POLITECNICO DI MILANO
Facoltà di Ingegneria dell'Informazione
Dipartimento di Elettronica e Informazione

Corso di Laurea Specialistica in Ingegneria Informatica

Interfaccia comando motore con controllo PID realizzato con Arduino

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Andrea Bonarini

Tesi di Laurea di
Andrea Campana
Matricola n. 682380

Anno Accademico 2013/2014

Dedicato a mio fratello,
ormai sono passati tanti anni
ma mi manchi ancora …
Ciao Massimo

Indice

Indice delle figure

Sommario

Il controllo automatico è ormai ampiamente diffuso e utilizzato in diversi campi e ambiti, serve soprattutto a verificare che un dato sistema si stia comportando come previsto, e restituisca in uscita valori coerenti con i parametri e dati di ingresso.

Un noto esempio di controllo automatico è quello denominato PID [9] [10], dove P significa “Proporzionale”, I indica “Integrale” e infine D per “Derivativo”, questi sono le 3 parti necessarie a ottenere un controllo retro azionato, il quale riduce automaticamente l’errore tra il valore di uscita atteso rispetto a quello effettivamente ottenuto.

Vista la semplicità di realizzazione, risulta essere una delle soluzioni più diffuse e utilizzate in svariati ambiti, non solo per il controllo motore, dato che consente di ottenere risultati molto soddisfacenti.

Quello che viene proposto in questo progetto è la realizzazione di un interfaccia di comando con cui gestire un sistema di controllo automatico PID, per poter variare i parametri del controllo e valutare i risultati ottenuti.

Questo sistema di controllo verrà implementato e gestito all’interno di una scheda Arduino [1], alla quale sarà connesso un servo motore elettrico che si muove di conseguenza alle elaborazioni effettuate dallo stessa scheda Arduino, in base ai comandi e parametri ricevuti dall’interfaccia, questa realizzata tramite Processing.

1 Capitolo 1

1.1 Introduzione

Il progetto si suddivide nell'implementazione di 2 elementi applicativi, il primo consiste nel modellare sistema di controllo PID che verrà sviluppato in ambiente Arduino, creando uno sketch da caricare su una scheda, il secondo elemento prevede la realizzazione un'interfaccia grafica di comando utilizzata tramite Computer, generata mediante l'ambiente di sviluppo Processing.

Per la realizzazione di questi elementi del progetto è stato necessario apprendere prima il funzionamento dei vari ambienti di sviluppo, delle librerie di supporto e utilizzabili per il progetto, fino ad arrivare a implementare gli sketch necessari al completamento del progetto.

1.2 Struttura della Tesi

Viene qui di seguito presentata brevemente la struttura della tesi, con una spiegazione degli argomenti che verranno trattati in ogni capitolo.

Nel Capitolo 2 viene introdotta la piattaforma Arduino come è costituita e come funziona, il software di programmazione disponibile per implementare il codice da utilizzare, e infine la descrizione e lo scopo delle librerie PID e VarSpeedServo impiegate in questo progetto.

Nel Capitolo 3 è descritto l'ambiente di sviluppo Processing spiegando come è realizzata e quali usi sono possibili, mostrando un esempio di applicazione realizzabile in questo sistema, e alla fine verrà presentata brevemente la libreria ControlP5 utilizzata in questo progetto.

Nel Capitolo 4 viene delineato il lavoro svolto, presentando per primo lo sketch che è stato realizzato in Arduino, con i relativi metodi da cui è formato.

Subito di seguito è descritto lo sketch di Processing, come funziona e i metodi che lo compongono. Infine viene illustrata l'interfaccia grafica creata per comandare il sistema presente su Arduino, con la definizione delle sezioni da cui è formata.

Nel Capitolo 5 vengono presentati i possibili casi d’uso e dei test effettuabili, dando poi una illustrazione dei risultati ottenuti, con e senza l’utilizzo del motore connesso alla scheda Arduino.

Nel Capitolo 6 vengono descritti brevemente dei possibili sviluppi futuri per questo progetto.

2 Capitolo 2

2.1 Arduino

Utilizzando la definizione disponibile sul sito ufficiale ***"Arduino è una piattaforma open source, incentrata su software e hardware semplici da utilizzare"*** [1] quindi si compone di una scheda hardware programmabile e una applicazione software in cui sviluppare e caricare sulla scheda lo sketch prodotto.

Il nome Arduino è derivato dal re d'Italia "*Arduino d'Ivrea"* [10], in onore a un personaggio storico della città di Ivrea dove ha la sede **Interaction Design Institute Ivrea,** l'istituto dove è stata sviluppata la piattaforma Arduino

Per la realizzazione di questo progetto è stata utilizzata la scheda hardware Arduino Uno[2] e la versione 1.6.0 del software ufficiale di Arduino.

2.1.1 Scheda Arduino Uno

La scheda Arduino Uno è provvista di un micro controllore basato su ATmega328 [1] [2], una serie di pin analogici e digitali da utilizzare come input o output a seconda delle esigenze, un connettore USB e uno spinotto alimentazione.

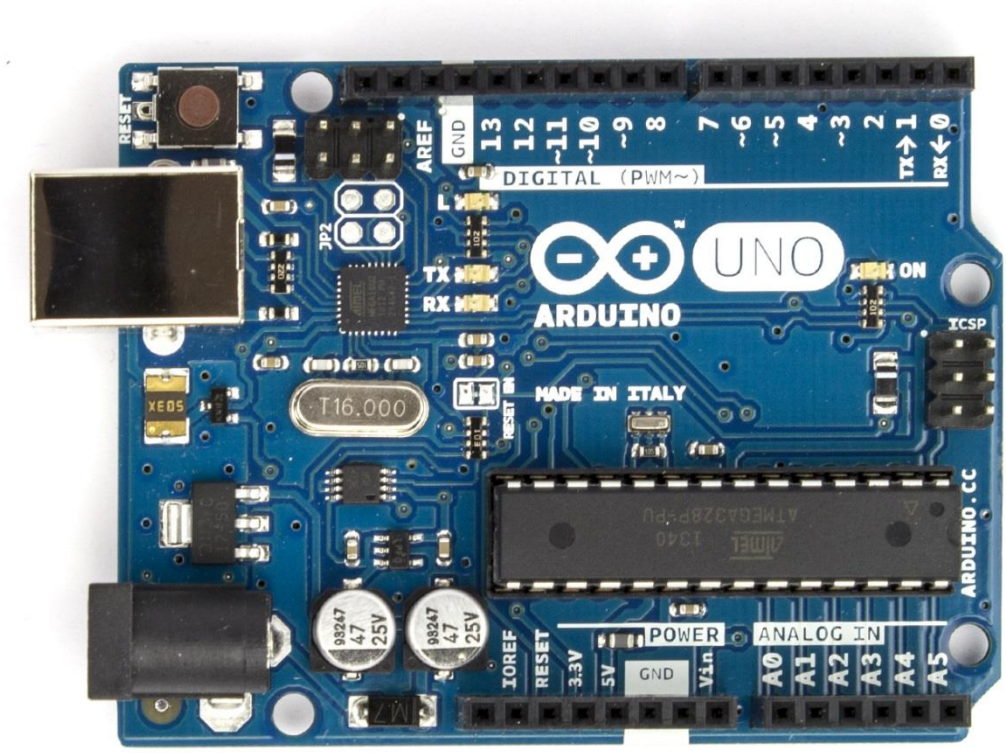

Figura 1 Fronte scheda Arduino Uno

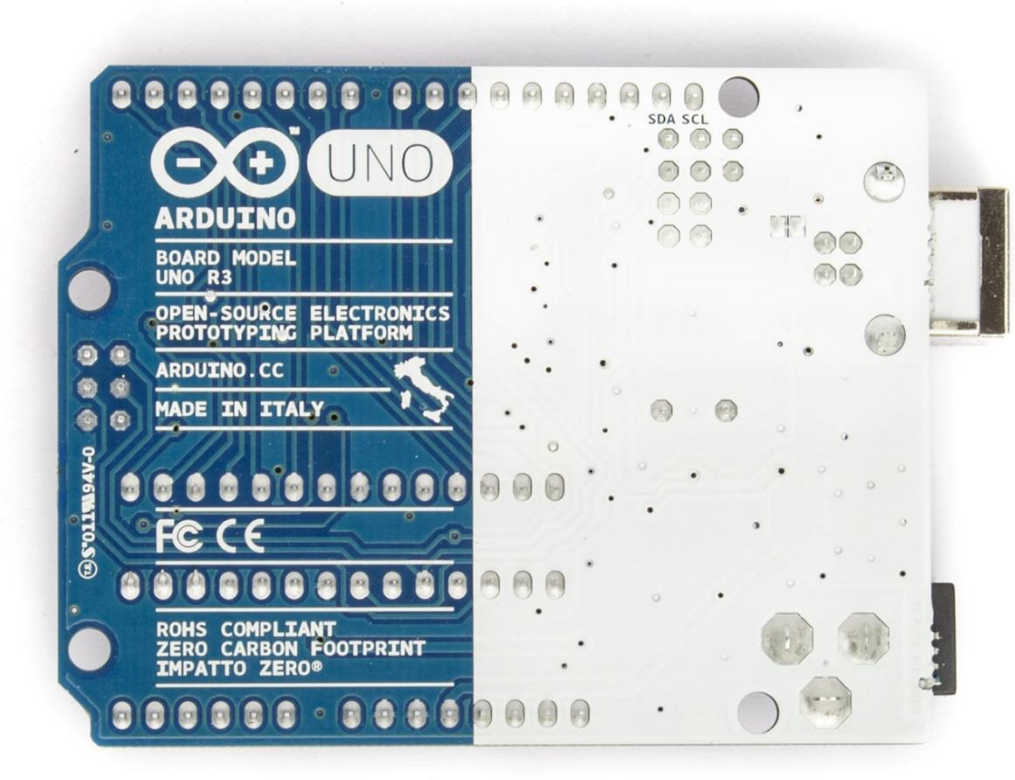

Figura 2 Retro scheda Arduino Uno

Utilizzando la connessione USB è possibile passare alla scheda il programma da realizzare, durante la scrittura nella memoria lampeggeranno i led per indicare questa operazione, una volta completata il programma verrà eseguito dal micro controllore.

L'unica cosa necessaria per poter effettuare una connessione tra un computer e la scheda è l'installazione delle driver, necessarie affinché l'hardware venga correttamente riconosciuto e configurato per l'utilizzo.

2.1.2 Arduino software

Come già accennato nel paragrafo precedente la scheda Arduino Uno può essere collegata a un computer per poter caricare il software sviluppato, questo è possibile farlo tramite un'applicazione gratuita disponibile sul sito ufficiale [1].

Il programma si chiama anch'esso Arduino, nella versione 1.6.0 utilizzata per questo progetto.

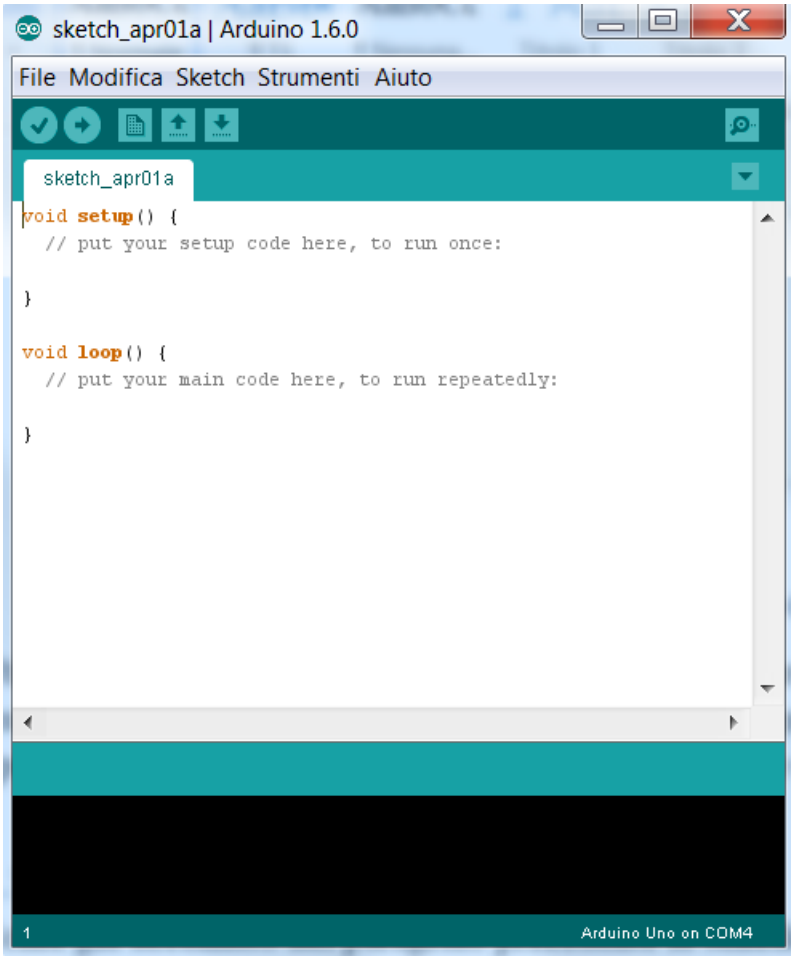

Figura 3 Interfaccia applicazione Arduino 1.6.0 (Windows 7)

Le applicazioni che vengono sviluppate si chiamano sketch (termine derivato dalla omonima scena teatrale comica, per la brevità e semplicità). Ogni sketch è costituito principalmente di 2 metodi:

- **Setup()**: come dice il nome serve ad effettuare un impostazione iniziale del programma, è il primo metodo che viene invocato ogni volta che la scheda Arduino viene avviata (o viene effettuato un nuovo caricamento di uno sketch), e la sua esecuzione viene effettuata solamente una volta;

- **Loop()**: questo è il secondo metodo che viene processato dall'avvio dell'Arduino subito dopo il setup, come dice il nome è ripetuto a ciclo continuo dove effettivamente deve essere inserito il programma da eseguire;

I metodi sopra presentati sono basilari e devono essere sempre presenti in uno sketch, ma è lasciata piena libertà all'utente di creare e definire altri metodi da utilizzare parallelamente.

Il linguaggio di programmazione che si deve utilizzare è C/C++ in una versione semplificata, questo fatto non introduce nessuna limitazione, infatti grazie all'utilizzo delle librerie è possibile espandere le possibilità implementative.

Le limitazioni sono dovute per lo più alla scheda stessa, infatti se si crea un programma di elevata complessità, con diverse variabili, relativamente grandi in termini di spazio, si rischia di occupare tutta la memoria a disposizione.

Per ovviare a queste problematiche è possibile ovviamente utilizzare una scheda con maggiore memoria, ma è anche possibile ampliare la scheda con moduli da connettere ai pin disponibili, questo permette di ottenere un hardware capace di soddisfare le proprie esigenze e necessità.

2.1.3 Libreria PID Arduino

Questa è la libreria da importare nel software di Arduino, e includerla nello sketch che la deve utilizzare, permette di creare e gestire un controllo PID [3], prima di arrivare a descrivere e definire i metodi della compongono, viene qui di seguito descritto in breve il funzionamento generale del controllo PID.

2.1.3.1 Controllo PID

Il PID come già accennato è un sistema in "retroazione" [9] [10] negativa ampiamente impiegato nei sistemi di controllo, di gran lunga più comune nell'industria.

Grazie a un input che determina il valore attuale, è in grado di reagire a un eventuale errore positivo o negativo tendendo verso il valore 0 per bilanciarne l'effetto. La reazione all'errore può essere regolata e ciò rende questo sistema molto versatile.

Esso regola l'uscita in base alle seguenti azioni:

- Valore del segnale di errore (azione proporzionale) $Ap = KP * e(t)$ è pari a una costante per l'errore;

- Valori passati del segnale di errore (azione integrale) $Ai = KI \int e(t)\, dt$ è pari a una costante per l'integrale dell'errore rispetto al tempo;

- Quanto velocemente il segnale di errore varia (azione derivativa) $Ad = KD\, (de(t)/dt)$ è pari a una costante per la derivata dell'errore rispetto al tempo;

Queste azioni calcolate singolarmente, come presentato in precedenza, vengono poi sommate algebricamente nella seguente formula:

uscita = Ap + Ai + Ad

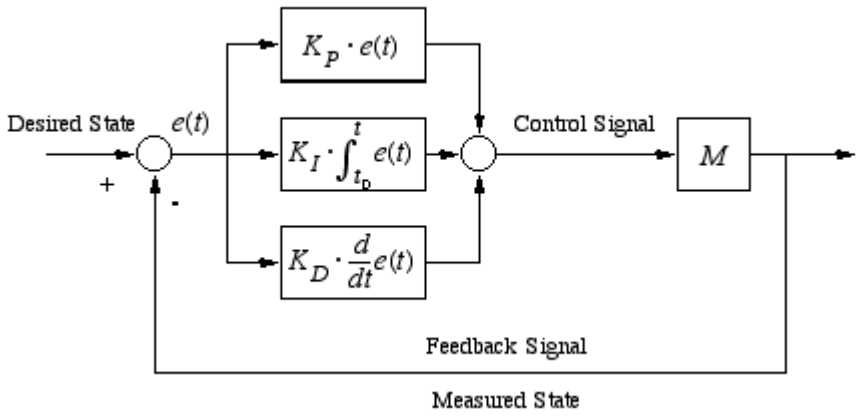

Figura 4 Controllo PID rappresentazione a blocchi

Variando i valori dei parametri del controllo si può ottenere un diverso comportamento, che può essere più e meno aggressivo, questo a seconda delle richieste e del segnale che si vuole ottenere in uscita.

2.1.3.2 Metodi Libreria PID

Vengono ora illustrati i metodi forniti dalla Libreria PID di Arduino, vediamo quali sono e che operazioni permettono:

- **PID(&Input, &Output, &Setpoint, KP, KI, KD, DirRev):** costruttore della classe che ci permette di creare un oggetto PID, passandogli i riferimenti a parametri di Input, Output e i valori del Setpoint, Proporzionale, Integrale, Derivativo. L'ultimo parametro serve a definire se vogliamo che il PID sia calcolato diretto o inverso;

- **SetMode(AutoMan):** permette di settare se il PID deve essere in modalità manuale oppure automatica;

- **SetOutputLimits(Min, Max):** per definire dei valore entro cui non deve andare il valore di output calcolato;

- **SetTunings(KP, KI, KD):** con questo metodo possiamo cambiare i parametri del PID;

- **SetSampleTime(time):** imposta la frequenza con cui deve essere effettuato il calcolo del PID;

- **Compute():** metodo con cui viene richiesto il calcolo del PID;

Grazie ai metodi di questa libreria è possibile realizzare un sistema di controllo di tipo PID in Arduino, utilizzabile ovviamente in più casi non necessariamente e solamente per gestire motori elettrici.

2.1.4 Libreria VarSpeedServo

La libreria Servo viene utilizzata per controllare un servo motore, in particolare per questo progetto è stata utilizzata una versione modificata che permette di controllare la velocità del servo, libreria chiamata "VarSpeedServo" [5], che è un estensione di quella base aggiungendo solo alcune funzionalità.

Come per la libreria PID basta importarla nel software di Arduino e includerla nello sketch per utilizzarla, al suo interno vengono forniti una serie di metodi da utilizzare per inizializzare e gestire il servo motore.

Vediamo brevemente quali sono i metodi di questa libreria:

- **VarSpeedServo():** metodo costruttore della libreria per creare un nuovo oggetto di tipo servo;

- **attach(pin):** permette di legare il nostro oggetto di tipo servo al pin dell'Arduino a cui è collegato il servo che vogliamo controllare, in questa libreria viene fornita un'altra versione così definita **attach(pin, min, max),** il comportamento è lo stesso della versione base ma in più è possibile definire un valore minimo e massimo del valore passato al servo;

- **detach():** l'operazione opposta della precedente, infatti permette di slegare l'oggetto servo dal pin a cui era connesso;

- **write(value):** scrive un valore mandato al servo motore, se inferiore a 200 viene considerato come un angolo di movimento che il servo deve effettuare;

- **writeMicroseconds(value):** simile al metodo precedente scrive un valore inviato al servo, ma questo è un impulso di espresso in millisecondi;

- **slowmove(value, speed):** è una funzione che permette di muovere il servo motore a un data posizione mantenendo un certa velocità richiesta;

Grazie a questi metodi è possibile connettere ad una scheda Arduino un servo motore e controllarlo come voluto, e grazie alle funzioni aggiunte di questa particolare libreria è possibile avere un controllo più preciso della velocità del servo.

3 Capitolo 3

3.1 Processing

"Processing è un linguaggio di programmazione, un ambiente di sviluppo e anche una comunità online" questa è la breve descrizione presente sulla pagina principale del sito ufficiale di Processing [6], che descrive lo scopo e l'uso dell'ambiente Processing.

Come Arduino anche Processing è un progetto sviluppato all'interno del **Interaction Design Institute Ivrea,** infatti l'IDE (*Integrated Development Environment),* in italiano ambiente di sviluppo integrato, ha un interfaccia e una composizione simile al software Arduino.

Anche in questo software si deve creare un sketch da compilare e avviare in tempo reale, per provare e utilizzare quello che si è realizzato.

La versione utilizzata in questo progetto è la 2.2.1.

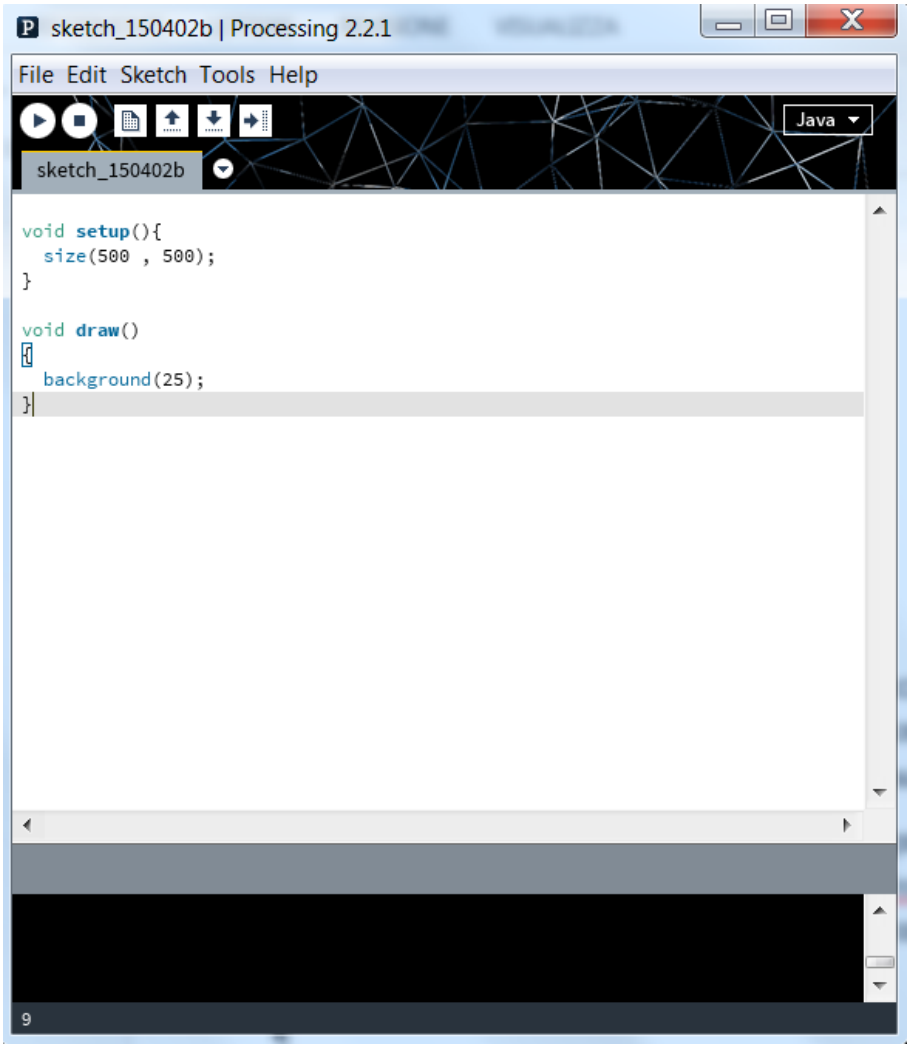

Figura 5 Interfaccia applicazione Processing 2.2.1 (Windows 7)

La differenza sostanziale rispetto al linguaggio che si usa in Arduino, la programmazione Processing eredita completamente la sintassi, i comandi e il paradigma di programmazione a oggetti di Java, ma in più aggiunge funzioni e metodi di alto livello per semplificare la gestione di aspetti grafici e multimediali.

Anche qui esistono 2 metodi fondamentali, che compongono lo sketch:

- **Setup():** metodo che serve a impostare i parametri iniziali del programma Processing, come ad esempio definire le dimensioni della finestra in cui verrà eseguita l'applicazione, oppure i font da utilizzare, etc. ;

- **Draw ():** metodo che viene eseguito a ciclo continuo (simile al loop di Arduino), dove viene effettivamente eseguita l'applicazione Processing;

Come in tutti gli ambienti di programmazione è possibile definire metodi personalizzati per le proprie applicazioni, inoltre è anche disponibile l'importazione di librerie e classi java per ampliare le possibilità di programmazione.

Questo ambiente di sviluppo è stato scelto per la sua ampia potenzialità, visto che può essere utilizzato per parametrizzazione in tempo reale, oltre a essere un affidabile e notevole strumento per il supporto al design alla verifica del controllore PID.

3.1.1 Esempio: moto browniano

Qui di seguito viene riportato un esempio di codice per realizzare un moto browniano [7] tramite Processing, questo per dare un idea di cosa è realizzabile in questo ambiente di sviluppo.

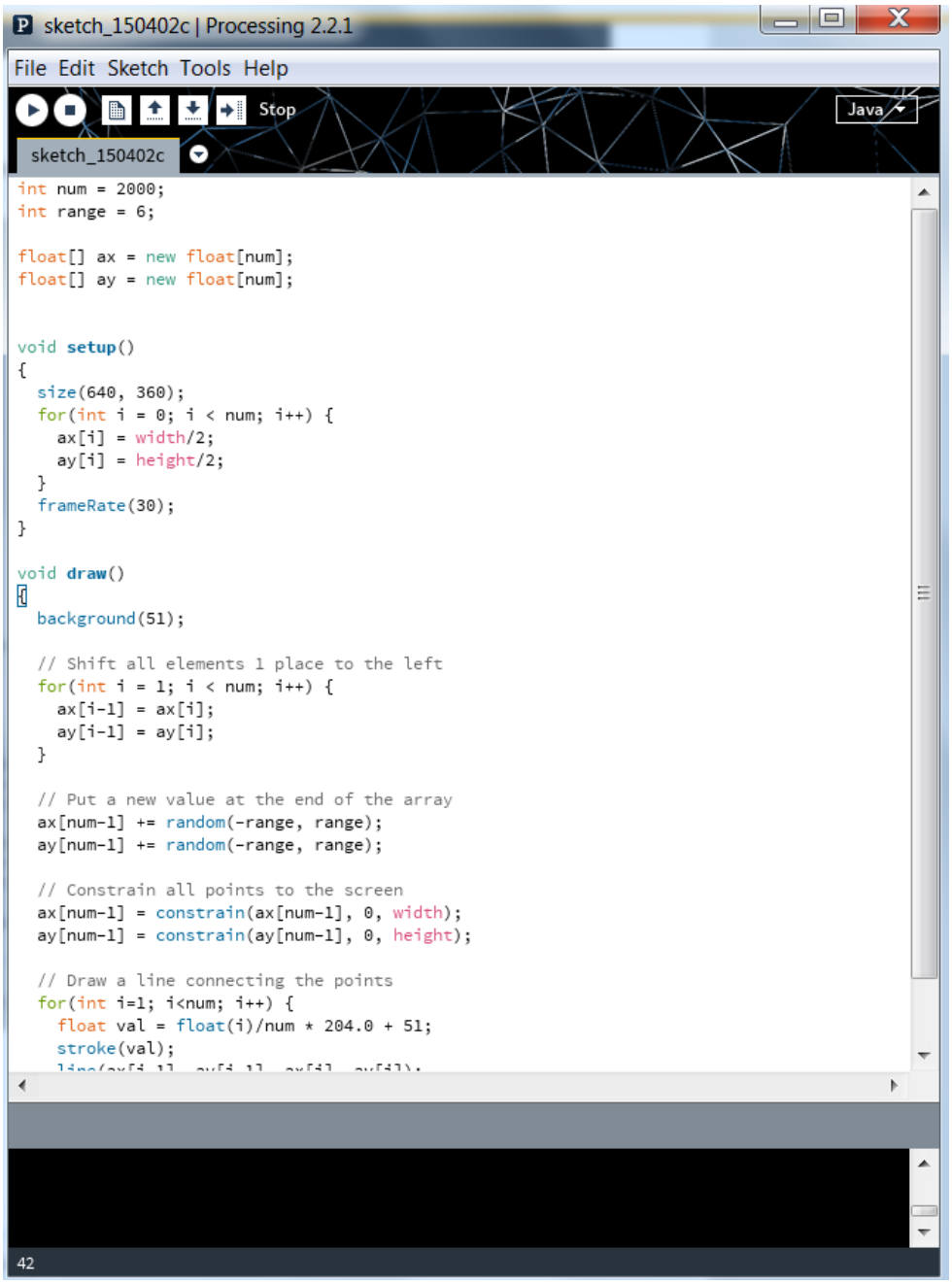

Figura 6 Codice esempio di moto Browniano

Il moto browniano è un movimento disordinato di particelle di dimensioni microscopiche presenti nei fluidi o in sospensioni fluide, questa applicazione simula questo moto e viene visualizzato in una finestra.

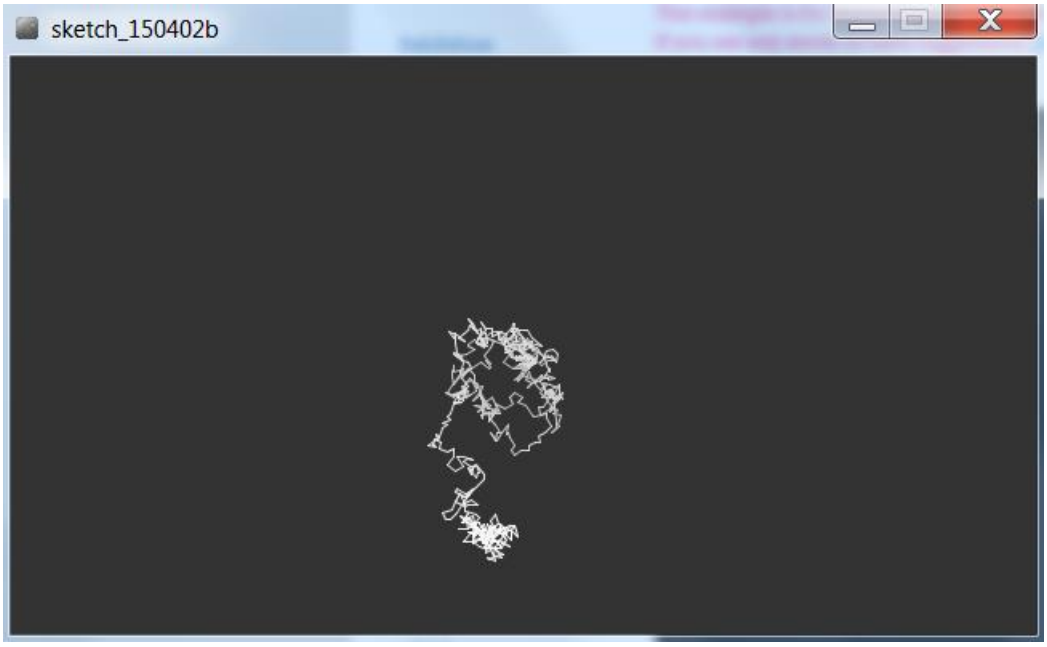

Figura 7 Esempio di moto Browniano creato tramite Processing

3.1.2 Libreria ControlP5

La libreria ControlP5 [8] è stata utilizzata in questo progetto per realizzare alcune parti dell'interfaccia di Processing, in particolare i pulsanti, le text area e le etichette.

Ecco un semplice codice di esempio di come è possibile utilizzare questa libreria, in cui sono stati inseriti 3 elementi, una area per inserire del testo (evidenziata in rosso), un pulsante (evidenziato in blu) e un'etichetta (evidenziata in verde)

```
import controlP5.*;
int windowWidth = 200;      // set the size of the
int windowHeight = 200;
ControlP5 controlP5;
controlP5.Button Button;
controlP5.Textlabel Label;
controlP5.Textfield Field;

void setup()
{
  frameRate(30);
  background(12);
  size(windowWidth , windowHeight);
  controlP5 = new ControlP5(this);
  Field = controlP5.addTextfield("Test_text_Field",10,20,120,20);
  Button = controlP5.addButton("Test_Button",0.0,10,70,120,20);
  Label = controlP5.addTextlabel("Test_Label","Test_Label",10,130);
}
```

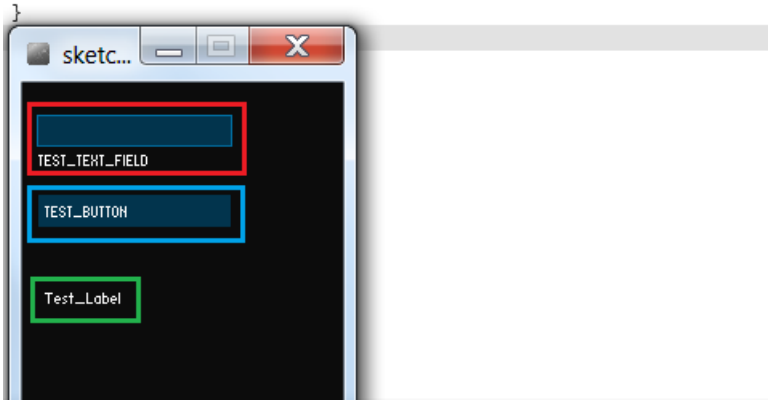

Figura 8 Esempio utilizzo della libreria ControP5

Questa libreria presenta svariate funzioni e metodi usabili, ma sono qui presentati solo gli esempi degli elementi che sono necessari ed è sono stati utilizzati in questo progetto.

4 Capitolo 4

4.1 Sketch del progetto

In questo capitolo vengono presentati gli sketch che sono stati implementati per arrivare a completare il progetto.

Nel primo paragrafo si descrive lo sketch realizzato per essere caricato sulla scheda Arduino, quali metodi lo compongono e per quale scopo sono stati implementati e come funzionano,

Nel secondo paragrafo viene presentato il lavoro svolto nella creazione dello sketch di Processing per l'interfaccia grafica di comando.

4.1.1 Sketch Arduino

In questo paragrafo viene definita l'implementazione dello sketch di Arduino, in particolare viene prima data una descrizione di ciò che viene eseguito nei metodi principali setup e loop, e successivamente una spiegazione di ogni metodo creato motivandone lo scopo per il quale sono stati realizzati.

4.1.1.1 Metodi dello sketch

Vediamo qui di seguito l'elenco dei metodi realizzati nello sketch da caricare su Arduino, di ognuno è fornita la spiegazione precisa del fine per cui sono stati realizzati:

- **setup():** esegue solo il metodo **initialize()** (vedi sotto), questo è dovuto solamente a una scelta implementativa;

- **loop():** parte centrale dell'applicazione su Arduino, nel caso di questo progetto si suddivide in una sezione che effettua solo un dialogo continuo con l'interfaccia, con invio e ricezioni di dati sulla seriale.
  Se invece si riceve un comando di avvio del servo motore entra in gioco la seconda parte, che contiene a sua volta il sistema di dialogo seriale, ma in più utilizza il valore di output per regolare la velocità del servo motore.
  Data la natura del motore che risulta comandabile tramite valori angolari, è stato realizzato un sistema per mappare il valore prodotto dal controllore PID rispetto a un angolo da inviare al servo;

- **initialize():** è metodo di inizializzazione dello sketch, apre la connessione seriale, setta il servo motore e imposta i parametri del PID, viene richiamata solo dal setup;

- **SerialReceive():** questo è il primo dei due metodi necessari a dialogare con l'interfaccia grafica di Processing (vedi prossimo capitolo), ogni volta che viene richiamata questa funzione vengono inviati i dati necessari tramite la connessione seriale;

- **SerialSend():** secondo metodo di dialogo sulla seriale con l'interfaccia di Processing, a ogni esecuzione legge i dati che vengono ricevuti e li memorizza nelle variabili di Arduino. Questo è la funzione che viene utilizzato per ricevere e processare i comandi ricevuti e di conseguenza variare il comportamento del PID e del servo motore;

- **resetmotor():** un metodo implementato soprattutto per resettare il motore servo in una posizione di fermo, allo scopo di poter invertire la marcia, ovviando a un'inversione immediata causando un stop del motore;

- **resetFunc():** un metodo per resettare la scheda Arduino alla prima operazione, in pratica permette di riportarla allo stato iniziale;

Per un funzionamento indipendente dall'interfaccia grafica è possibile impostare facilmente le variabili presenti all'inizio dello sketch, settandole come voluto e effettuando un caricamento sulla scheda Arduino, che eseguirà il programma in base ai dati impostati.

Nel seguente paragrafo viene illustrata la seconda parte del progetto, l'interfaccia di comando realizzata tramite Processing.

4.1.2 Sketch Processing

In questo paragrafo viene definita l'implementazione dello sketch di Processing, in particolare viene prima data una descrizione di ciò che viene eseguito nei metodi principali setup e draw, e successivamente una spiegazione di ogni metodo creato motivandone lo scopo per il quale sono stati realizzati.

4.1.2.1 Metodi dello sketch

Qui di seguito vengono presentati i metodi implementati all'interno dello sketch di Processing, e le funzionalità che implementano al fine di comandare il processo in esecuzione sulla scheda Arduino.

Tutti i pulsanti presenti nell'interfaccia richiameranno direttamente l'esecuzione del metodo corrispondente (il nome deve esser il medesimo) quando verranno premuti tramite il puntatore del mouse.

Andiamo ora a vedere quali sono le funzioni definite nello sketch:

- **setup():** effettua le impostazioni di partenza dell'interfaccia, in particolare viene definito e creato un file di testo dove verranno loggati i dati inviati dall'Arduino, viene fatto il setup della parte grafica di tutti gli elementi necessari (etichette, pulsanti…) e infine viene impostata la finestra di interfaccia. Inoltre viene anche stabilita la connessione tramite seriale con la scheda Arduino;

- **draw():** lo scopo di questo metodo è di impostare lo sfondo e richiamare i metodi **drawGraph()** e **drawButtonArea()** (vedi descrizioni successive). La sua esecuzione viene ripetuta ciclicamente fino al termine dell'applicazione o alla chiusura della finestra;

- **drawGraph():** questo metodo serve a disegnare i grafici di alcuni dati, letti tramite seriale, provenienti dalla scheda Arduino, ed esattamente sono i valori dei Setpoint, dell'Input, dell'Output generato dal PID e l'angolo calcolato e inviato al servo motore;

- **drawButtonArea():** metodo per disegnare l'area dove sono presenti gli elementi come pulsanti, aree di testo o etichette;

- **Toggle_AM():** per cambiare il comportamento del PID da automatico in manuale, richiamato tramite il pulsante corrispondente:

- **Toggle_DR():** per cambiare il PID da diretto a inverso richiamato tramite il pulsante corrispondente;

- **Send_To_Arduino():** metodo semplice ma fondamentale, che serve a leggere i valori inseriti nell'interfaccia e inviarli ad Arduino attraverso la seriale, questo permetterà di comandare il comportamento del sistema di controllo PID tramite le variazioni di parametri passati;

- **Reset_Arduino():** simile alla funzione precedente, ma che invia un segnale per resettare la scheda Arduino, e ritornare all'istruzione 0;

- **floatArrayToByteArray(input):** un metodo di supporto che serve per convertire i dati letti dall'interfaccia in byte da inviare sulla seriale, viene richiamato ad ogni esecuzione di una funzione che deve inviare dati verso la scheda Arduino;

- **serialEvent(myPort):** il metodo che serve a leggere i dati provenienti dalla scheda Arduino, appena li riceve prima scrive nel file di log e poi inserisce i valori nelle varie etichette da aggiornare;

- **Start_M() e Stop_M():** sono le due funzioni per avviare o stoppare il passaggio dei valori calcolati dal PID verso il pin su cui è previsto la connessione con il servo motore;

- **readData(myFileName):** un metodo che dato un file di input, se esiste, legge i valori al suo interno, separati tra di loro da un punto e virgola, li salva in una lista di array, viene utilizzato per leggere una serie di possibili setpoint definiti in un file di testo (o .CSV), i quali possono essere prima visualizzati sull'interfaccia utilizzando la funzione **Send_Conf()**;

- **Send_Conf():** come detto poco sopra questo metodo serve a leggere una dei possibili setpoint memorizzati, utilizzando un valore definito nella text area corrispondente per specificare quale riga di parametri selezionare;

- **Reload_File():** permette di ricaricare un nuovo file (con lo stesso nome ma ad esempio modificato) quando l'interfaccia e già avviata e in uso;

Data la spiegazione dei metodi che compongono lo sketch, nel prossimo paragrafo vediamo come si presenta l'interfaccia mentre è in esecuzione.

4.1.2.2 Interfaccia grafica

Viene ora presentata l'interfaccia Processing in funzione, dando un spiegazione precisa degli elementi che la compongono, quali funzioni eseguono e lo scopo per cui sono stati sviluppati.

All'avvio dello sketch di processing verrà chiesto se selezionare l'interfaccia di comando oppure quella di invio di sequenze, la prima permette un tuning preciso e completo del sistema controllo PID, la seconda permette di inviare una serie di set di parametri letta da un file CSV e farla eseguire sulla scheda in base ai tempi definiti nello stesso file di input.

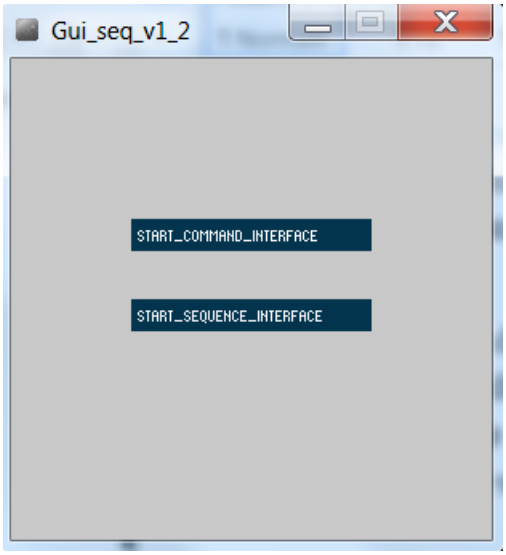

Figura 9 Selezione Interfaccia grafica

4.1.2.2.1 Interfaccia di Comando

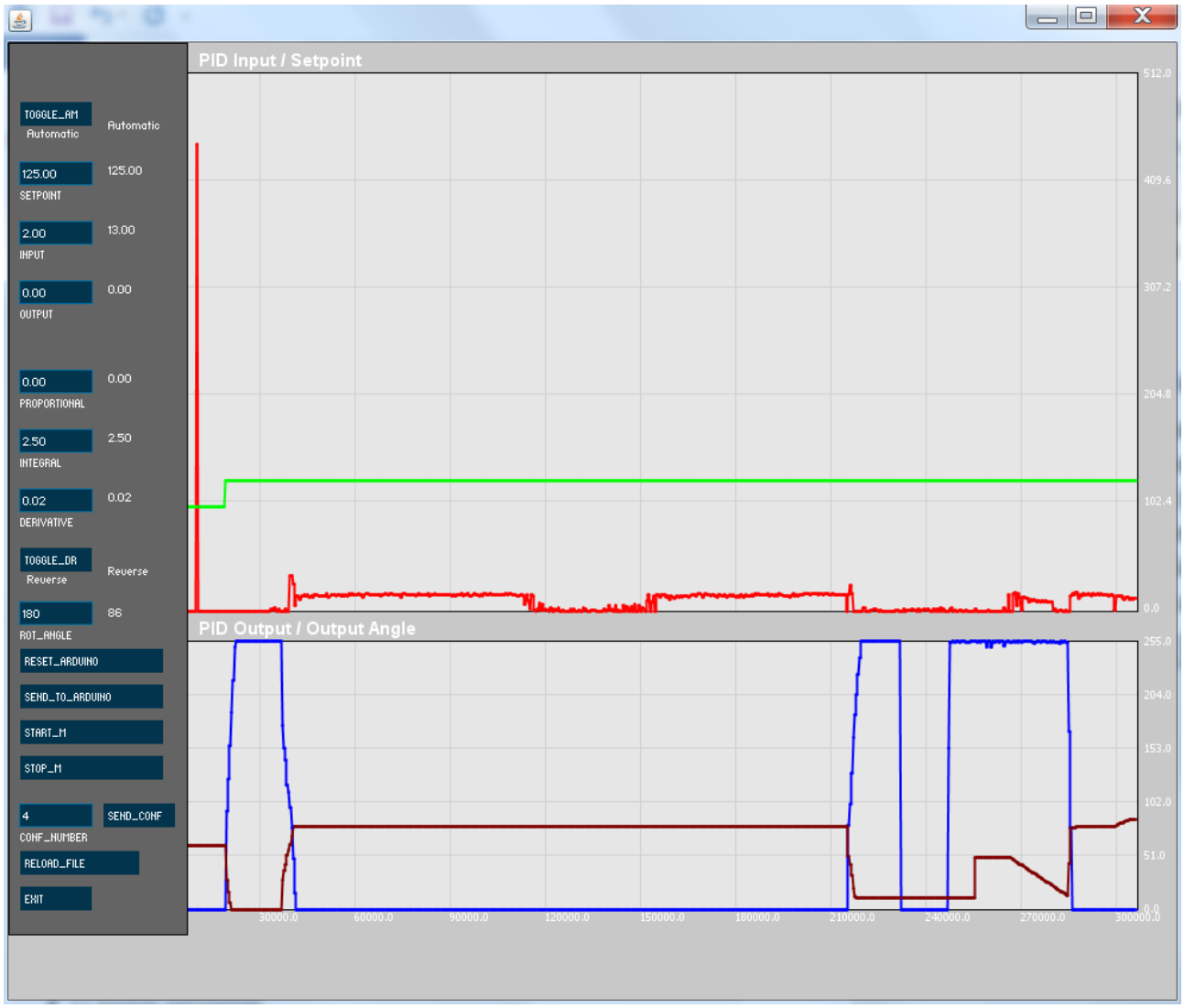

Figura 10 Interfaccia di comando completa

Come detto in precedenza questa interfaccia ci permette un comando e settaggio completo del controllo PID presente su Arduino.

Nella parte più grande dell'interfaccia sono riportati i grafici dei dati letti via seriale dalla scheda Arduino, nella sezione superiore, sono disegnati gli andamenti dei valori del pin di ingresso (in rosso) e dei setpoint (in verde), nella sezione inferiore sono invece raffigurati gli andamenti nel tempo del valore di output calcolato tramite il controllo PID (in blu), e il relativo angolo che deve essere passato al servo motore (in rosso scuro), quest'ultimo è calcolato proporzionalmente rispetto al output del PID.

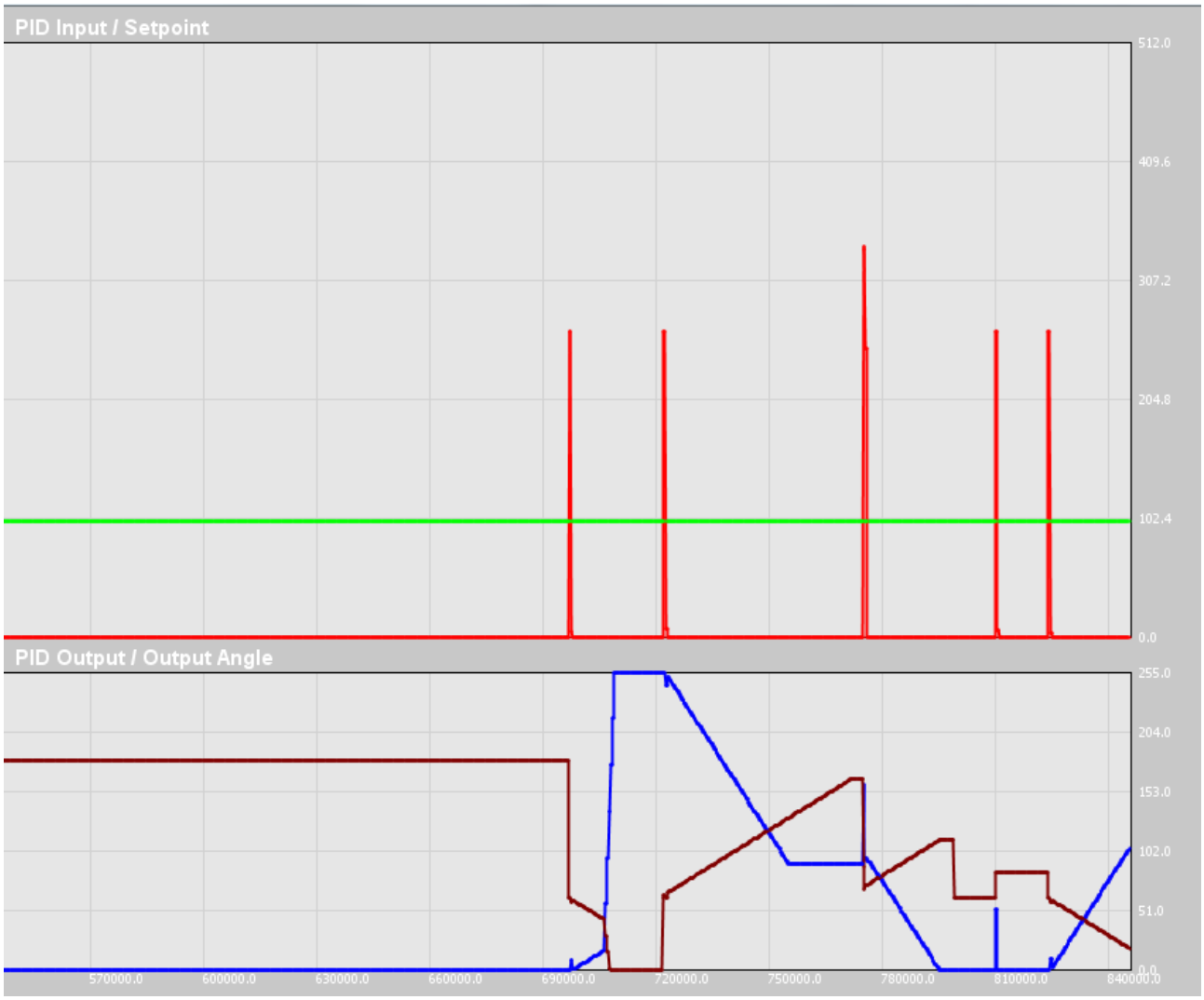

Figura 11 Sezione grafici

Ovviamente questi valori sono rappresentati rispetto al tempo di avvio dell'interfaccia, con un valore temporale espresso in millisecondi.

La sezione a sinistra dell'interfaccia si suddivide a sua volta in 3 sotto parti:

- La prima è costituita delle aree di testo dove è possibile inserire i valori dei parametri da inviare alla scheda Arduino, compreso un'area di testo dove definire un angolo di rotazione minimo (massimo) da inviare al servo connesso. Inoltre sono presenti 2 pulsanti, il primo per impostare il calcolo dell'output del PID da manuale a automatico e viceversa, mentre il secondo pulsante per invertire il senso di marcia da diretto a inverso e viceversa.
  Ognuno di questi elementi ha alla sua destra un etichetta che riporta il valore attuale sulla scheda Arduino del corrispondente parametro;

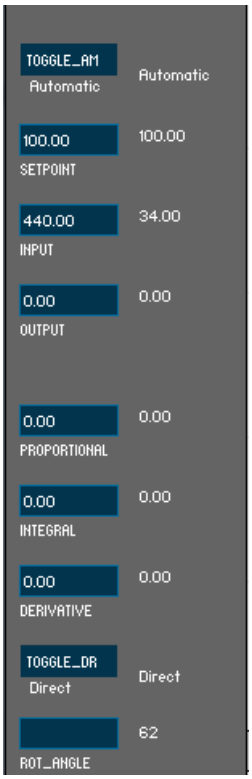

Figura 12 Sezione parametri Arduino

- La seconda sezione è composta da 4 pulsanti, tutti per l'invio di comandi verso la scheda Arduino, ognuno di essi ha il nome che corrisponde ai metodi dello sketch di Processing, e servono ad avviare la funzione descritta in precedenza;

Figura 13 Sezione pulsanti comando

- L’ultima sezione serve per effettuare un caricamento di una singola configurazione letta da un file, infatti è composta da un’area di testo dove inserire un valore numerico corrispondente al riga del file caricato, un pulsante per settare questi valori nella prima sezione dell’interfaccia, e successivamente inviarli verso la scheda Arduino. Infine è presente un secondo pulsante dove è possibile effettuare un nuovo caricamento dello stesso file nel caso si voglia modificare e utilizzare nuove configurazioni, senza fermare l’esecuzione;

Figura 14 Sezione caricamento file

Per terminare l’esecuzione dell’interfaccia è possibile utilizzare il pulsante **Exit**.

4.1.2.2.2 Interfaccia sequenza

L'interfaccia che si occupa solo dell'invio della sequenza risulta simile alla precedente ma con molti elementi in meno, dato che non sono necessari in quest'interfaccia.

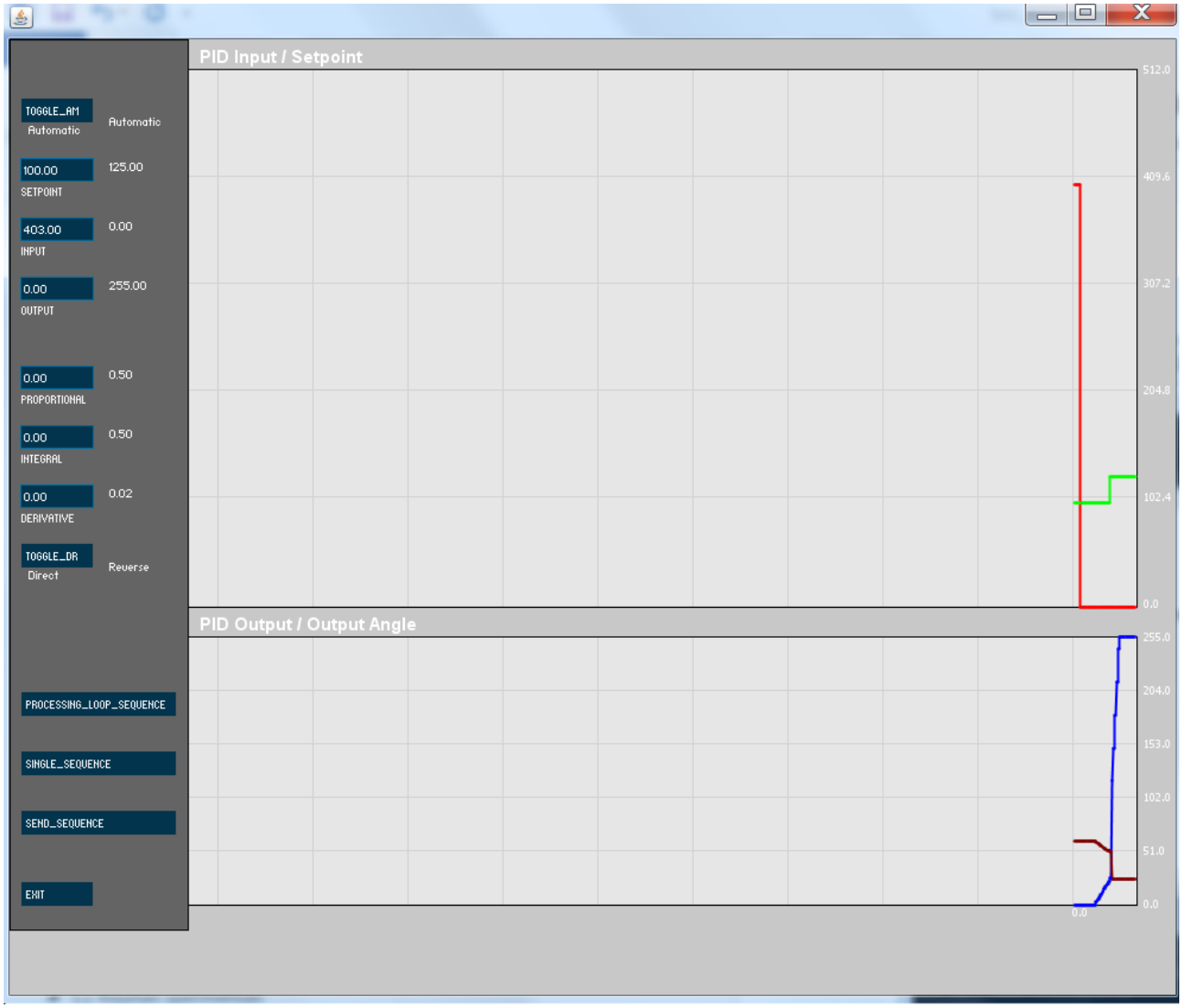

Figura 15 Interfaccia sequenza completa

Le parti della lettura dei parametri e dei grafici sono le stesse, la differenza sostanziale è nei pulsanti che inviano comandi alla Scheda Arduino.

Infatti sono disponibili 3 pulsanti di comando:

- Il primo **Processing_Loop_Sequence** permette eseguire un loop infinito, gestito tramite l'applicazione processing, la quale invia un set di parametri letti da un file di input.
  Ogni set ha tra i valori definiti un tempo espresso in millisecondi al termine del quale l'applicazione deve passare al set successivo, una volta che arriva all'ultimo record definito, ricomincia la sequenza da capo, questo finché non viene interrotta l'interfaccia tramite il pulsante **Exit** o scollegato l'Arduino.

- Il secondo pulsante, **Single_Sequence**, permette di inviare una sequenza di set di parametri alla scheda Arduino, valori sempre letti dallo stesso file di input, ma l'esecuzione viene effettuata solamente una volta al termine della quale verrà mantenuto l'ultimo set letto.

- Il terzo e ultimo pulsante, **Single_Sequence**, funziona come il precedente, infatti invia alla scheda Arduino una sequenza di set di parametri, ogni volta la scheda memorizzerà i valori ricevuti in una serie definita di vettori, uno per parametro. Al termine della sequenza Processing invia un set di default per avvisare la scheda che la sequenza è terminata, da questo momento l'Arduino ripete la sequenza all'infinito finché non viene inviato un comando di reset dall'interfaccia.

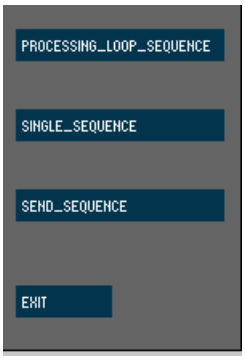

Figura 16 Pulsanti di comando sequenza

Per terminare l'esecuzione dell'interfaccia è possibile utilizzare il pulsante **Exit**.

4.1.2.2.3 Input e Output file

Come definito nella presentazione dell'interfaccia grafica, vengono utilizzati 2 tipi file di input, 1 per tipo di interfaccia.

I file di input sono dei file CSV, con ";" come separatore, che possono essere modificati con un semplice file di testo o con un gestore di foglio di calcolo, che mantenga la formattazione, e l'unica differenza tra i due file e che quello della sequenza contiene anche il tempo di esecuzione di un set di parametri espresso in millisecondi.

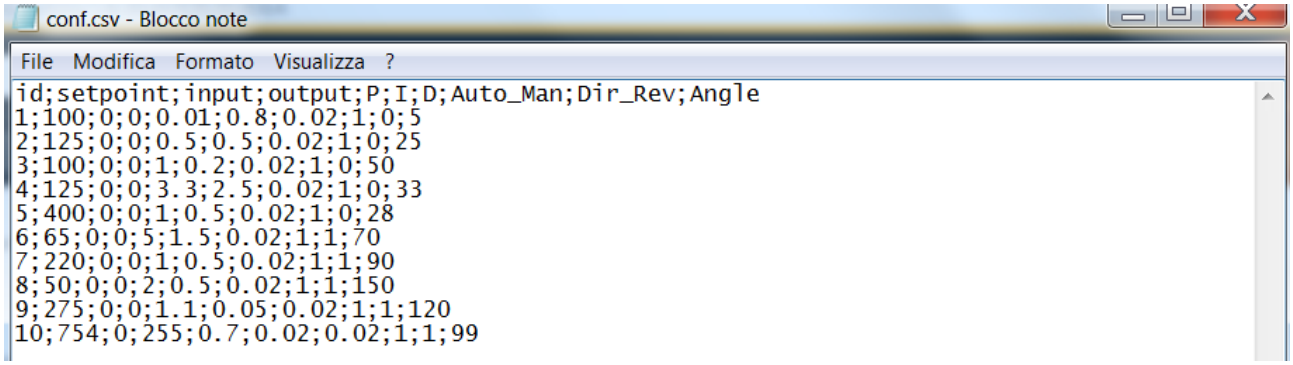

Figura 17 Esempio CSV utilizzato dall'interfaccia di comando

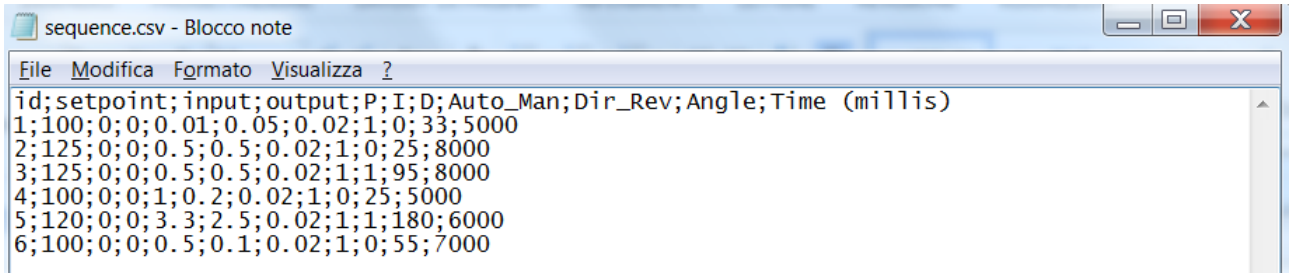

Figura 18 Esempio CSV utilizzato dall'interfaccia sequenza

Inoltre entrambe le interfacce, quella di comando e quella di sequenza, eseguono una funzione non visibile, ad ogni avvio creano un nuovo file di testo, all'interno del quale vengono salvati i dati letti dalla scheda Arduino finché è in esecuzione l'applicazione Processing.

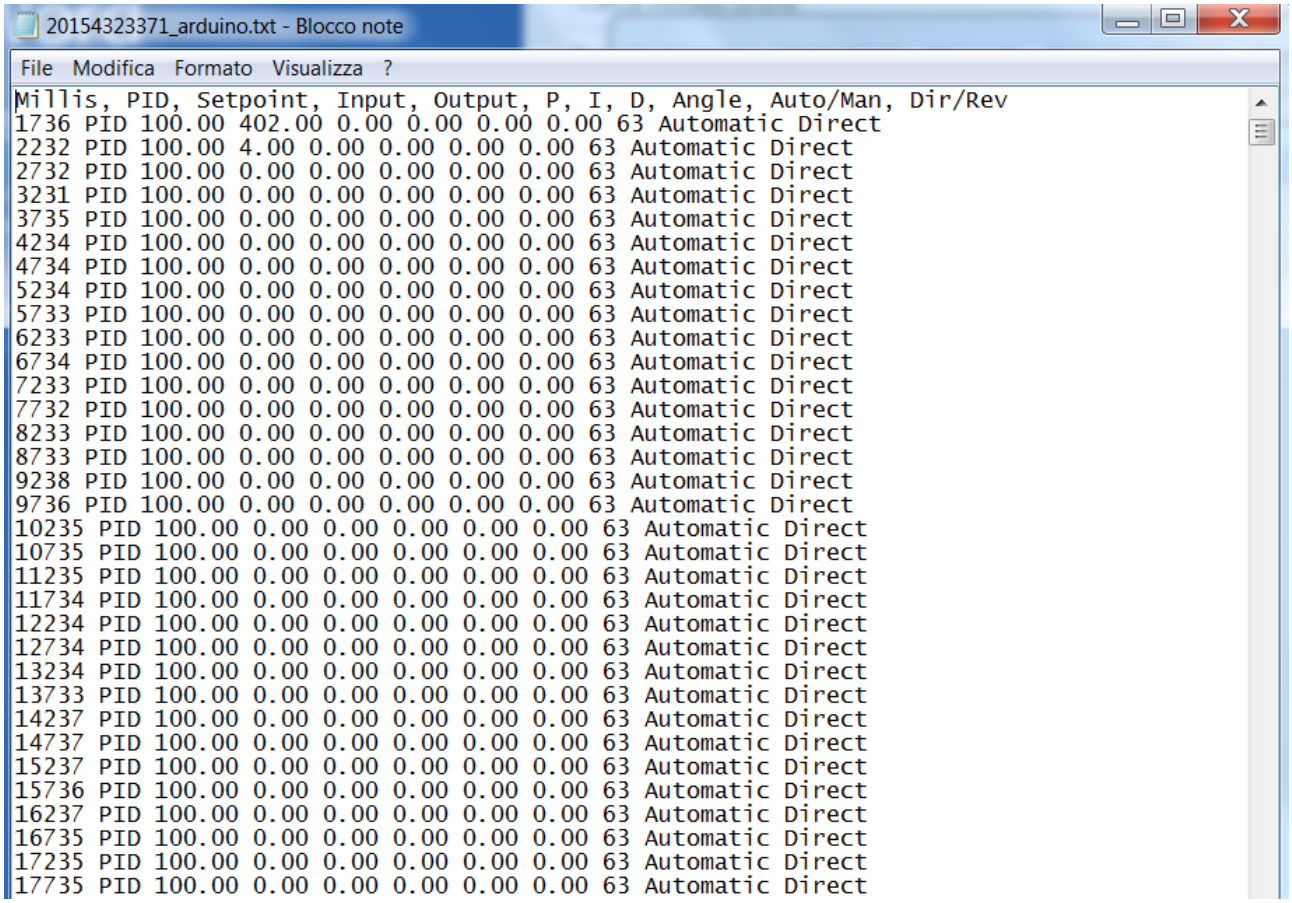

Figura 19 Esempio di file testo creato da Processing

E possibile cambiare il nome e la cartella di destinazione dei file di input e del log di output, agendo sulle variabili outputFileName e inputFilename presenti nella parte iniziale del codice di ogni interfaccia.

5 Capitolo 5

5.1 Risultati sperimentali

Per concludere andiamo ora a vedere quali possibili casi d'uso possono essere effettuati con l'interazione tra i due sketch e i possibili casi di test effettuati, successivamente vengono mostrati i comportamenti del controllo PID, con dei set di parametri inviati, riportando i grafici ottenuti rispetto a questi valori.

Infine è presentata una breve relazione dei risultati ottenuti con i test effettuati utilizzando un motore servo.

5.1.1 Casi d'uso e scenari di test

Vediamo a partire dai possibili casi d'uso le interazioni che un utente può effettuare con l'interfaccia grafica, descrivendo una serie di casi di test.

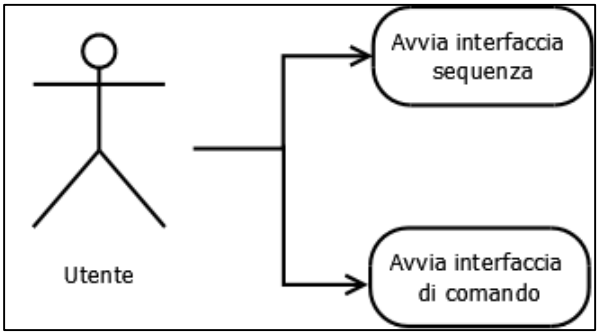

Figura 20 Selezione interfaccia

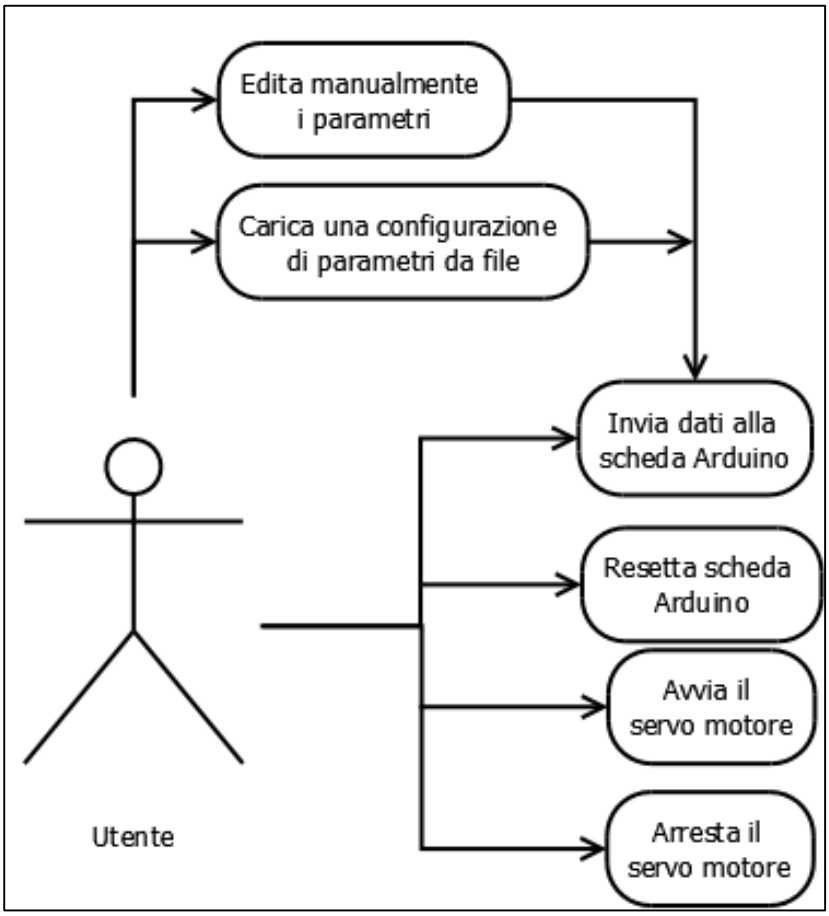

Figura 21 Casi d'uso interfaccia di comando

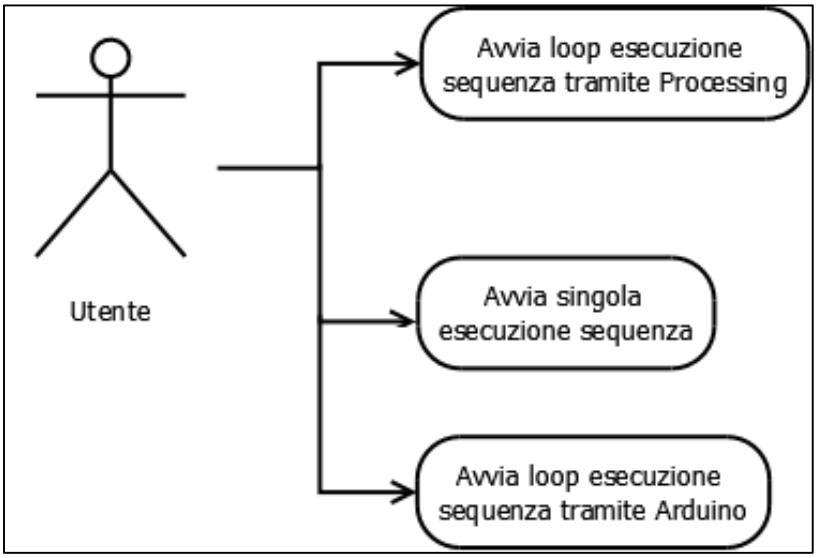

Figura 22 Casi d'uso interfaccia sequenza

Caso di test numero 1

Titolo	Invio dati a scheda Arduino Uno
Precondizioni	Caricamento sketch su Arduino
Passaggi	1) Avvio dell'interfaccia di comando 2) Modifica manuale dei Parametri 3) Pressione del pulsante Send_to_Arduino
Risultati	Arduino restituisce i parametri attuali che corrispondono a quelli inviati tramite interfaccia

Caso di test numero 2

Titolo	Invio dati a scheda Arduino Uno, con configurazione letta da file
Precondizioni	Caricamento sketch su Arduino
Passaggi	1) Avvio dell'interfaccia di comando 2) Inserire un valore della configurazione da utilizzare 3) Premere il pulsante Send_Conf 4) Premere il pulsante Send_to_Arduino
Risultati	Arduino restituisce i parametri attuali che corrispondono a quelli inviati tramite interfaccia

Caso di test numero 3

Titolo	Avvio e arresto del servo motore
Precondizioni	Caricamento sketch su Arduino
Passaggi	1) Avvio dell'interfaccia di comando 2) Settare i parametri come descritto nei test case precedenti (facoltativo) 3) Premere il pulsante Start_M 4) Premere il pulsante Stop_M dopo un tempo voluto
Risultati	Arduino comanderà il servo motore, il quale si avvierà seguendo il valore calcolato dal PID e si arresterà quando verrà richiesto

Caso di test numero 4

Titolo	Reset della scheda Arduino
Precondizioni	Caricamento sketch su Arduino
Passaggi	1) Avvio dell'interfaccia di comando 2) Settare i parametri come descritto nei test case precedenti 3) Premere il pulsante Reset_Arduino
Risultati	Arduino torna ad eseguire la prima istruzione di quando è stato avviato

Caso di test numero 5

Titolo	Avvio sequenza con singola esecuzione
Precondizioni	Caricamento sketch su Arduino Modifica dei dati nel file CSV
Passaggi	1) Avvio dell'interfaccia sequenza 2) Premere il pulsante Single_Sequence
Risultati	Processing invia un set di parametri previsto da una sequenza letta da una file, questi dati vengono letti e eseguiti dalla scheda, al termine della sequenza Processing non invia più dati.

Caso di test numero 6

Titolo	Avvio sequenza con loop gestito da Processing
Precondizioni	Caricamento sketch su Arduino Modifica dei dati nel file CSV
Passaggi	1) Avvio dell'interfaccia sequenza 2) Premere il pulsante Processing_Loop_Sequence
Risultati	Processing invia un set di parametri previsto da una sequenza letta da una file, questi dati vengono letti e eseguiti dalla scheda, al termine della sequenza Processing riparte dal set iniziale e ripete la sequenza in un loop infinito.

Caso di test numero 7

Titolo	Avvio sequenza con loop gestito da Arduino
Precondizioni	Caricamento sketch su Arduino Modifica dei dati nel file CSV
Passaggi	1) Avvio dell'interfaccia sequenza 2) Premere il pulsante Send_Sequence
Risultati	Processing invia un set di parametri previsto da una sequenza letta da una file, questi dati vengono letti e eseguiti dalla scheda, al termine della sequenza Arduino inizierà a ripeterla in un loop finché non viene resettata la scheda.

5.1.2 Risultati grafici

In questo paragrafo vengono riportati alcuni risultati grafici, ottenuti inviando dei set di parametri alla scheda Arduino e leggendone i valori di ritorno nella sezione dei grafici dall'interfaccia.

- **Set 1:** Setpoint=100 P=1 I=0.8 D=0.1 Automatic Direct

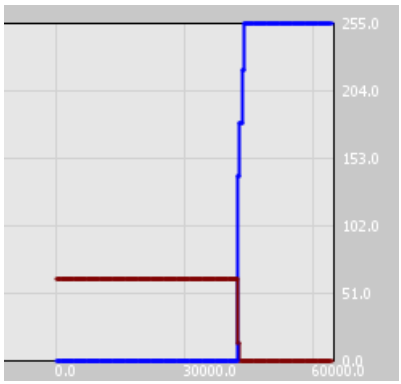

Figura 23 Risultato rispetto al set 1

- **Set 2:** Setpoint=100 P=0 I=0.01 D=0.01 Automatic Direct

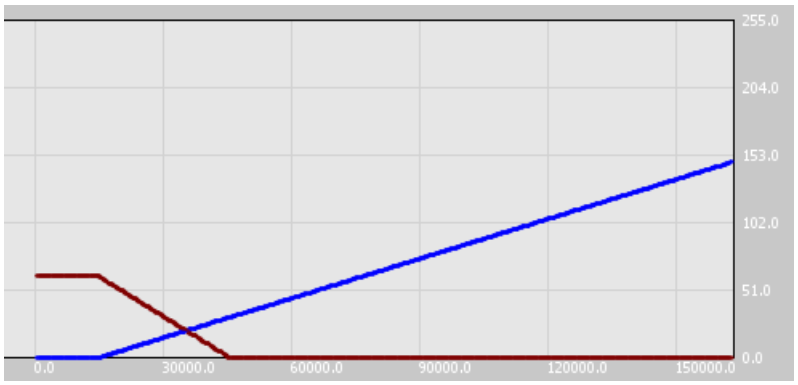

Figura 24 Risultato rispetto al set 2

- **Set 3:** Setpoint=100 P=0.5 I=0.5 D=0.2 Automatic Direct

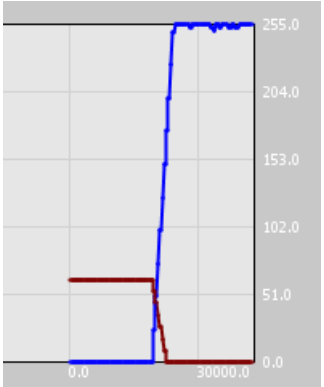

Figura 25 Risultato rispetto al set 3

- **Set 4:** Setpoint=100 P=0.2 I=0.05 D=0.02 Automatic Direct
  Setpoint=100 P=0.2 I=0.05 D=0.02 Automatic Reverse

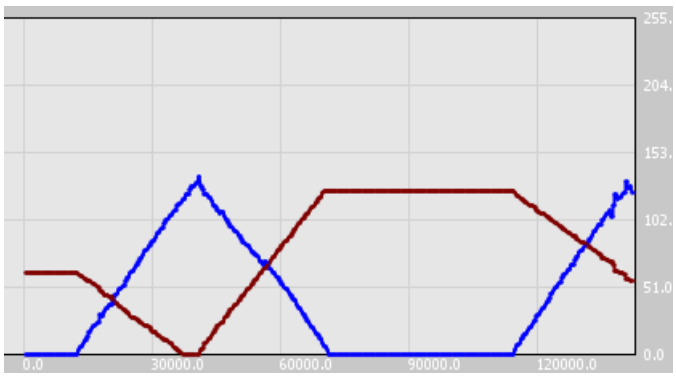

Figura 26 Risultato rispetto al set 4

Dai test effettuati si è riscontrato che con valori elevati il PID risponde sempre come uno scalino (ad esempio con P=4 D=0.5 I=0.5 o superiori).

5.1.3 Test con servo motore

Vengono qui descritti i test effettuati con il servo motore, per prima cosa viene illustrato il funzionamento e le caratteristiche del modello utilizzato per questo progetto, e successivamente sono presentati i risultati dei test ottenuti.

5.1.3.1 Introduzione al servo

I servo motori elettrici sono spesso utilizzati nella robotica per gli azionamenti, per il fatto che possono eseguire rotazioni a un angolo prestabilito e mantenerlo fino a un nuovo comando.

Si presentano come piccoli contenitori di materiale plastico da cui fuoriesce un perno in grado di ruotare in un angolo compreso tra 0 e 180°, mantenendo stabilmente la posizione raggiunta.

La rotazione del perno è ottenuta grazie all'uso di un motore a corrente continua e un meccanismo di de-moltiplica, il quale consente di aumentare la coppia in fase di rotazione.

La rotazione del motore è effettuata tramite un circuito di controllo interno in grado di rilevare l'angolo di rotazione raggiunto dal perno tramite un potenziometro resistivo e bloccare il motore sul punto desiderato.

In questo progetto è stato utilizzato il modello AR-3603HB Servo di Power HD, con le seguenti caratteristiche tecniche:

- *Torque(4.8V): 3.5 kg-cm (48.61 oz/in)*
- *Torque(6.0V): 4.4 kg-cm (61.10 oz/in)*
- *Speed: 0.14 sec (4.8V) │ 0.12 sec (6.0V)*
- *Operating Voltage：4.8 ~ 6.0 DC Volts*
- *Weight: 36.0 g (1.27 oz)*
- *Bearing Type：Ball Bearing x 2*
- *Motor Type：DC Motor*
- *Operating Temperature：-20℃~60℃*
- *Working frequence：1520μs / 50hz*

Figura 27 Servo motore

Per connettere il servo motore alla scheda Arduino basta connettere il filo rosso all'alimentazione da 5v, quello marrone alla terra e quello arancione al pin di output scelto (in questo progetto è stato utilizzato il pin numero 9), dato che è il cavo di controllo del servo.

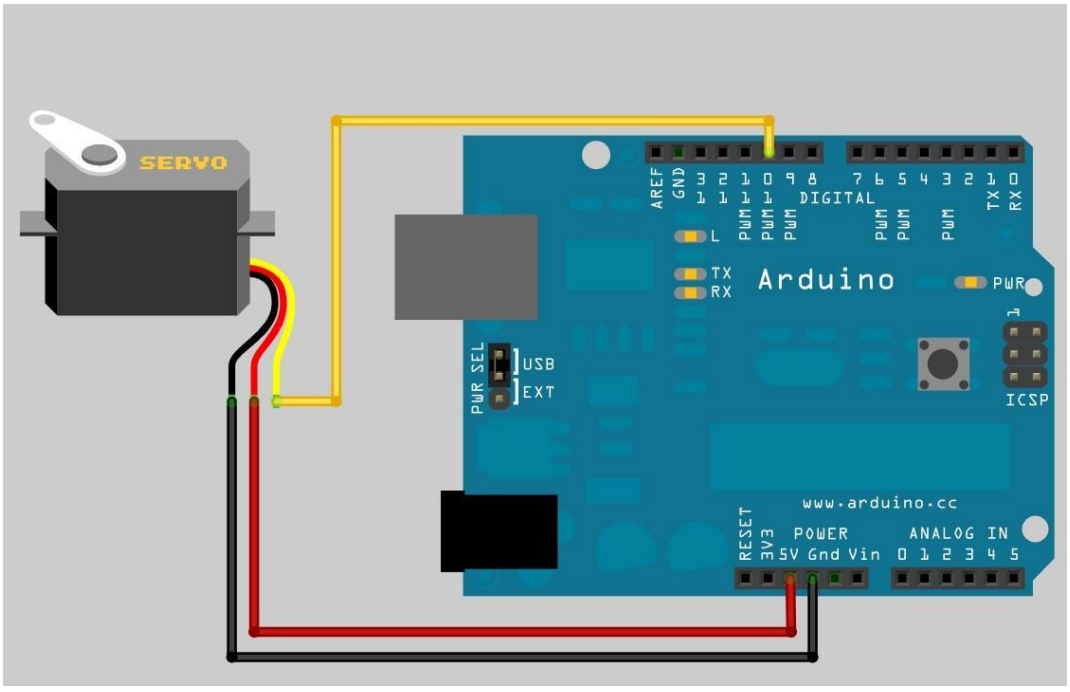

Figura 28 Collegamento Arduino Servo

5.1.3.2 Risultati dei test

Per testare il funzionamento del servo motore, è stato necessario impostare dei valori e dei controlli qui di seguito elencati, ovviamente sono scelte che possono esser modificate in base alle esigenze e situazioni:

- L'angolo di quiete del servo è settato a 63 gradi;

- Se l'output del PID va da 0→255 il servo motore inizia da una rotazione di 63 gradi fino ad arrivare a 0 gradi, ruotando in senso orario, oppure a un valore minimo impostato dall'utente;

- Se l'output del PID va da 255→0 il servo motore inizia da una rotazione di 63 gradi fino ad arrivare a 125 gradi, ruotando in senso anti orario, oppure a un valore massimo impostato dall'utente;

- Data la dimensione massima della memoria disponibile nella scheda Arduino utilizzata, per l'esecuzione della sequenza memorizzata da Arduino è stato imposto un valore massimo di 30 possibili set, questo per ovviare ai problemi di memoria;

Date queste impostazioni iniziali si sono effettuati i seguenti test, utilizzando diversi set di configurazioni di parametri, ma si è valutato la risposta del motore al valore di output del controllo PID:

Avvio del motore dall'interfaccia:

- **Risultato 1**: il motore si avvia pian piano seguendo il valore dell'output del PID, se questo è crescente, ad esempio una rampa, e raggiunge il valore massimo in pochi istanti;
- **Risultato 2**: il motore si avvia alla massima velocità, nel caso l'output del PID sia proporzionalmente più grande dell'angolo di velocità massima del servo;

Inversione di marcia invertendo il calcolo del PID:

- **Risultato 3**: l'inversione avviene senza problemi, indipendentemente dal senso di marcia che aveva in precedenza;
- **Risultato 4**: se il proporzionale è importato a una valore che genera uno scalino ampio l'inversione è istantanea;

Arresto del motore:

- **Risultato 5**: il motore si arresta completamente all'angolo di quiete in pochi istanti;

Riavvio dell'Arduino:

- **Risultato 6**: con motore in movimento arresta completamente immediatamente all'angolo di quiete;
- **Risultato 7**: con motore già fermo prima del reset non si nota nessun movimento;

Esecuzione di una sequenza:

- **Risultato 8**: il motore si avvia in risposta ai valori ricevuti, mantenendo anche l'angolo di rotazione definito nella sequenza, invertendo la rotazione quando richiesto;

6 Capitolo 6

6.1 Conclusioni e possibili sviluppi futuri

Dai risultati ottenuti e presentati nel precedente capitolo è possibile trarre le seguenti conclusioni. La prima è che l'interfaccia riesce a comandare correttamente la scheda Arduino e a inviare i valori dei parametri manualmente o attraverso i set caricati attraverso il file. La seconda è che la parte di implementazione del PID con la conversione proporzionale output/angolo per controllare il servo risulta coerente e corretta rispetto a quanto richiesto in fase di sviluppo.

Da ciò però è possibile definire dei sviluppi per miglioramenti futuri del progetto:

- Progettare un interfaccia indipendente dall'ambiente Processing, dato che il codice è Java, si potrebbe creare applicativi Java oppure che possano essere utilizzati tramite browser;

- Un nuovo sistema di connessione tra la scheda Arduino e l'interfaccia, ad esempio Wi-Fi, Bluetooth o altre connessioni supportabili e applicabili alla scheda, con la possibilità di utilizzare l'interfaccia anche in ambienti mobile (Android, IOS o Windows phone);

- Aumentare i comandi e le funzioni inserite nell'interfaccia, a seconda di nuove possibili esigenze;

Bibliografia e riferimenti.

1. Sito web ufficiale di Arduino
   http://arduino.cc/

2. Scheda tecnica di Arduino Uno
   http://www.atmel.com/Images/doc8161.pdf

3. Libreria PID per Arduino
   http://playground.arduino.cc/Code/PIDLibrary
   http://playground.arduino.cc/Code/PIDLibaryBasicExample

4. Tutorial di esempio di utilizzo di un servo motore con Arduino
   http://www.mauroalfieri.it/elettronica/tutorial-arduino-servo.html

5. Libreria Servo Adattata per comandarlo in velocità
   http://forum.arduino.cc/index.php?topic=61586.0

6. Sito Ufficiale Processing
   https://processing.org/

7. Esempio di moto browniano
   https://processing.org/examples/brownian.html

8. Libreria di Processing per la creazione di elementi grafici
   http://www.sojamo.de/libraries/controlP5/#installation

9. “Pratical Process Control” di Douglas J. Cooper
   http://www.controlguru.com/

10. “Exploring Classical Control” di E. J. Mastascusa
    http://www.facstaff.bucknell.edu/mastascu/econtrolhtml/CourseIndex.html

11. Enciclopedia Treccani storia Arduino d’Ivrea
    http://www.treccani.it/enciclopedia/arduino-re-d-italia/

12. Guida alla programmazione C
    http://www.html.it/guide/guida-c/

13. Esempio di creazione controllo PID Arduino
http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

Ringraziamenti

Per concludere questa tesi vorrei fare alcuni ringraziamenti, nei confronti di molte persone che mi hanno sostenuto e in parte aiutato.
Un mille grazie al prof. ***Bonarini,*** *che mi ha aiutato a concludere un percorso che si era perso in un tunnel senza uscita, mi ha dato l'opportunità e la possibilità di arrivare finalmente a una conclusione di questa mia lunga avventura al Politecnico.*
Successivamente voglio ringraziare tutta la mia famiglia, che nonostante i tanti problemi avuti negli ultimi anni, i tanti sacrifici soprattutto economici, e qualche occasione lavorativa mancata, mi hanno comunque sempre aiutato e sostenuto, senza mai lamentarsi del lungo periodo di tempo per arrivare a terminare questo seconda avventura universitaria.
Inoltre voglio ringraziare ***Fabrizio****, per le discussioni su Skype, per i consigli, anche per quei malumori e dubbi sul futuro che mi hanno aiutato a pensare, e al tempo stesso proseguire senza mollare.*
Voglio anche ringraziare gli amici che non mi hanno dato un aiuto diretto ma un sostegno morale mentre realizzavo questo lavoro, in particolare ***Paola****, che tra mille problemi e insonnie è sempre disposta a discutere, dare e accettare consigli,* ***Dark*** *e* ***Cri*** *che sono amici lontani, ma sanno sempre essere i più vicini nei momenti giusti, un grazie anche a* ***Giorgio*** *che è sempre pronto a cercare o recuperare qualcosa che possa interessarmi.*
Ancora un grazie tutti gli amici conosciuti nei tornei di giochi di carte, per il tempo passato insieme, in particolare ***Luca*** *e* ***Vale****,* ***Sca****,* ***Dosso****,* ***Magni****,* ***Gunny****,* ***Gatto****,* ***Fede****,* ***Tuzzo****,* ***Gaia*** *e molti altri ancora…*
Infine un grazie va anche a tutti i miei compagni di un università con cui ho ricordi indelebili di un viaggio insieme, tra lezioni stressanti, progetti che non terminano mai, esami fatti più volte e spesso non passati per mezzo punto, concludendo

…Grazie a tutti quanti…

Da parte di Andrea