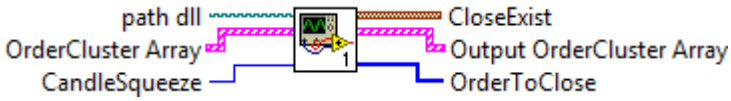


Figura 92 - Input / output del VI EXIST_2b.

6.6.5.4 *EXIST_3*

Questa strategia si occupa di calcolare, fintanto che ci si trova all'interno del minuto di apertura della posizione, se vi è una percentuale di ritracciamento del livello di prezzo tale da rendere necessaria la chiusura della posizione (se questa è contraria al movimento), a causa di una classica "finta" di movimento dello strumento. Inoltre per tale strategia è possibile imporre che il ritracciamento sia verificato solamente per movimenti maggiori di 30 PIP, rispetto al prezzo di apertura.

Per le operazioni in Buy, la chiusura si ha (con il limite mimino di 30 PIP) se:

$$|Bid - High| \geq 20\% \cdot (|OpenPrice - Bid|) \wedge (|OpenPrice - Bid|) \geq 30\ PIP$$

Per le operazioni in Sell, la chiusura si ha (con il limite mimino di 30 PIP) se:

$$|Ask - Low| \geq 20\% \cdot (|OpenPrice - Ask|) \wedge (|OpenPrice - Ask|) \geq 30\ PIP$$

EXIST_3 Riceve in input:

- *PathDLL*: path alla DLL di interfaccia;
- *%Ritracciamento:* percentuale di ritracciamento della candela da considerare
- *dPIPOver30:* booleano di attivazione/disattivazione del secondo controllo
- *Digit:* cifre significative dello strumento

Output per EXIST_3:

- Array degli ordini aggiornato;

La Figura 93 mostra l'input/output del VI, mentre la Figura 94 ne mostra il block diagram.

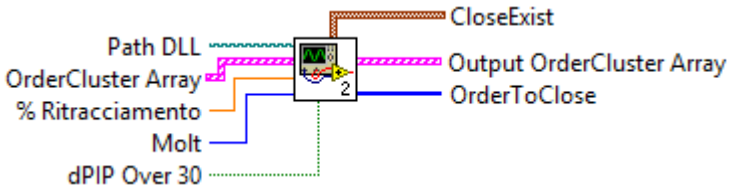


Figura 93 - Input / output del VI EXIST_3.

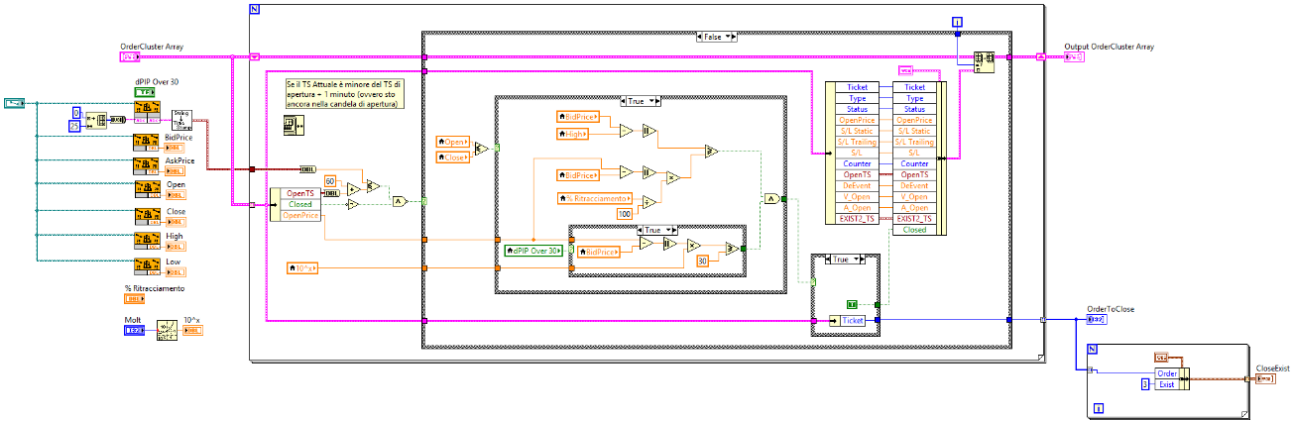


Figura 94 - Block Diagram del VI EXIST_3.

6.6.5.5 *EXIST_4*

Questa strategia di chiusura interviene quando l'attuale velocità e/o accelerazione dello strumento finanziario si sono ridotti di una specifica percentuale (selezionata dall'utente), rispetto a quella misurata all'atto dell'apertura della posizione stessa (e conservata nei relativi campi all'interno del cluster). Nello specifico si ha la chiusura della posizione se:

Posto β percentuale della velocità dell'evento e

Posto δ percentuale della accelerazione dell'evento

$$|Curr\_V| \leq \beta \cdot |OpenV| \wedge \vee |Curr\_A| \leq \delta \cdot |OpenA|$$

EXIST_4 Riceve in input:

- *PathDLL*: path alla DLL di interfaccia;
- *β:* percentuale da considerare della velocità di apertura
- *δ:* percentuale da considerare della velocità di apertura
- *Curr_V:* velocità corrente dello strumento finanziario
- *Curr_A:* accelerazione corrente dello strumento finanziario
- *AND/OR:* controllo per la selezione della modalità di verifica della EXIST

Output per EXIST_4:

- Array degli ordini aggiornato;

La Figura 95 mostra l'input/output del VI, mentre la Figura 96 ne mostra il block diagram.

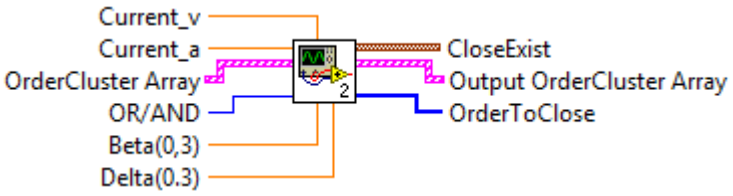


Figura 95 - Input / output del VI EXIST_4.

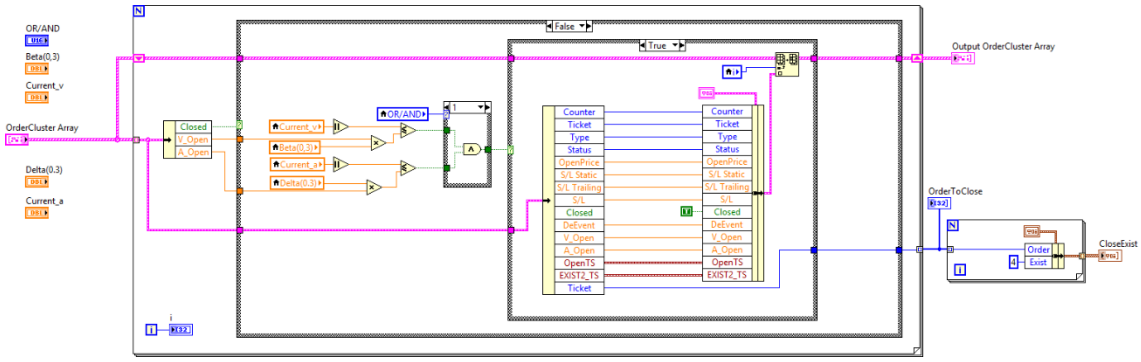


Figura 96 - Block Diagram del VI EXIST_4.

6.6.5.6 *EXIST_5*

L'ultima strategia di uscita è basata sul pattern della candela dell'evento, ovvero si verifica che tale candela si trovi con una parte del "*naso*" (ovvero l'ombra) ed una parte del "*corpo*" oltre la Banda di Bollinger (superiore o inferiore in base alla tipologia di ordine). Tale pattern, in congiunzione con l'uso delle Bande di Bollinger, indicano una riduzione della volatilità, che è indice della fine del movimento, condizione per la quale è preferibile chiudere l'operazione. Nello specifico, si identifica prima la tipologia di candela, stabilendo se questa possiede le caratteristiche necessarie per verificare la EXIST_5:

$$\begin{cases} Up = Close, Down = Open \quad se\ Open \leq Close \\ \vee \\ Up = Open\ , Down = Close \quad se\ Open > Close \\ \\ Body = |Open - Close| \\ Nose = \ |Up - High|\ se\ Buy\ Operation \\ \vee \\ Nose = \ |Down - Low|\ se\ Sell\ Operation \end{cases}$$

L'operazione verrà chiusa se, per le operazioni in Buy:

$$\begin{cases} Nose \ \geq \dfrac{1}{PartOfNose} \cdot |High - Low| \\ \wedge \\ BBSup \ \leq Down + \dfrac{Body}{PartOfInsideBody} \end{cases}$$

Mentre per le operazioni in Sell:

$$\begin{cases} Nose \geq \dfrac{1}{PartOfNose} \cdot |High - Low| \\ \wedge \\ BBInf \geq Up - \dfrac{Body}{PartOfInsideBody} \end{cases}$$

EXIST_5 Riceve in input:

- *PathDLL*: path alla DLL di interfaccia;
- *SecondToWait*: Tempo trascorso il quale effettuare la verifica della strategia (30 secondi di default)
- *PartOfNose*: Frazione del "naso" da considerare per la verifica della strategia
- *PartOfInsideBody*: Frazione del "corpo" della candela da considerare per la verifica della strategia.

Output per EXIST_5:

- Array degli ordini aggiornato;

La Figura 97 mostra l'input/output del VI, mentre la Figura 98 ne mostra il block diagram.

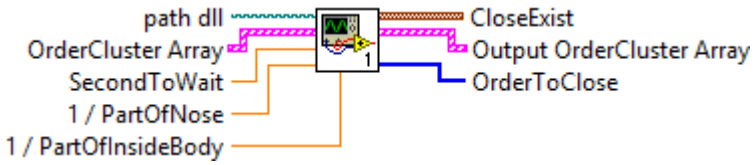


Figura 97 - Input / output del VI EXIST_5.

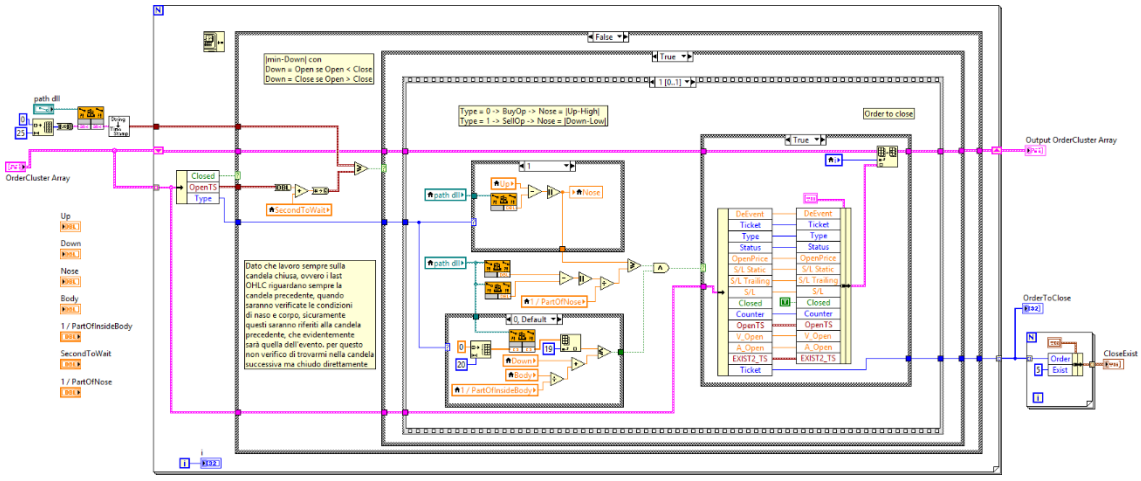


Figura 98 - Block diagram del VI EXIST_5.

6.6.6 Chiusura delle operazioni

Dopo aver testato in cascata, su ogni ordine aperto, le strategie di uscita appena descritte, l'applicativo completa la sua iterazione creando e scrivendo all'interno della directory di interfaccia il file contenente gli ID degli Ordini da chiudere, dal nome "*OrderToClose_SymbolName.txt*".

Tale operazione viene svolta dal VI *CreateCloseOrderFile*.

Riceve in input:

- *InterfaceDir*: path alla Directory di interfaccia;
- *OrderToClose*: Array contenente gli ID degli ordini da chiudere
- *SymbolName:* Nome dello strumento finanziario.

Output:

- Nessuno a livello di codice LabView;
- Crea il file con gli ordini da chiudere destinato a MetaTrader.

L'input / output del VI è mostrato in Figura 99.

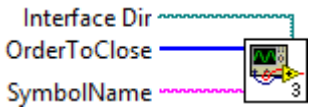


Figura 99 - Input / output del VI CreateCloseOrderFile.

Oltre al VI appena descritto, viene richiamato anche il VI *ClosedByStrategyFile*. Questo VI, esaminando uno per uno gli elementi dell'array di cluster degli ordini posti in chiusura, ha lo scopo di creare un file di Log in cui, per ogni riga, si avrà l'ID dell'ordine chiuso ed il rispettivo identificativo della strategia che ha dato l'ordine di chiusura.

Riceve in input:

- *CloseExist*: Array di cluster contenente gli ID degli ordini chiusi ed il relativo ID di strategia di chiusura;
- *SymbolName:* Nome dello strumento finanziario.

Output:

- Nessuno a livello di codice LabView;
- Crea il file con il log degli ordini chiusi e le rispettive strategie di chiusura.

In Figura 100 si mostra l'input / output del VI.

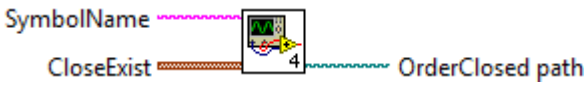


Figura 100 - Input / output del VI ClosedByStrategyFile.

6.6.7 Area 6

Tale area è dedicata al backup degli ordini aperti all'interno di un file testuale. Ogni 10 Tick viene richiamato il VI *BackupOpenOrder* che esamina l'array di cluster degli ordini e scrive su di un file dal nome *OpenedOrder_SymbolName.txt* quelli ancora a mercato.

Questo VI riceve in input:

- *Pending*: Array di cluster degli ordini;
- *SymbolName:* Nome dello strumento finanziario.

Output:

- Nessuno a livello di codice LabView;
- Crea il file con il log degli ordini ancora a mercato.

La Figura 101 mostra l'input / output del VI BackupOpenOrder.

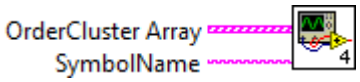


Figura 101 - Input / output del VI BackupOpenOrder.

6.6.8 Area 7

Congiuntamente al termine dell'esecuzione dell'Area 6, termina l'esecuzione dell'iterazione corrente del ciclo principale. Al successivo aggiornamento del livello di prezzo da parte del broker, verranno eseguite nuovamente tutte le aree appena descritte.

L'Area 7 viene invece eseguita solamente quando l'utente interrompe l'esecuzione dell'applicativo. In tal caso, al fine di poter avere dei file di log separati per la singola esecuzione, si rinominano tutti i file di log (*Dump_MOAA.csv, Dump_MOVA.csv e OrderClosedByStrategy_SymbolName.txt*), concatenando al nome del file il Timestamp corrente di chiusura dell'applicativo. Tale operazione viene eseguita tramite la porzione di block diagram illustrata in Figura 102.

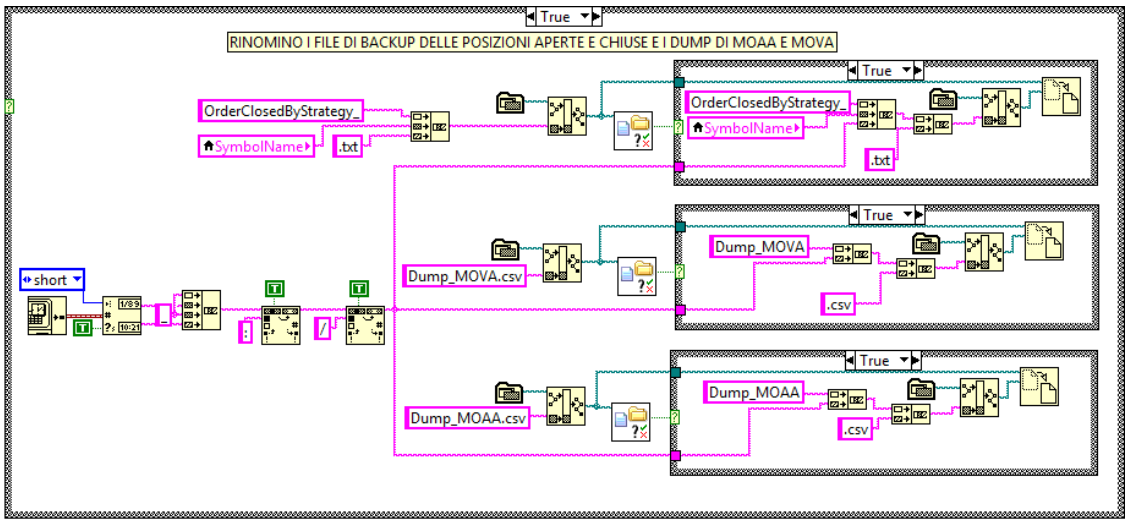


Figura 102 - Ridenominazione dei file di log dell'applicativo al termine dell'esecuzione corrente.

6.7 Interfaccia utente dell'applicativo

Questo paragrafo descrive l'utilizzo dell'applicativo Volatility Calendar. L'applicativo risulta essere suddiviso in cinque schede, ognuna delle quali permette l'interazione con l'utente secondo le diverse funzionalità implementate.

Si anticipa che tutti i parametri per il dimensionamento delle posizioni, l'apertura e la chiusura possono essere modificate a piacimento anche durante l'esecuzione del programma.

Tutti i parametri di Dimensionamento, Apertura e Chiusura posizione hanno già preimpostati dei valori di default per l'esecuzione dell'applicativo.

6.7.1 Scheda 1 – Dim

La Scheda 1 dell'applicativo, di nome *Dim* (v. Figura 103), ovvero "*Dimensionamento*", ha l'obiettivo di permettere all'utente di impostare i parametri di configurazione per l'interfacciamento tra l'applicativo e MetaTrader4, e i parametri di dimensionamento delle posizioni.

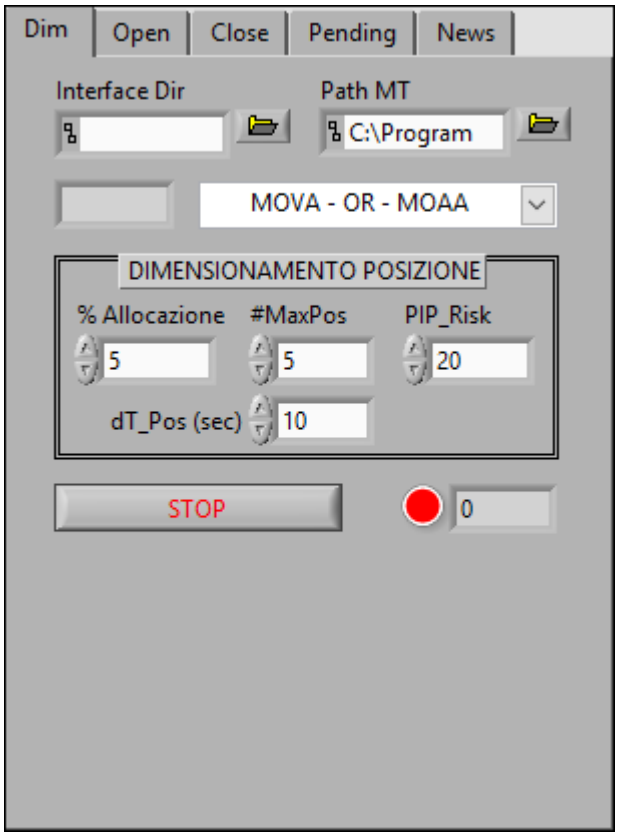


Figura 103 - Scheda Dim per la configurazione delle impostazioni generali e il money management.

Nello specifico, questa scheda prevede che vengano impostati, obbligatoriamente i parametri:

- *Interface Dir*: Path alla directory di interfaccia tra Volatility e la Piattaforma MetaTrader. Tale directory è fondamentale per il dialogo tra i due software per l'apertura e la chiusura delle posizioni;
- *Path MT*: Path alla directory di installazione di MetaTrader. All'interno di questa directory dovrà essere stata preventivamente copiata la DLL di comunicazione tra i due applicativi.

Tutti gli altri parametri, per questa scheda (per i quali è facoltativa la modifica) sono:

- Combinazione degli Engine:
- MOVA
- MOAA
- MOVA OR MOAA
- MOVA AND MOAA
- Per il Dimensionamento della posizione:
- *% Allocazione*: Percentuale di allocazione del margine libero;
- *#MaxPos*: Massimo numero di posizioni aperte;
- *PIP_Risk*: Massimo numero di PIP di rischio ammissibili per ogni posizione aperta;
- *dT_Pos (sec)*: intervallo temporale tra l'apertura di una operazione e la successiva.

Sono inoltre disponibili un indicatore che mostra lo strumento finanziario su cui si sta operando, un indicatore che mostra l'ultimo livello di prezzo ottenuto dal broker e un LED il cui stato indica lo stato dell'applicativo:

- Rosso → Offline
- Verde → Online.

6.7.2 Scheda 2 – Open

La Scheda 2 dell'applicativo, denominata *Open* (v. Figura 104), si riferisce ai parametri che vincolano, in base alla volatilità del mercato, l'apertura di nuove posizioni. Tutti i parametri assumono già dei valori di default, per cui l'utente può decidere di modificarli prima di avviare l'esecuzione dell'applicativo oppure di mantenerli al valore iniziale.

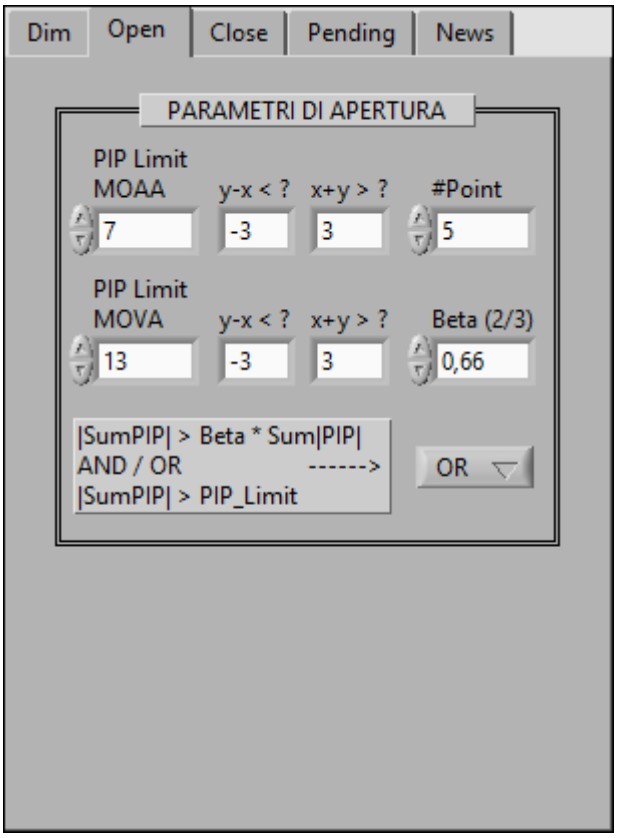


Figura 104 - Scheda Open, per la configurazione dei parametri di apertura

I parametri sono suddivisi per algoritmo, ovvero *MOAA* sulla prima riga, *MOVA* sulla seconda e infine due controlli comuni ad entrambi gli algoritmi. Nello specifico si ha:

Per MOAA, sulla prima riga

- PIP Limit MOAA: parametro di variazione minima di PIP;
- y-x < ? : limite inferiore di coordinate dell'ultimo punto della retta, per il semipiano negativo;
- x+y > ? : limite inferiore di coordinate dell'ultimo punto della retta, per il semipiano positivo.

Per MOVA, sulla seconda riga:

- *PIP Limit MOVA*: parametro di variazione minima di PIP;
- *y -x < ?* : limite inferiore di coordinate dell'ultimo punto della retta, per il semipiano negativo;
- *x+y > ?* : limite inferiore di coordinate dell'ultimo punto della retta, per il semipiano positivo.

Mentre in comune i due algoritmi utilizzano

- *#Point*: numero di livelli di prezzo su cui costruire la retta dei minimi quadrati

- |SumPIP| > Beta * Sum|PIP| AND/OR |SumPIP| > PIP_Limit:
    - *OR*: Vincolo meno restrittivo per l'apertura della posizione;
    - *AND*: Vincolo più restrittivo per l'apertura della posizione.

6.7.3 Scheda 3 – Close

La scheda 3, *Close* (v. Figura 105) si riferisce a tutti i parametri di uscita per le strategie di EXIST implementate all'interno dell'applicativo.

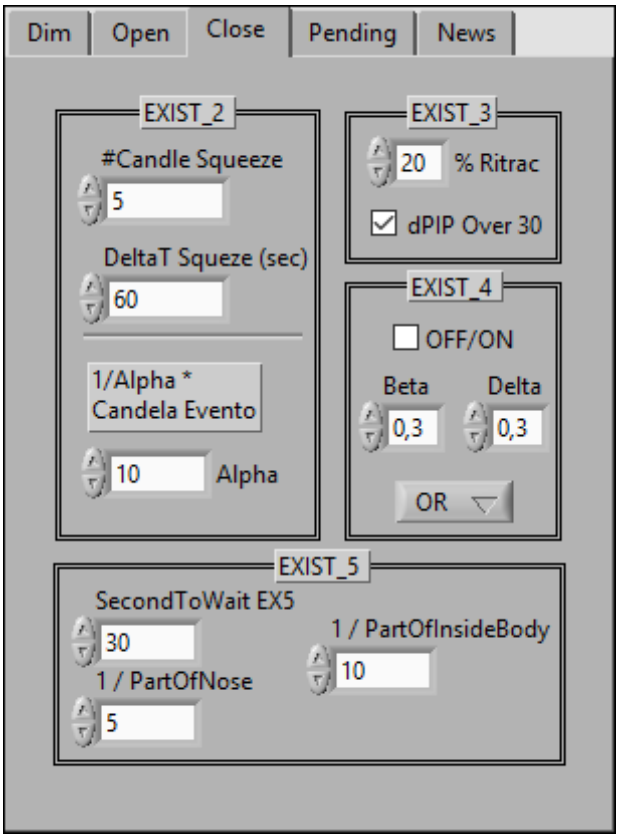


Figura 105 - Scheda Close, per la configurazione dei parametri di chiusura

I parametri per ogni strategia sono suddivisi in subframes. Di seguito si elencano i vari controlli per strategia. N.B. per alcune strategie di uscita non sono previsti parametri modificabili dall'utente.

6.7.3.1 *EXIST 2*

La strategia *EXIST_2* analizza la presenza di una squeeze rispetto all'apertura della posizione

- *#Candle Squeeze*: numero di candele in squeeze post-evento per la chiusura della posizione;
- *DeltaT Squeeze*: tempo in secondi tra la verifica di una squeeze e la successiva;
- *1/Alpha * Candela Evento*: frazione della dimensione della candela dell'evento energetico.

6.7.3.2 *EXIST 3*

La strategia *EXIST_3* verifica una eventuale condizione di ritracciamento del livello di prezzo dello strumento finanziario, ancora all'interno della candela dell'evento. I parametri per questa strategia sono:

- *%Ritracciamento*: percentuale della candela dell'evento oltre la quale si considera verificata la strategia di uscita;
- *dPIPOver30*: se attivato, considera per questa exist solo per movimenti superiori a 30 PIP.

6.7.3.3 *EXIST 4*

La strategia *EXIST_4* si verifica quando, dopo l'evento energetico che ha provocato l'apertura della posizione, si rileva una brusca riduzione della velocità e/o dell'accelerazione dello strumento finanziario, rapportata alla velocità / accelerazione rilevata in corrispondenza dell'evento stesso.

I parametri per questa strategia sono:

- *ON/OFF*: Permette di abilitare o disabilitare la strategia di uscita;
- *Beta*: frazione della velocità al momento dell'apertura della posizione;
- *Delta*: frazione dell'accelerazione al momento dell'apertura della posizione;
- *AND/OR*: controllo per la selezione della modalità di verifica della EXIST, ovvero se deve essere verificata un riduzione della velocità e/o dell'accelerazione.

6.7.3.4 *EXIST 5*

La strategia *EXIST_5* verifica il pattern della candela di apertura della posizione rispetto alle Bande di Bollinger. I parametri sono

- *SecondToWait*: tempo di attesa, dall'apertura della posizione, per la verifica della strategia;
- *1 / PartOfNose*: frazione del "naso" della candela dell'evento;
- *1 / PartOfInsideBody*: frazione del corpo della candela dell'evento.

6.7.4 Scheda 4 - Pending

La scheda 4, *Pending* (v. Figura 106), mostra semplicemente l'insieme di tutti gli ordini aperti dall'esecuzione del software.

Vengono mostrate varie informazioni, quali:

- *Ticket: ID* dell'ordine in MetaTrader;
- *Type*: Tipologia dell'ordine;
  - 1 = Buy (long position);
  - 0 = Sell (short position);
- *OpenPrice*: Prezzo al quale è stato aperto l'ordine;
- *S/L Static*: Stop Loss statico impostato all'apertura dell'ordine;
- *S/L Trailing*: Stop Loss dinamico calcolato ad ogni variazione del livello di prezzo;
- *S/L*: Valore di Stop Loss effettivamente verificato per la chiusura ad ogni variazione del livello di prezzo;
- *DeEvent*: Estensione della candela dell'evento che ha portato all'apertura della posizione;
- *Closed*: Se attivo, indica che la posizione è chiusa, ancora aperta in caso contrario;
- *V_Open*: Velocità calcolata al momento della richiesta di apertura dell'ordine da parte dell'applicativo;
- *A_Open*: Accelerazione calcolata al momento della richiesta di apertura dell'ordine da parte dell'applicativo;
- *EXIST2_TS*: Timestamp relativo alla gestione per la strategia di uscita EXIST_2;

- *Counter*: Indica il numero di candele attualmente identificate per attivare l'uscita per riduzione di volatilità significativa.

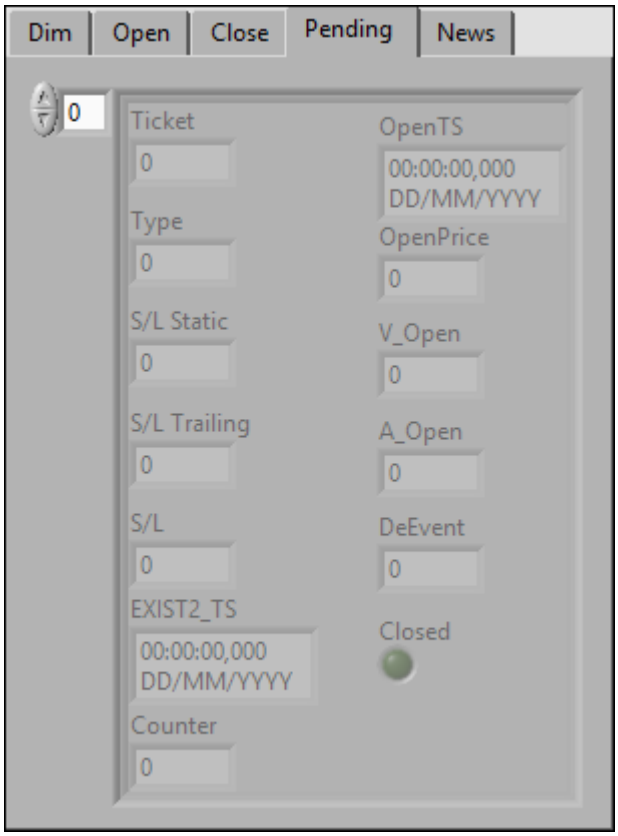


Figura 106 - Scheda Pending, contenente lo storico delle operazioni

6.7.5 Scheda 5 – News

La scheda 5, (v. Figura 107), essendo questa la versione Calendar di Volatility, permette la gestione delle news.

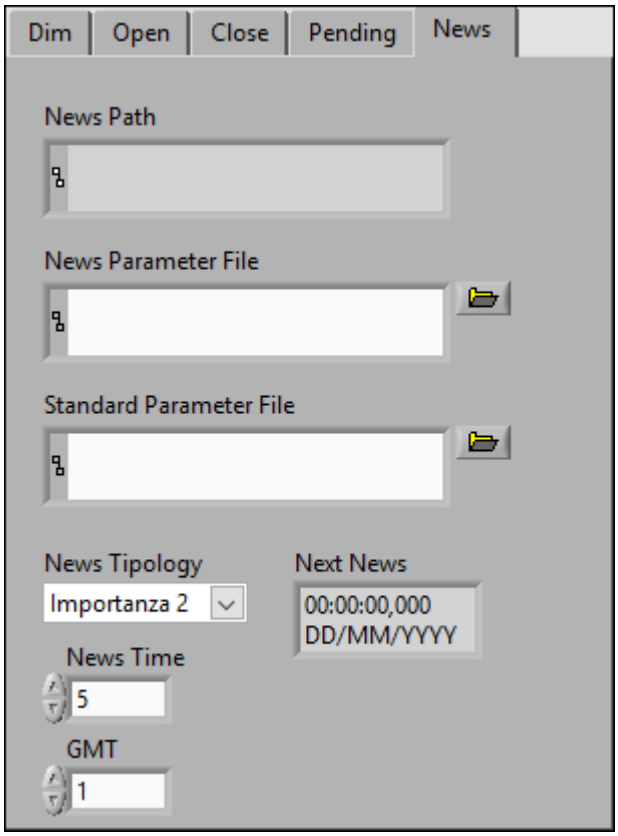


Figura 107 - Scheda News, per la configurazione dei parametri per la gestione delle news

Come è possibile vedere dalla Figura 107, questa scheda prevede che l'utente selezioni:

- *News File*: percorso assoluto al file "Dati-News" contenente la procedura per il download delle news da calendario
- *News Parameter File*: percorso assoluto al file di configurazione dei parametri di apertura delle posizioni in caso di evento News
- *Standard Parameter File*: percorso assoluto al file di configurazione dei parametri di apertura in condizioni di NON-News.
    - *News Tipology*: menu di scelta rapida della tipologia (importanza) delle news di cui l'utente vuole tenere traccia per far operare l'applicativo. Quattro possibili scelte, ovvero *Importanza 1, Importanza 2, Importanza 3, Tutte le news;*
- *News Time*: tempo in minuti antecedenti la news per porre in alert l'appicativo;
- GMT: numero di ore cui si trova la piattaforma rispetto all'ora Italiana, considerata il riferimento temporale per la sincronizzazione delle news rispetto ai dati provenienti dal broker.

7 LoMiTTraS

In questo capitolo viene presentato un algoritmo automatico di trading, capace quindi di aprire e chiudere posizioni a mercato in comunicazione con una piattaforma di trading sulla quale sia attivo un conto di trading.

I'algoritmo *Calendar_LoMiTTraS* (*Calendar_Long and Mid-Term Trading Strategy*), è una strategia pensata per il FOREX market basato sulle bande di Bollinger e sull'apertura di posizioni One Cancels the Other (*OCO*) nella modalità ordinaria, e su aperture direzionali nella modalità con calendario.

Per quanto concerne la comunicazione tra l'applicazione LabView e la piattaforma di trading, si può fare riferimento al paragrafo 3.3 che riporta l'analisi della libreria di comunicazione; tramite le funzioni in essa presenti, infatti, è possibile, all'interno dell'applicazione LabView, interagire con il contro di trading, richiedendo apertura e chiusura di posizioni ed una lunga lista di funzionalità, sempre legate all'operatività.

La scelta di presentare un algoritmo di tale natura è da ricondurre alla presenza, al suo interno, di porzioni di codice ben separate, ciascuna delle quali dedita a svolgere un compito ben specifico. L'analisi del codice di questo applicativo, quindi, fornirà un'idea di come sia possibile strutturare un'applicazione complessa ed organizzarla in maniera modulare e tale da semplificarne la comprensione. In particolare, sarà analizzata sia la struttura complessiva dell'applicazione, sia ciascuna componente fondamentale per la comprensione delle scelte funzionali.

L'idea è quella di presentare l'implementazione di un'applicazione LabView per ambiente Windows che, collegata ad un account di trading, sia capace di interagire con la piattaforma posizionando ordini a mercato in maniera automatica. Come descritto nei capitoli precedenti, occorre anzitutto disporre di un account di trading su una piattaforma, nel caso in esame la piattaforma più utilizzata in assoluto, MetaTrader 4 della MetaQuotes Software Corp., oltre che della libreria di comunicazione tra applicativo e piattaforma, descritta al paragrafo 3.3.1, ed un Expert Advisor che, lato piattaforma, scriva e legga le informazioni di interesse attraverso la libreria. Tale schema di comunicazione risulta essere il più semplice ed efficace, dal momento che la piattaforma in esame, così come tutti gli altri software di trading analizzati, impone alcune limitazioni sul percorso di salvataggio dei file e sull'utilizzo di funzionalità dall'esterno

L'applicazione LabView, al fine di comandare l'apertura o la chiusura di una data posizione, scrive una stringa di controllo precedentemente impostata sui file condivisi; l'Expert Advisor, con frequenza pari al numero di aggiornamenti ricevuti dal server della piattaforma di trading, legge da tali file ed interpreta ciascuna stringa di controllo implementando la corrispondente azione all'interno della piattaforma di trading. La comunicazione, in definitiva, si sostanzia attraverso la lettura e la scrittura su variabili all'interno della DLL per quanto concerne tutte le informazioni ricavate dalla piattaforma di trading e caratterizzanti l'ambiente "mercato finanziario", e la scrittura e lettura di stringhe di controllo su file condivisi per quanto concerne l'esecuzione di istruzioni che implementano l'operatività, quali quelle di apertura e chiusura posizioni.

Al fine di chiarire gli aspetti funzionali ed implementativi del software, si è deciso di descriverlo secondo due modalità, la prima si riferisce alla descrizione dettagliata ma concettuale di come il software è stato modellato e strutturato, la seconda invece, con riferimento alla prima, ne dettaglia le varie sezioni da un punto di vista implementativo. Dopo aver analizzato la suddivisione in settori del sorgente, saranno descritte nel dettaglio le varie sezioni.

L'interfaccia principale "Calendar_LoMiTTraS _MAIN.vi" presenta un cruscotto con controlli ed indicatori, utilizzati sia per ricevere in ingresso parametri per il funzionamento dell'applicazione sia per dare indicazioni sulla corretta esecuzione della strategia, l'esito delle operazioni e l'operatività in generale. La strategia vera e propria, nelle sue due differenti modalità operative (Calendar e LoMiTTraS, a sua volta suddivisa in due differenti versioni, una online ed una offline), è stata implementata separatamente

e richiamata all'interno del vi principale per garantire la modularità del progetto e la riusabilità del codice. Sono presenti inoltre moduli separati che contribuiscono al funzionamento dell'intero applicativo; questi saranno descritti nel dettaglio nei capitoli che seguono.

7.1 Suddivisione in Aree Funzionali

Per garantire la massima modularità, la fase di implementazione ha teso a ripartire il codice in maniera tale da suddividere la logica in aree funzionali, tali da facilitare la comprensione dei meccanismi di funzionamento ed al tempo stesso, facilitandone la manutenzione. La Figura 108 illustra uno schema concettuale di come tale suddivisione sia stata realizzata.

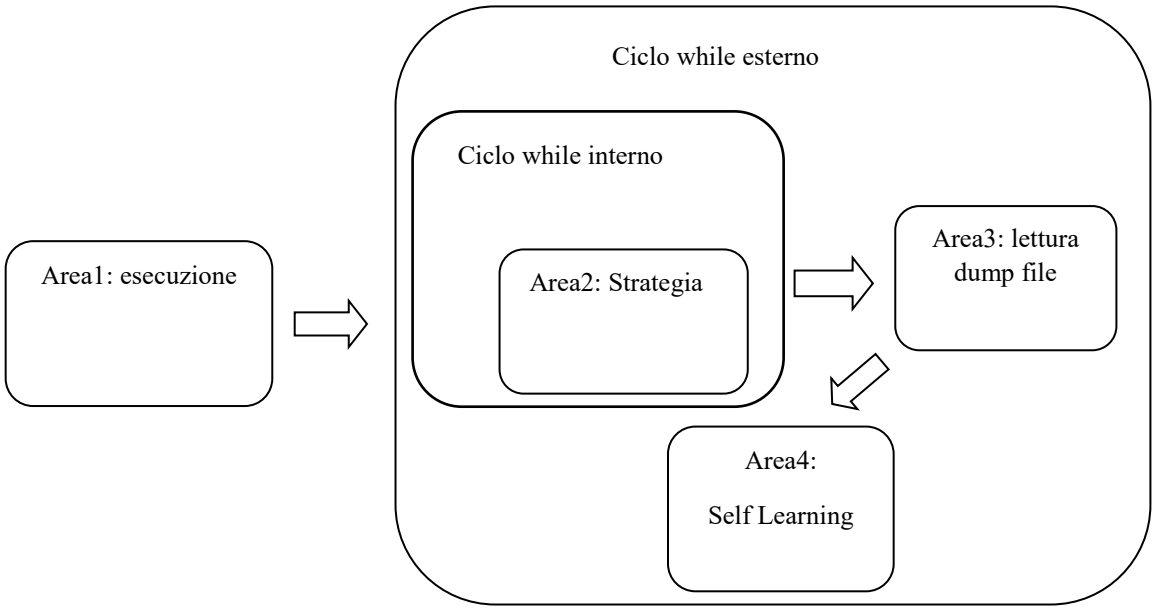


Figura 108 - Suddivisione in aree funzionali.

Ciascuna area è deputata ad una funzione specifica, descritte di seguito.

7.1.1 Area 1:

Porzione di codice per la gestione dell'esecuzione dell'applicativo da riga di comando o eseguito da altro applicativo: in quest'ultimo caso, in luogo dell'inserimento manuale di alcuni controlli necessari al funzionamento, tali parametri vengono letti da riga di comando (o da file), rendendo così possibile automatizzare l'esecuzione di una o più istanze del software (v. Figura 109).

Questa area si trova al di fuori del ciclo while principale di esecuzione, come indicato in figura, proprio perché riguarda delle operazioni che vengono eseguite soltanto all'avvio del software.

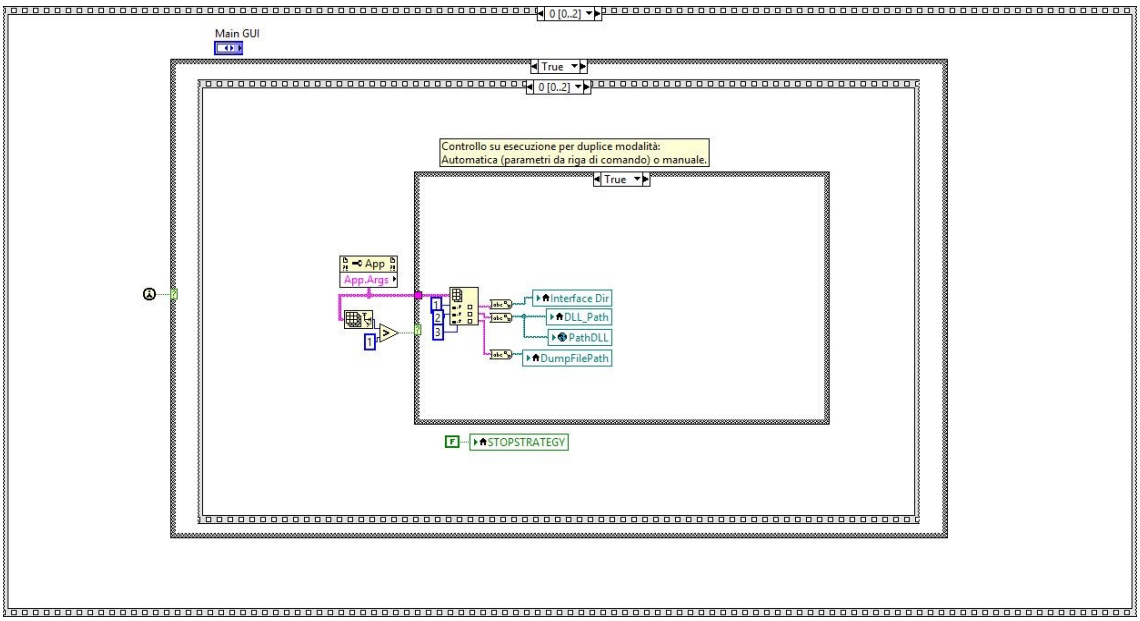


Figura 109 - Area 1 deputata all'esecuzione della strategia in modalità automatica o manuale.

Il ciclo while principale contiene tutta la logica applicativa e termina soltanto quando l'eseguibile viene terminato (anche tramite il bottone "stop" dell'interfaccia).

All'interno del ciclo principale sono presenti le seguenti aree funzionali:

7.1.2 Area 2:

Quest'area è deputata all'esecuzione della strategia di trading collegata al conto sulla piattaforma ed è suddivisa in tre sottosezioni: una di pre-processing (Area 2.1), una di esecuzione della strategia (Area 2.2) ed una di reset di alcune variabili (Area 2.3) (v. Figura 110).

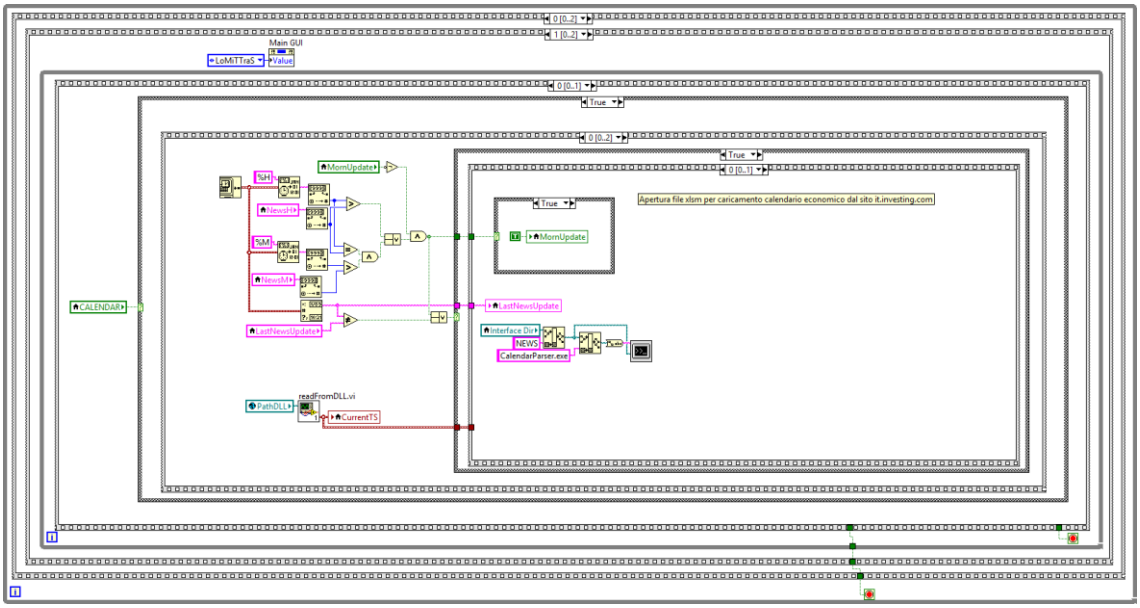


Figura 110 - Area funzionale 2

7.1.2.1 *Area 2.1:*

In questa porzione del codice avviene l'inizializzazione di ciascuna delle variabili (locali e globali) utilizzate dall'applicativo. Tale inizializzazione è utile sia per permettere la corretta esecuzione iniziale, sia per inizializzare nuovamente i valori tra una iterazione e l'altra della strategia al loro valore di default. Nella seconda sezione, invece, avviene il primo interfacciamento con la piattaforma MT4: vengono letti dalla libreria i dati relativi alla coppia strumento finanziario-timeframe su cui è stata istanziata l'esecuzione corrente ed al conto associato. A partire da tale lettura, si attende la ricezione della prossima candela per l'inizio dell'esecuzione della strategia di trading (v. Figura 111).

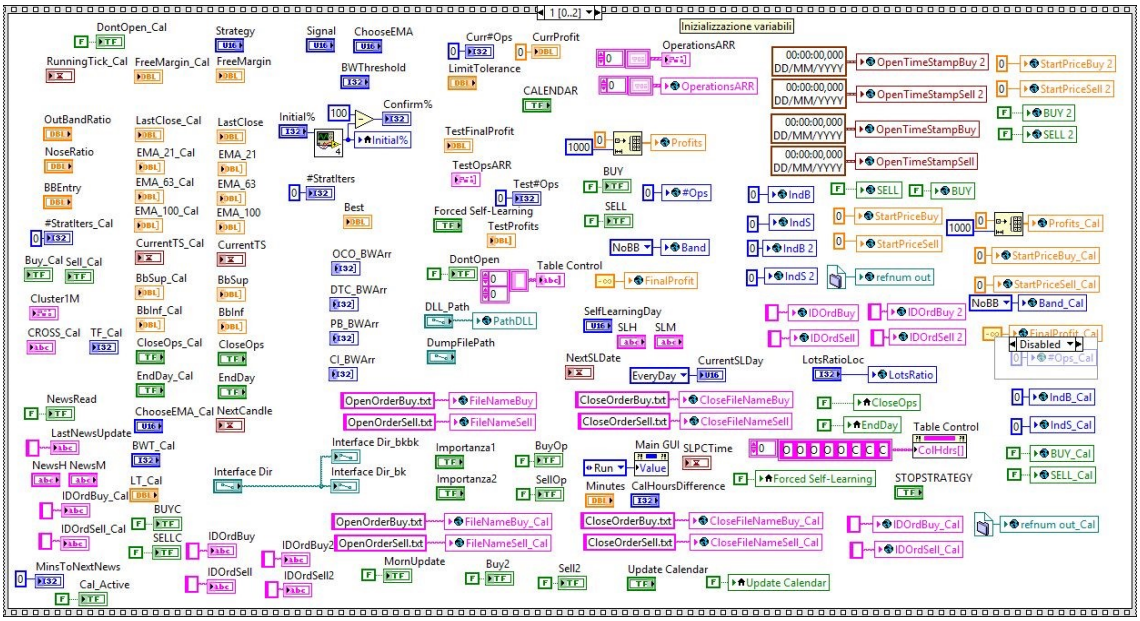


Figura 111 - Area 2.1 atta all'inizializzazione delle variabili

Nel caso si tratti della prima esecuzione in assoluto, vengono inizializzati in maniera esplicita alcuni valori, quali l'indicatore locale di CROSS (su quale strumento finanziario stiamo operando) e timeframe, ed il valore di default del parametro BandWidth Threshold (ovvero il vincolo sulla larghezza minima delle bande di Bollinger per l'attivazione dell'operatività). Infine viene gestita la sospensione delle attività della strategia prevista per permettere l'esecuzione periodica dell'auto apprendimento. Dapprima vengono normalizzati i valori che l'utente ha inserito come scadenza temporale: i minuti vengono mantenuti entro il range [0, 59], le ore entro valori nel range [00, 23]. Successivamente viene controllato il giorno impostato: nel caso esso sia compreso tra le 23 del venerdì e le 23.59 della domenica, esso viene impostato alle 22.59 del venerdì, che corrisponde all'ultima orario di ricezione di aggiornamenti dalla piattaforma prima della chiusura settimanale del mercato. Dopo la fase di inizializzazione, è quindi possibile utilizzare tali parametri in ingresso al vi GetSelfLearningTimeStamp che, a partire dal Timestamp corrente, imposta nella variabile locale del main (output del subvi) NextSLDate il Timestamp corrispondente alla scadenza temporale della prossima iterazione del modulo di Self Learning (SL, auto apprendimento) (v. Figura 112).

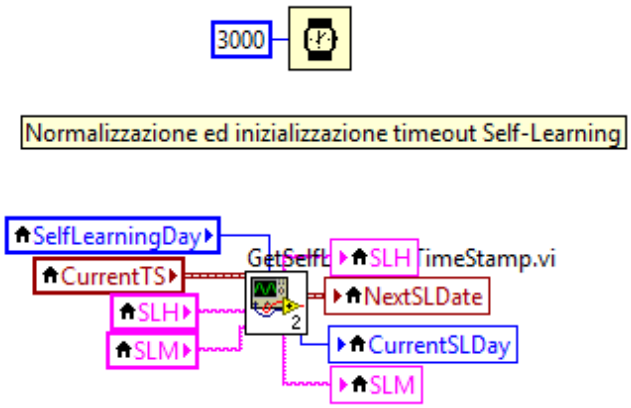


Figura 112 - Inizializzazione del valore per il time-out per la fase di auto apprendimento.

7.1.2.2 *Area 2.2:*

In quest'area è presente il ciclo while interno che termina quando la variabile booleana "EndDay" assume valore "TRUE"; ciò può avvenire perché l'utente dall'interfaccia, in maniera forzata, ha settato a "TRUE" tale variabile, oppure per azione delle condizioni che verificano la terminazione della strategia. L'applicazione sospende l'esecuzione della strategia, ad esempio, al raggiungimento della scadenza temporale impostata dall'utente in interfaccia. Tale Area prevede quattro differenti macro-azioni inserite in un costrutto condizionale legato alla effettiva ricezione di aggiornamenti dalla piattaforma di trading MetaTrader4: in primis vengono letti i valori correnti di prezzo, Timestamp, Bande di Bollinger ed EMA; successivamente vi è il controllo per garantire la tempestiva chiusura delle operazioni correnti prima della sospensione della strategia, quindi viene eseguita effettivamente la strategia attraverso l'esecuzione del sub-vi LoMiTTraS_ONLINE e viene verificata la definitiva sospensione della strategia. Infine vengono aggiornati il profitto complessivo della giornata ed il valore degli indicatori d'interfaccia collegati alle variabili globali e, nel caso la strategia debba essere sospesa, l'esito delle operazioni effettuate viene scritto in una tabella per la visualizzazione sull'interfaccia (v. Figura 113).

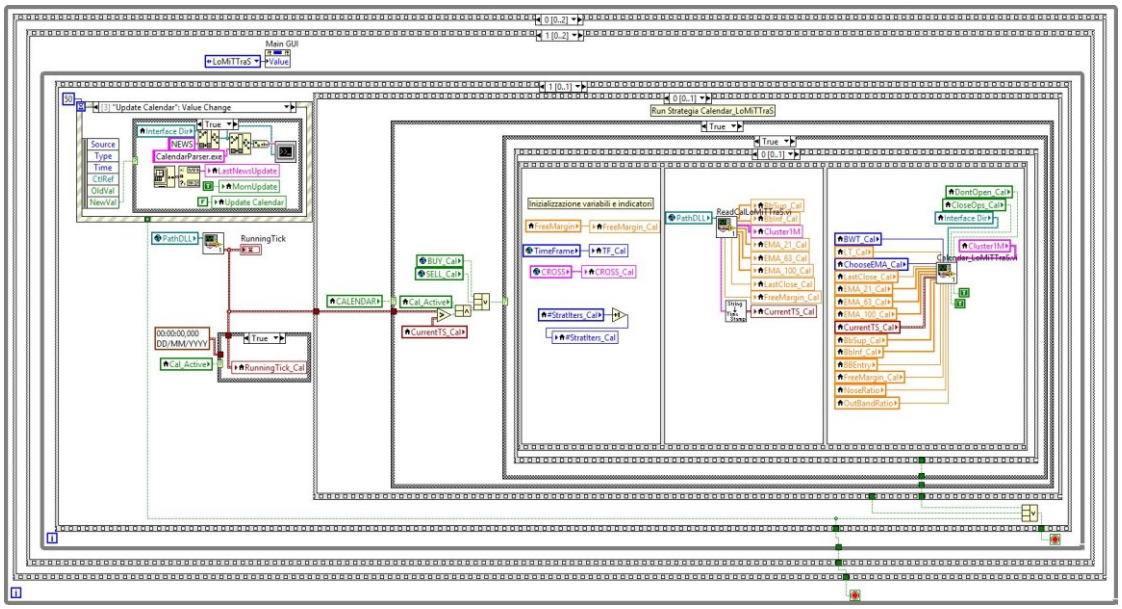


Figura 113 - Area 2.2 destinata all'esecuzione della versione calendar della strategia.

7.1.2.3 *Area 2.3:*

In quest'area di codice vi si accede solo al verificarsi della condizione di terminazione del ciclo while interno di strategia, ovvero quando la strategia è stata sospesa e tutte le operazioni a mercato sono state chiuse. Il codice di quest'area permette l'azzeramento dei valori delle variabili, locali e globali, utilizzate durante l'esecuzione della strategia affinché possano essere correttamente riutilizzate alla prossima esecuzione (v. Figura 114).

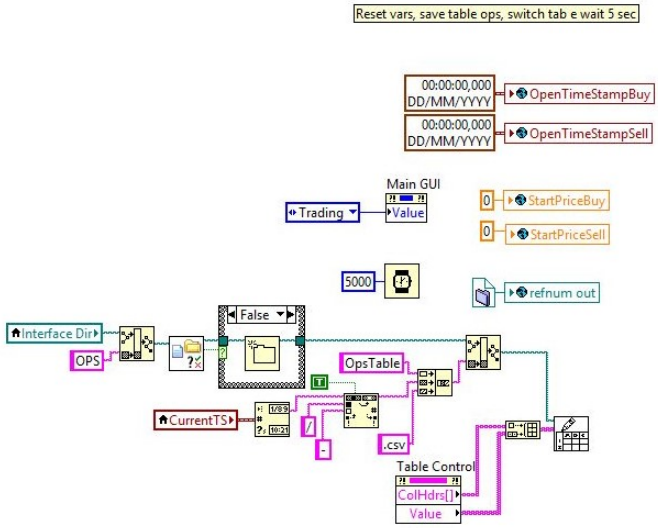


Figura 114 - Area 2.3 per il reset delle variabili di strategia.

7.1.3 Area 3:

In quest'area avvengono le operazioni di preparazione dei dati per il Self Learning, tra le quali la creazione della cartella principale per il Self Learning, l'apertura del file completo di dump ed il suo split in file giornalieri attraverso il sub-vi *DumpSplitter* (v. Figura 115).

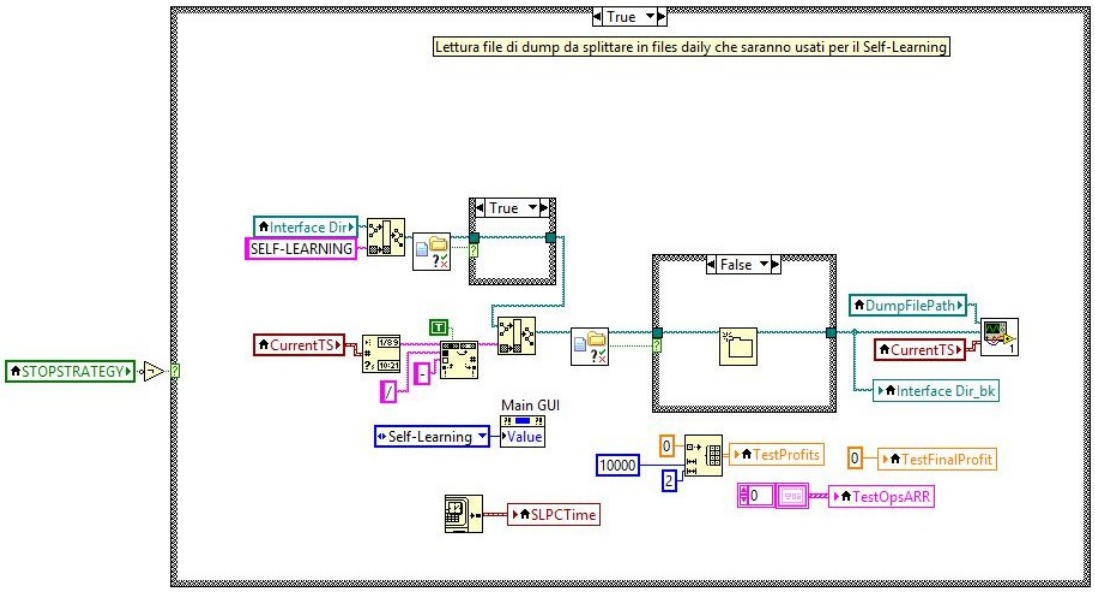


Figura 115 - Area 3 per l'inizializzazione della fase di Self Learning.

7.1.4 Area 4:

L'area 4 è deputata all'esecuzione del Self Learning prima del riavvio della strategia; essa si suddivide in tre sotto aree funzionali, eseguite ciclicamente per ciascun file di dump giornaliero presente nella cartella di lavoro del Self Learning (v. Figura 116).

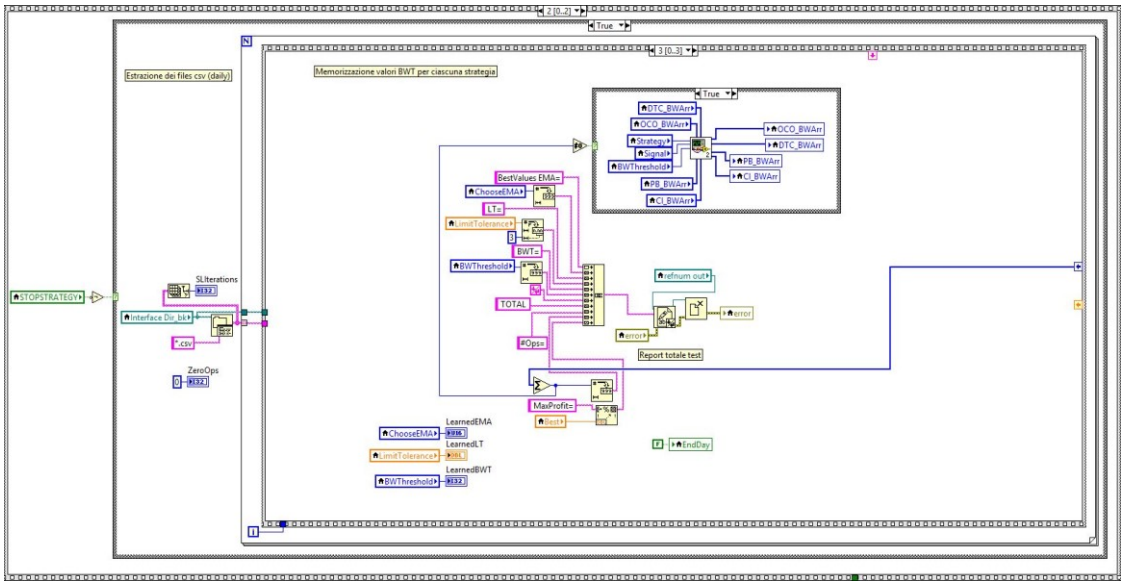


Figura 116 - Area 4 per l'esecuzione della fase di Self Learning.

7.1.4.1 *Area 4.1:*

In questa area viene effettuata la lettura del corrente file di dump giornaliero dal quale vengono estratte tutte le informazioni esso presenti concernenti i valori di prezzo, bande di Bollinger ed EMA. Viene creata una cartella che conterrà la corrente iterazione e viene creato il file di configurazione dei parametri per la prima iterazione del Self Learning. Tale file di configurazione, avente estensione ".def", contiene i valori che ciascun parametro dovrà assumere ai fini del test per l'auto-addestramento. Tale file viene aperto in lettura e l'insieme delle righe lette viene passato in ingresso all'area di codice successiva (v. Figura 117).

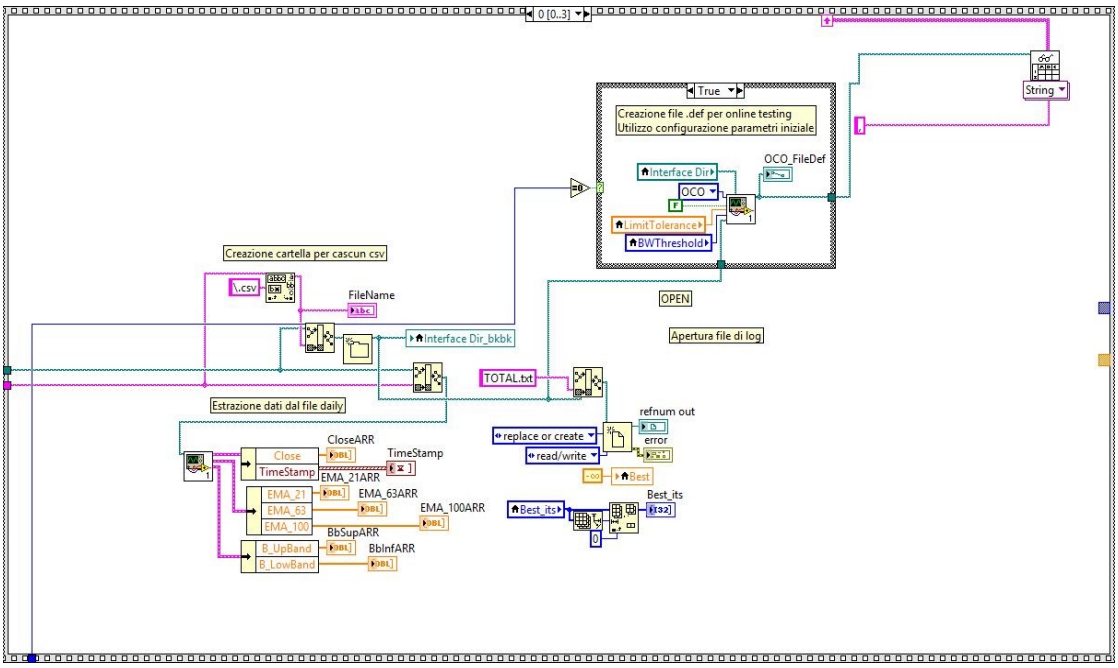


Figura 117 - Area 4.1 deputata alla lettura del corrente dump file giornaliero.

7.1.4.2 *Area 4.2:*

In questa area il codice viene eseguito ciclicamente per ciascuna riga presente nel file di configurazione. Dapprima vengono estratti i valori di ciascun parametro per la corrente iterazione e creata la cartella relativa alla corrente configurazione parametrica; utilizzando tali valori, subito dopo, viene eseguita la strategia nella sua versione offline, tenendo traccia degli esiti in termini di profitto ed aggiornando le informazioni correlate al corrente massimo profitto (v. Figura 118).

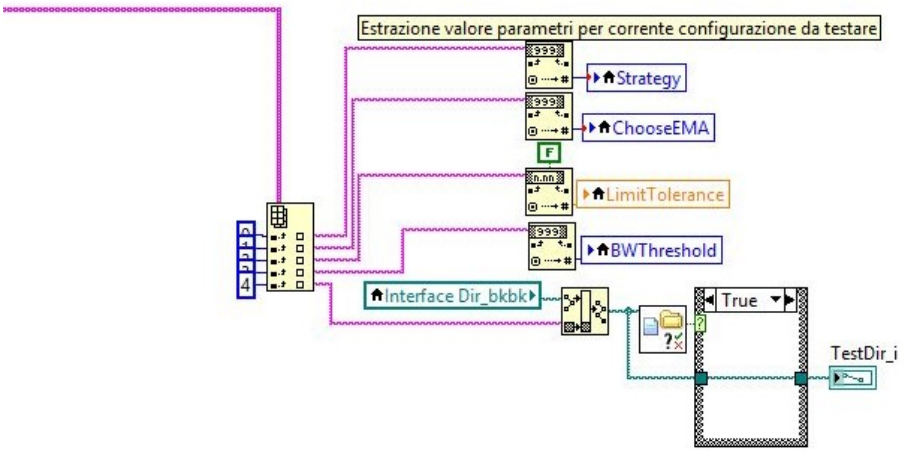


Figura 118 - Area 4.2 per l'esecuzione della versione offline della strategia.

7.1.4.3 *Area 4.3:*

Dopo aver testato ciascun valore di ciascun parametro per il corrente file giornaliero, in questa sezione del codice avviene la scelta dei valori parametrici corrispondenti all'iterazione più proficua in termini di profitto. Successivamente, su un file di report vengono memorizzate alcune informazioni di natura statistica relative all'iterazione corrente (v. Figura 119).

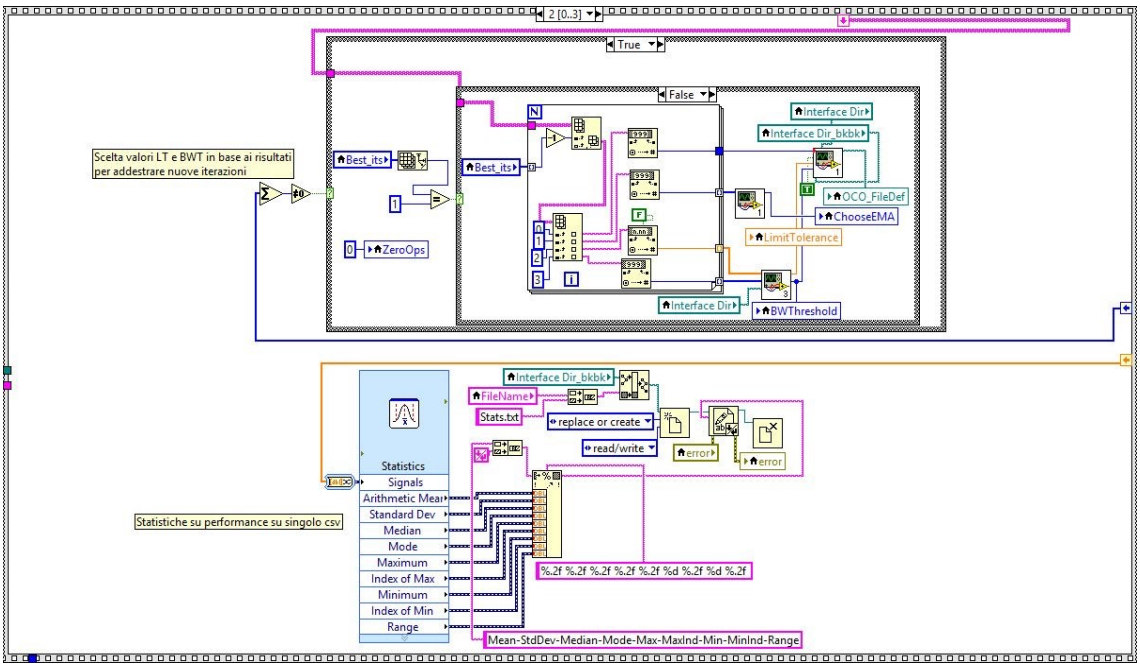


Figura 119 - Area 4.3 per la scelta della migliore configurazione parametrica.

7.1.4.4 *Area 4.4:*

Per concludere la fase di Self Learning, viene dapprima memorizzato il valore che il parametro Band-Width Threshold (BWT) ha assunto in corrispondenza della iterazione più profittevole; successivamente viene scritto su file un breve report riassuntivo degli esiti della corrente iterazione di Self Learning. L'ultima operazione, indispensabile per permettere all'applicativo di ripartire dall'esecuzione iniziale, consiste nel settare nuovamente a "False" la variabile booleana che indicava che era necessario sospendere l'esecuzione della strategia (v. Figura 120).

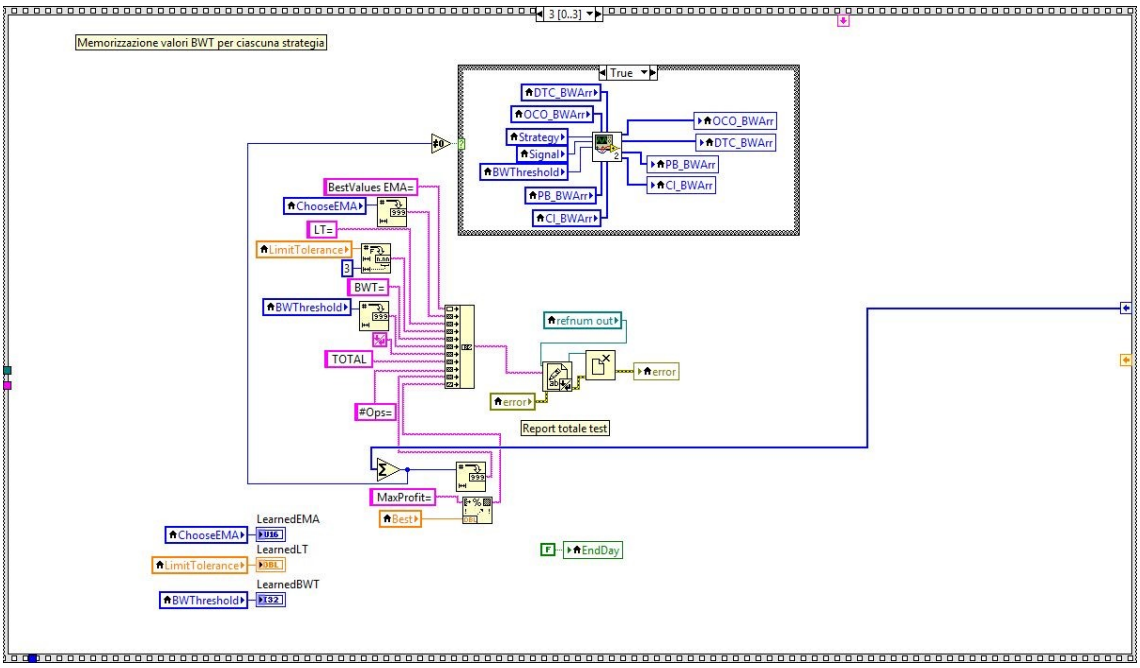


Figura 120 - Area 4.4 per la terminazione della fase di Self Learning.

Con questa operazione il controllo ritorna all'inizio del ciclo while esterno, essendo ancora non verificata la sua condizione di terminazione (chiusura dell'applicativo o attivazione del bottone "Stop"), e l'esecuzione riparte dall'Area 2.

7.2 Core Engine LoMiTTraS

Presentiamo ora una descrizione dal punto di vista teorico della strategia di trading LoMiTTraS nelle sue due versioni, una da eseguire in tempo reale, l'altra con utilizzo di dati precedentemente acquisiti e quindi in modalità offline; proprio per la significativa intersezione funzionale delle due versioni, si procederà dapprima al dettaglio dell'approccio teorico della versione online e, successivamente, per differenze, sarà descritta la versione offline.

La strategia LoMiTTraS si basa sull'utilizzo delle bande di Bollinger e l'apertura di posizioni OCO, ovvero l'apertura contemporanea di due operazioni di "segno" opposto, una Buy ed una Sell, da cui la denominazione OCO "*One Cancels the Other*", nel senso che un'operazione cancella l'altra. In presenza di operazioni aperte, la strategia non aprirà nuove posizioni prima di aver chiuso le operazioni in essere. L'idea di base è quella di utilizzare un segnale, come quello dello sfondamento delle bande di Bollinger, tenendosi pronti contemporaneamente ad un trend rialzista e ribassista; scegliendo opportunamente delle condizioni di limitazione delle perdite (stop loss), si dovrebbe riuscire, nella maggior parte dei casi, a chiudere tempestivamente l'operazione in perdita e a tenere aperta l'operazione in profitto ed in coerenza col trend acquisito dal segnale dopo lo sfondamento delle bande di Bollinger. Altro punto cruciale, come è facile intuire, è la corretta impostazione di condizioni di uscita che massimizzino il profitto (take profit), ovvero che consentano di chiudere l'operazione in maniera profittevole prima che essa cominci a vedere decrementato il suo profitto a causa delle variazioni del prezzo.

7.3 LoMiTTraS_ONLINE

L'algoritmo riceve in input il valore aggiornato di prezzo ed i valori delle bande di Bollinger e delle EMA relativi alle ultime candele chiuse. Tali valori sono stati estratti dall'Expert Advisor (EA) collegato alla piattaforma MetaTrader4 e scritti, tramite la libreria condivisa, in memoria; attraverso le corrispondenti funzioni di DLL, l'applicativo, nel vi principale precedentemente descritto, ha già estrapolato dalla memoria tali valori e li ha posti in ingresso alla corrente strategia; ulteriori input alla strategia sono i

parametri principali per il suo funzionamento, BandWidth Threshold (BWT, soglia sulla larghezza minima delle bande di Bollinger) e Limit Threshold (LT, soglia legata alla distanza in percentuale dalle bande di Bollinger), un controllo che indica quale tipo di media utilizzare (media a 21, 63 o 100 periodi), le informazioni sul conto e la piattaforma ed il percorso al file relativo alla attuale esecuzione dell'applicativo. La strategia non restituisce in output alcun valore ma utilizza variabili locali e globali per la sua esecuzione, sia come input che come output (v. Figura 121).

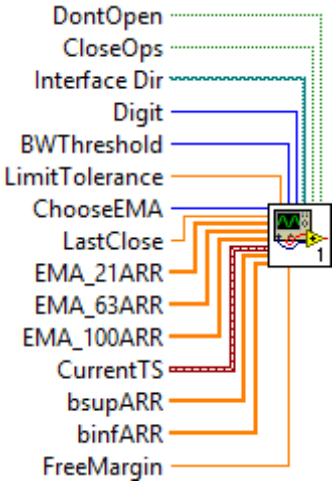


Figura 121 - Input/Output del VI per la versione online della strategia LoMiTTraS.

Nello specifico, gli input che è necessario fornire alla strategia sono:

- *LastClose*: il valore più recente relativo al prezzo.
- *EMA_21, EMA_63* ed *EMA_100*: vettori dei valori delle EMA relativi alle ultime 20 iterazioni.
- *CurrentTS*: il timestamp più recente;
- *BbSup* e *BbInf*: i vettori dei valori delle bande di Bollinger relativi alle ultime 20 iterazioni.
- *FreeMargin*: margine libero del conto.
- *Digit*: numero di cifre decimali significative in relazione al corrente strumento finanziario.
- *Interface Dir*: la corrente cartella di lavoro.
- *CloseOps*: variabile booleana di controllo per indicare, se impostata con valore "True", che ci approssimiamo alla sospensione della strategia ed è quindi necessario chiudere forzatamente tutte le operazioni in essere.

Innanzitutto viene selezionata la corretta media mobile esponenziale (v. Figura 122), da utilizzare per le condizioni di uscita, ovvero di chiusura della relativa operazione; quindi il controllo passa alla strategia vera e propria, che può essere vista come suddivisa in cinque differenti aree funzionali.

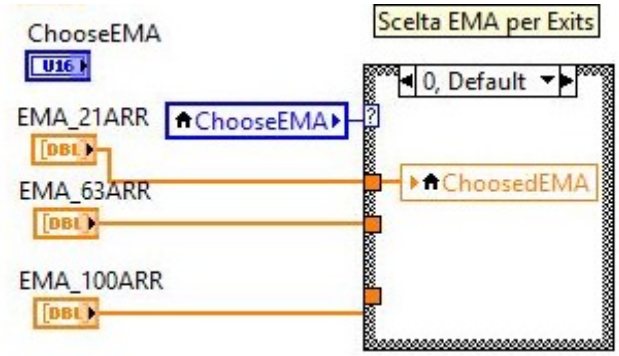


Figura 122 - Selezione media mobile ed inizializzazione variabili

Nella prima area funzionale viene aperto in scrittura (in modalità *append*, ovvero di aggiunta in coda al contenuto già esistente) un file di report, che fornirà un log delle operazioni giornaliere e l'esito delle stesse. Inoltre vengono utilizzati i valori che le bande di Bollinger e la media mobile a 21 periodi hanno assunto in corrispondenza delle ultime venti candele, per calcolare i corrispondenti valori della larghezza delle bande di Bollinger, anche espressa in termini di PIP, ed il valore "%b" corrente, che indica la posizione del prezzo rispetto alle bande di Bollinger (da zero ad uno se interno alle bande). Per ognuna delle strutture dati in cui sono memorizzati i dati delle ultime venti candele, l'elemento di posizione *n* corrisponde alla candela più recente (v. Figura 123).

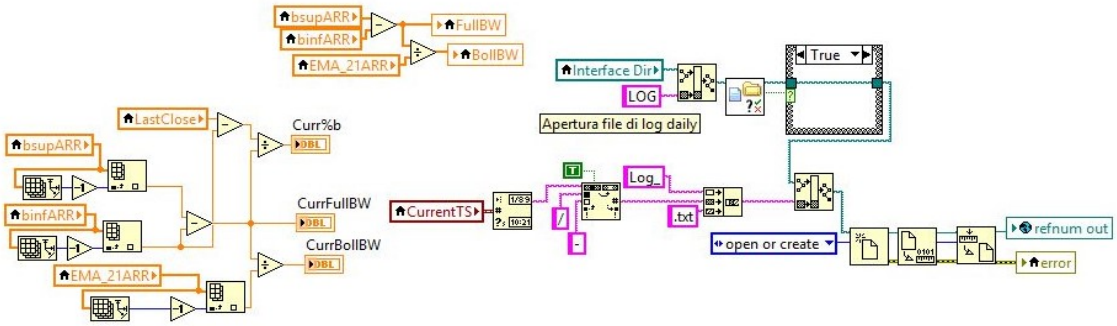


Figura 123 - Accesso al file di log giornaliero e computo bande di Bollinger e relativi parametri.

Dopo aver effettuato le precedenti operazioni di pre-processing, vi è il codice corrispondente alla vera e propria logica della strategia, in quanto deputata alla verifica delle condizioni di apertura della posizione in corrispondenza della banda di Bollinger superiore (v. Figura 124 e Figura 125).

L'area di codice immediatamente successiva è strutturata in maniera speculare, con l'unica differenza di considerare la banda di Bollinger inferiore.

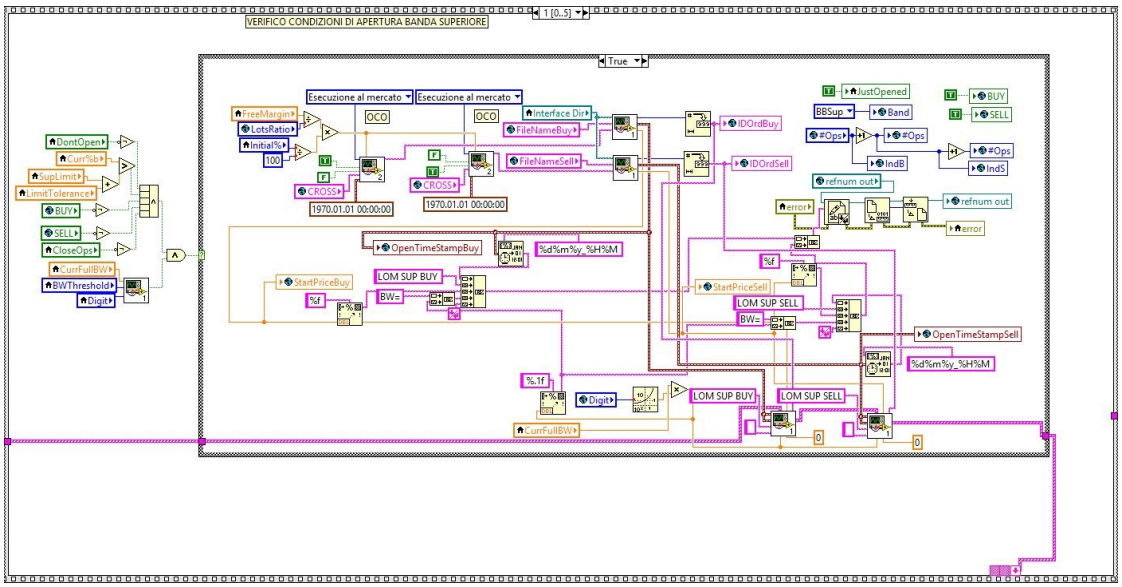


Figura 124 - Verifica vincoli apertura per sfondamento della banda di Bollinger superiore.

Viene verificata quindi la sussistenza delle seguenti condizioni, tutte contemporaneamente (in AND):

- Il corrente valore "%b" sia maggiore del limite superiore delle bande di Bollinger (pari ad 1) incrementato di un fattore additivo equivalente al valore del parametro *Limit Threshold*. Tale condizione può essere sintetizzata come "sfondamento, ad opera del valore del prezzo, della banda superiore di Bollinger".
- Non esiste alcuna operazione Buy a mercato; tale controllo avviene accedendo alla variabile booleana globale "BUY".
- Non esiste alcuna operazione Sell a mercato; tale controllo avviene accedendo alla variabile booleana globale "SELL".
- Il controllo dell'approssimarsi della sospensione della strategia è impostato a "False"; diversamente non viene posta in essere alcuna operazione ma, come dettagliato nel seguito, si pongono in chiusura forzata tutte le operazioni in essere.
- Lo strumento finanziario non si trova in condizione di squeeze (appiattimento del mercato relativo a momenti di volatilità molto bassa): avendo impostato un valore di BandWidth Threshold espresso in PIP, si verifica che la attuale larghezza di banda di Bollinger sia superiore a tale valore di soglia.

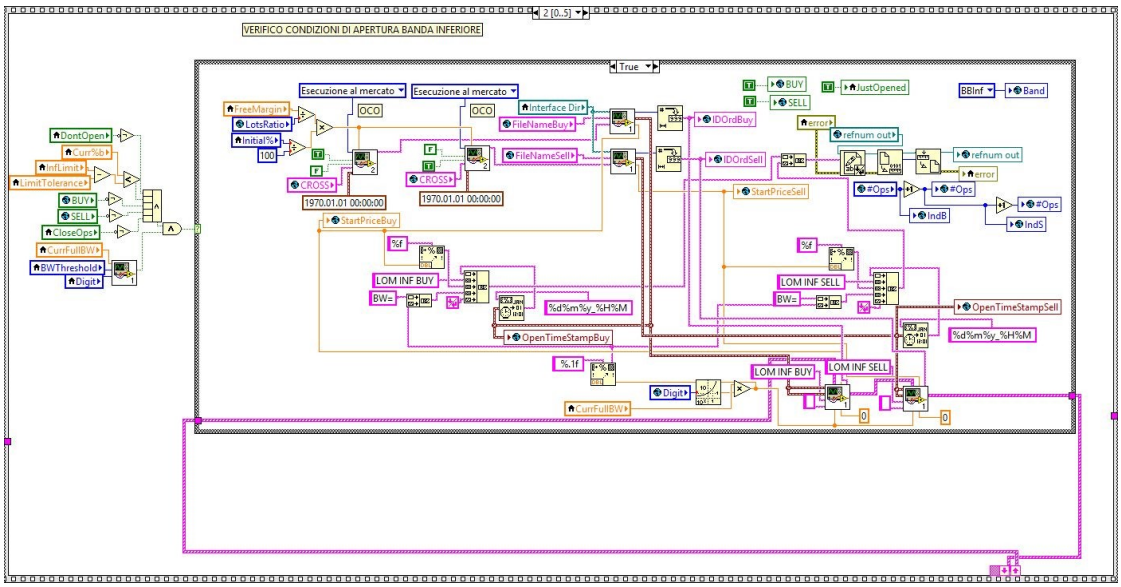


Figura 125 - Verifica vincoli apertura per sfondamento della banda di Bollinger inferiore.

Se tutte le condizioni sopra indicate sono verificate allora la strategia pone in essere le azioni necessarie all'apertura di una posizione OCO sul mercato.

7.3.1 Apertura posizione OCO

La comunicazione tra la strategia e la piattaforma per l'apertura e la chiusura delle operazioni long e short avviene attraverso la scrittura e la lettura di file denominati, rispettivamente, *OpenOrderBuy*, *CloseOrderBuy*, *OpenOrderSell* e *CloseOrderSell*.

Attraverso l'utilizzo del sub-vi *CreateNewOrder.vi* viene preparata la stringa di controllo da scrivere nel relativo file (v. Figura 126); gli input a tale sub-vi sono:

- Il numero di lotti: calcolato come rapporto tra il margine libero attuale ed un coefficiente globale LotsRatio avente valore di default uguale a 5000. Ciò corrisponde, nel caso di un conto con margine libero pari a 100.000, ad aprire posizioni con 20 lotti.
- Strumento finanziario (la variabile globale CROSS).
- Un controllo per indicare se si intende ordinare l'apertura di una posizione long o short attraverso il settaggio a "True" di una delle due corrispondenti variabili booleane;
- Il tipo di operazione di cui richiedere l'apertura: in linea teorica si può richiedere l'apertura di una operazione al mercato o di un'operazione pendente; questa seconda tipologia, tuttavia, non viene mai utilizzata nell'ambito della strategia considerata.
- L'expiration time, settato a zero (o meglio al primo gennaio 1970, trattandosi di una variabile di tipo timestamp), che in realtà non influisce in alcun modo sul funzionamento dell'operazione.

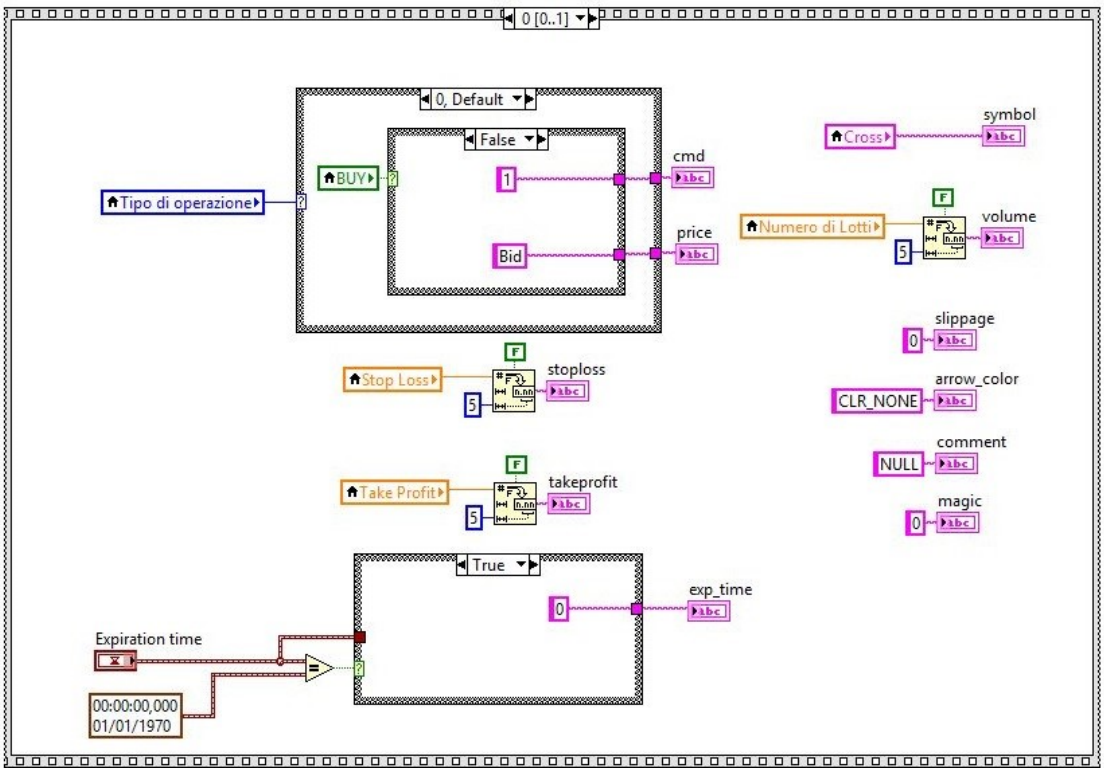


Figura 126 - Scrittura della stringa di controllo per la creazione di un nuovo ordine.

La stringa formata dai valori sopra citati viene così costruita e restituita in output dal vi *CreateNewOrder* ed utilizzata come ingresso al vi *OpStringWriterBuy*, deputato alla scrittura su file e all'attesa della ricezione di un ticket valido di operazione. Nello specifico il sub-vi riceve in ingresso:

- La stringa risultato del vi CreateNewOrder;
- Il path ove salvare il file per l'apertura della posizione ed il nome dello stesso.

Senza entrare nello specifico del sub-vi *OpStringWriterBuy*, che sarà invece analizzato nel capitolo relativo ai vi di uso comune o di supporto, si rende noto in questa fase che l'output di tale vi sarà un ticket valido associato all'operazione, il prezzo di apertura offerto dalla piattaforma ed il relativo Timestamp. Nel caso in cui l'operazione non sia effettivamente eseguita, il controllo rimane in loop (continua ad effettuare cicli operativi) finché non verrà restituito dalla piattaforma un ticket di operazione valido.

Altre operazioni legate all'apertura della posizione ed utili ai fini del corretto funzionamento della strategia sono:

- L'impostazione al valore "True" delle variabili globali BUY e SELL ad indicare che vi sono posizioni aperte sia long che short;
- L'impostazione al valore "True" della variabile di controllo ad indicare che si è appena aperti una posizione e, di conseguenza, non va eseguito il codice relativo alla chiusura delle operazioni;
- L'impostazione della variabile di controllo che indica se si è aperti una posizione a seguito dello sfondamento della banda di Bollinger superiore o inferiore; ciò sarà utile in fase di valutazione degli stop loss;
- Memorizzazione dell'indice di ciascuna delle due operazioni (Buy e Sell) ed incremento del numero totale di operazioni effettuate;

- Scrittura su file (in modalità append) e memorizzazione nel vettore di cluster relativo alle operazioni (*OperationsARR*) dei dati identificativi di ciascuna delle due posizioni appena aperte, quali:
    - Ticket dell'operazione (*IDOrderBuy* ed *IDOrderSell*);
    - Stringa di controllo relativa alla tipologia di posizione aperta (Buy o Sell in corrispondenza della banda di Bollinger superiore o inferiore);
    - Timestamp (*OpenTimeStampBuy* ed *OpenTimeStampSell*) e prezzo (*StartPriceBuy* e *StartPriceSell*) di apertura della posizione;
    - Valore della larghezza di banda di Bollinger espresso in PIP (espressa dalla variabile *CurrFullBW*) al momento dell'apertura della posizione.

Infine il vettore di cluster delle operazioni viene posto in input, così come avviene anche per le altre aree funzionali, alla successiva area di codice.

Come anticipato all'inizio del paragrafo, la successiva area di codice implementa la logica per l'apertura della posizione *OCO* in corrispondenza dello sfondamento della banda inferiore di Bollinger ed agisce in maniera speculare a come descritto per questa area di codice con la differenza che la condizione relativa alle bande è che il valore "%b" sia minore del limite inferiore delle bande di Bollinger (zero) decrementato del valore di tolleranza (per rilevare gli sfondamenti delle bande anche quando non il prezzo non raggiunge necessariamente la banda di Bollinger).

7.3.2 Verifica delle condizioni di chiusura (Stop Loss)

In quest'area vengono verificate le condizioni di chiusura delle operazioni in relazione al verificarsi di specifiche condizioni; i casi considerati sono complessivamente quattro e riguardano le operazioni Buy e Sell aperte in corrispondenza dello sfondamento delle bande di Bollinger inferiore e superiore (v. Figura 127, Figura 128, Figura 129, Figura 130).

Nel merito, nel caso vi sia attualmente un'operazione Buy aperta in corrispondenza dello sfondamento della banda superiore di Bollinger, viene decisa la sua chiusura al verificarsi delle seguenti condizioni:

- Che il corrente valore "%b" sia inferiore o uguale al limite superiore delle bande di Bollinger decrementato di un valore pari a Limit Tolerance;
- Vi sia effettivamente un'operazione Buy aperta, attraverso la verifica che la variabile booleana globale "BUY" abbia valore True;
- L'ultima posizione OCO sia stata aperta in relazione allo sfondamento della banda superiore di Bollinger;
- Non abbia aperto tale operazione nell'iterazione corrente della strategia, per evitare l'apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale *JustOpened*.

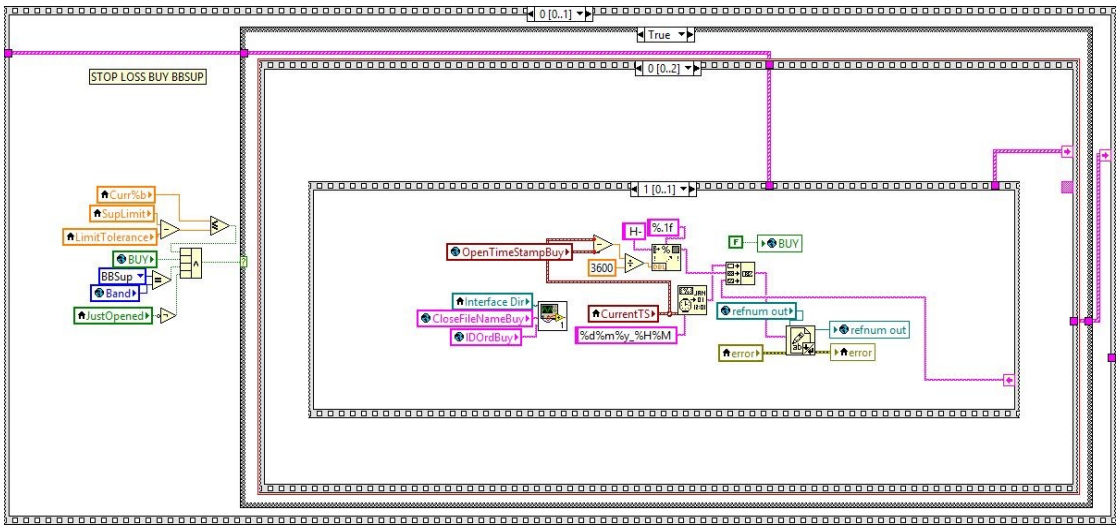


Figura 127 - Verifica condizioni di chiusura per stop loss delle posizioni long in corrispondenza della banda superiore di Bollinger.

Per quanto concerne le operazioni Buy aperte in corrispondenza della banda inferiore di Bollinger le condizioni sono:

- Che il corrente valore "%b" sia inferiore o uguale al limite inferiore delle bande di Bollinger decrementato di un valore pari a Limit Tolerance;
- Vi sia effettivamente un'operazione Buy aperta, attraverso la verifica che la variabile booleana globale "BUY" abbia valore True;
- L'ultima posizione OCO sia stata aperta in relazione allo sfondamento della banda inferiore di Bollinger;
- L'operazione non sia stata eseguita nell'iterazione corrente della strategia, al fine di evitare l'apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale *JustOpened*.

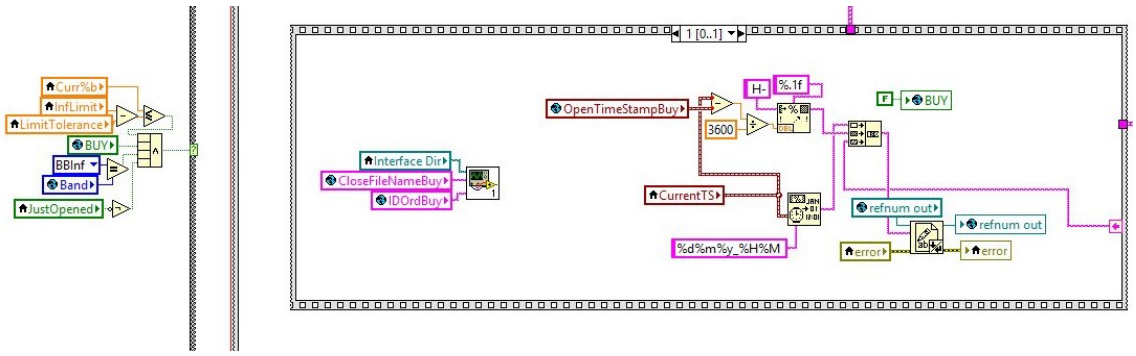


Figura 128 - Verifica condizioni di chiusura per stop loss delle posizioni long in corrispondenza della banda inferiore di Bollinger.

In relazione alle operazioni short (sell), nel caso vi sia attualmente un'operazione sell aperta in corrispondenza dello sfondamento della banda superiore di Bollinger, viene decisa la sua chiusura al verificarsi delle seguenti condizioni:

- Che il corrente valore “%b” sia maggiore o uguale al limite superiore delle bande di Bollinger incrementato di un valore pari a Limit Tolerance;
- Vi sia effettivamente un’operazione sell aperta, attraverso la verifica che la variabile booleana globale “SELL” abbia valore True;
- L’ultima posizione OCO sia stata aperta in relazione allo sfondamento della banda superiore di Bollinger;
- Non abbia aperto tale operazione nell’iterazione corrente della strategia, per evitare l’apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale *JustOpened*.

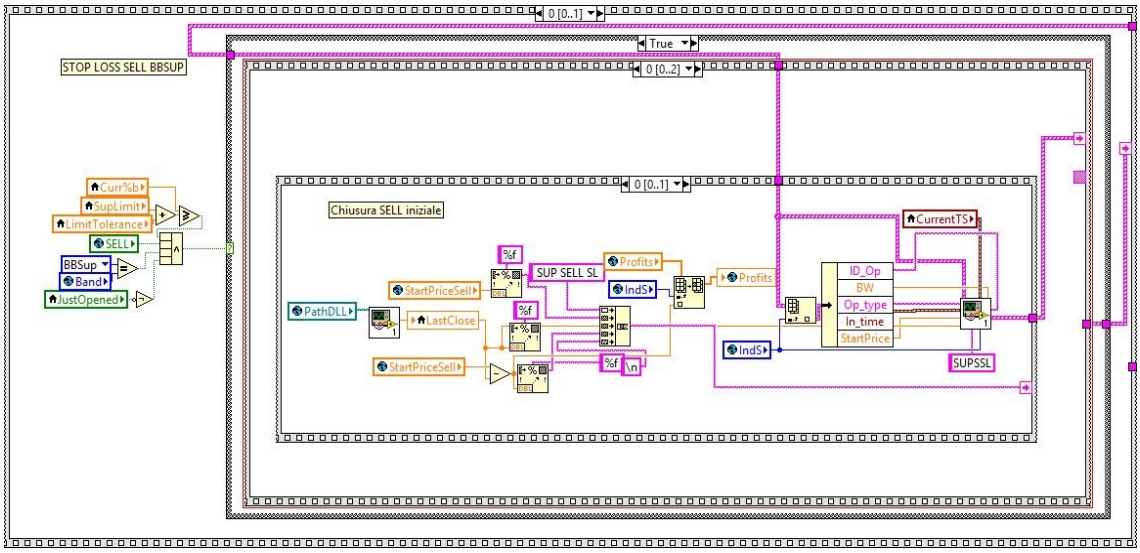


Figura 129 - Verifica condizioni di chiusura per stop loss delle posizioni short in corrispondenza della banda superiore di Bollinger.

Per le operazioni sell aperte in corrispondenza della banda inferiore di Bollinger le condizioni sono:

- Che il corrente valore “%b” sia maggiore o uguale al limite inferiore delle bande di Bollinger (zero) incrementato di un valore pari a Limit Tolerance;
- Vi sia effettivamente un’operazione sell aperta, attraverso la verifica che la variabile booleana globale “SELL” abbia valore True;
- L’ultima posizione OCO sia stata aperta in relazione allo sfondamento della banda inferiore di Bollinger;
- Non abbia aperto tale operazione nell’iterazione corrente della strategia, per evitare l’apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale “JustOpened”.

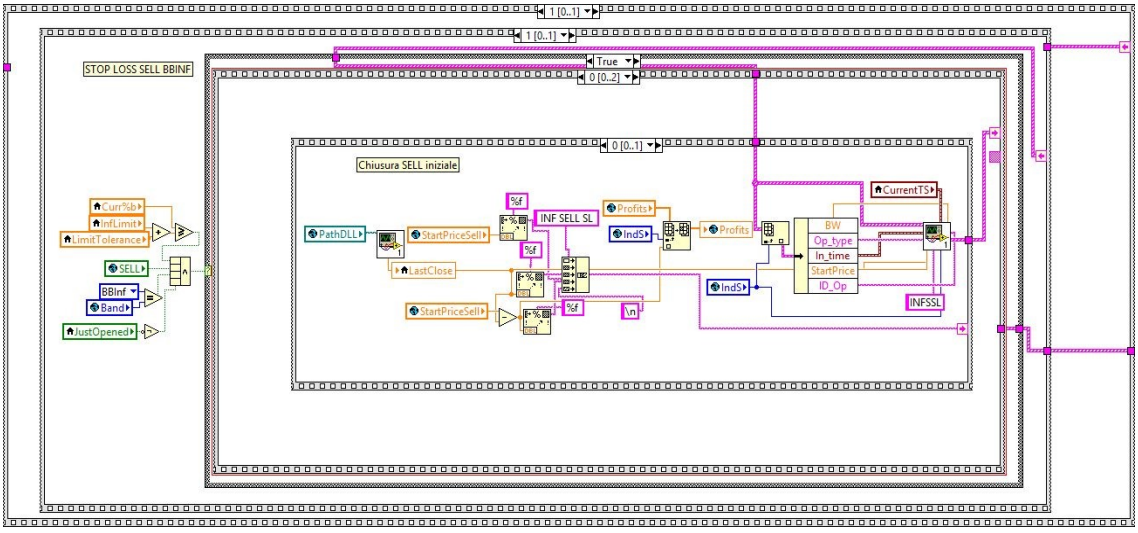


Figura 130 - Verifica condizioni di chiusura per stop loss delle posizioni short in corrispondenza della banda inferiore di Bollinger.

7.3.3 Verifica delle condizioni di chiusura (Take Profit)

In quest'area avviene la verifica delle condizioni di chiusura delle operazioni Buy e Sell finalizzato alla massimizzazione dei profitti o a ridurre le perdite nei casi non regolamentati dalle precedenti strategie di uscita per stop loss (v. Figura 131, Figura 132, Figura 133, Figura 134). In particolare, viene richiesta la chiusura dell'operazione Buy aperta in seguito allo sfondamento della banda superiore di Bollinger al verificarsi delle seguenti condizioni:

- Che l'operazione sia stata eseguita successivamente allo sfondamento della banda superiore di Bollinger;
- Che il valore della larghezza di banda di Bollinger relativo alle ultime tre iterazioni (la t-esima (corrente), la (t-1)-esima e la (t-2)-esima) sia decrescente;
- Vi sia effettivamente un'operazione Buy aperta, attraverso la verifica che la variabile booleana globale *BUY* abbia valore True;
- Che il valore della media mobile adoperata (a 21, 63 o 100 periodi) relativo alle ultime tre iterazioni (la t-esima (corrente), la (t-1)-esima e la (t-2)-esima) sia decrescente;
- Non abbia aperto tale operazione nell'iterazione corrente della strategia, per evitare l'apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale *JustOpened*.

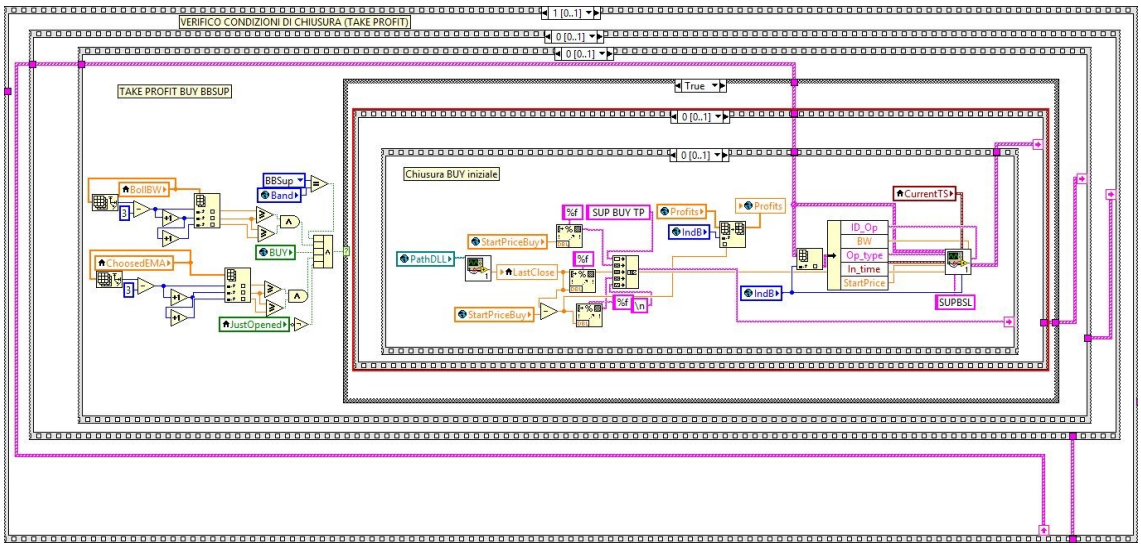


Figura 131 - Verifica delle condizioni di chiusura per Take Profit delle posizioni long aperte a seguito dello sfondamento della banda di Bollinger superiore.

In relazione all'operazione sell aperta in seguito allo sfondamento della banda superiore di Bollinger, ne viene decisa la chiusura al verificarsi di uno dei due (in OR) seguenti insiemi di condizioni (in AND):

- Che valgano tutte le seguenti condizioni:
    - Che l'operazione sia stata eseguita a seguito dello sfondamento della banda superiore di Bollinger;
    - Che il valore della larghezza di banda di Bollinger relativo alle ultime tre iterazioni (la t-esima (corrente), la (t-1)-esima e la (t-2)-esima) sia decrescente;
    - Vi sia effettivamente un'operazione sell aperta, attraverso la verifica che la variabile booleana globale *SELL* abbia valore True;
    - Che il valore della media mobile adoperata (a 21, 63 o 100 periodi) relativo alle ultime tre iterazioni (la t-esima (corrente), la (t-1)-esima e la (t-2)-esima) sia crescente;
    - Non abbia aperto tale operazione nell'iterazione corrente della strategia, per evitare l'apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale *JustOpened*.
- Che siano verificate, in alternativa, tutte le seguenti condizioni:
    - Che l'operazione sia stata eseguita a seguito dello sfondamento della banda superiore di Bollinger;
    - Vi sia effettivamente un'operazione sell aperta, attraverso la verifica che la variabile booleana globale *SELL* abbia valore True.
    - Che il valore del prezzo abbia toccato o sfondato verso il basso la banda inferiore di Bollinger, attraverso la verifica che "%b" sia minore o uguale al limite inferiore di Bollinger;
    - Non abbia aperto tale operazione nell'iterazione corrente della strategia, per evitare l'apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale *JustOpened*.

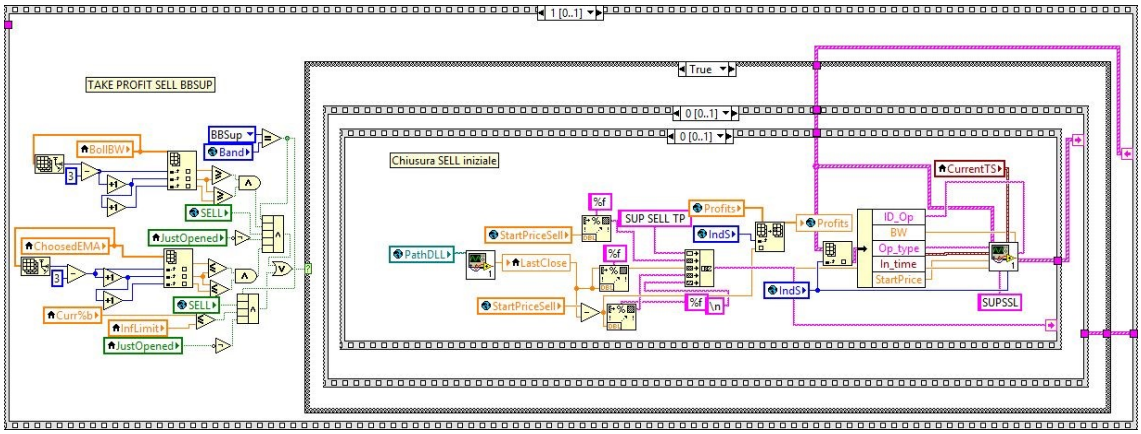


Figura 132 - Verifica delle condizioni di chiusura per Take Profit delle posizioni short aperte a seguito dello sfondamento della banda di Bollinger superiore.

Considerazioni analoghe vengono fatte per la banda inferiore di Bollinger: viene decisa la chiusura dell'operazione Buy aperta in seguito allo sfondamento della banda inferiore di Bollinger al verificarsi delle seguenti condizioni:

- Che sussistano tutte le seguenti condizioni:
  - Che l'operazione sia stata eseguita a seguito dello sfondamento della banda inferiore di Bollinger;
  - Che il valore della larghezza di banda di Bollinger relativo alle ultime tre iterazioni (la t-esima (corrente), la (t-1)-esima e la (t-2)-esima) sia decrescente;
  - Vi sia effettivamente un'operazione Buy aperta, attraverso la verifica che la variabile booleana globale *BUY* abbia valore True;
  - Che il valore della media mobile (EMA) adoperata (a 21, 63 o 100 periodi) relativo alle ultime tre iterazioni (la t-esima (corrente), la (t-1)-esima e la (t-2)-esima) sia decrescente;
  - Non abbia aperto tale operazione nell'iterazione corrente della strategia, per evitare l'apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale *JustOpened*.
- Che siano verificate, in alternativa, tutte le seguenti condizioni:
  - Che l'operazione sia stata eseguita a seguito dello sfondamento della banda inferiore di Bollinger;
  - Vi sia effettivamente un'operazione Buy aperta, attraverso la verifica che la variabile booleana globale *BUY* abbia valore True.
  - Che il valore del prezzo abbia toccato o sfondato verso l'alto la banda superiore di Bollinger, attraverso la verifica che "%b" sia maggiore o uguale al limite superiore di Bollinger (uno);
  - Non abbia aperto tale operazione nell'iterazione corrente della strategia, per evitare l'apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale *JustOpened*.

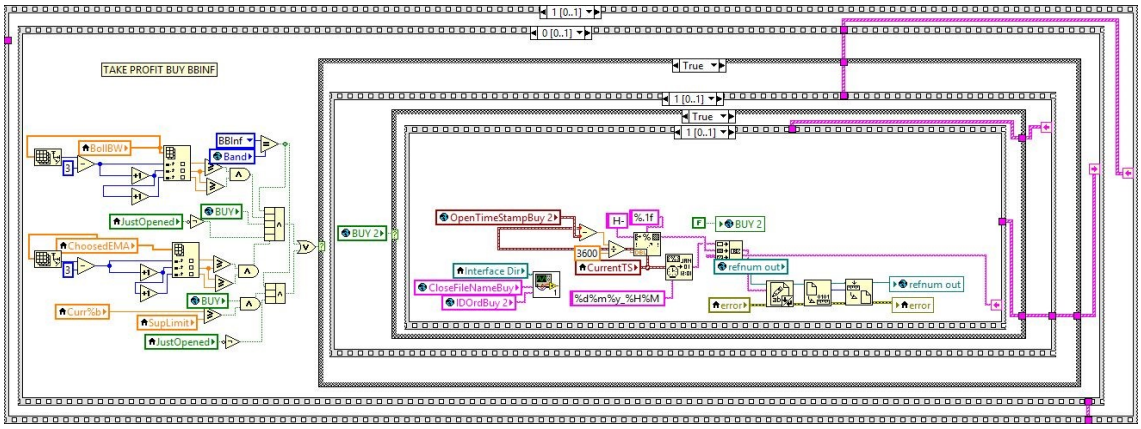


Figura 133 - Verifica delle condizioni di chiusura per Take Profit delle posizioni long aperte a seguito dello sfondamento della banda di Bollinger inferiore.

Infine, viene ordinata la chiusura dell'operazione sell aperta in seguito allo sfondamento della banda superiore di Bollinger se risultano verificate le seguenti condizioni:

- L'operazione è stata eseguita a seguito dello sfondamento della banda inferiore di Bollinger;
- Il valore della larghezza di banda di Bollinger relativo alle ultime tre iterazioni (la t-esima (corrente), la (t-1)-esima e la (t-2)-esima) è decrescente;
- Vi è effettivamente un'operazione sell aperta, attraverso la verifica che il valore della variabile booleana globale *SELL* è uguale a True;
- Il valore della media mobile adoperata (a 21, 63 o 100 periodi) relativo alle ultime tre iterazioni (la t-esima (corrente), la (t-1)-esima e la (t-2)-esima) è decrescente;
- Non abbia aperto tale operazione nell'iterazione corrente della strategia, per evitare l'apertura e la chiusura contestuale di operazioni; tale controllo avviene per mezzo della variabile booleana locale *JustOpened*.

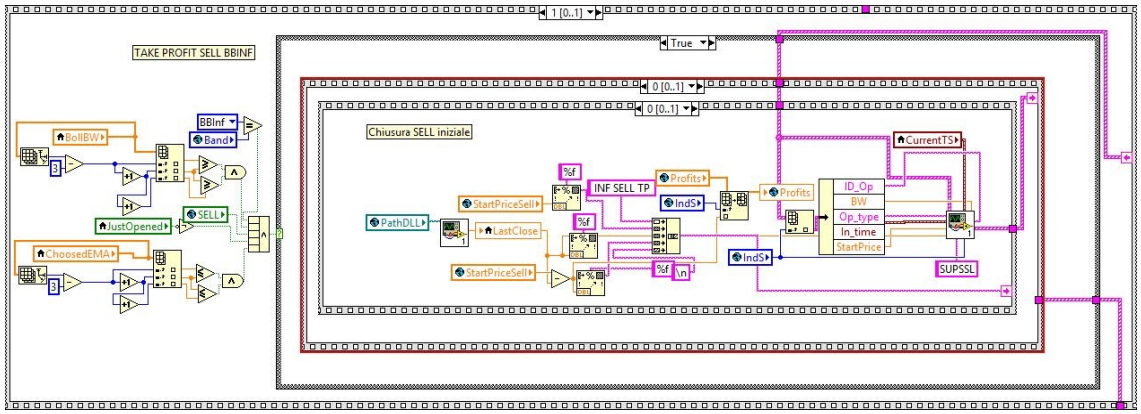


Figura 134 - Verifica delle condizioni di chiusura per Take Profit delle posizioni short aperte a seguito dello sfondamento della banda di Bollinger inferiore.

7.3.4 Verifica delle condizioni di chiusura forzata

Nel caso in cui la variabile booleana di controllo *MarketClosing* in ingresso alla strategia abbia valore True allora è necessario che sia ordinata la chiusura delle operazioni ancora aperte, sia queste Buy o Sell

a causa della incombente chiusura del mercato o, comunque, della necessità di sospendere la strategia (v. Figura 135).

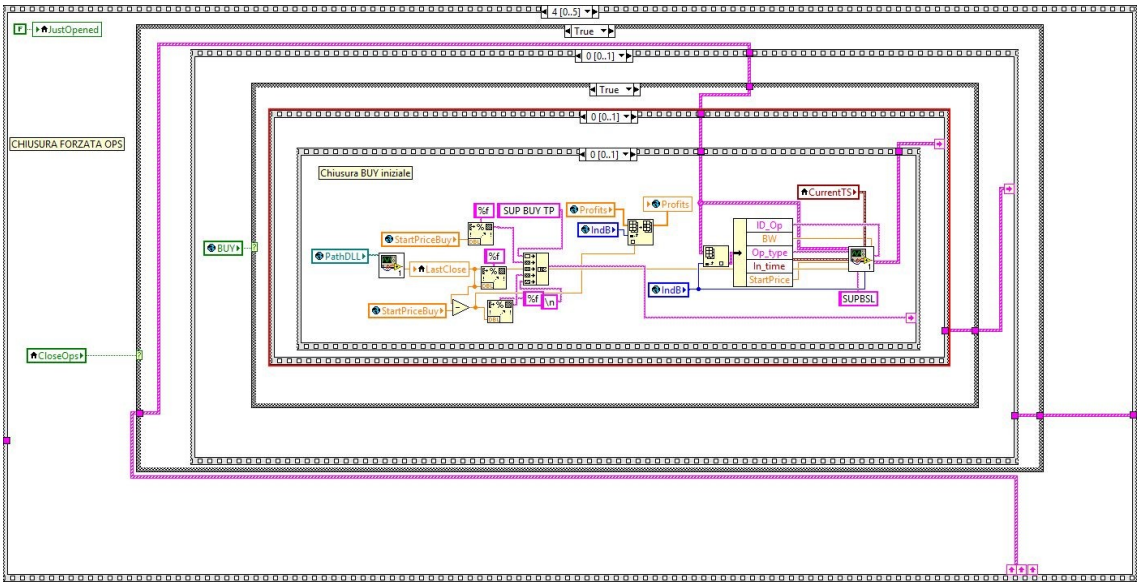


Figura 135 - Verifica delle condizioni di chiusura forzata delle posizioni.

7.3.5 Attivazione modalità di chiusura di un'operazione

In tutti i casi precedentemente esposti, al verificarsi di condizioni ben definite, la strategia ha necessità di comunicare alla piattaforma la richiesta di chiusura della relativa posizione; come per l'apertura di posizioni, la comunicazione con la piattaforma MetaTrader4 per la chiusura delle posizioni avviene mediante la scrittura e lettura di un file di configurazione per ciascuna delle due tipologie di operazione, Buy e Sell.

Nello specifico, per ordinare la chiusura di una posizione vengono eseguite le seguenti operazioni:

- Lettura del valore aggiornato del prezzo (a cui richiedere la chiusura della operazione) mediante invocazione della funzione di libreria;
- Memorizzazione del profitto di operazione nella variabile globale vettore dei profitti *Profits*;
- Memorizzazione nel vettore globale dei cluster delle operazioni *OperationsARR* dei valori relativi alla chiusura delle operazioni; in particolare la entry, inizialmente riempita con i valori relativi all'apertura della posizione, viene completata aggiungendo il Timestamp, il prezzo di chiusura e la motivazione della chiusura, ovvero quale delle strategie di uscita ne ha causato la chiusura;
- Scrittura su file (in append) di report dei dati relativi alla chiusura dell'operazione (tutte le informazioni che sono state anche memorizzate nel cluster *OperationsARR*);
- Scrittura del valore False nella variabile globale booleana relativa all'operazione che si sta chiudendo;
- Reset della variabile globale di controllo delle bande di Bollinger, nel caso in cui non rimanga alcuna operazione aperta a mercato;
- Invocazione del sub-vi *OrderToClose* per la scrittura della entry relativa all'ordine di chiusura sul file di configurazione da cui la piattaforma leggerà la richiesta.

Nella figura che segue è mostrato ad esempio il codice relativo alla chiusura della posizione Buy iniziale (v. Figura 136).

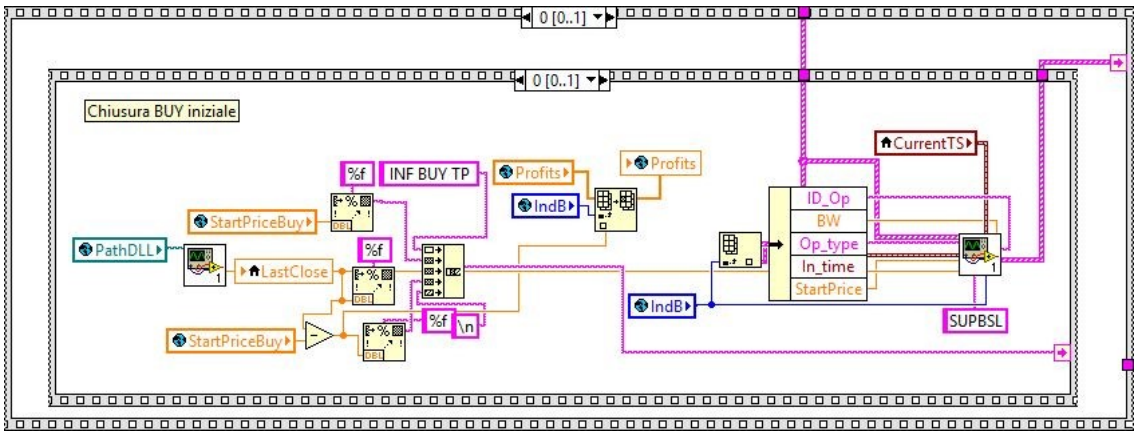


Figura 136 - Chiusura della posizione iniziale long.

7.3.6 Report finale delle operazioni

L'ultima area di codice viene eseguita soltanto se la variabile booleana di controllo *MarketClose* ha valore True e non è stato ancora computato il valore del profitto complessivo della strategia.

Se queste condizioni sono verificate, allora viene computato il profitto complessivo che la strategia ha ottenuto affinché possa essere scritto e chiuso il file di report giornaliero delle operazioni (v. Figura 137).

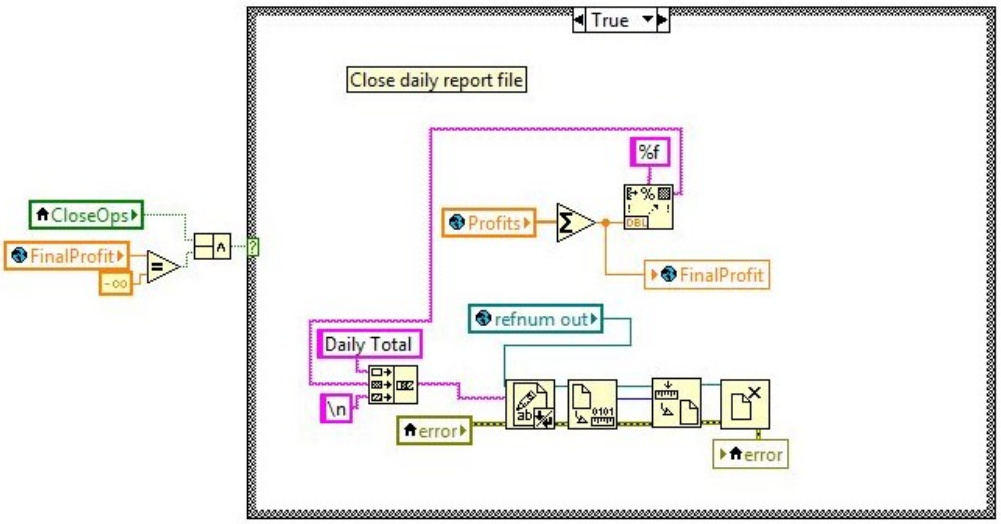


Figura 137 - Aggiornamento e chiusura del daily report file.

7.4 LoMiTTraS_OFFLINE

Per consentire la fase di auto apprendimento (Self Learning) utile ad una maggiore efficienza della strategia nelle iterazioni successive, è stato necessario prevedere una versione della strategia LoMiTTraS opportunamente modificata (v. Figura 138).

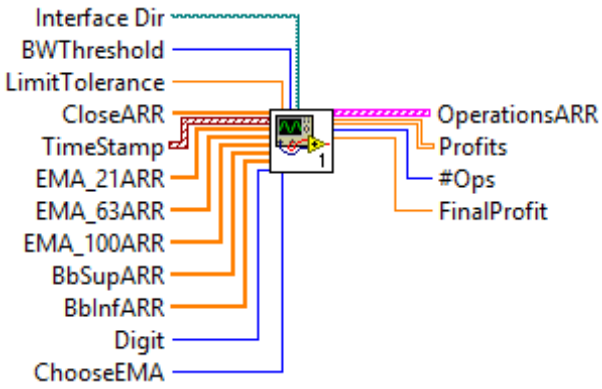


Figura 138 - Input/Output della versione offline della strategia LoMiTTraS.

Ferma restando la logica applicativa descritta nel paragrafo precedente, che sarà inoltre documentata nei dettagli implementativi nel seguito di questo documento, riportiamo qui le principali differenze concettuali tra le due versioni:

- Dovendo operare in post-processing, i dati relativi a prezzo, Timestamp, bande di Bollinger e medie mobili vengono estratti dal file di dump e forniti in input alla strategia sotto forma di vettore. Viene inoltre fornito in input il percorso relativo alla cartella di lavoro.
- Per quanto concerne la verifica delle condizioni di chiusura e di apertura, la strategia, non funzionando in tempo reale, non ha alcun valore "corrente" da utilizzare ma scorrerà iterativamente tutti i valori presenti nei vettori dei dati estratti.
- In corrispondenza del verificarsi delle condizioni di apertura o chiusura, la strategia non implementa la comunicazione con la piattaforma ma memorizza soltanto nel cluster delle operazioni e su file le informazioni sulle operazioni.
- La strategia non scrive sulle variabili globali, anche per evitare sovrapposizioni con la versione online della strategia; essa piuttosto utilizza delle variabili locali, che restituirà in output essendo queste necessarie per alimentare la successiva iterazione del modulo di Self Learning.

7.5 Calendar_LoMiTTraS

Come anticipato, la strategia automatica di trading prevede una duplice modalità operativa: in regime ordinario essa utilizza le bande di Bollinger, le informazioni sulla larghezza di banda ed altri indicatori per l'apertura di posizioni OCO; in corrispondenza di particolari eventi, indicati all'interno di un calendario economico, essa opera invece in maniera differente. In particolare, in questo paragrafo saranno descritti gli aspetti teorici caratterizzanti della strategia legata al calendario e la sua coesistenza con la strategia LoMiTTras, tale da renderle un'unica strategia di trading (v. Figura 139).

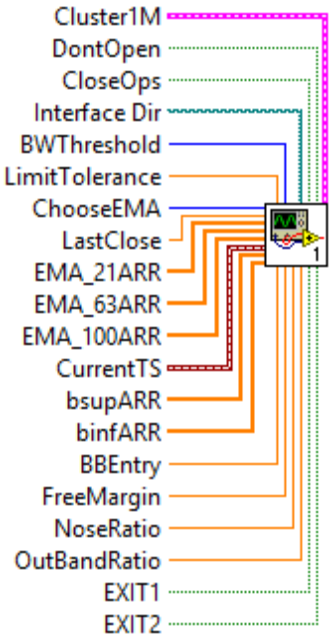


Figura 139 - Input/Output della versione Calendari della strategia LoMiTTraS.

7.5.1 Importazione del Calendario Economico

Occorre anzitutto la disponibilità di un calendario economico che scandisca gli eventi di particolare importanza di ciascuna giornata di trading, indicati in maniera generica con il termine *news*. A tal fine viene impiegato il modulo applicativo *CalendarAlert*, documentato al Paragrafo 4.2 di questo testo, capace di collegarsi alla pagina html online http://it.investing.com/economic-calendar/, leggere e memorizzare il calendario economico in essa presente e, a partire dagli eventi estratti, creare tre diversi file .csv (common separated values, valori separati da virgola) in cui sono state suddivise tutte le news lette per importanza crescente, 1, 2 oppure 3 (corrispondenti ad 1, 2 oppure 3 tori).

Ottenuto il calendario degli eventi finanziari rilevanti per la corrente giornata di trading, occorre che tali eventi vengano memorizzati in opportune strutture dati per poter essere utilizzati all'interno della strategia di trading.

A tal fine, viene utilizzato il vi *CalendarAlert*, che riceve in ingresso i seguenti parametri:

- Percorso alla cartella contenente i file delle news;
- Variabile booleana *RunTimers* che, se a True, attiva dei timer in corrispondenza dell'approssimarsi di una news;
- Variabile double *Mins,* utile a specificare il numero di minuti antecedenti la news per far scattare il timer;
- La stringa *NewsCross*, tramite la quale è possibile introdurre un filtro sulle *currency*;
- Un Timestamp *CurrentTS*, indicante l'istante temporale attuale.

Per maggiori dettagli fare riferimento al paragrafo 4.2.

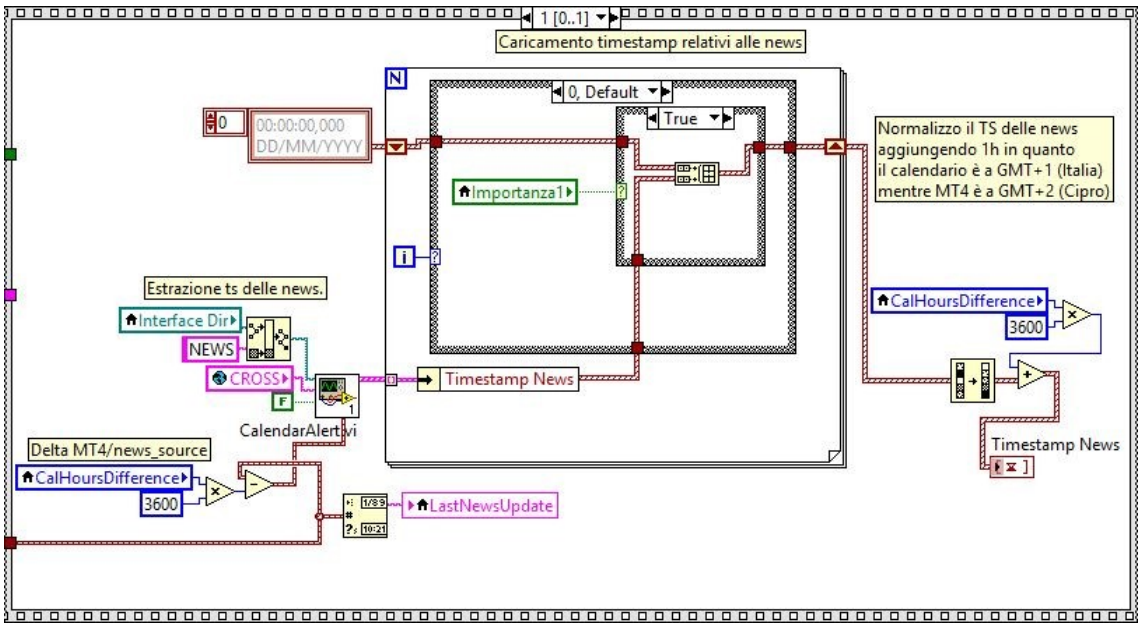


Figura 140 - Area di codice relativa all'utilizzo del sub-vi CalendarAlert per l'utilizzo delle news dal calendario economico.

L'output del vi *CalendarAlert* quindi, sarà un vettore di Timestamp corrispondente agli istanti temporali di ciascuna news. Tale vettore, ordinato in senso cronologico crescente (dal più recente a quello meno recente), sarà la struttura utilizzata dalla strategia per attivare la modalità Calendar (v. Figura 140).

7.5.2 Coesistenza delle modalità Calendar e LoMiTTraS

Una volta ottenuto il calendario degli eventi economici, rappresentato da un vettore di Timestamp contenente l'ora esatta di scheduling di ciascuna notizia prevista in giornata, occorre che la strategia di trading implementi un meccanismo di switch dalla modalità ordinaria a quella Calendar, e viceversa.

A tal scopo, lo switch da una modalità all'altra è implementato con un meccanismo di intervalli temporali, centrati sul Timestamp dell'evento di interesse. E' quindi possibile individuare due diversi intervalli, centrati al tempo zero della news e ciascuno di ampiezza corrispondente al numero di minuti scelti in interfaccia per prepararsi alla news (di default tale valore è di 5 minuti): l'intervallo sinistro [-5 minuti, 0], che parte quindi cinque minuti esatti prima dell'ora prevista dell'eventi, e l'intervallo destro [0, 5 minuti], che è l'intervallo di interesse della news, quello nel quale l'andamento del mercato è fortemente condizionato dal rilascio della news al tempo zero. In corrispondenza di una news in corso, a partire dall'inizio dell'intervallo sinistro fino al termine dell'intervallo destro, all'interno della strategia viene settata una variabile booleana Calendar, ad indicare che è in corso la modalità Calendar.

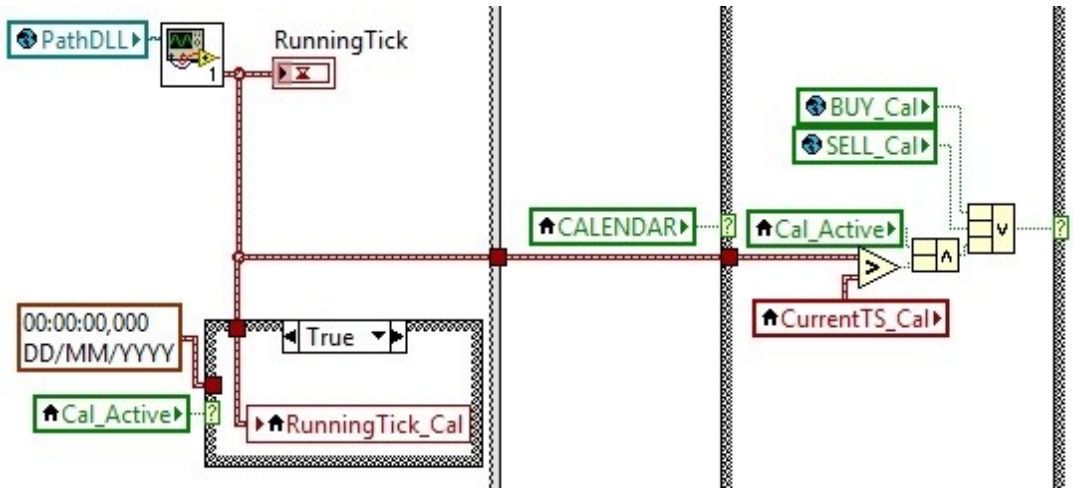


Figura 141 - Controllo per lo switch tra le modalità operative della strategia.

Individuati gli intervalli sinistro e destro centrati nel Timestamp dell'evento, possiamo ora descrivere la modalità di switch della strategia: ad ogni iterazione viene confrontato il primo elemento del vettore di Timestamp delle news *Timestamp News* con il Timestamp corrente *CurrentTS*. L'elemento di indice zero (il primo) del vettore di Timestamp, infatti, corrisponde all'istante temporale previsto per la prossima news. In particolare, viene controllato se, a partire dal Timestamp corrente, manchino 5 minuti o meno alla prossima news prevista; in tal caso, infatti, saremmo nell'intervallo sinistro dell'evento, per questo viene settata a *True* la variabile booleana Calendar, vengono inibite entrambe le modalità LoMiTTraS e Calendar ed aggiornato l'indicatore intero *MinsToNextNews* indicante il numero di minuti mancanti all'evento. L'inibizione delle modalità di strategia corrisponde all'impossibilità di aprire posizioni, mentre tutte le altre funzionalità sono attive (principalmente le operazioni di chiusura posizioni), ed è implementata attraverso l'impostazione a valore *True* delle due variabili booleane *DontOpen* e DontOpen_Cal, riferite rispettivamente alla modalità LoMiTTraS e Calendar. Quando una strategia è inibita, quindi, essa viene invocata normalmente, riceve tutti i dati in ingresso, valuta gli ingressi ma non apre alcuna posizione, piuttosto valuta le uscite e nel caso chiude eventuali posizioni aperte. Svolge comunque tutte le altre funzionalità ordinarie; l'inibizione, in definitiva, si riferisce unicamente all'impossibilità di aprire posizioni (v. Figura 142).

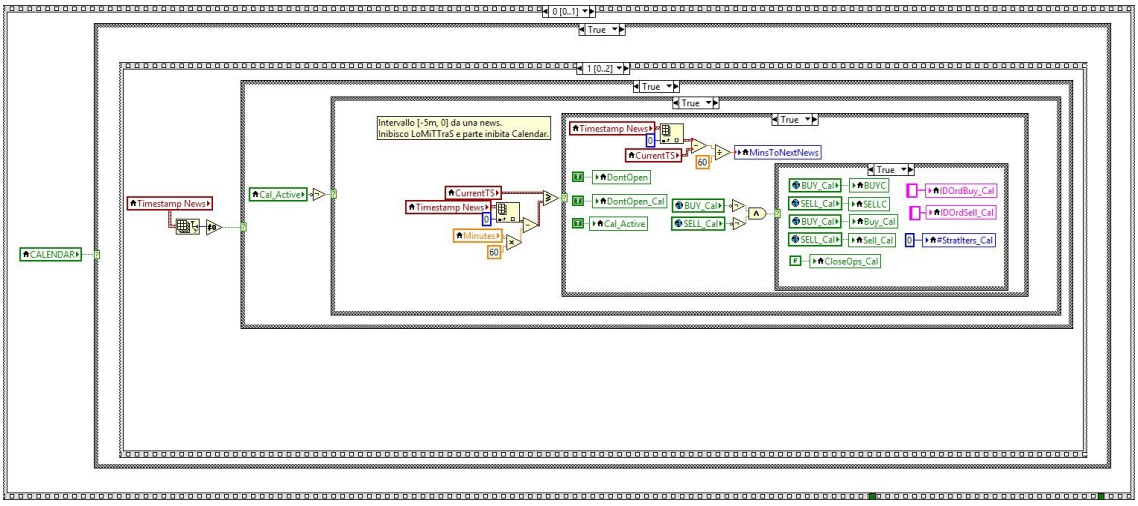


Figura 142 - Controllo degli intervalli temporali per lo switch tra le modalità operative della strategia.

Se invece ci si trova già in regime Calendar, occorre verificare se temporalmente si è già nell'intervallo sinistro; per questo motivo, quando il corrente Timestamp è successivo a quello relativo all'evento di interesse, la strategia LoMiTTraS rimane inibita, la modalità Calendar viene attivata (variabile booleana DontOpen_Cal a false), l'indicatore di minuti mancanti alla news viene resettato a zero e l'interfaccia scorre fino alla scheda relativa alla strategia Calendar (v. Figura 143).

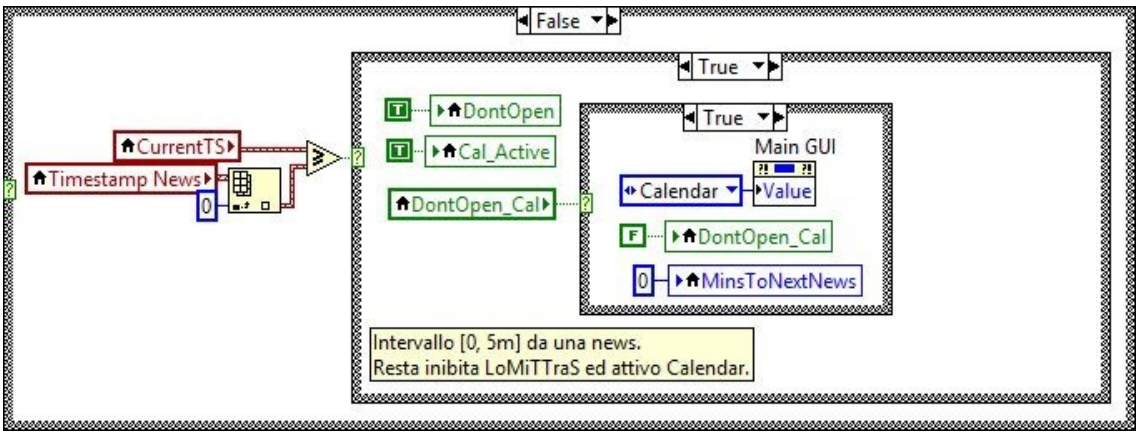


Figura 143 - Controllo per l'attivazione della modalità Calendar.

Quando, infine, termina anche l'intervallo sinistro, ovvero quando *CurrentTS* è maggiore del Timestamp dell'evento incrementato di 5 minuti, occorre ritornare alla modalità LoMiTTraS, per cui vengono eseguite le seguenti operazioni (v. Figura 144):

- Variabile booleana *Calendar* a false;
- Variabile booleana *DontOpen* a false, per riattivare la modalità LoMiTTraS;
- Variabile booleana *DontOpen_Cal* a *True*, ad indicare l'inibizione della modalità Calendar;
- Switch della GUI alla scheda LoMiTTraS;
- Indicatori booleani delle operazioni Buy e Sell in modalità Calendar aggiornati con il valore contenuto nelle variabili booleane globali *BUY_Cal* e *SELL_Cal*;
- Reset delle stringhe *IDOrdBuy_Cal* e *IDOrdSell_Cal*, corrispondenti al ticket delle operazioni Buy e Sell in modalità Calendar;
- Reset della variabile intero *#StratIters_Cal*, indicante il numero di iterazioni della strategia Calendar;
- Ricalcolo del numero di minuti mancanti alla prossima news.

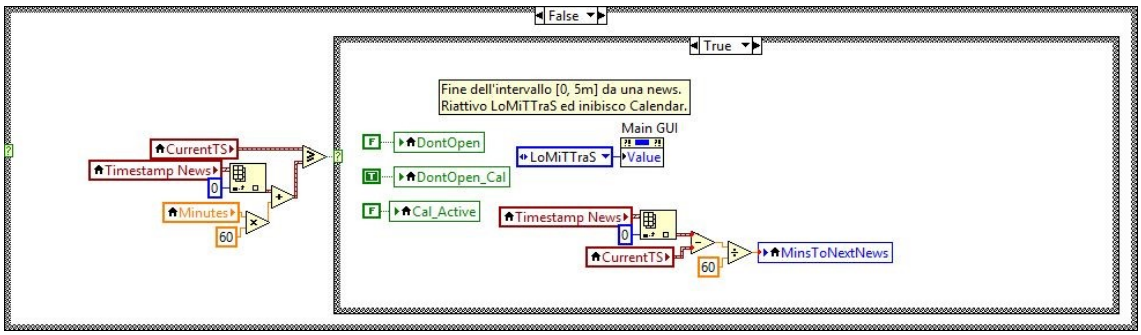


Figura 144 - Controllo per la terminazione della modalità Calendar.

Tutte le suddette operazioni, come è facile intuire, vengono eseguite soltanto in presenza di eventi di calendario, ovvero solo se la dimensione del vettore di istanti temporali delle news è diversa da zero.

Successivamente alla porzione di codice deputata al controllo sullo switch delle modalità di strategia, si trova l'area di codice deputata a tenere costantemente aggiornato il vettore di Timestamp delle news, affinché news obsolete vengano quindi eliminate dal vettore. A tal fine, se il vettore Timestamp News contiene effettivamente degli elementi, se non si è in regime Calendar (in tal caso non bisogna apportare modifiche al vettore) l'applicativo esamina il vettore e confronta ciascun elemento in esso contenuto con il Timestamp corrente, onde rimuovere dal vettore i Timestamp obsoleti. Appena individuato il primo Timestamp non obsoleto, oppure al raggiungimento della dimensione nulla del vettore, si termina la procedura ottenendo così il vettore dei Timestamp delle news aggiornato: in esso saranno contenuti soltanto i Timestamp relativi ad eventi non ancora avvenuti (v. Figura 145).

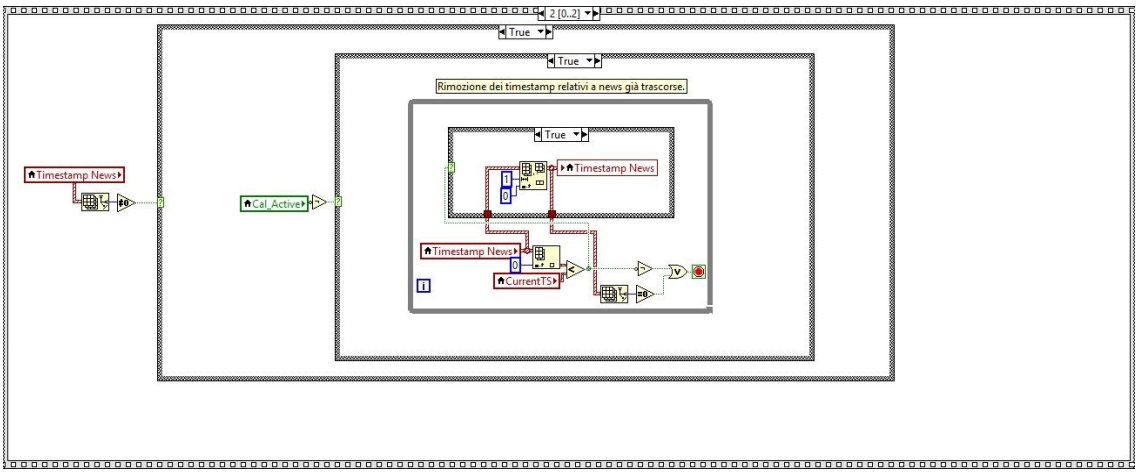


Figura 145 - Codice per la rimozione dei Timestamp relativi a news già passate.

7.5.3 Funzionamento della strategia Calendar_LoMiTTraS

La strategia Calendar_LoMiTTraS viene implementata come modalità operativa Calendar all'interno di un'unica strategia automatica di trading. Essa condivide alcune funzionalità della strategia Calendar, ma differisce da essa profondamente per la verifica delle condizioni di entrata (apertura posizioni), modalità di apertura ordini (non OCO ma direzionale), strategie di uscita (in luogo dei Take Profit essa implementa le due strategie di uscita specifiche, Exit1 ed Exit2, documentate nel seguito). In questo paragrafo saranno descritte tutte le sue funzionalità, trattando in maniera sintetica quelle condivise con la strategia LoMiTTraS e, quindi, già documentate precedentemente (v. Figura 146).

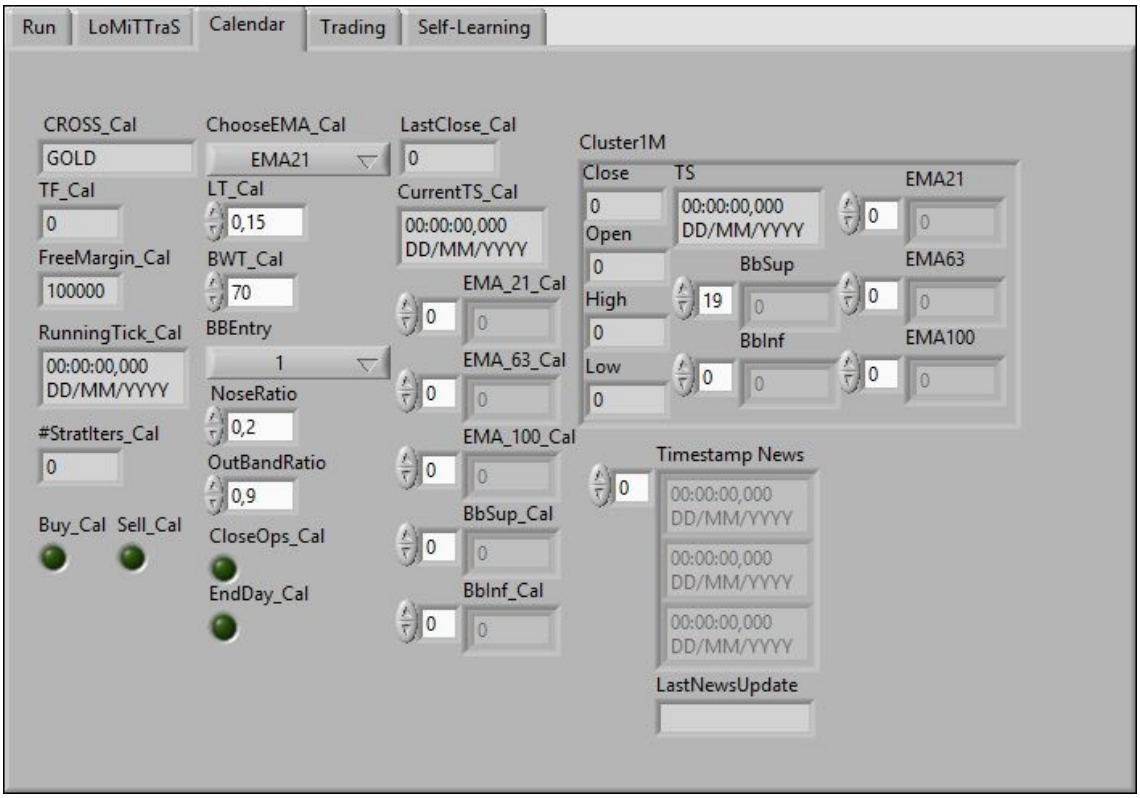


Figura 146 - Scheda dell'interfaccia grafica relativa all'esecuzione della versione Calendar della strategia.

Il vi *Calendar_LoMITTraS.vi* implementa la strategia, avendo come ingressi i seguenti:

- *LastClose_Cal*: il valore più recente relativo al prezzo (tick);
- *EMA_21_Cal, EMA_63_Cal ed EMA_100_Cal*: vettori dei valori più recenti delle medie mobili esponenziali delle ultime 20 iterazioni; nella versione corrente dell'applicativo si è scelto di usare i valori relativi alle candele 1 minuto, essendo le variazioni di quelli relativi ai tick praticamente trascurabili;
- *CurrentTS_Cal*: il Timestamp più recente;
- *BbSup_Cal* e *BbInf_Cal*: i vettori dei valori delle bande di Bollinger relativi alle ultime 20 candele 1 minuto;
- *FreeMargin_Cal*: margine libero del conto;
- *Interface Dir*: la corrente cartella di lavoro;
- *CloseOps_Cal*: una variabile booleana di controllo per indicare, se settata a *True*, che ci si approssima alla sospensione della strategia ed è quindi necessario chiudere forzatamente tutte le operazioni in essere;
- *DontOpen_Cal*: variabile booleana di controllo per indicare l'inibizione; se a *True*, infatti, la strategia è inibita nell'apertura di posizioni;
- *Cluster1M*, cluster contenente tutte le informazioni sul timeframe 1 minuto, ovvero il Timestamp e l'ultima candela (open, close, high e low), le medie mobili esponenziali (a 21, 63 e 100 periodi) e le bande di Bollinger (superiore ed inferiore) relative alle ultime 20 candele 1 minuto;
- *BWT_Cal*: variabile intera per il parametro BandWidth Threshold;

- *ChooseEMA_Cal*: variabile intera per la scelta della media mobile da utilizzare, 0 per la media a 21 periodi, 1 per quella a 63 e 2 per quella a 100 periodi;
- *LT_Cal*: variabile double per il parametro LimitThreshold;
- *BBEntry*: variabile double relativa al fattore moltiplicativo della larghezza di banda per la valutazione delle entrate;
- *NoseRatio*: variabile double per valutare la lunghezza in percentuale dell'ombra della candela che, eventualmente fuoriesce dalla banda di Bollinger, utilizzata nella strategia di uscita *Exit2*;
- *OutBandRatio*: variabile double per valutare la lunghezza in percentuale del corpo della candela che fuoriesce dalla banda di Bollinger, utilizzata nella strategia di uscita *Exit2*;
- *Exit1* ed *Exit2*, due variabili booleane, che indicano rispettivamente se attivare (*True*) o meno (*False*) le corrispondenti strategia di uscita.

Il codice della strategia è suddiviso in sei differenti aree funzionali.

Nell'area 1 viene anzitutto aperto in scrittura il file di log giornaliero della strategia, al cui interno saranno memorizzate le informazioni di log di apertura a chiusura posizioni (v. Figura 147). Quindi, a partire dai valori delle medie mobili e delle bande di Bollinger, vengono calcolate le seguenti quantità sia in relazione ai tick che alle candele 1 minuto:

- Valore del parametro %b;
- FullBW;
- BollBW;
- PIPBW.

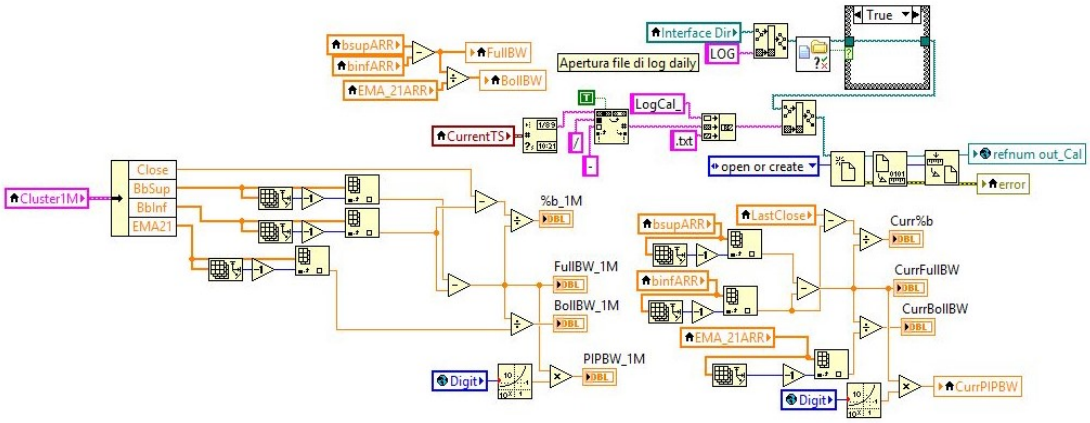


Figura 147 - Area 1 per l'apertura del daily log file della strategia Calendar LoMiTTraS.

Nelle aree funzionali 2 e 3 vengono valutate le condizioni di apertura posizione in corrispondenza rispettivamente della banda superiore ed inferiore.

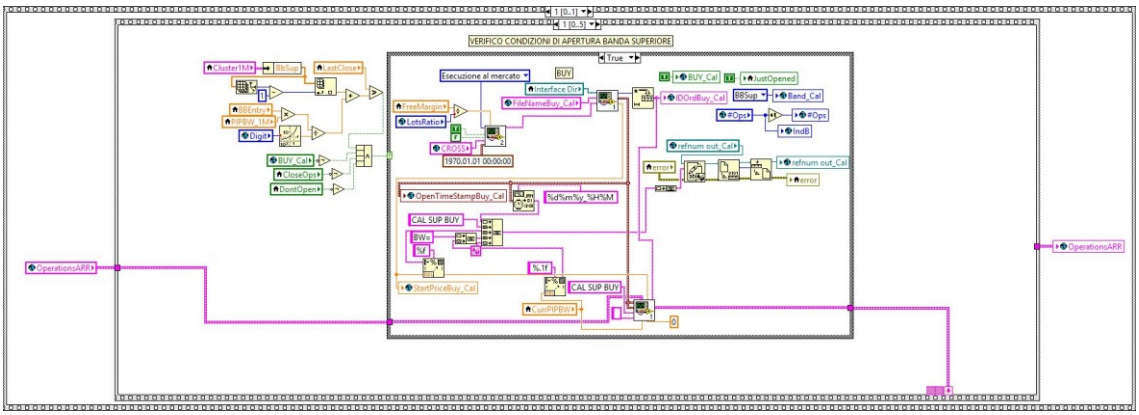


Figura 148 - Verifica delle condizioni di apertura posizioni sulla banda superiore di Bollinger.

Nello specifico, se il valore di prezzo attuale *LastClose* (tick) è superiore al valore della banda di Bollinger superiore dell'ultima candela 1 minuto incrementato di *BBEntry* volte la larghezza di banda in PIPs sempre relativa all'ultima candela 1 minuto, se esistono posizioni Buy aperte (variabile booleana globale *BUY_Cal* impostata a false), se *CloseOps* è false e se *DontOpen* è false, allora viene ordinata l'apertura di una posizione di tipo Buy attraverso la scrittura su un file di configurazione, in maniera del tutto analoga a quanto avveniva per la strategia LoMiTTraS, così come vengono memorizzate le relative informazioni nel file di log giornaliero. In maniera duale, per la banda inferiore verrà ordinata l'apertura di una posizione di tipo sell se sussistono tutte le seguenti condizioni (v. Figura 148 e Figura 149):

- Il valore di prezzo attuale *LastClose* (tick) è inferiore al valore della banda di Bollinger inferiore dell'ultima candela 1 minuto decrementato di *BBEntry* volte la larghezza di banda in PIPs, sempre relativa all'ultima candela 1 minuto;
- Non sussistono posizioni Sell aperte (variabile booleana globale *SELL_Cal* settata a false);
- *CloseOps* è false;
- *DontOpen* è false.

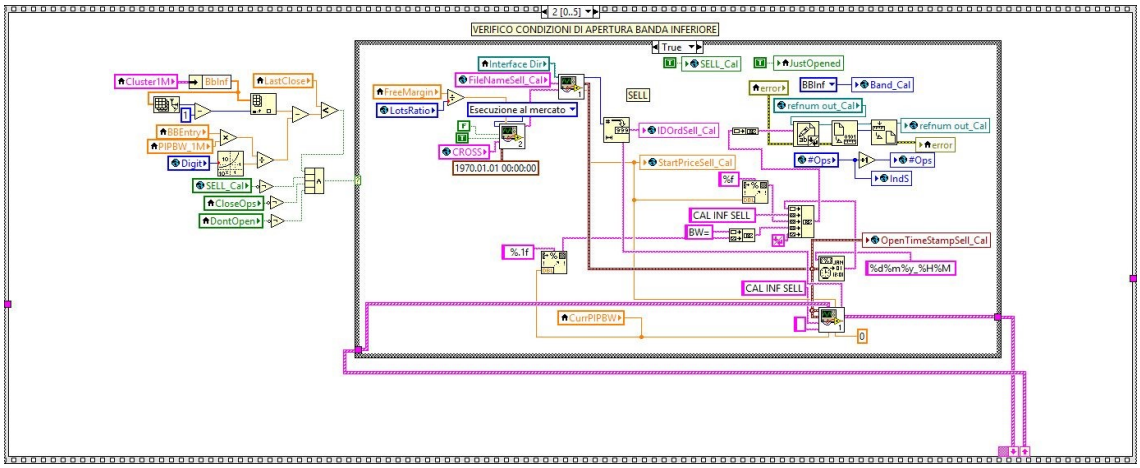


Figura 149 - Verifica delle condizioni di apertura posizioni sulla banda inferiore di Bollinger.

Nell'area funzionale 4 sono presenti le condizioni di uscita. Così come accadeva per la strategia LoMiTTraS, sono presenti innanzitutto i controlli di uscita per stop loss, implementati in maniera esattamente analoga a quelli utilizzati per la strategia ordinaria (v. Figura 150).

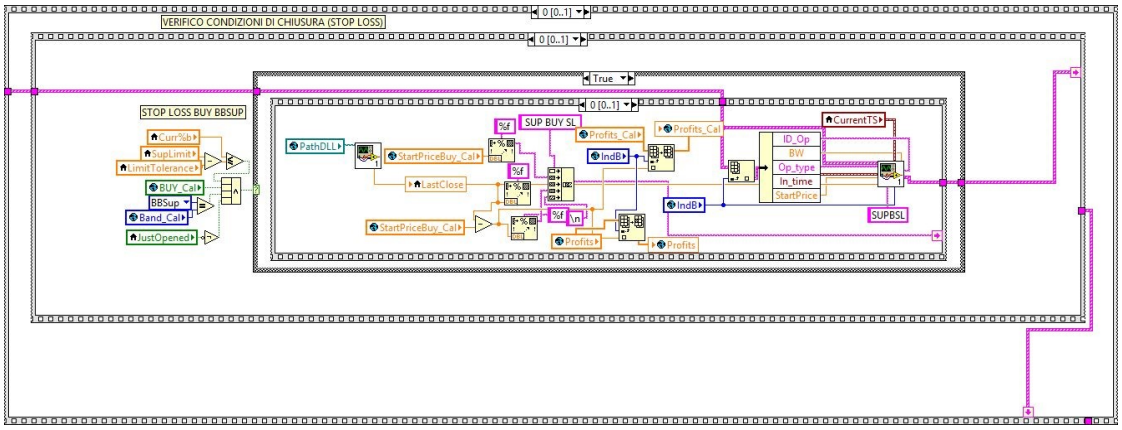


Figura 150 - Condizioni di chiusura posizioni long per stop loss (perdita massima impostata) sulla banda superiore di Bollinger.

In aggiunta a tali controlli, la strategia *Calendar_LoMiTTraS* implementa due ulteriori strategia di uscita, *Exit1* ed *Exit2*, che saranno descritte separatamente nel seguito (v. Figura 151 e Figura 152).

Nell'area funzionale 5 vi è il codice relativo alla chiusura forzata delle operazioni. Infine, nell'area funzionale 6 viene calcolato il profitto complessivo di giornata, scritto su file di log e chiuso il file stesso.

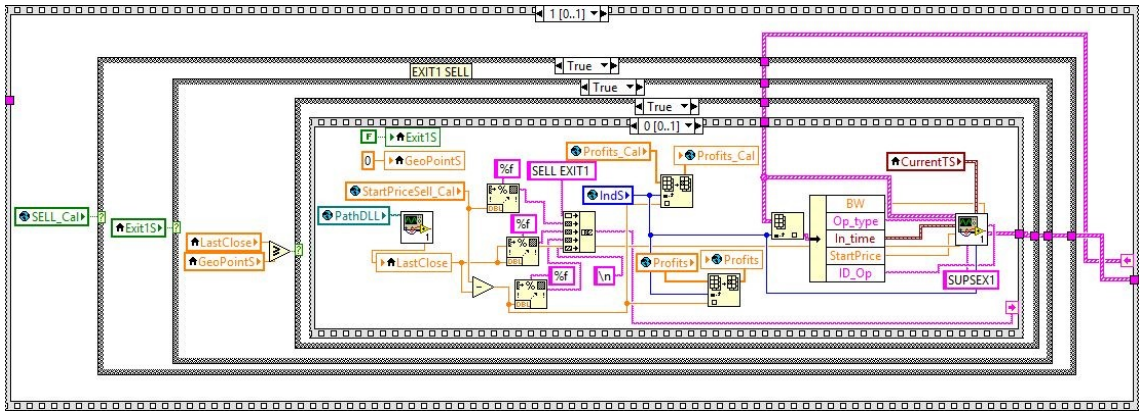


Figura 151 - Condizioni di chiusura posizioni short per euristica di uscita Exit 1.

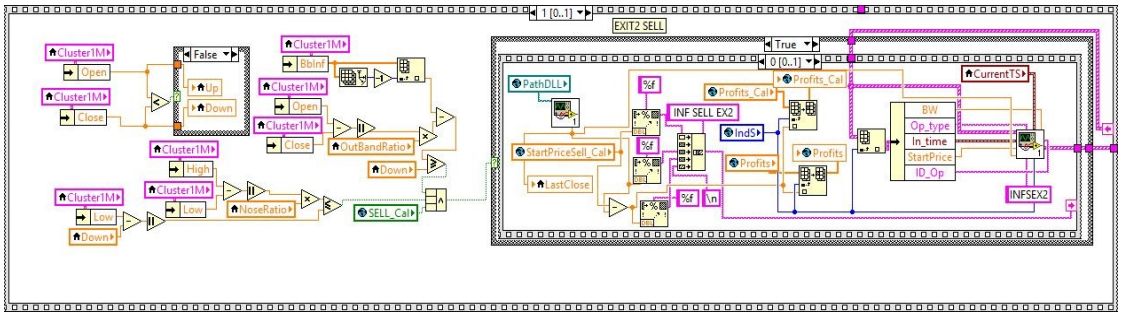


Figura 152 - Condizioni di chiusura posizioni long per euristica di uscita Exit 2.

7.6 Strategia di uscita Exit1

La strategia di uscita *Exit1* implementa il concetto di inseguimento del punto geometrico: in relazione all'ultima candela chiusa (1 minuto), si calcola il centro geometrico corrente e si attende il verificarsi del raggiungimento di tale valore nella successiva candela 1 minuto. Se il raggiungimento non avviene,

si reitera il procedimento, ovvero si calcola il nuovo valore del centro geometrico e si verifica il raggiungimento di tale livello nella successiva candela chiusa.

Nello specifico, la verifica delle condizioni di uscita della strategia *Exit1* per le operazioni Sell avviene soltanto al verificarsi delle seguenti condizioni (v. Figura 153):

- Il valore della banda di Bollinger inferiore nelle ultime tre iterazioni è decrescente;
- Vi è a mercato un'operazione Sell di strategia;
- L'operazione non è stata aperta nella corrente iterazione della strategia.

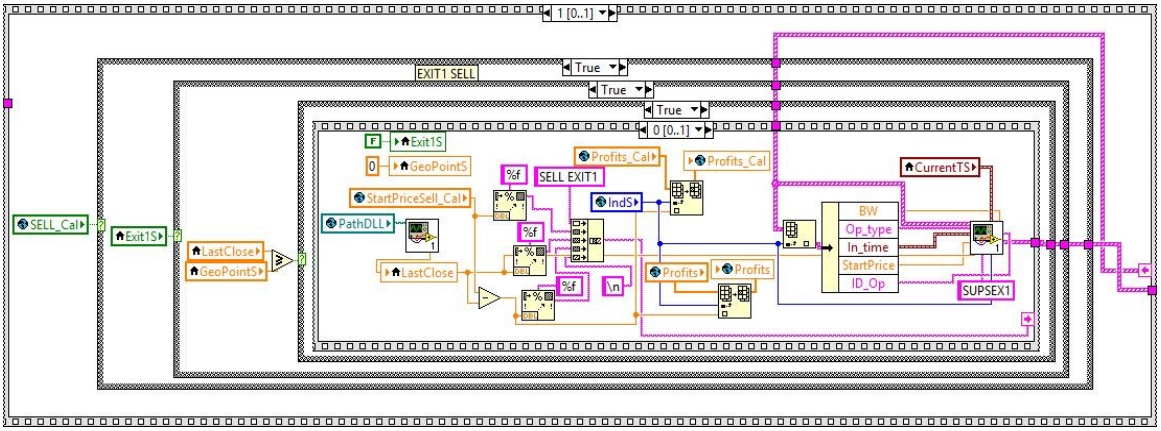


Figura 153 - Codice per la verifica delle condizioni di chiusura secondo l'euristica Exit1.

Nel caso sussistano le suddette condizioni viene attivata una variabile booleana di controllo *Exit1S* e viene calcolato il punto geometrico corrente come di seguito descritto (v. Figura 154):

$$GeoPointB = GeoPointS = \frac{|High - Low|}{2}$$

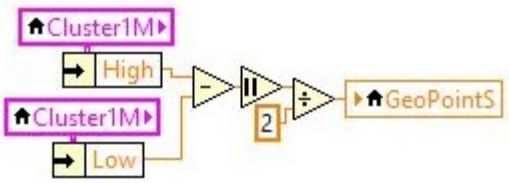


Figura 154 - Codice per il ricalcolo del centro geometrico.

A partire dall'iterazione successiva, quindi, finché *Exit1S* è *True, si* verifica il raggiungimento del centro geometrico, ovvero se il valore di prezzo corrente *LastClose* è minore o uguale al valore del centro geometrico *GeoPointS*: in tal caso, viene ordinata la chiusura dell'operazione Sell; in caso contrario, viene ricalcolato il valore di *GeoPointS* che sarà utilizzato per il controllo nell'iterazione successiva.

La porzione di codice per il controllo della strategia di uscita *Exit1* per l'operazione Buy è del tutto duale, differendo unicamente nella verifica del raggiungimento del centro geometrico dal basso verso l'alto (valore di *LastClose* maggiore o uguale al valore di *GeoPointB*). Va infine notato che per tali controlli vengono usate diverse variabili, quali *GeoPointB* ed *Exit1B*.

7.7 Strategia di uscita Exit2

La strategia di uscita *Exit2* implementa l'occorrenza di uno specifico pattern di candela; in presenza di tale pattern, infatti, viene ordinata la chiusura della relativa operazione.

Nello specifico, il pattern di candela di cui verificare l'occorrenza per la chiusura delle operazioni di tipo Buy (oppure Sell) possiede le seguenti proprietà:

- Il naso verso l'alto (o verso il basso) è esteso almeno un quinto (o una frazione diversa indicata dalla variabile double *NoseRatio*) dell'estensione dell'intera candela (ovvero minimo - massimo);
- Il corpo della candela è fuori dalla banda di Bollinger superiore (o inferiore) per almeno nove decimi (o una frazione diversa indicata dalla variabile double *OutBandRatio*) del corpo della candela (ovvero open - close).

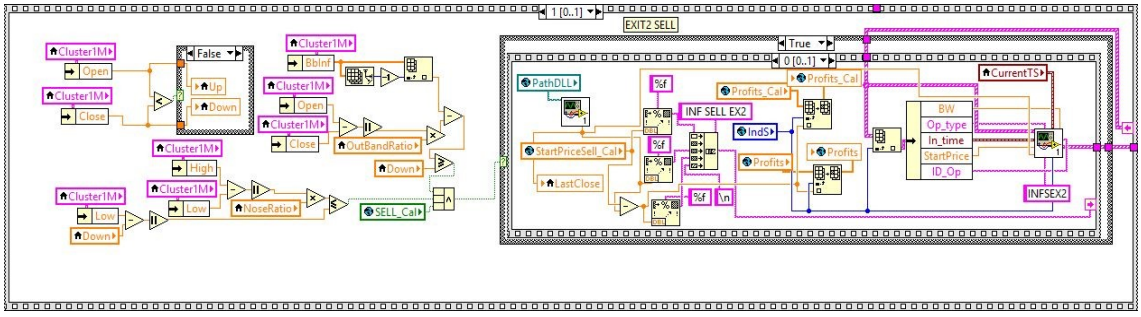


Figura 155 - Codice per la verifica delle condizioni di chiusura delle posizioni short secondo l'euristica Exit2.

In presenza delle suddette condizioni, viene ordinata la chiusura della operazione di tipo Buy (o Sell) (v. Figura 155).

7.8 Aspetti implementativi - Variabili Globali e Strutture dati principali

Per facilitare la comprensione della descrizione del codice che seguirà nel capitolo, in questo paragrafo viene riportato un dettaglio sulle variabili globali e le principali strutture dati impiegate dalla strategia.

7.8.1 Global_LoMiTTraS

Nel vi *Global_LoMiTTraS* sono presenti i controlli relativi a variabili globali del progetto, utilizzate quindi dai vari vi e che permettono, grazie appunto al loro scope globale, a ciascun vi di modificare il loro valore e, al contempo, che tale valore aggiornato possa essere letto da altri vi di progetto (v. Figura 156).

Nello specifico, è stato previsto l'utilizzo delle seguenti variabili globali:

- *Band e Band_Cal*: controllo che indica se la corrente operazione OCO sia stata ordinata a seguito dello sfondamento della banda di Bollinger superiore o inferiore;
- *BUY e SELL, BUY_Cal e SELL_Cal*: variabili globali booleane per indicare che allo stato attuale una corrispondente posizione (Buy o Sell) sia aperta per le due differenti modalità operative, *LoMiTTraS* e *Calendar*;
- *FileNameSell e FileNameBuy, FileNameSell_Cal e FileNameBuy_Cal*: variabili globali di tipo stringa relative al nome del file di configurazione per la comunicazione con la piattaforma per

l'apertura di posizioni, rispettivamente di tipo Buy e Sell per la strategia *LoMiTTraS* e quella *Calendar*;

- *CloseFileName e CloseFileNameBuy, CloseFileName_Cal e CloseFileNameBuy_Cal*: variabili globali di tipo stringa per indicare il nome del file di configurazione per la comunicazione con la piattaforma per la chiusura di posizioni, rispettivamente di tipo Buy e Sell;
- *IndS ed IndB, IndS_Cal ed IndB_Cal*: variabili globali di tipo intero per identificare in maniera univoca l'indice di operazione della corrente iterazione della strategia;
- *IDOrdBuy ed IDOrdSell, IDOrdBuy_Cal ed IDOrdSell_Cal*: ID univoci (ticket) restituiti dalla piattaforma MetaTrader4 ed associati alle rispettive operazioni, Buy e Sell;
- StartPriceBuy e StartPriceSell, StartPriceBuy_Cal e StartPriceSell_Cal: prezzo di apertura della corrispondente operazione;
- OpenTimeStampBuy e OpenTimeStampSell, OpenTimeStampBuy_Cal e OpenTime-StampSell_Cal: Timestamp di apertura della corrispondente operazione;
- *Refnum out, refnum out_Cal:* variabile globale di tipo reference associato al file di report della strategia;
- *PathDLL:* percorso relativo alla libreria DLL;
- *CROSS:* attuale strumento finanziario;
- *Digit:* variabile globale di tipo intero per indicare il numero di cifre decimali significative in relazione allo strumento finanziario attuale;
- *Timeframe:* variabile globale di tipo intero ad indicare il corrente timeframe espresso in secondi;
- *LotsRatio:* variabile globale di tipo intero relativa al rapporto di dimensionamento dei lotti, da usare come denominatore nel rapporto con il *FreeMargin* per determinare il numero di lotti delle posizioni da aprire;
- *OperationsARR:* vettore globale di cluster relativi alle operazioni, aggiornato costantemente dalla strategia durante la sua esecuzione;
- *Profits e Profits_Cal:* vettore globale di interi relativo ai profitti corrispondenti a ciascuna operazione aperta e chiusa a mercato;
- *#Ops e #Ops_Cal:* numero di operazioni correntemente effettuate dalla strategia; si noti che tale numero riflette il numero complessivo di operazioni Buy e Sell e coincide con il doppio del numero di operazioni OCO effettuate;
- *FinalProfit e FinalProfit_Cal:* variabile globale di tipo double per indicare il profitto complessivo della corrente iterazione della strategia.

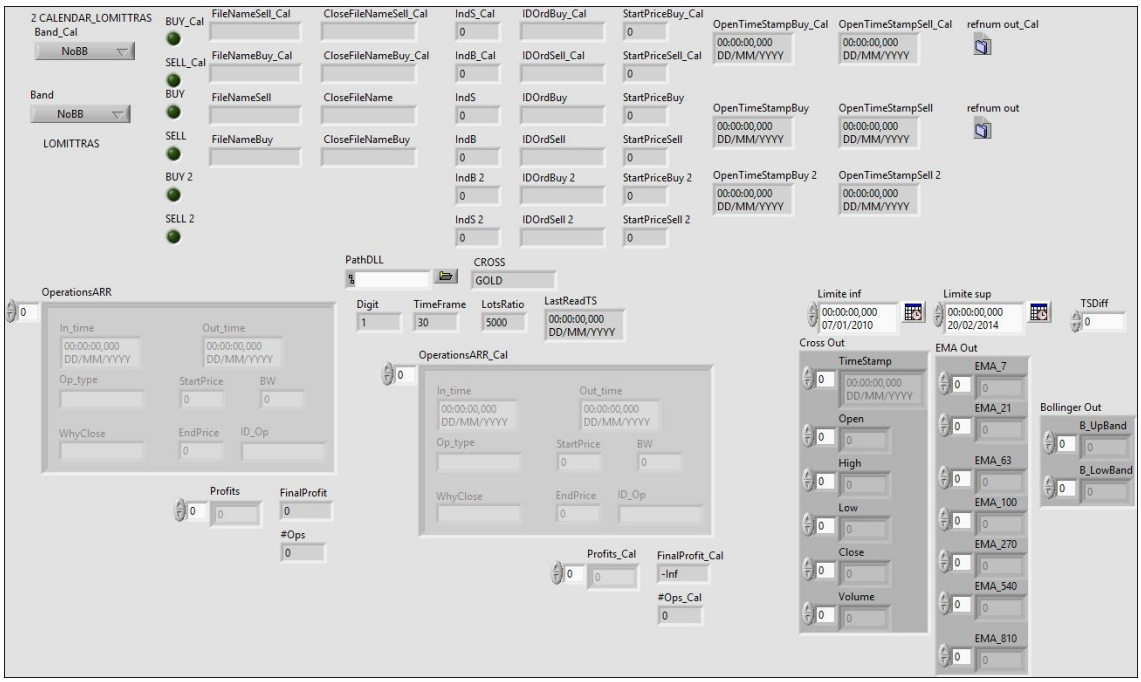


Figura 156 - Front Panel del vi GlobalLoMiTTraS, all'interno del quale sono presenti tutti i controlli globali dell'applicazione.

7.8.2 Cluster OperationsARR

L'array di cluster *OperationsARR* è utilizzato all'interno dell'applicativo per memorizzare tutte le informazioni relative agli ordini, indipendentemente dal fatto che essi siano già stati chiusi in piattaforma o meno; questo array viene azzerato soltanto alla fine della corrente iterazione della strategia.

Alla modifica di questo vettore di cluster possono contribuire diverse aree del codice: vi accede, ad esempio, in scrittura la porzione di codice relativa all'apertura delle posizioni per l'inserimento delle informazioni relative al prezzo, al Timestamp, al tipo di operazione, il ticket dell'ordine e la larghezza delle bande di Bollinger espressa in termini di PIPs; vi accedono anche, sempre in scrittura, tutte quelle aree di codice relative alla chiusura di un ordine per le differenti motivazioni possibili al fine di memorizzare nel cluster di operazione il prezzo, il Timestamp e la motivazione della chiusura del relativo ordine.

La struttura del singolo cluster di operazione è la seguente (v. Figura 157):

1. *In_time*: valore di tipo Timestamp che indica l'istante temporale relativo all'apertura dell'ordine;
2. *Out_time*: valore di tipo Timestamp che indica l'istante temporale relativo alla eventuale chiusura dell'ordine; nel caso di ordini ancora aperti, esso coincide con *In_time*.
3. *Op_type*: campo testuale ad indicare lo sfondamento di una delle due bande di Bollinger (*SUP* per superiore, *INF* per inferiore) e la tipologia di operazione (BUY o SELL);
4. *StartPrice*: campo numerico relativo al prezzo di apertura della posizione, ottenuto dalla piattaforma dopo che l'ordine è stato effettivamente eseguito ed il relativo ticket generato;
5. *BW*: valore intero relativo alla larghezza di banda di Bollinger, espresso in PIPs, relativo all'ultima candela chiusa;
6. *WhyClose*: valore di tipo stringa ad indicare la motivazione della chiusura dell'operazione (SL per Stop Loss, TP per Take Profit, F per Chiusura Forzata);

7. *EndPrice*: campo numerico relativo al prezzo di chiusura dell'ordine, ottenuto dalla piattaforma dopo che l'ordine è stato effettivamente chiuso;
8. *ID_Op*: valore di tipo intero indicante in ticket univoco associato dalla piattaforma MetaTrader4 al corrente ordine.

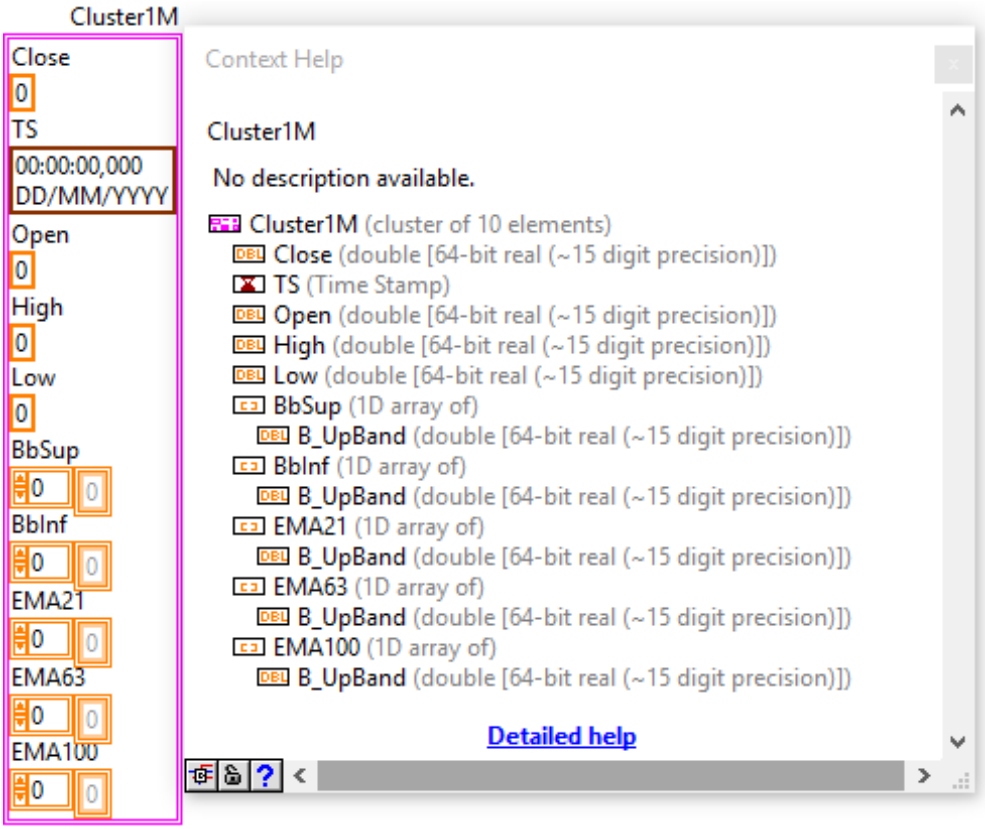


Figura 157 - Struttura del Cluster delle operazioni OperationsARR.

7.9 Aspetti implementativi - VI di uso comune o di supporto

Alcuni vi svolgono funzionalità generiche, non specifiche per un singolo obiettivo. Al fine di evitare ripetizioni, si descrivono in questa sezione tali vi, così da poterli menzionare senza descrizioni ripetute nelle pagine successive.

- *ReadCalLoMiTTraS*: Tale VI si interfaccia alla DLL realizzata per la comunicazione tra applicativo e Piattaforma. Riceve in input il percorso assoluto alla DLL, ed il suo scopo è quello di leggere e restituire i valori correnti di prezzo e relativo istante temporale, bande di Bollinger, EMA, margine libero, timeframe e cross, che vengono scritti nelle variabili contenute nella DLL dall'Expert Advisor in piattaforma (v. Figura 158).

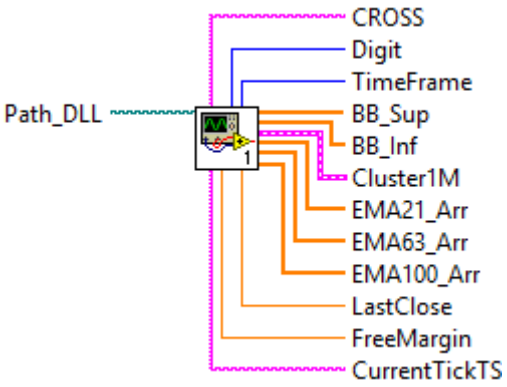


Figura 158 - Input/Output del vi di lettura dalla libreria dei dati utili per la versione Calendar.

- *ReadFromDLL*: VI per l'interfacciamento alla DLL realizzata per la comunicazione tra applicativo e Piattaforma MT4. Riceve in input il percorso assoluto alla DLL e restituisce i valori correnti di prezzo (riferito al tick) e relativo istante temporale, che vengono scritti nelle variabili contenute nella DLL dall'Expert Advisor in piattaforma (v. Figura 159).

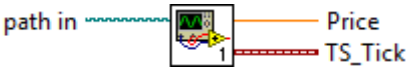


Figura 159 - Input/Output del vi di lettura dalla libreria di interfacciamento.

- *ConvertStringToTimeStamp*: questo VI riceve in input un Timestamp sotto forma di stringa di testo nel formato YYYY.MM.DD HH:MM:SS e lo converte in un oggetto Timestamp di LabView (v. Figura 160).

Figura 160 - Input/Output del vi di conversione di una stringa di controllo in Timestamp con precisione ai minuti.

- *ConvertStringToTimeStamp_msec*: questo VI riceve in input un Timestamp sotto forma di stringa di testo nel formato YYYY.MM.DD HH:MM:SS:UUU e lo converte in un oggetto Timestamp di LabView, riportando quindi in output anche i decimi di secondo (v. Figura 161).

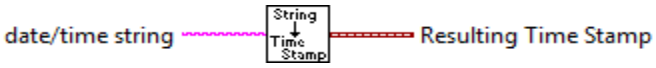


Figura 161 - Input/Output del vi di conversione di una stringa di controllo in Timestamp con precisione ai secondi.

- *InsertIntoOpCluster*: sub-vi utile ad inserire un nuovo elemento (un cluster) all'interno del vettore di cluster delle operazioni e viene utilizzato all'apertura di una posizione. Esso prende in input il cluster sorgente, due valori di tipo Timestamp (apertura e chiusura), tre valori di tipo double (prezzi di apertura e chiusura e larghezza di banda in PIP all'apertura), tre stringhe di controllo (tipo e ticket operazione e motivazione della chiusura) e restituisce il vettore di cluster con il nuovo elemento aggiunto (v. Figura 162).

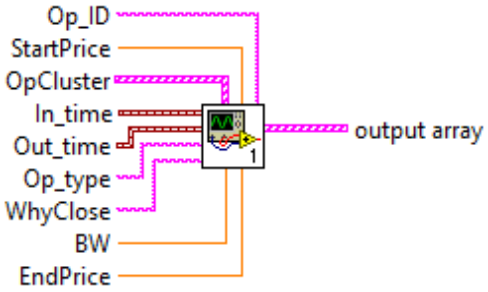


Figura 162 - Input/Output del vi di inserimento di un nuovo elemento nel cluster delle operazioni.

- *ReplaceIntoOpCluster*: Questo sub-vi si occupa ad aggiornare i valori di un cluster nel vettore di cluster delle operazioni e viene invocato alla chiusura di una posizione. In maniera analoga alla *InsertIntoOpCluster*, prende in input il vettore di cluster delle operazioni e ciascuno degli elementi che formano il cluster e restituisce il vettore di cluster con l'elemento aggiornato (v. Figura 163).

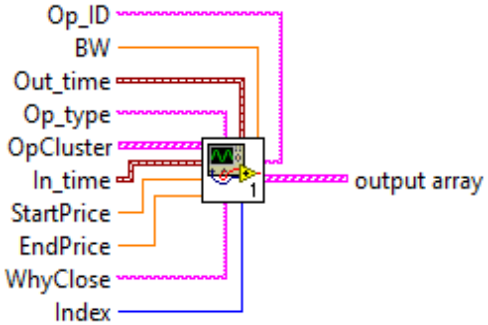


Figura 163 - Input/Output del vi di rimpiazzamento di un elemento nel cluster delle operazioni.

- *CreateNewOrder*: questo vi si occupa unicamente di costruire la stringa di richiesta di apertura dell'ordine. Esso riceve in ingresso le informazioni relative all'ordine da aprire, quali tipo di operazione (al mercato o pendente), tipo (Buy o Sell), numero di lotti, cross, nome del file su cui andrà scritta la stringa di richiesta, Timestamp di scadenza della richiesta, strumento finanziario, prezzo di richiesta stop loss e take profit (se presenti) e restituisce appunto la stringa di controllo da scrivere su file. Questo vi è comune alle operazioni di Buy e Sell, a differenza di quelli che verranno descritti nel seguito, di cui esistono due versioni

esattamente identiche ma deputate l'una alle operazioni di Buy e l'altra a quelle di Sell (v. Figura 164).

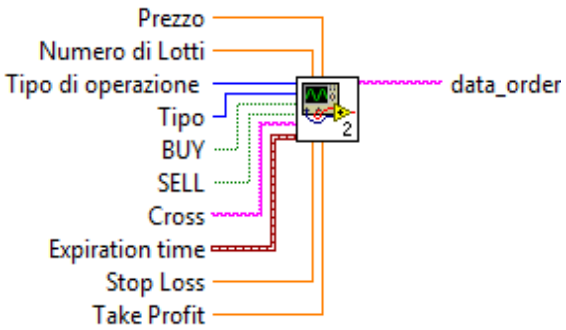


Figura 164 - Input/Output del vi per la creazione della stringa per l'apertura di un nuovo ordine.

- *OpStringWriterBuy* e *OpStringWriterSell*: questi vi sono deputati a garantire l'avvenuta apertura della posizione dalla piattaforma, attraverso l'effettiva scrittura della stringa di richiesta di apertura della posizione e l'attesa di avvenuta operazione; come anticipato in precedenza. Gli ingressi sono il percorso della libreria DLL e l'intero Digit (entrambe variabili globali di progetto), la cartella di lavoro ed il nome del file su cui comunicare alla piattaforma la richiesta, la stringa di richiesta da scrivere su tale file (la stringa è output del vi *CreateNewOrder*). Il vi entra in un while che invoca ripetutamente il vi *GetOpTicketFromPlatformBuy* (o *GetOpTicketFromPlatformSell*) finché non ne ottiene un ticket valido (valore diverso da zero) ed il Timestamp ed il prezzo di apertura. Nel caso in cui vi siano errori nella gestione del file di configurazione, al ticket viene associato un valore pari a -1 per segnalare tale errore (v. Figura 165).

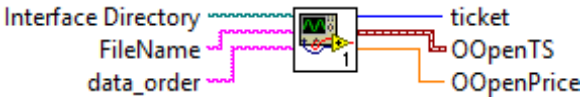


Figura 165 - Input/Output del vi per la scrittura della stringa per la richiesta di apertura di una posizione long.

- *GetOpTicketFromPlatformBuy* e *GetOpTicketFromPlatformSell*: questi vi vengono richiamati ripetutamente all'interno del vi *OpStringWriterX* ed implementano la comunicazione con la piattaforma attraverso la funzione di libreria *GetOrderInfo* per ricevere un numero di ticket ordine valido con relativi Timestamp e prezzo di apertura. Si noti che, ai fini della memorizzazione del corretto Timestamp, vista la gestione da parte di LabView del parametro DST (ora legale/solare) dei Timestamp, si è reso necessario l'utilizzo del vi ausiliario *TimeStampToGMT* per assicurare la corrispondenza temporale (v. Figura 166).

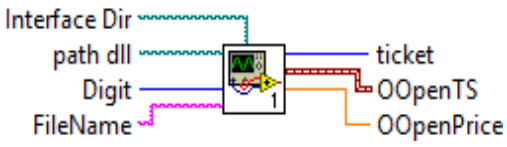


Figura 166 - Input/Output del vi per la ricezione dalla piattaforma di un numero di ticket valido.

- *OrderToClose*: questa funzione è incaricata di comunicare alla piattaforma la richiesta di chiusura ordine attraverso la scrittura sul file di configurazione della relativa stringa di controllo. Essa prende in input la cartella di lavoro, il nome del file di configurazione su cui scrivere ed il ticket dell'ordine di cui richiedere la chiusura e, molto semplicemente, scrive sul file su una riga tale ID. Sarà poi la piattaforma, attraverso l'Expert Advisor a leggere da tale file la richiesta ed assolverla (v. Figura 167).

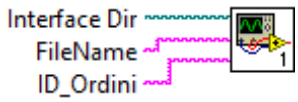


Figura 167 - Input/Output del vi per la scrittura della stringa per la richiesta di chiusura di una posizione.

- *GetValueFromFile*: questo vi riceve in input il percorso ad un file contenenti dati su uno specifico strumento finanziario e da esso estrae i dati presenti e li restituisce sotto forma di differenti cluster. Nello specifico, esso restituisce il vettore dei Timestamp, un cluster contenente, oltre al Timestamp, anche i vettori dei dati costituenti una Candela, ovvero open, close, high, low e volume, l'intero Digit, un cluster contenente i vettori delle bande di Bollinger superiore ed inferiore, un cluster dei vettori degli EMA (a 7, 21, 63, 100, 270, 540 e 810 periodi), un cluster per i valori dell'indicatore Stocastico, un cluster per i valori dell'indicatore ADX, un cluster per i valori Accumulation Distribution, un luster per i valori dell'Awesome Oscillator, un cluster per i valori del Parabolic SAR (v. Figura 168).

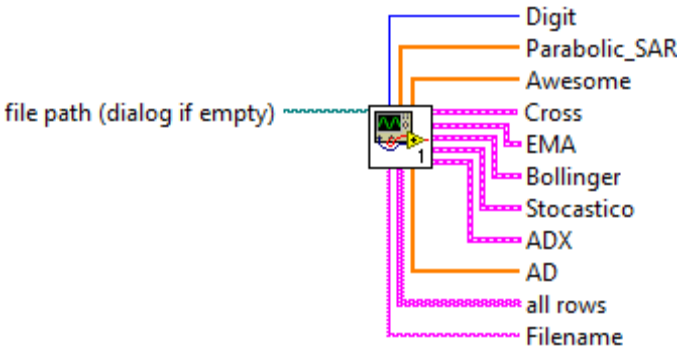


Figura 168 - Input/Output del vi per l'estrazione dei dati di interesse e la loro memorizzazione in opportune strutture dati.

7.10 Aspetti implementativi - Aree funzionali

In questo paragrafo saranno descritte in maniera dettagliata le aree funzionali individuate precedentemente in questo documento, ponendo maggiormente in luce gli aspetti connessi alla logica ed alle scelte implementative e, ove necessario, ribadendo ed integrando concetti già espressi nei paragrafi precedenti.

7.10.1 Area 1

L'area 1, come detto, si occupa di gestire l'esecuzione dell'applicativo in maniera automatizzata: nel caso esso venga eseguito da un altro programma, infatti, il numero di argomenti passati da riga di comando è strettamente maggiore di uno (quello di default nel caso opposto, ovvero "LabView"); per questo motivo gli argomenti passati da riga di comando, a cui si accede attraverso il Property Node di LabView *App.Args*, vengono utilizzati per inizializzare tre parametri di tipo path: l'interface directory che indica la cartella di lavoro del progetto, il percorso alla libreria DLL ed il percorso al file di dump per il Self Learning (v. Figura 169).

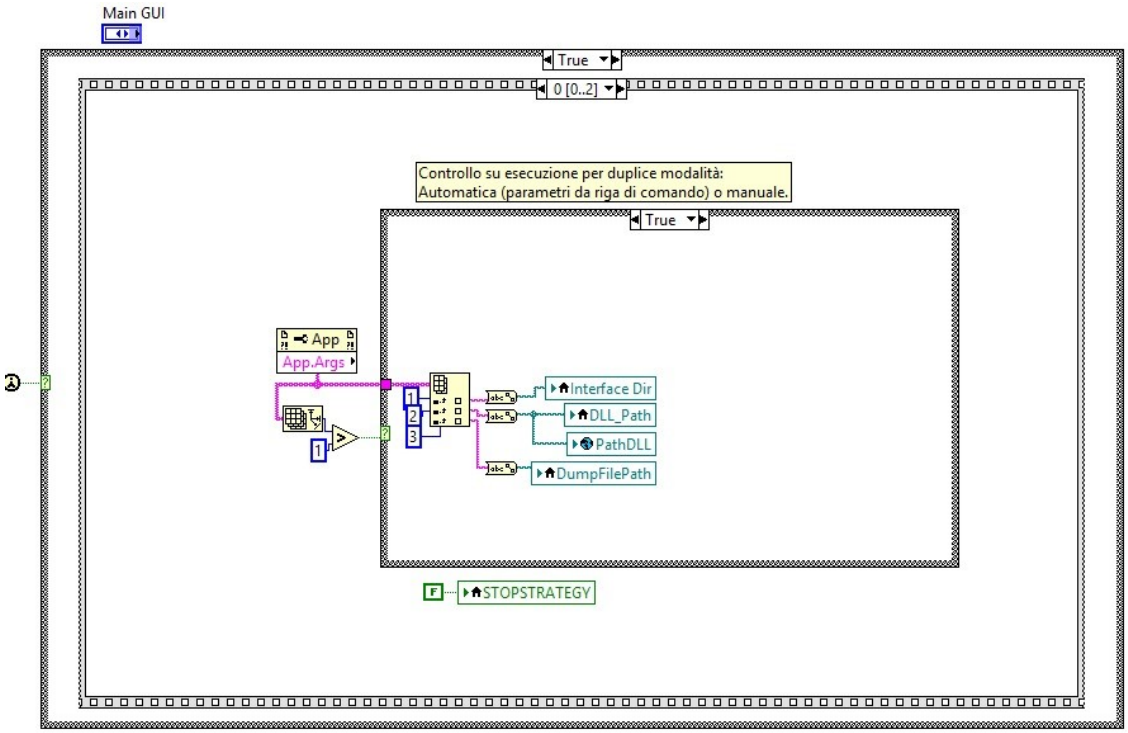


Figura 169 - Scelta della modalità di esecuzione (manuale o automatico con passaggio di argomenti da riga di comando).

Si tenga presente che quest'area viene eseguita unicamente al primo lancio dell'esecuzione, grazie all'impiego della funzione *First Run*.

7.10.2 Area 2

L'area 2, si occupa di gestire l'inizializzazione del software, di eseguire ciclicamente la strategia ed infine effettuare il reset delle varabili. Essa può essere vista come funzionalmente suddivisa in ulteriori tre sotto aree.

7.10.2.1 *Area 2.1*

L'area 2.1 è deputata all'inizializzazione delle variabili di utilizzo del main; una delle funzionali principali, ad esempio riportare allo stato iniziale le strutture dati, le variabili globali della main GUI e degli altri vi, i vari controlli numerici, i percorsi a file e cartelle, le strutture dati utilizzate nella esecuzione del Self Learning, gli indicatori utilizzati per visualizzare il valore di alcune variabili globali, il vettore di cluster delle operazioni della corrente esecuzione della strategia, e così via (v. Figura 170).

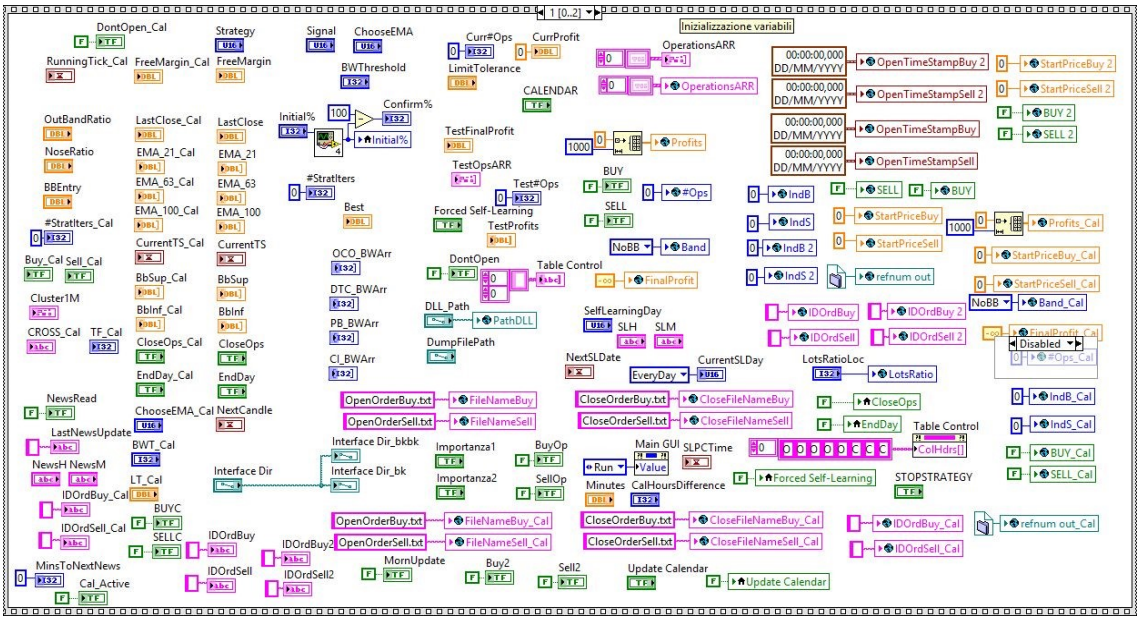


Figura 170 - Area deputata al posizionamento dei controlli relativi alle variabili in uso all'intera applicazione.

Una seconda porzione di codice dell'area 2.1 si occupa della comunicazione con la piattaforma attraverso l'Expert Advisor e la libreria condivisa per una lettura dei valori precedente all'esecuzione della strategia: questa operazione è necessaria affinché la strategia, all'atto della sua esecuzione, possa già disporre di dati aggiornati dalla piattaforma (v. Figura 171).

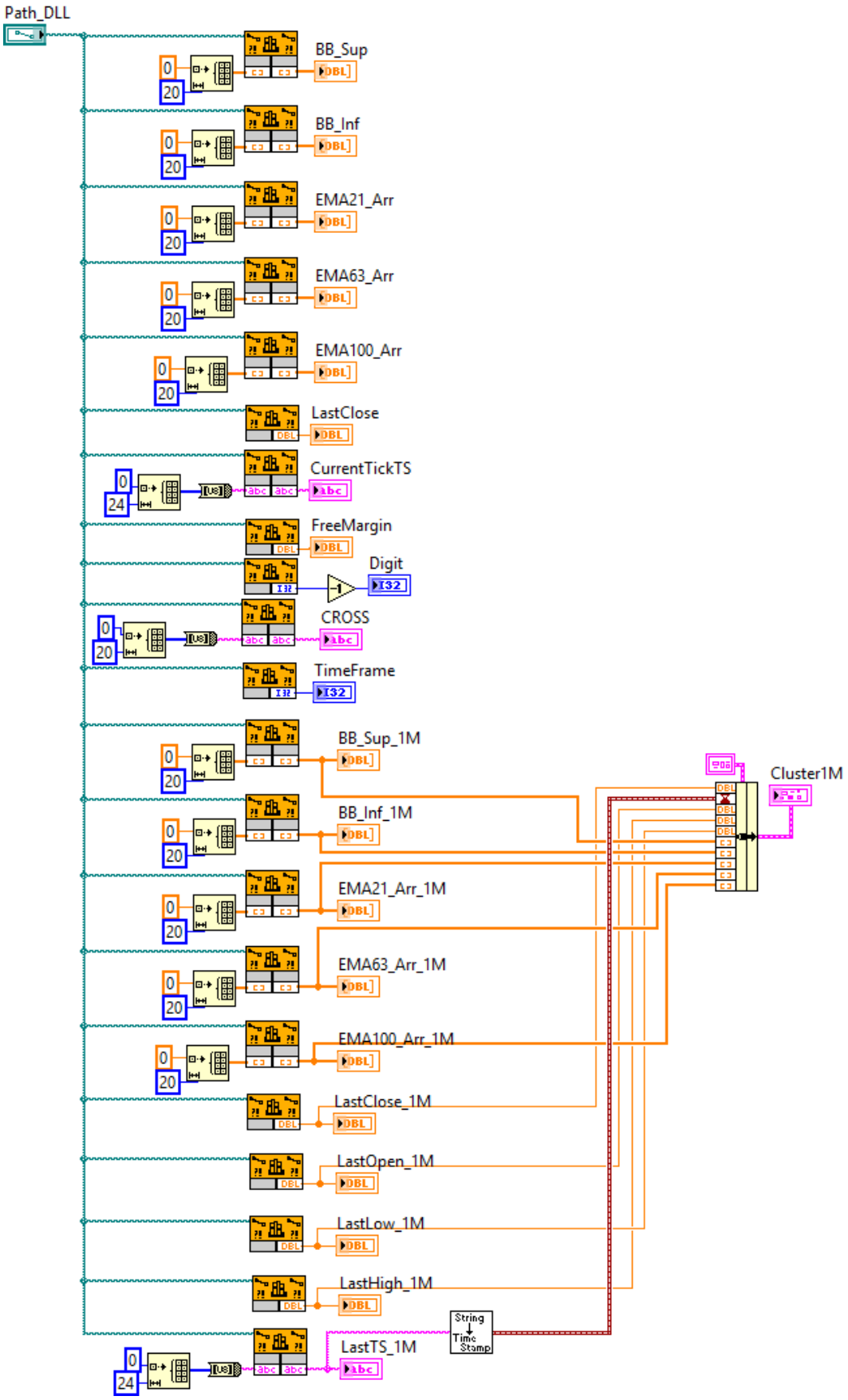


Figura 171 - Codice relativo alla lettura dei dati dalla libreria di interfacciamento con la piattaforma.

Nello specifico, attraverso la chiamata al sub-vi *ReadCalLoMiTTraS* con in input il percorso alla DLL, vengono letti e memorizzati i seguenti valori:

- *CROSS*: una stringa indicante lo strumento finanziario a cui è stata associata la corrente esecuzione dell'applicativo, essa viene memorizzata nella variabile globale CROSS;
- *Digit*: un intero che indica il numero di cifre significative dopo la virgola, fino all'unità di PIP; questo valore è strettamente dipendente dallo strumento finanziario associato e viene memorizzato nella variabile globale *Digit*;
- *Timeframe*: dopo aver letto dalla DLL un intero indicante il timeframe correntemente associato all'esecuzione ed espresso in minuti, tale valore viene moltiplicato per 60 e memorizzato nella variabile globale *Timeframe* che sarà pertanto espressa in secondi;
- *FreeMargin*: il margine libero relativo al conto attualmente associato all'esecuzione dell'applicativo;
- *BbSup* e *BbInf*: due array di double nei quali vengono memorizzati i valori delle bande di Bollinger relativi alle ultime venti candele;
- *EMA21*, *EMA63* ed *EMA100*: tre vettori di double nei quali vengono memorizzati i valori delle medie mobile a 21, 63 e 100 periodi relativi alle ultime venti candele;
- *LastClose*: il prezzo relativo all'ultimo aggiornamento disponibile in piattaforma;
- *CurrentTS*: una stringa che indica il Timestamp relativo all'ultimo aggiornamento disponibile in piattaforma. Essendo sotto forma di stinga, quest'output del vi *ReadCalLoMiTTraS* viene passato in ingresso al vi di uso comune *Convert_String_to_TimeStamp_msec* per la sua conversione in variabile di tipo Timestamp.

Il Timestamp, inoltre, viene utilizzato per inizializzare la variabile *Timestamp Start*, che indicherà l'istante temporale di inizio esecuzione della strategia, e la variabile Timestamp *NextCandle*, che indicherà il Timestamp relativo alla prossima candela di cui leggere i valori dalla DLL e sarà aggiornato in una porzione successiva del codice. Per inizializzare queste due variabili il Timestamp letto dalla libreria viene dapprima trasformato in double, poi viene calcolato il resto della sua divisione per il timeframe associato alla corrente esecuzione: in tal modo sarà possibile conoscere esattamente quanti secondi mancano alla chiusura della prossima candela; il numero di secondi così ottenuto viene infine sommato alla conversione in double del Timestamp letto dalla DLL, convertito in Timestamp e scritto nelle variabili *Start* e *NextCandle*.

Nella terza sezione del codice dell'Area 2.1 vi sono le operazioni da eseguire alla prima esecuzione dell'applicativo; nel caso in cui la variabile booleana *FirstRun*, infatti, abbia valore *True*, vengono eseguite le seguenti operazioni:

- Lettura da file di configurazione *.lit* (*Learning Initialization Values*) dell'insieme di definizione e del valore iniziale da utilizzare per il parametro BandWidth Threshold attraverso la chiamata al vi *BWTValuesReader* (v. Figura 172). Tale vi prende in ingresso la directory di lavoro dell'applicativo tramite la variabile locale *InterfaceDir*, lo strumento finanziario attuale tramite la variabile globale CROSS ed il timeframe tramite la variabile globale Timeframe e apre in lettura il file *BWTLearningInitValues.lit*; dopo aver aperto il file cerca tra le entry quella corrispondente al cross ed al timeframe attuale per ottenere il valore iniziale da associare al parametro BWT.

Qualora non fosse presente una entry che corrisponda alla attuale coppia strumento finanziario-timeframe, viene restituito un insieme di definizione di default; se invece l'entry è presente nel file, vengono restituiti l'insieme di definizione del parametro BWT, il suo valore centrale da utilizzare per la corrente esecuzione della strategia ed il passo ovvero la differenza tra ciascun valore dell'insieme di definizione ed il suo successivo. Un ulteriore controllo viene effettuato sull'esistenza del file: se non venisse trovato alcun file di configurazione, al parametro BWT viene associato un insieme di definizione e dei valori in accordo al timeframe collegato alla corrente esecuzione. Si noti che si è scelto di associare ad i timeframe 1 e 5 minuti un valore centrale pari a 30 con passo 10, mentre per tutti gli altri timeframe un valore centrale pari a 70 con passo 30;

- Scrittura del valore di alcune variabili globali (come la stringa CROSS ed altri dati), output della lettura dalla DLL, nelle variabili locali di visualizzazione corrispondenti (v. Figura 173);
- Infine viene inizializzata con una stringa vuota la variabile locale di tipo stringa *SelfLearning-Date*, utilizzata nel controllo per eseguire il Self Learning, onde evitare di eseguire più iterazioni del modulo per la stessa iterazione della strategia (correntemente impostata ad una giornata).

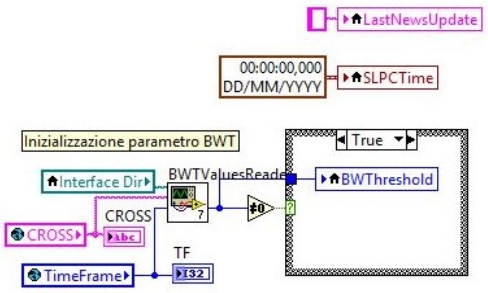


Figura 172 - Codice per l'inizializzazione della variabile di soglia minima della larghezza delle bande di Bollinger.

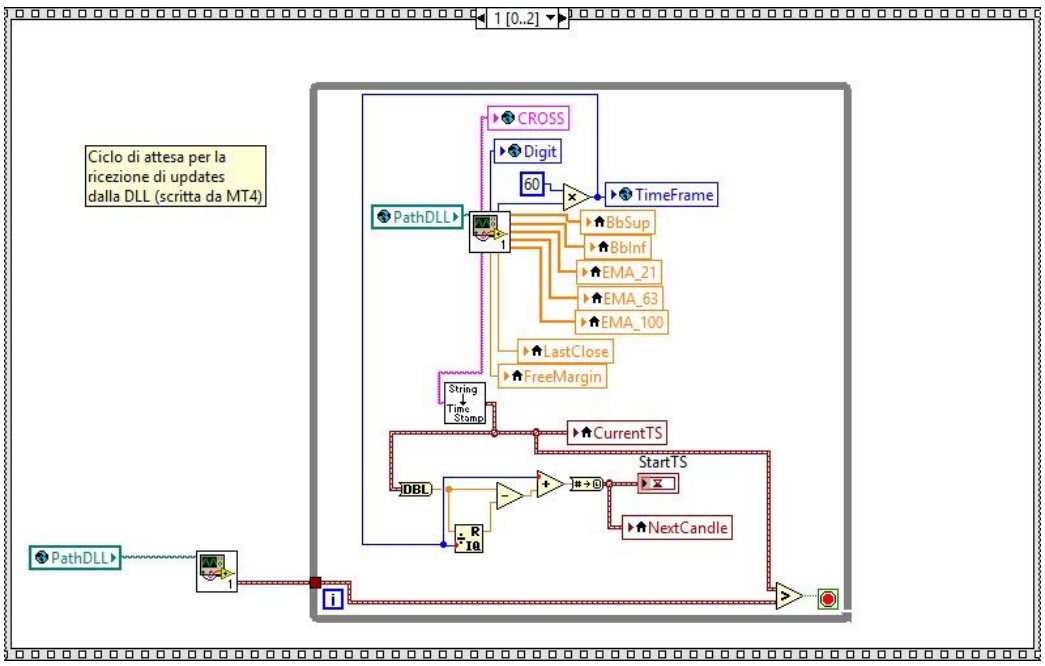


Figura 173 - Ciclo per l'inizializzazione delle variabili necessarie al funzionamento della strategia.

La quarta ed ultima sezione di quest'area è deputata alla gestione della scadenza temporale, impostata dall'utente tramite l'interfaccia e avente il formato giorno della settimana – ore - minuti, per l'esecuzione del Self Learning (v. Figura 174).

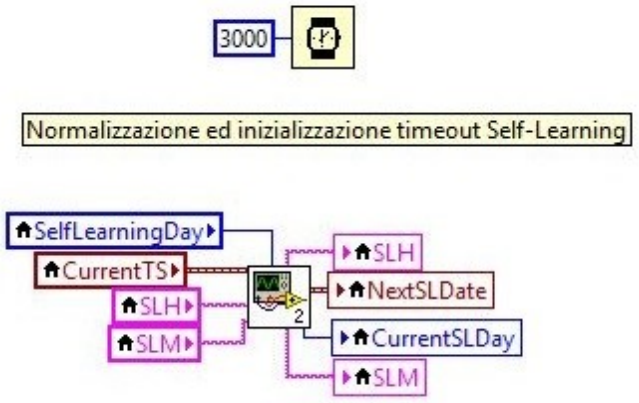


Figura 174 - Codice per la normalizzazione e l'inizializzazione del time-out per la fase di auto apprendimento.

In particolare vengono eseguite le seguenti operazioni:

- Normalizzazione dei valori di ore (variabile stinga *SLH*) e minuti (variabile stringa *SLM*):
  - *Parametro minuti*: se si è inserito un valore minore di zero, il parametro viene impostato a zero; se si è inserito un valore maggiore di 59, il valore viene riportato a 59.
  - *Parametro ore*: se si è inserito un valore minore di zero, il parametro viene impostato a 00; se si è inserito un valore maggiore di 23, il valore viene riportato a 23.
- Normalizzazione della scadenza: in caso l'utente abbia selezionato una scadenza successiva alle 22:59 del venerdì (e ovviamente precedente alla mezzanotte del lunedì), la scadenza viene reimpostata proprio alle 22.59 del venerdì in quanto corrisponde all'ultima possibilità utile di ricezione di aggiornamenti dalla piattaforma di trading MT4. Viene inoltre gestito il caso in cui l'utente abbia scelto l'opzione di Self Learning con cadenza giornaliera: in tal caso, infatti, il valore della variabile *SelfLearningDay* è pari a 7; occorre anzitutto controllare se le ore e minuti inseriti siano inferiori alle ore e minuti contenuti nel Timestamp corrente. Se è questo il caso, il valore del giorno del Self Learning viene memorizzato nella variabile *CurrentSLDay* con valore pari al giorno odierno; viceversa, si sceglie il giorno successivo, se esso non coincide con un sabato o una domenica.
- Il valore normalizzato di tali parametri, unitamente al Timestamp corrente vengono infine passati in input al vi *GetSelfLearningTimeStamp* per impostare la prossima scadenza temporale per l'esecuzione del modulo di auto apprendimento nella variabile locale *NextSLDate*.

Il vi *GetSelfLearningTimeStamp* riceve in input (v. Figura 175):

- Un Timestamp corrispondente all'attuale istante temporale;
- Il giorno della settimana a cui è stato associato un valore intero (0 per lunedì, 1 per martedì, e così via), sotto forma di ring enum;
- Due stringhe corrispondente alle ore e i minuti.

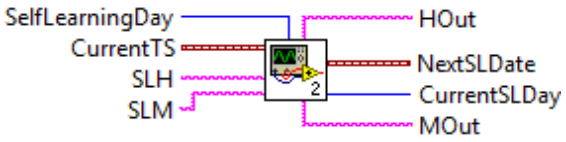


Figura 175 - Input/Output del vi per l'impostazione della prossima scadenza temporale per la fase di auto apprendimento.

Prima di tutto l'intero corrispondente al giorno della settimana viene trasformato in una stringa di tre caratteri (per permetterne il confronto con il valore restituito dall'oggetto LabView Format Date/Time String). Successivamente, ai fini della creazione del Timestamp di output, vengono distinti due diversi casi:

1. L'utente ha scelto una scadenza (giorno della settimana – ore - minuti) successiva a quella contenuta nel Timestamp corrente, ovvero è verificata almeno una delle seguenti tre condizioni sulla terna giorno settimanale – ore – minuti in input e quella contenuta nel Timestamp:
   i. I giorni della settimana non coincidono;
   ii. Le ore inserite sono maggiori di quelle contenute nel Timestamp;
   iii. I minuti inseriti sono maggiori di quelli contenuti nel Timestamp.

   In tal caso si determina innanzitutto il numero di giorni mancanti (trasformato poi nei secondi corrispondenti) per raggiungere il giorno della settimana richiesto. Successivamente viene creato un Timestamp per la copia nella variabile di output che, rispetto a quello in input, contiene alcuni valori aggiornati, quali il giorno del mese, il giorno della settimana, le ore ed i minuti (viene assunto che la scadenza temporale per il Self Learning abbia sempre secondi e millisecondi pari a zero).
2. L'utente ha scelto una scadenza (giorno della settimana – ore - minuti) precedente a quella contenuta nel Timestamp corrente, ovvero nessuna delle tre condizioni del punto 1 è verificata e, di conseguenza, l'utente ha inserito il giorno settimanale ed il numero di ore corrispondenti a quelli contenuti nel Timestamp corrente ed un numero di minuti non superiore. In questo caso si somma al Timestamp corrente un numero di secondi pari ad una settimana, ottenendo così un nuovo Timestamp restituito in output nella variabile *OutTS (v.* Figura 176*)*.

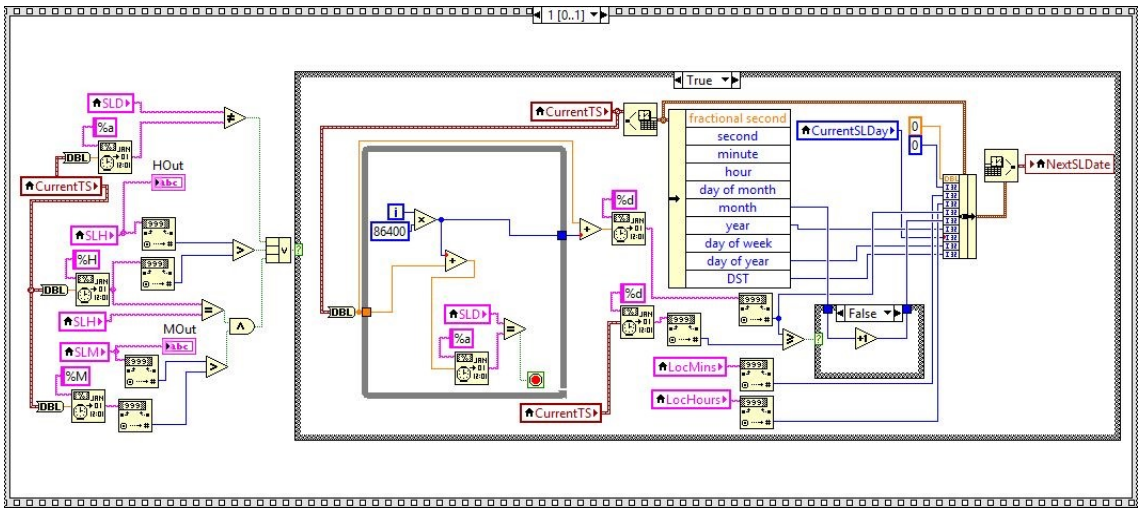


Figura 176 - Gestione dei Timestamp per l'esecuzione della fase di Self Learning scelto dall'utente.

7.10.2.2 *Area 2.2*

L'area 2.2 contiene il ciclo while interno e implementa la logica per l'esecuzione della strategia *LoMiT-TraS*. La condizione principale per l'esecuzione del ciclo è connessa al valore della variabile booleana *EndDay*, inizialmente settato a *False*; quando tale variabile assume valore *True*, il ciclo termina. All'ingresso nel ciclo viene anzitutto invocata la funzione di lettura dalla libreria *ReadFromDLL*, che in comunicazione con la piattaforma estrae il valore del Timestamp relativo al dato di prezzo più aggiornato presente sul server (v. Figura 177).

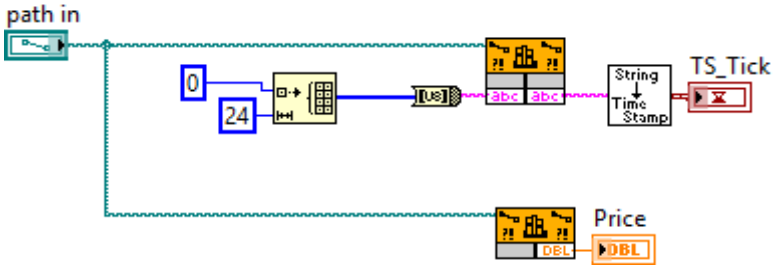


Figura 177 - Codice per la lettura della libreria di interfacciamento con MetaTrader 4.

Tale valore viene confrontato con un altro valore di tipo Timestamp, ottenuto dalla somma dell'istante temporale di inizio esecuzione della strategia ed il prodotto del numero di iterazioni della strategia (inizialmente zero) ed il timeframe attuale (ricordiamo espresso in secondi): se il valore letto dalla libreria è maggiore del valore di Timestamp ottenuto come poc'anzi indicato, allora si entra in un costrutto condizionale suddiviso in quattro macro-azioni per l'esecuzione della strategia. Con questo controllo sul Timestamp viene implementato il meccanismo di esecuzione della strategia soltanto a candela chiusa; a partire dal Timestamp di inizio esecuzione infatti, posto *X* il nostro timeframe e *Y* il numero di iterazioni già effettuate, il prossimo Timestamp letto dalla libreria e contenente i dati aggiornati rispetto alla prossima candela sarà certamente maggiore del Timestamp di inizio esecuzione incrementato di un numero di candele pari a quelle già analizzate dalla strategia nella sua corrente esecuzione.

Il valore del Timestamp viene ulteriormente incrementato di un numero di secondi pari alla prossima candela e scritto, all'interno del costrutto condizionale di strategia, nella variabile locale *NextCandle*, che viene quindi aggiornata ogni qual volta viene eseguita la strategia (v. Figura 178).

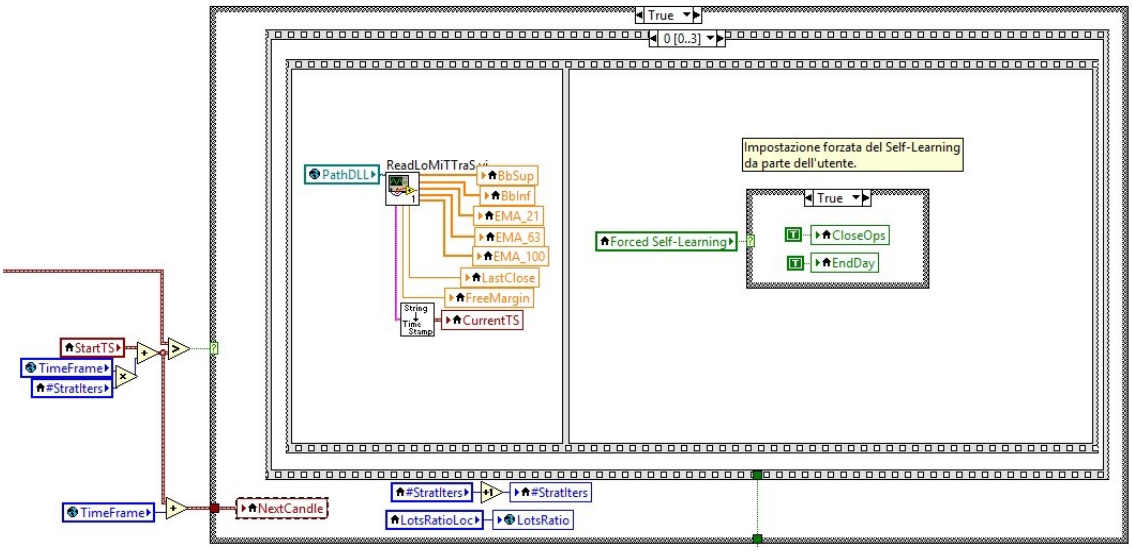


Figura 178 - Area di codice deputata ai controlli per l'esecuzione della strategia.

Se si è in possesso di dati aggiornati su una nuova candela ricevuti dalla piattaforma, dunque, si entra all'interno di un costrutto condizionale di strategia in cui, oltre ad aggiornare la variabile Timestamp *NextCandle*, viene incrementato il numero di iterazioni della strategia e riscritto il valore della variabile locale intera *LotsRatio* nella variabile globale intera *LotsRatio*; ciò consente all'utente, durante l'esecuzione della strategia, di modificare il numero di lotti di cui richiedere l'apertura alla piattaforma.

La prima delle quattro macro-azioni effettuate all'interno del costrutto condizionale di strategia consiste nella lettura dalla libreria DLL di tutte le informazioni aggiornate necessarie all'esecuzione della strategia.

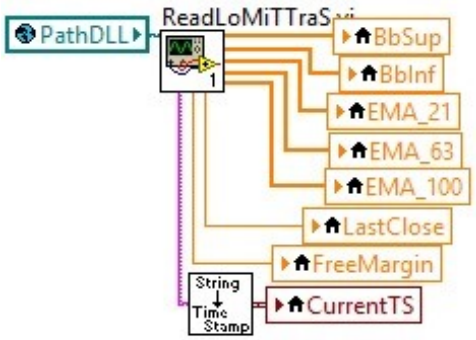


Figura 179 - Lettura dalla DLL dei dati di mercato aggiornati.

Attraverso il vi *ReadCalLoMiTTraS*, infatti, ponendo in input il percorso alla libreria, vengono estratti e memorizzati i valori di prezzo, Timestamp, margine libero, EMA e bande di Bollinger aggiornati (v. Figura 179).

La seconda macro-azione eseguita all'interno del costrutto condizionale di strategia attiene al controllo sulla eventuale imminente necessità di sospendere l'esecuzione della strategia, implementata attraverso l'attivazione della variabile booleana *CloseOps*.

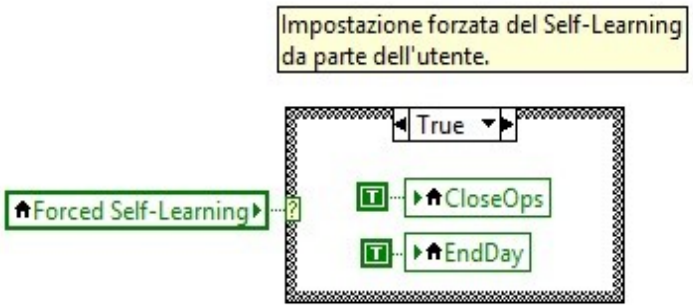


Figura 180 - Controllo per l'eventuale sospensione in caso di necessità di eseguire la fase di Self Learning.

Va osservato come tale eventualità possa avvenire non solo per logica applicativa ma anche a seguito dell'intervento manuale dell'utente sull'interfaccia grafica, essendo *CloseOps* un controllo booleano presente in interfaccia nella main GUI (v. Figura 180).

L'applicazione, al fine di attivare la sospensione dell'esecuzione della strategia, controlla se il Timestamp più aggiornato (*CurrentTS*) letto dalla DLL, incrementato di una quantità di secondi pari a due candele (e quindi dipendente dal timeframe corrente) più 10 secondi per permettere la ricezione di aggiornamenti nella maggior parte dei casi: in questo caso la variabile booleana *CloseOps* assume valore *True*.

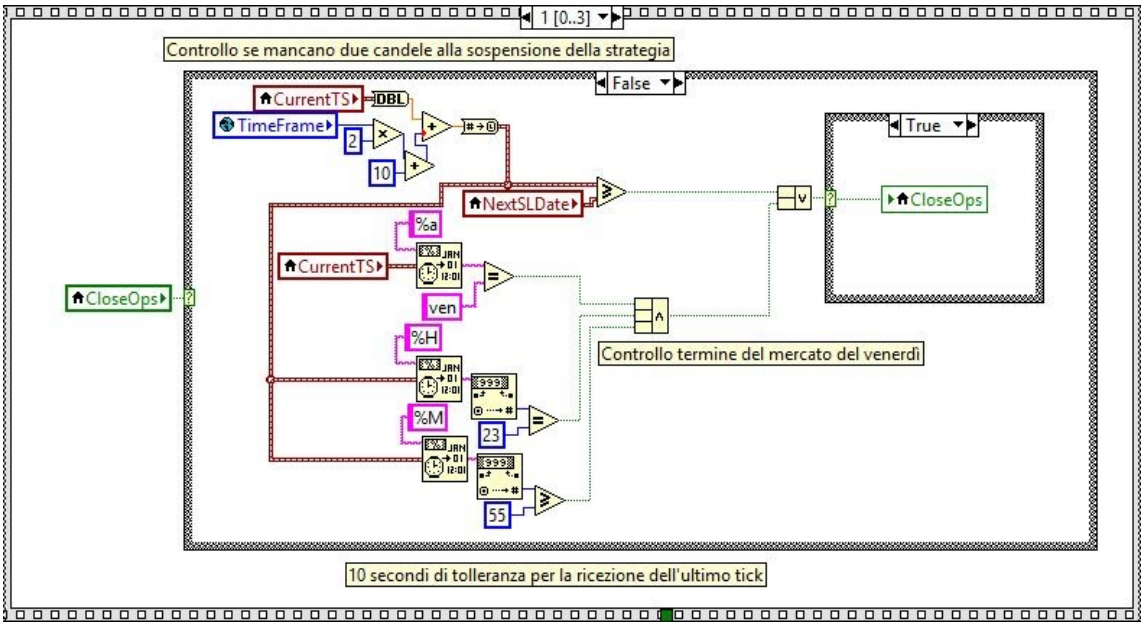


Figura 181 - Controllo del Timestamp ricevuto per la sospensione della strategia di trading.

La terza macro-azione corrisponde alla chiamata al vi *LoMiTTraS_ONLINE*, che implementa la strategia vera e propria. Oltre alle variabili globali, cui tale vi ha naturalmente accesso, alla strategia occorrono i seguenti valori in ingresso:

- Timestamp (*CurrentTS*) e prezzo (*LastClose*) aggiornati;
- Valori aggiornati degli EMA rispetto alle ultime 20 candele (*EMA_21, EMA_63 ed EMA_100*);
- Valori aggiornati delle bande di Bollinger rispetto alle ultime 20 candele (*BbSup* e *BbInf*);
- Margine libero del conto associato alla attuale esecuzione dell'applicazione (*FreeMargin*);

- Valore del parametro intero BandWidth Threshold, ottenuto dalla lettura del file di configurazione nel caso di prima esecuzione, o dalla esecuzione del modulo di Self Learning per tutte le altre esecuzioni (*BWThreshold*);
- Valore del parametro di prossimità alle bande di Bollinger (*LimitTolerance*);
- Media mobile da utilizzare (intero *ChooseEMA*, 0 per la media a 21 periodi, 1 per quella a 63 e 2 per quella a 100 periodi);
- Numero di cifre decimali significative rispetto all'attuale cross (*Digit*);
- Cartella di lavoro dell'attuale esecuzione (*InterfaceDir*);
- Variabile booleana che indica alla strategia di non aprire nuove posizione e richiedere la chiusura degli ordini attualmente a mercato (*CloseOps*).

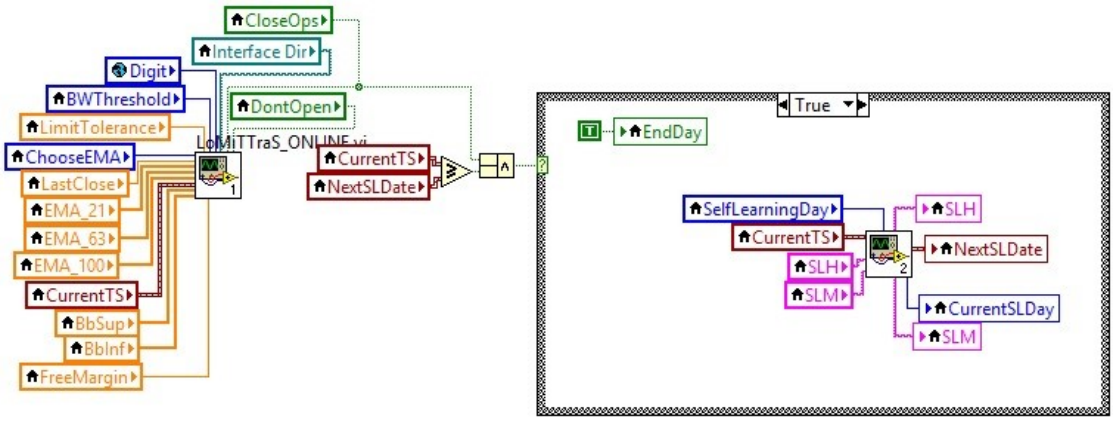


Figura 182 - Esecuzione della strategia di trading e controllo sull'esecuzione della fase di auto apprendimento.

In quest'area del codice viene inoltre effettuato il controllo sulla effettiva sospensione della esecuzione della strategia, attraverso l'attivazione della variabile booleana *EndDay*. Anche in questo caso, come accaduto per il controllo sulla chiusura forzata delle operazioni, i controlli si riferiscono alla scadenza temporale inserita dall'utente nell'interfaccia grafica (v. Figura 182).

Nel caso in cui la variabile *CloseOps* abbia valore True (è già stato dato il preallarme per segnalare che mancano meno di due candele alla sospensione), se il Timestamp corrente non è inferiore al Timestamp corrispondente alla scadenza temporale per il Self Learning, allora vanno sospese definitivamente le attività della strategia e preparata l'esecuzione del modulo di auto apprendimento. A tal fine, viene anzitutto attivata la variabile booleana *EndDay*; successivamente, viene computato il nuovo valore della scadenza temporale per il Self Learning nella variabile locale *NextSLDay* attraverso l'esecuzione del vi *GetSelfLearningTimeStamp*, a cui viene passato il valore contenuto in *CurrentSLDay* nei casi normali, mentre se attiva la variabile booleana *DailySL* (l'utente ha scelto di effettuare il Self Learning con cadenza giornaliera) viene passato il valore incrementato di una unità, corrispondente al giorno successivo (v. Figura 182).

La quarta ed ultima macro-azione che la strategia deve svolgere concerne l'aggiornamento di alcune variabili e di alcuni indicatori d'interfaccia (v. Figura 183).

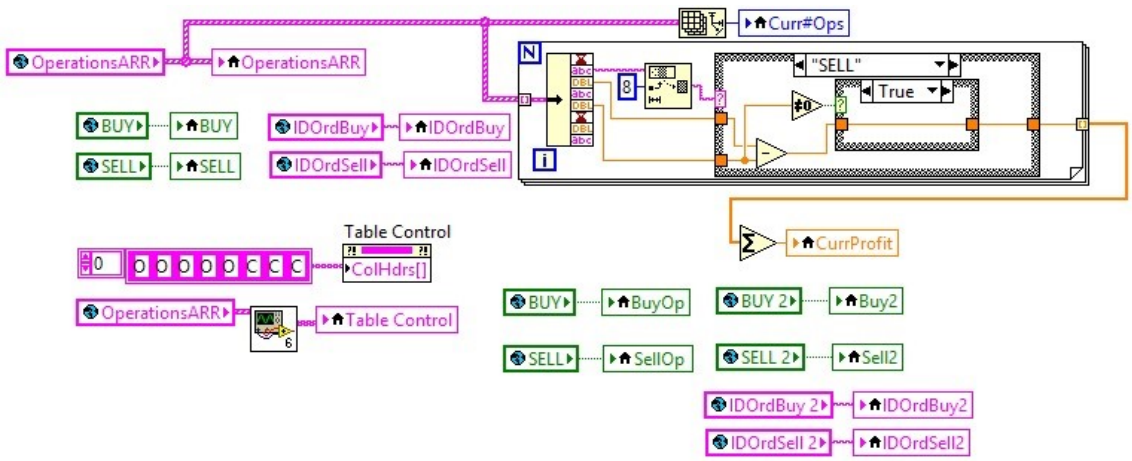


Figura 183 - Operazioni di aggiornamento variabili dopo una run della strategia di trading.

Nello specifico:

- Contenuto del vettore di cluster delle operazioni globale in quello locale per la visualizzazione;
- Valore delle variabili booleane globali *BUY* e *SELL* nelle variabili booleane locali *BUY*, *SELL*, *BuyOp* e *SellOp* (contenute in due differenti schede dell'interfaccia grafica) per indicare la presenza di ordini sul mercato;
- Aggiornamento delle variabili stringa globali *IDOrdBuy* e *IDOrdSell* nelle variabili stringa locali *IDOrdBuy* e *IDOrdSell* per visualizzare il ticket relativo agli ordini sul mercato;
- Valore intero relativo alla dimensione del vettore di cluster delle operazioni globali all'interno della variabile intera *Curr#Ops* per visualizzare sull'interfaccia il numero di operazioni finora effettuata dalla strategia nella sua corrente esecuzione;
- Aggiornamento del profitto relativo a ciascuna operazione chiusa, avendo cura quindi di non tenere in conto i profitti relativi ad operazioni ancora a mercato. Il valore derivante dalla sommatoria di tutti i profitti connessi a ciascuna singola operazione viene scritto nella variabile locale double *CurrProfit* per la sua visualizzazione in interfaccia;
- Nel caso in cui l'esecuzione della strategia debba essere sospesa, si inizializza nuovamente l'header della tabella in interfaccia TableControl per la visualizzazione dello storico di tutti gli ordini relativi alla corrente esecuzione della strategia attraverso l'esecuzione del sub-vi *FillOpsTable*;
- Il valore della variabile booleana di controllo *EndDay*, infine, viene passato al controllo della condizione di terminazione del ciclo interno while.

Figura 184 - Input/Output del vi FillOpsTable per la scrittura nella tabella delle operazioni.

Il sub-vi *FillOpsTable* (v. Figura 184) riceve in ingresso il vettore globale di cluster delle operazioni e, accedendo al valore di ciascun elemento di ciascun cluster presente nel vettore, li ordina in coerenza con gli header della tabella e ne scrive al suo interno il contenuto, che viene posto in output a termine del ciclo (v. Figura 185).

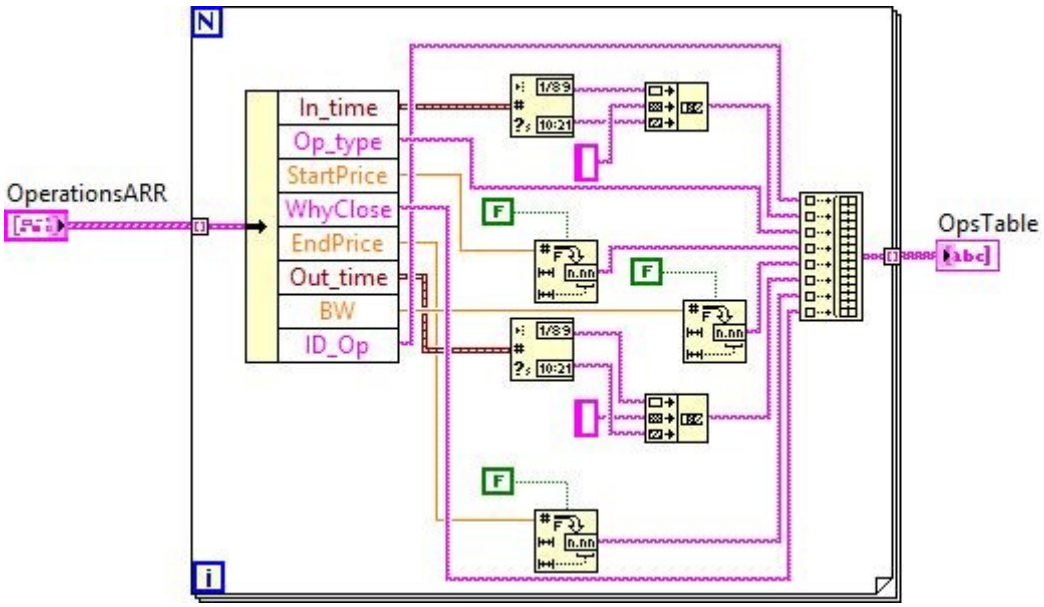


Figura 185 - Codice per la scrittura dei valori contenuti nel vettore di cluster delle operazioni all'interno della tabella delle operazioni.

7.10.2.3 *Area 2.3*

L'area 2.3 esegue le seguenti operazioni, utili al reset di variabili utilizzate durante l'esecuzione della strategia (v. Figura 186):

- Azzeramento del vettore globale di cluster delle operazioni *OperationsARR*;
- Azzeramento del vettore globale di double dei profitti *Profits*;
- Azzeramento delle variabili intere globali relative al numero di operazioni effettuate *#Ops*, all'indice delle correnti operazioni di Buy e di Sell *IndB* ed *IndS*;
- Reset della variabile globale indicante la banda di Bollinger di interesse *Band* al suo valore di default *NoBB*;
- Disattivazione delle variabili booleane globali *BUY* e *SELL*;
- Azzeramento delle variabili double globali relative al prezzo di apertura delle operazioni di Buy e di Sell *StartPriceBuy* e *StartPriceSell*;
- Impostazione del valore di default, pari a $-\infty$, della variabile double *FinalProfit*;
- Reset al suo valore di default (costante nulla) della variabile *refnum* riferita al valore intero associato al file di report giornaliero;
- Settaggio al valore stringa vuota delle variabili riferite agli ID delle operazioni *IDOrderBuy* ed *IDOrderSell*;
- Settaggio al loro valore di default (costante vuota) delle variabili *Timestamp* globali relative al Timestamp di apertura delle operazioni di Buy e di Sell *OpenTimeStampBuy* ed *OpenTime-StampSell*.

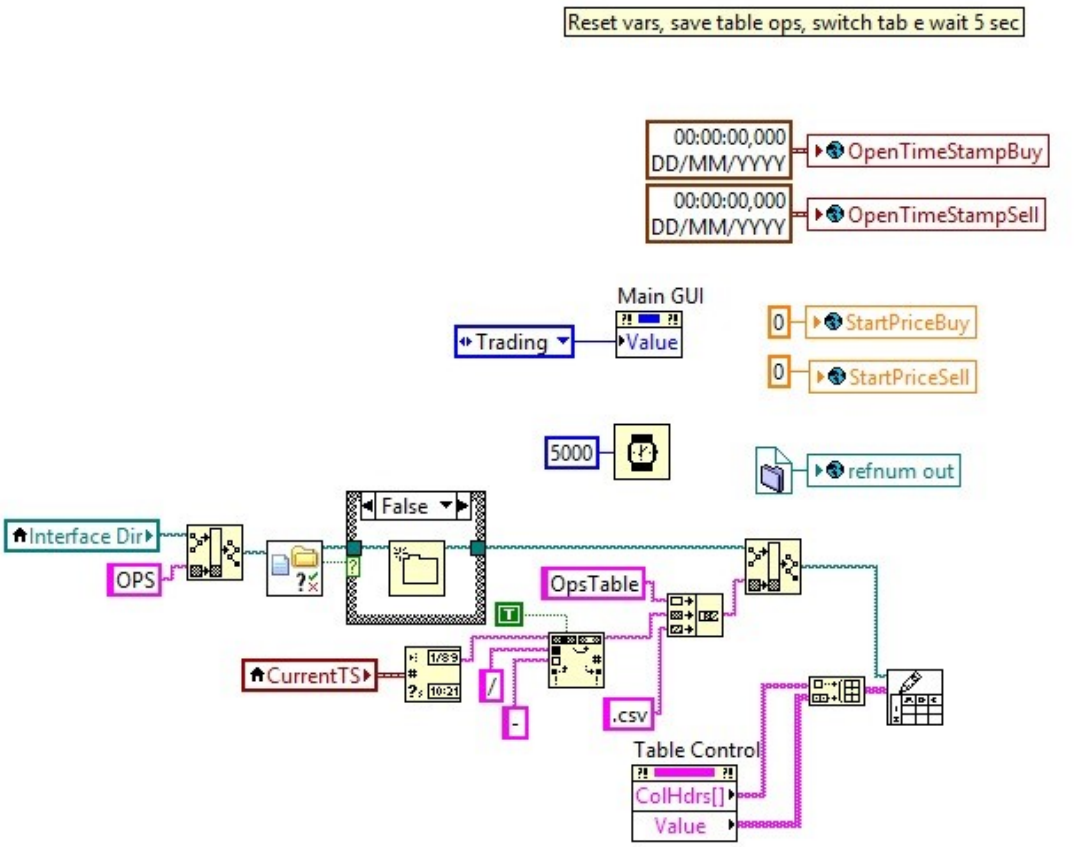


Figura 186 - Area deputata al reset delle variabili e propedeutica all'esecuzione della fase di Self Learning.

Completate queste operazioni, il controllo passa all'Area 3.

7.10.3 Area 3

L'area 3 implementa la lettura del file di dump, la creazione della cartella di lavoro del modulo di Self Learning e la suddivisione del file di dump in file giornalieri adatti ad essere estratti dal modulo di auto apprendimento.

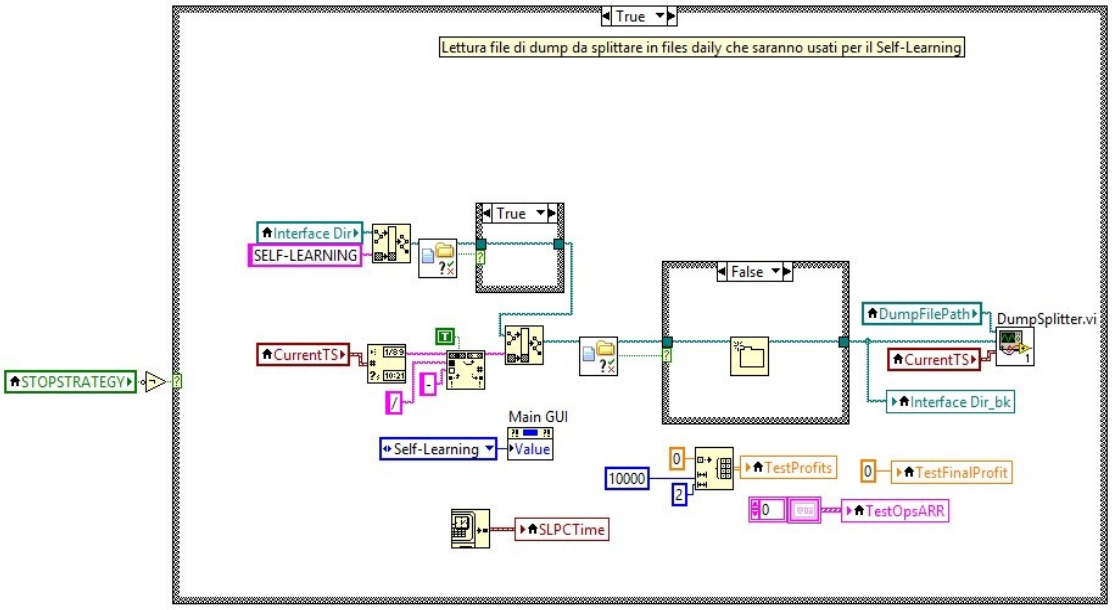


Figura 187 - Codice per le operazioni preliminari alla fase di Self Learning.

A partire dal Timestamp più aggiornato *CurrentTS*, viene creata una cartella di lavoro per il modulo di Self Learning avente come nome la data odierna, al fine di poter identificare l'iterazione del Self Learning ed evitare che vengano effettuate più esecuzioni per la stessa data (v. Figura 187). Successivamente Viene invocato il vi *DumpSplitter*, incaricato di costruire i files giornalieri a partire da un unico file di dump.

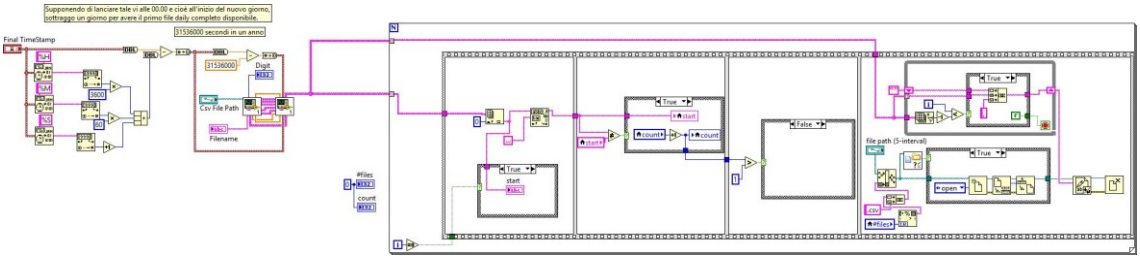


Figura 188 - Codice per l'ottenimento dei files di dump giornalieri.

Tale vi prende in ingresso il percorso al file di dump, quello relativo alla cartella di lavoro corrente ed il Timestamp più aggiornato; a tale Timestamp viene sottratto un numero di secondi tale da riportarlo alle 23.59 del giorno precedente al Timestamp in input, fornendo così il limite temporale superiore. Il limite temporale inferiore è invece calcolato sottraendo una quantità di secondi pari al numero di secondi in un anno (31536000), ottenendo così un Timestamp pari ad esattamente un anno precedente il Timestamp in ingresso al vi, utilizzato come limite temporale inferiore. In questo modo è possibile estrapolare dal file di dump, tramite i vi di uso comune *GetValueFromFileDS* e *GetSubPeriodDS,* l'intervallo temporale di dati necessario ad eseguire la fase di auto apprendimento. I dati vengono infine suddivisi in un numero di file pari al numero di giorni appartenenti all'intervallo temporale scelto (v. Figura 188).

7.10.4 Area 4

L'area 4 è deputata all'esecuzione del Self Learning e può essere ulteriormente suddivisa in quattro sotto aree, eseguite iterativamente per ciascun file giornaliero presente nella cartella di lavoro del Self Learning.

Per ogni file presente nella cartella di lavoro del Self Learning, vengono eseguite le operazioni, suddivise come anticipato in quattro sotto aree funzionali. Si utilizza una variabile di backup della variabile *InterfaceDir*, denominata *InterfaceDir_bkbk,* per la necessità di apportare delle modifiche al path stesso volendo però evitare di sovrascrivere il valore originale contenuto nella variabile *InterfaceDir*, che è in uso nelle esecuzioni al di fuori del modulo di Self Learning.

Viene inoltre settato a zero il valore della variabile intera *Zero_Ops*, che sarà utilizzato per indicare il numero di iterazioni del Self Learning che hanno dato a luogo ad un numero di operazioni di trading pari a zero.

7.10.4.1 *Area 4.1*

L'area 4.1 esegue crea prima di tutto la cartella di lavoro del corrente file di dump giornaliero csv; la cartella creata ha lo stesso nome del file della corrente iterazione ed il path risultante viene salvato nella variabile di tipo path InterfaceDir_bkbk, ad indicare due livelli di profondità rispetto all'InterfaceDir che è cartella di lavoro dell'intero applicativo.

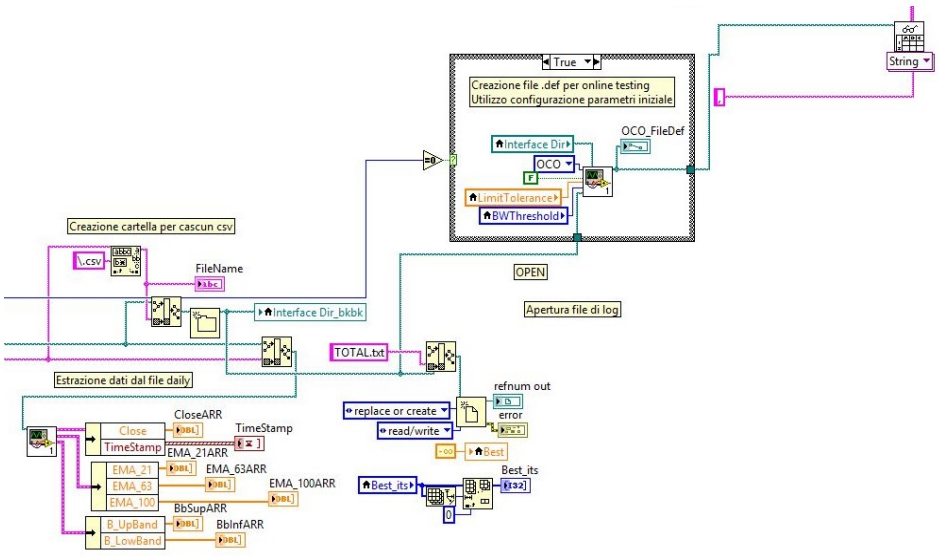


Figura 189 - Area del codice relativa alle prime operazioni della fase di auto apprendimento.

Successivamente il file di dump giornaliero viene elaborato tramite il vi *GetValueFromFile* per ottenere le strutture dati necessarie alla strategia LoMiTTraS versione offline per il Self Learning.

Si noti che, a differenza dei valori necessari alla strategia nella sua versione online, per la strategia offline i vettori contengono i valori relativi a tutte le candele le cui informazioni sono state estratte dal file di dump.

Viene inoltre creato report per ciascuna iterazione del Self Learning associato al corrente file csv. Vengono inizializzate le variabili che saranno utilizzate in questa iterazione, la variabile *Best* corrispondente al profitto massimo ottenuto nella corrente iterazione, la variabile *ZeroOps* per indicare un numero di operazioni pari a zero per questa iterazione, il vettore *Best_its* contenente l'indice delle iterazioni che hanno dato luogo al massimo profitto.

Occorre inoltre conoscere quali valori dei parametri *BandWidth Threshold* (BWT), *Limit Tolerance* (LT) e *ChooseEMA* il modulo di Self Learning dovrà testare ed utilizzare per migliorare le performance della strategia online. A tale scopo, viene creato un file di definizione *.def* nel quale, per ciascuna riga numerata in maniera univoca a partire da 1, sono presenti i valori che ciascun parametro deve assumere ai fini del test.

Per la creazione del file di definizione è previsto un diverso comportamento tra la prima iterazione e le successive: nel caso in cui è in esecuzione la prima iterazione del modulo di auto apprendimento, corrispondente al primo file csv, viene creato un file *.def* ad-hoc. Ciò avviene attraverso l'invocazione del vi *AI_Engine_LoMiTTraS*. Ciò equivale a generare un file di definizione in cui saranno testati i parametri sui loro insiemi di definizione completi, dando luogo così a 216 configurazioni possibili complessive. Tali configurazioni vengono salvate sul file di definizione *.def*.

Nel caso in cui, invece, siano già state effettuate altre iterazioni, viene utilizzato il file di definizione presente al percorso indicato da *OCO_FileDef*, creato in una porzione successiva del codice.

L'insieme dei record estratti dal file di definizione viene posto in input alla seconda sotto-area funzionale (v. Figura 189).

7.10.4.2 *Area 4.2*

L'area 4.2 riceve in ingresso l'insieme dei record estratti dal file di definizione e scorre iterativamente ciascun record contenente la configurazione parametrica da testare (v. Figura 190).

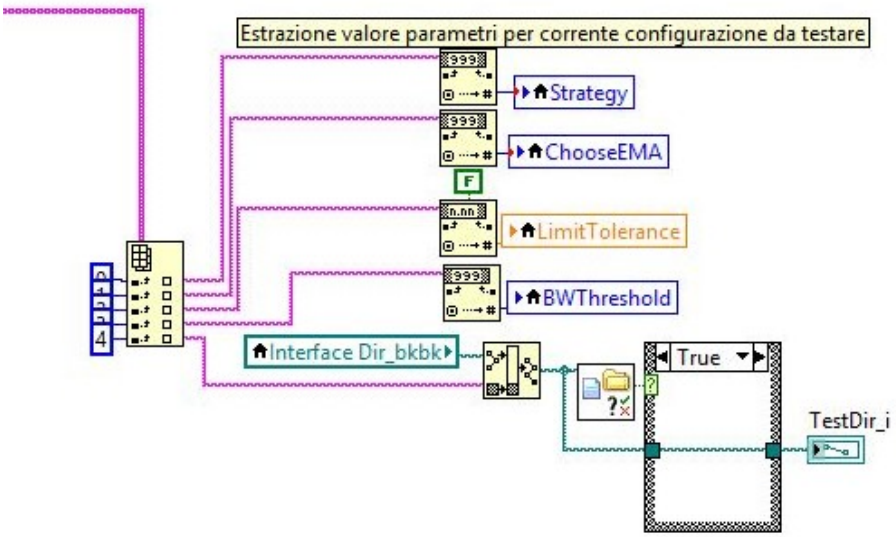


Figura 190 - Porzione di codice per l'esecuzione di ciascuna configurazione dei parametri da sottoporre a test.

Ciascuna record contiene i seguenti parametri:

- Un intero che specifica la tipologia di strategia da testare denominato *Strategy*. Nella sua versione attuale l'unica strategia possibile è la strategia LoMiTTraS (OCO).
- Un intero che indica quale media mobile utilizzare *ChooseEMA*;
- Un double che indica il valore del parametro *Limit Tolerance*;
- Un intero che indica il valore del parametro *BandWidth Threshold* BWT;
- Il numero univoco che identifica la configurazione preceduto dalla dicitura *TEST*. La stringa così formata viene utilizzata per creare la cartella di lavoro della corrente iterazione del test parametrico.

Dopo aver estratto il valore dei parametri per la corrente iterazione viene richiamato il vi *LoMiTTraS_OFFLINE* avente in ingresso i seguenti valori:

- Vettore dei Timestamp (*Timestamp*) e vettore dei prezzi di chiusura delle candele (*CloseARR*) estratti dal file di dump giornaliero;
- Vettori dei valori degli EMA (*EMA_21ARR*, *EMA_63ARR* ed *EMA_100ARR*) estratti dal file di dump giornaliero;
- Vettori dei valori delle bande di Bollinger (*BbSupARR* e *BbInfARR*) estratti dal file di dump giornaliero;
- Numero di cifre decimali significative rispetto all'attuale cross (*Digit*);
- Valore del parametro intero BandWidth Threshold (*BWThreshold*);
- Valore del parametro di prossimità alle bande di Bollinger (*LimitTolerance*);
- Media mobile da utilizzare (intero *ChooseEMA*, 0 per la media a 21 periodi, 1 per quella a 63 e 2 per quella a 100 periodi);
- Cartella di lavoro dell'attuale esecuzione (*TestDir_i*).

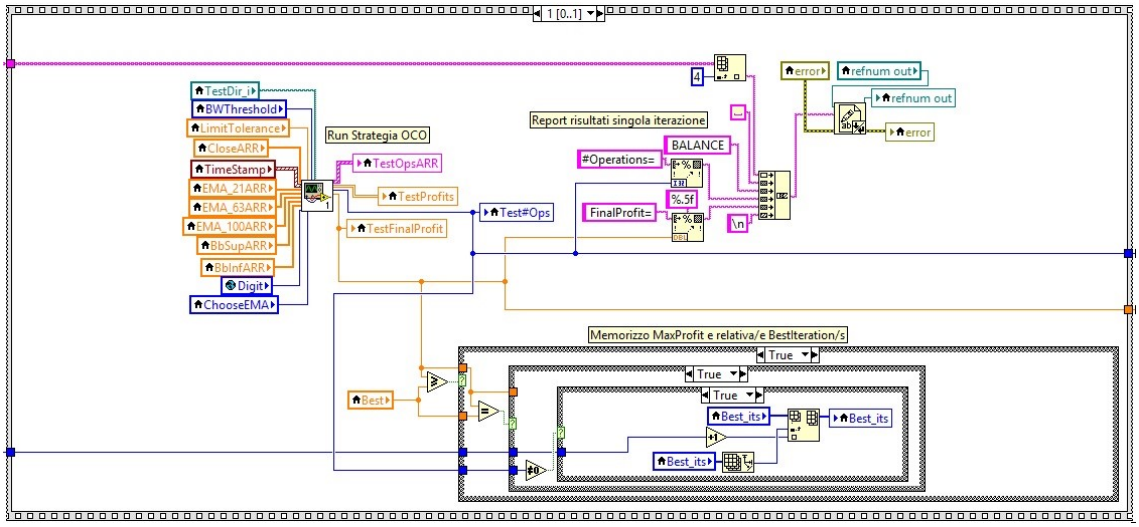


Figura 191 - Esecuzione della versione offline della strategia per la fase di Self Learning.

Come anticipato precedentemente, la versione offline della strategia, a differenza della versione online, restituisce anche degli output:

- *TestOpsARR*, array di cluster delle operazioni riferito alla corrente iterazione di test;
- *TestProfits*, vettore di double che memorizza i profitti delle singole operazioni;
- *Test#Ops*, intero che indica il numero di operazioni effettuate;
- *TestFinalProfit*, valore double che indica il profitto complessivo della corrente iterazione di test.

L'esito delle operazioni relative alla corrente iterazione viene non solo scritto su un file di report (creato nella sotto area 4.1) ma anche memorizzato in opportune strutture dati (v. Figura 191).

In particolare, per ogni iterazione viene verificato se si è in presenza di un nuovo massimo profitto. Se il profitto della corrente iterazione, quindi, è maggiore o uguale al profitto massimo finora osservato, possono verificarsi due casi:

1. Il profitto corrente è strettamente maggiore del massimo: il nuovo massimo, indicato dal valore della variabile double *Best*, viene aggiornato col valore di profitto corrente. Inoltre il vettore *Best_its* viene inizializzato ed in esso viene scritto il numero di iterazione corrente incrementato di 1 (dal momento che le iterazioni partono da zero, mentre le configurazioni possibili sono numerate a partire da uno).
2. Il profitto corrente è uguale al profitto memorizzato nella variabile double *Best*: occorre ora distinguere due ulteriori diversi casi connessi al numero di operazioni effettuate:
    a. Se il profitto massimo corrente (uguale al massimo finora osservato) si è ottenuto con un numero di operazioni pari a zero (equivale a dire che il massimo profitto è zero in quanto non si sono effettuate operazioni), allora non occorre memorizzare alcuna informazione sulla iterazione corrente.
    b. Se, di contro, il massimo corrente è uguale al valore in *Best* e *Test#Ops* è diverso da zero allora si è in presenza di una ulteriore configurazione parametrica che dà luogo

allo stesso profitto massimo già precedentemente osservato. Per questo motivo, si inserisce un nuovo elemento nel vettore delle migliori iterazioni *Best_its* contente la corrente iterazione (sempre incrementata di 1), mantenendo invariato il valore della variabile Best.

7.10.4.3 *Area 4.3*

L'area 4.3 si occupa principalmente di adempiere a due diverse funzionalità: la scelta del valore ottimale per i parametri *BWT* e *LT* (con particolare enfasi sul primo parametro, BWT, dal momento che il suo impatto sulle performance della strategia è prevalente rispetto al secondo) e la scrittura su un file di alcune statistiche estrapolate dal test sul corrente file (v. Figura 192).

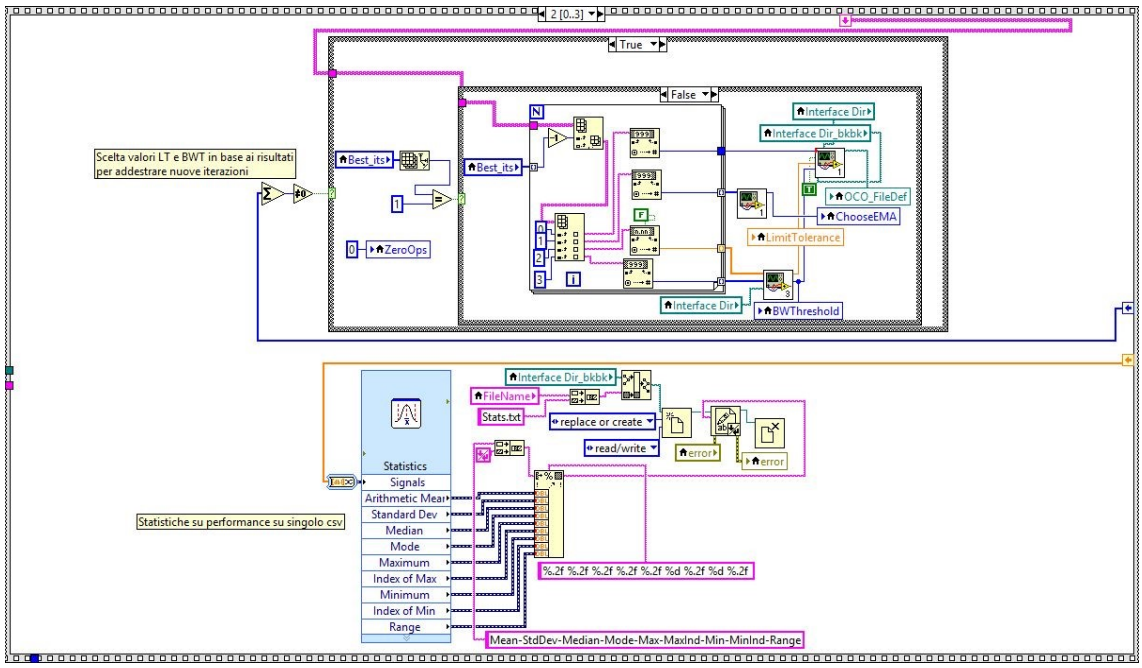


Figura 192 - Codice per la scelta dei valori ottimali per i parametri di controllo BWT ed LT.

L'euristica per la scelta della migliore configurazione per i parametri *BWT* e *LT* distingue due differenti casi:

1. Il numero totale di operazioni compiute dalla strategia offline sul corrente file di test è pari a zero: Si verifica il valore della variabile *ZeroOps*, relativa il numero di volte consecutive in cui la strategia non ha effettuato operazioni. Se tale numero è minore o pari a due, allora si seleziona il valore modale di *BWT*, ovvero quello con maggiore frequenza tra i valori che hanno dato luogo al massimo profitto, mediante l'invocazione del sub vi *GetBWTModalValue*. Tale valore di *BWT*, insieme al valore di default per il parametro LT (0,15), viene posto in ingresso al vi *AI_Engine_LoMiTTraS* per la generazione del file di definizione da utilizzare nella prossima iterazione del Self Learning. Se, invece, *ZeroOps* risulta essere maggiore di due (si sono avute più di due iterazioni consecutive con numero di operazioni pari a zero) allora verrà utilizzato il valore centrale dell'insieme di definizione di *BWT*, ottenuto attraverso la chiamata al metodo *BWTValuesReader*.

Tale valore viene infine confrontato col valore attuale di *BWT*: se il valore corrente del parametro è maggiore del valore ottenuto dal sub vi *BWTValuesReader*, allora occorre riportarlo al suo valore centrale di definizione in quanto è probabile che sia andato fuori range, e potrebbe essere questa la ragione per cui si sono avute diverse iterazioni con numero di iterazioni pari a zero. Viene così chiamato il vi *AI_Engine_LoMiTTraS* con variabile di controllo *Update* impostata a *False*, così da forzare il test dei parametri sugli interi insiemi di definizione per la successiva iterazione (come avvenuto per la prima iterazione).

2. Il numero totale di operazioni compiute dalla strategia offline sul corrente file è diverso da zero: si riporta quindi la variabile *ZeroOps* al suo valore originale e si verifica se il profitto massimo è relativo ad una o più configurazioni parametriche:
    a. Nel caso di una sola configurazione parametrica si estraggono i relativi valori per i parametri *BWT*, *LT* e *ChooseEMA* e si crea il nuovo file di definizione .def da utilizzare alla prossima iterazione del modulo di Self Learning.
    b. Se, al contrario, nel vettore *Best_its* sono presenti diversi elementi, allora si estrapolano tutti i valori di *BWT*, *LT* e *ChooseEMA* per cui si è avuto il profitto massimo e, grazie all'impiego del sub vi *BWTUpdater* ed *EMACounter*, si seleziona il valore migliore per i parametri. Tali valori sono posti in ingresso al vi *AI_Engine_LoMiTTraS* con variabile di controllo *Update* a *True*. Ciò implica che il nuovo file di definizione generato conterrà esattamente 75 configurazioni parametriche, corrispondenti ai valori che i parametri possono assumere.

Infine, Nell'area di codice 4.3, viene utilizzato il vettore dei profitti complessivi ottenuti con ciascuna delle possibili configurazioni parametriche per estrapolare una analisi statistica, riportata su file testuale. Nello specifico le informazioni statistiche estratte riguardano:

- media aritmetica
- deviazione standard;
- mediana;
- moda;
- massimo e relativo indice di massimo;
- minimo e relativo indice di minimo;
- intervallo di variazione.

7.10.4.4 *Area 4.4*

L'area 4.4 effettua le operazioni finali per ciascuna iterazione principale del modulo di Self Learning.

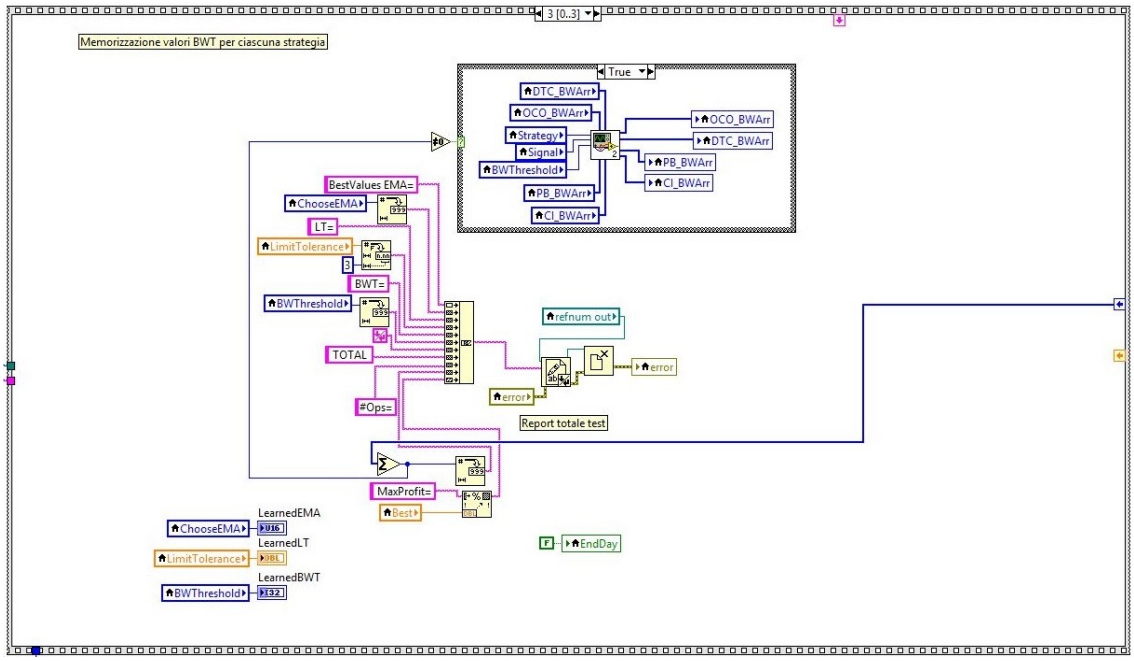


Figura 193 - Memorizzazione degli esiti della fase di auto apprendimento.

Nel caso in cui nell'iterazione di Self Learning sul corrente file di dump siano state effettuate effettivamente operazioni (a prescindere dal profitto che abbiano generato), occorre tenere traccia delle performance ottenute in relazioni al valore che i parametri hanno assunto, affinché tali informazioni possano essere usate nelle iterazioni successive. In particolare, tale operazione è necessaria per la scelta del valore modale per il parametro *BandWidth Threshold*. A tal fine viene invocato il sub vi *InsertBWTIntoArray* il quale riceve in input i vettori dei valori che il parametro BWT ha assunto nelle differenti strategie testate, inserisce il nuovo valore BWT, corrispondente al profitto massimo per la corrente iterazione, restituendo in output i vettori ordinati in senso crescente. Tali vettori potranno così essere utilizzati nelle iterazioni successive per la selezione del valore modale di BWT.

Viene inoltre completata la scrittura sul file di report della corrente iterazione indicando quali valori di *LT*, *BWT* e *ChooseEMA* sono risultati ottimali, il massimo profitto ad essi collegato ed il numero di operazioni effettuate.

L'ultima operazione da effettuare, affinché l'esecuzione dell'applicativo possa riprendere dalla strategia *LoMiTTraS* non appena si riceveranno nuovi aggiornamenti dalla piattaforma consiste nell'impostazione della variabile booleana di controllo *EndDay* a *False* (v. Figura 193).

7.11 Aspetti Implementativi - LoMiTTraS

In questo capitolo verrà analizzata nel dettaglio l'implementazione della strategia di trading, oggetto di questo capitolo, nelle sue due versioni, ponendo particolare enfasi alle scelte implementative e documentando, al contempo, gli strumenti virtuali (vi) che essa utilizza.

7.11.1 Core Engine

La strategia di base, da un punto di vista concettuale, è stata documentata in precedenza; pur avendo fornito dettagli sulle basi teoriche su cui tale strategia si basa, è necessario fornire qualche informazione aggiuntiva sulla parte implementativa.

La strategia prende in ingresso valori che concernono gli aggiornamenti ricevuti dalla piattaforma di trading, informazioni sul conto e lo strumento finanziario ad essa collegato, il valore dei parametri e dei percorsi file per il suo funzionamento. Nello specifico:

- *CurrentTS*, variabile di tipo Timestamp che coincide con l'aggiornamento più recente ricevuto dalla piattaforma;
- *LastClose*: variabile di tipo double che corrisponde al prezzo più recente ricevuto dalla piattaforma;
- *BbSup*: vettore di tipo double che indica i valori assunti dalla banda superiore di Bollinger in corrispondenza delle ultime 20 candele;
- *BbInf*: vettore di tipo double che indica i valori assunti dalla banda inferiore di Bollinger in corrispondenza delle ultime 20 candele;
- *EMA_21*: vettore di tipo double che indica i valori assunti dalla media mobile a 21 periodi in corrispondenza delle ultime 20 candele;
- *EMA_63*: vettore di tipo double che indica i valori assunti dalla media mobile a 63 periodi in corrispondenza delle ultime 20 candele;
- *EMA_100*: vettore di tipo double che indica i valori assunti dalla media mobile a 100 periodi in corrispondenza delle ultime 20 candele;
- *FreeMargin*: variabile di tipo double che indica il margine libero aggiornato del conto attualmente collegato alla piattaforma;
- *Digit*: variabile di tipo intero che indica il numero di cifre decimali significative relative allo strumento finanziario attualmente collegato alla piattaforma;
- *ChooseEMA*: variabile di tipo intero che indica quale tipo di media mobile si intende utilizzare per le strategie di uscita (0 per la media a 21 periodi, 1 per quella a 63 e 2 per quella a 100 periodi);
- *LimitTolerance*: variabile di tipo double che indica il margine percentuale da aggiungere o sottrare alle bande di Bollinger per verificarne lo sfondamento ad opera del prezzo;
- *BWThreshold*: variabile di tipo intero che corrisponde al limite inferiore della larghezza di banda di Bollinger, al di sotto del quale la strategia decide di non operare (non apre posizioni);
- *InterfaceDir*: variabile di tipo path che rappresenta la cartella di lavoro principale dell'applicativo;
- *CloseOps*: variabile di tipo booleano che indica l'approssimarsi della sospensione della strategia; essa serve principalmente a richiedere la chiusura delle operazioni a mercato e effettuare il report sugli esiti giornalieri della strategia.

Oltre ai suddetti input, la strategia può inoltre contare, così come ogni altro vi all'interno del progetto, su una serie di variabili con scope globale, documentate nel paragrafo 7.2.

Nella prima sezione del codice, oltre alla presenza dei controlli e degli indicatori che la strategia utilizza, vi è l'inizializzazione del vettore *ChoosedEMA*, in base alla scelta fatta tramite l'impostazione della variabile relativa al 0, 1 o 2, corrispondente rispettivamente alle medie 21, 63 e 100 periodi.

Tutto il codice della strategia può essere descritto come suddiviso in 6 aree, che coincidono peraltro con la suddivisione dello stesso in una struttura stacked sequence costituita da 6 frame, numerati da 0 a 5.

In ingresso ed in uscita a tale struttura e, di conseguenza, alla complessiva logica della strategia, vi è il vettore globale dei cluster delle operazioni; ciò si rende necessario in quanto durante la sua esecuzione, sia quando vengono aperte nuove posizioni, sia quando ne vengono chiuse di già aperte, la strategia modifica tale vettore globale. Dal momento inoltre che il vettore globale *OperationsARR* contiene le informazioni su tutte le operazioni effettuate durante la corrente esecuzione dell'applicativo principale, è indispensabile che si tenga traccia dei valori in esso contenuti prima dell'esecuzione della strategia.

Nel primo dei sei frame (quello di indice zero) della struttura stacked sequence interna, i valori delle bande di Bollinger vengono utilizzati per calcolare alcuni dati ad esse collegati. In particolare:

- Viene calcolata la larghezza di banda *CurrFullBW* come differenza tra il valore delle bande di Bollinger. Tale valore viene calcolato sia in relazione all'ultima candela chiusa *Curr-FullBW*, sia per le ultime venti candele *FullBW*.
- Viene calcolata la larghezza di banda di Bollinger come il rapporto tra la differenza tra il valore delle bande di Bollinger ed il valore della media a 21 periodi.
- Viene calcolato il valore del parametro *Curr%b*, che indica la posizione del valore del prezzo rispetto alle bande di Bollinger, come segue:

$$Curr\%b = \frac{Close_i - BBInf_i}{BBSup_i - BBInf_i} = \frac{LastClose - BBInf}{CurrFullBW}$$

dove *LastClose* corrisponde al valore di prezzo più aggiornato e *BBSup* e *BBInf* corrispondono al valore delle bande di Bollinger relative all'ultima candela chiusa, ovvero al valore contenuto nei rispettivi vettori all'ultima posizione *n*-esima.

Inoltre viene aperto in lettura (in modalità append) il file di log giornaliero, il cui nome è "Log_" seguito dalla data odierna, dal momento che la corrente iterazione della strategia riguarderà l'arco dell'intera giornata odierna. L'intero associato a questo file viene scritto nella variabile globale refnum out (v. Figura 194).

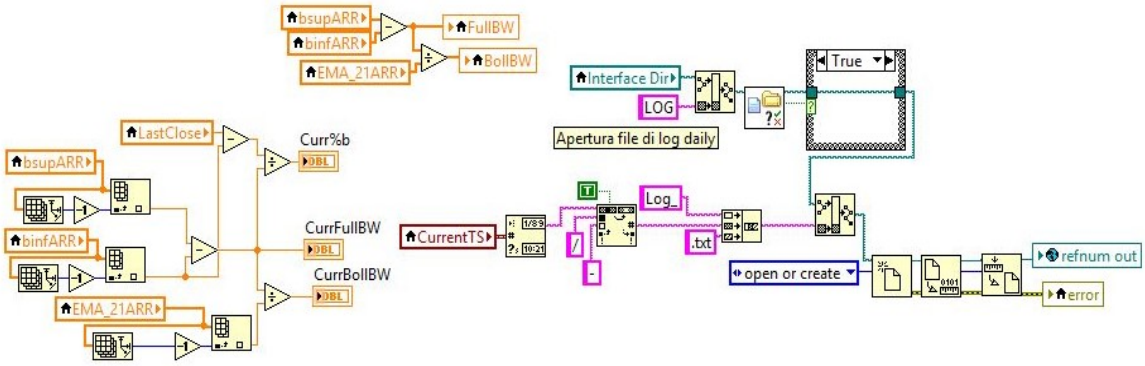


Figura 194 - Operazioni preliminari del codice relativo alla strategia di trading.

Le successive due aree di codice, contenute nei frame di indice uno e due, sviluppano il codice per l'apertura delle posizioni in corrispondenza delle due bande di Bollinger. Le due aree di codice sono speculari e differiscono unicamente nella banda di Bollinger considerata: la prima verifica lo sfondamento della banda superiore, la seconda verifica quello della banda inferiore. Per questo motivo, nel seguito descriveremo il codice relativo alla verifica sulla banda superiore indicando, in parentesi, le eventuali differenze presenti nel codice per la banda inferiore.

Il frame di indice uno si presenta con il codice per l'apertura della posizione contenuto in un costrutto condizionale le cui condizioni da verificare in AND sono (v. Figura 195):

- Che il valore di *Curr%b* sia maggiore (minore) del limite superiore delle bande di Bollinger impostato ad 1 (limite inferiore delle bande di Bollinger impostato a 0) incrementato (decrementato) di una quantità pari a *LimitTolerance*;
- Che la variabile booleana globale *BUY* abbia valore False, ovvero che non vi siano posizione Buy aperte al momento;
- Che la variabile booleana globale *SELL* abbia valore False, ovvero che non vi siano posizione Sell aperte al momento;
- Che la variabile booleana di controllo *CloseOps* abbia valore False, ovvero che non sia prevista la sospensione della strategia a breve;
- Che il valore corrente della larghezza di banda *CurrFullBW* sia superiore (strettamente maggiore) al valore soglia memorizzato nel parametro *BWT*. Tale controllo viene effettuato attraverso il sub vi *BWConstraint*.

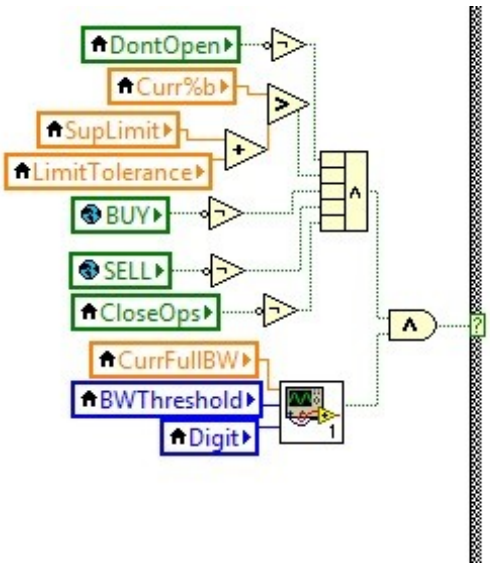


Figura 195 - Verifica dei vincoli di ingresso per la banda superiore di Bollinger.

Se tutte le suddette condizioni sono verificate, l'applicativo evolve con il codice per l'apertura di una posizione OCO.

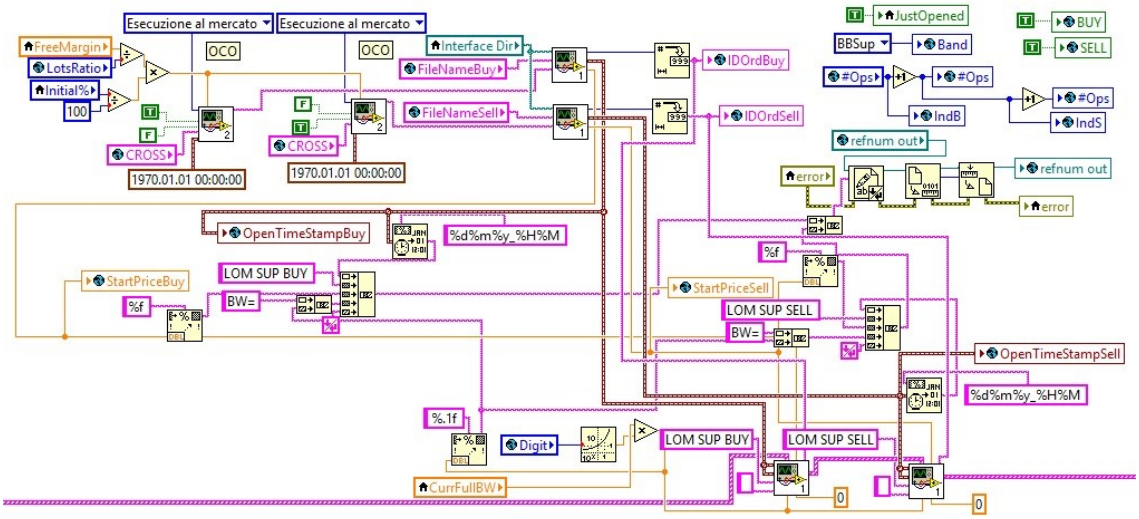


Figura 196 - Codice per le operazioni necessarie alla richiesta di apertura di una posizione OCO.

La prima operazione da effettuare è la costruzione della stringa da scrivere nel file di configurazione cui l'Expert Advisor periodicamente accede per verificare la presenza di richieste di apertura di ordini. Tale incombenza è deputata al vi *CreateNewOrder*, che prende in input:

- Una variabile di tipo ring indicante il tipo di operazione, potendo specificare due tipologie:
  - esecuzione al mercato;
  - ordine pendente;
- Una variabile di tipo ring indicante il tipo, potendo specificare due tipologie: Buy e Sell;
- Una variabile booleana indicante se si richieda di aprire una posizione di tipo Buy;
- Una variabile booleana indicante se si richieda di aprire una posizione di tipo sell;
- Una variabile di tipo stringa che indica lo strumento finanziario;
- Una variabile di tipo Timestamp che indica la scadenza della richiesta;
- Il numero di lotti richiesto;
- Il prezzo richiesto;
- L'eventuale stop loss;
- L'eventuale take profit;

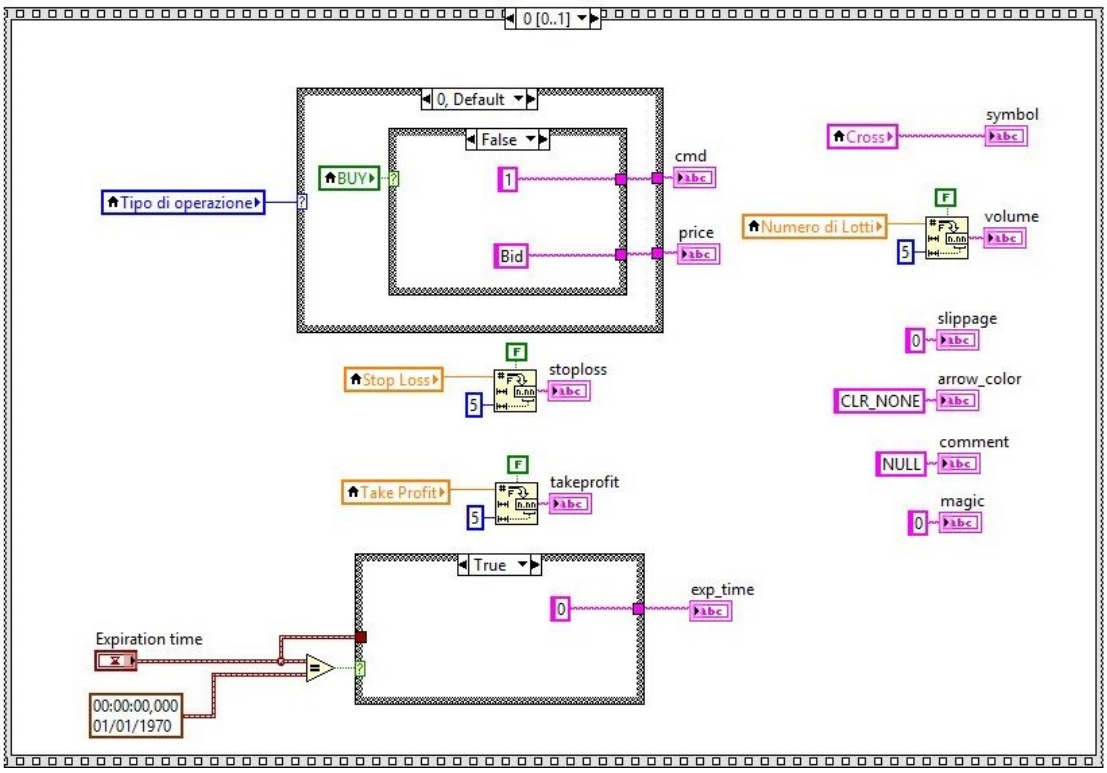


Figura 197 - Codice del vi deputato alla creazione di un nuovo ordine.

La funzione, a partire dagli input ricevuti, crea una stringa contenente i seguenti valori (v. Figura 198):

- *Symbol*, che coincide con il Cross;
- *Cmd*, un intero pari a 0 per la richiesta di una posizione Sell, 1 per la Buy;
- *Volume*, il numero di lotti richiesti;
- *Price*, la stringa *Bid* per la richiesta di una posizione Sell, la stringa *Ask* per la Buy;
- *Slippage*, impostato a zero;
- *Stoploss*, collegato all'input fornito;
- *TakeProfit*, collegato all'input fornito;
- *Comment*, impostato a NULL;
- *Magic*, impostato a zero;
- *Exp_time*, collegato all'input fornito;
- *Arrow_color*, impostato a CLR_NONE.

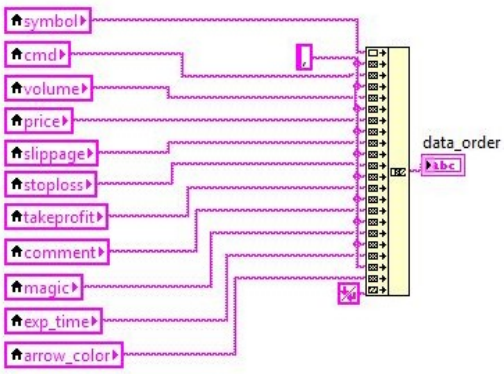


Figura 198 - Creazione della stinga di controllo per la richiesta alla piattaforma.

Il controllo passa quindi ad un altro vi, *OpStringWriterBuy*, che prende in ingresso la stringa precedentemente formata, il nome del file su cui scrivere tale stringa (contenuto nella variabile globale di tipo stringa *FileNameBuy*) e la variabile locale *InterfaceDir* coincidente con la cartella di lavoro. Se l'operazione di apertura posizione da parte dell'Expert Advisor in piattaforma si completa senza errori, il vi restituisce il numero di ticket valido associato all'ordine aperto, un Timestamp *OOpenTS* ed un double *OOpenPrice* che indicano rispettivamente il Timestamp ed il prezzo di apertura dell'ordine; in caso contrario il vi restituirà un valore di ticket pari a -1 ed un *OOpenTS* ed un *OOpenPrice* nulli (v. Figura 199).

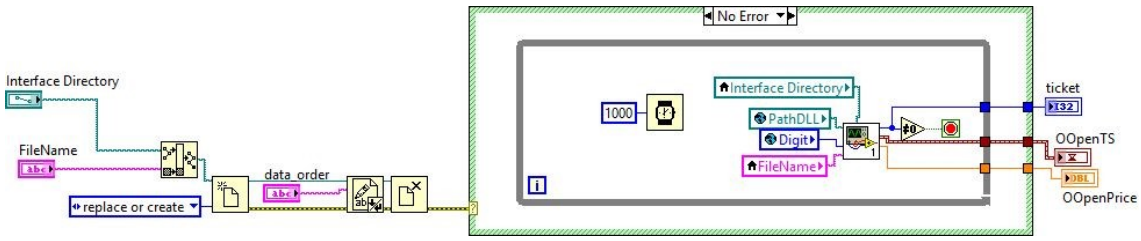


Figura 199 - Codice per l'ottenimento di un numero di ticket valido a partire dalla stringa di controllo.

Il vi *OpStringWriterBuy*, nello specifico, dapprima crea un file (se non esiste ancora) al percorso memorizzato in *InterfaceDir* con nome *FileNameBuy* e vi scrive dentro la stringa per la richiesta di apertura dell'ordine.

Come già discusso in precedenza, vi sono quattro differenti file di configurazione, uno per ciascuna tipologia di operazione (Buy e Sell), sia per l'apertura che per la chiusura di ordini. Tali file sono presenti solo se vi sono richieste pendenti, inserite nei file dall'applicativo; periodicamente, infatti, la piattaforma, attraverso l'Expert Advisor, controlla la presenza di tali file e, se verificata, accede ad essi, estrae la richiesta e, non appena l'ha assolta, cancella tali file.

Da un punto di vista dell'applicazione LabView, di conseguenza, affinché si possa considerare completata la richiesta occorre che il file sia stato cancellato ed il ticket valido associato all'ordine sia stato scritto in libreria DLL dall'Expert Advisor. Il vi *OpStringWriterBuy*, quindi, entra in un ciclo while infinito, temporizzato per essere eseguito ogni 1000 millisecondi (1 secondo), in cui controlla l'avvenuta apertura dell'ordine invocando il vi *GetOpTicketFromPlatformBuy*. Soltanto quando questo ritorna un ticket diverso da zero il ciclo termina ed il vi restituisce il ticket ed i corrispondenti Timestamp e prezzo di apertura.

Il vi *GetOpTicketFromPlatformBuy* riceve in ingresso il percorso alla libreria DLL contenuto nella variabile globale *PathDLL*, il percorso alla cartella di lavoro principale contenuto nella variabile locale *InterfaceDir*, la variabile globale *Digit* ed il nome del file da controllare. Il vi controlla quindi anzitutto

che il file di configurazione non esista, dal momento che ciò è condizione essenziale affinché si possa supporre che la richiesta in esso presente sia stata assolta. Se il file dovesse esistere, il vi ritorna un valore di ticket pari a zero e valori nulli per il *OOpenTS* e *OOpenPrice*; ciò permette al ciclo infinito presente nel vi *OpStringWriterBuy* precedentemente descritto, di continuare ad eseguire i cicli in attesa che il file di configurazione venga rimosso. Se invece il file non è presente, allora attraverso la funzione di libreria DLL *GetOrderInfoBuy* vengono letti dalla memoria i valori relativo al ticket, prezzo e Time-stamp di apertura dell'ordine eseguito e vengono restituiti come output (v. Figura 200).

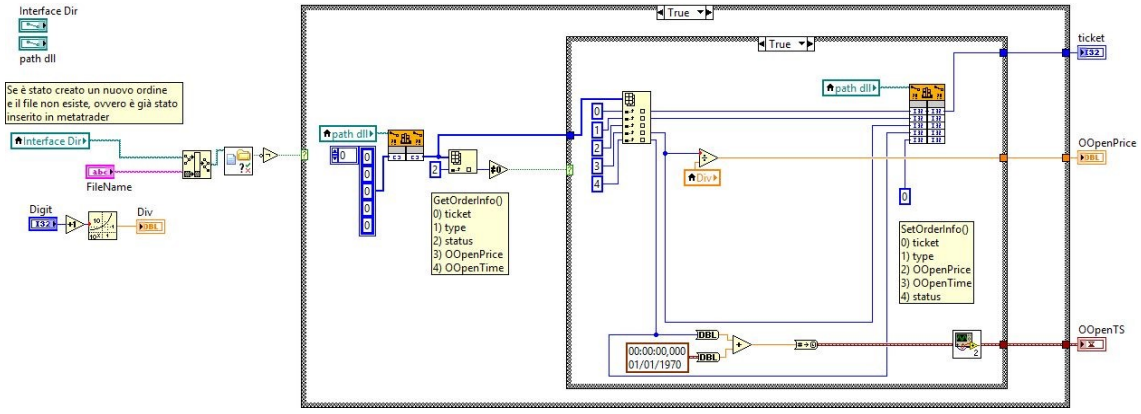


Figura 200 - Codice di attesa della rimozione del file contenente la stringa di controllo per l'apertura della posizione Buy.

Il codice appena descritto riguarda l'apertura della posizione in Buy; in maniera del tutto analoga, è presente anche il codice per l'apertura della posizione in sell, che differisce unicamente per il fatto di utilizzare vi speculari a quelli descritti ma finalizzati alla richiesta di apertura di una posizione di tipo sell.

Quindi, per le operazioni di Buy (v. Figura 201) e Sell (v. Figura 202), la catena di chiamate a funzione è la seguente:

Figura 201 - Catena di Chiamate - Operazione Buy

Figura 202 - Catena di Chiamate - Operazione Sell

Il ticket associato ai due ordini viene scritto nelle variabili globali di tipo stringa *IDOrdBuy* ed *IDOrd-Sell*, così come il Timestamp di apertura viene scritto nelle variabili globali di tipo stringa *OpenTime-StampBuy* ed *OpenTimeStampSell* ed il prezzo di apertura viene scritto nelle variabili globali di tipo stringa *StartPriceBuy* e *StartPriceSell*.

Ai fini di realizzare file di log completi sulle operazioni effettuate e di valutare l'operato della strategia anche al netto della piattaforma di trading, vengono gestite due differenti modalità di report: la prima, indipendente da MetaTrader4 e quindi idonea per valutare la strategia al netto di spread, delay e altre variabili platform-dependent, la seconda invece assolutamente allineata con i dati presenti in piattaforma ed i cui esiti vengono memorizzati nel vettore globale di cluster delle operazioni *OperationsARR*.

Ne consegue che sul file di report viene inserito un record per ciascuna operazione effettivamente aperta, costituito da:

- Timestamp di apertura ordine;
- Strategia (attualmente solo OCO);
- Banda di Bollinger considerata (SUP o INF);
- Tipo di operazione (BUY o SELL);
- Prezzo di apertura;
- Larghezza di banda all'apertura.

Analogamente, tutte queste informazioni vengono scritte nel cluster *OperationsARR* attraverso la chiamata al vi *InsertIntoOpCluster*, documentato nel paragrafo 7.9.

Per i valori relativi alla chiusura dell'operazione, il cui valore sarà aggiornato soltanto quando il relativo ordine sarà effettivamente chiuso, si è scelto di memorizzare inoltre i valori di:

- OpenTimeStampBuy (timestamp di apertura operazione);
- WhyClose (motivazione della chiusura dell'ordine)
- EndPrice (prezzo di chiusura)

Queste due operazioni, come era lecito attendersi, vengono eseguite sia per l'ordine relativo alla posizione in Buy che per quello relativo alla posizione in Sell, scrivendo in tal modo due differenti righe sul file di log, ed inserendo due diversi cluster nel vettore OperationsARR, che conterrà quindi un cluster per ciascuna operazione effettivamente aperta. Il vettore, così modificato, viene posto in input alla porzione di codice successiva. Se le condizioni di apertura non sono verificate, il vettore viene comunque posto in input all'area successiva senza avervi apportato alcuna modifica.

Inoltre vengono effettuate le seguenti operazioni:

- Attivazione della variabile booleana locale *JustOpened*, ad indicare che nella corrente iterazione della strategia è avvenuta l'apertura di una posizione OCO;
- Settaggio al valore *BBSup* della variabile di controllo globale Band, ad indicare che le operazioni attualmente aperte sono state eseguite in corrispondenza della banda superiore di Bollinger;
- Attivazione delle due variabili booleane globali *BUY* e *SELL*, ad indicare che attualmente sono aperte una operazione di Sell ed una di Buy;
- Incremento del contatore di operazioni effettuate globale *#Ops* e salvataggio dell'indice corrispondente nelle variabili globali *IndB* ed *IndS* (indicizzate a partire da zero ed utilizzate per l'accesso al corrispondente cluster nel vettore globale degli ordini);

Come anticipato, il frame di indice due contiene la logica, del tutto speculare a quella appena descritta, per la verifica delle condizioni sulla banda inferiore di Bollinger e successiva apertura di posizione OCO, con annessa memorizzazione su file e nell'array di cluster OperationsARR delle informazioni riguardanti la posizione (v. Figura 203).

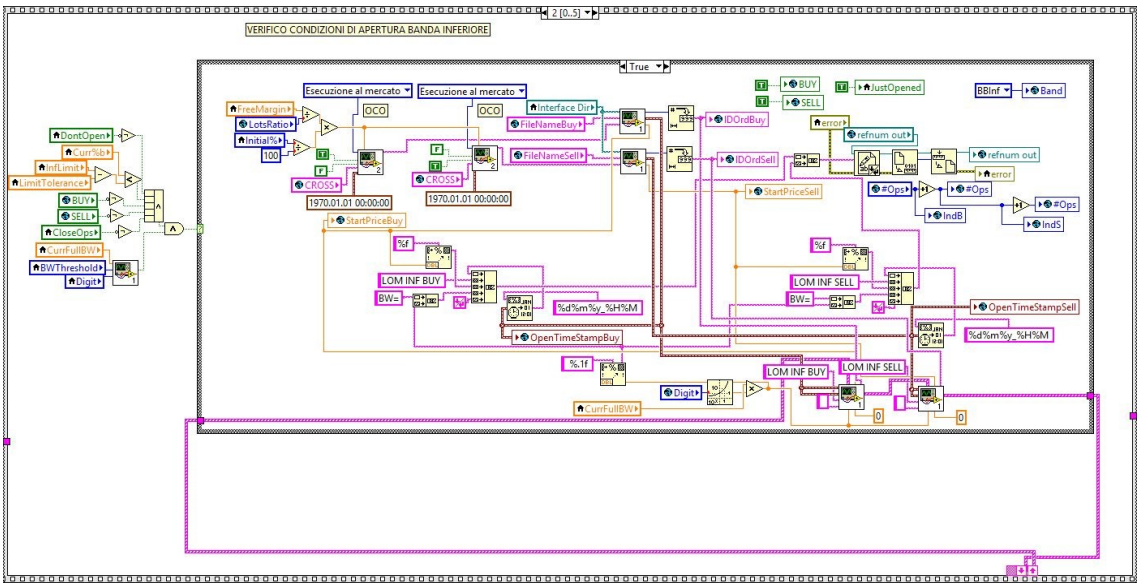


Figura 203 - Verifica delle condizioni di ingresso a mercato in corrispondenza di sfondamento della banda di Bollinger inferiore.

Il frame di indice tre contiene la verifica delle condizioni di uscita e, di conseguenza, di richiesta di chiusura di una specifica operazione. Le verifiche di chiusura operazione possono essere raggruppate come segue:

In perdita:

- Stop Loss per l'operazione di tipo Buy aperta in corrispondenza della banda superiore di Bollinger;
- Stop Loss per l'operazione di tipo Buy aperta in corrispondenza della banda inferiore di Bollinger;
- Stop Loss per l'operazione di tipo Sell aperta in corrispondenza della banda superiore di Bollinger;
- Stop Loss per l'operazione di tipo Sell aperta in corrispondenza della banda inferiore di Bollinger;

In profitto:

- Take Profit per l'operazione di tipo Buy aperta in corrispondenza della banda superiore di Bollinger;
- Take Profit per l'operazione di tipo Buy aperta in corrispondenza della banda inferiore di Bollinger;
- Take Profit per l'operazione di tipo Sell aperta in corrispondenza della banda superiore di Bollinger;
- Take Profit per l'operazione di tipo Sell aperta in corrispondenza della banda inferiore di Bollinger.

Nel seguito analizziamo le condizioni affinché sia richiesta la chiusura di un'operazione in ciascuno dei suddetti otto casi.

Viene richiesta la chiusura dell'operazione Buy corrente se (v. Figura 204):

- Il corrente valore del parametro *%b*, contenuto nella variabile locale *Curr%b*, risulta essere minore o uguale al valore del limite superiore delle bande di Bollinger, pari ad uno, decrementato di una quantità pari a *LimitTolerance*;
- La variabile booleana globale *BUY* ha valore True;
- La variabile di controllo globale *Band* è settata a *BBSup*;
- La variabile booleana locale *JustOpened* ha valore False.

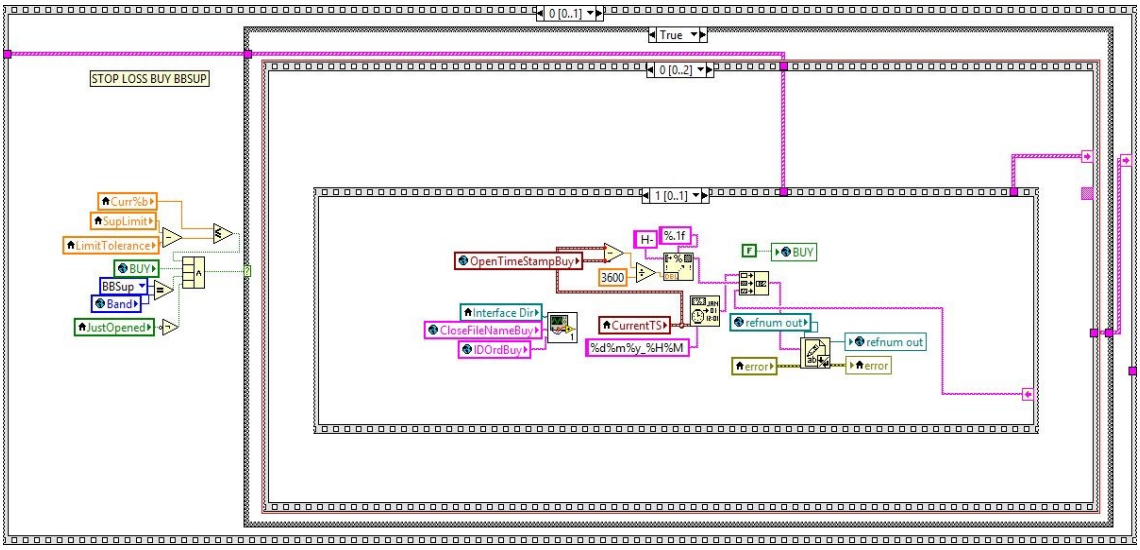


Figura 204 - Verifica delle condizioni di chiusura (in perdita) per la posizione long aperta per sfondamento della banda superiore di Bollinger.

Viene ugualmente richiesta la chiusura dell'operazione Buy corrente se (v. Figura 205):

- Il corrente valore del parametro *%b*, contenuto nella variabile locale *Curr%b*, risulta essere minore o uguale al valore del limite inferiore delle bande di Bollinger, pari a zero, decrementato di una quantità pari a *LimitTolerance*;
- La variabile booleana globale *BUY* ha valore True;
- La variabile di controllo globale *Band* è settata a *BBInf*;
- La variabile booleana locale *JustOpened* ha valore False.

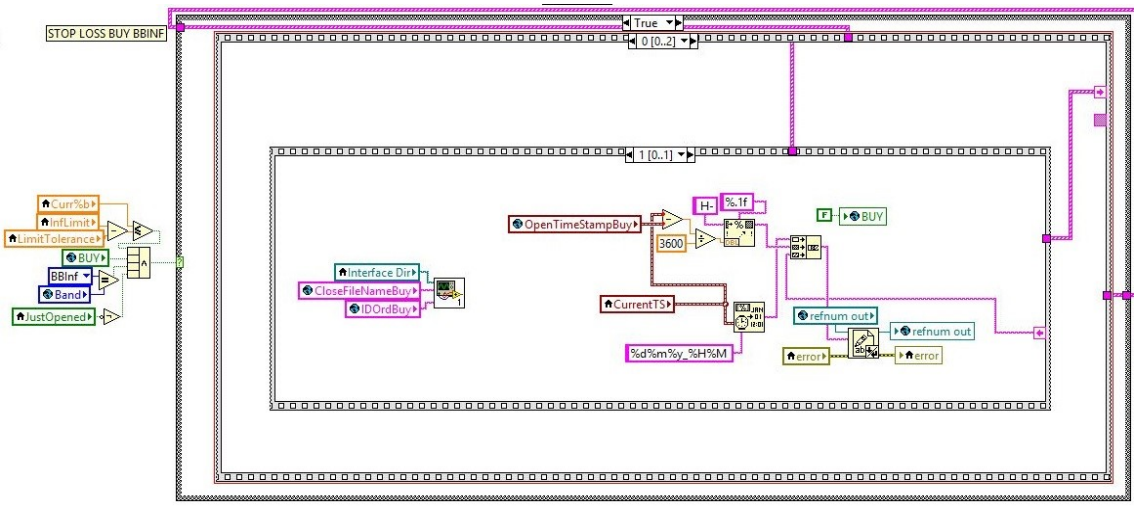


Figura 205 - Verifica delle condizioni di chiusura (in perdita) per la posizione long aperta per sfondamento della banda inferiore di Bollinger.

Per l'operazione Sell corrente, ne viene ordinata la chiusura se (v. Figura 206):

- Il corrente valore del parametro *%b*, contenuto nella variabile locale *Curr%b*, risulta essere maggiore o uguale al valore del limite superiore delle bande di Bollinger, pari ad uno, incrementato di una quantità pari a *LimitTolerance*;
- La variabile booleana globale *SELL* ha valore True;
- La variabile di controllo globale *Band* è settata a *BBSup*;
- La variabile booleana locale *JustOpened* ha valore False.

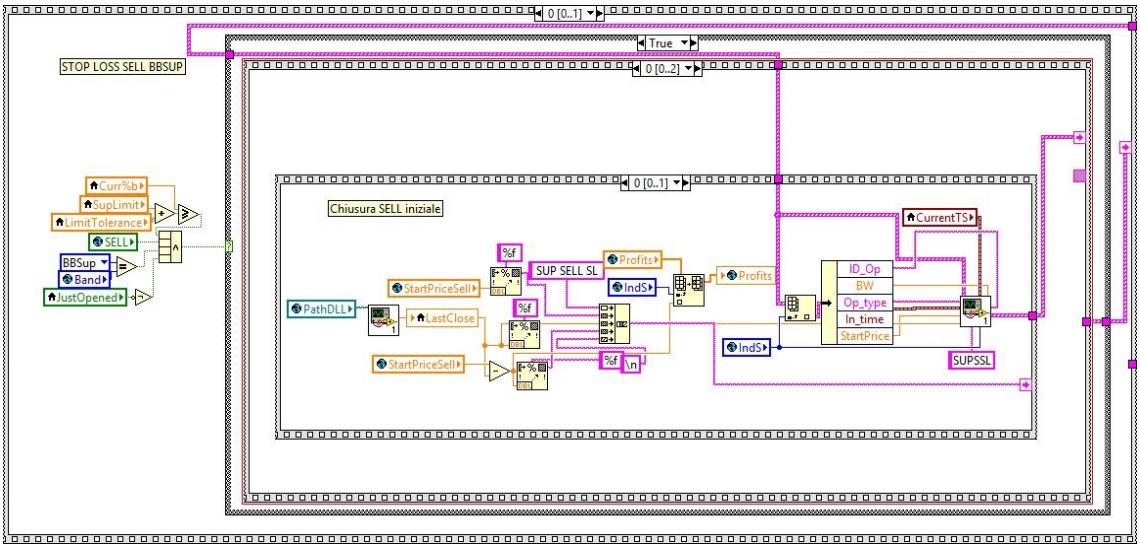


Figura 206 - Verifica delle condizioni di chiusura (in perdita) per la posizione short aperta per sfondamento della banda superiore di Bollinger.

Viene parimenti richiesta alla piattaforma la chiusura dell'operazione Sell corrente se (v. Figura 207):

- Il corrente valore del parametro *%b*, contenuto nella variabile locale *Curr%b*, risulta essere maggiore o uguale al valore del limite inferiore delle bande di Bollinger, pari a zero, incrementato di una quantità pari a *LimitTolerance*;

- La variabile booleana globale *SELL* ha valore True;
- La variabile di controllo globale *Band* è settata a *BBInf*;
- La variabile booleana locale *JustOpened* ha valore False.

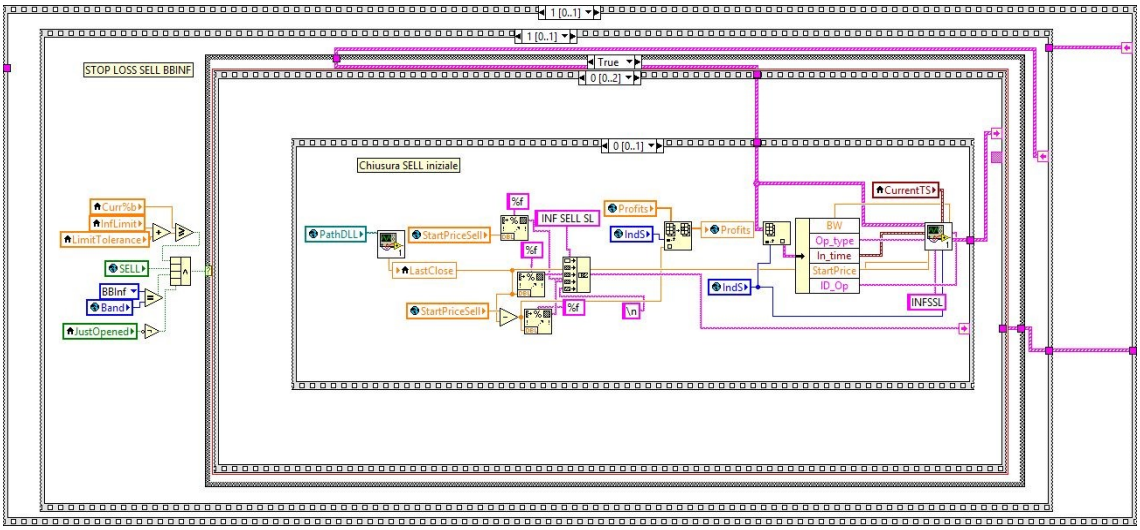


Figura 207 - Verifica delle condizioni di chiusura (in perdita) per la posizione short aperta per sfondamento della banda inferiore di Bollinger.

L'operazione Buy corrente viene chiusa (Take Profit) se (v. Figura 208):

- La larghezza di banda di Bollinger nelle ultime tre candele sta diminuendo, ovvero il valore contenuto nel vettore *BollBW* relativo alla larghezza di banda di Bollinger riferita alle ultime tre candele è decrescente. Ciò implica che il valore contenuto nel vettore *BollBW* alla posizione *n-3* debba essere maggiore o uguale al valore contenuto alla posizione *n-2* che, a sua volta, deve essere maggiore o uguale al valore contenuto alla posizione *n -1*;
- La media scelta *ChoosedEMA* si sta orientando verso il basso, ovvero il valore contenuto nel vettore *ChoosedEMA* relativo alla media scelta riferita alle ultime tre candele è decrescente. Ciò implica che il valore contenuto nel vettore ChoosedEMA alla posizione *n-3* debba essere maggiore o uguale al valore contenuto alla posizione *n-2* che, a sua volta, deve essere maggiore o uguale al valore contenuto alla posizione *n-1*;
- La variabile booleana globale *BUY* ha valore True;
- La variabile di controllo globale *Band* è settata a *BBSup*;
- La variabile booleana locale *JustOpened* ha valore False.

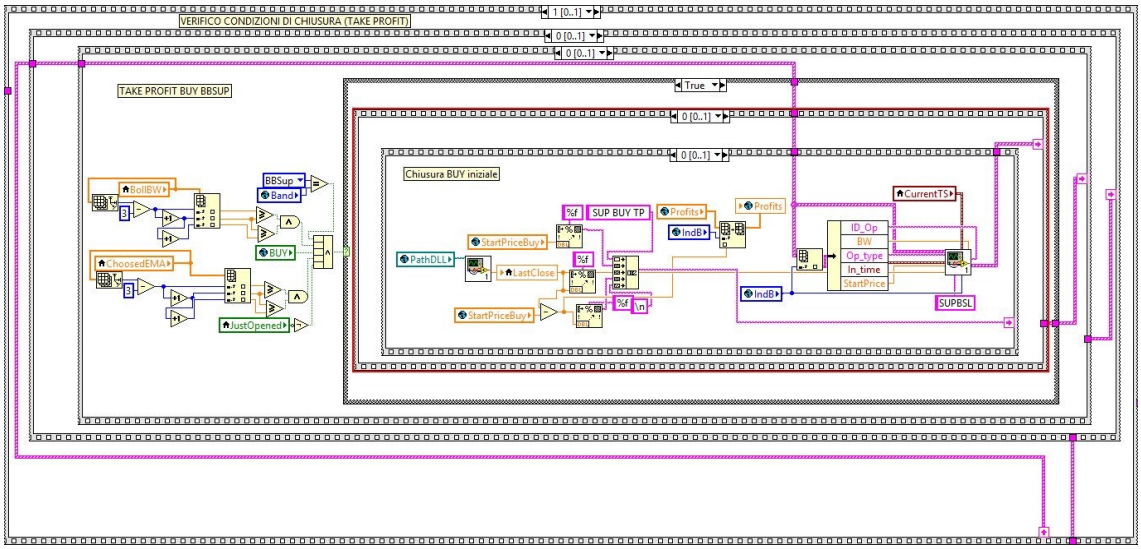


Figura 208 - Verifica delle condizioni di chiusura (Take Profit) per la posizione long precedentemente aperta per sfondamento della banda superiore di Bollinger.

Viene ugualmente chiusa l'operazione Buy corrente se risulta verificato uno qualsiasi (quindi in OR) dei seguenti due insieme di condizioni (v. Figura 209):

- Siano verificate tutte le seguenti condizioni:
  - La larghezza di banda di Bollinger nelle ultime tre candele sta diminuendo, ovvero il valore contenuto nel vettore BollBW relativo alla larghezza di banda di Bollinger riferita alle ultime tre candele è decrescente;
  - La media scelta ChoosedEMA si sta orientando verso il basso, ovvero il valore contenuto nel vettore *ChoosedEMA* relativo alla media scelta riferita alle ultime tre candele è decrescente;
  - La variabile booleana globale *BUY* ha valore True;
  - La variabile di controllo globale *Band* è settata a *BBInf*;
  - La variabile booleana locale *JustOpened* ha valore False.
- In alternativa, siano verificate tutte le seguenti condizioni:
  - Il corrente valore del parametro *%b*, contenuto nella variabile locale *Curr%b*, risulta essere maggiore o uguale al valore del limite superiore delle bande di Bollinger, pari ad uno;
  - La variabile di controllo globale *Band* è settata a *BBInf*;
  - La variabile booleana globale *BUY* ha valore True;
  - La variabile booleana locale *JustOpened* ha valore False.

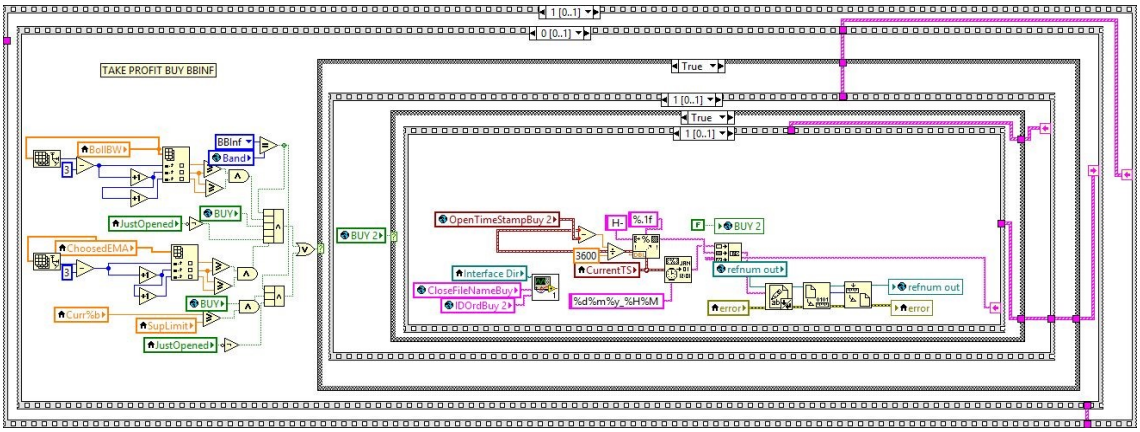


Figura 209 - Verifica delle condizioni di chiusura (Take Profit) per la posizione long precedentemente aperta per sfondamento della banda inferiore di Bollinger.

Viene richiesta la chiusura dell'operazione sell corrente se risulta verificato uno qualsiasi (quindi in OR) dei seguenti due insieme di condizioni (v. Figura 210):

- Siano verificate tutte le seguenti condizioni:
  - La larghezza di banda di Bollinger nelle ultime tre candele sta diminuendo, ovvero il valore contenuto nel vettore BollBW relativo alla larghezza di banda di Bollinger riferita alle ultime tre candele è decrescente;
  - La media scelta *ChoosedEMA* si sta orientando verso l'alto, ovvero il valore contenuto nel vettore *ChoosedEMA* relativo alla media scelta riferita alle ultime tre candele è crescente;
  - La variabile booleana globale *SELL* ha valore True;
  - La variabile di controllo globale *Band* è settata a *BBSup*;
  - La variabile booleana locale *JustOpened* ha valore False.
- In alternativa, siano verificate tutte le seguenti condizioni:
  - Il corrente valore del parametro *%b*, contenuto nella variabile locale *Curr%b*, risulta essere minore o uguale al valore del limite inferiore delle bande di Bollinger, pari a zero;
  - La variabile di controllo globale *Band* è settata a *BBSup*;
  - La variabile booleana globale *SELL* ha valore True;
  - La variabile booleana locale *JustOpened* ha valore False.

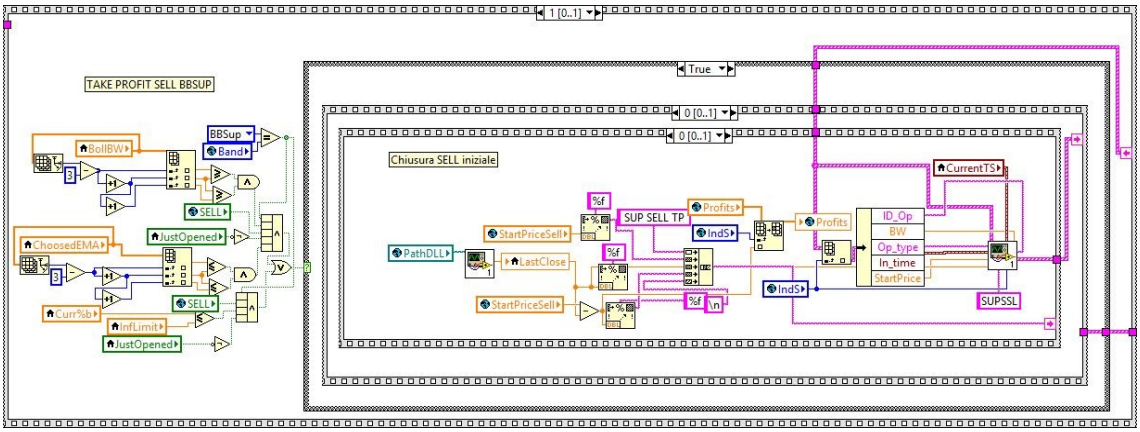


Figura 210 - Verifica delle condizioni di chiusura (Take Profit) per la posizione short precedentemente aperta per sfondamento della banda superiore di Bollinger.

Allo stesso modo, viene richiesta la chiusura dell'operazione sell corrente se (v. Figura 211):

- La larghezza di banda di Bollinger nelle ultime tre candele sta diminuendo, ovvero il valore contenuto nel vettore *BollBW* relativo alla larghezza di banda di Bollinger riferita alle ultime tre candele è decrescente;
- La media scelta *ChoosedEMA* si sta orientando verso l'alto, ovvero il valore contenuto nel vettore *ChoosedEMA* relativo alla media scelta riferita alle ultime tre candele è crescente;
- La variabile booleana globale *SELL* ha valore True;
- La variabile di controllo globale *Band* è settata a *BBInf*;
- La variabile booleana locale *JustOpened* ha valore False.

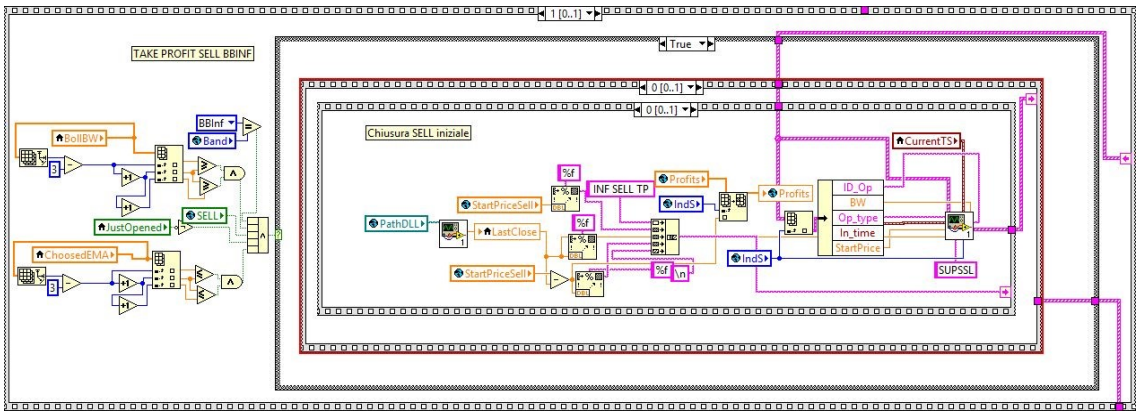


Figura 211 - Verifica delle condizioni di chiusura (Take Profit) per la posizione short precedentemente aperta per sfondamento della banda inferiore di Bollinger.

In ognuno dei succitati casi si configurano le condizioni per cui la strategia richiede la chiusura dell'ordine in esame. Il codice per l'effettiva chiusura di una operazione è comune ad ognuna delle otto differenti aree di codice e differisce unicamente tra operazioni di Buy e di Sell nei parametri utilizzati, che sono relativi alla tipologia specifica di operazione di cui si richiede la chiusura.

Al fine di ottenere il prezzo aggiornato in relazione del quale richiedere la chiusura di un ordine, viene invocato il vi *ReadCalLoMiTTraS* per ottenere il valore aggiornato *LastClose*. Per la gestione del report su file di log viene preparata la stringa descrittiva dell'operazione motivata di aperture e chiusure, la quale viene quindi inserita (in append) sul file di log giornaliero di strategia. Il profitto dell'operazione, inoltre, viene scritto nel vettore globale dei profitti *Profits* nella cella di indice *IndS* o *IndB*, a seconda dell'operazione che si sta chiudendo.

Per la gestione della memorizzazione degli ordini su struttura dati, i nuovi valori ottenuti dalla chiusura dell'operazione vengono dati in ingresso al vi di uso comune *ReplaceIntoOpCluster* per aggiornare il cluster di operazione presente alla cella di indice *IndS* o *IndB* (in base all'operazione in esame) nel vettore globale di cluster delle operazioni *OperationsARR*.

L'operazione indispensabile, affinché l'ordine venga effettivamente chiuso in piattaforma, consiste nello scrivere la relativa stringa di controllo sul file di configurazione (*CloseOrderBuy* o *CloseOrderSell*, a seconda del caso); ciò avviene attraverso il vi *OrderToClose*, che riceve in ingresso la cartella di lavoro *InterfaceDir*, il nome del file di configurazione (*CloseFileNameSell* o *CloseFileNameBuy*) e il ticket dell'ordine da chiudere *IDOrderBuy* o *IDOrderSell*.

Il vi *OrderToClose* non restituisce alcun output, bensì si occupa unicamente di scrivere l'ordine da chiudere sul file di configurazione. Come accadeva per l'apertura degli ordini, la piattaforma, attraverso l'EA, periodicamente controlla la presenza di file di configurazione dedicato alla chiusura ordini e, quando presente, ne estrae il contenuto ed esegue l'operazione di chiusura (v. Figura 212).

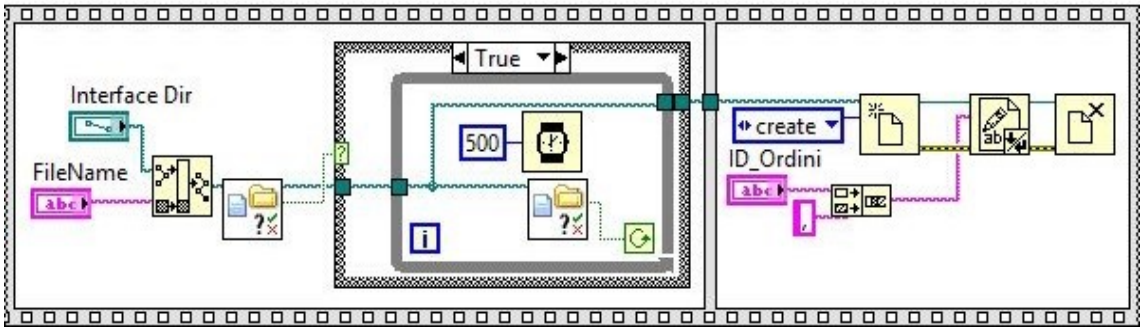


Figura 212 - Codice per la richiesta di chiusura ordine a mercato alla piattaforma.

Ulteriori operazioni effettuate nella porzione di codice deputata alla chiusura degli ordini sono la disattivazione della variabile globale booleana, *BUY* o *SELL* a seconda del caso, per indicare che non vi è più un'operazione di tale tipologia a mercato. Inoltre, nel caso in cui non vi siano più operazioni a mercato attualmente, il valore della variabile globale di controllo *Band* viene di nuovo settato a *NoBB*.

Nel frame di indice quattro vi è il codice per la chiusura forzata delle operazioni (v. Figura 213). Nel caso in cui la variabile booleana *CloseOps* abbia valore True, infatti, viene eseguita, sia per l'operazione di Sell che per quella di Buy, la porzione di codice per la chiusura dell'operazione, indicando come motivazione la chiusura forzata (***END***, ***B*** o ***S*** per Buy o Sell ed ***F*** per Forced)

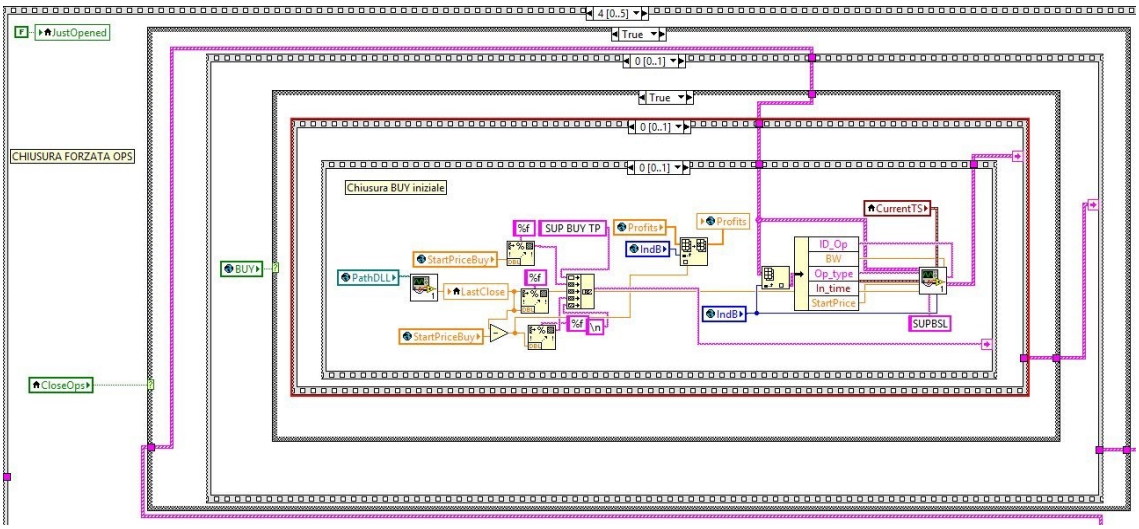


Figura 213 - Chiusura forzata delle operazioni attualmente a mercato.

Infine, nel sesto ed ultimo frame è implementata la logica per la scrittura su file di log giornaliero del profitto complessivo associato alla corrente iterazione della strategia. A tal fine, bisogna verificare che sussistano due condizioni: che la variabile booleana *CloseOps* abbia valore True e che il valore della variabile globale *FinalProfit* sia pari a meno infinito, che rappresenta il valore di inizializzazione di tale variabile. Tale controllo si rende necessario in quanto l'operazione di scrittura del profitto complessivo della strategia nella variabile va effettuata una sola volta e non va ripetuta nelle successive iterazioni.

Nel caso in cui sussistano le condizioni, quindi, di imminente sospensione dell'esecuzione della strategia, viene effettuata la scrittura su file di log giornaliero del profitto complessivo (contenuto nella variabile globale double *FinalProfit*), ottenuto come sommatoria dei valori contenuti nel vettore globale dei profitti *Profits* (v. Figura 214).

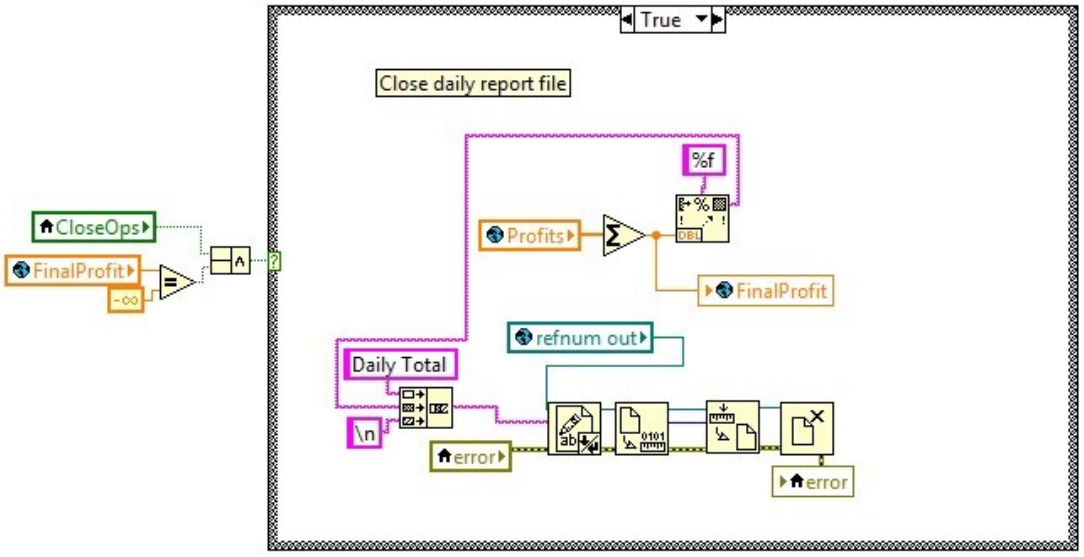


Figura 214 - Chiusura del file giornaliero di report della strategia.

7.11.2 Engine per la creazione del file di definizione

Il vi *AI_Engine_LoMiTTraS*, nella sua versione attuale, è utile alla creazione di un file di configurazione per elencare le possibili configurazioni parametriche per la strategia OCO e, qualora si rendesse necessario in futuro, per ulteriori strategie.

Come già analizzato in precedenza, esso può essere utilizzato in modalità di inizializzazione, con valore della variabile booleana di controllo *Update* impostata al valore False, per configurare la prima esecuzione dei test e la necessità, quindi, di testare ciascun parametro sul suo insieme di definizione completo, oppure in modalità di update, con la variabile Update impostata a True, per configurare le iterazioni di test successive alla prima, in cui i parametri *BandWidth Threshold* (BWT) e *Limit Tolerance* (LT) vengono testati su un intervallo di variazione costituito da cinque valori, e *ChooseEMA* su tre possibili valori, dando così luogo ad un numero complessivo pari a 75 configurazioni parametriche.

Il vi riceve in ingresso i seguenti parametri (v. Figura 215):

- Il percorso in cui salvare il file di definizione *.def*;
- La tipologia di strategia, attualmente sempre la *OCO*;
- Una variabile booleana di controllo per attivare o meno la modalità di inizializzazione;
- Un double corrispondente al parametro *Limit Tolerance*;
- Un intero corrispondente al parametro *BandWidth Threshold*;
- Il percorso al file da cui leggere i valori per inizializzare il parametro *BWT*, contenuto in *BWTDefFile*.

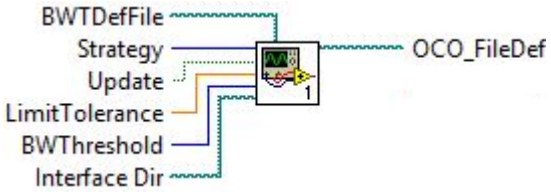


Figura 215 - Input/Output del vi per la creazione del file contenente le configurazioni parametriche da sottoporre a test per finalità di auto apprendimento.

Una volta creato il file di definizione, ne viene restituito in uscita il percorso ove reperire il file stesso.

Il codice del vi inizia con la chiamata al sub vi *BWTValuesReader*, atto a leggere l'insieme di definizione per il parametro *BWT* dal file *BWTLearningInitValues*.lit, dipendente dal timeframe e dallo strumento finanziario.

Il sub vi *BWTValuesReader* riceve in ingresso il percoros al file *.lit*, il CROSS ed il timeframe (v. Figura 216).

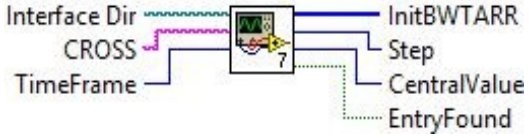


Figura 216 - Input/Output del vi per ottenere l'insieme di definizione del parametro BWT.

Il primo controllo consiste nella verifica di esistenza del file di definizione: qualora esso non sia presente, infatti, viene associato un insieme di definizione di default in relazione al timeframe in input.

A livello concettuale si è scelto di prevedere due diversi insiemi di default in assenza del file *.lit*:

1. Per timeframe, espresso in secondi, minori o uguali a 300, e quindi per i timeframe 1 e 5 minuti, l'insieme di definizione è composto dal vettore D={10, 20, 30, 40, 50, 60, 70, 80}, con valore centrale 30 (l'elemento di indice 2) e passo 10;
2. Per timeframe maggiori di 300, e quindi per i timeframe da 15 minuti in poi, l'insieme di definizione è composto dal vettore D={10, 40, 70, 100, 130, 160, 190, 220}, con valore centrale 70 (l'elemento di indice 2) e passo 30.

Al contrario, se il file *.lit* esiste, viene analizzato ogni record in esso presente alla ricerca del dataset riferito al CROSS e Timeframe ricevuto in input. Una volta individuato il record corrispondente ai requisiti, ciascun valore estratto viene convertito e memorizzato in un vettore, infine restituito in output come insieme di definizione del parametro *BWT* insieme con il suo valore centrale ed il passo (differenza tra ciascun elemento del vettore ed il suo successivo/precedente). Se invece non vi è corrispondenza nel file *.lit* con lo strumento finanziario/timeframe ricevuti in ingresso, viene restituito un insieme di definizione in relazione al timeframe in input, con gli stessi criteri utilizzati per la gestione del caso di inesistenza del file *.lit* (v. Figura 217).

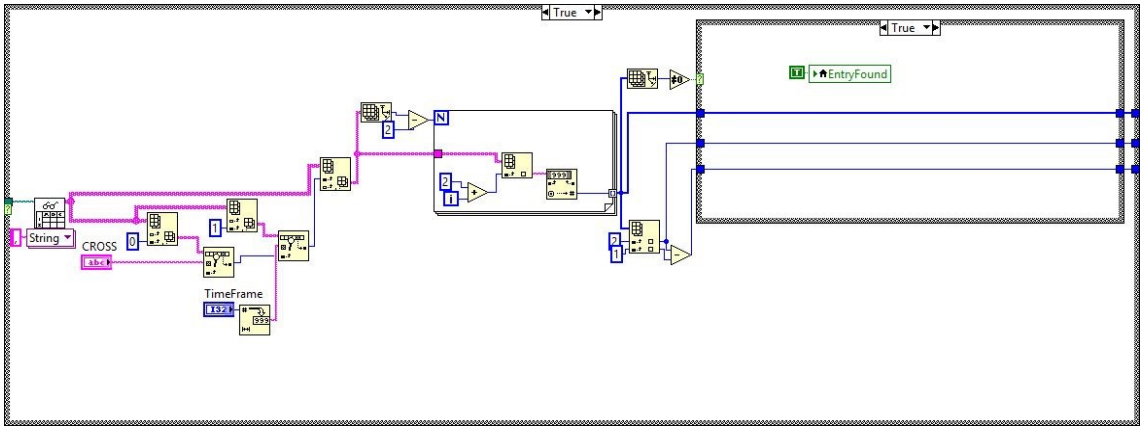


Figura 217 - Codice per la gestione dell'insieme di definizione del parametro BWT in presenza del file di configurazione .lit.

Una volta ottenuti quindi l'insieme di definizione (il vettore dei valori che il parametro può assumere), il valore centrale ed il passo per il parametro *BWT* vi è il controllo sulla modalità di creazione del file *.def*:

1. Se la variabile booleana *Update* ha valore False, siamo nel caso della prima iterazione di test e, di conseguenza, i parametri *LT* e *BWT* vanno testati sui loro interi insiemi di definizione. Per *Limit Threshold* si è scelto di testare i valori compresi tra 0.05 e 0.25 con passo 0.025, ovvero nove step di variazione del parametro (v Figura 218).

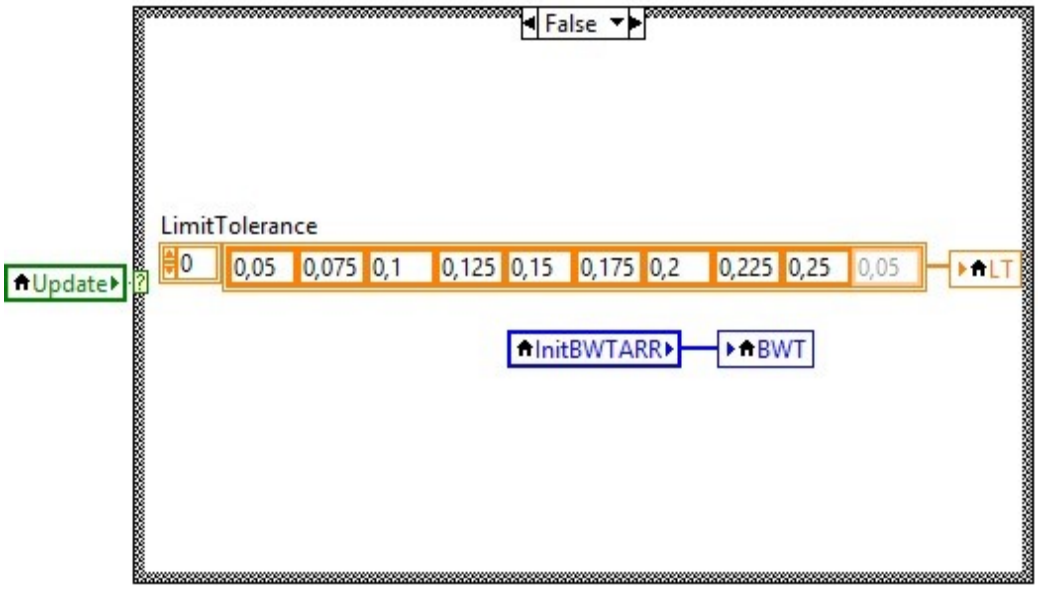


Figura 218 - Assegnazione dell'insieme di definizione ai parametri BWT ed LT in caso di modalità di inizializzazione (prima run di esecuzione della fase di Self Learning).

2. Se la variabile booleana *Update* ha valore True, non siamo nel caso della prima iterazione di test e, di conseguenza, *BWT* e *LT* vanno testati su un insieme di cinque valori centrato sui valori ricevuti di ciascun parametro in ingresso. Al fine di evitare che il parametro *LT* possa assumere valori non appartenenti al suo insieme di definizione, il valore ricevuto in ingresso viene controllato e, se necessario, riportato a valori consentiti, sia per valori minori del minimo consentito, sia per i valori superiori al massimo consentito. Al contrario, per il parametro *BWT* viene controllato unicamente che esso non assuma valori minori del minimo valore consentito, mentre è lasciato crescere potenzialmente ad infinito per valori maggiori. Tale scelta è dovuta alla centralità del parametro nella strategia *LoMiTTraS* e alla considerazione che, proprio in relazione al Self Learning, è necessario che il parametro *BWT* possa assumere valori anche molto maggiori per intercettare situazioni in cui la volatilità del mercato sia particolarmente elevata. Di contro, si ritiene che nel caso in cui il mercato poi ritorni ad assumere valori di larghezza di banda nella media, il modulo di auto apprendimento, grazie alla sua struttura, sia in grado di riconoscere tale situazione impiegando appena due iterazioni con numero di operazioni pari a zero per inizializzare nuovamente il parametro *BWT* con il suo insieme di definizione (v. Figura 219).

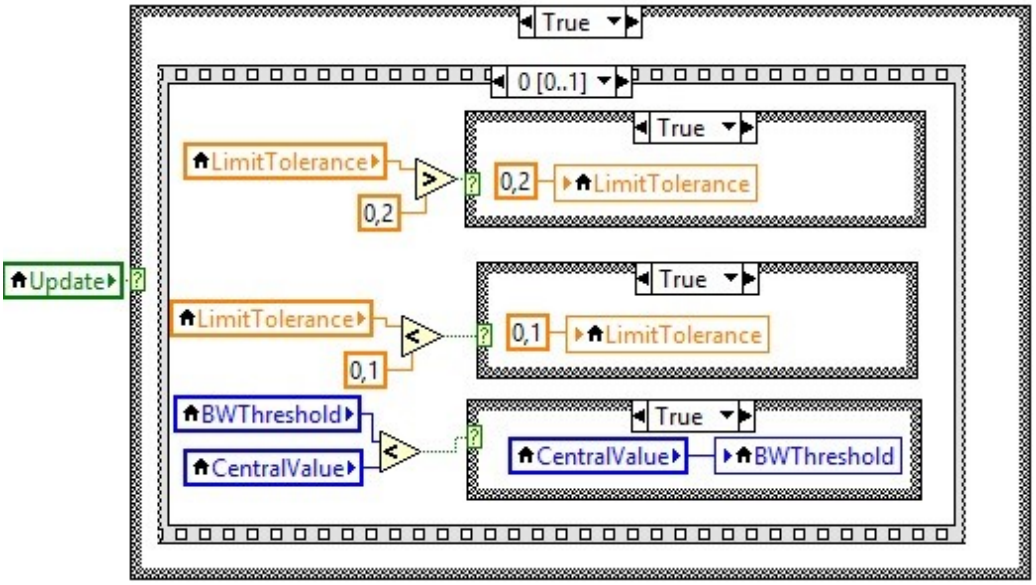


Figura 219 - Assegnazione dell'insieme di definizione ai parametri BWT ed LT in caso di modalità di aggiornamento (run successive alla prima della fase di Self Learning).

Una volta stabilito l'intervallo di variazione per i test dei parametri *BWT* e *LT*, il controllo passa alla porzione di codice atta a creare il file di definizione.

Per mezzo di una serie di cicli for annidati tutte le possibili configurazioni parametriche vengono generate sotto forma di stringa, con valori separati da virgola, e memorizzate sul file di definizione indicato dalla variabile d'uscita *OCO_FileDef* (v. Figura 220).

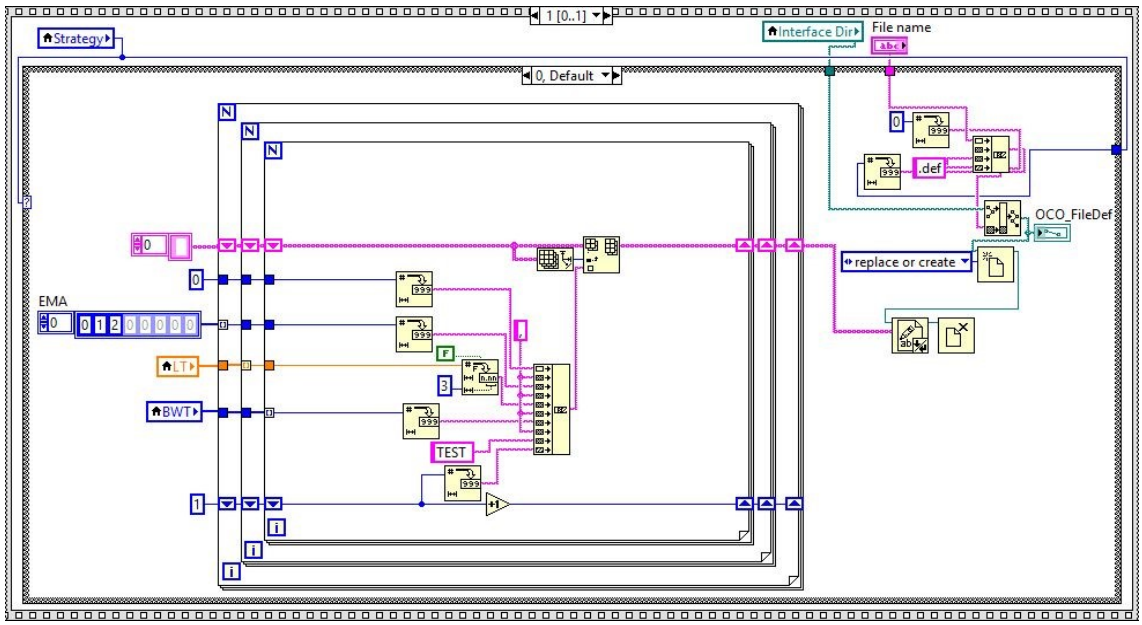


Figura 220 - Scrittura della stringa contenente le configurazioni parametriche da testare.

7.11.3 Gestione del parametro BandWidth Threshold

Ad integrazione di quanto già documentato, per una completa gestione dei valori del parametro di soglia sulla larghezza di banda, che ricopre un ruolo centrale all'interno della strategia *LoMiTTraS*, occorre documentare altri due vi che vengono impiegati durante l'esecuzione del modulo di auto apprendimento.

Nel caso, infatti, che alla corrente iterazione del Self Learning non sia stata effettuata alcuna operazione, si generano due uteriori possibilità:

1. L'algoritmo di Self Learning si trova alla terza iterazione consecutiva con zero operazioni: si rende necessario modificare l'insieme di definizione del parametro tramite lettura del file *.lit* da parte del vi *BWTValuesReader*. Infine il vi AI_Engine_LoMiTTraS si occuperà della creazione del file di definizione per la prossima iterazione in modalità non *Update*, ovvero inizializzazione con test dei parametri sui loro insiemi di definizione completi.
2. Nelle due precedenti iterazioni era stata effettuata almeno una operazione: in tal caso occorre selezionare per *BWT* il valore modale, ovvero quel valore cui corrispondono il maggior numero di iterazioni con massimo profitto.

La definizione del valore modale per il parametro avviene per mezzo del vi *GetBWTModalValue*, il quale riceve in ingresso il vettore contenente i valori che il parametro *BWT* ha assunto in corrispondenza delle iterazioni che hanno condotto al profitto massimo (*BWTArray*), ed il percorso alla cartella principale di progetto *InterfaceDir* per reperire il file *.lit* contenente le informazioni di base sul parametro *BWT*.

Le prime operazioni riguardano l'estrazione dell'insieme di definizione, il valore centrale ed il passo per il parametro BWT in relazione al corrente strumento finanziario ed al timeframe, tramite il vi *BWTValuesReader*. Inoltre, la variabile booleana locale *EntryFound* (letteralmente, entry trovata) viene attivata se tali valori sono stati trovati in corrispondenza di una precisa entry presente nel file *.lit*. (v Figura 221).

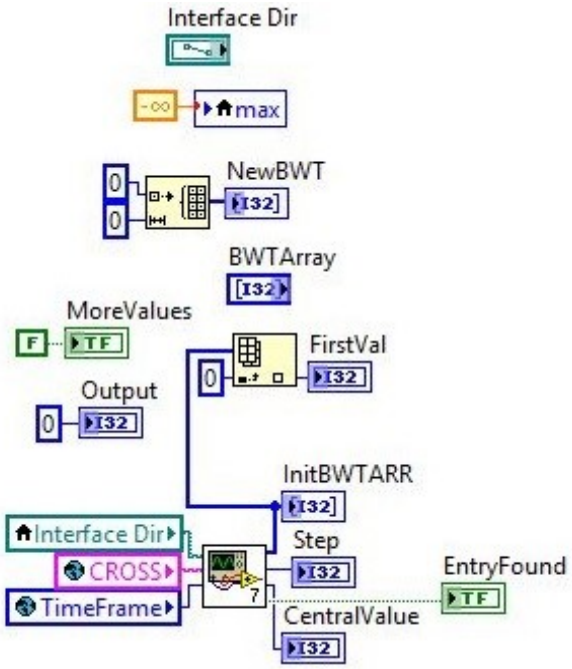


Figura 221 - Settaggio dell'insieme di definizione del parametro BWT.

Se la variabile *EntryFound* ha valore False, allora il valore di *BWT* viene determinato tramite il vi *BWTUpdater*, piuttosto che selezionare il suo valore centrale. Non conoscendo infatti le informazioni di base su *BWT*, si è scelto invece di passare il vettore ricevuto in ingresso *BWTArray* e *l'InterfaceDir* al vi *BWTUpdater* che selezionerà il valore medio tra quelli presenti nel vettore d'ingresso, piuttosto che assegnare un valore che potrebbe non appartenere al range di valori consentiti (v. Figura 222).

Figura 222 - Scelta del valore di maggiore frequenza del parametro BWT in assenza di informazioni di base sullo stesso.

Se, al contrario, la variabile *EntryFound* ha valore True, allora si conosce l'insieme di definizione di *BWT* ed il suo passo di variazione, ed è quindi possibile scorrere sequenzialmente l'intero vettore in ingresso *BWTArray* alla ricerca del valore con maggiore frequenza.

Se la dimensione del vettore BWTArray è nulla, il valore in uscita viene impostato al valore centrale di BWT precedentemente letto da file. Se, al contrario, il vettore BWTArray contiene effettivamente degli elementi occorre scorrerli e selezionare quello con frequenza maggiore, tenendo presente che essi sono ordinati in senso crescente.

Il controllo è effettuato all'interno di un ciclo while; per ciascuna iterazione, una variabile intera *First-Val*, inizializzata al primo elemento dell'insieme di definizione del parametro viene incrementata di una quantità pari al passo. Il ciclo termina quando *FirstVal* diventa maggiore dell'ultimo elemento presente in *BWTArray*.

I controlli effettuati all'interno del ciclo while consistono in:

- Ricerca del massimo, con due possibili casi:
    1. Il massimo corrente coincide con quello individuato alla precedente iterazione; in tal caso siamo in presenza di due (o più) valori con massima frequenza. Si attiva quindi la variabile booleana *MoreValues*, si aggiorna il valore della variabile *FirstVal* incrementandolo di un'entità pari al passo e si aggiunge il valore appena individuato (antecedente l'incremento) all'interno del vettore *NewBWT*.
    2. Il massimo corrente è maggiore del precedente individuato, ovvero il numero di occorrenze del valore corrente di *FirstVal* è strettamente maggiore del valore precedente relativo al numero di occorrenze massimo finora osservato. In tal caso la variabile *More-Values* viene disattivata, il vettore *NewBWT* inizializzato il solo elemento *FirstVal*. Infine, il valore di FirstVal viene aggiornato incrementandolo di una entità pari al passo.
- Alla fine di ciascuna iterazione del while, si verifica la condizione di terminazione del ciclo.

Il termine della computazione avviene quando *FirstVal* assume un valore maggiore del valore contenuto nell'ultima cella del vettore *BWTArray* (v. *Figura 223*).

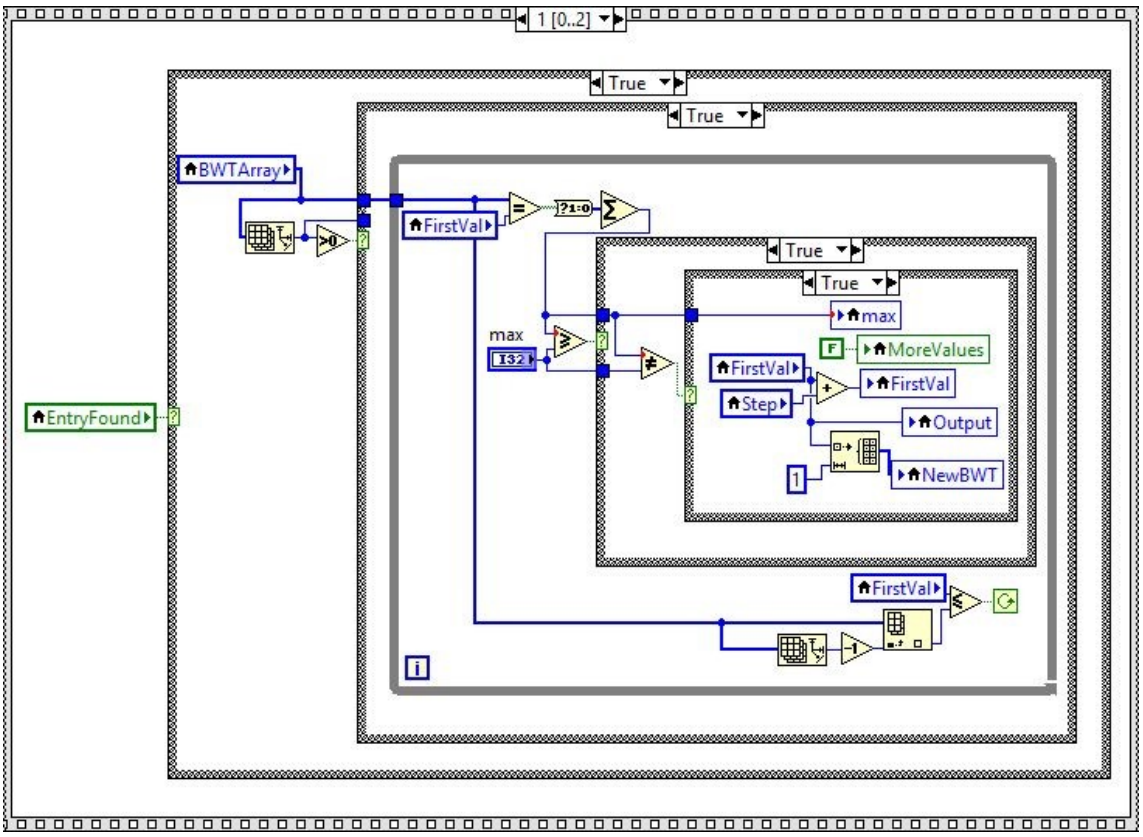


Figura 223 - Selezione del valore di maggiore frequenza del parametro BWT all'interno dell'insieme di definizione noto.

L'ultimo caso da gestire riguarda la presenza di più valori di frequenza massima. In tal caso, all'ultima porzione di codice, ovvero al vi *BWTUpdater,* vengono posti in ingresso il vettore dei valori con massima frequenza *NewBWT* e *l'InterfaceDir*, restituendo il valore finale per il parametro *BWT*.

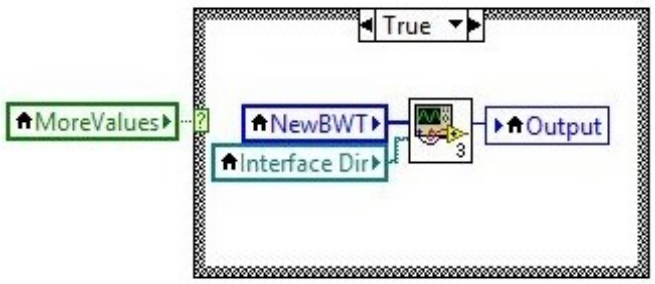


Figura 224 - Gestione della presenza di molteplici (e diversi) valori di frequenza massima.

L'ultimo passaggio atto a completare la documentazione del codice implementato per gestire il valore del parametro BandWidth Threshold è la descrizione del codice del vi *BWTUpdater* (v. Figura 225).

Figura 225 - Input/Output del vi per la selezione dei valori ottimali per i parametri BWT ed LT.

Tale vi riceve in ingresso un insieme di valori relativi a *BWT* (contenuti in un vettore di interi *BWT_Arr*), un insieme di valori relativi a *LT* (contenuti in un vettore di double *LT_Arr*) ed il percorso alla cartella

di progetto *InterfaceDir*. Il vi restituisce in output il valore selezionato di BandWidth Threshold *New-BWT* ed uno per Limit Tolerance, *NewLT*.

Per quanto concerne *BWT*, viene innanzitutto controllata la dimensione del vettore in ingresso: se tale dimensione è pari ad uno, l'unico valore presente nel vettore *BWT_Arr* viene copiato nella variabile output *NewBWT*. Se invece la dimensione è nulla, il valore da restituire tramite la variabile *NewBWT* viene impostato al valore centrale ottenuto tramite il vi *BWTValuesReader* (v. Figura 226).

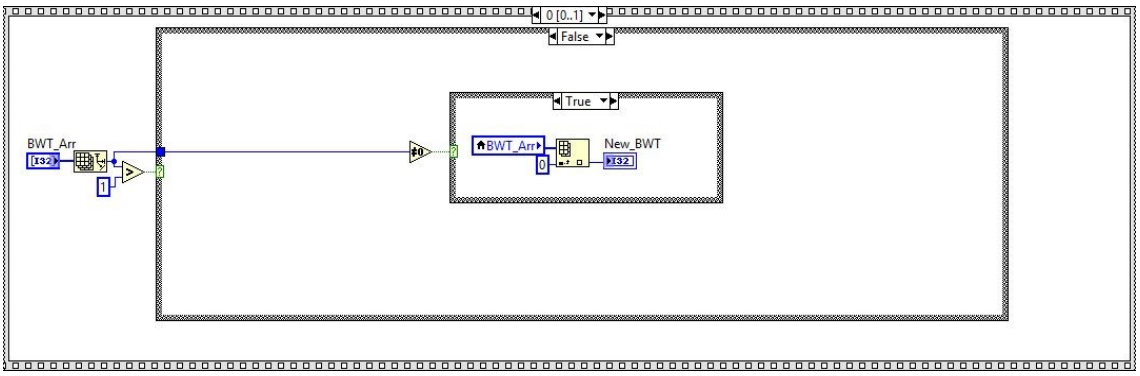


Figura 226 - Assegnazione di un valore a BWT nel caso in cui la dimensione del vettore dei suoi valori sia non superiore ad 1 (0 o 1).

Se, infine, nel vettore sono presente più elementi, allora occorre analizzarli per selezionarne il valore medio. A tal fine, viene prima calcolata la media aritmetica dei valori contenuti nel vettore *BWT_Arr*; successivamente, viene determinato l'indice del primo valore nel vettore maggiore del valore medio. Viene calcolata la differenza tra i due valori del vettore più vicini alla media aritmetica (quello minore e quello maggiore) e, tra i due, viene selezionato quello che ha distanza minore dalla media aritmetica precedentemente determinata. Tale valore, infine, viene scritto nella variabile di uscita *NewBWT* (Figura 227).

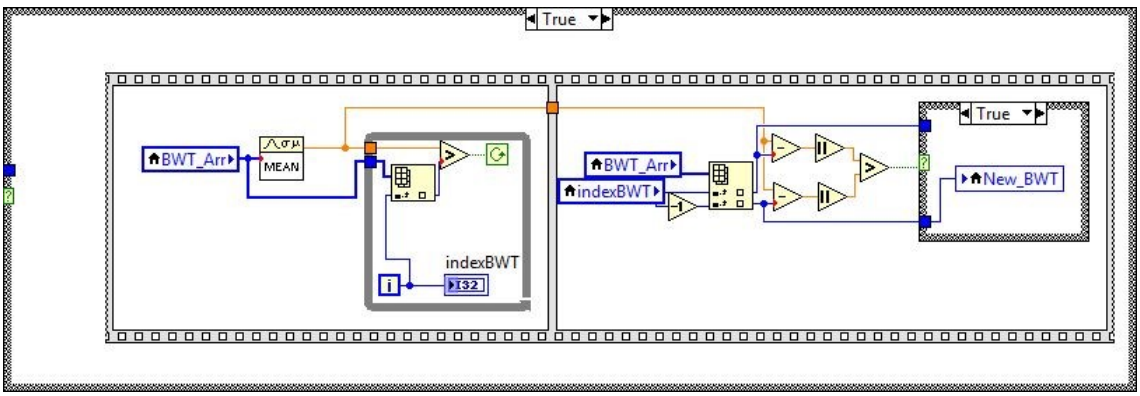


Figura 227 - Selezione del valore ottimale per il parametro BWT nel caso di numero di valori maggiore di 1.

Il codice per la selezione del valore per il parametro *LT* è assolutamente speculare, al netto della differenza sui nomi delle variabili e per il caso in cui il vettore in ingresso *LT_Arr* abbia dimensione nulla (v. Figura 228).

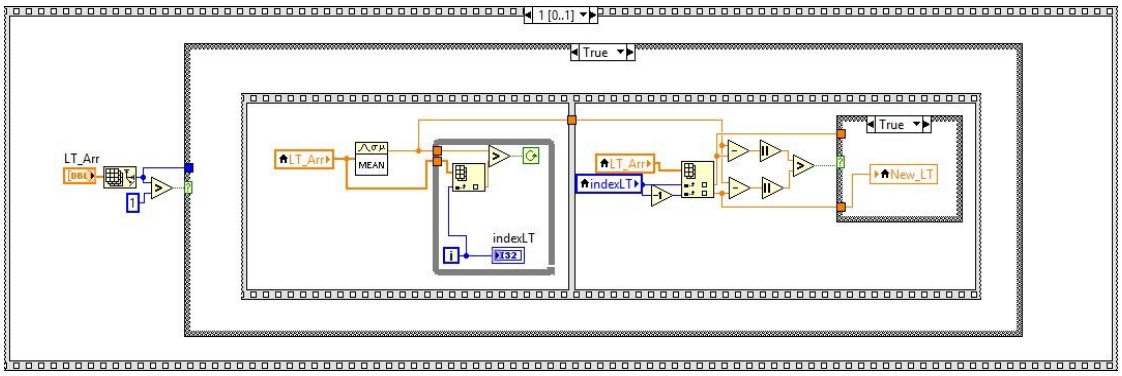


Figura 228 - Selezione del valore ottimale per il parametro LT nel caso di numero di valori maggiore di 1.

In tal caso, infatti, in assenza di un vi per la gestione del parametro LT che operasse in maniera comparabile a quanto fa il vi *BWTValuesReader* per la gestione del parametro *BWT*, il valore selezionato per l'uscita viene impostato al valore di default del parametro *LT*, che è stato attualmente stimato a 0,15. Tale scelta implementativa è legata alla minore centralità del parametro Limit Threshold rispetto a quello sulla larghezza di banda (*BWT*); appare quindi sufficiente individuare il valore 0,15 come valore centrale di default per il parametro Limit Threshold. Per quanto concerne il valore *ChooseEMA* relativo alle media mobili esponenziali, esso viene memorizzato nella relativa variabile nel caso in cui il massimo profitto sia stato ottenuto in corrispondenza di una specifica iterazione; viceversa, nel caso in cui il numero di iterazioni che corrispondono al massimo profitto è superiore ad uno, il vi *EMACounter* risolve tale ambiguità: esso riceve in ingresso un vettore di interi corrispondente ad i valori di *ChooseEMA* corrispondenti alle iterazioni con profitto massimo ed estrae quello con maggiore frequenza. In caso di pari valori di massima frequenza, viene preferito sempre il valore inferiore (v. Figura 229).

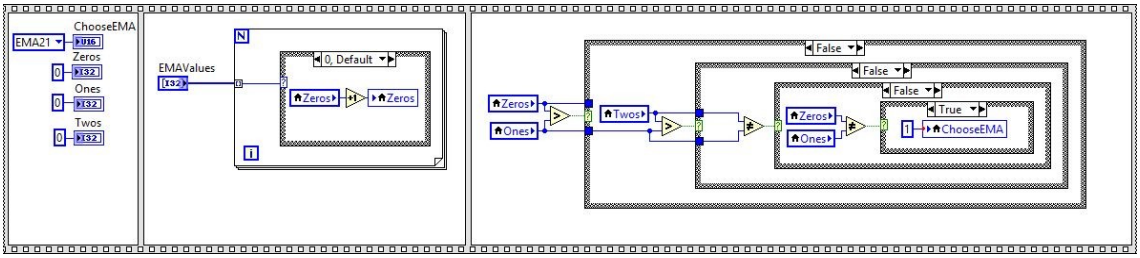


Figura 229 - Selezione del numero di periodo per la media mobile esponenziale maggiormente frequente nelle migliori iterazioni durante la fase di auto apprendimento.

8 Simulazione, Ottimizzazione, Validazione e Risultati

In questo capitolo si andranno a dettagliare le metodologie definite per effettuare il test delle strategie presentate nei capitoli 6, ovvero Volatility e 7, ovvero LoMiTTraS.

I test sono stati realizzati seguendo due differenti approcci per le differenti strategie, ovvero per quanto riguarda l'algoritmo Volatility si è preferito, essendo un algoritmo prevalentemente dedicato ad operare nei momenti di elevata volatilità, di testarlo in modalità offline. Difatti i momenti di alta volatilità, considerati come dei picchi non appartenenti ai normali movimenti di prezzo, caratterizzano il mercato solo in brevi intervalli di tempo. Infatti talvolta possono trascorrere giorni interi durante i quali tali eventi non si verificano.

Per cui al fine di effettuare i test su questi specifici eventi per cui l'algoritmo è pensato, ed inoltre non potendo attendere il verificarsi di tali eventi direttamente sul mercato reale, si è deciso di sviluppare un ulteriore applicativo atto a simulare il comportamento della piattaforma di trading.

Questo applicativo opera in sinergia con la strategia e, concettualmente, si occupa di estrarre da file di dump specifici i tick (ovvero gli aggiornamenti dei livelli di prezzo che il broker invia alla piattaforma istante per istante) dello strumento finanziario oggetto del test e scriverli all'interno della libreria condivisa con l'istanza della strategia, come se fosse una vera e propria piattaforma di trading.

Tali file di dump contenenti i tick sono stati creati manualmente, dopo aver analizzato il grafico del prezzo in piattaforma ed evidenziando situazioni tipiche nei quali l'algoritmo Volatility dovrebbe operare. Infine si è estratto il contenuto da file estesi, contenenti un ristretto intervallo di aggiornamenti di livelli di prezzo.

Per quanto concerne invece la strategia LoMiTTraS, questa è stata testata in real-time, ovvero sono stati aperti dei conti demo su piattaforma di trading ed è stata eseguita una specifica batteria di test.

Tale fase di test online è stata preceduta da una fase di analisi, durante la quale è stata eseguita la versione di *LoMiTTraS_OFFLINE*, al fine di poter stimare ed ottimizzare i parametri *Bandwidth Threshold* (ovvero l'ampiezza di banda di Bollinger consentita per l'operatività) e *Limit Tolerance* (ovvero la percentuale di sfondamento delle bande di Bollinger, inferiore e superiore, oltre la quale considerare l'apertura della posizione OCO).

Nell'ordine vedremo prima i test ed i risultati dell'analisi sugli stessi per Volatility, successivamente lo stesso per LoMiTTraS.

8.1 Test Volatility

Per ogni singolo evento è stato generato un file di definizione dei parametri composto da 3888 configurazioni. Tale numero si distacca di molto dal totale numero di configurazioni possibili. Difatti, essendo la simulazione eseguita su eventi di calendario, l'obiettivo primario è quello di rendere l'ingresso a mercato quanto più rapido possibile nella corretta direzione, massimizzando il guadagno. Per fare ciò si è reso necessario restringere il dominio di variazione di ogni singolo parametro.

Nello specifico, in Tabella 3 si individuano le seguenti possibili variazioni per i parametri:

Tabella 3 - Intervallo di variazione dei parametri per la fase di test.

1	Algoritmo	MOAA||MOVA, MOAA && MOVA;
2	MOAA PIP Limit	7,8,9,10,11,12
3	MOAA X+Y > ?	3,4,5
4	MOAA Y-X < ?	-3,-4,-5
5	MOVA PIP Limit	9,10,11,12,13,14
6	MOVA X+Y > ?	3,4,5
7	MOVA Y-X < ?	-3,-4,-5
8	Beta Value → { |Sum(PIP)| > Beta * (Sum|PIP|) }	0.50, 0.66, 0.75
9	|Sum(PIP)| > PIP_Limit in congiunzione con parametro 7	AND, OR
10	NumOfPoint per costruzione retta dei minimi quadrati sui tick	5

Alcuni parametri invece sono stati stabiliti preliminarmente, così da poterli fissare nella batteria di test (v. Tabella 4):

Tabella 4 - Parametri fissati per la batteria di test sugli eventi di volatilità.

1	NumSqueezeCandle	≥ 5
2	Alpha Value → { 1/alpha * EstensioneCandelaEvento }	10
3	Percentuale Ritracciamento	20%
4	DePIP_Over30 (Considera variazioni di almeno 30 Pip)	Attiva
5	EXIST_4 (Riduzione di Vel ed Acc)	Attiva
6	Percentuale di Riduzione di Velocità	30%
7	Percentuale di Riduzione di Accelerazione	30%
8	Riduzione di Vel AND/OR Acc	AND

Gli eventi, individuati su differenti strumenti finanziari, su cui sono stati eseguiti i test sono i seguenti:

1. GBP/USD – Evento del 19.02.2014 - 11.30
2. GBP/USD – Evento del 21.02.2014 - 11.30
3. GBP/USD – Evento del 06.03.2014 - 15.30
4. GBP/USD – Evento del 06.03.2014 - 14.00
5. EUR/GBP – Evento del 21.02.2014 - 11.30
6. EUR/USD – Evento del 06.03.2014 - 14.40
7. EUR/USD – Evento del 06.03.2014 - 15.30
8. GOLD – Evento del 07.03.2014 - 15.25
9. GOLD – Evento del 07.03.2014 - 16.30
10. USD/CAD – Evento del 21.02.2014 - 15.30

11. USD/CAD – Evento del 06.03.2014 - 17.00

Al fine di rendere più semplice l'interpretazione della tipologia di eventi di mercato per cui è stato modellato l'algoritmo Volatility, nella Figura 230, Figura 231 e Figura 232 si mostrano alcuni eventi di esempio su cui l'algoritmo è stato testato. Essi rappresentano l'andamento dello strumento finanziario, nell'arco temporale di interesse per l'algoritmo (ovvero nei momenti in cui è avvenuto l'ingresso a mercato), rappresentati su timeframe 1 minuto.

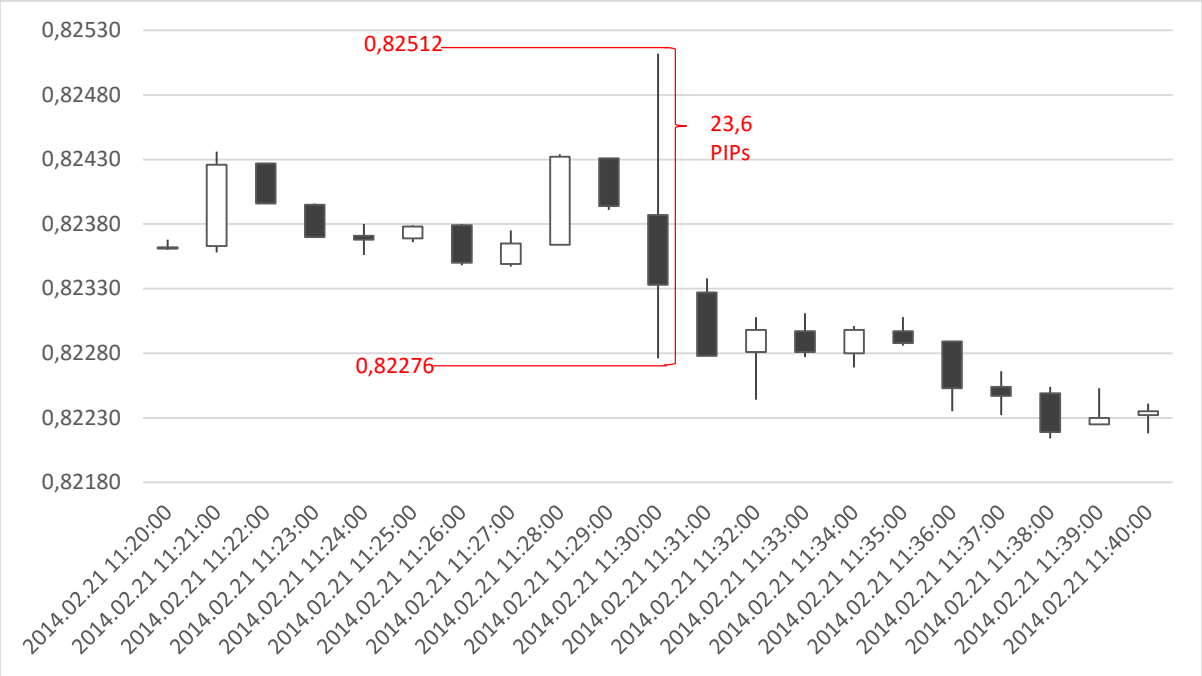


Figura 230 - EUR/GBP - EVENTO 21/02/2014 - 11.30

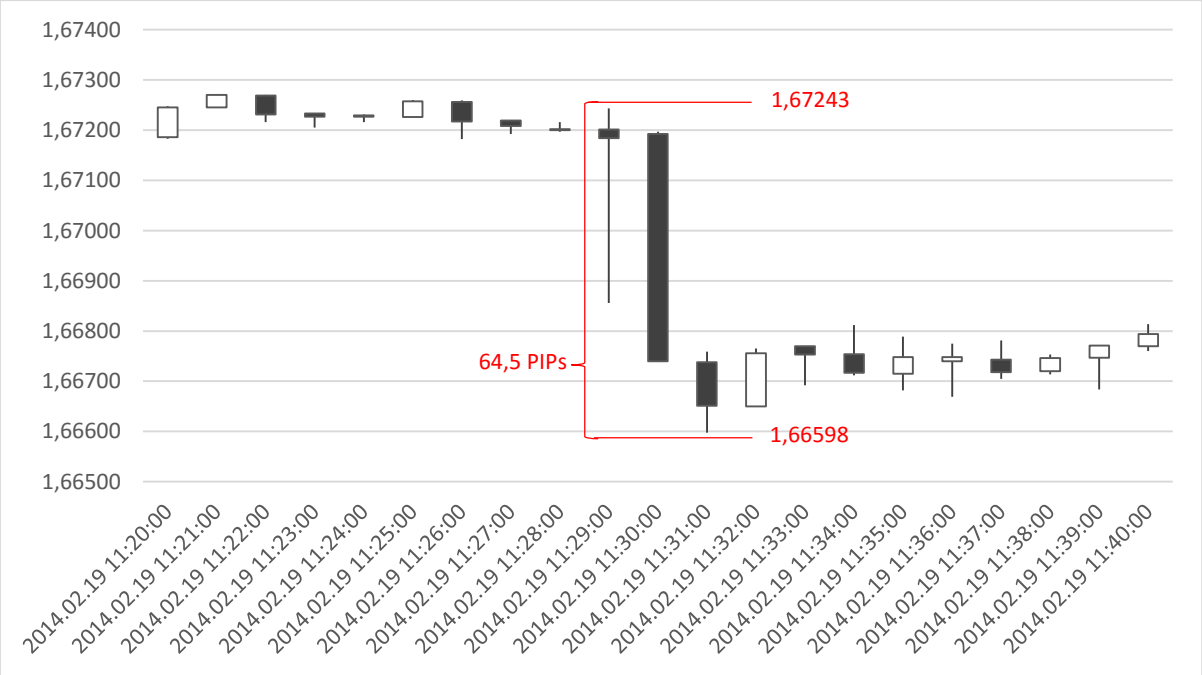


Figura 231 - USD/CAD - EVENTO 19/02/2014 - 11.30

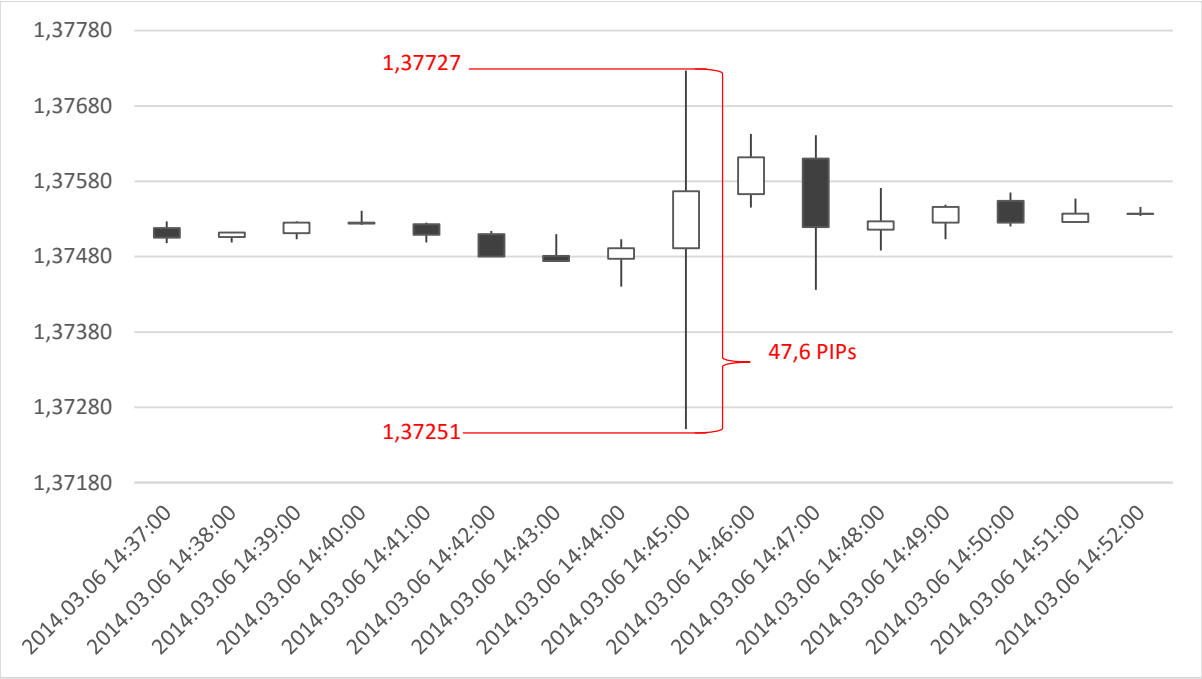


Figura 232 - EUR/USD - EVENTO 06/03/2014 - 14.40

In Figura 233 si mostra un esempio di evento ad una granularità ancora più fine del timeframe 1 minuto, ovvero l'andamento dei tick relativi ad un evento ad elevata volatilità. Tale grafico mostra ancora più nel dettaglio come le variazioni dello strumento finanziario siano repentine e non prevedibili, a meno che non vi sia eventualmente la presenza di una news inerente lo strumento finanziario, anche se ciò non garantisce la presenza di un movimento. Appunto per questo risulta necessario avere degli algoritmi che siano in grado di rilevare rapidamente le variazioni di volatilità, stabilire la direzione di ingresso ed eseguire l'operazione. Allo stesso modo è necessario che le strategie d'uscita siano rapide nell'individuare il momento migliore di uscita dal mercato, onde evitare perdita di profitti già consolidati per improvvise variazioni di volatilità o ritracciamenti sullo stesso evento.

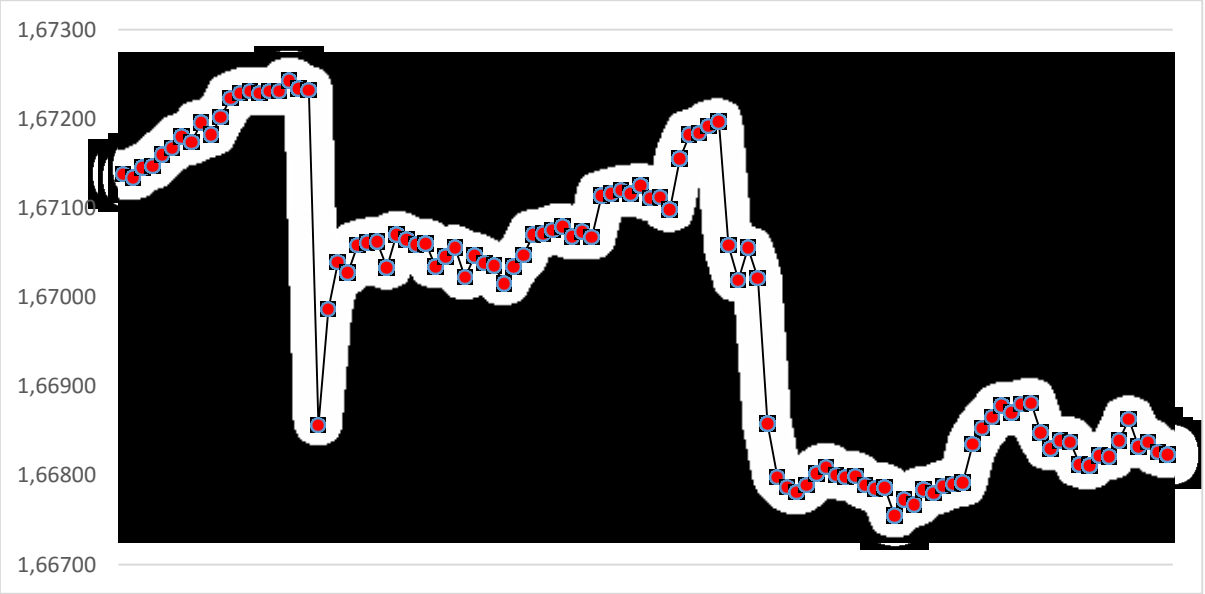


Figura 233 - Esempio Evento Alta Volatilità: Andamento dei Tick

8.1.1 Analisi dei risultati

L'analisi dei profitti è stata realizzata in due modalità, ovvero prima analizzando i profitti del singolo test, individuando quale sia la migliore iterazione e il relativo guadagno in termini di PIPs.

Si individuano quindi i risultati riassunti in Tabella 5:

Tabella 5 - Migliore configurazione parametrica per il singolo evento di volatilità.

Event ID	Best Configuration ID	Profit (PIP)	#Market Operation
1	27	38.8	6
2	1274	203	7
3	1207	52.3	8
4	681	63.8	11
5	55	4.5	3
6	93	3.7	1
7	1287	68.1	15
8	332	415.1	39
9	825	158.2	9
10	177	56.6	13
11	147	1.8	3

Si può notare che determinati eventi di mercato, in base alla tipologia del movimento, permettono all'applicativo di produrre o meno profitti, indipendentemente dal set di parametri utilizzato. Ad esempio il Test_11, nella sua configurazione più profittevole, ha prodotto solamente 1.8 PIPs di guadagno. Guadagni simili si ottengono anche nel test 5 e 6.

I valori dei parametri relativamente alle migliori iterazioni degli 11 eventi sono dettagliate in Tabella 6:

Tabella 6 - Dettaglio delle configurazioni parametriche relative alle migliori configurazioni individuate.

Evento	MOAA MOVA	P_Limit MOAA	Y-X<?	X+Y>?	P_Limit MOVA	Y-X<?	X+Y>?	beta	SumPip OR|AND	#point
1	OR	7	-4	4	9	-4	4	0,75	OR	5
2	OR	10	-4	4	14	-5	5	0,66	OR	5
3	OR	10	-4	4	13	-3	3	0,5	OR	5
4	OR	9	-4	4	9	-5	5	0,75	OR	5
5	OR	7	-3	3	10	-3	3	0,5	OR	5
6	OR	7	-5	5	10	-3	3	0,75	OR	5
7	OR	10	-5	5	14	-4	4	0,75	OR	5
8	OR	8	-3	3	9	-4	4	0,66	OR	5
9	OR	9	-3	3	12	-5	5	0,75	OR	5
10	OR	7	-3	3	12	-5	5	0,75	OR	5
11	OR	7	-5	5	11	-3	3	0,75	OR	5

In seconda istanza si è deciso di analizzare i test globalmente, al fine di individuare l'ottimo globale, così da capire quale sia la configurazione dei parametri che risulta essere più profittevole su tutti gli eventi.

Questa seconda analisi è stata effettuata individuando dapprima il massimo profitto per ogni singolo evento. La sommatoria di tali massimi è divenuta il divisore per la cumulata dei profitti per ogni configurazione dei parametri sul singolo evento. Il massimo tra questi rapporti, infine, compreso nell'intervallo [0,1], rappresenta il guadagno effettivo, fatto 1 il massimo, che l'applicativo è stato in grado di ottenere.

In Formule:

$$d = \sum_{i=1}^{11} MaxProfitEvent_i$$

$$Ratio_n = \frac{\sum_{k=1}^{11} ProfitTest_n Event_k}{d}, \qquad n = 1 \dots 3888.$$

Infine si individua il massimo:

$$BestRatio = \max_n(Ratio)$$

Inoltre si è riusciti ad identificare quale sia stata, per questi 11 eventi, la migliore configurazione globale di parametri.

Nel Dettaglio:

Massimo Profitto	481.6 PIP
Max Profitto %	45% (0.453)
Best Configuration ID	3573

I Parametri per la configurazione 3573 sono indicati in Tabella 7:

Tabella 7 - Parametri relativi alla configurazione parametrica globale.

1	Algoritmo	MOAA && MOVA
2	MOAA PIP Limit	12
3	MOAA X+Y > ?	3
4	MOAA Y-X < ?	-3
5	MOVA PIP Limit	9
6	MOVA X+Y > ?	4
7	MOVA Y-X < ?	-4
8	Beta Value → { |Sum(PIP)| > Beta * (Sum|PIP|) }	0.75
9	|Sum(PIP)| > PIP_Limit in congiunzione con parametro 7	OR
10	NumOfPoint per costruzione retta dei minimi quadrati sui tick	5

8.1.2 Analisi Sui Singoli Strumenti

Da una analisi più approfondita (per individuare la migliore configurazione in base ai profitti), effettuata includendo nel test soltanto eventi relativi allo stesso strumento, al fine di annullare l'ulteriore variabile aggiunta dalla diversa tipologia di movimento dei singoli strumenti finanziari (si veda ad esempio il GOLD e l'EUR/USD), si può sicuramente riuscire ad individuare un set di parametri adeguato al singolo strumento. Difatti sono correntemente in fase di esecuzione dei test su ulteriori eventi individuati sugli strumenti analizzati in questo report, per poter approfondire tale teoria.

Dai test effettuati, riportando su di un grafico i profitti, si evince come determinati set di parametri inibiscono l'ingresso a mercato o comportano minimi profitti/perdite, per tutti gli eventi in analisi. Ciò fa intuire come vi siano alcune configurazioni particolari che potrebbero essere eliminate dalla batteria di test. Indipendentemente dall'evento quindi, approssimativamente le aree in cui non vengono aperte

posizioni (oppure in cui tali posizioni non forniscono informazioni riguardo l'operato dell'applicativo) includono le configurazioni di test dal *#645* al *#1950* e dal *#2110* al *#2750*. La Figura 234 rappresenta un esempio di tale condizione.

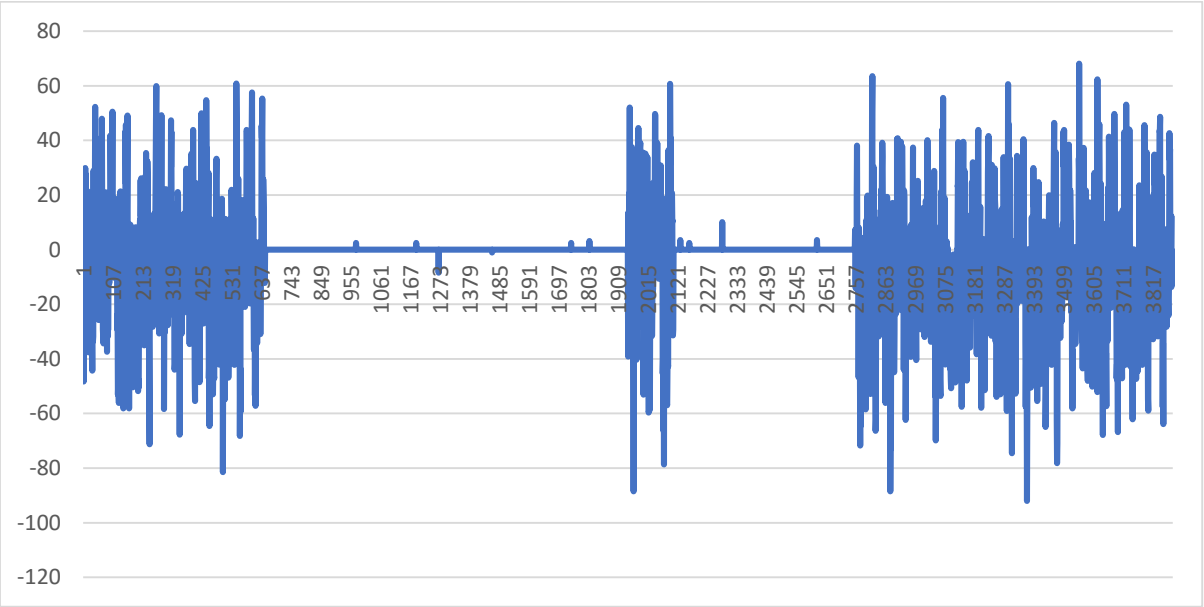


Figura 234 – EUR/USD: Risultati configurazioni di test su eventi ad alta volatilità.

Di seguito invece si mostrano i profitti tramite istogramma. Questa rappresentazione evidenzia come la distribuzione dei profitti rispetto alle configurazioni parametriche comporti un accumulo delle configurazioni su determinati intervalli di profitto, in base ovviamente alla tipologia di movimento del mercato. Ciò mostra come le configurazioni dei parametri dell'algoritmo che effettivamente permettono di sfruttare tutto il movimento del mercato sono ridotte, ovvero le configurazioni maggiormente profittevoli ricadono nelle code della distribuzione. Per brevità di rappresentazione, si mostrano solo alcuni degli istogrammi riferiti agli eventi su cui sono stati effettuati i test.

Ad Esempio, in Figura 235 si nota come vi siano 2363 configurazioni parametriche che, complessivamente, generano un profitto/perdita compreso tra -2 e 8 PIPs. Invece, a dimostrazione di quanto già affermato, tale numero si riduce in maniera consistente man mano che aumenta il guadagno, in termini di PIPs: una sola configurazione comporta un profitto compreso tra 68 e 78 PIPs, 6 per un profitto compreso tra 58 e 68 PIPs, e così via, anche per gli altri eventi (v. Figura 236, Figura 237, Figura 238 e Figura 239).

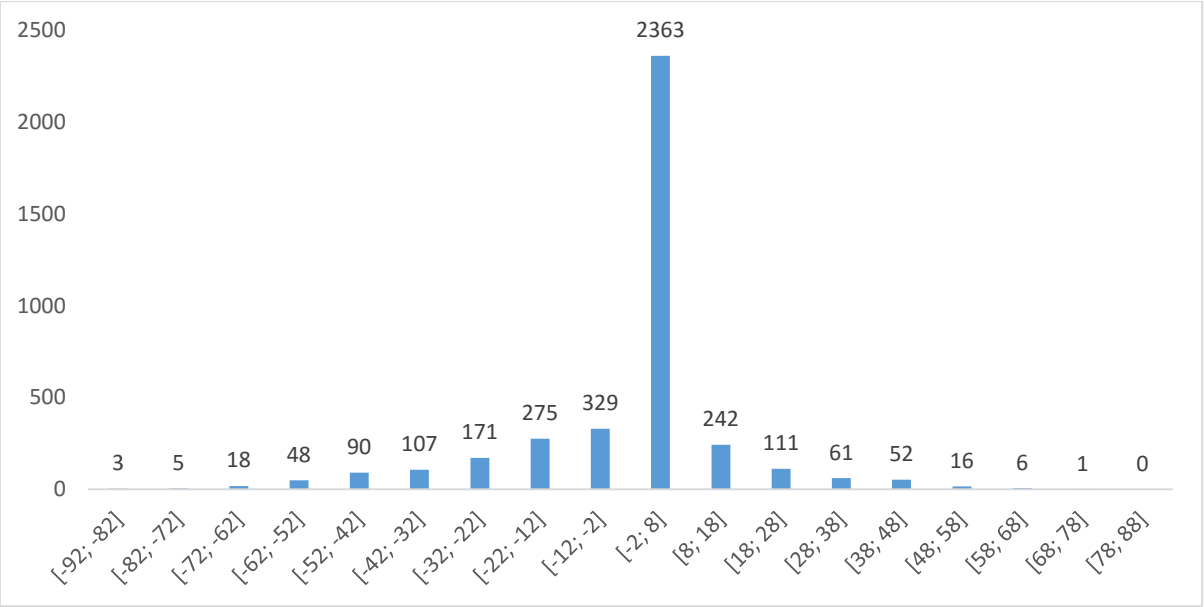


Figura 235 - EUR/USD: Istogramma di distribuzione dei profitti.

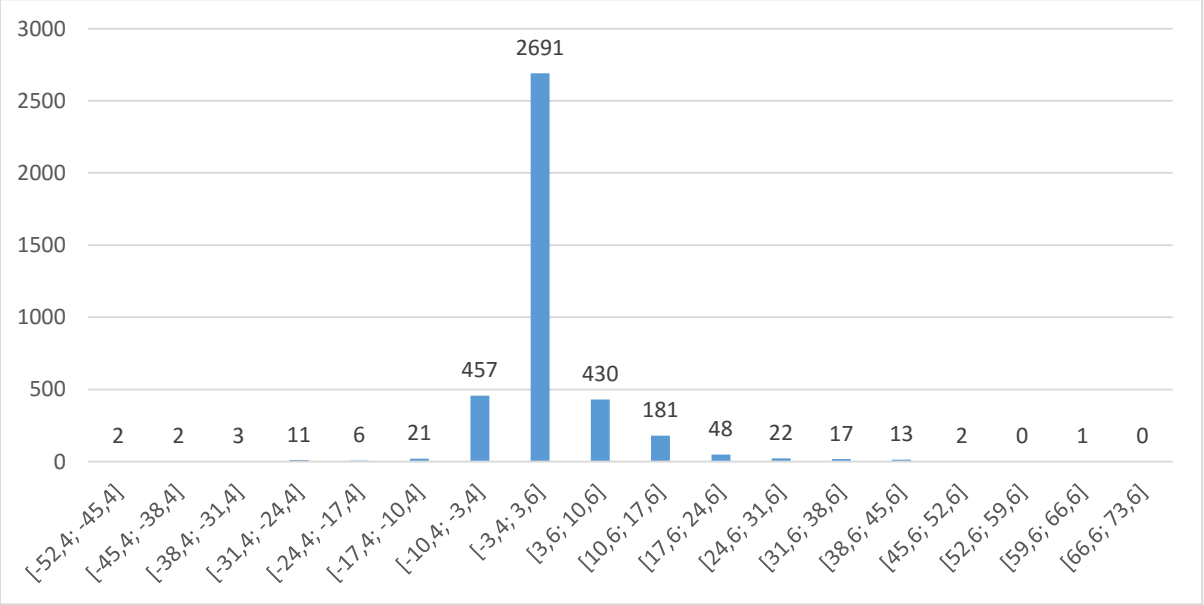


Figura 236 - GBP/USD: Istogramma distribuzioni profitti (Evento 1).

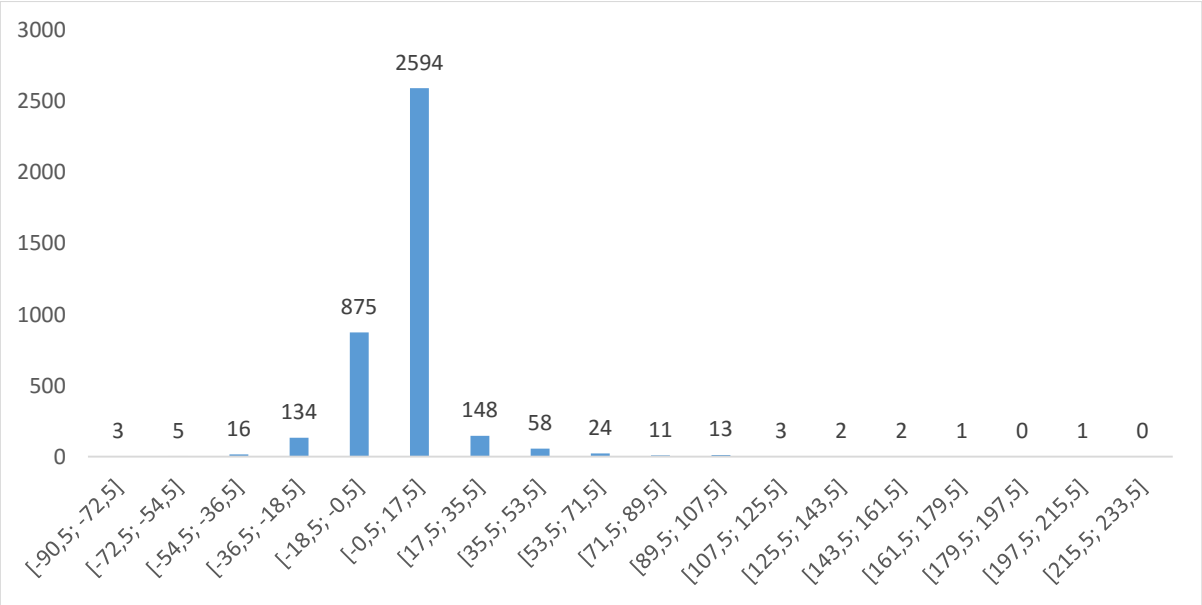


Figura 237 - GBP/USD: Istogramma distribuzioni profitti (Evento 4).

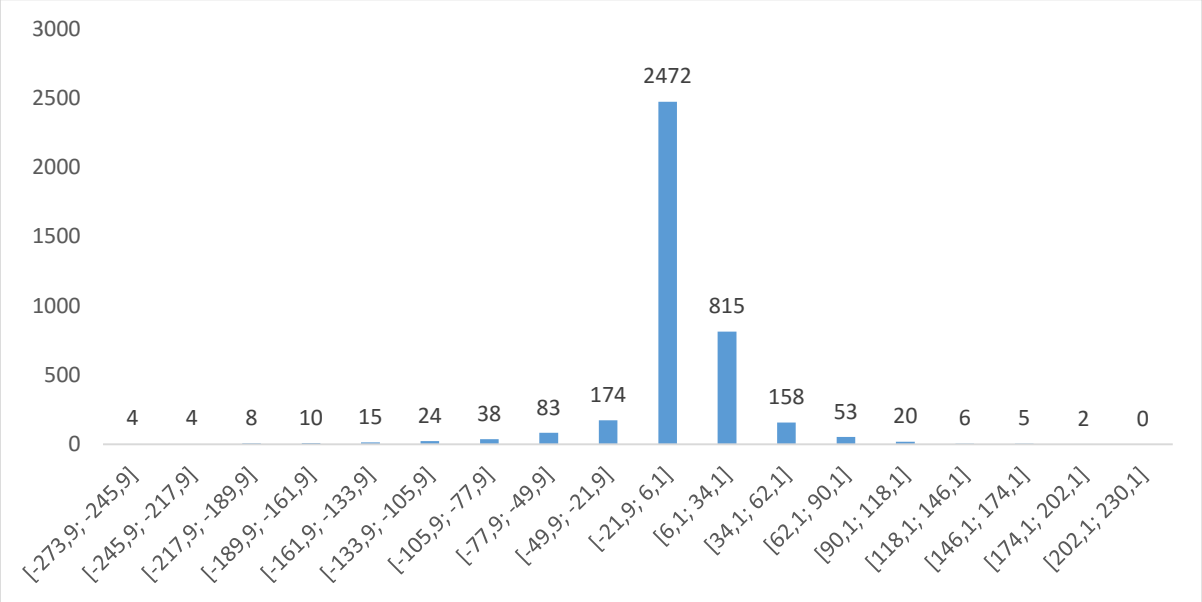


Figura 238 - GOLD: Istogramma distribuzioni profitti.

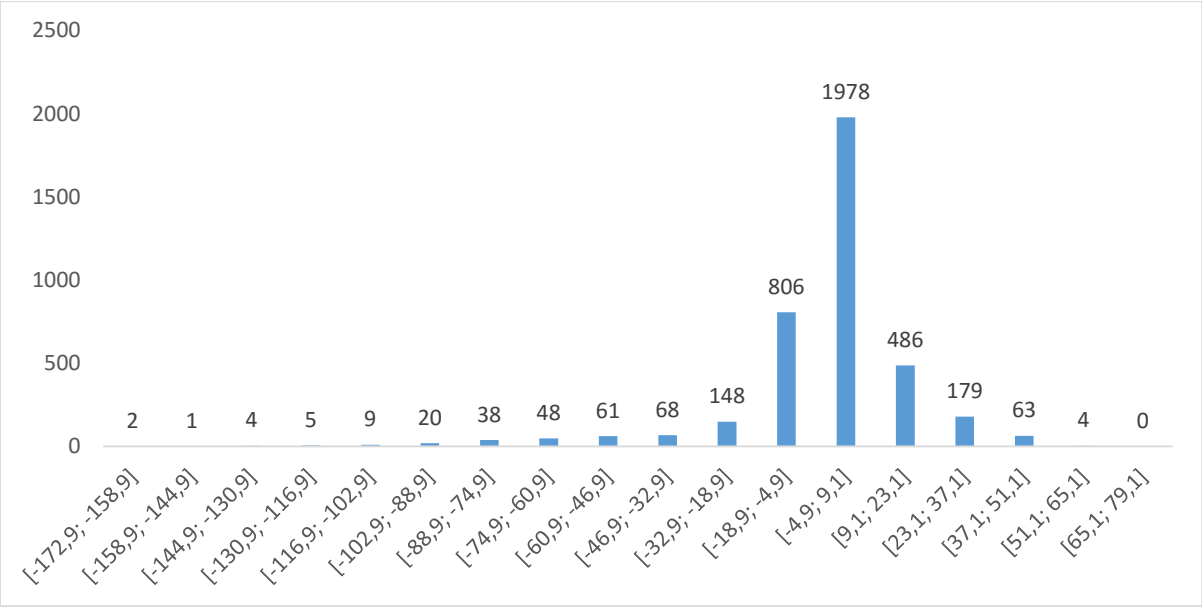


Figura 239 - USD/CAD: Istogramma distribuzioni profitti.

8.2 Test LoMiTTraS

Riguardo l'algoritmo LoMiTTraS sono stati, come detto, eseguiti dei test ponendolo in esecuzione su conti demo. Nello specifico sono stati testati gli strumenti di seguito elencati, su quattro timeframe differenti, ovvero 5min, 15min, 30min, 1h:

- EURUSD
- GBPUSD
- GOLD
- AUDUSD
- S&P500 (#EPM4)

Il test ha avuto durata complessiva di cinque settimane, ovvero dal 31/03/2014 al 02/05/2015, ed al termine di ogni settimana è stato estratto un resoconto. Tale resoconto analizza l'operatività dell'algoritmo su ogni strumento, per ogni timeframe, e ne estrae profitto complessivo (in PIPs) e numero di operazioni totali. Di seguito le tabelle riassuntive, suddivise per settimana:

Si specifica che le celle in cui non è presente il profitto/perdita in PIPs e il numero di operazioni, indicano che in quella giornata l'algoritmo, per lo specifico timeframe, non ha effettuato operazioni a mercato.

1° Settimana

	GOLD 5m	**GOLD 15m**	**GOLD 30m**	**GOLD 1h**	**EURUSD 5m**	**EURUSD 15m**	**EURUSD 30m**	**EURUSD 1h**
31/3	-30.5/4	-22.3/2					-14.2/2	
1/4	-26.4/2							
2/4	-39.4/2		-2.6/2	-24/2				
3/4				-12.2/2				-4/2
4/4	-0.1/4	-84.7/2	-9/2	22.6/2				
Tot	**-96,4/12**	**107/4**	**-11.6/4**	**-13.6/6**	**0/0**	**0/0**	**-14.2/2**	**-4/2**
	GBPUSD 5m	**GBPUSD 15m**	**GBPUSD 30m**	**GBPUSD 1h**	**AUDUSD 5m**	**AUDUSD 15m**	**AUDUSD 30m**	**AUDUSD 1h**
31/3		-5.9/2			-3.2/2			
1/4	20.7/4				-3.5/2			
2/4								
3/4								
4/4							6.3/2	-4.1/2
Tot	**20.7/4**	**-5.9/2**	**0/0**	**0/0**	**-6.8/4**	**0/0**	**6.3/2**	**-4.1/2**

	SP&500 5m	**SP&500 15m**	**SP&500 30m**	**SP&500 1h**
31/3				
1/4				
2/4	-7.5/6	-1.5/2	-5/2	-5/2
3/4	37.5/2			
4/4	-5/2	77.5/4	140.5/2	15/2
Tot	**25/10**	**76/6**	**135.5/4**	**10/4**

2° Settimana:

	GOLD 5m	GOLD 15m	GOLD 30m	GOLD 1h	EURUSD 5m	EURUSD 15m	EURUSD 30m	EURUSD 1h
7/1	-32.8/4				-23.7/8			-5/2
8/4	-15.2/2	-16.5/2		6.5/2	-17/10			-5.5/2
9/4	-67.7/8	-49.1/4						
10/4	47.1/8							
11/4	-16.4/2				-10/2			
Tot	**-85/24**	**-65.6/6**	**0/0**	**6.5/2**	**-50.7/18**	**0/0**	**0/0**	**-10.5/4**
	GBPUSD 5m	**GBPUSD 15m**	**GBPUSD 30m**	**GBPUSD 1h**	**AUDUSD 5m**	**AUDUSD 15m**	**AUDUSD 30m**	**AUDUSD 1h**
7/1	-2.3/2	2/2	-17.8/2	-15.7/2		2.8/2		
8/4	-28.8/6	10.9/2	1.6/2	24.1/2	-8.8/2	-11.5/2	-20/2	8/2
9/4	18.8/2	-8.4/2			-4.9/2	-7.6/2		
10/4					-17.2/4	-24.8/2	-4/2	
11/4	-8.7/2	-4.7/2		6.9/2	-19.4/18	17.6/4	-35.1/4	
Tot	**-21/12**	**-0.2/8**	**-16.2/4**	**15.3/6**	**-50.3/26**	**-23.5/12**	**-59.1/8**	**8/2**

	SP&500 5m	SP&500 15m	SP&500 30m	SP&500 1h
7/1	-57.5/4	-7.5/2	-27.5/2	-65/2
8/4	25/2	10/6	-30/2	
9/4				
10/4	-105/6	160/4	222.5/2	50/4
11/4		30/4	-27.5/2	
Tot	**-137.5/12**	**192.5/16**	**137.5/8**	**-15/6**

3° Settimana

	GOLD 5m	GOLD 15m	GOLD 30m	GOLD 1h	EURUSD 5m	EURUSD 15m	EURUSD 30m	EURUSD 1h
14/1	-35.1/4				1.3/2	-4.8/2		
15/4	-151.8/8	-103.8/4	-46.1/6	-94.4/2	-16.8/4	-12.2/4		-3.2/2
16/4	33.4/4				-15.4/6	-3.4/6	3.3/2	-14.7/2
17/4	-41.4/6	-26.5/2	-16.4/2		-15.9/12	-18.6/8	-26.5/6	1.7/2
18/4								
Tot	**-194.9/22**	**-130.3/6**	**-62.5/8**	**-94.4/2**	**-46.8/24**	**-39/20**	**-23.2/8**	**-16.2/6**
	GBPUSD 5m	**GBPUSD 15m**	**GBPUSD 30m**	**GBPUSD 1h**	**AUDUSD 5m**	**AUDUSD 15m**	**AUDUSD 30m**	**AUDUSD 1h**
14/1					-7.5/2			
15/4	-25.7/10	-10.9/6	-28.5/2		-3.6/4	-27/6	0.6/2	
16/4	-23/10	-18.9/2	-27.3/2					
17/4	-0.7/12	-8.6/2	-20/4					
18/4	-7.8/2							
Tot	**-57.2/34**	**-38.4/10**	**-75.8/8**	**0/0**	**11.1/6**	**-27/6**	**0.6/2**	**0/0**

	SP&500 5m	SP&500 15m	SP&500 30m	SP&500 1h
14/1	30/4	7.5/4	-45/2	30/2
15/4	155/4	150/6	232.5/6	72.5/2
16/4		-50/6	37.5/2	
17/4	40/2	20/4	-70/4	-27.5/2
18/4				
Tot	**225/10**	**195/20**	**155/14**	**75/6**

4° Settimana:

	GOLD 5m	GOLD 15m	GOLD 30m	GOLD 1h	EURUSD 5m	EURUSD 15m	EURUSD 30m	EURUSD 1h
21/4	-16.2/8	43.9/2		-18.6/2	1.2/4	-21.1/4	-6.8/2	-3.8/2
22/4	-2.2/2	-10.9/2	22.1/2		-17.2/8			
23/4					-10.9/6			
24/4	-164.9/8	-2.5/4	22.9/2	-10/2	-19/8	-12.1/4	5.7/2	0.2/4
25/4	3.4/2	0.2/2	-38.7/2		-17.4/6			
Tot	**-179.9/20**	**30.7/10**	**6.3/6**	**28.6/4**	**-63.3/32**	**-33.2/8**	**-1.1/4**	**-3.6/6**
	GBPUSD 5m	**GBPUSD 15m**	**GBPUSD 30m**	**GBPUSD 1h**	**AUDUSD 5m**	**AUDUSD 15m**	**AUDUSD 30m**	**AUDUSD 1h**
21/4	-19.9/6							
22/4	-17.7/6							
23/4	-58.7/14	-33.2/6	-32.2/2	-5.2/2	7.9/4	-9.4/2	23.9/2	0.8/2
24/4	-37.3/8	-8/2			-15.2/4			
25/4	-42.3/14				-27.3/12			-1.4/2
Tot	**-175.9/48**	**-41.2/8**	**-32.2/2**	**-5.2/2**	**-34.6/20**	**-9.4/2**	**23.9/2**	**-0.6/4**

	SP&500 5m	SP&500 15m	SP&500 30m	SP&500 1h
21/4	-10/2			
22/4	7.5/2	2.5/2	35/2	12.5/2
23/4	-5/2	-20/2	-10/2	
24/4	-5/4	-35/2	-55/4	
25/4	-30/2	-75/6	-22.5/4	-5/2
Tot	**-42.5/12**	**-132.5/12**	**-52.5/12**	**7.3/4**

5° Settimana

	GOLD 5m	GOLD 15m	GOLD 30m	GOLD 1h	EURUSD 5m	EURUSD 15m	EURUSD 30m	EURUSD 1h
28/4	-1.9/2	-23.6/2			5.5/2	11.5/6	-2.3/2	5.7/2
29/4	-29.4/4	-27/2			2.8/2	-26.2/6	-17.9/2	4.6/2
30/4	-8.7/6				-29.7/20	-40.8/4	-18.1/2	
1/5								
2/5	-16.7/6	0.3/4	-45.5/5		-21.3/4	-10.9/2	-25.7/2	
Tot	**-56.7/18**	**-50.3/8**	**-45.5/5**	**0/0**	**-42.7/28**	**-66.4/18**	**-64/8**	**10.3/4**
	GBPUSD 5m	**GBPUSD 15m**	**GBPUSD 30m**	**GBPUSD 1h**	**AUDUSD 5m**	**AUDUSD 15m**	**AUDUSD 30m**	**AUDUSD 1h**
28/4	17.5/6	-1.9/6	29/4		-20.5/10	-34.1/8		
29/4	-19.8/8	-11.2/2	-18.4/2		-41/12	-20.2/6		
30/4	-31.8/4	-4.2/2			-17.7/8	1.6/2		
1/5								
2/5	-30.9/4	-14.5/2			-3.3/2	17/4	-38.1/2	
Tot	**-65/22**	**31.8/12**	**10.6/6**	**0/0**	**-82.5/32**	**-35.7/20**	**-38.1/2**	**0/0**

	SP&500 5m	SP&500 15m	SP&500 30m	SP&500 1h
28/4	15/4	-20/10	-95/4	-7.5/2
29/4	50/2	-15/2	5/2	
30/4				
1/5				
2/5				
Tot	**65/6**	**-35/12**	**-90/6**	**-7.5/2**

Riassumendo, sull'intero arco temporale di test si hanno i risultati mostrati in Tabella 8:

Tabella 8 - Risultati dell'operatività totale per l'algoritmo LoMiTTraS

	GOLD 5m	GOLD 15m	GOLD 30m	GOLD 1h
Tot	**-612,9/96**	**-108.5/34**	**-113,3/23**	**-72.9/14**
	EURUSD 5m	**EURUSD 15m**	**EURUSD 30m**	**EURUSD 1h**
Tot	**-203.5/104**	**-138.6/46**	**-103.3/22**	**-24.2/22**
	GBPUSD 5m	**GBPUSD 15m**	**GBPUSD 30m**	**GBPUSD 1h**
Tot	**-298.4/120**	**-53.9/40**	**-113.6/20**	**10.1/8**
	AUDUSD 5m	**AUDUSD 15m**	**AUDUSD 30m**	**AUDUSD 1h**
Tot	**-163.1/88**	**-95.6/40**	**-67.5/16**	**3,3/8**
	SP&500 5m	**SP&500 15m**	**SP&500 30m**	**SP&500 1h**
Tot	**135.5/50**	**296/66**	**285.5/42**	**69.8/22**

Come si evince dai risultati, l'algoritmo non risulta essere profittevole, quantomeno lo è solamente sullo strumento S&P500, probabilmente in quanto LoMiTTraS non è in grado di operare su strumenti notoriamente con una maggiore volatilità intrinseca, come il Gold. Inoltre necessita che vi sia un certo grado di direzionalità del mercato, ovvero che il mercato, una volta avviato il movimento, perseveri in questa variazione di stato, al fine di permettere, una volta effettuata la chiusura dell'operazione in direzione opposta a quella individuata dall'algoritmo, di raggiungere un profitto, anche minimo. Sicuramente l'introduzione della fase di Self Learning ha migliorato di molto il comportamento dell'algoritmo, evitando l'apertura di posizioni in condizioni del mercato non adatte a questa tipologia di operatività.

Inoltre l'operatività è sicuramente maggiore su bassi timeframe, dove ovviamente si ha maggiore possibilità di entrare a mercato.

9 Conclusioni

Il lavoro di ricerca descritto nei capitoli precedenti, come ampiamente discusso, si basa sull'applicazione dei sistemi di supporto alle decisioni ai mercati finanziari e, nel dettaglio, nella modellazione e sviluppo di un sistema decisionale automatizzato, atto a superare le criticità note nelle attività di trading manuale o semi-automatico. Inoltre la modellazione e lo sviluppo di sistemi di automatizzati complessi, permette l'approccio a tecniche di High Frequency Trading.

È stata quindi sviluppata una complessa infrastruttura software, composta da svariati moduli atti ad adempiere a compiti specifici, con l'obiettivo comune di realizzare un sistema decisionale automatico per il trading finanziario. Nello specifico, dopo una intensa fase di studio del dominio applicativo, il quale può variare il comportamento, anche in modo profondo, in brevissimo tempo, vista la forte dipendenza da sollecitazioni esterne, si è proceduto con la modellazione e lo sviluppo di varie tecnologie che potessero supportare il decisore (nel caso specifico il trader) nell'attività decisionale. La fase successiva è stata quella di rendere completamente automatizzata (o quasi) l'attività decisionale e di applicazione delle strategie individuate, trasformando il sistema di supporto alle decisioni in un sistema decisionale automatizzato.

A tal proposito è stata realizzata una interfaccia per la comunicazione tra la piattaforma di trading MetaTrader4, la quale non permette l'accesso ai dati provenienti dal broker se non tramite l'uso di Expert Advisor, e due algoritmi modellati per operare in condizioni di mercato differenti. Inoltre sono stati realizzati ed integrati nell'infrastruttura di trading, dei software atti a fornire dei servizi accessori sia al trader umano (vedi la possibilità di effettuare il cloning dell'operatività di un conto master e la possibilità di usufruire di un calendario relativo alle news finanziarie del giorno), sia agli algoritmi, i quali sfruttano tali sistemi per modificare il loro comportamento in base alle richieste del trader e/o alle condizioni del mercato.

Gli algoritmi di trading veri e propri, ovvero l'algoritmo pensato per il trading in condizioni di alta volatilità e quello per le operazioni OCO, sono stati sviluppati a seguito di una ampia modellazione matematica, atta a considerare, quanto possibile, molte delle condizioni in cui il mercato possa trovarsi. In quest'ottica si è cercato di evitare che il trader possa subire delle ingenti perdite dovute a condizioni di mercato non ottimali, dotando gli algoritmi di numerose strategie di chiusura delle posizioni.

La fase di test degli algoritmi ha portato ad analizzare ovviamente l'operatività in fasi di mercato completamente differenti, dato che gli algoritmi sfruttano movimenti del mercato differenti ed operando su timeframe differenti. Dall'analisi dei risultati si evidenzia come l'algoritmo per la volatilità, testato in modalità offline, su dati acquisiti dalla piattaforma, riesce ad essere profittevole in situazioni nelle quali la volatilità è elevata, ma il mercato non mostra indecisioni, in quanto le così dette "finte" possono provocare l'apertura di posizioni errate. Inoltre l'utilizzo del software di simulazione della piattaforma di trading ha dimostrato come, considerando un insieme ridotto delle possibili variazioni dei parametri, le configurazioni profittevoli si distribuiscono nelle code di una gaussiana, a dimostrazione della non stocasticità del mercato. Di contro, l'algoritmo per le operazioni OCO, testato su conto demo per cinque settimane di trading, su cinque strumenti finanziari e quattro timeframe differenti, mostra come tale configurazione non è particolarmente profittevole su mercati particolarmente volatili, ma comunque necessita di un certo grado di direzionalità.

Gli sviluppi futuri prevedono sicuramente l'integrazione all'interno dell'infrastruttura software di un modulo di gestione ed armonizzazione del portafoglio titoli; per quanto riguarda gli algoritmi di trading, vi è la necessità di analizzare in maniera più approfondita i risultati ottenuti, al fine di migliorarne le performance ed eventualmente individuare quali sono gli elementi della modellazione non ancora maturi. Uno step successivo potrebbe essere quello di considerare gli algoritmi come agenti all'interno di una soluzione complessa, nella quale tali entità cooperano e competono, perseguendo un fine comune, ovvero rendere profittevole in qualunque condizione di mercato la strategia.

Bibliografia

[1] Markets.com, «Analisi Tecnica,» [Online]. Available: http://www.markets.com/it/education/technical-analysis/.

[2] MetaQuotes Software Corp., «Technical Indicators,» [Online]. Available: http://ta.mql4.com/indicators.

[3] ForexGuida, «Forex Trading & Analisi dei Mercati,» [Online]. Available: http://www.forexguida.com/.

[4] R. Colby, The Encyclopedia of Technical Market Indicators, McGraw-Hill, 2002.

[5] B. Williams, New Trading Dimensions: How to Profit from Chaos in Stocks, Bonds and Commodities, Wiley, 1998.

[6] B. Williams, Trading Chaos: Applying Expert Techniques to Maximize your Profits, Wiley, 1995.

[7] M. Thomsett, «CMF Chaikin Money Flow: Changes Anticipating Price Reversal,» *Financial Times Press Delivers,* 2010.

[8] J. W. Wilder, New Concepts in Technical Trading Systems, Trend Research, 1978.

[9] J. Bollinger, Il trading con le bande di Bollinger, Trading Library, 2003.

[10] G. Appel e E. Dobson, Understanding MACD (Moving Average Convergence Divergence), Traders Press Inc., 2008.

[11] G. Lane, «Lane's Stochastic,» *Technical Analysis of Stocks and Commodities magazine,* pp. 87-90, 1984.

[12] H. Simon, The Sciences of the Artificial - 3rd Edition, The MIT Press, 1996.

[13] H. Simon e A. Newell, Human Problem Solving, Englewood Cliffs, NJ: Prentice-Hall, 1972.

[14] J. Donovan e S. Madnick , Institutional and Ad Hoc DSS and Their Effective Use, 1977.

[15] S. Alter, Decision Support Systems: Current Practice and Continuing Challenge, Addison-Wesley, 1980.

[16] R. Hackathorn e P. Keen, Organizational Strategies for Personal Computing in Decision Support Systems, MIS Quarterly, 1981.

[17] R. Sprague e E. Carlson, Building Effective Decision Support Systems, Prentice-Hall, 1982.

[18] R. Sprague, A Framework for the Development of Decision Support Systems, MIS Quarterly, 1980.

[19] P. Haettenschwiler, «Neues anwenderfreundliches Konzept der Entscheidungsunterstützung. Gutes Entscheiden in Wirtschaft, Politik und Gesellschaft.,» *vdf Hochschulverlag AG,* pp. 189-208, 1999.

[20] D. Power, Decision Support Systems: Concepts and Resources for Managers, Quorum Books, 2002.

[21] A. Belli, «Applicazioni dei sistemi di supporto alle decisioni manageriali in un ente fortemente decentrato : sviluppo di un prototipo per la valutazione di nuovi progetti,» *ENEA,* 1998.

[22] Wikipedia, «Decision Support System,» [Online]. Available: https://it.wikipedia.org/wiki/Decision_support_system.

[23] B. Chiandotto, *Teoria delle Decisioni,* 2006.

[24] Statix - Metodi quantitativi d'impresa, «Analisi Decisionali,» [Online]. Available: http://www.statix.ch/Analisi%20decisionali.htm.

[25] A. Agarwal, «High Frequency Trading: Evolution and the Future.,» *Capital Markets,* 2012.

[26] Wikipedia, «Colocation Center,» [Online]. Available: https://en.wikipedia.org/wiki/Colocation_centre.

[27] J. Harris e T. Davenport, «Automated Decision Making Comes of Age,» Accenture, 2005.

[28] Engle R. F., «Autoregressive conditional heteroskedasticity with estimates of the U.K. inflation,» *Econometrica,* n. 50, pp. 987-1008, 1982.

[29] Mandelbrot B.B., «The variation of certain speculative prices,» *Journal of Business,* n. 36, pp. 394-419, 1963.

[30] Fama E. F., «The behavior of stock market prices,» *Journal of Business,* n. 38, pp. 34-105, 1965.

[31] T. Bollerslev, R. Engle e D. Nelson, «ARCH Models,» *Handbook of Econometrics,* vol. 4, 1994.

[32] R. Engle e T. Bollerslev, «Modelling the persistence of Conditional Variances,» *Econometric Reviews,* n. 5, pp. 1-50, 1986.

[33] L. Calvet, A. Fisher e B. Mandelbrot, «A multifractal model of asset returns,» *Cowles Foundation Discussion Papers,* n. 1164-1166, 1997.

[34] R. Quandt, «The estimation of the parameters of a linear regression system obeying two separate regimes,» *Journal of the american statistical association,* vol. 53, n. 284, pp. 873-880, 1958.

[35] S. Goldfeld e R. Quandt, «A Markov Model for Switching Regressions,» *Journal of Econometrics,* n. 1, pp. 3-16, 1973.

[36] T. Kohonen, Self-organization and associative memory, Germany: Springer, 1984.

[37] S. Haykin, Neural networks - A comprehensive foundation, New York: McMillan College Publishing, 1988.

[38] A. Bahrammirzaee, «A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems,» *Neural Comput & Applic,* vol. 19, pp. 1165-1195, 2010.

[39] M. Lam, «Neural network techniques for financial performance prediction, integrating fundamental and technical analysis,» *Decision Support Systems,* vol. 37, p. 567–581, 2004.

[40] C. Hsieh, «Some potential applications of artificial neural systems in financial management,» *J Syst Manage,* vol. 44, n. 4, pp. 12-16, 1993.

[41] L. Medsker, E. Turban e R. Trippi, «Neural network fundamentals for financial analysts,» in *Neural networks in finance and investing*, Irwin, 1996, pp. 329-365.

[42] H. White, «Economic prediction using neural networks: the case of IBM daily stock returns,» in *IEEE International Conference on Neural Networks*, 1988.

[43] C. Panda e V. Narasimhan, «Forecasting exchange rate better with artificial neural network,» *J Policy Model,* vol. 29, pp. 227-236, 2007.

[44] P. Harmon e D. King, Artificial intelligence in business–expert systems, Wiley, 1985.

[45] P. Jackson, Introduction to expert systems, WokinghaM: Addison-Wesley, 1986.

[46] K. Lee, *An expert real-time trading system,* Master Thesis - Univ. of Nevada, 1998.

[47] Y. Kwon e B. Moon, «A Hybrid Neurogenetic Approach for Stock Forecasting,» *IEEE Transactions on Neural Networks,* vol. 18, n. 3, pp. 851-864, 2007.

[48] M. Khadka, B. Popp, K. George e N. Park, «A New Approach for Time Series Forecasting based on Genetic Algorithm,» in *23rd International Conference on Computer Applications in Industry and Engineering*, 2010.

[49] MetaQuotes Software Corp., «Abouts Us - MetaQuote Software,» [Online]. Available: http://www.metaquotes.net/en/company.

[50] MetaQuotes Software Corp., «Automated Trading with MetaTrader 4,» [Online]. Available: http://www.metaquotes.net/en/metatrader4/automated_trading.

[51] D. Lambert, «Commodities Channel Index: Tools for Trading Cyclical Trends,» *Commodities Magazine,* 1980.

[52] J. Granville, Granville's New Strategy of Daily Stock Market Timing for Maximum Profit, Prentice-Hall, 1976.

[53] A. Elder, Trading for a Living: Psychology, Trading Tactics, Money Management, Wiley Finance Edition, 1993.

[54] T. DeMark e J. Perl, DeMarker Indicators, Wiley, 2010.

[55] B. T., «Generalized autoregressive conditional heteroskedasticity,» *Journal of Econometrics,* n. 31, pp. 307-327, 1986.

[56] T. Lux e L. Morales-Arias, «Forecasting volatility under fractality, regime-switching, long memory and student-t innovations,» *Computational Statistics and Data Analysis,* vol. 54, pp. 2676-2692, 2010.