



**UNIVERSITA' POLITECNICA DELLE MARCHE**  
**FACOLTA' DI INGEGNERIA**

---

Corso di Laurea triennale in Ingegneria Meccanica

**Prototipazione di un banco mecatronico per il sollevamento**

**Prototyping of a mechatronic testbench for lifting**

Relatore:

Prof. **Matteo Claudio Palpacelli**

Tesi di Laurea di:

**Francesco Ciarrocchi**

**A.A. 2021/2022**

## Sommario

<b>1 Introduzione</b> .....	<b>4</b>
1.1 Dati del progetto .....	5
1.2 Funzionamento dell'ascensore.....	5
<b>2 Scelta del motore</b> .....	<b>7</b>
2.1 Tipi di motore.....	7
2.2 Scelta effettiva del motore.....	8
2.3 Definizione puleggia.....	12
<b>3 Arduino</b> .....	<b>13</b>
3.1 Definizione scheda Arduino .....	14
3.2 Utilizzo Shield per collegare motore .....	15
<b>4 Progettazione motore tramite arduino</b> .....	<b>17</b>
4.1 Collegamento motore.....	18
4.2 Collegamento encoder.....	20
4.3 Sistema PID.....	22
4.4 Plot dati in Excel.....	27



# **CAPITOLO 1**

## **INTRODUZIONE AL PROGETTO**

Lo studio affrontato in questa tesi riguarda la progettazione di un banco di prova concentrandosi maggiormente sulla scelta del motore e sul suo funzionamento tramite la piattaforma Arduino.

Partendo quindi dalla definizione dei parametri di costruzione del sistema di sollevamento, ottenuti mediante criteri di scelta che hanno portato alla soluzione più consona, si è passati alla scelta del motore che abbia le caratteristiche necessarie alla realizzazione.

Il motore una volta selezionato è stato collegato alla scheda programmabile Arduino che con le dovute accortezze, e tramite l'ausilio di un alimentatore esterno, riesce a far girare il motore con una velocità costante e fino ad un punto prestabilito. Per far ciò si userà il software Arduino IDE che permette la scrittura di sketch appositi alla realizzazione del progetto.

## 1.1 DATI DI PARTENZA

Il banco di sollevamento funziona essenzialmente come un ascensore che ha il compito di portare un carico da un punto iniziale a uno finale programmato, mediante l'azione di un motore che fa ruotare una puleggia a cui è collegata una fune.

Inizialmente vanno decisi i pesi del carico, che essendo un banco di prova di piccole dimensioni, possono essere scelti relativamente piccoli sia il peso della cabina  $M_c = 0,1\text{kg}$  che quello del carico trasportabile  $M_u = 0,2\text{kg}$ . La massa del contrappeso può essere calcolata come:  $M_q = M_c + 0,4M_u$  che nel caso in esame sarà pari a  $M_q = 0,18\text{kg}$ .

## 1.2 FUNZIONAMENTO ASCENSORE

L'ascensore funzionerà a velocità costante ma soggetto a transitori in avviamento e in frenata, si studierà così sia il caso in transitorio che a regime.

Nel caso a regime dopo aver determinato se le condizioni di moto sono dirette o meno, si va ad osservare la parte dell'ascensore comprendente puleggia, cabina e contrappeso e se ne può ricavare il bilancio di potenza, valutando se  $W_r$  sia maggiore o minore di 0.

Analizzando i risultati si vedrà come il flusso sarà diretto in caso l'ascensore sia pieno in salita e vuoto in discesa, nei casi opposti ci sarà flusso retrogrado.

Con i dati a disposizione è possibile valutare la potenza minima che il motore deve avere per rendere possibile il sollevamento.

Essendoci una fase di accelerazione e di frenata occorrerà osservare anche le coppie e le inerzie in gioco, ciò verrà effettuato tramite il calcolo delle condizioni di moto. Prima cosa è analizzare se ci troviamo in condizioni di flusso diretto o meno, valutando i bilanci di potenza,

considerando questa volta anche le inerzie in gioco arriveremo alla considerazione di

$$W = \{(P - Pq) D/2 + [(M + Mq) D/4 + Jp]w'r\}wr \quad (1.1)$$

Si possono quindi ricavare le condizioni di moto per ognuno dei 4 casi:

Salita senza carico

Salita con carico

Discesa senza carico

Discesa con carico

Essenziale è inoltre la conoscenza di  $Cr$ :

$$Cr = (P - Pq) D/2 [Nm] \quad (1.2)$$

Che non è altro che la coppia richiesta dal carico, dato essenziale per la corretta scelta del motore per il progetto.

# CAPITOLO 2

## SCELTA DEL MOTORE

Il primo passo fondamentale per la realizzazione del processo riguarda la scelta del motore, esso per l'applicazione in questione non avrà bisogno di un funzionamento continuo, ma intermittente.

La caratteristica d'interesse è la massima coppia erogabile dal motore, che deve essere maggiore della coppia richiesta dal carico, pari a quella nel caso di ascensore pieno:  $C_r = 0,0352 \text{ Nm}$ .

Essenziale in un ascensore è che esso riesca ad andare da un punto A ad un punto B ad una velocità costante e con l'accelerazione adeguatamente scelta. Una precisione assoluta nello spostamento non è fondamentale, ma è {importante che essa non si discosti troppo dal target prestabilito.

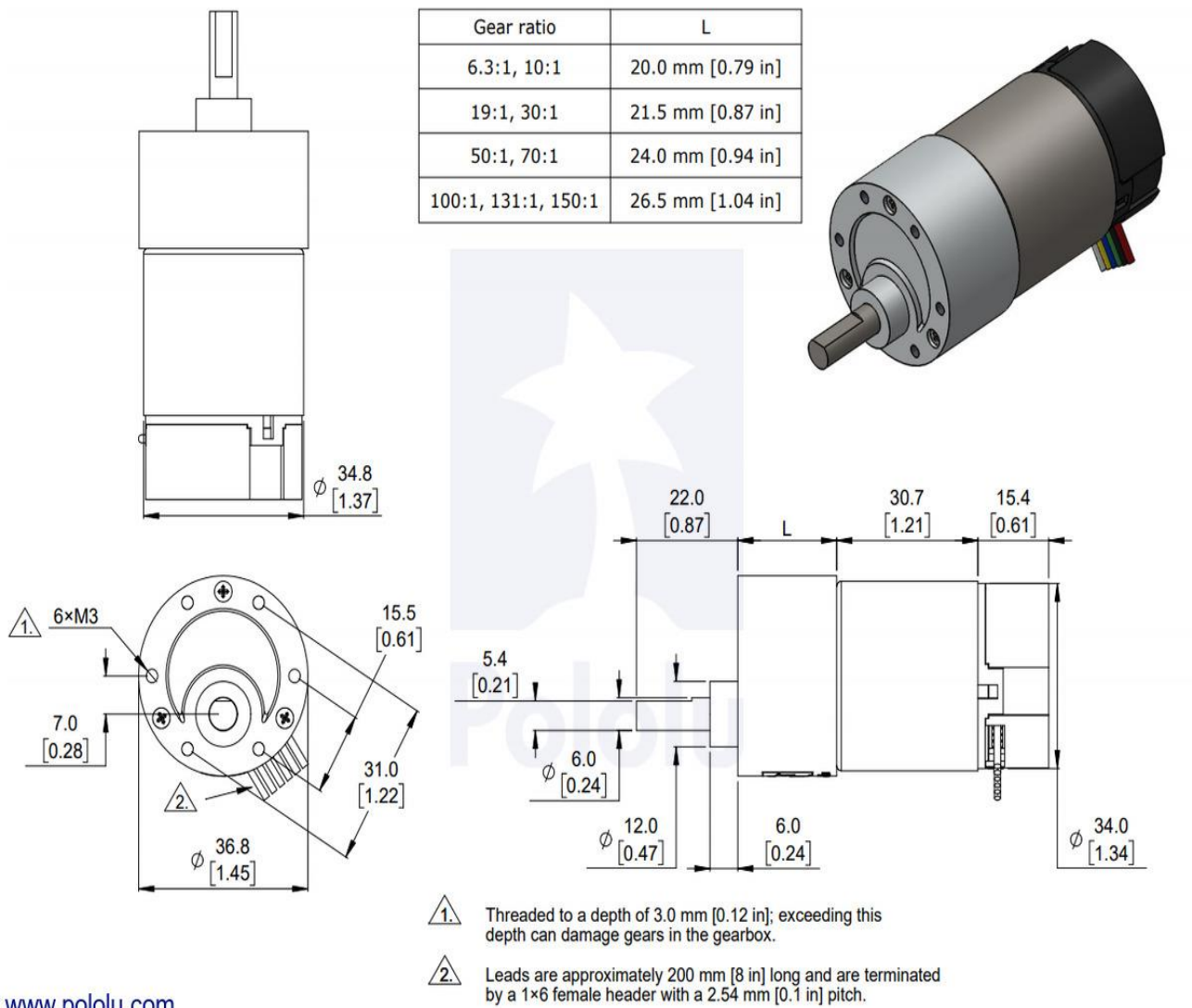
### 2.1 TIPO DI MOTORE

Per questa applicazione si è scelto di usare un motore brushed DC per via della possibilità di essere controllati in velocità, di essere poco costosi e possono essere regolamentati dalle schede Arduino. Altri motori buoni con cui è possibile svolgere la stessa applicazione sono i servomotori, anche essi compatibili con Arduino.

I motori brushed sono dei motori elettrici alimentati in corrente continua che trasformano l'energia elettrica in energia meccanica

## 2.2 SCELTA REALE DEL MOTORE

Dopo una ricerca su eventuali motori da utilizzare si è optato per un motore 37D 12V Brushed DC:



[www.pololu.com](http://www.pololu.com)

FIGURA 2.1: MOTORE 37D BRUSHED



Tra i vari motori a disposizione si è dovuto scegliere quello con il rapporto di trasmissione ideale ricordando che la coppia richiesta dal carico deve essere almeno pari a  $C_r = 0,0352 \text{ Nm}$ .

Rated Voltage	Stall Current	No-Load Current	Gear Ratio	No-Load Speed (RPM)	Extrapolated Stall Torque		Max Power (W)	Without Encoder	With Encoder
					(kg · cm)	(oz · in)			
12 V	5.5 A	0.2 A	1:1 (no gearbox)	10,000	0.5	7	–	–	<a href="#">item #4750</a>
			6.3:1	1600	3.0	42	12	<a href="#">item #4747</a>	<a href="#">item #4757</a>
			10:1	1000	4.9	68	12	<a href="#">item #4748</a>	<a href="#">item #4758</a>
			19:1	530	8.5	120	12	<a href="#">item #4741</a>	<a href="#">item #4751</a>
			30:1	330	14	190	12	<a href="#">item #4742</a>	<a href="#">item #4752</a>
			50:1	200	21	290	10	<a href="#">item #4743</a>	<a href="#">item #4753</a>
			70:1	150	27	380	10*	<a href="#">item #4744</a>	<a href="#">item #4754</a>

			100:1	100	34	470	8*	<a href="#">item #4745</a>	<a href="#">item #4755</a>
			131:1	76	45	630	6*	<a href="#">item #4746</a>	<a href="#">item #4756</a>
			150:1	67	49	680	6*		

*TABELLA 2.1: TABELLA CARATTERISTICHE MOTORE*

Dalla tabella soprariportata si può notare come qualsiasi motore a partire da un rapporto di trasmissione  $\tau = 6.3:1$  sia valido per l'applicazione.

Il motore da noi scelto sarà quello con un rapporto 50:1 dotato di encoder, questo perché il suo uso è essenziale, essendo un dispositivo elettromeccanico che converte la posizione angolare meccanica del suo asse rotante in posizione angolare elettrica sotto forma di segnale elettrico numerico digitale e/o analogico. L'encoder permette di poter osservare continuamente la posizione, e quindi la distanza percorsa dal motore durante il suo funzionamento.

L'albero in uscita è lungo 16cm con un diametro di 6cm ideale per calettare la puleggia.

Il motore e l'encoder sono dotati di 6 conduttori codificati a colori che dovranno essere collegati con la scheda elettronica. Nella tabella seguente si descrivono la funzione dei fili<sup>1</sup>:

<b>Color</b>	<b>Function</b>
Red	motor power (connects to one motor terminal)
Black	motor power (connects to the other motor terminal)
Green	encoder GND
Blue	encoder Vcc (3.5 – 20 V)
Yellow	encoder A output
White	encoder B output

*TABELLA 2.2 : TABELLA COLLEGAMENTO FILI MOTORE/ENCODER*

## 2.3 SCELTA PULEGGIA

Per far sì che il motore possa sollevare un oggetto tramite il suo funzionamento è necessario che si caletti una puleggia sull'albero di uscita del motore.

Le caratteristiche della puleggia scelta sono:

- Diametro primitivo  $D_p = 80\text{mm}$  valutato da uno studio precedente
- Gola ad U per aumentare la superficie d'attrito tra fune e puleggia
- Basso costo.

Scartate le varie pulegge disponibili nei cataloghi online, per via del loro elevato costo, si è optato per la realizzazione di essa tramite l'ausilio di una stampante 3D che ne permette una riuscita della puleggia grossomodo ottimale.

Per la sua realizzazione si è proceduto tramite il software SolidEdge alla sua creazione, rispettando i vari parametri che venivano imposti dalle dimensioni dell'asse di uscita. Quindi il foro doveva avere una dimensione  $D$  di 6mm e non doveva raggiungere una lunghezza maggiore dei 16mm.

La puleggia sarà inoltre dotata di un ulteriore foro nella parte superiore per permettere l'inserimento di una spina e il collegamento con l'albero.

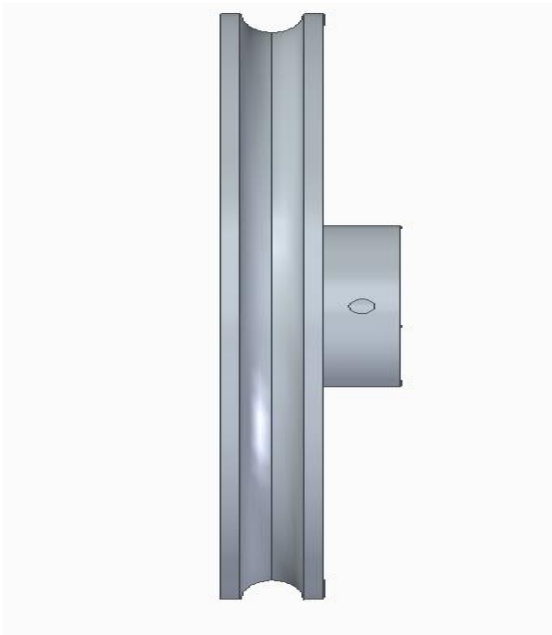


FIGURA 2.2: PULEGGIA

# **CAPITOLO 3**

## **ARDUINO**

Per regolare l'ascensore avremo bisogno di un sistema ad anello chiuso che ne regoli la velocità e la posizione.

### 3.1 DEFINIZIONE SCHEDA ARDUINO



*FIGURA 3.1: SCHEDA ARDUINO UNO*

ArduinoUno è una scheda programmabile dotata di un microcontrollore centrale e da pin analogici e digitali che permettono il controllo di diverse apparecchiature elettriche. Vi è inoltre la presenza di un cavo USB il quale permette di comunicare con il computer, essenziale, poiché sarà da esso tramite software che riusciremo a scrivere codici ed azionare il motore nelle condizioni desiderate.

La scheda viene alimentata da una tensione di ingresso dal valore compreso tra 7-12V, e può alimentare dispositivi mediante l'utilizzo di due uscite con valori di voltaggio pari a 3V e 5V.

## 3.2 UTILIZZO SHIELD PER COLLEGARE MOTORE

Per poter connettere il motore con l'Arduino però non basta la sola presenza della scheda, poiché il motore richiede una corrente e a volte una tensione maggiore di quella sopportabile dalla scheda.

Uno dei modi per poter effettuare il collegamento è dato dall'acquisto di un MotorShield per la scheda Arduino che ci permetterà di poter associare il motore alla scheda e di farlo funzionare. Tra i vari Shield a disposizione si è scelto il MotorShieldRev3:

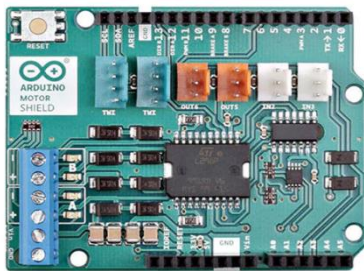


FIGURA 3.2: SHIELD ARDUINO UNO REV3

Esso è stato scelto innanzitutto poiché le sue caratteristiche di tensione ( 5V-12V) e di corrente massima sopportabile, fino a 2A se collegato con un alimentatore esterno il quale inoltre è l'unico modo per alimentarlo, erano quelle ottimali per la riuscita del processo.

Un altro motivo per la scelta di questo Shield è la presenza già inclusa dei pin che permettono il collegamento tra esso e la scheda semplicemente sovrapponendoli ed inserendo i pin correttamente.

Inoltre, è bene notare che lo Shield in questione è valido anche per la connessione di due motori, lavorando con un solo motore potremo scegliere in quale dei due canali A o B collegarlo. La scelta non produrrà alcuna differenza nella realizzazione del progetto, ma occorre tenere a mente che ogni canale ha quattro diversi pin a esso dedicato:

Function	pins per Ch. A	pins per Ch. B
<i>Direction</i>	D12	D13
<i>PWM</i>	D3	D11
<i>Brake</i>	D9	D8
<i>Current Sensing</i>	A0	A1

*TABELLA 3.1: TABELLA PIN DELLO SHIELD*

Quindi nel caso si scelga il canale B, gli unici pin da considerare sono il 13, l'11, l'8 e l'1. <sup>(2)</sup>



# **CAPITOLO 4**

## **PROGETTAZIONE MOTORE TRAMITE ARDUINO**

Fatte le dovute premesse riguardanti gli strumenti che si andranno effettivamente ad utilizzare per la realizzazione dell'ascensore, si andrà ora ad osservare nel dettaglio come avviene il collegamento tra il motore e la scheda elettronica tramite il software Arduino IDE.

Arduino IDE è un software gratuito che permette tramite la scrittura in Wiring, ovvero un derivato del linguaggio C++, di riuscire nella nostra realizzazione di regolare il motore e permettere il sollevamento di un oggetto di scarso peso.

## 4.1 COLLEGAMENTO MOTORE

Il primo passo nella realizzazione dell'ascensore è consistito nel collegamento del singolo motore con la scheda Arduino UNO per verificare l'effettivo funzionamento di esso, andando a constatare che avvenga la rotazione dell'asse di uscita del motore. Per far ciò viene inizialmente effettuato il collegamento manuale tra il motore e la scheda elettronica connettendo il filo rosso del motore nell'ingresso + e quello nero nel -, entrambi nel canale B.

Inoltre, bisogna connettere lo Shield all'alimentazione esterna poiché è l'unico modo per alimentare il motore. Collegati il polo positivo dell'alimentatore esterno al canale VIN e quello negativo in quello GND, situati vicino al canale A, si può procedere a compilare la parte IDE.

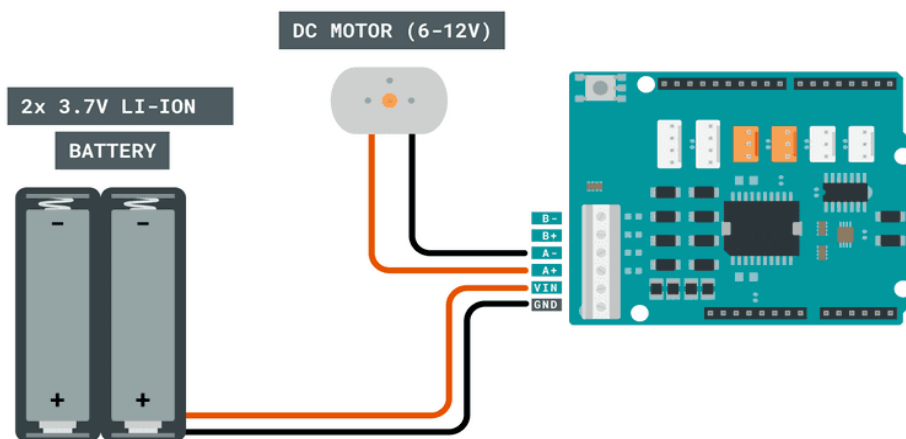


FIGURA 4.1: COLLEGAMENTO MOTORE SHIELD

Per far sì che il motore giri a vuoto, lo sketch da caricare in Arduino è relativamente semplice:

```
int directionPin = 13;  
int pwmPin = 11;
```

```

int brakePin = 8;

bool directionState;
void setup() {
pinMode(directionPin, OUTPUT);
pinMode(pwmPin, OUTPUT);
pinMode(brakePin, OUTPUT);
}

void loop() {

//change direction every loop
directionState = !directionState;

if(directionState == false){
digitalWrite(directionPin, LOW);
}

else{
digitalWrite(directionPin, HIGH);
}

digitalWrite(brakePin, LOW);

analogWrite(pwmPin, 30);

digitalWrite(brakePin, HIGH);

analogWrite(pwmPin, 0);
}

```

I vari significati dei codici scritti sopra possono essere ricavati semplicemente passando il cursore sopra essi una volta caricati su IDE, oppure per una conoscenza più approfondita si può sfruttare il sito di Arduino nella pagina dedicata: <https://www.arduino.cc/reference/en/> Andando a giocare con lo sketch si possono cambiare i valori nel comando analogWrite con un valore che va da 1 a 255 che ne determina la potenza del motore, maggiore sarà il numero scelto, maggiore sarà la potenza del motore. <sup>(3)</sup>

## 4.2 COLLEGAMENTO ENCODER

Appurato il funzionamento del motore è ora necessario far funzionare l'encoder posizionato sul motore, che ci permetterà di tener traccia dei movimenti compiuti dal motore DC e di poterlo stoppare nel momento più consono.

Come fatto per il motore, inizialmente occorre unire manualmente l'encoder alla scheda Arduino, andando a controllare la tabella 2 si otterrà un risultato come in figura:

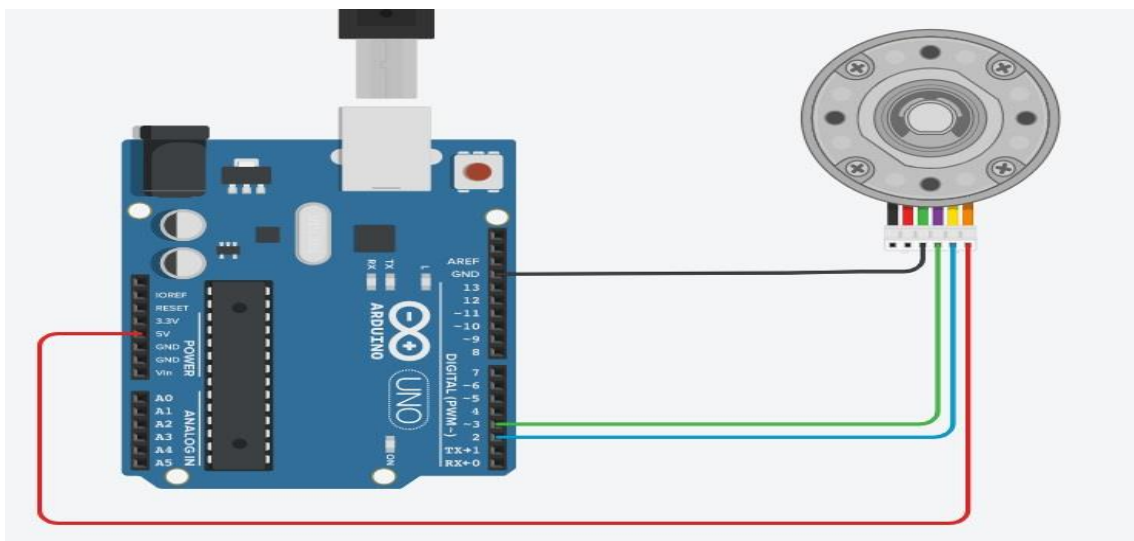


FIGURA 4.2: COLLEGAMENTO ENCODER SHIELD

L'encoder in questione è dotato di 2 output A e B, il loro innesco dipenderà dal verso di rotazione del motore (A orario, B antiorario), in questo modo si è sempre a conoscenza del verso di rotazione.

Per capire il funzionamento dell'encoder e verificare il suo funzionamento, occorre collegare la scheda elettronica al computer, e tramite IDE creare un nuovo sketch come riportato:

```
#define Encoder_output_A 2
#define Encoder_output_B 3

int Count_pulses = 0;
```

```

void setup() {
  Serial.begin(9600);
  pinMode(Encoder_output_A, INPUT);
  pinMode(Encoder_output_B, INPUT);
  attachInterrupt(digitalPinToInterrupt(Encoder_output_A), DC_Motor_Encoder,
  RISING);
}

void loop() {
  Serial.println("Result: ");
  Serial.println(Count_pulses);
}

void DC_Motor_Encoder(){
  int b = digitalRead(Encoder_output_B);
  if(b > 0){
    Count_pulses++;
  }
  else{
    Count_pulses--;
  }
}

```

(4)

Importante notare il Serial.begin da impostare pari a 9600 baud che rappresenta la velocità di trasmissione tipica per comunicare con il pc.

Tramite lo sketch riportato sopra, senza aver collegato il motore, basterà aprire il serial monitor all'interno del IDE e verificare che il solo roteare l'encoder manualmente comporterà un aumento del valore nel monitor seriale o un suo diminuitamento, in base al verso di rotazione impostato.

Se sia il motore che l'encoder funzionano correttamente si può procedere con la connessione di entrambi contemporaneamente. Per far ciò è però necessario far uso di un sistema di controllo PID.

### 4.3 SISTEMA PID

Il controllo PID è un sistema che permette di gestire una grandezza di processo tenendola sotto controllo.

Il PID essenzialmente è un algoritmo che riceve in ingresso i valori di processo da analizzare ed il set-point, ovvero il valore desiderato della grandezza da controllare. In uscita si avrà quindi un valore utilizzato da attuatore che influenzerà la variabile d'ingresso per riportarla al set-point desiderato. La comparazione tra il set-point e la grandezza in tempo reale ne farà derivare un errore denominato  $e(t)$ .

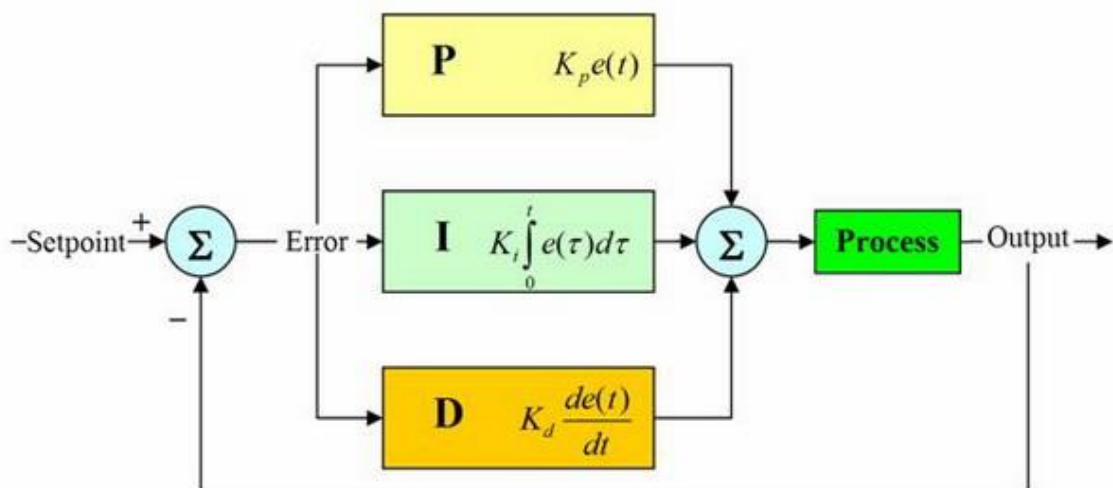


FIGURA 4.3: SCHEMA PID

Il PID utilizza quindi tre sistemi:

- Proporzionale che valuta la differenza tra il set-point ed il valore di processo amplificandone la differenza tramite un fattore moltiplicativo  $K_p$ .
- Integrativo che tiene conto nel tempo integrando. Questa componente è più o meno veloce a seconda del valore di  $K_i$  che viene scelto
- Derivato tiene conto della velocità con cui varia la grandezza ed ha il compito di reagire per contrastarla tramite la componente  $K_d$ .

Non necessariamente tutte e tre le componenti devono essere presenti durante un sistema PID, infatti le componenti Ki o Kd possono essere uguali a 0 creando così sistemi PD o PI.

Una volta collegato il motore e l'encoder all' ArduinoUNO e fatto partire l'IDE, è possibile scrivere uno sketch che permetterà il funzionamento desiderato del progetto:

```
#define ENCA 3
#define ENCB 2
#define PWM 11
#define IN2 13
#define IN1 8

int i = 0;
int pos = 0;
long prevT = 0;
float eprev = 0;
float eintegral = 0;

void setup() {
  Serial.begin(9600);
  pinMode(ENCA,INPUT);
  pinMode(ENCB,INPUT);
  attachInterrupt(digitalPinToInterrupt(ENCA),readEncoder,RISING);
  Serial.println("target pos");
  Serial.println("CLEARDATA");
  Serial.println("LABEL,Ora,target,pos,currT");
}
void loop() {

  // set target position
  int target = 1000;

  // PID constants
  float kp = 8;
  float kd = 1.5;
  float ki = 0;

  // time difference
  long currT = millis();
  float deltaT = ((float) (currT - prevT))/( 1.0e3 );
  prevT = currT;
  // error
  int e = target-pos;
  // derivative
```

```

float dedt = (e-eprev)/(deltaT);
// integral
eintegral = eintegral + e*deltaT;
// control signal
float u = kp*e + kd*dedt + ki*eintegral;
// motor power
float pwr = fabs(u);
if( pwr > 255 ){
    pwr = 255;
}
// motor direction
int dir = 1;
if(u<0){
    dir = -1;
}
// signal the motor
setMotor(dir,pwr,PWM,IN1,IN2);
// store previous error
eprev = e;
Serial.print("DATA");
Serial.print(",");
Serial.print("TIME");
Serial.print(",");
Serial.print(target);
Serial.print(",");
Serial.print(pos);
Serial.print(",");
Serial.print(currT);
Serial.println();
}

void setMotor(int dir, int pwmVal, int pwm, int in1, int in2){
    analogWrite(pwm,pwmVal);
    if(dir == 1){
        digitalWrite(in1,HIGH);
        digitalWrite(in2,LOW);
    }
    else if(dir == -1){
        digitalWrite(in1,LOW);
        digitalWrite(in2,HIGH);
    }
    else{
        digitalWrite(in1,LOW);
        digitalWrite(in2,LOW);
    }
}
}

```



```

void readEncoder(){
  int b = digitalRead(ENCB);
  if(b > 0){
    pos++;
  }
  else{
    pos--;
  }
}

```

(4)

Lo sketch sopra riportato è quello definitivo. Inizia tramite una definizione dei pin utilizzati nella scheda tramite il comando #define. Successivamente all'interno del void loop ci sono i vari parametri da dover andare a modificare.

Innanzitutto, vi è il target, che una volta stabilito farà girare il motore fino al raggiungimento di esso. La definizione del target è di grande importanza perché in un ascensore è essenziale che esso arrivi da un punto iniziale ad uno finale anche se con un minimo margine di errore.

Successivamente si è definito l'errore "e" come la differenza tra il target e la posizione, e poi la differenza di tempo in millisecondi che permette di valutare il tempo necessario a far sì che il motore raggiunga il target desiderato. Altri parametri che sono di notevole importanza sono:

$$dedt = (e - e_{prev}) / (\delta T) \quad (4.1)$$

e

$$eintegral = eintegral + e * \delta T \quad (4.2)$$

I valori sopra riportati, sommati tra loro e moltiplicati per i coefficienti del PID (Ki, Kd, Kp) daranno il valore dell'output u:

$$u = Kp * e + Kd * dedt + Ki * eintegral \quad (4.3)$$

(4)

Ulteriori considerazioni riguardano la possibilità di poter cambiare a proprio piacimento i valori dei coefficienti  $K_i$ ,  $K_d$ ,  $K_p$ , semplicemente modificandone i valori nello sketch.

Cambiare i valori dei coefficienti permette il variare la curva che porta al raggiungimento del target prestabilito, tramite una fase di studio denominata tuning in cui si andranno a variare i valori delle costanti fino all'ottenimento di una curva buona per lo scopo.

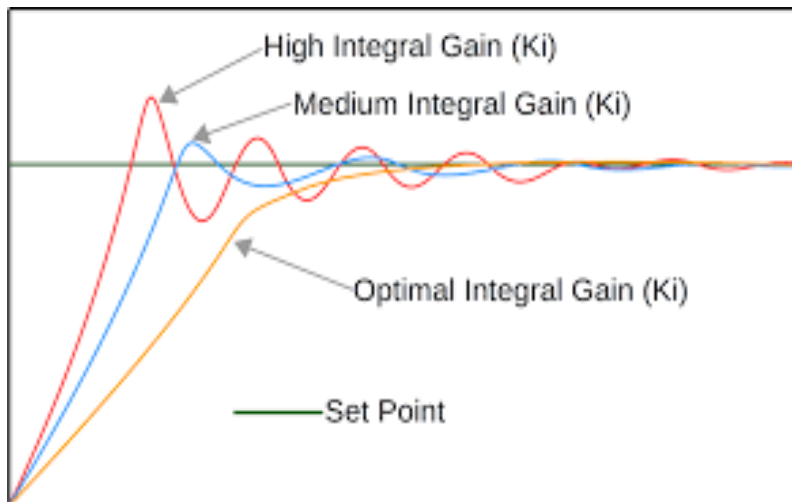


FIGURA 4.4: CURVE OTTENUTE TRAMITE PID

Ogni diverso parametro influisce diversamente nella realizzazione della curva, ad esempio aumentare  $K_p$  produce un valore di overshoot maggiore, ovvero il raggiungimento e il superamento del valore del target preimpostato, nel mentre  $K_d$  ne va a diminuire il raggiungimento, creando così una curva Medium Integral Gain riportata nel grafico che rappresenta una curva che parte da 0, raggiunge e supera il target per poi oscillare nell'intorno di esso.

I valori scelti  $K_p = 8$ ;  $K_d = 1.5$  e  $K_i = 0$  sono dei buoni valori che permettono si un piccolo overshoot non eccessivo, ma vi è un valore  $K_d$  tale da poter stabilizzare il valore della curva attorno al target selezionato con uno smorzamento non eccessivo.

## 4.4 PLOT DATI SU EXCEL

Una volta fatto partire lo sketch nel software IDE, nonostante la presenza di un serial monitor e di un plotter all'interno di esso, si nota che essi non siano sufficienti per una corretta valutazione dell'effetto dei parametri PID anche per via dell'impossibilità nel salvare i dati calcolati.

Occorre così visualizzare i dati al di fuori dell'IDE tramite l'ausilio di uno dei tanti programmi disponibili per questa funzione.

Il programma scelto per permettere ciò, è stato PLX-DAQ<sup>(5)</sup>, un programma che permette di trasferire i dati dall'Arduino ad un file Excel semplicemente collegandosi alla stessa porta seriale con cui è collegata la scheda elettronica.

È bene notare come i dati riportati sul file sono sempre scelti a piacimento e si possono cambiare andando a modificare il codice. Si è scelto di riportare per questa applicazione i dati riguardanti la pos, ovvero la posizione del albero di uscita del motore calcolata tramite l'utilizzo dell'encoder durante il funzionamento del motore, ed il currT che sta ad indicare il tempo di ciclo calcolato in microsecondi.

I dati visualizzati saranno gli stessi che si possono osservare tramite il serial Monitor, ma con la possibilità di salvarli e di rendere più facile il confronto delle curve generate nel caso di ulteriori prove di modifica dei parametri PID.

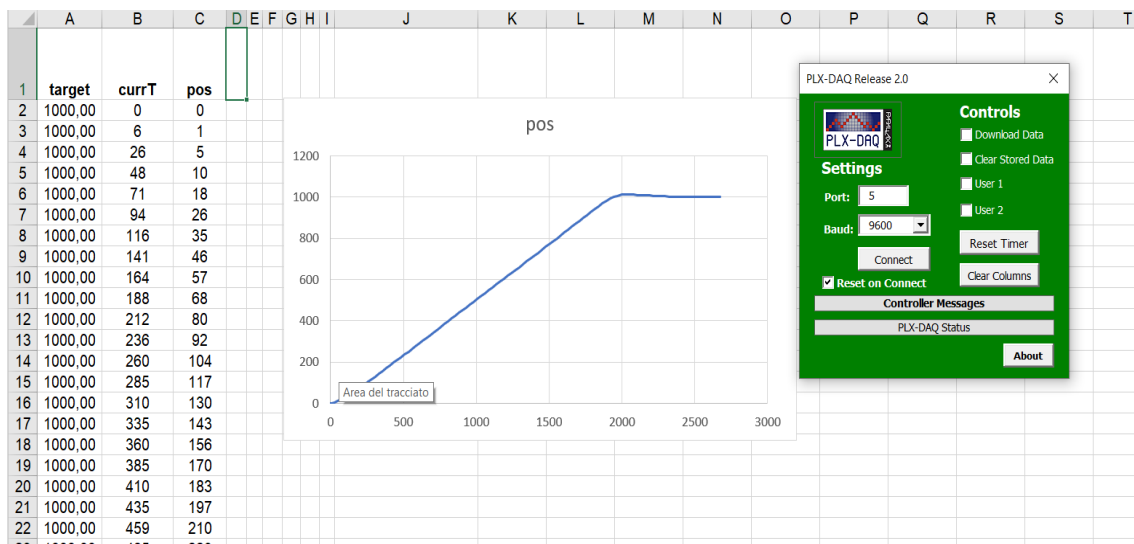


FIGURA 4.5: APPLICAZIONE EXCEL

# BIBLIOGRAFIA

FIGURA 2.1: <https://www.pololu.com/product/4755>

TABELLA 2.1: <https://www.pololu.com/product/4755>

TABELLA 2.2: <https://www.pololu.com/product/4755>

(1): <https://www.pololu.com/product/4755>

FIGURA 3.2 : [https://store.arduino.cc/products/arduino-motor-shield-rev3?\\_gl=1%2Av2jeqt%2A\\_ga%2AMTI4NjU5OTU5NS4xNjcxMTg2ODg4%2A\\_ga\\_NEXN8H46L5%2AMTY3NjE5OTk3MC4xNi4xLjE2NzYxOTk5NzMuMC4wLjA](https://store.arduino.cc/products/arduino-motor-shield-rev3?_gl=1%2Av2jeqt%2A_ga%2AMTI4NjU5OTU5NS4xNjcxMTg2ODg4%2A_ga_NEXN8H46L5%2AMTY3NjE5OTk3MC4xNi4xLjE2NzYxOTk5NzMuMC4wLjA).

Tabella 3.1 : [https://store.arduino.cc/products/arduino-motor-shield-rev3?\\_gl=1%2Av2jeqt%2A\\_ga%2AMTI4NjU5OTU5NS4xNjcxMTg2ODg4%2A\\_ga\\_NEXN8H46L5%2AMTY3NjE5OTk3MC4xNi4xLjE2NzYxOTk5NzMuMC4wLjA](https://store.arduino.cc/products/arduino-motor-shield-rev3?_gl=1%2Av2jeqt%2A_ga%2AMTI4NjU5OTU5NS4xNjcxMTg2ODg4%2A_ga_NEXN8H46L5%2AMTY3NjE5OTk3MC4xNi4xLjE2NzYxOTk5NzMuMC4wLjA).

(2): [https://store.arduino.cc/products/arduino-motor-shield-rev3?\\_gl=1%2Av2jeqt%2A\\_ga%2AMTI4NjU5OTU5NS4xNjcxMTg2ODg4%2A\\_ga\\_NEXN8H46L5%2AMTY3NjE5OTk3MC4xNi4xLjE2NzYxOTk5NzMuMC4wLjA](https://store.arduino.cc/products/arduino-motor-shield-rev3?_gl=1%2Av2jeqt%2A_ga%2AMTI4NjU5OTU5NS4xNjcxMTg2ODg4%2A_ga_NEXN8H46L5%2AMTY3NjE5OTk3MC4xNi4xLjE2NzYxOTk5NzMuMC4wLjA).

Figura 4.1: [https://docs.arduino.cc/tutorials/motor-shield-rev3/msr3-controlling-dc-motor?queryID=ade426cfbff2dd968e0f1146fdda4939&\\_gl=1\\*511zj7\\*\\_ga\\*MTI4NjU5OTU5NS4xNjcxMTg2ODg4\\*\\_ga\\_NEXN8H46L5\\*MTY3NTI1ODA3MC41LjEuMTY3NTI1ODM3OC4wLjAuMA..](https://docs.arduino.cc/tutorials/motor-shield-rev3/msr3-controlling-dc-motor?queryID=ade426cfbff2dd968e0f1146fdda4939&_gl=1*511zj7*_ga*MTI4NjU5OTU5NS4xNjcxMTg2ODg4*_ga_NEXN8H46L5*MTY3NTI1ODA3MC41LjEuMTY3NTI1ODM3OC4wLjAuMA..)

(3): [https://docs.arduino.cc/tutorials/motor-shield-rev3/msr3-controlling-dc-motor?queryID=ade426cfbff2dd968e0f1146fdda4939&\\_gl=1\\*511zj7\\*\\_ga\\*MTI4NjU5OTU5NS4xNjcxMTg2ODg4\\*\\_ga\\_NEXN8H46L5\\*MTY3NTI1ODA3MC41LjEuMTY3NTI1ODM3OC4wLjAuMA..](https://docs.arduino.cc/tutorials/motor-shield-rev3/msr3-controlling-dc-motor?queryID=ade426cfbff2dd968e0f1146fdda4939&_gl=1*511zj7*_ga*MTI4NjU5OTU5NS4xNjcxMTg2ODg4*_ga_NEXN8H46L5*MTY3NTI1ODA3MC41LjEuMTY3NTI1ODM3OC4wLjAuMA..)

FIGURA 4.2 : <https://www.electronicclinic.com/arduino-dc-motor-speed-control-with-encoder-arduino-dc-motor-encoder/>

(4): <https://www.electronicclinic.com/arduino-dc-motor-speed-control-with-encoder-arduino-dc-motor-encoder/>

FIGURA 4.4 : <http://www.micheleardito.info/ma/it/arduino-it/controllo-pid-arduino-on-off/>

(5): <https://www.parallax.com/package/plx-daq/>

