ISTITUTO TECNICO INDUSTRIALE DI STATO
P. PALEOCAPA
24125 BERGAMO – Via Gavazzeni, 29 – Tel. 035/31.93.88 – Fax 035/31.87.41 – C.F. 80025490162
www.itispaleocapa.it – bgtf010003@pec.istruzione.it – segreteria@itispaleocapa.it
QAIC / IT / 91838 - A
CORSI DIURNI ELETTRONICA ED ELETTROTECNICA – INFORMATICA – MECCANICA MECCATRONICA ED ENERGIA – SISTEMA MODA
CORSI SERALI ELETTRONICA ED ELETTROTECNICA – INFORMATICA – MECCANICA MECCATRONICA ED ENERGIA
ESPERIA INDUSTRIAL MUSEUM

ANALISI DI UN ROBOT AUTO BILANCIANTE

Tesina di maturità

Candidato: Haddaoui Hamza
Classe: 5ª elettronica

Anno scolastico 2016/17

Sommario

In occasione dell'esame di maturità, data la mia passione per la robotica, ho deciso di dedicarmi allo studio e alla realizzazione di un robot auto bilanciante, un particolare automa in grado di mantenersi in equilibrio su due ruote, riuscendo a vincere la propria condizione di instabilità. Infatti è come se noi volessimo tenere in posizione verticale una matita sulla punta del dito.

La condizione di equilibrio viene realizzata mediante l'uso di sensori in grado di rilevare, istante per istante, l'angolazione rispetto al piano. La stabilità è garantita con l'utilizzo di motori necessari a ristabilire la condizione di equilibrio, tramite piccoli movimenti per mantenere il baricentro entro la base di appoggio.

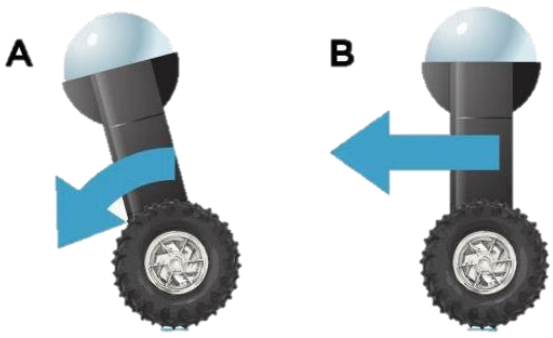
Lo scopo di tale progetto dunque è realizzare un sistema di controllo efficiente ed adeguato per il bilanciamento del robot, basandosi sul principio di funzionamento del pendolo inverso.

Il lavoro presentato riguarda l'intero sviluppo del progetto, a partire dalla costruzione meccanica, dalla scelta dei componenti elettronici, alla progettazione di un software di controllo adeguato.

Indice

1. Introduzione

Nell'ambito della robotica, negli ultimi anni sono stati fatti notevoli passi avanti. Oggi, la ricerca di tecniche per la costruzione di robot che possano interagire con l'uomo è un argomento di studio molto diffuso. I primi robot costruiti erano basati su strutture elementari rette da tre o più ruote. Successivamente, cominciarono a comparire i primi robot bipedi.



La realizzazione del robot si è articolata in varie fasi. Dopo una prima analisi prettamente teorica del progetto, ho realizzato la struttura meccanica.

Il progetto elettronico ha riguardato la scelta dei sensori, in particolare accelerometro e giroscopio per il calcolo dell'angolazione, della logica di controllo, basata su un microcontrollore PIC18 e dell'elettronica di potenza necessaria per il pilotaggio dei motori.

Terminata la parte hardware ho successivamente realizzato l'algoritmo necessario per il funzionamento dell'intero sistema, a partire dall'acquisizione e filtraggio dei dati sensoriali, all'implementazione del controllore PID[1].

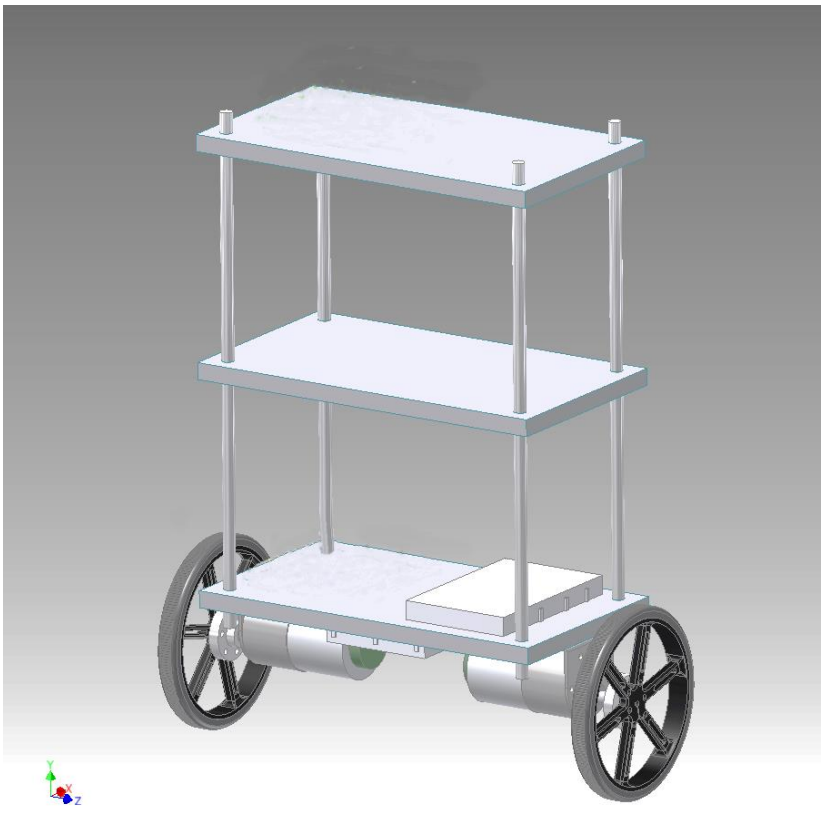
[1] Controllore Proporzionale, Integrativo, Derivativo

2. Progetto meccanico

2.1 Telaio

La prima fase del progetto ha riguardato la scelta sia della struttura, che dei materiali con cui costruire il robot. Ho deciso di utilizzare+ un materiale leggero e resistente al tempo stesso, facilmente lavorabile, come il polimetilmetacrilato[2] verso il quale si è indirizzata la scelta definitiva.

La struttura ideata è rettangolare e basata su tre piani; sul piano inferiore sono collocati i motori, le batterie e il sensore, mentre il piano intermedio ospita l'elettronica di controllo e di potenza. Il piano superiore invece è lasciato libero, per espansioni future quali l'implementazione di sensori per la navigazione autonoma del robot. Le lastre di polimetilmetacrilato hanno una dimensione di 150x100 mm, mentre l'altezza totale è di 30 cm. Il peso complessivo della struttura è di circa 0,84 kg.



[2] Plexiglass

2.2 Motori

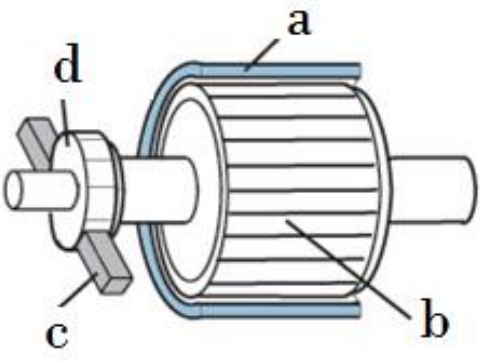
La movimentazione del robot è garantita dall'utilizzo di una coppia di motori in corrente continua, che offrono semplicità di utilizzo, possibilità regolazione della velocità e basso costo.

Ho selezionato dei motoriduttori[3] 47:1 Pololu, con encoder integrato per il monitoraggio della velocità e del senso di rotazione. La coppia dei motori scelti è di 1,177 Nm mentre la tensione di funzionamento è 12V. La velocità nominale è di 220 giri al minuto.

2.2.1 Cenni teorici

Il motore DC[4] è un dispositivo, che trasforma l'energia elettrica in energia meccanica. Il principio di funzionamento di un motore DC è basato sull'interazione di due campi magnetici che si attraggono e respingono a vicenda.

Esso è costituito da due parti principali: uno statore (*a*), costituito da magneti permanenti o elettromagneti, che induce un campo magnetico, e dal rotore (*b*), l'indotto soggetto al campo magnetico, costituito da una serie di spire solidali con l'albero rotante.

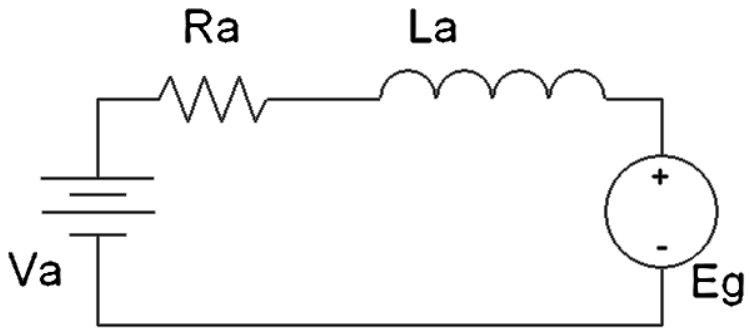


Il collegamento elettrico con l'alimentazione è costituito da due spazzole striscianti (*c*) che costituiscono la parte più delicata del motore. Si tratta di cilindri in carbone che strisciano sul collettore (*d*).

[3] Motore la cui potenza trasmessa al carico viene ridotta tramite ingranaggi, a scapito della velocità

[4] "Direct Current" – corrente continua

Il circuito equivalente di un motore DC è costituito da tre elementi quali una resistenza *Ra,* un'induttanza *La* e un generatore *Eg (Forza elettromotrice indotta).*



Le equazioni che regolano il comportamento del motore sono le seguenti:

$$V_a = R_a \cdot I_a + L_a * \frac{dI_a}{dt} + E_g$$

$$E_g = k_e * \omega$$

$$C_m = k_t * I_a$$

Va	Tensione di alimentazione
Ia	Corrente di alimentazione
Ra	Resistenza di armatura
La	Induttanza di armatura
Ke	Costante elettrica (parametro del motore)
Kt	Costante di coppia (parametro del motore)
Cm	Coppia meccanica del motore
ω	Velocità motore (rad/s)

Dall'analisi delle equazioni descritte precedentemente deduciamo che quando il motore è fermo, quindi nella situazione iniziale, la corrente è e pari a Va/Ra e di conseguenza anche la coppia è massima. Questa situazione si presenta all'avvio del motore, quando viene assorbita una grande quantità di corrente (corrente di spunto), oppure se il carico collegato al motore esprime una forza resistente troppo elevata.

Viceversa quando la velocità è massima, Eg è pari alla Va e di conseguenza la corrente dovrebbe essere nulla. Tuttavia questa situazione è teorica, in quanto una piccola coppia resistente è sempre presente a causa degli attriti meccanici.

3. Progetto elettronico

Una volta realizzata la struttura, ho progettato e realizzato la scheda madre del robot e successivamente ho selezionato i componenti necessari per il funzionamento del sistema.

La logica di controllo è affidata ad un microcontrollore PIC18F2580 che si occupa dell'acquisizione dei dati da parte dei sensori, dell'elaborazione dell'angolo e della velocità, e del calcolo di una risposta per i motori al fine di ristabilire l'equilibrio del mezzo.

Il sensore che ho scelto per la misura dell'angolazione è un MPU-6050; si tratta di un accelerometro a tre assi, combinato con un giroscopio a tre assi, per un totale di 6 DOF[5]. Il sensore comunica con il microcontrollore tramite protocollo I2C[6].

Per la misura della velocità dei motori, necessaria per il funzionamento del controllo ad anello chiuso del controllore PID, ho deciso di utilizzare gli encoder ad effetto Hall già montati sui motori.

I motori vengono pilotati con un driver L298N, che non è altro che un ponte H, che permette di invertire la polarità del motore stesso, vale a dire cambiare il verso di rotazione da orario ad antiorario e viceversa e variarne la velocità.

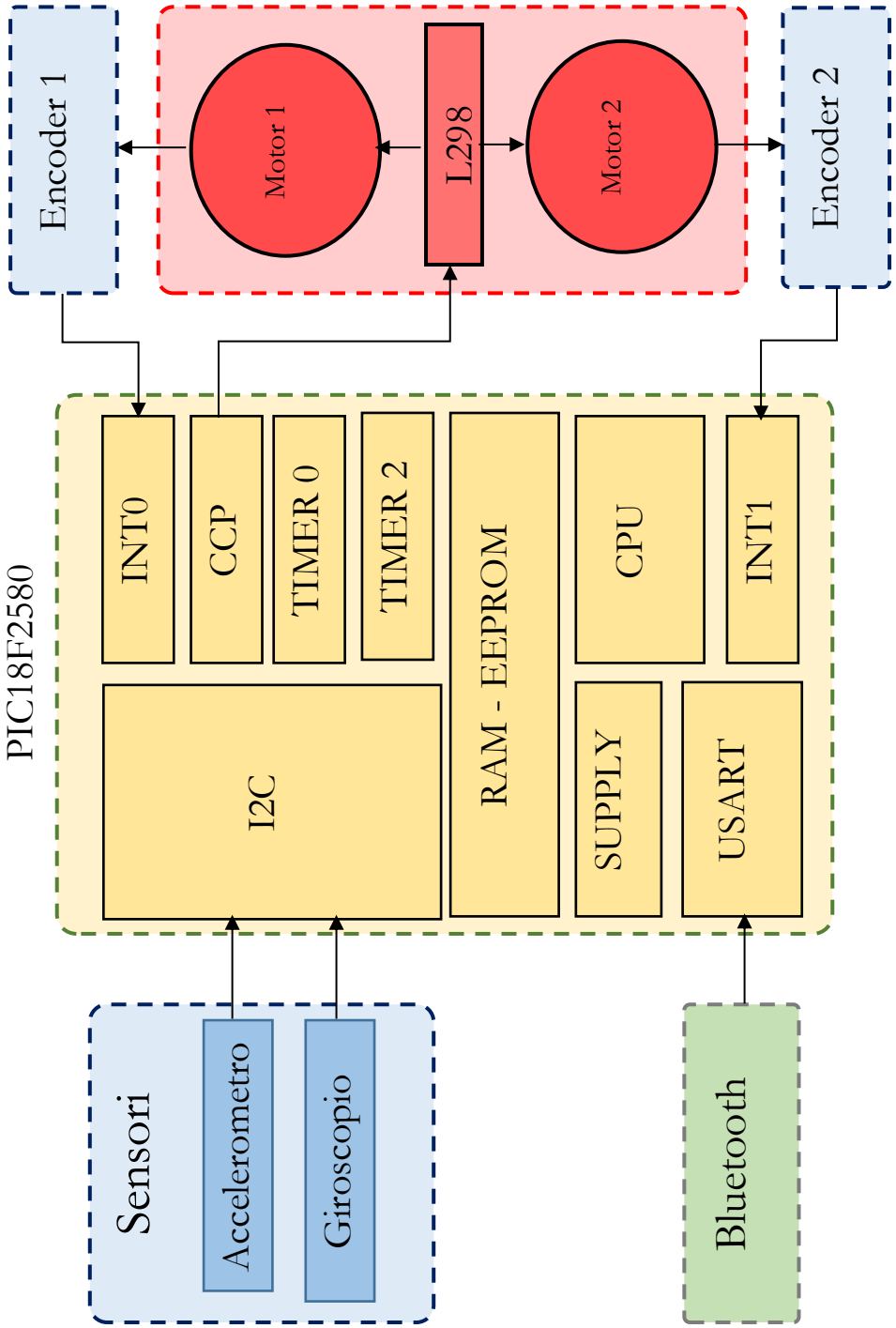
Ho inoltre aggiunto un modulo HC-06 per la comunicazione via Bluetooth del droide con un PC (oppure con uno smartphone) per il debug, ma anche per lo scambio di dati di calibrazione.

Infine ho dovuto scegliere come alimentare il robot, ed ho optato per delle batterie al litio con una tensione nominale di 14,8V e una capacità di 2600mAh. Ho dovuto inoltre utilizzare dei regolatori di tensioni sia per i motori (che funzionano a 12V) che per la sezione logica (che funziona a 5V).

Nelle sezioni seguenti descriverò più dettagliatamente i vari dispositivi utilizzati.

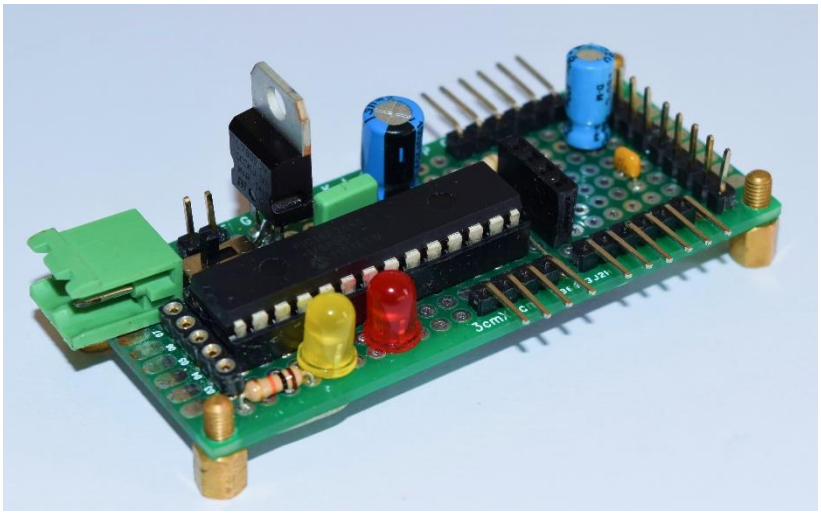
[5] Degrees Of Freedom – Gradi di libertà
[6] Inter Integrated Circuits – Protocollo di comunicazione basato su due linee (SDA, SCL)

3.1 Schema a blocchi



3.2 Scheda madre

La parte principale del robot è la scheda madre, sulla quale risiede il microcontrollore. La scheda madre ospita inoltre i connettori per il collegamento dei vari sensori, dei motori, del modulo Bluetooth e del driver dei motori. Sono inoltre presenti due led, uno per la segnalazione dell'accensione della scheda e l'altro programmabile via software e un regolatore lineare di tensione a 5V per l'alimentazione logica. La programmazione del microcontrollore avviene attraverso un connettore a 5 poli al quale viene collegato un PICkit2[7].



3.2.1 Microcontrollore

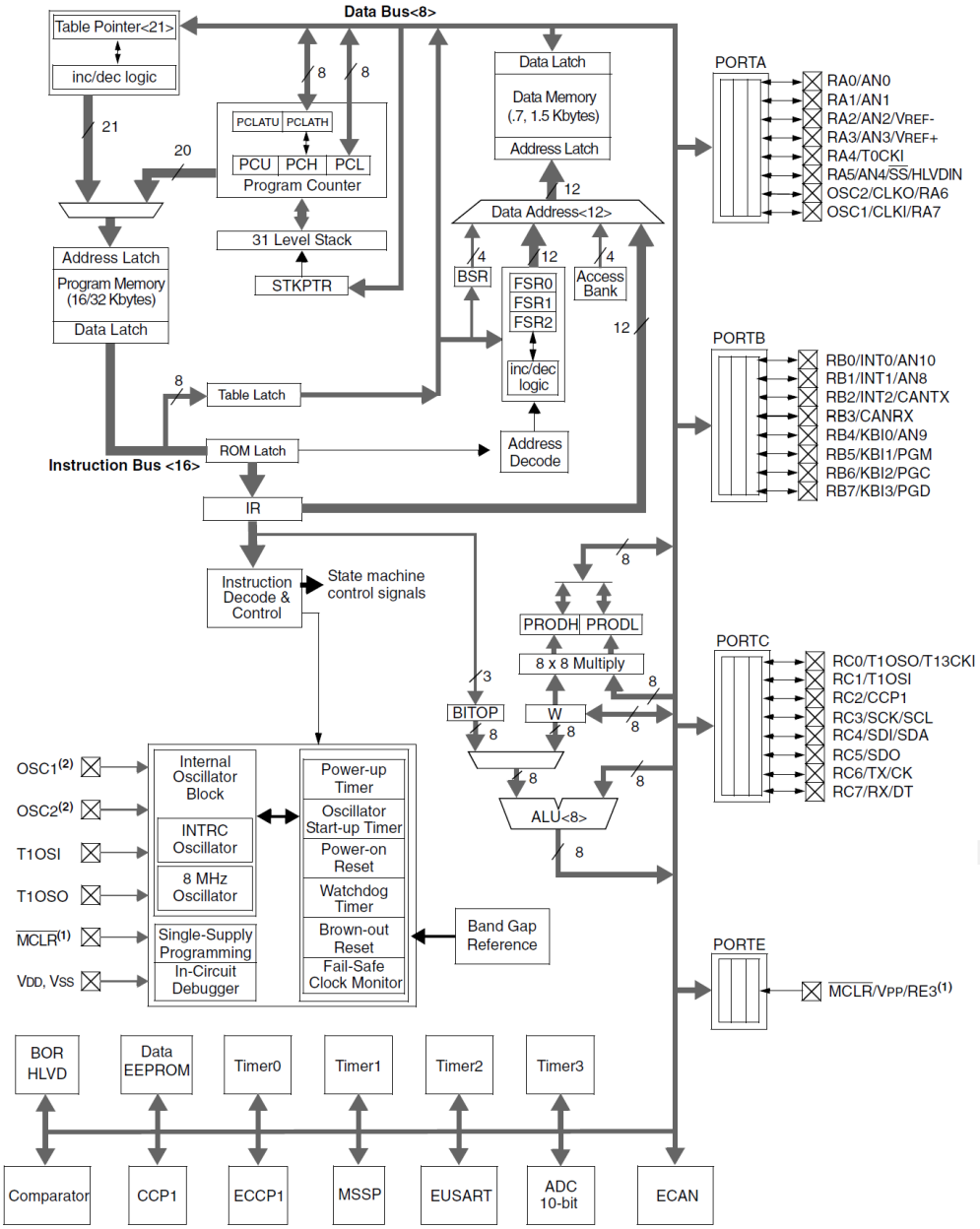
Il microcontrollore utilizzato è un PIC18F2580 prodotto dalla Microchip basato su architettura di tipo Harvard a 8bit. Le principali caratteristiche sono:

- Alimentazione da 2.0V a 5.5V
- Frequenza di lavoro massima 8 MHz (32MHz con PLL x4)
- 768 bytes di memoria RAM
- 3 port (A, B, C) per un totale di 24 I/O pin
- 4 timer
- 1 ADC, 8 linee disponibili
- 1 modulo CCP (Capture, Compare, PWM)
- 1x MSSP[8], 1x USART, 1x CAN

[7] Programmatore e debugger di microcontrollori PIC18
[8] Master Synchronous Serial Port (SPI, I2C)

Ho deciso di utilizzare il PIC a 32MHz attivando il PLL interno che moltiplica x4 la frequenza dell'oscillatore interno da 8MHz.

Le periferiche utilizzate sono l'interfaccia I2C dal sensore MPU-6050 (accelerometro e giroscopio), l'interfaccia USART per il modulo Bluetooth, il modulo PWM del CCP per l'azionamento dei motori e due pin di interrupt per l'acquisizione dei dati dagli encoder. L'architettura del microcontrollore è mostrata nella figura sottostante:



3.3 Sensori

A bordo del robot sono presenti sensori usati per stimare l'angolo rispetto alla verticale e sensori utilizzati per l'odometria.

3.3.1 Accelerometro e Giroscopio

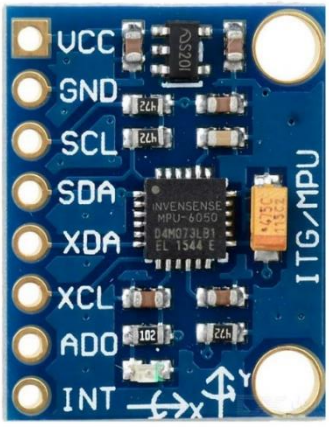
Per quanto riguarda la stima dell'angolo vengono utilizzati due sensori, in particolare un accelerometro e un giroscopio. L'accelerometro viene utilizzato per misurare l'accelerazione[9], mentre il giroscopio per il calcolo della velocità angolare.

Sul mercato sono presenti molte tipologie di accelerometri e giroscopi, che offrono diversi gradi di accuratezza e destinati a campi di applicazione diversi.

Ho optato per un sensore economico ma al tempo stesso preciso e affidabile, l'*MPU-6050*, che racchiude al suo interno un accelerometro e un giroscopio a 3 assi. Il collegamento con il microcontrollore avviene tramite bus di comunicazione I2C.

Data la sua dimensione assai ridotta, l'ho acquistato nella versione montata su breakout board[10], sulla quale sono già montati i componenti necessari al corretto funzionamento del sensore, tra cui un regolatore a 3.3V (il sensore funziona a 3.3V, ma grazie al regolatore è possibile alimentare la scheda anche a 5V) e le resistenze di pull-up necessarie per le linee SDA e SCL dell'I2C.

Il sensore presenta 8 pin, tra cui VCC e GND per l'alimentazione, SCL (Serial Clock) e SDA (Serial Data) per la comunicazione con il microcontrollore. I pin XDA e XCL permettono di collegare un sensore aggiuntivo utilizzando l'*MPU-6050* come master.

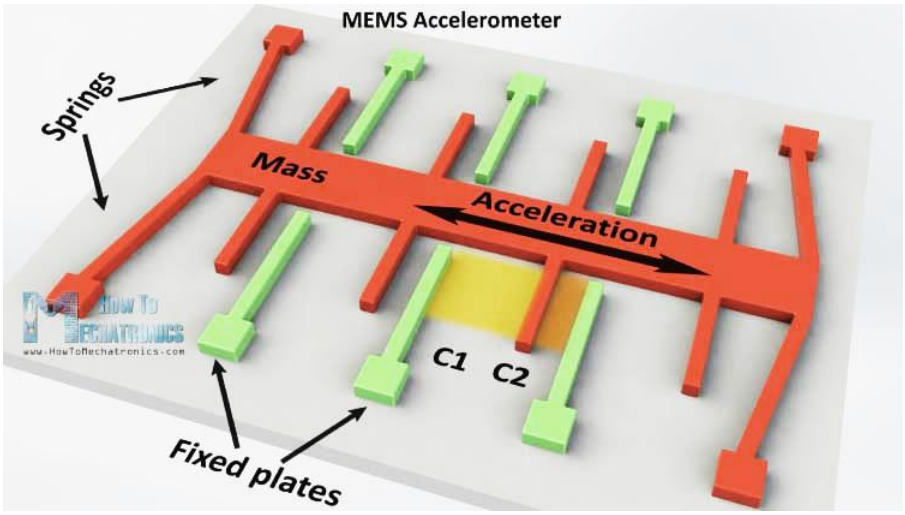


Infine il pin AD0 permette di cambiare l'indirizzo con il quale il sensore viene individuato sulla rete I2C, mentre il pin INT fornisce un segnale di interrupt in caso di eventi prestabiliti dall'utente.

[9] Variazione di velocità nel tempo (m/s^2)
[10] Schede elettroniche ad-hoc che rendono agevole l'utilizzo di componenti miniaturizzati

L'accelerometro e il giroscopio presenti all'interno del sensore sono MEMS[11], dispositivi elettro-meccanici integrati in forma altamente miniaturizzata, su un substrato di materiale semiconduttore.

Il funzionamento dell'accelerometro si basa sulla misura dello spostamento di una massa campione rispetto alla posizione di riposo, quando questa viene sottoposta ad un'accelerazione. La misura di tale spostamento avviene poiché è presente una struttura a pettini mobili e fissi, incrociati tra di loro; misurando la variazione di capacità tra i pettini in posizione di riposo e quando sottoposti ad una forza, è possibile ricostruire l'accelerazione applicata. Tale capacità viene poi convertita in una tensione proporzionale all'accelerazione.



Il funzionamento del giroscopio è simile all'accelerometro, infatti vengono utilizzate piccole masse che si muovono in funzione dei cambiamenti nella velocità angolare, secondo la forza apparente di Coriolis[12]. Queste minime variazioni vengono convertite in tensioni elettriche e amplificate come avviene per l'accelerometro.

I dati analogici dell'accelerometro e del giroscopio vengono successivamente elaborati e convertiti in digitale tramite un convertitore AD[13] a 16 bit dedicato per ogni asse. Il dato convertito viene poi immesso in un registro e reso disponibile per la lettura da parte del microcontrollore.

---

[11] Micro Electro-Mechanical Systems – Sistemi micro elettro-meccanici

[12] la forza di Coriolis è una forza apparente, a cui risulta soggetto un corpo quando si osserva il suo moto da un sistema di riferimento che sia in moto circolare rispetto a un sistema di riferimento inerziale

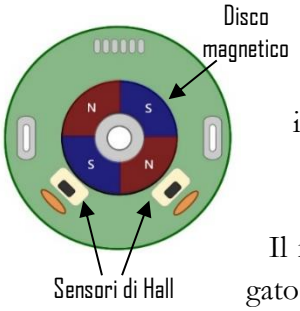
[13] Analogico - Digitale

3.3.2 Encoder ad effetto Hall

Molto importanti per il controllo di un robot sono i sensori odometrici che permettono di stimare la posizione, misurando lo spazio percorso dalle ruote. Questo permette di realizzare un sistema di controllo efficiente in grado di regolare la potenza da fornire ai motori in base alla loro velocità.

I sensori odometrici più diffusi nel mondo dell'elettronica sono gli encoder, cioè apparati elettromeccanici che convertono la posizione dell'asse rotante in un segnale elettrico digitale.

Esistono due tipi principali di encoder: gli *encoder assoluti* che forniscono l'esatta posizione dell'asse del motore e gli *encoder incrementali* che segnalano gli incrementi rispetto ad una posizione assunta come riferimento.



Non avendo la necessità di rilevare la posizione esatta del motore ma solo la velocità che esso assume, ho scelto di utilizzare degli encoder di tipo incrementale, in particolare encoder ad effetto Hall[14].

Il funzionamento di tale sensore è molto semplice; collegato all'asse del motore vi è un disco magnetico che ruota solidale con esso. Ogni qual volta la polarità nord (o sud) incontra il sensore di Hall (situato in prossimità del disco) viene generato un impulso.

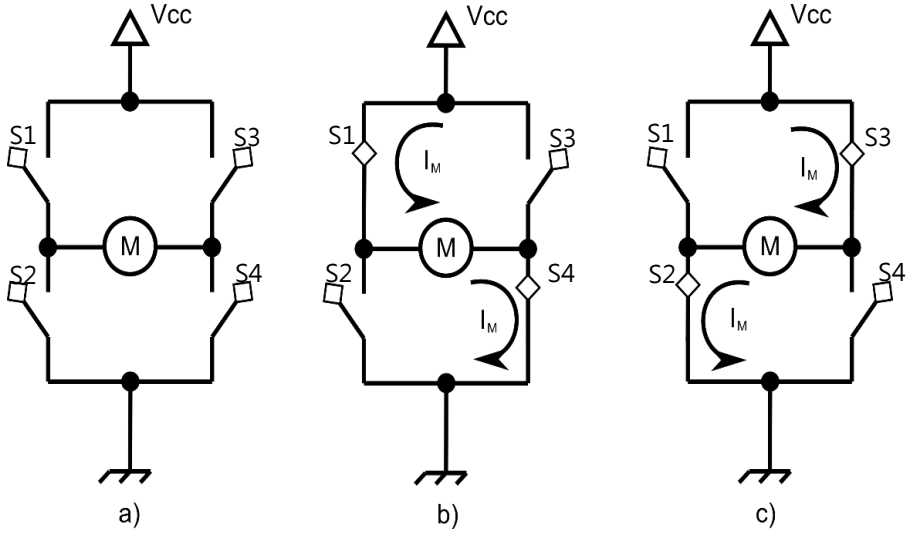
Il sensore che ho utilizzato ha 4 pin: due pin di alimentazione, VCC e GND e i due output del sensore di Hall, A e B, sfasati di 90° per rilevare il senso di rotazione del motore. Contando i fronti di salita e di discesa su entrambi gli output è possibile contare 48 impulsi per ogni giro del motore. Nel mio caso utilizzerò solo un canale rilevando solo i fronti di salita, perciò avrò solo 12 impulsi per ogni giro.

Un'altra considerazione da fare è il fatto che un giro del motore non corrisponde ad un giro dell'asse esterno, poiché vi è un riduttore 47:1, cioè ogni 47 giri del motore, corrispondono ad un giro dell'asse esterno. Avremo perciò 564 impulsi per ogni giro effettivo del motore.

[14] Sensore che rileva la presenza di campi magnetici aprendo o chiudendo un contatto elettrico

3.4 Elettronica di potenza

Per l'azionamento dei motori esistono diverse tecniche di pilotaggio, la più semplice delle quali è il pilotaggio ON-OFF (ON – motore alla massima velocità in un unico senso di rotazione, OFF – motore spento). Per permettere il cambio di rotazione, qualora necessario, è necessario invertire la polarità utilizzando un ponte H[15], costituito da 4 interruttori e altrettanti diodi di ricircolo[16].



Nella figura sovrastante si possono distinguere tre diverse situazioni in un ponte H:

a) Tutti e quattro gli interruttori sono aperti, perciò il motore è scollegato e non circola nessuna corrente. Il motore è fermo.
b) S1 e S4 sono chiusi perciò scorre una corrente Im nel motore. Il motore in questo caso gira in senso orario.
c) S3 e S2 sono chiusi perciò scorre una corrente Im nel motore opposta rispetto al caso b. Il motore gira in senso antiorario.

Al fine di non creare cortocircuiti, gli interruttori sullo stesso asse non devono mai essere chiusi insieme.

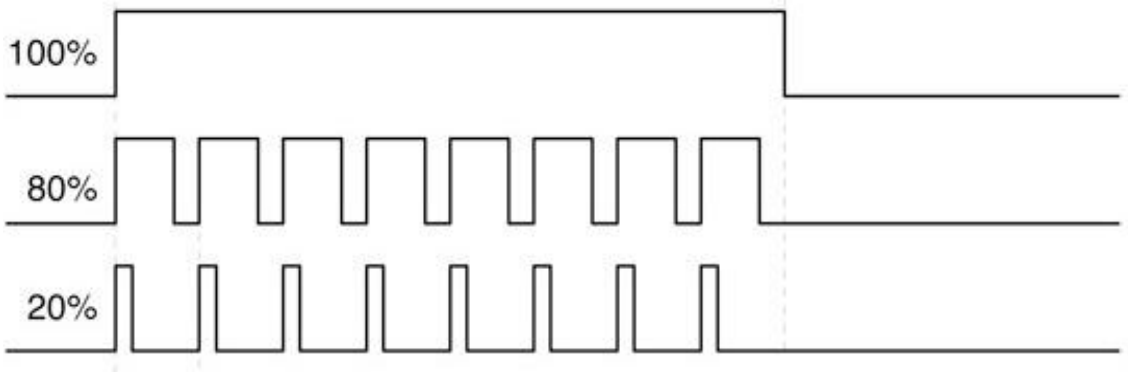
Per fermare il motore si può utilizzare la tecnica del freno elettrico, ponendo in cortocircuito i due terminali del motore, quindi attivando gli interruttori S1 e S3, oppure S2 e S4. In questo modo si scarica rapidamente la forza elettromotrice indotta dal motore.

---

[15] Circuito elettrico la cui forma ricorda la lettera 'H'

[16] Quando il motore viene fermato, si genera una tensione inversa ai suoi capi e per evitare di danneggiare i transistor di controllo vengono utilizzati dei diodi per scaricare le correnti residue

Per quanto riguarda la variazione di velocità al motore la tecnica più diffusa è la modulazione PWM[17]. Un segnale PWM è caratterizzato da una frequenza fissa e da un duty cycle variabile, che equivale al rapporto tra il tempo in cui l'onda assume un valore alto e il periodo T.



La modulazione PWM consente di controllare la potenza assorbita dal carico variando il duty cycle di un segnale ad onda quadra, infatti sul carico agisce una tensione proporzionale al duty cycle secondo la seguente formula:

$$V_{medio} = \frac{1}{T}\int_0^{t_{on}} V\,dt$$

Con T = periodo, t_{on} = tempo in cui il segnale alto, V = tensione del segnale.

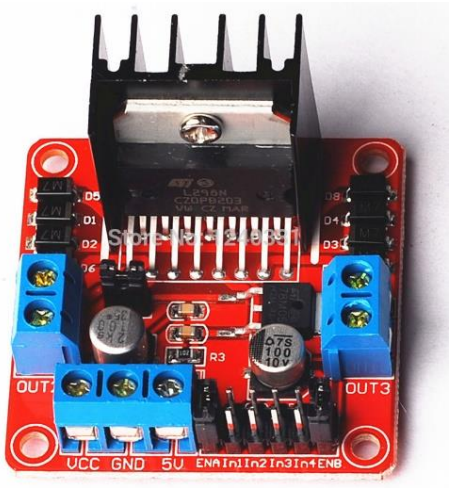
Questa tecnica è molto efficace poiché permette di modificare la velocità pur assicurando un rendimento energetico elevato, rispetto al pilotaggio dei transistor in zona attiva. Se la commutazione dell'onda quadra è piuttosto frequente (qualche KHz) a causa delle proprietà del motore la corrente media è costante e proporzionale al duty cycle del segnale.

Il modulo che ho scelto di utilizzare per il pilotaggio dei motori è l'*L298N,* un doppio ponte H, che permette di pilotare due motori in continua, con la possibilità di variare la velocità.

Il modulo richiede due alimentazioni, l'alimentazione logica a 5V e l'alimentazione di potenza che può arrivare fino a 50V. Il ponte H è in grado di fornire fino a 2 A di corrente per ogni motore. Le potenze in gioco sono molto elevate, infatti capita spesso che il driver si surriscaldi e perciò è necessario che l'L298N sia montato su un dissipatore.

[17] Pulse Width Modulation – Modulazione di larghezza d'impulso

Il driver è per questo disponibile già montato su una breakout board con un dissipatore e con tutta l'elettronica necessaria per il funzionamento; sono presenti i diodi necessari per il ricircolo della corrente su entrambi i canali ed è presente un regolatore a 5V per la generazione dell'alimentazione logica a partire da quello di potenza. Sono inoltre presenti due condensatori, necessari per il disaccoppiamento dell'alimentazione, in modo che i motori non disturbino la parte logica.



Sulla scheda sono disponibili 4 connettori, due dei quali sono i morsetti per il collegamento dei motori. Gli altri due connettori vanno alla scheda madre, infatti sono i comandi digitali per il controllo della velocità di rotazione e il verso di rotazione. I pin ENA ed ENB servono per abilitare o disabilitare i motori; nel nostro caso a questi due pin verranno collegati i pin del microcontrollore del PWM. I pin IN1 e IN2 controllano il senso di rotazione del motore A, così come i pin IN3 e IN4 controllano il senso di rotazione del motore B, secondo il seguente schema.

IN1 (o IN3)	IN2 (o IN4)	**MOTORE**
0	0	**Freno elettrico**
0	1	**Senso orario**
1	0	**Senso antiorario**
1	1	**Freno elettrico**

3.5 Comunicazioni

Per la comunicazione del robot con l'esterno ho predisposto l'utilizzo di un modulo Bluetooth.

Il Bluetooth è uno standard tecnico-industriale per la trasmissione di dati attraverso una rete a corto raggio (WPAN). Lo standard Bluetooth è stato progettato con l'obiettivo primario di ottenere bassi consumi, un corto raggio d'azione e un basso costo di produzione.

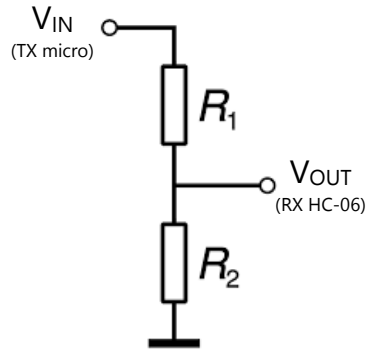
Il Bluetooth opera nel campo delle frequenze attorno ai 2.45 GHz, utilizzando la tecnica del frequency hopping[18].

Il modulo che ho scelto di utilizzare è *l'HC-06,* un modulo molto semplice da utilizzare che permette di trasformare una porta USART in una porta Bluetooth. Il modulo *HC-06* si presenta con 4 pin: due di alimentazione (Vcc e GND) e due pin per la comunicazione seriale (TX, RX).

Per collegarlo al PIC è sufficiente collegare la porta seriale del microcontrollore a quella del modulo, con l'accortezza di invertire i pin; TX andrà collegato a RX, e così RX a TX.

Il modulo ha una tensione di funzionamento di 3.3V; seppure sia possibile alimentarlo anche a 5V grazie alla presenza di un regolatore di tensione sulla breakout board, i pin di comunicazione non sono regolati, perciò con una tensione al di sopra di 3.3V si danneggerebbe il modulo.

Si ricorre dunque ad un partitore di tensione tra il pin TX del microcontrollore (che genera appunto un segnale a 5V) e il pin RX del modulo, affinché la tensione scenda a 3.3v.



$$V_{OUT} = V_{IN} * \frac{R2}{R1 + R2}$$

Con R1 = 8K2 e R2 = 15K

$$V_{OUT} = 5V * \frac{15K}{15K + 8K2} = 3{,}2327V$$

[18] Tecnica che consiste nel variare la frequenza di trasmissione a intervalli regolari in maniera pseudocasuale

3.6 Alimentazione

Per alimentare il robot ho deciso di utilizzare delle batterie agli ioni di litio poiché hanno una corrente di scarica elevata, non hanno effetto memoria e soprattutto hanno un ottimo rapporto densità/peso.

Ho utilizzato 4 batterie nel formato 18650 da 3.7V ciascuna, con capacità 2600mAh. Le ho collegate in serie ottenendo una tensione complessiva di 14.8V, che ho successivamente distribuito con tensioni differenziate per la corretta alimentazione della sezione logica (5V) e la sezione di potenza (12V).

Per ottenere la tensione necessaria alla sezione logica, ho utilizzato un regolatore lineare, il 7805, pur nella consapevolezza della sua inefficienza in termini energetici, per via della notevole dissipazione in calore dell'energia. Tuttavia valutando il consumo complessivo della sezione logica, che è pari a circa 20mA, ho ritenuto ragionevole l'utilizzo di tale regolatore poiché la potenza dissipata è circa 196mW. La potenza dissipata è proporzionale infatti alla tensione di alimentazione secondo la seguente formula:

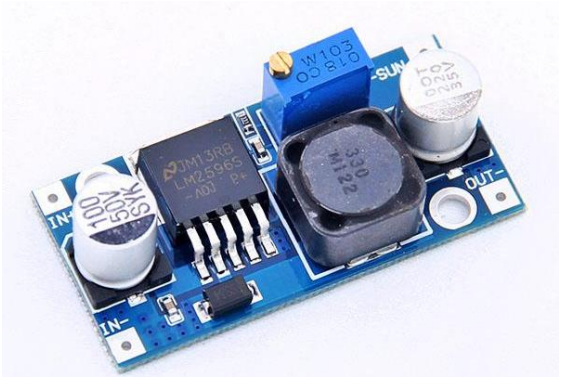
$$P_{dis} = (V_{cc} - 5v) * i$$

Vale a dire che la potenza dissipata inutilmente rispetto al totale è pari a:

$$P\%_{dis} = \frac{P_{dis}}{P_{tot}} \cdot 100 = \left(1 - \frac{5V}{V_{cc}}\right) \cdot 100$$

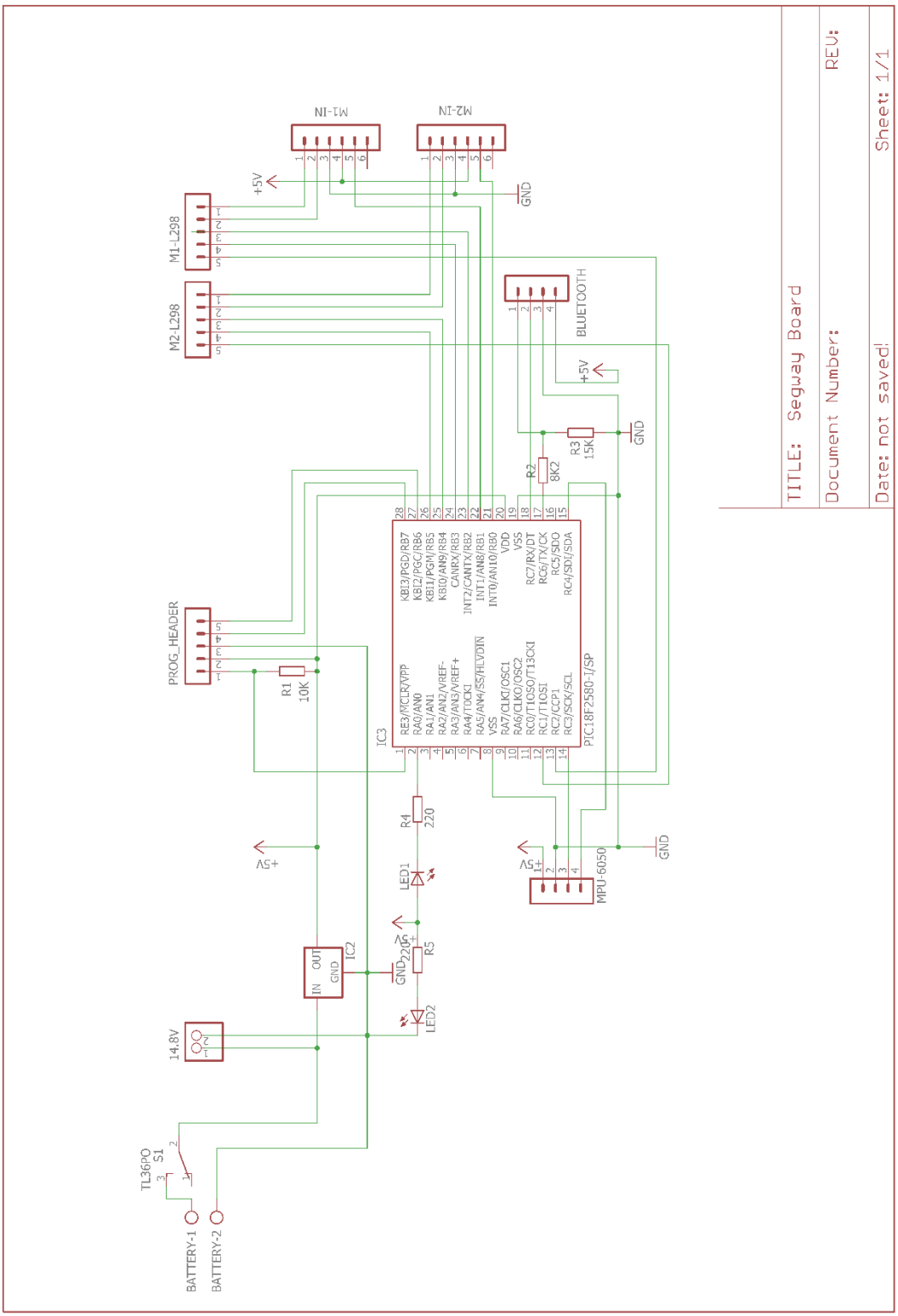
Per quanto riguarda l'alimentazione di potenza, l'utilizzo di un regolatore lineare è fuori discussione poiché le correnti in gioco sono molto elevate, perciò ho scelto di utilizzare un convertitore DC/DC switching[19] tipo buck che ha un'efficienza molto elevata, superiore all'80%.

Ho utilizzato quindi un modulo LM2596 montato su una breakout board, che permette di regolare la tensione di output tra 1.2v e 37v. La corrente massima di uscita è 5A.



[19] Alimentatore a commutazione elettronica

3.7 Schema elettrico



4. Progetto software

4.1 Logica di controllo

Il software a cui è affidato il controllo del robot è scritto in linguaggio ANSI C. Il compilatore utilizzato per la programmazione del PIC è l'xc8 nella versione 1.38.

Il software si occupa della lettura dei dati sensoriali, dell'esecuzione dell'algoritmo di controllo e della comunicazione con l'esterno. Un aspetto molto importante è la necessità di svolgere le varie funzioni a intervalli ben scanditi e precisi.

L'esecuzione dei task periodici è gestita tramite un timer (timer0), mantenuto costantemente in funzione, che genera interrupt ogni millisecondo, incrementando una variabile. Il programma principale, interrogando detta varabile verifica il momento di esecuzione dell'opportuna routine tra quelle impostate.

Per impedire la caduta del robot bisogna infatti agire abbastanza velocemente da impedire la sua caduta. Sapendo che il robot è alto 30 cm possiamo ricavare il suo tempo di oscillazione con la seguente formula:

$$T = 2\pi * \sqrt{\frac{l}{g}}$$

Con g = 9,81m/s^2 e l = 0,3 m

$$T = 1.0987\ s$$

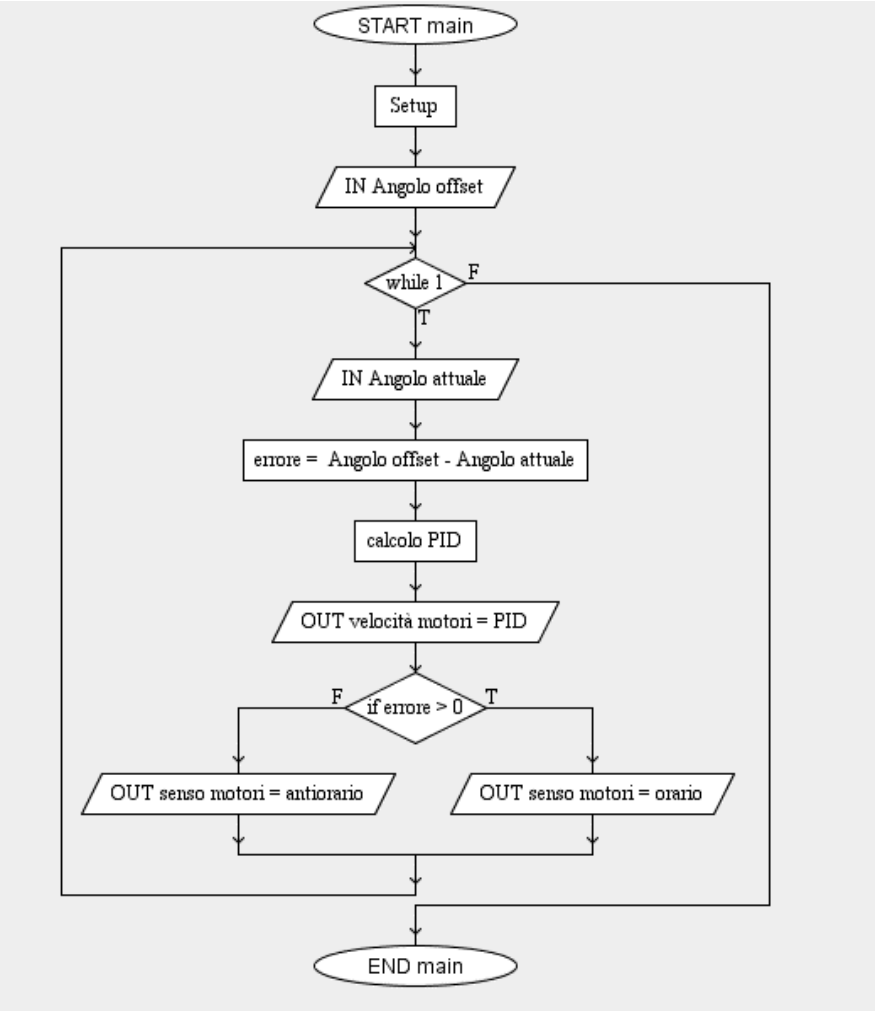
Il tempo che impiega il robot a cadere perciò è 1,0987 secondi; conoscendo tale dato, per intervenire e correggere in tempo la tendenza a cadere bisognerà essere almeno venti volte più veloci. Tendenzialmente può andare bene anche una velocità di controllo fino a 1/50 della velocità di oscillazione, oltre la quale il sistema va in overshoot.

Il software è stato sviluppato in diverse parti, in modo da rendere più semplici le operazioni di debug e ricerca degli errori. La divisone del programma in funzioni rende inoltre facilmente leggibile e comprensibile il programma.

I blocchi di programma principali sono:

main	Programma principale
MPU-6050	Programma che contiene tutte le funzioni per l'acquisizione dei dati dal sensore e per il calcolo dell'angolo
I2C	Programma per l'inizializzazione e l'utilizzo del bus I2C
Bluetooth	Programma per l'inizializzazione del modulo HC-06 e per la comunicazione Bluetooth
PID&Motors	Programma nel quale sono contenute le funzioni necessarie per l'algoritmo di controllo (PID) e per l'azionamento dei motori

Nella figura sottostante è mostrato l'algoritmo essenziale di funzionamento del robot:



4.2 Acquisizione dati sensoriali

L'acquisizione dei dati sensoriali di accelerometro e giroscopio avviene tramite interfaccia I2C. Dopodiché vengono elaborati dal microcontrollore per ottenere un angolo. Gli encoder montati sui motori vengono invece collegati agli ingressi di interrupt, per il calcolo della velocità.

4.2.1 Protocollo I2C

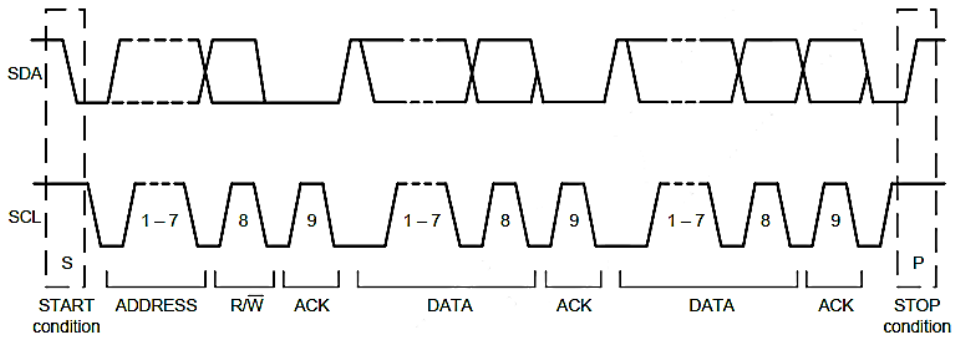
I2C (i-quadro-ci) è un'interfaccia composta da due fili, che trasmettono in seriale, uno il clock (SCL – Serial Clock) e l'altro i dati (SDA – Serial Data). Le due linee sono bi-direzionali e open-drain, e necessitano un collegamento pull-up all'alimentazione.

In un'implementazione generica I2C è presente un master e uno (o più) slave. Quando il master emette sul bus l'indirizzo relativo ad uno slave, esso risponde e attende un comando. Il sensore MPU-6050 opera sempre come slave (codice 0xD0) quando comunica con il microcontrollore. La velocità massima di trasferimento dati su una linea I2C è 400KHz.

La comunicazione comincia quando il master impone la condizione di start sul bus, definita da una transizione da livello logico alto a basso della linea SDA.

Il bus è da considerarsi occupato fino a che il master non impone la condizione di stop, definita da una transizione da livello logico basso ad alto della linea SDA mentre la linea SCL è a livello alto. Dopo lo start, il master invia un indirizzo di 7 bit, più un ottavo, che dice se l'istruzione è di lettura o scrittura.

I dati trasmessi con protocollo I2C sono definiti da 8 bit (1 byte). Ogni byte trasmesso deve essere seguito da un segnale di accettazione (acknowledge - ACK). Il clock per il segnale di ACK è generato dal microcontrollore, mentre il dispositivo ricevente (in questo caso il sensore MPU-6050) genera il segnale di ACK portando il segnale su SDA basso per il periodo di clock.



Il microcontrollore implementa un modulo MSSP, che può operare nelle modalità SPI e I2C. Per avviare una comunicazione con il sensore, bisogna inizializzare i registri relativi a tale modulo in modo da selezionare la modalità I2C a 400KHz (scelgo la velocità massima per rendere le operazioni di acquisizione dati più rapide).

Per inizializzare il modulo, bisogna scrivere che si intende utilizzare il modulo MSSP come master della comunicazione I2C sul registro SSPCON1. Fatto ciò bisogna decidere la frequenza di lavoro, che viene impostata scrivendo sul registro SSPADD un valore numerico in modo che:

$$FreqI2C = \frac{Fosc}{4 * (SSPADD + 1)}$$

Con FreqI2C = 400 KHz e Fosc = 32 MHz

$$SSPADD = \frac{8MHz}{400KHz} - 1 = 19$$

Di seguito sono elencate le routine software per l'inizializzazione, la scrittura e la lettura dei dati via I2C:

```
void OpenI2CStSpeed() {
        SSPCON1 = 0b00101000;       //attivo la porta seriale, freq = FOSC/(4* (SSPADD + 1))
        SSPADD = 19;                //dato ottenuto precedentemente dal calcolo
}
```

```
void WriteData (unsigned char devAdress, unsigned char regAdress, unsigned char data){
        IdleI2C();                  //Attendo che la linea sia libera
        StartI2C();                 //Attivo la sequenza di start
        WriteI2C(devAdress);        //Invio l'indirizzo del dispositivo
        WriteI2C(regAdress);        //Invio l'indirizzo del registro sul quale voglio scrivere
        WriteI2C(data);             //Invio il dato
        StopI2C();                  //Chiudo la linea
}
```

```
unsigned char ReadData (unsigned char devAdress, unsigned char regAdress){
        unsigned char data;
        IdleI2C();              //Attendo che la linea sia libera
        StartI2C();             //Attivo la sequenza di start
        WriteI2C(devAdress);    //Invio l'indirizzo del dispositivo
        WriteI2C(regAdress);    //Invio l'indirizzo del registro dal quale voglio leggere
        RestartI2C();           //Riattivo la sequenza di start
        WriteI2C(devAdress+1);  //Invio l'indirizzo del dispositivo (1 bit di lettura)
        data=ReadI2C();         //Leggo il dato
        NotAckI2C();            //Confermo la lettura avvenuta al sensore
        StopI2C();              //Chiudo la linea
        return data;
}
```

4.2.2 Calcolo angolo

I dati acquisiti dal sensore, sono relativi ad accelerazione e velocità angolare sui 3 assi; tuttavia questi dati non ci permettono di calcolare l'angolo del robot rispetto alla verticale ed è necessario ricorrere a formule che permettano il calcolo dell'angolo sia a partire dai dati dell'accelerometro, che dal giroscopio.

Per quanto riguarda l'accelerometro, immaginando di rappresentare i vettori di accelerazione in un piano tridimensionale, calcolando la fase della risultante dei tre vettori (uno per asse) rispetto al piano di nostro interesse (quello orizzontale) possiamo ricostruire l'angolo del robot rispetto al piano.

Immaginando che il sensore sia fermo e appoggiato su un piano, questo rileverebbe solo un'accelerazione, quella di gravità (pari a 9,81m/s^2). In questo caso, rispetto al piano risulterebbe un angolo di 90°.

La formula per calcolare l'angolo rispetto al piano di nostro interesse (asse x) è:

$$\theta_{accel} = arctg\left(\frac{acc_x}{\sqrt{{acc_y}^2 + {acc_z}^2}}\right)$$

Per quanto riguarda il giroscopio invece la misura dell'angolo è più semplice, poiché sapendo che la velocità angolare è la derivata della posizione angolare sul tempo, per ottenere la posizione angolare bisogna effettuare l'integrazione:

$$\int \dot{\theta_{gyro}}(t)\, dt = \theta_{gyro}(t)$$

Tuttavia nei sistemi digitali, effettuare un'integrazione continua è impossibile pertanto va effettuata una sommatoria delle velocità angolari calcolate entro un intervallo di tempo costante t_{sample}:

$$\theta_{gyro}(t) = \int \dot{\theta_{gyro}}(t)\, dt \cong \sum_{0}^{t} \dot{\theta_{gyro}}(t) \cdot t_{sample}$$

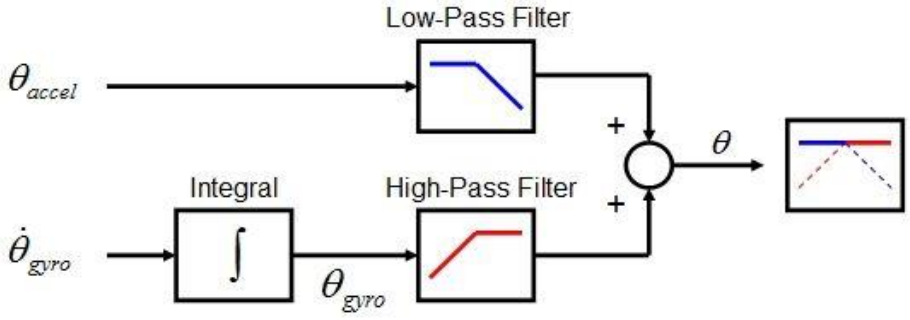
L'approssimazione ovviamente introduce errori, per il semplice fatto che se i valori letti dal giroscopio cambiassero più velocemente dell'intervallo temporale costante, non potremmo individuarli e quindi si accumulerebbe in errore nella rilevazione finale dell'angolo (drift).

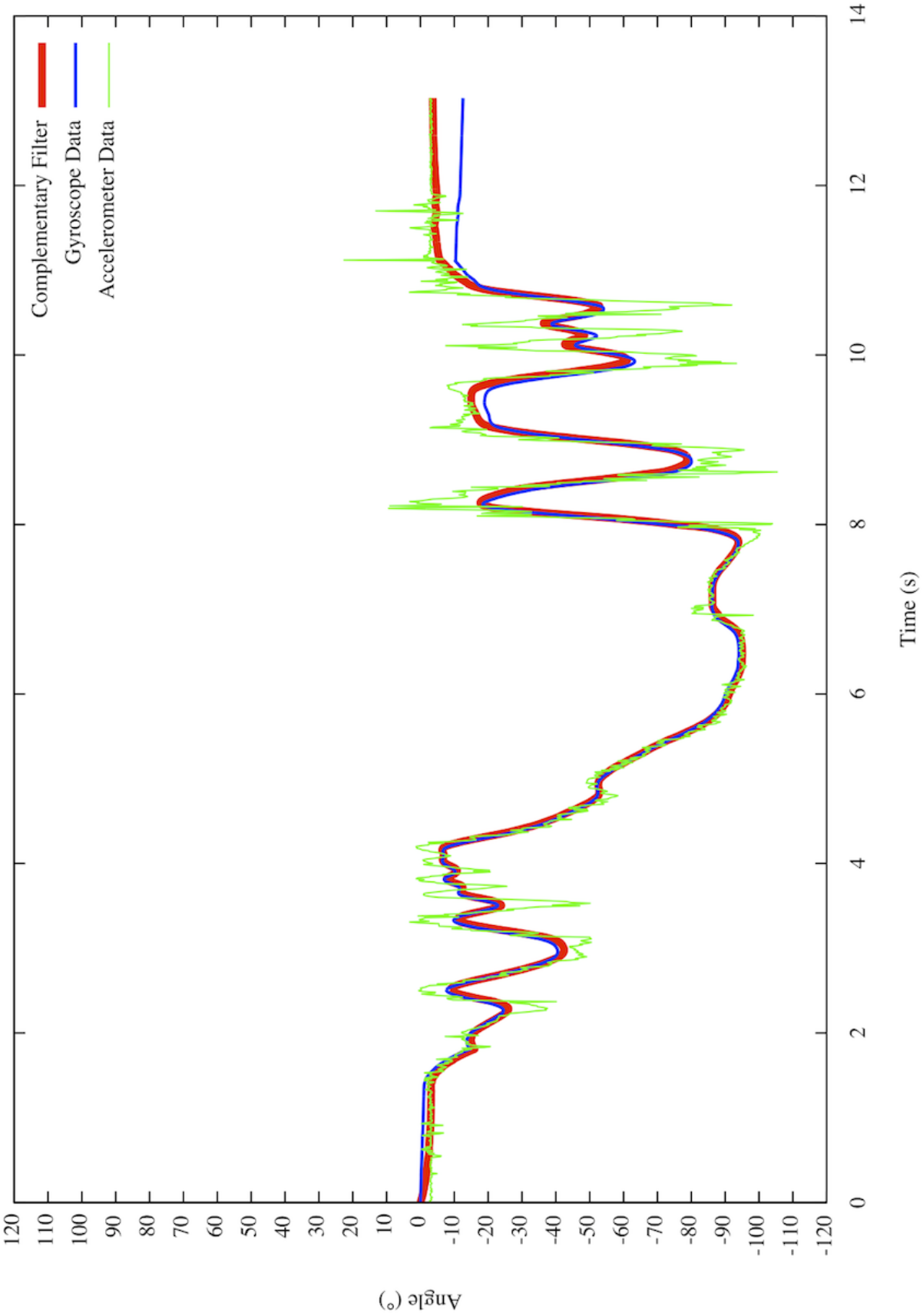
La scelta di utilizzare sia accelerometro che giroscopio nel calcolo dell'angolo è dato dal fatto che gli accelerometri sono affidabili sul lungo termine e i giroscopi affidabili nel breve termine.

Gli accelerometri infatti sono molto rumorosi poiché sensibili alle vibrazioni e affidabili solo per la rilevazione a lungo termine, mentre i dati angolari forniti dal giroscopio sono maggiormente precisi e immuni da rumore. Questi ultimi, al contrario degli accelerometri, per via del fenomeno di drift, risultano imprecisi nel lungo periodo.

La soluzione dunque è quella di combinare i dati dei due sensori utilizzando un filtro complementare (con $0.88 < \alpha < 0.98$):

$$\theta_{filter} = \alpha \cdot \left(\theta_{filter} + \left(\dot{\theta_{gyro}}(t) \cdot t_{sample}\right)\right) + (1 - \alpha) \cdot \theta_{accel}$$




Complementary Filter
Gyroscope Data
Accelerometer Data
Angle (°)
Time (s)
120
110
100
90
80
70
60
50
40
30
20
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
-120
0
2
4
6
8
10
12
14

4.2.3 Calcolo velocità

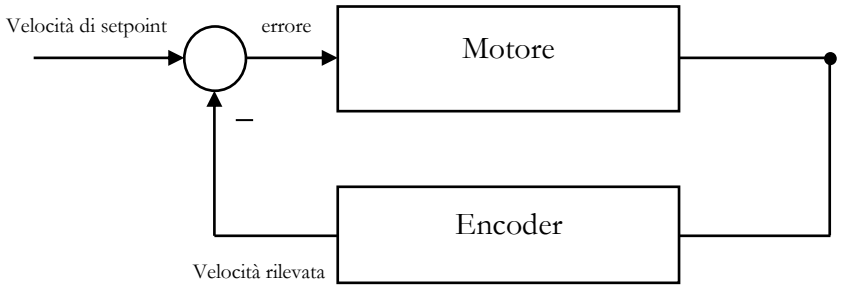
Gli encoder sono collegati agli ingressi di interrupt del microcontrollore in modo che ad ogni impulso venga incrementata una variabile. Il tempo che trascorre tra un impulso è l'altro permette di calcolare la velocità del motore.

Ricordando infatti che rilevando solo i fronti di salita su uno dei due pin dell'encoder (A o B), vengono rilevati 12 impulsi per ogni giro del motore. Dato che il motore è montato su un riduttore, per avere un giro completo di questo sono necessari 47 giri del motore, cioè 564 impulsi.

Ad ogni impulso dunque il motore ha compiuto 1/564 del giro, quindi la velocità del motore sarà (t = tempo trascorso tra un impulso e l'altro):

$$v[RPS] = \frac{1/564}{t[s]}$$

L'implementazione degli encoder per il controllo della velocità è necessario perché una retroazione permette un controllo più accurato della potenza fornita ai motori. Infatti i motori non forniscono una risposta lineare e inoltre due motori comandati con la stessa potenza non si muoveranno mai in maniera identica.



4.3 Algoritmo di controllo PID

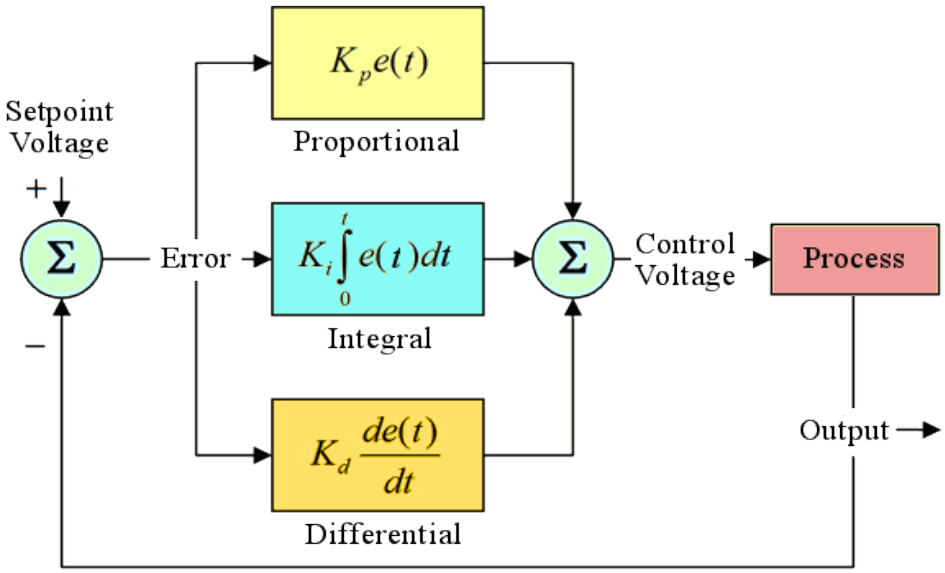
Il sistema di controllo che ho deciso di utilizzare per il robot è il controllore proporzionale-integrativo-derivativo (PID).

Il PID è un sistema di retroazione negativa, che grazie ad un input che determina il valore attuale, è in grado di reagire a un eventuale errore positivo o negativo tendendo verso 0. La reazione all'errore può essere regolata mediante i coefficienti delle singole parti del controllore (kp, ki, kd).

Il processo di funzionamento di tale controllore è molto semplice: acquisisce in ingresso un valore da un processo (nel caso del robot, l'angolo del robot) e lo confronta con un valore di riferimento (l'angolo di partenza del robot).

La differenza, viene quindi utilizzata per determinare il valore della variabile d'uscita del controllore, che è la variabile manipolabile del processo (cioè la velocità PWM per i motori). Il PID regola l'uscita in base a:

- Il valore del segnale di errore (*azione proporzionale*)
- I valori passati del segnale di errore (*azione integrativa*)
- Quando velocemente il segnale di errore varia (*azione derivativa*)



Come mostrato nello schema a blocchi nella pagina precedente, le tre azioni di un PID vengono calcolate separatamente e sommate algebricamente:

$$PID = P + I + D$$

L'azione proporzionale è ottenuta semplicemente moltiplicando il segnale di errore per un'opportuna costante:

$$P = K_P \cdot e$$

L'azione integrale è proporzionale all'integrale del segnale di errore, moltiplicato per la costante integrale:

$$I = K_I \int e\, dt$$

L'azione derivativa invece è pari alla derivata del segnale di errore sul tempo, moltiplicata per la costante derivativa:

$$D = K_D \cdot \frac{de}{dt}$$

Per regolare i parametri K_P, K_I e K_D, si può utilizzare il metodo di Ziegler Nichols che permette di trovare il "guadagno critico" dal quale derivano tutti gli altri parametri PID.

L'algoritmo per trovare i parametri del PID è il seguente:

1. Il controllore deve essere solo proporzionale, quindi azzeriamo l'integrale e il derivativo;
2. Si aumenta il guadagno K_P gradualmente;
3. Il guadagno critico (K_U) è il valore del guadagno per il quale la variabile controllata (il robot) presenta oscillazioni sostenute;
4. Una volta trovato il guadagno critico, registrare il periodo critico (P_U) delle oscillazioni sostenute.

Secondo la sottostante tabella, si determinano quindi le costanti per il controllore:

Tipo	**K_P**	**K_I**	**K_D**
P	0.50 K_U		
PI	0.45 K_U	1.2 K_P / K_U	
PID	0.60 K_U	2 K_P / K_U	K_P * K_U / 8

4.3 Interfaccia Bluetooth

Per quanto riguarda la comunicazione via Bluetooth con l'esterno, ho deciso di utilizzare il modulo HC-06, di cui ho parlato nel capitolo precedente.

Il modulo HC-06 permette di essere configurato nei suoi parametri, quali il nome, la password di associazione e la velocità di trasmissione dati con il microcontrollore. Per fare ciò è necessario utilizzare i comandi AT[20].

Ad esempio inviando via seriale a 9600 baud (la velocità di default del modulo che può essere cambiata) la stringa 'AT', il modulo dovrebbe rispondere 'OK'. Per cambiare il nome è necessario inviare la stringa 'AT+NAMExxx' sostituendo alle 'xxx' il nome che si desidera inserire. Per quanto riguarda la password il comando è 'AT+PINxxx' dove al posto di 'xxx' inseriremo il pin che desideriamo impostare.

Infine per cambiare la velocità di trasmissione dati bisogna inviare la stringa 'AT+BAUDx' con al posto della 'x' il carattere relativo alla velocità desiderata. Scelgo una velocità di trasferimento abbastanza rapida, cioè a 115200 baud, in modo da inviare e ricevere i dati velocemente per non disturbare l'esecuzione del software di acquisizione dati e di controllo.

A questo punto, messo a punto il modulo Bluetooth, bisogna inizializzare il modulo USART del microcontrollore nelle operazioni di lettura e scrittura alla velocità di 115200 baud.

Il software per l'inizializzazione è il seguente:

```
void bluetoothSetup(){
        BRG16=1;                    // Valore baud rate a 16bit
        BRGH=1;                     // High speed ON
        SPBRG=68;                   // baud rate => 115200 = fosc/(4*(SPBRG+1))
        TX9 = 0;                    // Invio dei dati a 8 bit
        TXEN=1;                     //Attivo il modulo trasmissione dati
        CREN=1;                     //Attivo il modulo ricezione dati
        SPEN = 1;                   //Rendo operativa la porta seriale
}
```

[20] Comandi AT, cioè d'attenzione (ATtention)

Per quanto riguarda la ricezione dei dati via Bluetooth è necessario verificare lo stato del flag RCIF; in caso sia '1' significa che nel registro RXREG è presente un dato appena ricevuto. Nel caso del software del robot ho attivato l'interrupt sulla ricezione (RCIE = 1), perciò ogni qualvolta viene ricevuto un dato, il programma si interrompe e viene acquisita la stringa ricevuta.

L'invio dei dati può essere effettuato mediante la scrittura di un carattere nel registro TXREG, attendendone l'invio controllando il flag TXIF; quando questa sarà a '1' l'operazione di invio sarà completata.

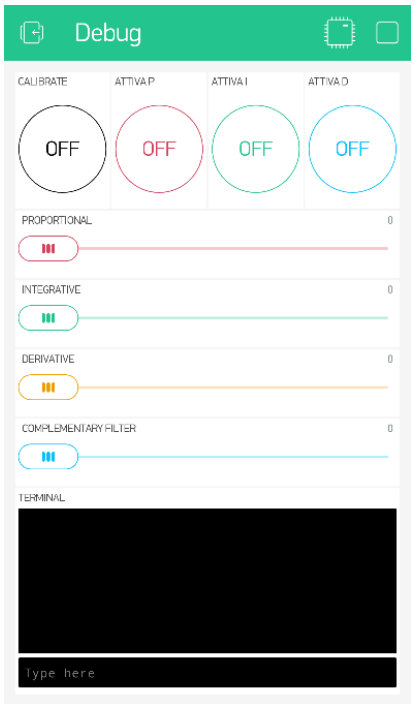
Per il controllo del robot e la regolazione dei parametri del PID da remoto, ho deciso di creare un'applicazione per smartphone Android nell'ambiente Blynk®.

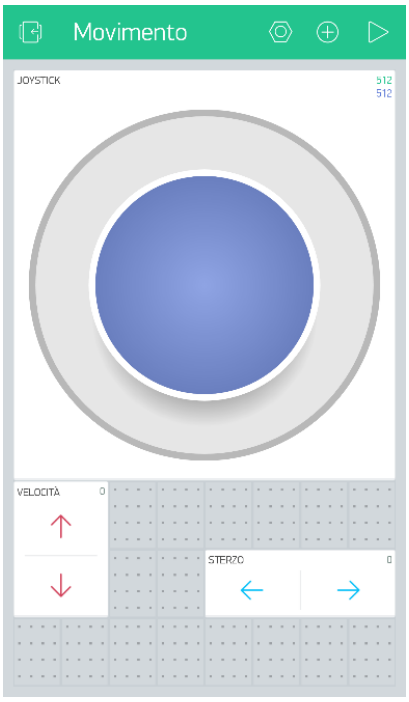
L'ambiente Blynk® permette di creare un'applicazione, facilmente interfacciabile con il modulo Bluetooth, nella quale è possibile inserire diversi oggetti, quali potenziometri, joystick, led, pulsanti e quant'altro.

Ho creato due maschere dell'applicazione:

- una per il *debug*, nella quale è possibile regolare i parametri PID e del filtro complementare, attivare o disattivare funzionalità e visualizzare tramite un monitor seriale i dati inviati e ricevuti dal microcontrollore;
- una per la *navigazione del robot*, nella quale ho inserito un joystick e dei pulsanti per "guidare" il droide.

Le immagini sottostanti mostrano le schermate dell'applicazione realizzata:





5. Conclusioni

Il lavoro presentato è la risultante della creazione di un prototipo di robot auto bilanciante, studiato e realizzato in ogni sua parte hardware e software.

La progettazione e realizzazione del robot, sono stati per me una vera e propria sfida poiché ad ogni problema risolto, se ne ripresentava uno nuovo che, tuttavia, mi ha indotto a rinforzare le mie basi di elettronica e di programmazione ampliando le conoscenze riguardanti i sistemi di controllo.

I maggiori problemi li ho avuti nella regolazione del controllore PID nei suoi parametri e in relazione all'acquisizione dei dati del giroscopio. Infatti, anche solo piccole imprecisioni temporali nel calcolo dell'angolo, danno origine a errori che compromettevano la stabilità del robot.

Mi ritengo nell'insieme soddisfatto della realizzazione del presente progetto, che come detto sopra, mi ha permesso di migliorare le necessarie competenze tecniche e organizzative per ogni sua fase, dalla progettazione iniziale, alla scelta dei materiali e componenti necessari, all'assemblaggio e cablaggio del sistema fino allo studio e successivo sviluppo di un software efficiente.