Liceo Scientifico e Classico

“**G. Peano – S. Pellico”**

*Corso Giovanni Giolitti 11*

*Via Monte Zovetto 8*

**STAGE DI ALTERNANZA SCUOLA – LAVORO**

**Bottero S.p.A.**

**Dati tirocinio:**

* Data di inizio: 02/07/2018
* Data di conclusione: 03/07/2018
* Centro di costo: XC714
* Tutor aziendale: Ing. Galliano Fabio
* Tutor operativo: Ing. Mazza Stefano
* Tipologia progetto formativo: analisi e test di un modulo SW (C#, .NET), nell’ambito del progetto Plunger Motion Control
* Reparto: UR & SVC / Automazione, ricerca e sviluppo Vetro Cavo
* Ambiente di sviluppo utilizzato: Microsoft Visual Studio 2015 Professional

**Rapporto stage:**

* **Descrizione del funzionamento dell’oggetto dello stage e problematica affrontata:**

Per lo svolgimento di questo stage è stato proposto di lavorare sul progetto Plunger Motion Control (PMC): un progetto atto a supervisionare il movimento di un maschio all’interno dello stampo abbozzatore di una sezione di una macchina IS (independent sections) progettata e realizzata da Bottero per la formatura di articoli in vetro cavo. Tali macchinari, come suggerisce il nome sono composti da sezioni tra loro indipendenti controllate e coordinate da un sistema centrale. Ogni sezione è in grado di trasformare una o più (a seconda del numero di cavità dello stampo che varia da una a quattro) gocce di vetro fuso (alla temperatura di 1200 [°C]) in un contenitore cavo come un barattolo o una bottiglia. Esistono vari metodi per la formatura di tali contenitori ma quello analizzato durante il tirocinio è stato quello definito come “Press and blow”. Questo processo consiste in due fasi principali: una di pressata e un’altra di soffiata. Questi due step vengono eseguiti rispettivamente sul lato abbozzatore e finitore della sezione. Di seguito una breve descrizione delle due fasi:

* **Abbozzatura:** Il maschio (plunger) è situato al di sotto dello stampo stesso ed è mantenuto in posizione abbassata (posizione 0) tramite l’impiego di una pressione pneumatica (Pressione Down). Con l’annullamento della pressione Down, il maschio, spinto da una molla, sale fino alla posizione di attesa della goccia di vetro fuso, entrando nel canale aperto sul fondo dello stampo abbozzatore e attraversando il fondello del servo meccanismo Invert, la cui funzione sarà esplicitata in seguito. Dopo questa fase la sezione è pronta per ricevere una goccia, tagliata in dimensioni appropriate in funzione del tipo di contenitore da produrre. Una volta che la goccia è caduta nello stampo chiuso (blank mould), guidata da un imbuto (funnel), l’apertura superiore dello stesso viene sigillata dal tampone (baffle). Ora il maschio, spinto da una pressione di Up, sale fino alla sua posizione massima formando in questo modo l’imboccatura e il collo dell’articolo. Ora il plunger rimane fermo in tale posizione per un certo intervallo di tempo, che può essere scelto dall’utilizzatore del macchinario finito. Questo periodo di attesa in posizione è definito pressata (Dwell). La pressata serve ad imprimere la forma sbozzata del contenitore nel vetro in lento raffreddamento, per ottenere un risultato più omogeneo, nel prodotto finito. Il calcolo del Dwell time (ovvero il tempo di durata della pressata) è stata la questione centrale di questo stage. Trascorso il tempo di pressata, il maschio torna nella sua posizione 0, quella iniziale e di minima elevazione, prima con assenza di pressione Up e infine con applicazione di pressione Down. A questo punto gli stampi vengono aperti e il meccanismo Invert (ancorato al collo dell’oggetto in formazione) trasporta l’abbozzo (parison) all’interno dello stampo finitore aperto.
* **Finitura:** una volta che la parison è in posizione, lo stampo viene chiuso ed inizia il processo di soffiatura. Dal basso lo stampo è sigillato da un fondello, su cui è impressa la zigrinatura del fondo bottiglia e informazioni riguardanti lo stampo e la sezione, mentre dall’alto viene calata la testa soffiante. Dopo un riscaldamento del vetro ancora incandescente viene immesso nella cavità della parison un getto di aria compressa, che gonfia l’abbozzo il quale a sua volta acquisisce le dimensioni della cavità dello stampo. In seguito, la testa soffiante si alza, lo stampo viene aperto e un servo meccanismo (take out) dotato di pinze preleva l’oggetto finito e lo posa su una piastra morta (dead plate) dove l’articolo viene raffreddato da un getto d’aria dal basso. Infine un meccanismo servo Pusher si occupa di spostare il contenitore formato su un nastro trasportatore che condurrà tutti gli articoli formati dalle sezioni della stessa linea (linea di produzione di una macchina IS) e da più linee ai convogli dei forni di ricottura, controllo qualità e pallettizzazione.

I meccanismi menzionati (ad esclusione dei servo meccanismi) sono ad azionamento pneumatico, questo per ovviare alla difficoltà di produrre meccanismi elettrici o elettronici in grado di resistere alle stressanti e poco igieniche condizioni di lavorazione del vetro (alte temperature, sporco…). Meccanismi pneumatici implicano la presenza di valvole, le quali sono di due tipologie: valvole standard ON – OFF e valvole proporzionali controllate a loop chiuso (con un sensore di pressione interno per una maggiore precisione e autocorrezione) o aperto in grado di modulare un flusso di aria per mantenere o raggiungere una pressione impostata.

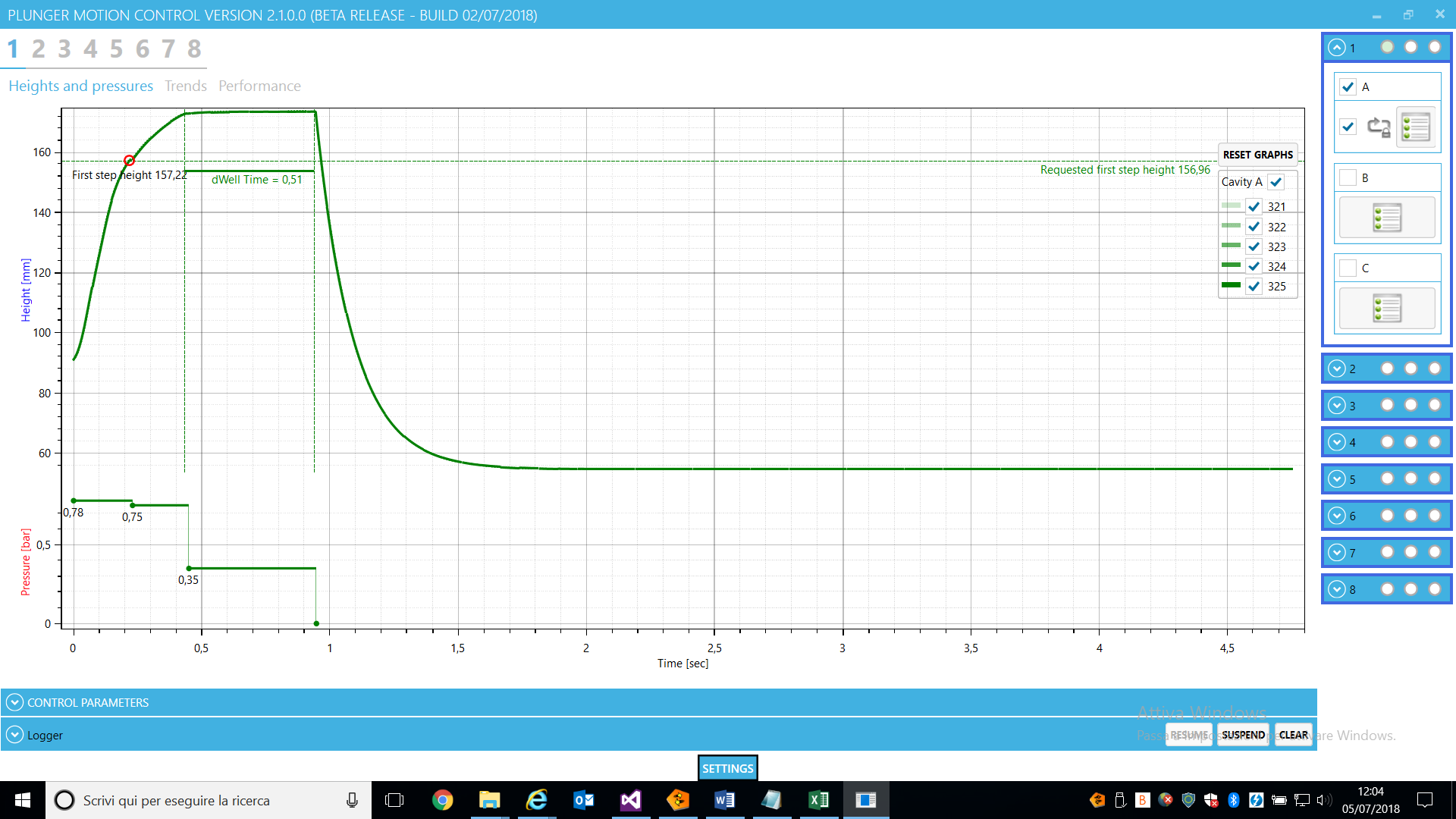
Ovviamente una macchina di una complessità elevata come quella di una macchina IS, presenta molti più meccanismi e componenti rispetto a quelli citati in queso rapporto, ognuno con le proprie caratteristiche e funzionalità ed ognuno gestito da un apposito controllo software.

I sistemi di controllo di una macchina IS, anch’essi progettati e prodotti da Bottero, si compongono di un software che tramite uno schermo e un’interfaccia grafica interloquiscono con l’utente. Essi si occupano del controllo e del dialogo dei componenti della macchina dal feeder, ovvero il sistema di alimentazione della macchina con gocce di vetro, fino all’ultima fase della formatura dell’articolo. Tali controlli sono:

* ELECTRONIC TIMING CONTROL: per la gestione della formatura dell’articolo per ogni sezione;
* FLEXIBLE MOTOR CONTROL: per la gestione dei motori esterni alla sezione;
* SERVO MECHANISM CONTROL: per il controllo dei servo meccanismi;
* ELECTRONIC PUSHER SYSTEM: per la gestione dei pusher.

Il lavoro affrontato durante lo stage è stato come annunciato la supervisione del controllo dei plunger del lato abbozzatore della macchina, meccanismo controllato dall’Electronic Timing Control. Questo sistema di controllo, come anche gli altri, è comandato da un PLC, ovvero l’equivalente di una CPU ma in grado di eseguire sistemi operativi e software real – time. Per software real – time si intende un programma tale per cui è noto il tempo di esecuzione di un dato processo e tale tempo è sempre costante: requisito essenziale per il controllo di un sistema automatizzato. Oltre al PLC, sono presenti schede di I/O, cioè di input e output con la macchina, che sono alloggiate in appositi zoccoli. Queste componenti sono in grado di comunicare con la macchina utilizzando opportune variabili valorizzate sia in lettura che in scrittura, durante il dialogo con la sezione. Sempre all’interno del sistema di controllo sono presenti trasformatori per l’alimentazione elettrica e relè di emergenza per disabilitare la sezione in caso di necessità. Tali interrutorri di emergenza sono attivabili dall’esterno del case di controllo o dalla plancia comandi della sezione. Un nuovo progetto è quello sostituire questi relè con PLC safe, in grado di svolgere la stessa funzione ma in maniera più controllabile. Avvicinandoci al controllo del plunger, come detto il sistema pneumatico del plunger è gestito da valvole, a loro volta controllate dal Electronic Timing Control; tali valvole sono appunto controllate a loop chiuso, cioè autocorrettive per garantire la costanza della pressione da erogare grazie a un sensore di pressione, o a loop aperto, in questo caso il sistema di controllo indica l’apertura della farfalla (in base ad una taratura effettuata prima del motaggio) senza effettuare misurazioni sulla pressione effettivamente erogata. Queste valvole proporzionali sono controllate in modo analogico: tramite valori PWM che tentano di approssimare una conversione da digitale ad analogico tramite l’emissione di impulsi di segnale a frequenze proporzionali al valore che si vuole trasmettere. Per modulare tali frequenza di impulsi viene usato un valore numerico compreso tra 0 e 255 o tra 0 e 1000. Ogni valvola proporzionale va opportunatamente tarata in modo da rendere possibile una conversione tra bar erogati e valore PWM affinché nella scrittura del software di controllo si possa esprimere semplicemente il valore di bar da erogare e non il valore PWM (che sarà calcolato automaticamente grazie alla taratura effettuata). La taratura si effettua trovando i valori corrispondenti di PWM per 0,5 bar e 3 bar per le valvole 1.0, e tra 0 e 4 bar per le 2.0; i punti corrispondenti vengono poi approssimati per i valori intermedi con una proporzionalità lineare. Le valvole montate sulla sezione osservata in laboratorio[[1]](#footnote-2) sono del tipo 1.0 cioè a open loop, senza capacità di autocorrezione e inoltre necessitano di taratura a banco per ogni valvola prodotta. Le valvole 2.0 a loop chiuso mantengono automaticamente la pressione richiesta e dunque non è necessario tarare ogni singola valvola prodotta.

Il progetto PMC offre all’utente una interfaccia grafica da cui si può scegliere quale linea e quale sezione di produzione controllare. L’applicativo presenta inoltre un’area grafico in cui è raffigurato il movimento del maschio come variazione di altezza in funzione del tempo insieme alla pressione erogata al pistone. Nell’immagine che segue si vede il grafico riportato ottenuto con una simulazione di misurazione.



Fase di pressata

Dwell time

Seconda fase di salita maschio

Discesa finale maschio

Pressione nel pistone

Fine fase di attesa goccia

Come si può notare, nel grafico simulato manca il primo step di salita del maschio, ma questo al fine della mia “mansione” non rappresenta un problema poiché mi è stato assegnato un lavoro sul dwell time.

* **Attività di stage:**

Dopo l’accoglienza all’azienda presso l’Ufficio del Personale da parte della Dott.ssa Revelli Romina, la quale mi ha illustrato il regolamento aziendale, sono stato inserito nello stage e dopo aver ottenuto la strumentazione per svolgere la mansione assegnatami[[2]](#footnote-3) mi è stato presentato il seguente piano di lavoro:

*Nell’ambito del progetto* Plunger Motion Control*:*

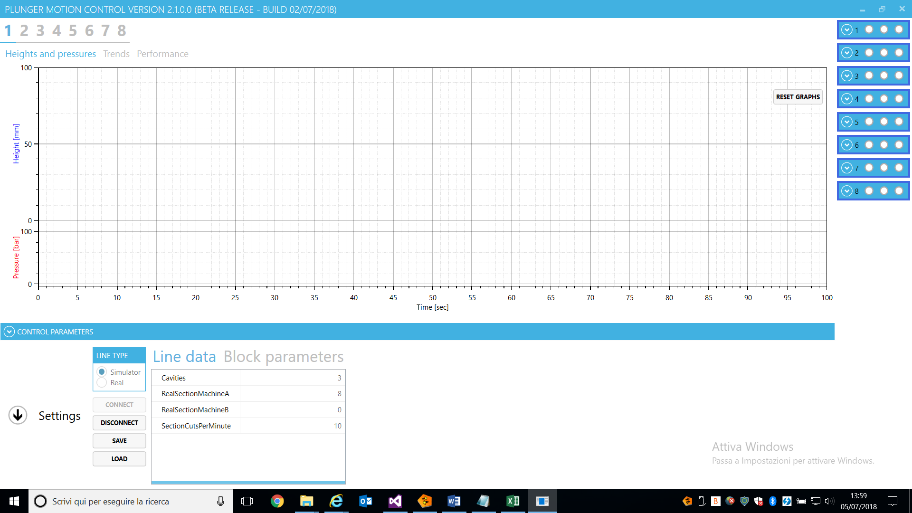
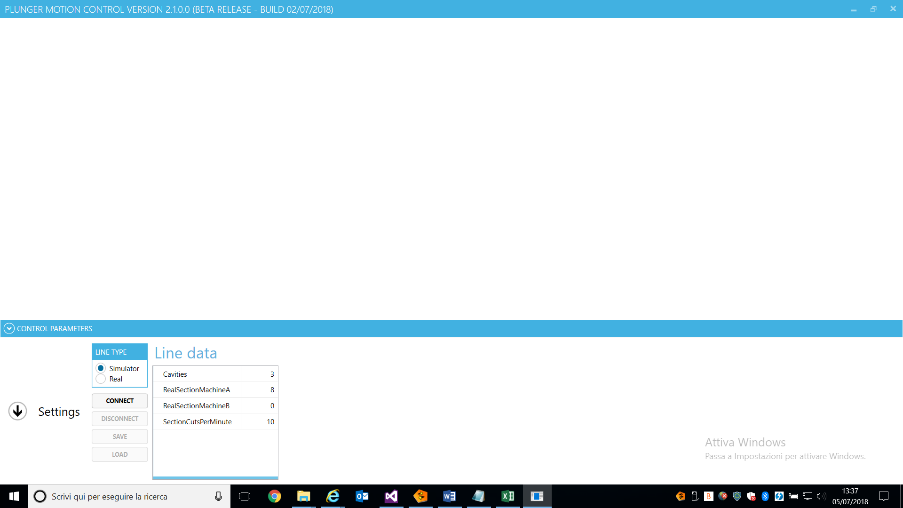
* *STEP 1: Analizzare nel dettaglio il file* PlungerCycleInfo.cs *contenente la classe* PlungerCycleInfo *che si occupa di caratterizzare la curva di movimento del plunger:*
  + *Familiarizzare con le funzionalità ed i principi base del programma “*Plunger Motion Control*”*
  + *Eseguire il programma “*Plunger Motion Control*” in simulazione e debuggare le funzioni della classe* PlungerCycleInfo
  + *Scrivere un breve rapportino che descriva tutte le funzioni della classe e la logica dell’algoritmo*
* *STEP 2: Cambiare la logica del calcolo del* DwellTime*: Mantenendo in prima battuta la logica attuale, una volta che l’algoritmo identifica i due punti di inizio e fine pressata, calcolare il valore medio dei campioni all’interno del range di pressata identificato precedentemente e ricalcolare nuovamente i due punti di inizio/fine pressata con la seguente logica:*
  + *Il punto di inizio è l’altezza del plunger corrispondente ad un valore uguale o superiore al 99% della differenza tra la media (calcolata precedentemente) ed il valore iniziale di pressata*
  + *Il punto finale è l’altezza del plunger corrispondente ad un valore uguale o inferiore al 99% della differenza tra la media (calcolata precedentemente) ed il valore iniziale di pressata*
* *STEP 3: Provare la modifica prima con il simulatore e poi in laboratorio dove andrà fatto un confronto tra i risultati tra il vecchio ed il nuovo algoritmo.*
* *STEP 4: Scrivere un breve rapportino sui risultati ottenuti*
* *STEP 5: (OPZIONALE) Pensare ad una strategia ancora più “furba” per fare il conto del tempo di pressata in maniera ancora più affidabile e meno sensibile ad eventuale rumore e/o disturbi sui valori dell’altezza del plunger.*
* **Svolgimento dell’attività proposta:**

Nei seguenti paragrafi verrà esplicitata l’attività effettivamente svolta:

* STEP 1:

Il file PlungerCycleInfo.cs contiene appunto la classe definita come public class PlungerCycleInfo che definisce un oggetto in grado di memorizzare i dati relativi ad un ciclo di movimento del plunger per poi caratterizzare la curva di movimento che si può vedere sul grafico.

* Il programma “Plunger Motion Control” permette di visualizzare in maniera intuitiva la posizione del maschio durante un ciclo di lavoro. All’avvio del programma è possibile scegliere se connettere il software an una linea reale, specificando indirizzo IP (quello del laboratorio utilizzato era 10.0.0.196) oppure lanciare il programma in modalità simulazione; quest’ultima è la modalità che è stata maggiormente utilizzata ai fini di test del codice. Una volta scelta la modalità desiderata è necessario inserire le caratteristiche della macchina quali numero di sezioni, numero di cavità per sezione, numero di gocce al minuto e altri parametri. Fatto ciò si può connettere il programma alla macchina (anche virtuale): apparirà la schermata di controllo e monitoraggio. La schermata è così caratterizzata: in centro è presente l’area dove verrà stampato il grafico di altezze e pressioni, a destra sono presenti palettes da cui è possibile scegliere quale sezione o cavità monitorare e scegliere se impostare o meno il controllo sulla first step height, in alto a sinistra è disponibile una scelta di quale sezione visualizzare e infine in basso a sinistra e nella parte bassa dello schermo si possono visualizzare e impostare vari parametri riguardo la sezione desiderata. Inoltre tramite l’utilizzo del mouse è possibile visualizzare i valori corrispondenti ad un determinato punto del grafico e muovere la stessa vista grafico. Durante questa fase si è osservata la variazione dei dati eseguendo più cicli simulati, con e senza controllo sulla first step height. Di seguito rispettivamente le schermate di avvio e monitoraggio del software:



* Durante la seconda fase è stato eseguito lo stesso programma in modalità simulazione e compilandolo come debug per poterlo eseguire passo passo e comprendere meglio il funzionamento della classe PlungerCycleInfo. L’analisi della classe è stata fatta per ogni funzione, leggendo e commentando con il significato compreso, la maggior parte delle righe del codice. Gradualmente si è eseguito il programma più volte in debug posizionando break point in opportune parti del codice per procedere con l’esecuzione passo passo e per visualizzare il valore corrente delle variabili utilizzate mettendole sotto watch: una comoda funzione dell’ambiente di sviluppo utilizzato che consente di visualizzare il valore attuale di una variabile e anche di effettuarci operazioni come copiarlo. In questo modo si sono appresi i funzionamenti dei metodi e il significato di attributi e variabili membro della classe.
* Una volta compreso il funzionamento della classe nella sua totalità, si è cominciato a scriverne la documentazione inizialmente in italiano e successivamente tradotta in lingua inglese per mantenere uniformità con le altre parti della documentazione già scritte. Una osservazione fatta può essere la seguente: la classe è caratterizzata da molti metodi privati, dunque non richiamabili dall’esterno, lo stesso costruttore è definito privato ed infatti l’unico modo per richiamarlo è quello di invocare il metodo statico “public static IPlungerCycleInfo FromPlungerMotionData(int cicleNumber, IPressureProfile pressureProfile, IEnumerable<double> times, IEnumerable<double> heights, bool validCycle = true, double profileStartPhase = double.NaN, double tDwellStartPrev = 0.0, double tDwellEndPrev = 0.0)” passando come parametri i dati del ciclo appena svolto. Tale metodo restituisce però un’interfaccia IPlungerCycleInfo. Diversamente, molti attributi caratterizzanti il ciclo sono pubblici. Nell’analisi della classe si sono notati casi di overload di metodi (ad esempio lo stesso costruttore) e strutture di dati quali List, IEnumerable, Queue non ancora note. Infine evidente la divisione del codice in macroregioni, definite dalle parole #region ed #endregion seguite dal nome assegnato alla regione di codice.
* STEP 2:

Per questa fase si è effettuato un overload del metodo “private double CalculateDwellTime(List<double> X, List<double> Y)”, variando i parametri passati alla funzione: nella mia riscrittura si è trattato di due oggetti IEnumerable. La firma del metodo nuovo è la seguente: “private double CalculateDwellTime(IEnumerable<double> times, IEnumerable<double> heights)”. Oltre a questo metodo ne sono stati scritti altri due. Uno per il calcolo degli estremi della porzione di vettore contenete i dati riguardanti il dwell time, data una soglia di sensibilità sull’altezza relativa del maschio: “private List<int> FindDwellBorders(IEnumerable<double> heights, double hThreshold, double tollerance = \_hConstRelativeTolerance)”; l’altro per il calcolo della media ponderata di un array di valori, passando oltre alla sequenza di valori i pesi da assegnare alle due parti in cui viene diviso l’array e la dimensione relativa della prima parte dell’array: “private double WeightedAverage(List<double> values, double weight1, double weight2, double whereToSplit)”. Il valore della media sia ponderata che semplice sono stati utilizzati per il test sull’array di valori altezza per verificare la condizione di identificazione della pressata enunciate nella proposta di lavoro sopra riportata. La scrittura di questo codice è stata fatta prendendo spunto dall’algoritmo di calcolo già presente nella classe.

* STEP 3:

In questa fase si è effettuato il test dei metodi scritti alla fase precedente, eseguendo l’applicativo sempre in modalità debug. Per effettuare un controllo più evidente anche in run – time, è stato creato ulteriormente un metodo “public double TestDwellTimeAlgorithms(int cycleNumber, List<double> X, List<double> Y, IEnumerable<double> times, IEnumerable<double> heights)” in grado di eseguire alternativamente il nuovo ed il vecchio algoritmo CalculateDwellTime eseguendo uno switch sul numero del ciclo (pari o dispari). Come ci si poteva aspettare il dwell time calcolato con il nuovo algoritmo è maggiore rispetto a quello calcolato con il vecchio, durante il test in simulazione; mentre entrambi i valori assumevano lo stesso valore nella prova in laboratorio[[3]](#footnote-4). I test sono stati fatti inserendo o meno il controllo sul dwell time e utilizzando oppure no la media ponderata calcolata con il mio algoritmo. Per migliorare la raccolta dai dati di dwell time si è deciso di far eseguire ad ogni ciclo entrambi gli algoritmi e salvare su List e poi su file (utilizzando le classi FileStream e StreamWriter) i dati di dwell time con i due algoritmi e la relativa varianza (calcolata su consiglio del tutor Mazza).

* STEP 4:

I dati ricavati dal programma sono stati memorizzati nei file di testo generati dallo stesso oppure registrati su fogli di calcolo Excel per essere elaborati. Come già detto i dwell time calcolati con l’algoritmo nuovo sono più ampi rispetto a quelli calcolati con quello vecchio, il motivo sta nella scelta dalla soglia di valore del criterio con cui si individua la pressata: nell’algoritmo vecchio si definiva pressata il luogo in cui l’altezza del maschio era superiore al 99% dell’altezza massima raggiunta nella sua corsa, ma l’altezza massima poteva essere registrata in seguito a eventi imprevedibili come rumori o innalzamento extra del maschio dovuto al raffreddamento del vetro. Per evitare questo errore, si può appunto prendere un valore medio, valutando poco eventuali valori anomali o prossimi alla fine della pressata (come nel caso della media ponderata). Tale valore medio è comunque inferiore rispetto al valore massimo e dunque i valori superiori al 99% del valore medio sono di più rispetto a quelli riferiti al valore massimo, questo è il motivo della lettura di un dwell time maggiore. Inoltre, sempre relativamente alla prova simulata, la varianza sui dati ottenuti con il vecchio algoritmo è maggiore rispetto a quella calcolata sui valori del nuovo metodo nuovamente per via della soglia scelta: prendendo la massima elevazione, la soglia a cui si riferisce il calcolo del dwell time può oscillare in un intervallo più ampio rispetto alla soglia su valore medio e dunque il dwell time può oscillare in un range più ampio di quanto possa fare con una soglia più bassa e meno soggetta a variazioni come un valore medio, da qui deriva la maggiore varianza del dwell time calcolato con l’algoritmo vecchio (si veda il file Excel Confronto algoritmi.xlsx). Diversamente è andata la prova in laboratorio: non essendo ovviamente presente il rumore generato casualmente dal simulatore ma solo quello rilevato dalla macchina, i valori di dwell time erano identici per entrambi gli algoritmi, prova della robustezza del nuovo algoritmo. Il fatto che fossero identici per tutte le cifre decimali del dato è dovuto alla conversione in tempo dell’avanzamento del ciclo espresso internamente dal controllo in gradi[[4]](#footnote-5). Sui file Excel generati (Confronto algoritmi laboratorio.xlsx) è possibile vedere in modo chiaro i risultati di tutti i test, con e senza controllo sul dwell time.

* STEP 5:

In questa fase è stata sviluppata una nuova strategia di calcolo del dwell time. Per farlo si è scelto di effettuare una interpolazione della retta che approssima meglio i dati di altezza e tempo nella ipotetica fase di pressata individuata con l’algoritmo sviluppato allo STEP 2. Inizialmente, si è detto che il dwell time è il lasso di tempo che intercorre tra l’istante in cui la linea di tendenza supera la soglia del 99,5% sull’altezza media del maschio nell’ipotetica fase di pressata ricavata e l’istante in cui la differenza di valori tra la retta e la curva del maschio supera l’0.01% dell’altezza totale del plunger. Le funzionalità aggiuntive per l’interpolazione della retta sono state create utilizzando nuove classi: public abstract class Equation, public class LinearEquation : Equation e public class LineInterpolator.

Con il primo arrangiamento la soluzione ha restituito valori errati sul dwell time (circa il doppio rispetto a quelli calcolati con l’algoritmo tradizionale). Dopo aver apportato modifiche al sorgente e aver eseguito in modalità debug il programma, confrontando i risultati del nuovissimo algoritmo con quelli del vecchio, è stato finalmente possibile vederne una somiglianza. Tali risultati sono stati registrati dal programma in modalità simulazione e caricati su file Excel per un’analisi dettagliata.

Dai grafici è risultato innanzitutto che il valore nuovissimo tende ad essere più basso rispetto a quello del vecchio algoritmo, mentre sulla varianza dei due risultati non si riscontrano particolari differenze.

In seguito al test in laboratorio, l’algoritmo è stato ulteriormente modificato: al posto di una condizione sulla differenza dal valore medio, si è posto una condizione sulla differenza con il valore misurato dalla macchina. Questa correzione è stata apportata in seguito ai risultati evidentemente errati durante il test in laboratorio. Infatti, se si fosse continuato a considerare il valore medio dell’altezza come soglia (graficamente un retta costante), più l’altezza nel maschio si fosse avvicinata al caso ideale (cioè al ragguingimento di un valore costante), più il dwell time sarebbe stato calcolato con errore: il metodo scitto sarebbe andato a calcolare un punto molto vicino all’intersezione di due rette praticamente parallele, generando così valori di dwell time non accettabili. Con la correzione proposta, ci si è leggermente allontanati dalla richiesta del punto 5 a favore di una maggiore accettabilità dei risultati sul dwell time. Il confronto tra i tre algoritmi di calcolo del dwell time ora presenti si può vedere sul file Confronto algoritmi (vecchio e nuovo opzionale) laboratorio.xlsx

**Nuove nozioni apprese:**

Questa esperienza di stage si è rivelata molto utile ed interessante per le nozioni in settore tecnico – tecnologico apprese. Innanzitutto ho potuto conoscere in maniera molto approfondita il processo di formazione degli articoli in vetro seguito dalle macchine Bottero. Successivamente ho imparato terminologia tecnica circa i vari componenti di una macchina IS e fasi della lavorazione del vetro. Fondamentale infine le competenze informatiche consolidate e arricchite grazie al confronto con il linguaggio Visual C#. Di seguito un elenco delle nozioni incontrate:

* Concetto di Solution in una serie di progetti C#;
* Concetto di NuGet;
* Differenza tra compilazione di debug e release;
* Concetto di namespace;
* Struttura getters e setters in C#;
* Strutture dati dinamiche Queue, List, IEnumerable;
* Variabili di tipo generico var;
* Lancio di eccezioni in C#;
* Utilizzo di notazione compatta per le selezioni;
* Extension di una classe in C#;
* Definizione di regioni di codice tramite parola #region ed #endregion;
* Scrittura di file su disco con classi FileStream e StreamWriter;
* Struttura iterativa foreach;
* Import di classi da altri namespace utilizzando parola chiave using;
* Creazione di nuove classi in nuovi namespaces.

1. Nel laboratorio del reparto di ricerca e sviluppo del vetro cavo sono presenti parti di sezioni montate disponibili per il test dei software di controllo. I componenti della sezione sono separati in servo pusher, scatola meccanismi e teste soffianti. [↑](#footnote-ref-2)
2. Mi è stato assegnato un PC dotato di alimentatore e mouse. È stato fornito il progetto Plunger Motion Control e si è completata l’installazione dei pacchetti per la compilazione del progetto (tra cui librerie e pacchetti NuGet). [↑](#footnote-ref-3)
3. Nel test di laboratorio è stato anche possibile vedere una imperfezione nella salita del maschio causata a sua volta da un’imperfezione nel montaggio del meccanismo Invert nella sezione di prova. Questo difetto causava un contatto inatteso tra il maschio e il collarino posto sull’Invert. [↑](#footnote-ref-4)
4. Si tratta di misurare l’avanzamento di un processo con un valore tra 0 e 360 [°]. Quando, prima del controllo elettronico dei macchinari, si utilizzava un carillon per mettere in moto una macchina, il grado di rotazione del carillon identificava l’avanzamento del processo ciclico che la macchina stava eseguendo. [↑](#footnote-ref-5)