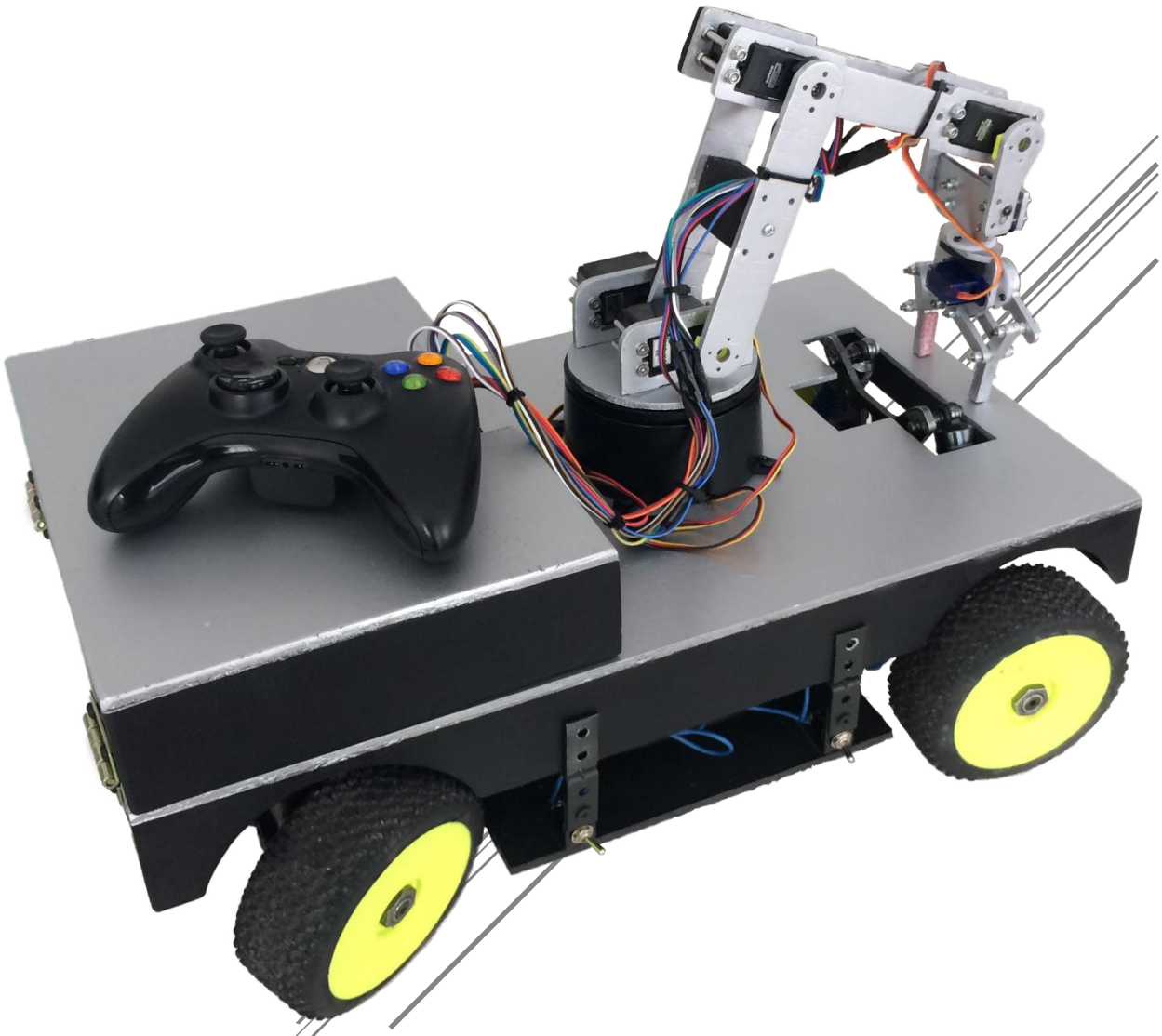
MACCHINA CON BRACCIO RADIOCOMANDATO

Progetto per esame di stato



I.I.S Castelli Anno 2014/15

Indirizzo Elettronica, Elettrotecnica e Automazione

Ramo elettronico

Daniele Pavanelli – Dario Ferrari – Cannone Roberto

SOMMARIO

ABSTRACT

The project we built is basically a car with a robotic arm mounted on the back. To make this device, we had to use a microcontroller called Arduino, which is programmable with a computer to solve specific tasks. Using this kind of "user-friendly" microcontroller, we started building up all the parts and connecting the together. The main core is Arduino, which is connected to two main circuits:

- The **car** one, which is basically composed by a motor, that changes direction with specific commands, as well as a servo used to make the car turn right and left while moving;
- The **robotic arm**, composed by 6 servos, which can be moved when the car is stopped and is provided with a claw which can lift up things.

Everything is powered up by two different kind of batteries. The pack of batteries of 9V are used to turn on the motor and make it run faster to pull the car itself while the other battery of 7.4V is the one which is regulated and powers on the various circuits. We created three circuits which are necessary for the whole thing to work:

- The **step-down** voltage regulator, which is used not only to regulate the voltage to 5V fixed to power on the circuits, but it's also capable of an output of maximum 3A. This means that we can also connect the entire arm because the current requests are not higher than 2.3A.
- The **H-bridge**, which is needed to control efficiently the motor, where only three cables from Arduino are needed to change direction and velocity.
- The Arduino's **USB Shield** and its **adapter**, which are needed to control remotely everything. In fact, we used an Xbox wireless controller and we had to connect it remotely with Arduino. The only way was to use a shield that supports the USB hosting to connect a receiver. The adapter is a simple board with pre-configured connections for the arm's servos and for the power.

Talking about the control part, we had to import some libraries in Arduino to make it communicate with the controller. Then we had to make it read the values assigned to the buttons on the controller and we choose what movement was connected to that action.

The arm is made of pieces of wood, like the car box, and there are 7 servos which make it rotate, fold and close a claw. Every servo is assigned with a button on the controller and it is provided with movement limitation for security reasons.

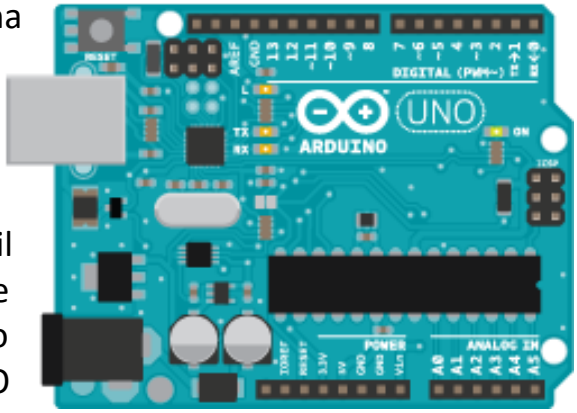
The car is basically a four-wheel drive buggy and we added a wooden box to attach the arm and the circuits. The lower part grants the space necessary to place the motor and the two batteries. All the cables coming from the lower part are brought in the upper part using a particular hole and they are connected with the rest of the circuits.
These cables come from the battery so they power up everything. But of course we don’t need the circuits to power on unless we need it. We added a switch for every battery so we could turn it on or off whenever we need it.

1 INTRODUZIONE

1.1 ARDUINO

Per la realizzazione del progetto è stata utilizzata la board Arduino, una base di programmazione open-source. Essa programmabile attraverso un software dedicato in Windows, e utilizza un linguaggio derivato dal C/C++. È facilmente interfacciabile a un gran numero di sensori e con vari dispositivi, il che lo rende utilizzabile in un gran numero di applicazioni. Integra un microcontrollore con pin connessi a porte che possono essere impostate input o output (I/O), un regolatore di tensione e un'interfaccia USB che permette la comunicazione con il computer. Tra le varie versioni disponibili abbiamo utilizzato Arduino UNO, una board dalle dimensioni ridotte e dal prezzo contenuto, che risulta essere particolarmente ottimizzata per la conversione USB-seriale.

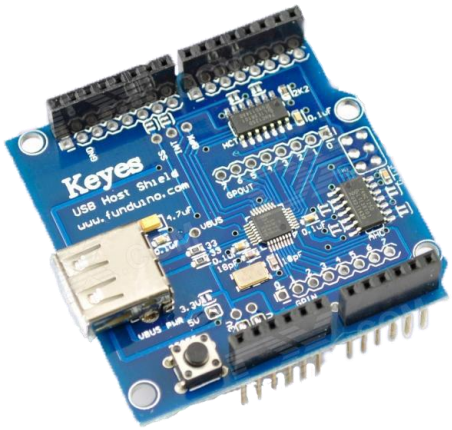
1.2 PINS



La scheda Arduino fornisce il controllo su una serie di pin con funzionamento sia come input (quindi con la possibilità di ricevere dati) e come output (con la possibilità di inviare dati). Questa caratteristica viene definita dall'utente, il quale attraverso il comando *pinMode()* è in grado di scegliere per ogni singolo pin se è necessario usarlo come INPUT o OUTPUT. Tutti i pin di I/O sono collocati sulla parte superiore della scheda mediante connettori e sono numerati da 0 a 13. I primi due (il pin 0 e 1) sono definiti TX e RX, e hanno la possibilità rispettivamente solo di inviare o ricevere dati. I restanti sono definiti "*Digital*" cioè digitali, comandabili attraverso i comandi *digitalRead()* e *digitalWrite()* (rispettivamente leggere il dato ricevuto e scrivere un dato fornito). Questi pin possono avere solo dei valori digitali, quindi 0 o 1, definiti come stato alto (*HIGH*) o basso (*LOW*). Alcuni di questi pin hanno un segno accanto che sta a significare il loro possibile utilizzo come PWM: hanno infatti la possibilità, attraverso la modifica del valore medio della tensione fornita, di ottenere un valore

analogico partendo da uno digitale. Si crea quindi un'onda quadra di cui viene variato il duty-cycle, ovvero il tempo in cui sta alta rispetto a quello in cui sta bassa. Utilizzando questi valori è possibile ottenere una vasta gamma di dati partendo da quelli digitali.
I pin presenti nella parte inferiore destra della scheda sono input analogici, e da essi è possibile leggere valori di tensione da un dispositivo esterno. Questi pin hanno una risoluzione di 10bit, il che significa che restituiscono il valore letto attraverso un numero intero compreso tra 0 e 1023.

1.3 LIBRERIE



Per poter utilizzare Arduino, e quindi stabilire un collegamento con i dispositivi che utilizzeremo nel progetto, è necessario inserire delle librerie. Le librerie non sono altro che dei file scritti in linguaggio C/C++ dove viene indicato una serie di comandi. In questo modo, dopo avere impostato la lettura di queste librerie attraverso il comando *#include,* è possibile usare dei comandi preimpostati che permettono un interfacciamento. Grazie alle librerie siamo in grado di comunicare facilmente con lo shield USB collegato ad Arduino, potendo gestire i dati ricevuti dalla porta USB e gli eventuali collegamenti aggiuntivi. Una delle librerie utilizzate è quella per il controllo della porta USB, *"Xbox_Controller_Library"*, la quale garantisce la corretta lettura dei valori quando viene collegato un controller. Nel nostro caso tra le varie configurazioni è stata scelta quella che riguarda il controller wireless, il quale viene collegato attraverso un adattatore apposito. Un'ulteriore libreria per lo shield è quella per il controllo dei piedini esclusivi della scheda, i GPIO. Infatti questa board impiega attivamente i pin dal 9 al 13, rendendoli inutilizzabili e quindi non a disposizione dell'utente. Fortunatamente vengono forniti dei pin digitali che nel nostro caso si sono rivelati fondamentali per il funzionamento del ponte ad H, necessario alla gestione del motore. Infine è stata utilizzata anche l'essenziale libreria per i servomotori, grazie alla quale è possibile controllare i movimenti dei servomotori direttamente con dei comandi *"servo.write()"* e specificando l'angolo della rotazione.

1.4 SKETCH (PROGRAMMA)

La programmazione di Arduino avviene attraverso un IDE (Integrated Development Environment) dedicato, ovvero un programma apposito che permette la programmazione e il caricamento dei dati nella scheda. I file generati hanno estensione *.ino* e vengono chiamati "sketch". Vengono quindi compilati secondo uno schema preciso, e vengono suddivisi in 3 parti principali:

- "**L'introduzione**" dove vengono importate le librerie, settati i valori predefiniti e definite le variabili;
- ***"void setup()"*** dove vengono definiti dei valori standard iniziali dello sketch, ovvero quelle funzioni che devono essere eseguite solo una volta;
- ***"void loop()"*** dove viene definito il programma per intero che, come definisce il nome, si ripeterà all'infinito.

Come già accennato in precedenza, il linguaggio utilizzato da Arduino è un derivato del C e C++ più "leggero", in modo da rendere molto più semplice ed intuitivo il controllo della scheda. All'interno dello sketch sono inseribili vari elementi al fine di creare un programma che possa poi svolgere le operazioni desiderate. In particolar modo si definiscono:

- Le **variabili**, che sono essenzialmente i dati a cui si può attribuire un nome e un valore specifico e sono utilizzate sia come valori fissi che come valori modificabili nello sketch. Possono essere dei semplici numeri interi (con la necessità di dichiararli attraverso la funzione *int*), valori decimali (quindi con la possibilità di avere numeri più precisi con la virgola e definiti tramite il comando *float)* e anche valori booleani (valori che derivano dall'algebra booleana e che possono quindi assumere valore *TRUE* o *FALSE* definiti dalla funzione *boolean*);
- Le **funzioni**, che compongono il codice in sé, indicano come vengono utilizzate le variabili e i pin di Arduino. Vengono collegate fra di loro e spesso si usa una sola di esse per poi poter eseguire un intero blocco di funzioni.
- I **commenti** non sono una parte fondamentale per il funzionamento del programma, ma sono molto utili per segnare o spiegare alcune righe di comando. Preceduti da un doppio slash (//), sono semplici testi che vengono ignorati dal programma, e che quindi hanno solamente un'utilità informativa.

2 PROGETTAZIONE

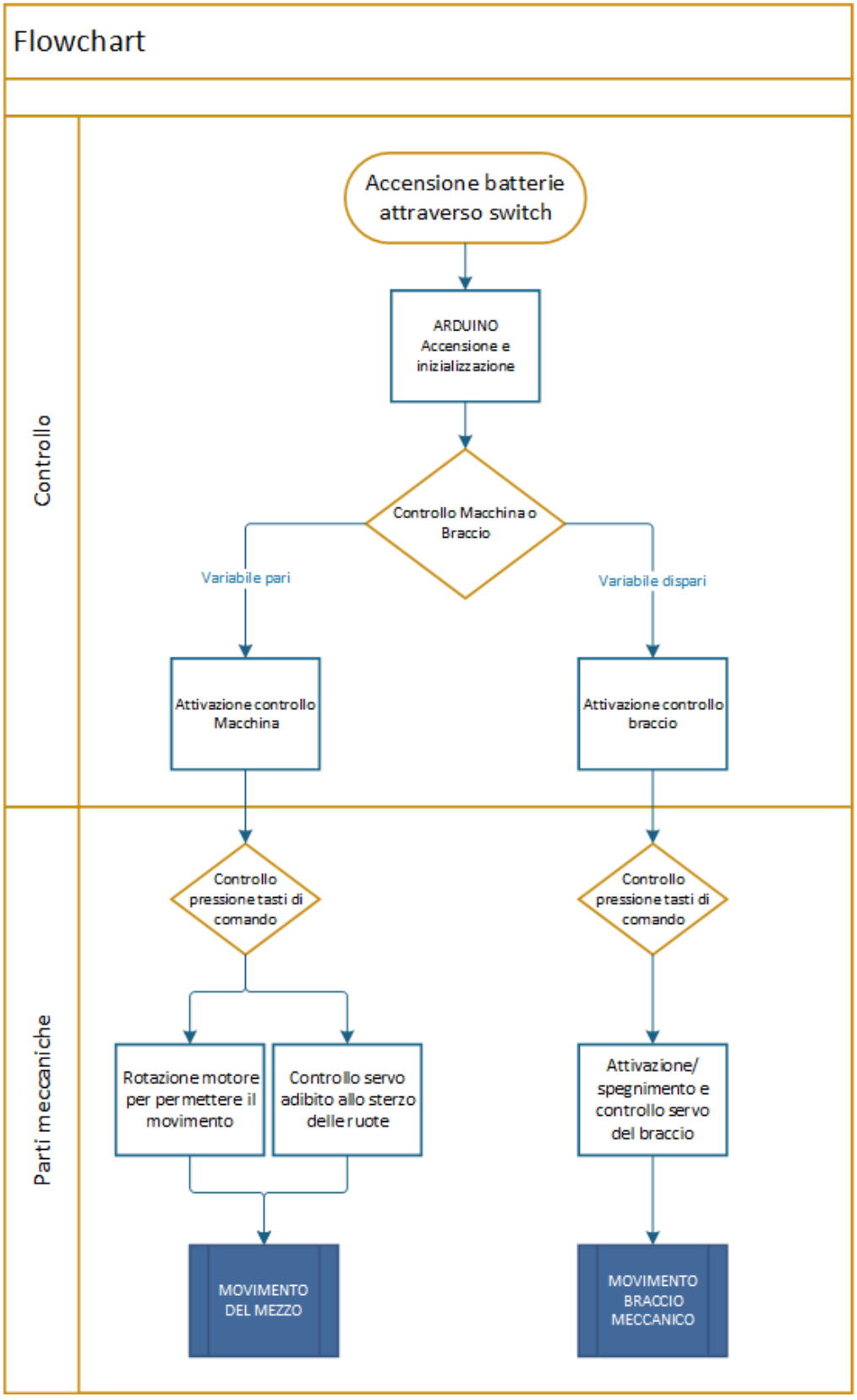
2.1 INTRODUZIONE

Il progetto realizzato consiste in un braccio formato da 7 servomotori montato su una macchina comandata a distanza. Il tutto si basa sull'interfacciamento di Arduino con i componenti e con il controllo remoto. Dovendo essere comandato attraverso un controller di una Xbox 360, è stato usato uno shield, ovvero una board compatibile con Arduino che aggiunge diverse funzionalità. In questo caso è stato integrato un host USB che, attraverso l'uso delle apposite librerie, è in grado di collegarsi al controller wireless attraverso un adattatore.

Connesso il dispositivo e acceso il controller, è necessario che Arduino si colleghi ad esso e che inizi la ricezione dei dati. Tramite i tasti presenti sul controller è possibile comandare sia il braccio che la macchina, non contemporaneamente per evitare correnti troppo alte o disturbi. Si può quindi controllare il motore della macchina con le varie marce e il servo utilizzato per sterzare, oppure controllare i 7 servomotori che compongono il braccio e che gli permettono di eseguire svariati movimenti. Trattandosi di una macchina deve potersi muovere senza l'utilizzo di fili ed è quindi necessario usare delle batterie per alimentare il tutto. In particolare la scelta si è volta su 2 tipi diversi di batterie:

- Per quanto riguarda l'alimentazione base di Arduino e dei servomotori (5 volt) è stata utilizzata una batteria da 7.4V con tensione abbassata grazie ad un regolatore step-down;
- Il motore, invece, il quale deve essere in grado di "trainare" l'intera struttura, è collegato ad un pacco di tre batterie in serie da 9V ciascuna, che fornisce una tensione abbastanza elevata da garantire al motore la giusta coppia.

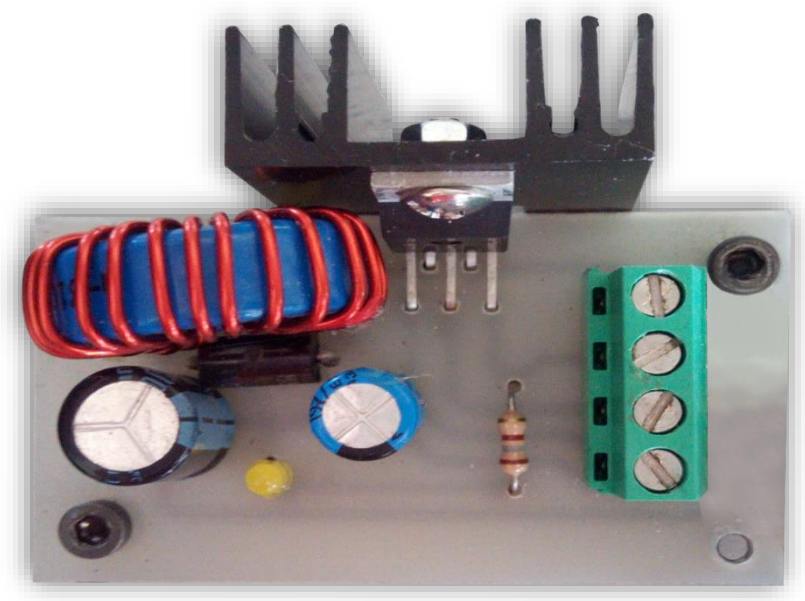
2.2 SPIEGAZIONE BLOCCHI



Per collegare i vari componenti del circuito sono state realizzate delle schede PCB apposite per l'interfacciamento di ognuno di essi. Create tramite l'utilizzo del software *Eagle*, sono elencate e suddivise per funzioni nella parte sottostante.

2.3 CONTROLLO

La parte del controllo è formata fondamentalmente dal programma compilato e caricato in Arduino. Grazie allo sketch compilato e alle librerie usate, siamo in grado di gestire in modo efficiente tutte le funzioni realizzabili. È infatti possibile modificare diversi valori legati al motore, alla velocità dei servomotori e alla loro inclinazione. Inoltre essendo i tasti del controller assegnati a specifici indirizzi fisici, sono configurabili a piacere.
La connessione tra controller e macchina è di tipo wireless e si basa su un segnale radio su banda prioritaria a $2.4GH_z$. Questo particolare tipo di segnale evita che si incorra in disturbi di connessione.
I circuiti sono tutti alimentati a 5V, ciò significa che è stato necessario realizzare un convertitore DC/DC step-down (anche detto buck) per ottenere la tensione desiderata. L'integrato utilizzato è un LM2576 che ha la particolarità di richiedere pochi componenti esterni e una corrente di OUTPUT massima di 3A. Rispetto ad un abbassatore lineare, il buck ha una maggiore stabilità e miglior rendimento (intorno al 95%). Lo step-down risulta quindi essere un piccolo circuito mostrato nell'immagine seguente:

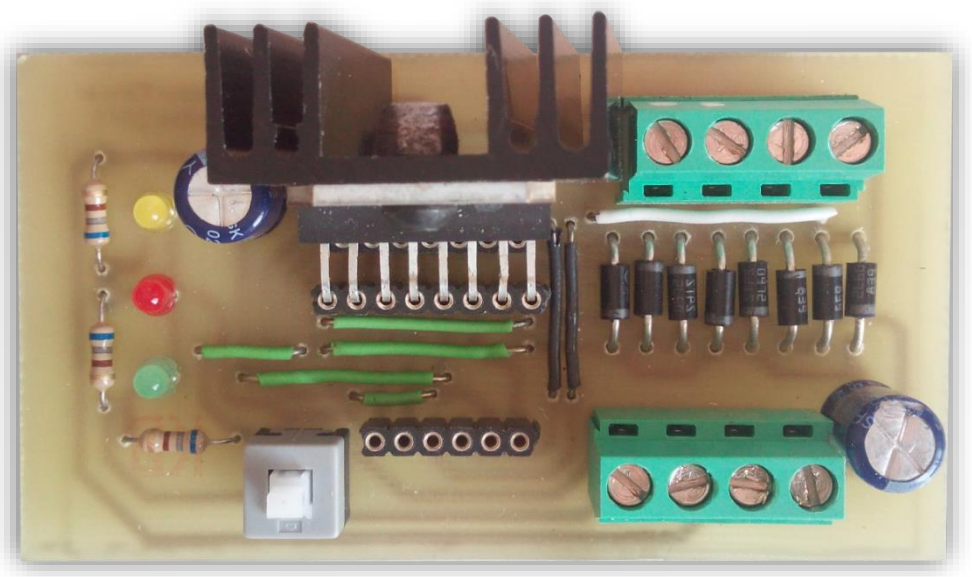


2.4 MACCHINA

La macchina è la base che permette il movimento. È dotata di quattro ruote motrici comandate da un motore e da un servomotore che permette a quelle anteriori di sterzare. Il servo viene alimentato alla tensione della batteria, raggiungendo quindi circa 2 volt oltre la tensione ideale di funzionamento, per fare in modo che esso generi abbastanza forza da riuscire a muovere le ruote sottoposte al peso della struttura. Il comando arriva invece direttamente da Arduino: viene gestito dal pin 8, attraverso il quale vengono definiti i diversi valori che hanno un'ampiezza totale poco inferiore a 180°.

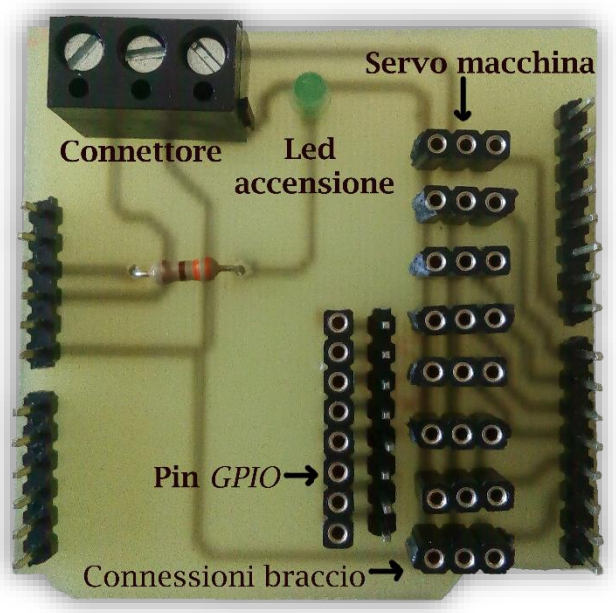
Per quanto riguarda il motore utilizzato è alimentato come spiegato prima ad una tensione più alta rispetto agli altri componenti del circuito. In questo modo aumentano i numeri di giri del motore il che comporta un miglior rapporto carico/corrente. Per poter controllare un motore in CC è necessario l'utilizzo di un "ponte". Grazie a questo particolare sistema è possibile controllare un motore utilizzando due ingressi che gestiscono il modo in cui deve essere alimentato. Si è in grado infatti di controllare il senso di rotazione del motore da due pin di Arduino in configurazione digitale. Per garantire questo tipo di controllo viene utilizzato un L298, un *ponte ad H*. Il vantaggio rispetto ad un altro tipo di ponte è la singola alimentazione del motore ed ovviamente il controllo semplificato. Il controllo avviene dai pin *GPIO* dello shield, utilizzabili caricando una libreria apposita in Arduino. In questo modo siamo in grado di avere accesso alla serie degli 8 pin che sono risultati essere essenziali, in quanto i restanti digitali erano inutilizzabili a causa dei collegamenti interni dello shield. Nel file della libreria è possibile notare che per rendere utilizzabili questi pin, è necessario riferirsi ad essi con il loro valore di registro rendendoli scrivibili. In questo modo la semplice dichiarazione della variabile assegnata permette di gestirli come qualsiasi altro pin semplicemente usando un altro comando.
Sono quindi usati tre degli otto *GPIO* disponibili, due per gestire la direzione del motore e uno per gestire l'enable. Questo particolare pin dell'integrato permette di spegnerlo o accenderlo attraverso un semplice segnale alto o basso. Nel nostro caso è stato anche usato per variare la velocità: facendo variare infatti il valore del pin digitale collegato tra 0 e 1, si viene a creare un segnale di onda quadra, e basta variarne il duty-cycle per modificarne la velocità.

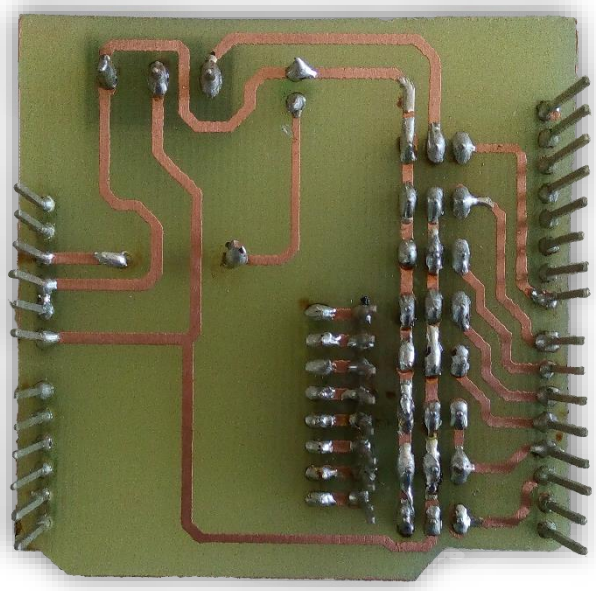
L'immagine del circuito una volta stampato il PCB:



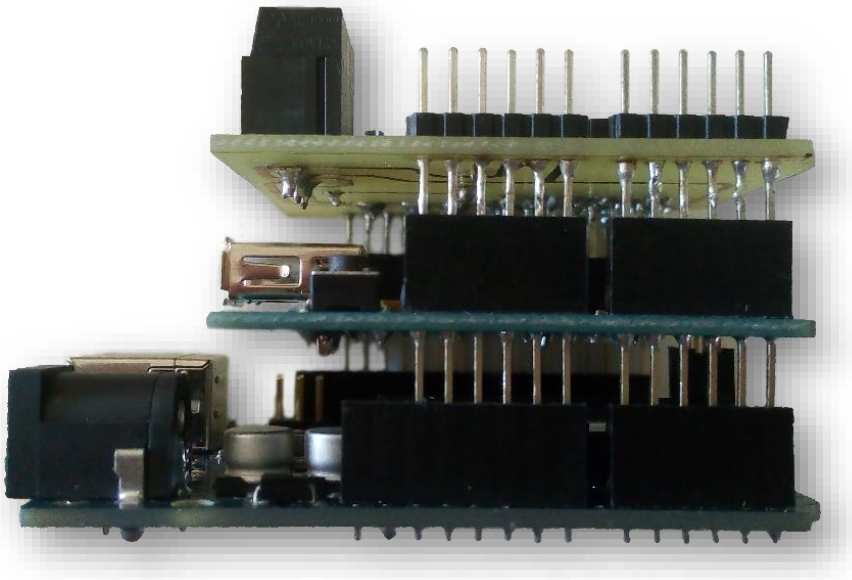
2.5 BRACCIO

Il comando del braccio è gestito dai pin dal 2 al 7 presenti su Arduino. In totale sono presenti sette servo (due della base sono collegati allo stesso controllo) e sono in grado di richiedere una corrente di picco di poco più di 2 Ampere. Ogni servomotore è collegato ai 5V forniti dallo step-down e al rispettivo pin di controllo settato in Arduino. Per evitare complessi e pericolosi collegamenti molto lontani fra loro, abbiamo realizzato un "adattamento" che si collega direttamente sopra lo shield. In questo modo abbiamo a disposizione un punto fisso dal quale partono tutti i collegamenti, dove viene già fornita l'alimentazione ad Arduino ed ai servomotori portando un solo cavo dal buck. Di seguito l'immagine dello shield realizzato:





Collegando quindi sia lo shield USB che quello da noi costruito per l'adattamento dei servomotori è risultato un blocco unico multifunzione:



3 CONTROLLO

Iniziamo a spiegare parte per parte il codice che permette la gestione del progetto realizzato.

3.1 INTRODUZIONE LISTATO

- Come accennato prima lo sketch di Arduino è composto da tre parti di cui la prima è quella dove vengono importate le librerie e settati i valori di default:

```
#include <Usb.h>
#include <UsbCore.h>
#include <usbhub.h>
#include <XBOXRECV.h>
#include <USB_Host_Shield_GPIO.h>
#include <Servo.h>
```

Come già spiegato prima, l'integrazione delle librerie avviene attraverso il comando *#include*. In particolare è possibile notare l'integrazione delle librerie necessarie all'interfacciamento con l'USB dello shield, alla connessione con il controller attraverso il ricevitore, alla corretta lettura dei pin *GPIO* e quella per l'adattamento dei servomotori.

```
#define SERVO1 2   //  Trigger destro e sinistro (R2,L2): Base
#define SERVO2 3   //  L1/R1            : Piega base
#define SERVO3 4   //  RightHat X       : Giunzione del gomito
#define SERVO4 5   //  RightHat Y       : polso
#define SERVO5 6   //  LeftHat X        : Rotaz. Pinza
#define SERVO6 7   //  LeftHat Y        : Pinza
#define SERVO_CAR 8 //  L1/R1           : Servo sterzo

#define EN 0;        //enable 1,2,3,4
#define in1 1;       //input buffer 1 e 4
#define in2 2;       //input buffer 2 e 3
```

Grazie al comando *#define*, è possibile dare un nome ad una costante prima della compilazione del programma. In questo modo è possibile riferirsi ad un certo valore settato inizialmente semplicemente richiamandone il nome. Nell'esempio vengono definiti i pin ai quali verranno collegati i servomotori del braccio, assegnando ad ognuno il nome del servomotore corrispondente. La serie di define successivi servono a impostare i valori predefiniti delle velocità dei servomotori, quindi gli step di incremento, i valori minimi e massimi che possono raggiungere, le posizioni iniziali e la direzione di rotazione. Grazie a questi dati è possibile gestire in modo molto dettagliato tutti i valori modificabili per quanto riguarda i servomotori e in particolare i valori iniziali con la possibilità di settare una posizione precisa in cui dovrà impostarsi all'accensione.

Inoltre allo stesso modo vengono definiti i pin a cui vanno collegati gli input per il controllo della direzione del motore e l'enable della macchina. Per quanto riguarda quest'ultimo ricordiamo che è attivo basso e che quindi necessita di un ingresso alto per poter funzionare. Questo controllo viene dato da Arduino attraverso il comando apposito.

```
USB Usb;
XBOXRECV Xbox(&Usb);
GPIO motor;
```

Questi semplici comandi non sono altro che l'assegnazione di un nome per richiamare le librerie aggiunte. Solitamente i comandi che le interessano iniziano con il nome della libreria, un punto e poi il comando.

```
int s1 = SERVO1_INIT;  //Variabili – posizione iniziale servo
int s2 = SERVO2_INIT;
int s3 = SERVO3_INIT;
int s4 = SERVO4_INIT;
int s5 = SERVO5_INIT;
int s6 = SERVO6_INIT;

int var;
int K;
int Sterzo;
```

Con i comandi di *int*, si definiscono delle variabili intere, in particolare quelle iniziali che verranno poi modificate nel corso del programma. Vengono quindi definite delle variabili che corrispondono alle posizioni iniziali dei servo del braccio, necessarie in quanto poi il braccio si muoverà sfruttando questi valori e modificandoli singolarmente. Vengono anche definite alcune variabili necessarie allo svolgimento dello sketch di cui verrà spiegato più avanti il funzionamento in modo più dettagliato.

```
Servo car;
Servo servo1;    //Associamo ai servo dei nomi
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;
Servo servo6;
```

L'insieme delle funzioni *Servo* è definito dalla libreria apposita e questo tipo di scrittura permette di assegnare dei nomi ai servomotori utilizzati, in modo che richiamando il nome venga riconosciuto come servo. Grazie alla libreria, i servomotori sono comandabili tramite il comando *'nome_servo'.write()*, dove è necessario specificare il nome assegnato e all'interno della parentesi il valore dell'angolo che deve raggiungere in gradi.

3.2 VOID SETUP()

- Di seguito è presente la parte composta da pochi ed essenziali comandi, racchiusa nel *Setup*:

```
void setup() {
        Serial.begin(115200);                    //Avvio della comunicazione con USB
        while (!Serial);
        if (Usb.Init() == -1) {
                Serial.print(F("\r\nAdattatore non collegato"));
                        while (1);
                }
```

```
Serial.print(F("\r\nController Connesso"));
}
```

Nel setup vengono definite le funzioni che devono essere eseguite solo una volta. In questa definizione rientra il collegamento del controller all'adattatore, che avviene attraverso un comando della libreria apposita. Si inizializza con *'Serial.begin(115200)'* la connessione seriale che avviene tra Arduino e un computer collegato via USB. Questo comando non è strettamente necessario al funzionamento del circuito, ma è stato utile durante i test per controllare l'esecuzione delle funzioni. Stesso discorso vale per i *print* che avvengono sulla seriale in quanto non viene usata nel progetto.

La parte fondamentale è decisa da una funzione *if*, cioè un insieme di funzioni che vengono eseguite solo quando vengono rispettate le condizioni imposte. In questo caso viene richiesta una lettura diretta sulla porta USB dello shield per verificare se è collegato il ricevitore (usando quindi una funzione con richiamo alla libreria settata in precedenza). Se la condizione è verificata si entra in un loop che avvisa del collegamento mancante. In caso contrario se è tutto già collegato in modo opportuno la funzione non viene eseguita e si procede con il resto dello sketch.

3.3 VOID LOOP()

- La parte più lunga è certamente l'insieme di funzioni che utilizzano i dati ricevuti dal controller per modificare variabili e fare muovere il sistema. L'insieme di questi comandi è racchiuso nel *'void loop()'* ed è la struttura che si ripete continuamente:

```
Usb.Task();
```

La prima operazione da compiere è necessaria per poter avviare la comunicazione la porta USB dello shield. Tramite la funzione che richiama la libreria specifica, viene creato un *Task*, termine inglese che indica un lavoro, un incarico che viene creato sulla pota USB per poi permetterne l'utilizzo.

```
if (Xbox.XboxReceiverConnected) {    //Connessione con Controller
```

```
for (byte i = 0; i < 2; i++) {
    if (Xbox.Xbox360Connected[i]) {
```

Una volta richiesto all'USB un collegamento, è necessario controllare se è effettivamente stato connesso un ricevitore. Richiamando la funzione della libreria del controller si può quindi avviare un ciclo che funziona solo quando è effettivamente connesso al ricevitore almeno un controller. La seconda funzione fornisce la possibilità di connettere più di un controller attraverso un ciclo *for*. Questa funzione esegue un gruppo di comandi per un numero di volte preciso definito nelle condizioni del ciclo. In questo caso viene creata una variabile *i* che inizialmente è 0, e ad ogni ciclo viene incrementata di 1 fino ad un massimo di 1 (deve essere infatti <2). Questo fa in modo che sia possibile collegare in contemporanea due controller che possono eseguire le funzioni presenti nel loop ognuno in maniera separata.
Viene infine iniziato il ciclo principale dove viene collegato effettivamente il controller.

```
if (Xbox.getButtonClick(R3, i)) {
    var++;
    }
```

La prima funzione incontrata è molto semplice: alla pressione di un tasto (in questo caso l'analogico destro R3) si incrementa una variabile. Usare il simbolo ++ accanto alla variabile corrisponde ad eseguire la funzione *'variabile + 1'.* Il motivo di questa funzione è utilizzato per evitare che il braccio funzioni assieme alla macchina e vice versa.

```
servo1.write(SERVO1_INIT);
  delay(200);
    servo2.write(SERVO2_INIT);
  delay(200);
    servo3.write(SERVO3_INIT);
  delay(200);
```

```
  servo4.write(SERVO4_INIT);
delay(200);
  servo5.write(SERVO5_INIT);
delay(200);
  servo6.write(SERVO6_INIT);
  delay(200);
  }
```

All'interno dello stesso ciclo che decide se si attiva la parte della macchina o quella del braccio, sono inseriti i comandi sopra riportati. In questo modo siamo in grado di ristabilire una posizione iniziale di sicurezza in cui il braccio deve tornare prima di scollegarlo. Questo perché nel caso in cui il braccio fosse in una posizione particolare, e venisse scollegato per poter usare la macchina, si rischia che i servomotori si lascino andare facendolo cadere di colpo. Si vuole quindi evitare questo caso facendo tornare il braccio in una posizione apposita (pur dovendo aspettare un po' di tempo in più).

```
if (var%2 == 0) {          //Loop per il funzionamento della macchina
```

La variabile precedente viene ripresa come condizione di funzionamento di un ciclo. Attraverso il simbolo di percentuale seguito da un numero è possibile ottenere il resto che risulterebbe da una divisione per quel numero. In questo caso l'intento era quello di avere solo due valori, cioè di rilevare se il valore è un numero pari o dispari. Eguagliare l'equazione a zero permette di eseguire in modo efficiente questa divisione. Grazie alla differenziazione di queste due tipologie di valori viene permesso al sistema di far lavorare la macchina per tutti i numeri pari, scollegandolo per utilizzare il braccio appena viene incrementata la variabile e diventa quindi dispari.

```
servo1.detach();          //Scolleghiamo tutti i servo del braccio
servo2.detach();
servo3.detach();
servo4.detach();
servo5.detach();
servo6.detach();
```

car.attach(8);

Con le funzioni sopra riportate si interagisce con i servomotori garantendo il loro

```
if (Xbox.getButtonPress(L1, i) && Sterzo <= 180) {
        Sterzo = Sterzo + 3;
        car.write(Sterzo);
        } else if (Xbox.getButtonPress(R1, i) && Sterzo >= 0) {
                Sterzo = Sterzo - 3;
                car.write(Sterzo);
                }  else if (Xbox.getButtonPress(L1, i)==0 && Xbox.getButtonPress(R1, i)==0) {
                        delay(5);
                        Sterzo = 90;
                        car.write(Sterzo);
                        }
```

scollegamento dalla tensione per questioni di sicurezza. Ovviamente i comandi riguarderanno i servomotori creati all'inizio dello sketck e verrà loro settato il comando *detach*, ovvero distacco.

L'insieme delle funzioni sopra riportate è il "blocco" che permette alla macchina di sterzare. Con la prima funzione *if* si rilevano due condizioni necessarie allo svolgimento del listato seguente: in primo luogo è necessario che lo sterzo non sia già al massimo valore ottenibile, ovvero ruotato a 180°, e questa condizione può essere evitata controllando che la variabile *'Sterzo'* sia minore di quel valore. L'altra condizione necessaria è ovviamente il tasto che deve essere associato al movimento, in questo caso il tasto nella parte posteriore a sinistra del controller, ovvero il tasto *L1*. Le stesse condizioni vengono poi inserite in un *if* usato per ruotare lo sterzo dalla parte opposta, con la differenza che verrà usato il tasto *R1* e verrà controllato che il valore dello sterzo sia maggiore di 0. Le condizioni che fanno rientrare nei limiti sono semplicemente un margine di sicurezza, in quanto bloccano la procedura una volta arrivati al valore limite del servo, evitando di continuare ad inviare dati che non possono essere letti. All'interno di questi cicli vengono definite delle condizioni di aumento o decremento del valore che verrà poi scritto sul servomotore per farlo spostare. Dovendo avere un incremento graduale dell'angolazione del servo, si pone la variabile Sterzo come composta dalla variabile

stessa sommato o sottratto qualcosa. In questo modo la variabile incrementa di un tot ogni volta che viene ripetuto il ciclo nel quale è contenuta. Si usa poi il comando *car.write(Sterzo)* per scrivere sul pin collegato al servo della macchina il valore che incrementa gradualmente. L'ultima funzione è quella che viene svolta nel momento in cui nessuno dei tasti assegnati viene premuto e deve essere quindi fornita la condizione secondo la quale il valore dei tasti è uguale a 0. Quest'ultima funzione come quella precedente sono introdotte da un'ulteriore funzione chiamata else. È una particolare funzione usata dopo la chiusura di un ciclo *if* per indicare un gruppo di funzioni utilizzabili nel caso in cui non vengano rispettate le condizioni precedenti. In questo caso viene utilizzata per concatenare un ulteriore *if* in modo che solo un ciclo alla volta può essere eseguito.
Se il sistema rimane senza comandi, ovvero la condizione rispettata è quella dell'ultimo *if*, allora il servo della macchina torna alla posizione iniziale di 90°.

Spieghiamo ora l'utilizzo del motore attraverso il controllo del ponte ad H per regolarne la direzione e la velocità.

```
if (Xbox.getButtonPress(R2, i)>250) {
        motor.write(Input_2, LOW);
        motor.write(Input_1, HIGH);
```

Una prima funzione *if* controlla se viene premuto il trigger destro. Per una maggiore precisione, e per evitare che le variazioni troppo rapide del tasto non vengano lette, è stato settato che deve essere premuto oltre ad un certo valore. La particolarità dei trigger è infatti la possibilità di essere considerati come *slider*, quindi con valori variabili. Sfruttando il fatto che la pressione fa variare il valore da 0 a 255, è stato settato un margine di sicurezza a 250. Già nelle funzioni precedenti sono state utilizzate due tipi di funzioni diverse per quanto riguarda la pressione dei tasti. Vi è infatti una differenza tra *.getButtonClick()* e *.getButtonPress()*: mentre la prima permette di leggere una pressione singola del tasto, fornendo quindi un impulso, la seconda continua a leggere se il tasto rimane premuto. In questo modo è possibile differenziare le funzioni dove ad esempio è necessario premere un tasto una volta per modificare una variabile (come nel caso della funzione per stabilire se utilizzare la macchina o il braccio) e quelle invece incrementali dove la pressione prolungata permette di variare in modo graduale dei valori. La funzione esposta

sopra permette di entrare in un ciclo che viene ripetuto quando viene mantenuto premuto il tasto R2. La prima cosa da fare è settare la direzione nella quale dovrà poi girare il motore. Per fare ciò ci avvarremo della funzione che richiama i pin di GPIO spiegati all'inizio. Essi saranno poi gli ingressi di comando del ponte ad H ed è necessario che vengano considerati pin digitali. Viene quindi settato uno dei pin a livello basso mentre l'altro a livello alto per settare la direzione e fare procedere in avanti la macchina.

```
if(Xbox.getButtonClick(A)) {
        K = K + 1;
        while(K>3){
                K = 3;
                }
        }
if(Xbox.getButtonClick(B)) {
        K = K-1;
        while(K<0){
                  K = 0;
                }
        }
```

Le funzioni successive sono inglobate nello stesso ciclo che consente il movimento della macchina in avanti. Esse permettono di modificare la velocità di rotazione del motore, situazione non disponibile inserendo la retromarcia. Questi comandi permettono di variare una variabile che aumenta o diminuisce alla pressione di tasti predefiniti, rispettivamente A e B. Premendo quindi il corrispondente fa salire la marcia, la variabile viene incrementata per poter essere utilizzata successivamente. La particolarità di queste funzioni è data dalla necessità di limitare il numero di marce possibili in quanto la variabile tenderebbe altrimenti ad assumere valori insensati. Bisogna quindi fare in modo che la variabile (con valore iniziale 0) venga incrementata alla pressione del tasto A tramite la funzione *K = K + 1,* e allo stesso modo decrementata alla pressione del tasto B. In entrambi i casi vengono poste funzioni in modo che non possano essere assunti valori superiori a 3 (il numero

delle marce disponibili) o inferiore a 0. Si utilizza una funzione *while*, che permette di svolgere un ciclo finché viene rispettata la condizione assegnata. Questo significa che è possibile impostare i massimi e i minimi e che nonostante le variazioni della variabile essa possa variare solo nell'intervallo.

```
switch (K) {
    case 1:
      Serial.println("Marcia prima");
      motor.write(EN, HIGH);
      delay(15);
      motor.write(EN, LOW);
       delay(1);
      break;
    case 2:
      Serial.println("Marcia seconda");
      motor.write(EN, HIGH);
      delay(50);
      motor.write(EN, LOW);
      delay(1);
      break;
    case 3:
      Serial.println("Marcia terza");
      motor.write(EN, HIGH);
      break;
    default:
      K = 0;
      }
```

Le funzioni proposte definiscono quindi la marcia inserita e di conseguenza la velocità alla quale gira il motore. Per poter distinguere i vari casi, si usa una funzione apposita: *switch()*. All'interno della parentesi graffa collegata vengono inserite le funzioni da eseguire sotto dei comandi detti *case*. In base al numero scritto accanto verranno eseguite solo le funzioni successive al valore corrispondente fino al break. Inserendo quindi nelle parentesi della funzione *switch* la variabile da noi modificata

volta per volta, siamo in grado di assegnare un particolare ciclo da eseguire per ogni caso. Per poter variare la velocità in questo modo è stato necessario controllare il driver attraverso un segnale *PWM.* Quest'ultimo viene ottenuto semplicemente facendo variare l'enable (il quale è un pin digitale), modificando il tempo in cui sta alto e quello in cui sta basso attraverso l'uso di *delay,* ovvero attese di definiti millisecondi. Questo fa sì che sul pin si crei un'onda quadra a duty-cycle variabile.
Tra i vari casi selezionabili, vi è il cosiddetto *default*, il quale racchiude l'insieme delle funzioni da eseguire per tutti quei valori diversi da quelli presenti nei casi precedenti.

```
} else if (Xbox.getButtonPress(L2, i)>250) {
        Serial.println("guiuuu");
        motor.write(EN, HIGH);
        motor.write(Input_2, HIGH);
        motor.write(Input_1, LOW);
        } else if (Xbox.getButtonPress(L2, i)<200 && Xbox.getButtonPress(R2, i)<200) {
                delay(5);
                motor.write(Input_1, LOW);
                motor.write(Input_2, LOW);
                K=0;
                }
```

Complementare alla funzione precedente è quella che permette di fare ruotare il motore nel senso opposto, innestando quindi la retromarcia. Non avendo la necessità di variare la velocità, viene posto l'*enable* sempre alto e vengono invertiti i valori dei pin di controllo. Infine è presente anche la funzione di controllo che viene attivata quando non viene premuto nessuno dei tasti per il movimento. Come nel caso della posizione iniziale del servo di prima, anche qui si riporta il sistema alle condizioni di base, reimpostando la variabile K a 0 (togliendo quindi l'eventuale marcia ingranata) e settando i pin di controllo a livello basso per evitare qualsiasi movimento al motore.

La parte vista sopra era tutto ciò che riguardava il controllo della macchina. Ora, per spiegare il funzionamento del braccio, è necessario riprendere dalla funzione

che verifica il numero pari. La spiegazione di seguito è, quindi, racchiusa in una funzione *else{}*, cioè attiva per tutti i valori non divisibili per due e quindi dispari.

```
motor.write(EN, LOW);          //Loop del braccio, tutti gli elementi della macchina vanno scollegati
car.detach();
motor.write(Input_1, LOW);
motor.write(Input_2, LOW);
```

I primi comandi servono semplicemente come sicurezza, in quanto lasciando collegati i componenti della macchina potrebbero presentarsi disturbi. È preferibile quindi verificare che tutti i pin del controllo del ponte ad H siano a livello basso (compreso l'enable il quale dovrebbe spegnerlo) e che il servo della macchina sia scollegato dalla tensione.
Successivamente per il funzionamento del braccio bisogna innanzitutto mettere in tensione i servo attraverso il comando *nome_servo.attach()*. Sia per questioni di sicurezza che per fare in modo di avere dei tasti riservati all'accensione e allo spegnimento, abbiamo deciso di assegnare il collegamento di tutti i servo attraverso la pressione del tasto A.

```
if (Xbox.getButtonClick(A, i))  {
        servo1.attach(2);
        delay(100);
        servo2.attach(3);
        delay(100);
        servo3.attach(4);
         delay(100);
        servo4.attach(5);
        delay(100);
        servo5.attach(6);
        delay(100);
        servo6.attach(7);
         delay(100);
        }
```

E come è ovvio pensare sarà il tasto B a permettere lo scollegamento complessivo dei servomotori utilizzando una serie di funzioni *detach.*
Durante le prove effettuate è stato riscontrato che non inserire dei ritardi tra il collegamento di un servo e l'altro provoca delle problematiche. Il listato cercherebbe di collegarli nello stesso momento provocando una corrente di spunto richiesta molto alta e il conseguente scollegamento momentaneo di Arduino. A causa di questo problema si vedrebbero quindi i servomotori muoversi con un piccolo scatto (entrando in tensione), ma scollegarsi subito dopo a causa della mancanza di alimentazione di Arduino che non permette l'effettivo collegamento. Grazie alle funzioni sopra proposte, si ritarda ogni collegamento in modo che i servo vengano messi in tensione uno alla volta eliminando quel problema.

Il listato mostra poi i comandi rispettivi di ogni servomotore, che sono pressoché simili e basta quindi spiegarne uno in generale:

```
if (Xbox.getButtonPress(L2, i)) {
        s1 = s1 + SERVO1_STEP * SERVO1_DIRECTION;
        } else if (Xbox.getButtonPress(R2, i)) {
                s1 = s1 - SERVO1_STEP * SERVO1_DIRECTION;
                }
```

Nell'esempio viene presentato il movimento che fa da base rotante su cui poggia il braccio. Essa è controllata dai tasti *R2* e *L2* e la funzione specifica è quella poi che si ripete nei comandi degli altri servomotori. È composta da un valore incrementale, che si ottiene ponendo una variabile uguale a se stessa più qualcosa. In questo caso tenendo premuto il tasto ogni volta che viene compiuto il ciclo si somma lo step definito all'inizio moltiplicato per la direzione nella quale deve essere rivolto il movimento.
Discorso analogo anche per quanto riguarda le parti controllate dagli stick analogici, i quali non possono venir richiamati dalle funzioni *.getButtonClick* o *.getButtonPress.* essendo valori variabili, viene controllato che il loro valore sia superiore alla *DEADZONE*, cioè lo spostamento minimo della levetta per poter essere letta dal programma come dato.

```
if (Xbox.getAnalogHat(RightHatX, i) > DEADZONE || Xbox.getAnalogHat(RightHatX, i) < DEADZONE) {
        s3 = s3 + Xbox.getAnalogHat(RightHatX, i) * SERVO3_RATE * SERVO3_DIRECTION;
        }
```

Si prenda ad esempio il servo del braccio. Si controlli che il valore sia maggiore o minore del valore prefissato di *deadzone* (nel caso del controller Xbox è di circa 7500) e si fa esattamente la funzione di prima con la semplice aggiunta di un valore. Questo incremento è portato dal valore letto dall'analogico e quindi dipende da quanto viene spostato dalla posizione iniziale. Nel caso dei servomotori comandati tramite analogico è quindi possibile variarne la velocità di rotazione semplicemente inclinando maggiormente lo stick nella direzione desiderata.
Dopo ogni comando, e quindi modifica delle variabili dei servomotori, esse vengono controllate e settate al valore massimo o minimo raggiungibile appena superati.

```
if (s1 > SERVO1_MAX) s1 = SERVO1_MAX;
if (s1 < SERVO1_MIN) s1 = SERVO1_MIN;
```

Questo garantisce il controllo dei valori dei servomotori in modo da evitare di forzarli a valori che non possono raggiungere.

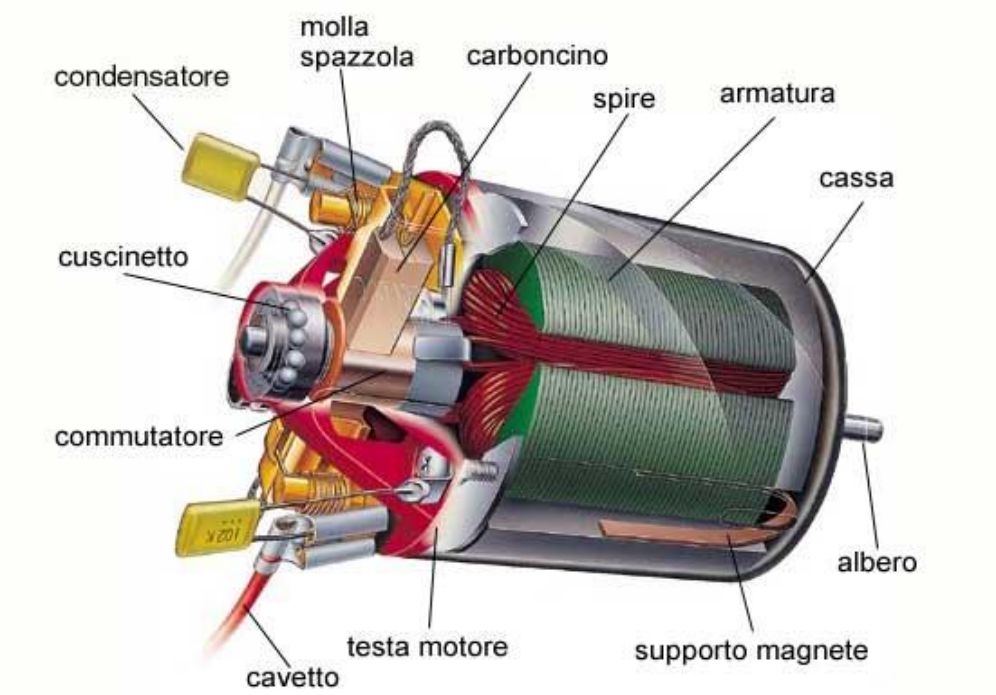
```
if (Xbox.getButtonClick(START, i)) {      //Reset dei Servo alla posizione iniziale
        s1 = SERVO1_INIT;
        s2 = SERVO2_INIT;
        s3 = SERVO3_INIT;
        s4 = SERVO4_INIT;
        s5 = SERVO5_INIT;
        s6 = SERVO6_INIT;
        }
```

Altra possibile funzione di un circuito simile è la possibilità di fare tornare il braccio alla posizione di partenza semplicemente premendo un singolo tasto. È stato quindi utilizzato il tasto *START* per assegnare ai servomotori le posizioni iniziali.

Infine è necessario scrivere questi nuovi valori sul servomotore corrispondente per farlo muovere nella posizione desiderata. Dei semplici comandi di *write* alla fine del ciclo svolgono la funzione permettendo di scrivere ogni volta la variazione di un valore.

```
servo1.write(s1);
servo2.write(s2);
servo3.write(s3);
servo4.write(s4);
servo5.write(s5);
servo6.write(s6);
```

4 MACCHINA



Il mezzo viene mosso da un motore in corrente continua e da due riduttori di velocità per garantire uno spostamento sia sotto carico che in presenza di movimenti effettuati dal braccio. Innanzitutto il motore è un attuatore che converte una potenza elettrica in una meccanica, garantendo quindi un movimento rotatorio.

Questi dispositivi vengono impiegati ad esempio nei servosistemi oppure nel settore dell'automazione, specie nelle catene di montaggio per muovere i bracci robotici.

Esistono vari tipi di motori:

- A corrente continua a magnete permanente
- A corrente continua a statore alimentato
- Passo - passo
- Brushless

Nel progetto verrà utilizzato un motore della prima tipologia, in modo da dover gestire potenze inferiori al kW (rimanendo a qualche centinaio di watt) e che quindi lo rendono alimentabile attraverso una batteria.

4.1 FUNZIONAMENTO

Questa tipologia di motori, così come quella a statore alimentato, si basa sull'interazione di due campi magnetici che si attraggono e respingono a vicenda.

I due motori differiscono però in alcuni particolari:

- **meccanicamente**, il campo magnetico viene generato tramite un materiale ferromagnetico e non degli avvolgimenti;
- **tecnicamente**, gli avvolgimenti se collegati in serie o in parallelo, formano una curva coppia-velocità diversa, oltre alla possibilità di variare la pendenza della caratteristica esterna.

Le parti meccaniche fondamentali di questo attuatore sono caratterizzate da una parte rotante detta rotore ed una fissa detta statore. Il motore a corrente continua mette in pratica fondamentalmente la legge di Lorenz: secondo questa legge, ogni filo di lunghezza $\boldsymbol{l}$ percorso da una corrente $\boldsymbol{I}$, ed immerso in un campo magnetico $\boldsymbol{B}$, è soggetto ad una forza fisica. La direzione di tali vettori è descrivibile secondo la "regola della mano sinistra".

Questa regola però subisce una leggera variazione nel momento in cui il campo magnetico rimane vincolato alla cassa statorica, e quindi le forze vettoriali non risultano più perpendicolari.

Pertanto tramite leggi vettoriali e trigonometriche, la formula subisce una variazione diventando

$$F = l * I * B * sen(\alpha)$$

Osservando questa legge si può affermare che la forza dipende dall'angolo formato dalla spira rispetto al campo, dalla corrente che attraversa il filo e dalla lunghezza dello stesso. Succede quindi che la spira, posta in una particolare posizione, è immersa totalmente nel flusso magnetico, e si genera quindi la forza massima. Allontanandosi da tale posizione la forza diminuisce, fino a diventare nulla. Questo però non permette al motore di ruotare con continuità perché la spinta avverrebbe

solo su metà giro. Quindi per ovviare a questo problema, e mantenere il motore in rotazione, bisogna utilizzare almeno due spire poste in maniera incrociata. In questo modo quando una spira è fuori dal campo magnetico l'altra si trova immersa, in modo da mantenere il rotore sempre con una coppia applicata.

Per alimentare le spire si ricorre ad uno stratagemma meccanico caratteristico solo del motore a corrente continua, il **collettore**. Il collettore è costituito da una serie di lamelle di rame poste vicine tra loro ed ognuna isolata elettricamente dalle altre che collegano un capo di ogni spira. Sopra il collettore strisciano due spazzole, poste a 180° tra loro, le quali forniscono l'alimentazione elettrica alle spire solo nella posizione in cui ricevono la massima spinta.

4.2 MODELLIZZAZIONE

In una modellazione matematica di un motore cc può essere visto come l'insieme di due parti fondamentali, una elettrica ed una meccanica:

- La *parte elettrica* è caratterizzata dalla resistenza e dall'induttanza presenti nelle spire avvolte sul rotore. Il primo componente è causato dalla presenza di un materiale metallico conduttore come il rame che, a causa di piccole perdite all'interno di esso, è necessario considerare come una resistenza. La componente induttiva è invece l'avvolgimento del filo a spirale presente nel motore.
- La *parte meccanica* è la componente fondamentale per le caratteristiche dinamiche, le quali possono dipendere dalla forma del rotore, dal materiale impiegato o dal peso stesso.

L'equazione generale della tensione presente ai capi di un motore è quindi:

$$V = R * i + L * \frac{di}{dt} + E$$

Si può notare come essa dipenda dalla tensione di alimentazione, da quella generata dalla non idealità del conduttore e dalle variazioni generate nell'induttanza.

Spieghiamo ora in modo dettagliato i vari tipi di funzionamento di un motore, suddividendoli in base al momento in cui viene studiato:

A. Funzionamento allo spunto

Il motore è inizialmente fermo e non si ha quindi velocità

$$\omega = 0 \text{ V/rad}*s^{-1}$$

Il campo magnetico rimane invariato e, facendo un rapido richiamo alla legge di Lenz, si può affermare che ad una variazione del campo magnetico che agisce su una spira si genera una forza controelettromotrice $\boldsymbol{E}$. Questa affermazione è data anche secondo il principio della conservazione dell'energia, secondo cui una forma energetica deve convertirsi in un'altra al fine di poter bilanciare le potenze in ingresso e in uscita, in quanto il motore altrimenti compirebbe un lavoro senza richiedere una certa potenza in entrata.

Partendo quindi dall'equazione di equivalenza tra le due potenze, si ha

$$F * \omega = E * I$$

dove:

- F = Forza
- ω = Velocità angolare
- E = Forza controelettromotrice
- I = Corrente

Si ricava che

$$E = \frac{F * \omega}{I}$$

e sostituendo la forza con la formula vista precedentemente $B * l * I$, data dalla legge di Lorenz, si ottiene

$$E = \frac{B * l * I * \omega}{I}$$

Semplificando la corrente, si ottiene:

$$E = B * l * \omega$$

il campo magnetico e la lunghezza del filo sottoposto ad esso vengono riuniti in un valore unico chiamato K_e , ovvero una costante elettrica che fornisce le caratteristiche interne del motore. Viene infatti definita come *costante di tensione* e viene espressa in $^{V}/_{rpm}$ (tensione su rotazioni per minuto).

Si evince quindi che la forza generata è nulla in questo caso:

$$E = K_e * \omega = K_e * 0 = 0V$$

Dopo aver calcolato la corrente massima allo spunto

$$I_s = \frac{V - E}{R} = \frac{V}{R}$$

si ottiene anche la coppia massima o di spunto

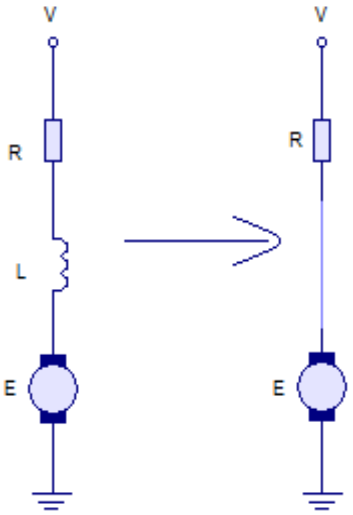
$$C_s = K_t * I_s = K_t * \frac{V}{R}$$

In questa funzione è possibile notare la costante K_t: è detta *costante di coppia* ed è anche in questo caso una costante del sistema con unità di misura $^{Nm}/_{A}$ (Newton metro per ampere). Generalmente la costante elettrica e quella meccanica coincidono, ovvero sono equivalenti e vale quindi la relazione

$$K_e = K_t$$

Si definisce coppia in quanto, come è stato detto in precedenza, sul rotore non viene applicata una sola forza, bensì una somma vettoriale di forze, per l'esattezza due per ogni spira.

B. Funzionamento a vuoto



Ipotizziamo ora che il motore sia in funzione da tempo ed abbia già raggiunto la velocità ideale. Quando la corrente è costante e la caduta di tensione sull'induttanza $\boldsymbol{L}$ è zero, l'induttanza stessa raggiunge il suo valore a regime essendo semplificata con un corto circuito. Il modello matematico si può quindi semplificare con:

$$V = R * I + E$$

Siccome la forza controelettromotrice aumenta all'aumentare dei giri del motore fino ad equivalere la tensione di ingresso, la corrente diminuisce perché la caduta di tensione maggiore la ha sul motore e non sulla resistenza.

Aumentando la velocità di rotazione $\boldsymbol{\omega}$, aumenta $\boldsymbol{E}$, per il seguente legame

$$E = Ke * \omega$$

La forza controelettromotrice sarà uguale alla tensione fornita quando il motore raggiungerà un certo numero di giri ($\boldsymbol{\omega_0}$).

Quando quindi $V = E$, si può scrivere $V = Ke * \omega$ per poi ricavare

$$\omega_0 = \frac{\mathrm{V}}{\mathrm{K_e}}$$

La corrente assorbita invece tenderà a zero perché la differenza di potenziale vista ai capi della resistenza tende a 0.

$$V = RI + E$$

$$I = \frac{V - E}{R}$$

Data la seguente relazione, e sostituendoci $V = E$, ci si trova davanti una frazione il cui numeratore vale 0 ed al denominatore si ha $\boldsymbol{R}$, una costante della macchina. È semplice quindi verificare che un motore senza forze esterne applicate raggiungerà quella rotazione tale per cui la corrente assorbita sarà 0 in quanto la differenza di potenziale ai capi sarà uguale a quella fornita dal generatore esterno.

C. Transitorio a carico

Durante questo transitorio, al motore viene applicato un carico resistivo mentre è a regime, quindi a velocità $\boldsymbol{\omega}$.

La coppia resistente del carico $\boldsymbol{C_l}$ deve essere minore della coppia di spunto $\boldsymbol{C_s}$ al fine di permette l'avvio della macchina. Si permettere al rotore di ruotare in condizioni in cui la forza controelettromotrice è minore.

$$E = Ke * \omega < V$$

Questo implica una riduzione di velocità e quindi $\boldsymbol{\omega < \omega_0}$ (a vuoto). $\boldsymbol{E}$, non essendo uguale alla tensione $\boldsymbol{V}$, crea i presupposti per avere una corrente costante maggiore rispetto quella a vuoto.

$$I = \frac{V - E}{R}$$

A sua volta si ha quindi un incremento della coppia, tale da vincere la $\boldsymbol{C_l}$ del carico.

$$C = Kt * I$$

$$C = Kt * \frac{V - E}{R}$$

Per ogni motore esiste un diagramma coppia-velocità che mostra una retta che collega il punto di funzionamento a vuoto (a $\boldsymbol{\omega_0}$) e il punto a coppia massima ($\boldsymbol{C_s}$). Questa caratteristica esterna varia aumentando la tensione.

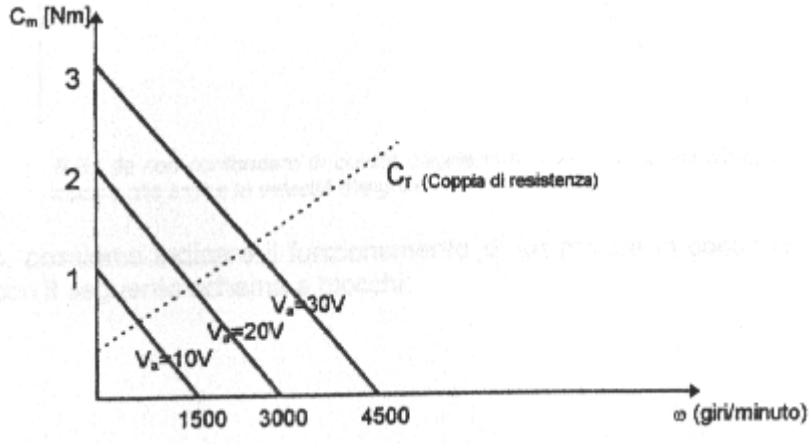
D. Potenza

Il motore riceve una potenza elettrica $\boldsymbol{P_t}$ che a sua volta viene suddivisa in due parti:

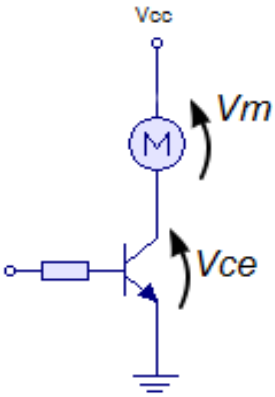
$$Pt = RI^2 + P_l$$

- Una parte che viene dissipata sulla resistenza dell'indotto del motore
- Una parte che viene fornita all'albero come potenza meccanica. Questa parte verrà a sua volta ridotta a causa dell'attrito dei cuscinetti o per la ventilazione interna.

Pertanto su ogni grafico coppia – velocità viene indicata l'area operativa di sicurezza per un funzionamento continuo del motore, all'interno della quale la potenza dissipata non raggiunge valori pericolosi da compromettere l'integrità della macchina.



4.3 AZIONAMENTO MOTORE C.C.



L'azionamento elettrico è un particolare sistema che converte l'energia elettrica che riceve in ingresso, in energia meccanica in uscita. Tale conversione avviene in genere mediante l'uso di elettronica di potenza attraverso la quale, eseguendo una particolare funzione, è possibile impartire dei comandi appositi al motore elettrico volti al compimento di una desiderata funzione. Un modo più semplice di controllare questo tipo di motore è tramite l'utilizzo della modalità ON-OFF, che permette di comandare il motore solo alla massima velocità di rotazione in un unico senso, oppure di fermarlo. In genere questo tipo di controllo viene gestito da un transistor o da un MOS che si trovano rispettivamente in condizioni di saturazione o di interdizione.

- Nel primo caso il transistor conduce e la tensione ai suoi capi è prossima a $0V$ $(0{,}6V)$, mentre quella ai capi del motore è circa pari alla V_{cc}. Il motore ruota quindi alla velocità massima possibile con la data tensione.

$$\mathrm{Vcc} = \mathrm{Vce} + \mathrm{Vm}$$
$$\mathrm{Vcc} = 0{,}6 + \mathrm{Vm}$$
$$Vcc - 0{,}6 = Vm$$

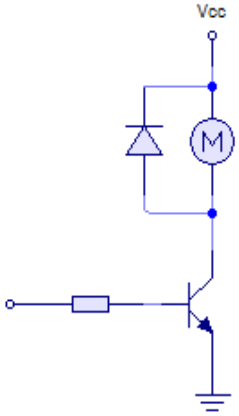
$$Vcc \cong Vm$$

- Nel secondo caso il transistor è "aperto" (quindi in condizione di interdizione) e la corrente passante dal collettore all'emettitore, dal drain al source nel caso dei mos, è pressoché nulla e quindi tale anche la potenza dissipata.

$$P = V * I = V * 0 = 0W$$

Il limite di questo tipo di pilotaggio è causato innanzitutto dall'impossibilità di poter regolare la tensione del motore e quindi la sua velocità. In seconda battuta ci si trova di fronte al problema che nel motore vi è una parte induttiva che tende a mantenere costante la corrente che scorre all'interno.

Quando il motore viene alimentato saturando il transistor, la corrente del motore raggiungere un valore a regime solo dopo un certo tempo secondo una curva esponenziale.



Il problema sorge nel momento in cui l'interruttore si apre dopo che il motore è ormai a regime.
Conoscendo la legge di conservazione dell'energia, si può affermare che quella immagazzinata dal motore, non può subire variazioni repentine in istanti di tempo molto piccoli.
Questo perché, secondo la seguente legge:

$$Potenza = \frac{energia}{tempo}$$

si può osservare che se si dovessero presentare delle grandi variazioni di energia in modo quasi istantaneo, la potenza tenderebbe ad assumere un valore infinito.
L'energia che viene accumulata sull'induttanza è direttamente proporzionale alla corrente

$$E = \frac{1}{2} L * I^2$$

quindi anch'essa non può subire variazioni repentine. Pertanto per la legge della differenza di potenziale ai capi di una induttanza, si può stabilire la seguente definizione:

$$v(t) = L\,\frac{di(t)}{dt}$$

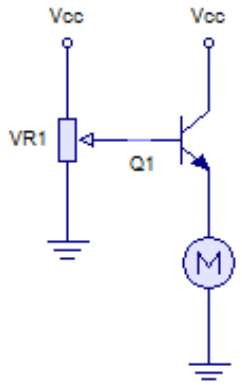
- Per variazioni repentine della corrente, la tensione tende ad aumentare a causa della componente induttiva. Infatti tende a mantenere costante il flusso di corrente che l'attraversava in precedenza sfruttando la differenza di potenziale presente ai suoi capi.
  Il diodo di ricircolo viene quindi posto in parallelo, e serve a scaricare l'induttanza nel caso in cui la tensione sul collettore del transistor superi di $0.6V$ il valore di V_{cc}. All'interno del motore si verrebbe infatti a creare una sovratensione molto elevata ma, grazie al sistema di sicurezza, appena si presenta una differenza di potenziale positiva (maggiore sul collettore del transistor) il diodo entra in conduzione proteggendo l'interruttore e scaricando la corrente residua dell'induttanza.

4.4 CONTROLLO MOTORE

Per poter essere in grado di modificare la velocità del motore, è necessario essere in grado di variare a piacere la tensione che si fornisce ai suoi capi. A tal proposito è possibile sfruttare due tipi di configurazione:

- Controllo lineare
- Controllo in PWM

Nel primo caso, si utilizza in genere un collegamento di tipo inseguitore di emettitore.



La tensione di ingresso viene fatta variare tra i valori di massa e V_{cc} , ad esempio tramite l'impiego di un potenziometro. La tensione di ingresso regola quindi quella sul carico, mentre il transistor provvede a fornire una corrente maggiore per poterlo alimentare.

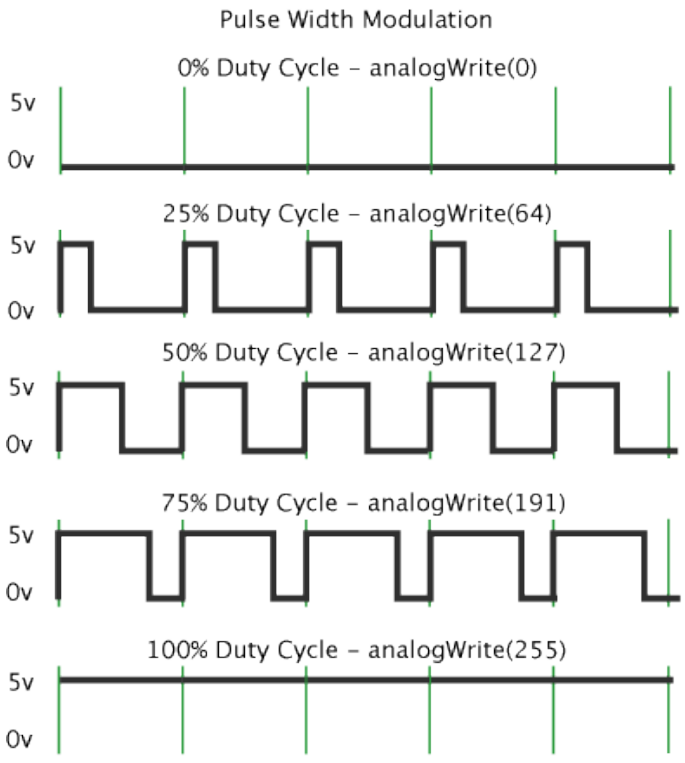
Questo sistema però pecca nel rendimento: una parte della potenza è quella sfruttata nel funzionamento effettivo del motore; l'altra viene dissipata sul transistor, questo a causa del suo funzionamento lineare. Il transistore è chiamato quindi a dissipare una notevole potenza, specie alle basse velocità dove la potenza da dissipare è addirittura maggiore rispetto a quella richiesta.

Una soluzione a questo problema è quello di variare il duty-cycle di un'onda quadra piuttosto che il valore effettivo della tensione. Questo è possibile attraverso un segnale di tipo PWM.

4.5 PWM (PULSE WIDTH MODULATION)

La modulazione di larghezza di impulso è un tipo di modulazione digitale che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso alto e di quello basso (duty-cycle).

Il duty cycle è il rapporto tra il tempo in cui l'onda assume valore alto e il tempo totale detto periodo ($T = {}^{1}/_{f}$).



Questo tipo di segnale viene spesso impiegato nella regolazione della velocità di motori a corrente continua presenti ad esempio nelle

ventole dei computer, oppure nei circuiti logici come microcontrollori, dove con l'ausilio di un semplice filtro RC si converte il segnale digitale in una corrispondente tensione continua.
Il vantaggio di questa tecnica è di ridurre drasticamente la potenza dissipata dal driver rispetto all'impiego di transistor controllati analogicamente.
In un semiconduttore la potenza dissipata è determinata dalla corrente che lo attraversa per la differenza di potenziale presente ai suoi capi.

$$P = V * I$$

In un circuito PWM il transistor in un istante conduce completamente, riducendo al minimo la caduta ai suoi capi, nell'altro stato non conduce, annullando quindi la corrente richiesta. In entrambi i casi la potenza dissipata è minima. Considerando un circuito ideale è possibile ritenerla nulla, ma nella realtà la velocità di commutazione dei dispositivi non è infinita ed istantanea, pertanto viene prodotta una potenza nel transitorio. Transistor in saturazione -> cortocircuito ($V = 0V$)

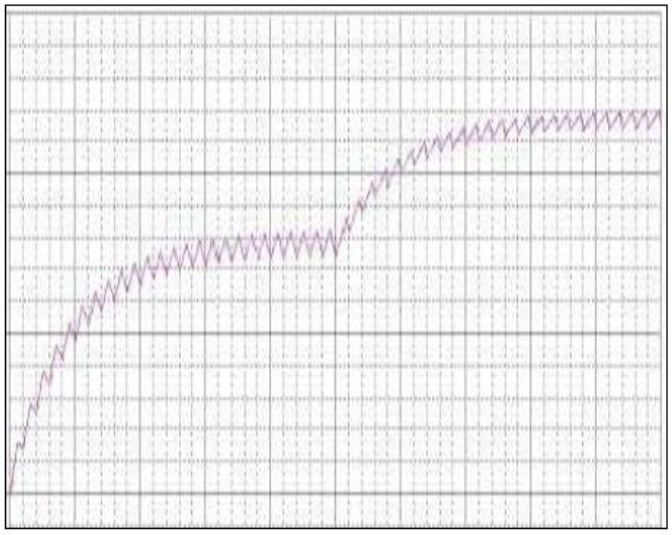
$$P = 0 * I = 0W$$

Transistor in interdizione -> circuito aperto ($I = 0A$)

$$P = V * 0 = 0W$$

Il PWM è un segnale periodico e dotato quindi di una frequenza non sempre tollerata dai vari componenti. Ad esempio, a causa del comportamento induttivo del transistor, al fine di ottenere una corrente stabile, media e proporzionale al duty cycle del segnale, bisogna avere una frequenza del segnale che sia di qualche kHz.
Se in altri casi la frequenza del segnale ingresso è bassa, il motore reagisce con un movimento non continuo ed altalenante. Se si dovesse collegarlo ad una trasmissione, la macchina si muoverebbe a scatti, quindi in genere si sceglie una frequenza maggiore ma allo stesso tempo fuori dallo spettro di frequenze percepibile dall'udito umano (20Hz-20KHz). Questo effetto è causato dalle componenti induttive nel motore (non a caso le casse acustiche sono formate da avvolgimenti). A frequenze troppo elevate (superiori a decine di kHz) si inizia ad avere una perdita nei circuiti magnetici che è proporzionale alla frequenza.

In figura viene mostrato l'andamento della corrente in un motore *C.C.*.

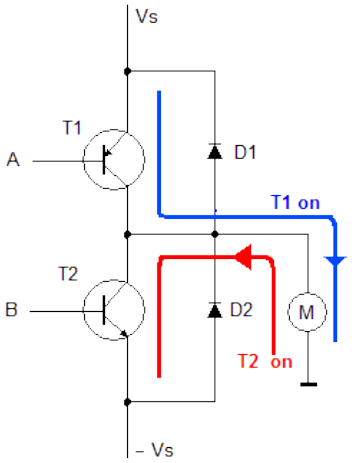


4.6 PONTE

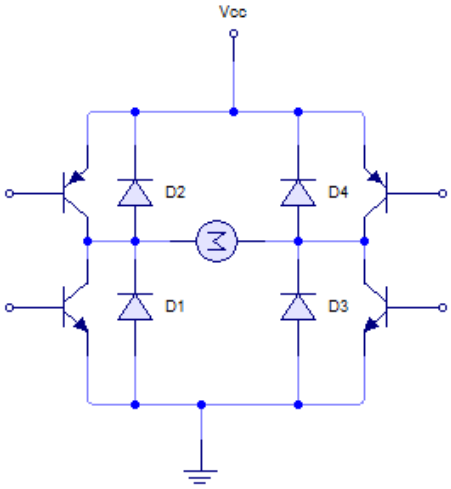
L'inseguitore di emettitore purtroppo non permette al motore di ruotare in entrambi i versi possibili. Questo perché è necessario invertire la direzione della corrente che passa all' interno del motore, di conseguenza anche la tensione applicata ai capi.
Per far ciò si possono utilizzare due tipi di circuito diversi:

- Ponte a T
- Ponte a H

Il circuito a semiponte sfrutta una tensione negativa e una positiva, che si applicano a due transistor. Questi due componenti vengono regolati per fare in modo che solo uno alla volta sia attivo, per evitare dei corto circuiti. Come è possibile vedere dall'immagine, in base al transistor che si trova in saturazione, la corrente scorrerà in determinato senso garantendo la possibilità di utilizzare entrambe le rotazioni possibili del motore.
Con T1 saturo e T2 interdetto, la corrente scorre attraverso T1 con direzione da +Vs a massa, mentre con T2 saturo e T1 interdetto, la corrente scorre da massa



verso –Vs. Lo scorrimento della corrente nella spira induce alla formazione di forze con verso opposto rispetto a quelle precedenti e provoca quindi la rotazione del rotore. I diodi D1 e D2 servono per proteggere i transistori dalle sovratensioni create dal motore durante le commutazioni.

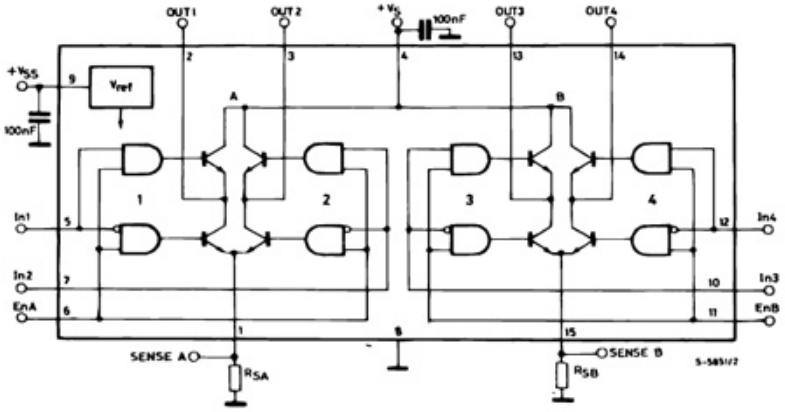


Questa tipologia di driver non è utilizzabile in questo progetto perché, essendo alimentato da una batteria, non è possibile generare una tensione negativa –Vs.
Per ovviare a questo problema si utilizza il ponte ad H, costituito da 4 interruttori comandati (buffer o transistor) e da 4 diodi di ricircolo.
A seconda di quali transistor sono attivi, la corrente nel motore potrà affluire da un punto ad un altro utilizzando una singola alimentazione. Se T1 e T4 si attivano e T2 e T3 rimangono interdetti, la corrente scorrerà in un verso, mentre nel verso opposto quando T1 e T4 vengono interdetti e T2 e T3 si saturano.
Se si attivasse solo una colonna, ad esempio T1 o T2, si creerebbe un cortocircuito tra massa e V_{cc} , ovvero una condizione che potrebbe compromettere il funzionamento del circuito danneggiandolo permanentemente.
Se tutti i transistor sono spenti non si creano maglie in cui la corrente possa passare, ma nel caso in cui l'induttanza del motore sia ancora carica troverebbe comunque un percorso di scarica tramite i diodi. Terminata la scarica dell'induttore, non si ha più il passaggio di corrente e se il motore era precedentemente in moto si arresta lentamente a causa degli attriti meccanici.
Se è attivo uno dei transistor superiori (T1 o T3), la corrente fornita dall'alimentazione non si forma, ma a differenza della situazione precedente, si viene a creare un cortocircuito ai capi del motore:
la tensione ai capi del motore è pari alla tensione diretta del diodo sommata a quella di conduzione del transistor. L'effetto risulta quindi una azione frenante causata dalla presenza del generatore equivalente e della conseguente corrente prodotta dal motore. Anche in questo circuito si inseriscono diodi di ricircolo necessari a proteggere i transistor dalle sovratensioni creare dalla parte induttiva del motore.

4.7 COMPONENTI UTILIZZATI

Per gestire il motore nel progetto si è scelto di utilizzare l'integrato L298. L298 è un integrato costituito da due ponti ad H ed è stato scelto perché, a differenza de L293, garantisce una corrente maggiore al motore (2A continui e 2.5A di picco).

La parte logica è gestita da porte logiche AND, nelle quali uno dei due ingressi è comune a tutte.



Questo collegamento, chiamato Enable (pin 6-11), permette di gestire in modo efficace il funzionamento del circuito, consentendone lo spegnimento o l'accensione di tutto il circuito. Gli altri ingressi delle porte logiche vengono collegati a due a due, ma con una negazione su una delle due al fine di non permettere l'abilitazione dei due transistor dello stesso lato e quindi rischiare di rovinare l'integrato. Questo collegamento non permette purtroppo di poter attivare un singolo transistor e quindi far frenare il motore immediatamente.
Il circuito di potenza è costituito da transistori che formano un ponte ad H. Il collegamento del ponte che dovrebbe andare a massa viene fornito tramite un pin (1-15) che permette il collegamento di una resistenza in serie che limita la corrente passante o che permette la misura della corrente assorbita tramite la caduta di tensione su di essa. Un altro aspetto importante di questo integrato è la distinzione delle alimentazioni. Quella fornita sul pin 9 gestisce la parte logica, dovendo necessariamente fornire un'alimentazione di 5v; l'altra alimentazione gestisce la tensione del motore, garantendo la possibilità di utilizzare tensioni molto più

elevate. Questa soluzione permette di utilizzare la giusta tensione per ogni tipo di motore, per un limite massimo di 40v.

Nel progetto il ponte H viene collegato direttamente ad Arduino.
I pin di attivazione dell'integrato e quelli che danno direzionalità al motore vengono collegati direttamente ai pin di uscita del microcontrollore.
Per muovere il motore viene fornito all'integrato il livello logico alto all'enable e 1 o 0 ai pin di input, in base alla direzione desiderata. Viene scritto anche lo 0 per evitare di abilitare involontariamente entrambe le due uscite rischiando di cortocircuitare le uscite dell'integrato compromettendolo.
Per poter regolare la velocità si utilizza il PWM e lo si applica, o nel pin che da direzione al motore, oppure in quello di enable. Nel progetto viene scelta la seconda opzione per facilitare la programmazione.

4.8 PARTE MECCANICA

Per quanto concerne la parte meccanica, bisogna effettuare dovute valutazioni sulla potenza ($P = \omega * C$) da fornire alle ruote. Questo per avere un'idea sul tipo di motore da utilizzare e sulle soluzioni meccaniche (motoriduzioni) da applicare per permettere al motore di poter lavorare a metà tra il funzionamento a vuoto e lo spunto.
Innanzitutto il mezzo è mosso da 4 ruote motrici collegate da tre differenziali, di cui ognuno è composto da una coppia conica che introduce un rapporto di trasmissione.
Tale parametro è definito come il coefficiente di moltiplicazione, o riduzione della velocità di rotazione, in una coppia di ruote dentate che trasmettono il movimento, ed è dato da un rapporto:

$$\frac{z2}{z1} = \frac{r2}{r1} = \frac{\omega 2}{\omega 1}$$

Le variabili presenti in queste equazioni sono rispettivamente:

- Il numero di denti della ruota motrice $z2$ ed il numero di denti della ruota condotta $z1$.
- Il raggio della ruota motrice $r2$ e quello della ruota condotta $r1$.
- La velocità angolare della ruota motrice $\omega 2$ e quella condotta $\omega 1$.

Questo rapporto, a seconda dei parametri in gioco, si definisce come:

- **Riducente**: nel caso in cui il rapporto sia minore di 1, dove la ruota condotta è più lenta ma può generare una coppia di forze maggiore.
- **Imparziale**, nel caso in cui il rapporto sia uguale a 1, dove la ruota condotta gira alla stessa velocità della ruota conduttrice e con ugual coppia
- **Moltiplicante**, nel caso in cui il rapporto sia maggiore di 1, dove la ruota condotta gira più velocemente della ruota conduttrice, ma con meno coppia.

Il rapporto da noi scelto è quello dato dal numero di denti, di seguito un esempio più concreto.
La coppia conica nei differenziali dei ponti del mezzo è 13/44 dove 13 sono i denti della ruota dentata motrice, a cui viene applicata una potenza di ingresso, mentre quella da 44 trasmette il moto alle ruote. Tale rapporto, essendo minore di 1, ricade nel primo caso, ovvero riduce la velocità entrante innalzando la coppia mantenendo la stessa potenza.
Oltre a questo fattore, se ne aggiunge un altro in presenza del differenziale centrale, e introducendo quindi un'altra riduzione. Questa normalmente serve per accoppiare i motori a scoppio che sviluppano una potenza che si basa prevalentemente sul numero di giri.
Utilizzando un motore elettrico si è dovuto utilizzare un motoriduttore per incrementare la coppia, mentre all'uscita è stato inserito un ingranaggio più grande che, interfacciato al differenziale centrale, introduce una riduzione circa pari ad 1. Tutto questo permette di convertire la sua velocità in coppia elevata per garantire lo spostamento della macchina, a discapito della velocità alle ruote che diminuisce notevolmente. Per poter dare una "spinta" in più viene aumentata la potenza elettrica fornita al motore. Sapendo che i motori a corrente continua per aumentare le loro performance coppia-velocità necessitano di una tensione maggiore, si è scelto di mettere in serie tre batterie da 9v per un totale di 27v.

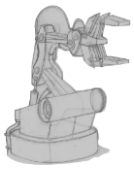
5 BRACCIO

Oltre alla macchina, Arduino controlla anche un braccio meccanico montato sopra di essa, in grado di compiere 6 movimenti differenti. Questo braccio è composto da lamine di compensato montate su sette differenti servomotori, in grado di compiere movimenti che coprono un'area di 360°.

Il Braccio meccanico è alimentato da una tensione fissa di $5V$ e da una corrente dipendente dal carico applicato ai servomotori, dovuto soprattutto ai movimenti e per mantenere la posizione raggiunta. Questa corrente è stata calcolata e al picco di richiesta non supera $2{,}5A$. Per ottenere tali valori è stato opportuno caricare una batteria a 6 celle da $7{,}5V$ e $4Ah$ collegata ad un regolatore switching di tipo *step-down buck*.

5.1 STEP-DOWN (BUCK)

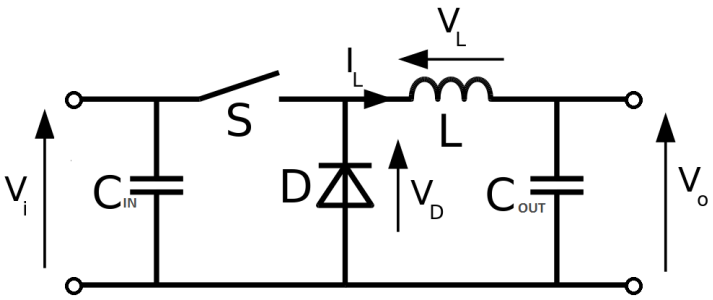
Abbiamo scelto un regolatore di tipo switching invece di un limitatore della serie *78xx* perché offre una maggiore stabilità e riesce a gestire picchi di corrente molto superiori. Il principio fondamentale su cui si basa il funzionamento di un alimentatore switching è detto *PWM*, dall'Inglese *"Pulse Width Modulation"*, e cioè modulazione della larghezza dell'impulso. Molto brevemente, la tensione di alimentazione arriva nella forma di una serie di impulsi dati dall'aprirsi e chiudersi di un interruttore *switch*, a frequenza costante, distanziati uno dall'altro da un tempo T. Chiameremo T_{ON} il tempo in cui l'impulso è alto (quindi in tensione) e T_{OFF} il tempo in cui l'impulso è zero (senza tensione). Poiché gli impulsi sono a frequenza costante, anche l'intervallo di tempo *T* ha valore costante: la modulazione PWM consiste nel far variare il tempo T_{ON}; naturalmente, quando T_{ON} si allunga, T_{OFF} diventa necessariamente più breve. Per ottenere l'effetto switching abbiamo sostituito l'interruttore con un integrato a transistor *LM2576*. Filtrando gli impulsi con una rete *LC*, si ottiene una tensione di uscita V_{OUT} il cui valore dipende dalla larghezza degli impulsi, ed è esattamente uguale al valore di picco moltiplicato per il duty cycle. Si comprende quindi come, modulando la larghezza dell'impulso, sia possibile ottenere qualsiasi tensione in uscita, e senza dissipare inutilmente parte della potenza. Naturalmente, affinché la tensione in uscita sia priva di disturbi,



occorrerà dimensionare opportunamente i componenti del filtro, scegliendo inoltre una frequenza di clock il più elevata possibile.

Il circuito di base è costituito da due interruttori (un transistor e un diodo), un induttore e un condensatore, più un condensatore posizionato a monte dell'ingresso per filtrare gli eventuali disturbi.

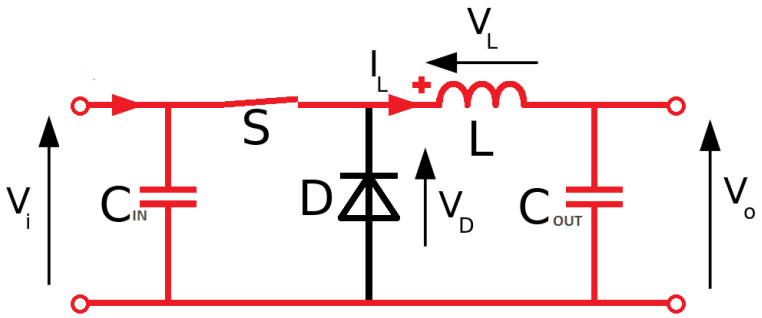
Questo è lo schema di base del circuito:

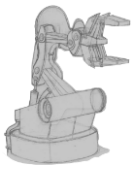


Lo studio di questo circuito avviene differenziando i due stati dello *switch*:

A. SWITCH CHIUSO

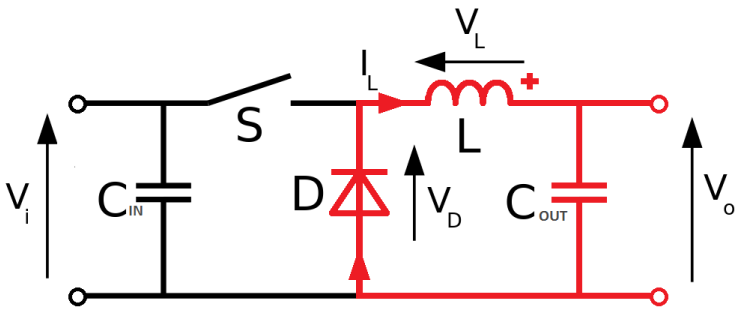
Supponiamo ora che sia il condensatore che l'induttanza siano inizialmente scarichi. Quando l'interruttore S viene chiuso, la tensione in ingresso polarizza inversamente il diodo D (il quale diventa a tutti gli effetti un circuito aperto) e la corrente tende invece ad aumentare. A causa dell'induttanza L si avrà un andamento esponenziale della corrente che caricherà il condensatore C_{out}. La tensione di uscita aumenta quindi secondo la legge di carica del condensatore fino a raggiungere il valore di soglia prefissato, aprendo l'interruttore S.





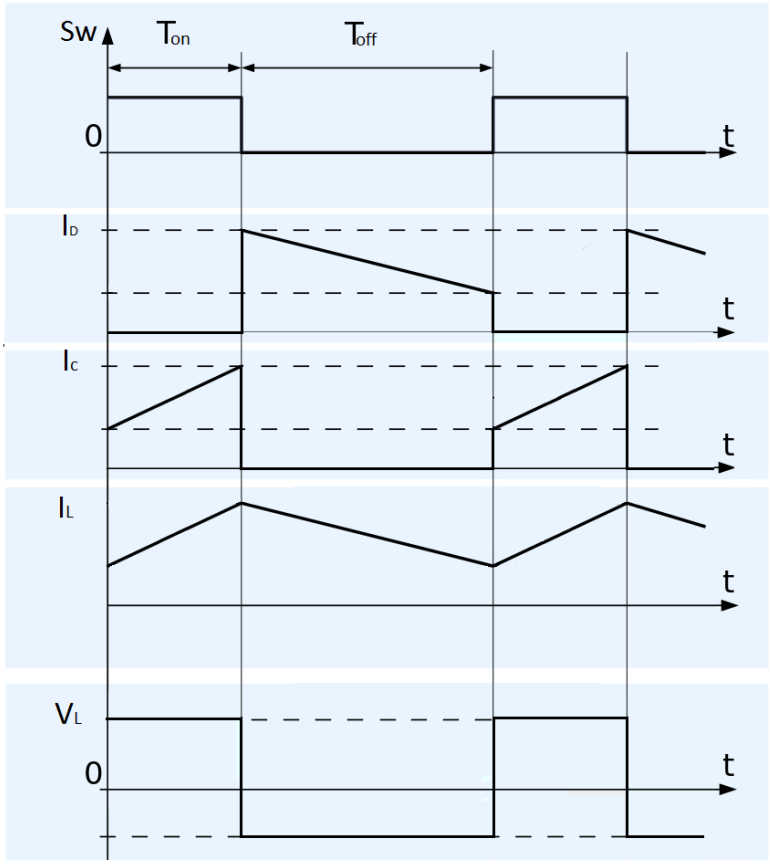
B. SWITCH APERTO

Aprendosi l'interruttore S blocca il flusso di corrente e l'induttanza inizia a scaricarsi generando una corrente passante per Il diodo D, ma continuando a caricare il condensatore C_{out} e ad alimentare il carico. Quando la corrente prodotta dall'induttanza diventa insufficiente ad alimentare il carico, sarà il condensatore che, avendo tensione maggiore, scarica ancora al carico il quale continua a richiedere corrente. Una volta raggiunto il valore minimo di soglia l'interruttore si chiude e ricomincia il ciclo.



Semplicemente guardando il funzionamento del circuito siamo in grado di definire come attraverso l'utilizzo combinato di induttanza e condensatore, sia possibile ottenere una tensione fissa in uscita. Lo switching è infatti un sistema nel quale grazie al continuo cambiamento dell'interruttore in ingresso, che fornisce la tensione iniziale e permette di fornire in uscita una combinazione di due tensioni. Con questo metodo si riesce a diminuire la potenza richiesta dal circuito essendo la tensione in parte fornita durante lo stato *off* dello *switch* (non viene richiesta tensione di ingresso).

Vediamo ora tramite grafici l'andamento delle tensioni e delle correnti nel tempo:

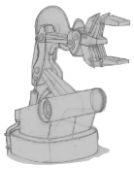


Come è possibile vedere sul grafico, si avranno diverse condizioni nello stato *ON* rispetto a quello *OFF* dello *switch*. Nello stato *ON* non c'è corrente nel diodo ma circola nel condensatore, la quale lo porta a caricarsi. Al contrario nello stato *OFF* non ci sarà più corrente sul condensatore, e il diodo vedrà invece la corrente che era stata immagazzinata nell'induttanza. Quest'ultima avrà sempre una corrente che circola in quanto viene attraversata in fase di *ON* e si scarica in fase di *OFF*. La tensione invece si inverte tra le due fasi, in quanto da utilizzatore in una fase diventa generatore nell'altra (si noti nei disegni il segno + ai capi).

5.2 CALCOLI MATEMATICI

La funzione di trasferimento è data dalla relazione:

$$\frac{V_i - V_o}{L} * T_{on} = \frac{V_o}{L} * T_{off}$$



Da cui ricaviamo V_o in funzione di V_i:

$$\frac{V_i - V_o}{L} * T_{on} = \frac{V_o}{L} * T_{off}$$

$$\frac{V_i * T_{on}}{L} - \frac{V_o * T_{on}}{L} = \frac{V_o * T_{off}}{L}$$

$$V_i = V_o * \frac{T_{on} + T_{off}}{T_{on}}$$

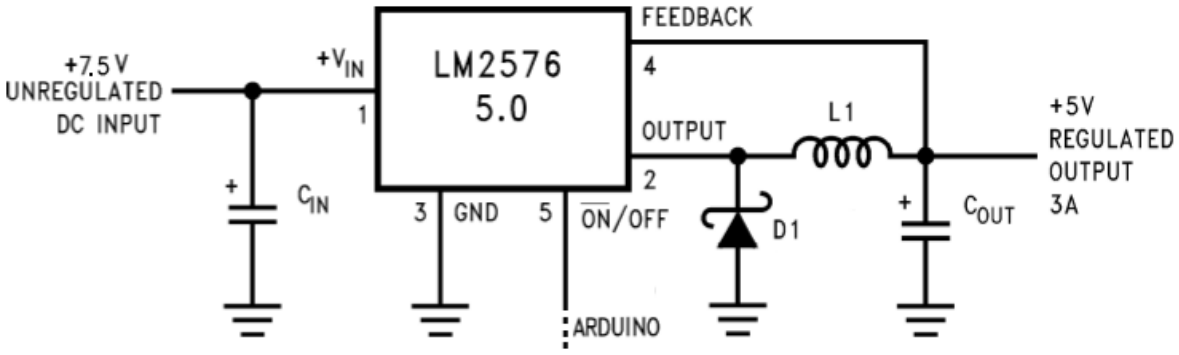
$$V_o = V_i * \frac{T_{on}}{T_{on} + T_{off}}$$

$$V_o = V_i * \delta$$

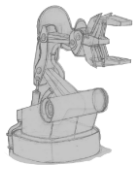
Osservando l'equazione ottenuta si può notare come la tensione d'uscita dipenda dal duty cycle (tempo in cui il circuito rimane alto su tempo totale) dato dal circuito.

5.3 CIRCUITO

Il circuito da noi montato è così composto:

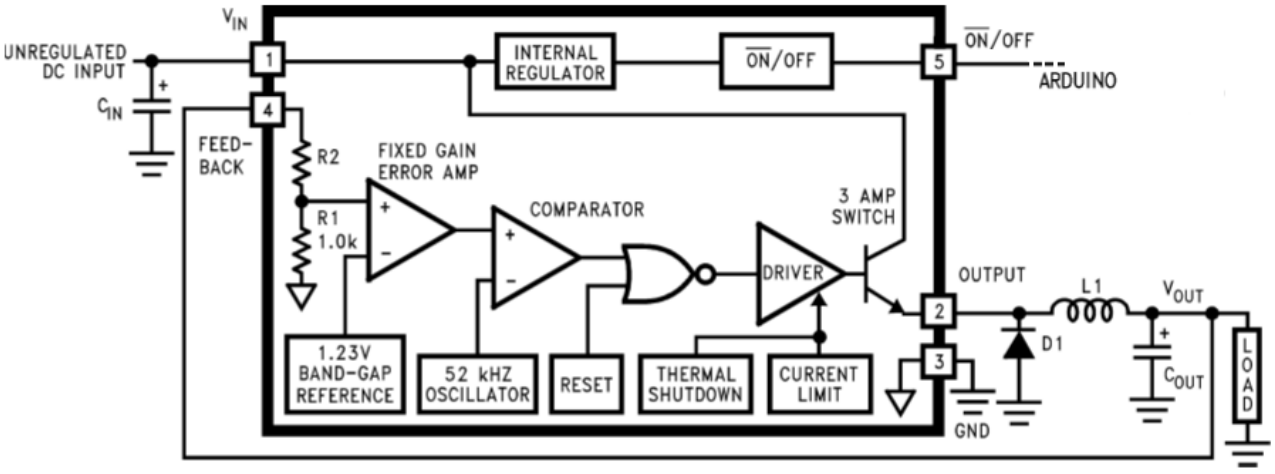


Ciò che è stato descritto sopra viene ora rappresentato tramite il circuito fisico che è stato poi stampato su PCB. Il componente che avrà la funzione di switch è *LM2576*, il quale con la corretta configurazione è in grado di fornire 5V in uscita. È stato scelto in base alle nostre esigenze, in quanto se il circuito non fosse stato in



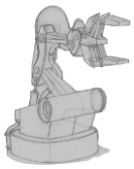
grado di fornire almeno 2.5A in uscita non sarebbe bastato all'assorbimento di corrente dei servomotori del braccio.

Internamente l'integrato è composto come mostrato di seguito:

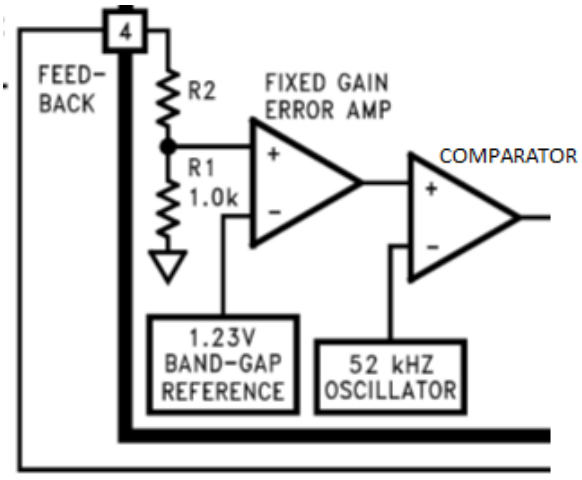


Per spiegarne il funzionamento risulta più semplice dividerlo in blocchi e spiegarne le singole funzioni.

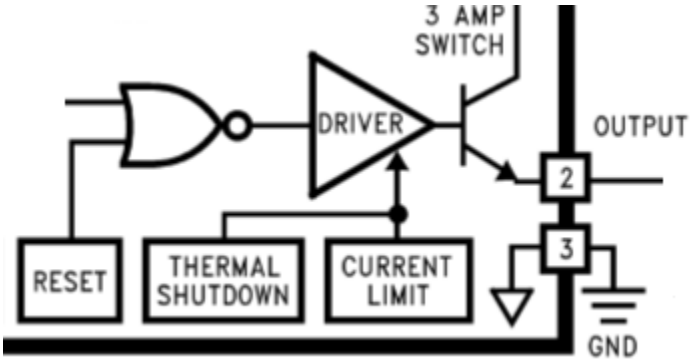
- Il blocco posto nella parte superiore svolge una funzione tanto semplice quanto essenziale: si tratta infatti dell'*enable* del dispositivo. Il *pin* corrispondente (con comando che arriva direttamente da Arduino) è collegato ad un blocco che permette di disattivare il dispositivo, e quindi di spegnerlo completamente quando viene dato un ingresso alto (come si nota dal segno posto sopra la scritta $\overline{ON}$, l'enable è attivo basso, cioè viene considerato acceso quando è basso il valore del pin). Chiamiamo questo pin spegnimento d'emergenza, in quanto il regolatore fornisce la tensione di alimentazione anche ad Arduino e il suo spegnimento comporterebbe lo spegnimento anche del controllore. Questa regolazione sarebbe stata utile nel caso in cui Arduino avesse avuto un'alimentazione differente, ma purtroppo necessita dei 5V per poter funzionare in configurazione *stand-alone*.

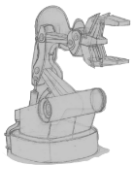


- Il secondo blocco principale è un anello in retroazione negativa che indica il valore di un eventuale errore da aggiustare (cioè di quanto eccede il segnale dal segnale di set-point). È composto da un sommatore invertente che effettua proprio questo compito prendendo un valore di riferimento di $1.23V$ e da un operazionale utilizzato come comparatore tra l'errore e un dente di sega a $52KH_z$ prodotto da un oscillatore interno.



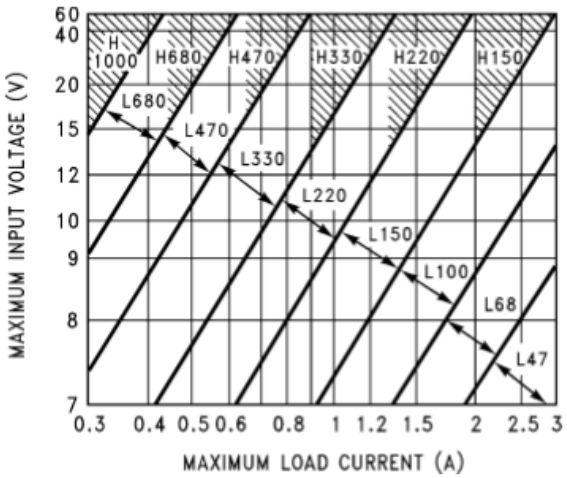
- L'ultimo blocco è il circuito d'uscita. È composto da una porta logica NOR, che fornisce il comando di reset del ciclo, un transistor utilizzato come switch (quindi detto in configurazione *ON-OFF*) e dal driver per il controllo associato. Il driver oltre che ad un modo efficiente per il controllo del transistor è anche utilizzato come protezione del transistor. In un circuito del genere infatti è possibile avere forti sbalzi di corrente (e di conseguenza rapidi cambiamenti di temperatura). Per evitare tali problematiche al driver sono assegnati dei valori standard per quanto riguarda la corrente massima e la temperatura. In questo modo si riesce a fare spegnere il transistor in caso di sovraccarichi di corrente richiesta o quando la temperatura dell'integrato (non adeguatamente dissipato) supera i limiti stabiliti.





Il passo successivo per la realizzazione del circuito è stato il dimensionamento dei componenti esterni tra cui i condensatori, l'induttanza e il diodo. Vista la differenza dei vari integrati, con la possibilità di avere svariati valori di corrente di uscita, il produttore ha reso disponibile sul datasheet delle tabelle utili all'identificazione dei componenti più adatti.

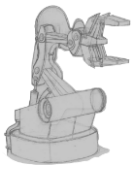
L'induttanza è stata scelta secondo le linee guida della seguente tabella:



Considerando la tensione in ingresso di circa $7.5V$ (compresa nella gamma quindi che sta tra 7 e 8V) e la corrente massima a carico di circa $2.5A$, la scelta è stata la L47. Corrispondente a questo valore è stata un'induttanza da $47\mu H$.

Nel caso sia necessario sapere la corrente di picco che circola nell'induttanza si usa la formula

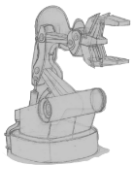
$$I_{p(max)} = I_{load(\max)} + \frac{(V_{in} - V_{out})t_{on}}{2L}$$



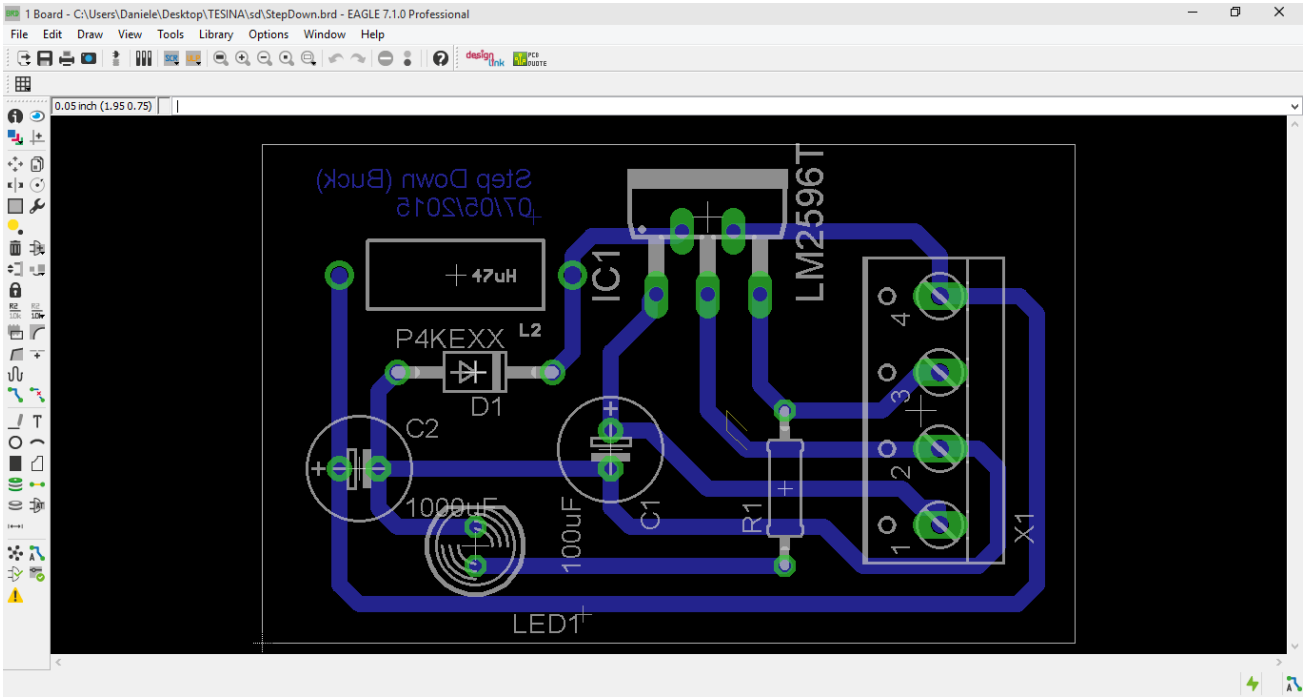
V_R	Schottky		Fast Recovery	
	3A	4A–6A	3A	4A–6A
20V	1N5820 MBR320P SR302	1N5823	The following diodes are all rated to 100V 31DF1 HER302	The following diodes are all rated to 100V 50WF10 MUR410 HER602
30V	1N5821 MBR330 31DQ03 SR303	50WQ03 1N5824		
40V	1N5822 MBR340 31DQ04 SR304	MBR340 50WQ04 1N5825		
50V	MBR350 31DQ05 SR305	50WQ05		
60V	MBR360 DQ06 SR306	50WR06 50SQ060		

Questa è invece la tabella delle possibili scelte per i diodi e dalle specifiche da noi adottate si nota che rientriamo nella fascia dei 1N5823. Questo perché considerando una tensione di ingresso pari a 7,5V e una corrente massima di 4A rientriamo abbondantemente in questa categoria anche se, a causa della mancanza del componente, abbiamo dovuto optare per un altro modello di diodo Schottky, l'MR854, dotato ovviamente di caratteristiche uguali.
La scelta del diodo deve essere fatta in modo preciso in quanto solitamente sbagliarlo causa perdite nei circuiti switching. La scelta ricade sui rettificatori *Schottky*, i quali sono i più adatti sia in termini di velocità che di tensione. Inoltre la corrente di picco che interessa il diodo è maggiore di quella vista dal carico del circuito, e per ragioni di stabilità e sicurezza è necessario un diodo che supporti 1.2 volte la corrente massima del carico.
Dal datasheet osserviamo che i diodi della serie MR85x supportano valori di corrente fino a 3A, hanno una bassa caduta di tensione di breakdown e supportano tensioni fino a 600V. Inoltre questi diodi hanno un'escursione termica da -65 a +150°C garantendo alta resistenza e durabilità.

Per la scelta dei condensatori bisogna innanzitutto distinguerne l'utilizzo nel circuito. Il condensatore in ingresso fornisce un filtro agli eventuali disturbi e per evitare dei picchi in ingresso che causerebbero problemi al circuito. Essendo di filtro tra l'ingresso e massa, deve essere in grado di supportare la tensione di alimentazione del circuito, mentre la capacità è stata scelta di $100\mu F$ elettrolitico. Il condensatore di output serve invece ad effettuare le operazioni di regolazione come spiegato prima. Nel datasheet dell'integrato viene spiegato come i poli dominanti siano regolati in base al valore dell'induttore e al valore del condensatore. Infine viene stabilito che per ottenere dei buoni valori di ripple (intorno all'1%) si deve scegliere un condensatore di valore tra $680\mu F$ e $1000\mu F$. La scelta è ricaduta su un condensatore da $1000\mu F$.



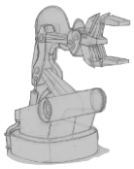
Infine il circuito da noi realizzato tramite il programma *eagle* mostra il PCB successivamente realizzato:



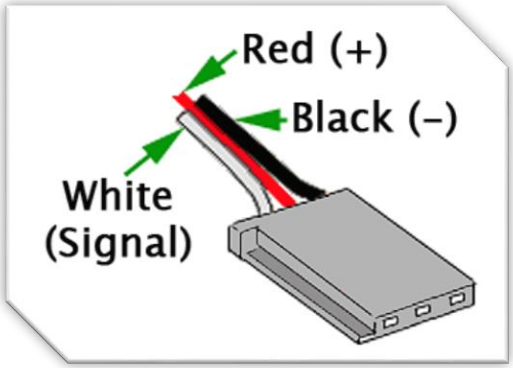
L'integrato utilizzato sul programma è un *LM2596*, in quanto era l'unico ad avere le stesse dimensioni dell'integrato effettivamente utilizzato.

5.4 SERVOMOTORI

Il servomotore, o semplicemente servo, è un sistema di controllo a retroazione in cui la grandezza controllata è la posizione angolare di un asse. Nella robotica i servomotori si presentano come piccoli contenitori di materiale plastico da cui fuoriesce un perno in grado di ruotare di un angolo compreso tra 0 e 180° mantenendo stabilmente la posizione raggiunta. Per ottenere la rotazione del perno è utilizzato un motore a corrente continua e un meccanismo di demoltiplica che 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. Questo circuito riesce quindi tramite un potenziometro resistivo a bloccare il motore sul punto desiderato.

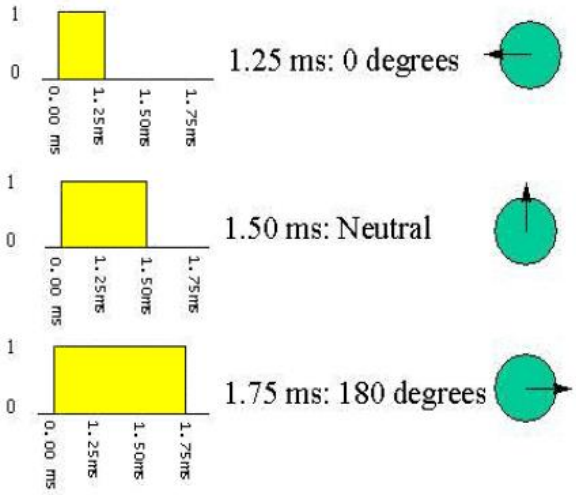


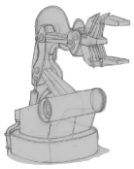
I servomotori sono progettati per essere pilotati nel modo più semplice possibile, eseguendo la movimentazione senza l'ausilio di circuiterie troppo complesse o l'uso di sistemi a microprocessore. Un servomotore dispone solitamente di soli tre fili: due di questi sono riservati all'alimentazione in corrente continua. Il positivo è di colore rosso, il negativo di colore nero, il terzo filo, normalmente di colore bianco, è riservato per il controllo del posizionamento. Il colore di questi fili può però variare a seconda della casa costruttrice.



Tramite il filo del controllo è necessario applicare un segnale impulsivo o PWM le cui caratteristiche sono "quasi" univoche per qualsiasi servomotore disponibile in commercio. Per essere sicuri di riuscire a pilotare qualsiasi servomotore il nostro circuito di pilotaggio dovrà essere in grado di trasmettere al servomotore circa 50 impulsi positivi al secondo di durata variabile, in un intervallo massimo compreso tra 0.25ms e 2.75ms.

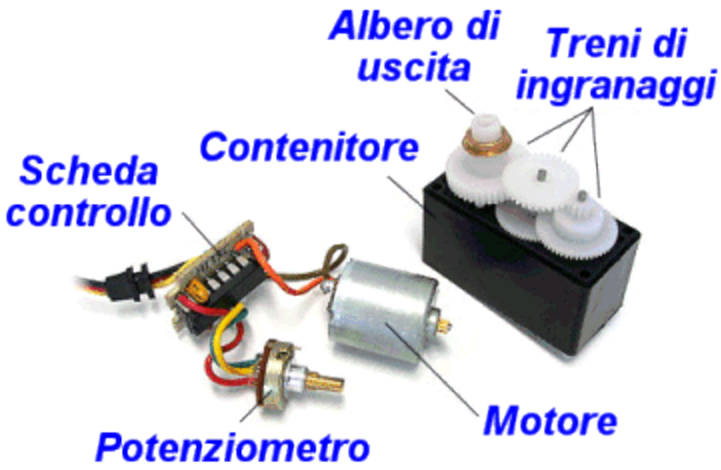
Generalmente con un impulso di durata pari a 1.5ms il perno del servomotore si pone esattamente al centro del suo intervallo di rotazione. Da questo punto, il perno può ruotare in senso antiorario se l'impulso fornito ha una durata inferiore a 1.5ms e in senso orario se l'impulso fornito ha durata superiore a 1.5ms. Il rapporto esatto tra la rotazione del perno e la larghezza dell'impulso fornito può variare tra i vari modelli di servomotore.





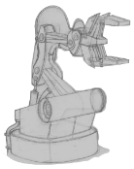
A. Composizione interna

La dimensione e il fissaggio di un servo standard non cambiano, all'interno dello stesso è presente un motore, una serie di ingranaggi che riducono la velocità del motore, un circuito di controllo e un potenziometro.

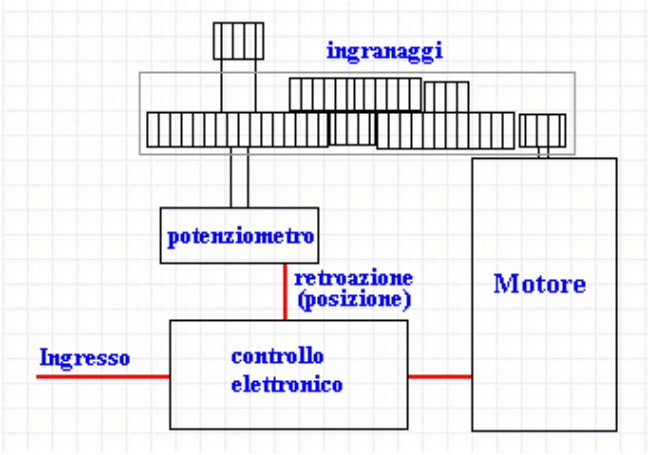


Il motore e il potenziometro sono collegati al circuito di controllo e l'insieme di questi tre elementi definisce un sistema di feedback ad anello chiuso. Il circuito e il motore vengono alimentati da una tensione continua stabilizzata, in genere di valore compreso tra 4,8 V e 6,0 V, anche se molti motori sono in grado di accettare input di alimentazione fino a 7,2 V.
Per far ruotare il motore bisogna inviare un segnale digitale al circuito di controllo. In questo modo esso si attiverà e, attraverso una serie di ingranaggi, varierà la posizione dell'albero del potenziometro indicando una misura della posizione dell'albero motore del servo. Quando il potenziometro raggiunge la posizione desiderata, il circuito di controllo spegne il motore.

I servomotori vengono progettati in genere per effettuare una rotazione parziale piuttosto di un moto rotatorio continuo in quanto l'impiego fondamentale di un servo consiste nel raggiungere una posizione accurata dell'albero del motore, con movimenti compresi nell'intervallo tra 0° e 180°. Vengono infatti bloccati internamente e superare i limiti meccanici provocherebbe lo sfregamento o la vibrazione degli ingranaggi. Se questi effetti proseguono per più di qualche secondo, gli ingranaggi del motore e io stesso potrebbero danneggiarsi in modo irreparabile. Anche se questo movimento non sembra considerevole, può risultare più che sufficiente per manovrare un robot, per sollevare e abbassare le gambe, per ruotare un sensore che deve esaminare ciò che le circonda o, come nel nostro caso,

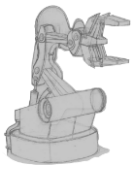


per muovere un braccio robotico. La rotazione precisa di un angolo da parte di un servo in risposta a determinati segnali digitali rappresenta una delle funzionalità più sfruttate in tutti i campi della robotica.



L'albero del motore di un servo R/C viene posizionato utilizzando una tecnica PWM. In un sistema di questo tipo, il servo risponde alla durata di un segnale definito all'interno di un treno di impulsi a frequenza fissa. In particolare, il circuito di controllo risponde a un segnale digitale i cui impulsi hanno una durata variabile da circa $1ms$ a circa $2ms$. Questi impulsi vengono trasmessi alla velocità di $50Khz$. La durata esatta di un impulso, espressa in frazioni di millisecondo, stabilisce la posizione del servo. Alcuni servo consentono di variare la frequenza del segnale PWM, altri invece non funzionano correttamente oppure "tremano" nel caso in cui gli impulsi vengano inviati a frequenze diverse. Per garantire il corretto funzionamento di un servo, bisogna verificare sempre che ci siano circa $20ms$ di pausa tra l'inizio di un impulso e quello successivo. Detto questo è quindi deducibile che alla durata di $1ms$ il servo viene comandato per ruotare completamente in una direzione, per esempio in senso antiorario mentre a $2ms$ il servo ruota completamente nella direzione opposta. Di conseguenza, un impulso di $1{,}5ms$ comanda il servo in modo da posizionarlo nella sua posizione centrale, o di riposo. Questa tecnica ha assunto negli anni parecchi nomi. Uno dei nomi più diffusi è probabilmente quello di segnale digitale proporzionale, in quanto il movimento del servo è proporzionale al segnale digitale con il quale viene attivato.

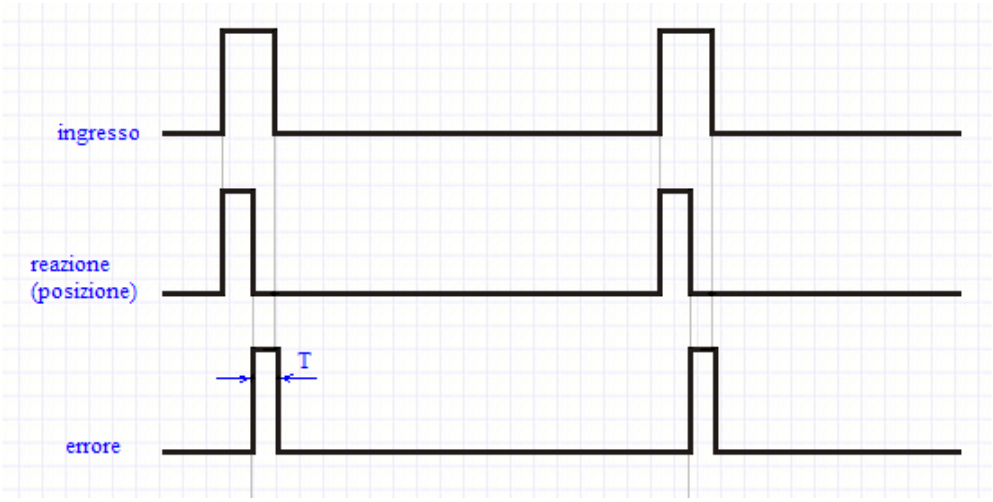
L'alimentazione fornita al motore all'interno del servo è anche proporzionale alla differenza tra la posizione attuale dell'albero e la posizione che deve raggiungere. Se il servo deve effettuare un movimento breve per raggiungere la nuova posizione, il motore viene guidato con una velocità di rotazione bassa. In questo modo si



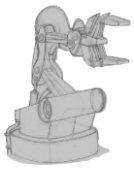
garantisce che il motore non superi la posizione desiderata. Al contrario, se il servo deve effettuare un movimento più accentuato per raggiungere la nuova posizione, il motore viene pilotato alla massima velocità consentita, in modo da arrivare appena possibile e ovviamente rallenta quando il servo si avvicina alla posizione finale. Questo processo in apparenza abbastanza complesso avviene in un breve intervallo di tempo.

POTENZIOMETRO

Il potenziometro del servo svolge un ruolo fondamentale che consente di stabilire l'istante in cui il motore ha impostato l'albero nella posizione desiderata. Questo potenziometro è fisicamente collegato all'albero di uscita del motore. In questo modo, la posizione del potenziometro coincide precisamente con quella dell'albero. Nei servomotori il potenziometro è configurato come un partitore di tensione e fornisce al circuito di controllo una tensione che variabile in funzione della variazione dell'uscita del servo. Il circuito di controllo del servo mette in relazione questa tensione con la temporizzazione degli impulsi digitali di ingresso e genera



un segnale di errore nel caso in cui debba correggere la tensione da inviare al motore. Questo segnale di errore è proporzionale alla differenza rilevata tra la posizione del potenziometro e la temporizzazione definita dal segnale in ingresso. Per compensare questa differenza, il circuito di controllo applica al motore un segnale che tiene conto di questo errore. Quando la tensione del potenziometro e la temporizzazione degli impulsi digitali coincidono, il segnale di errore viene annullato e il motore si ferma.



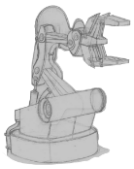
MOTORE

Il motore di un servo R/C ruota a una velocità di parecchi $^{giri}/_{min}$. Questa velocità è troppo elevata per essere impiegata direttamente nei modelli di aeroplani e automobili, per non parlare dei robot. Tutti i servo prevedono pertanto la presenza di ingranaggi che riducono l'uscita del motore a una velocità equivalente compresa tra $50\,^{giri}/_{min}$ e $100\,^{giri}/_{min}$.

I motori dei servo R/C rispettano una serie di caratteristiche standard, in particolare per quanto riguarda quelli di dimensioni standard, che hanno un ingombro di circa 40x20x35 mm. Altri tipi di servo possono avere dimensioni differenti, dato che vengono realizzati per applicazioni particolari. La tabella indica le specifiche tipiche di diversi servo, tra cui le dimensioni, il peso, la coppia motore e il tempo transitorio. Ovviamente, a parte le dimensioni standard, queste specifiche possono variare in funzione del modello e del produttore.

Tipo servo	Lunghezza (cm)	Larghezza (cm)	Altezza (cm)	Peso (gr)	Coppia (N*cm)	Velocità
Standard	4	2	3,6	48,5	36,3	0,19 s/60°
Scala 1/4	5,8	2,8	2,8	110	129,4	0,23 s/60°
Mini/micro	2,2	1	2	8	11,7	0,12 s/60°
Basso profilo	4	2	2	49	70	0,12 s/60°

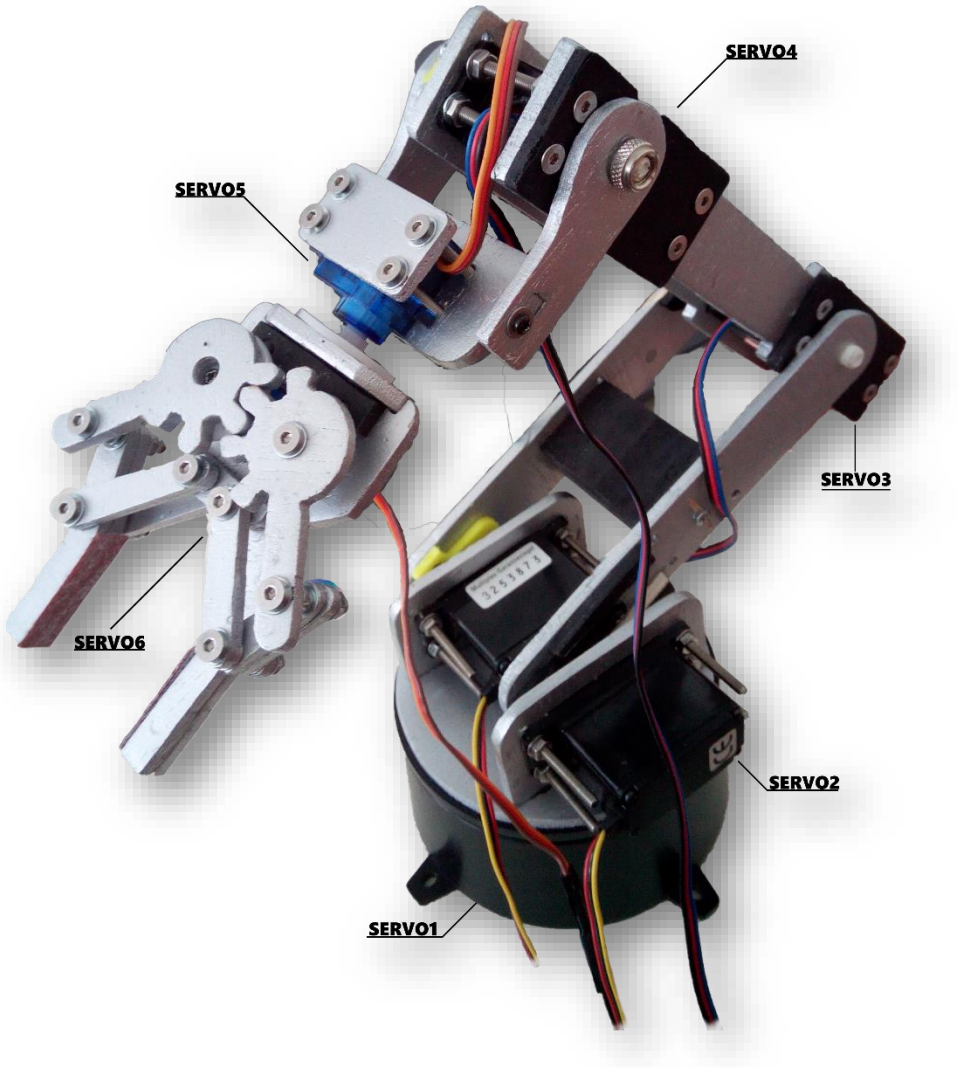
L'unità di misura standard dei servo R/C è espressa in once per pollice, ovvero dal numero di once che il servo è in grado di sollevare quando l'albero del suo motore viene spostato di 1 pollice. I servo evidenziano una considerevole coppia motore, soprattutto grazie agli ingranaggi di riduzione della velocità del motore.
Il transitorio (o slew rate) esprime una misura approssimativa del tempo richiesto dal servo per ruotare il motore di un determinato angolo, in genere di 60 °.Per calcolare la velocità equivalente espressa in $^{giri}/_{min}$ bisogna moltiplicare il transitorio relativo a un angolo di 60° per 6 (in modo da ottenere il transitorio di una rotazione a 360°), poi dividere il risultato per 60.
Per il nostro progetto abbiamo utilizzato sei servo standard, cinque utilizzati per i movimenti del braccio e il sesto collegato allo sterzo della macchinina, mentre per l'apertura e la rotazione della pinza del braccio abbiamo utilizzato due mini servo



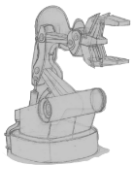
5.5 MODELLO FISICO

Per la realizzazione del braccio abbiamo utilizzato delle lamine di compensato sagomate manualmente nelle forme richieste dal progetto qui sotto riportate.

Assemblato e verniciato il braccio appare in questo modo:



Il braccio è composto da sette servo rinominati a partire dalla base, ognuno associato ad uno specifico pin di Arduino. Ogni servo ruota per 180°, quindi si può osservare che il braccio è in grado di coprire una superfice pari ad una semisfera di raggio verso l'alto, più una buona porzione di piano al di sotto dell'altezza della base.

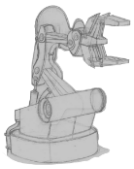


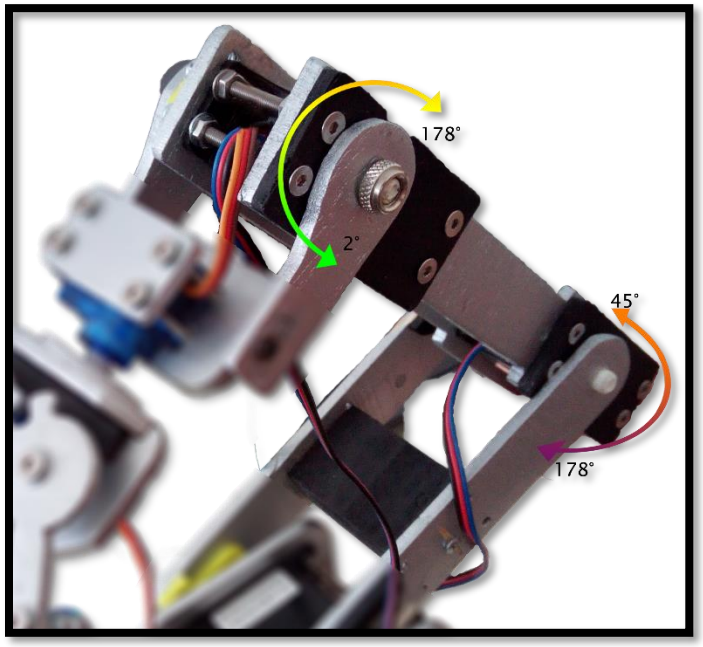
Il buck precedentemente visto serve per fornire la tensione ma soprattutto la corrente necessaria al funzionamento dei servo. Sono infatti alimentati da una tensione fissa di $5V$ e si ha una corrente massima richiesta di 2,5A. Come si può vedere dal listato del programma i servo vengono attivati uno alla volta partendo dalla base: questo è necessario in quanto l'attivazione simultanea di sette motori avrebbe portato una richiesta troppo elevata di corrente da gestire con Arduino. La scelta effettuata è stato introdurre una pausa di 100ms tra l'accensione di ogni servo, la quale garantisce l'attivazione in sicurezza rinunciando ad un tempo di accensione immediato.

Risolti i problemi di alimentazione siamo passati ad analizzare il comportamento di ogni servo, osservando la massima escursione che i vari assi potevano compiere. Da quest'analisi abbiamo ricavato l'escursione di ogni singolo servo, limitando in alcuni casi i movimenti per evitare sovraccarichi in grado di bruciare il sistema interno di controllo.

Nel dettaglio i servo con la massima escursione loro assegnata:

I tre servo più in basso compiono lo stesso movimento di 176°, rispettivamente la base orizzontalmente e i due di inclinazione verticalmente. Considerando che i due servo d'inclinazione compiono lo stesso movimento in modo sincronizzato li abbiamo considerati come un unico servo, sia nel listato sia nei collegamenti ad Arduino utilizzando un solo pin collegato ad entrambi. Il margine di 2° gradi su entrambe le direzioni dei movimenti è necessario a garantire la sicurezza dei servo, evitando di sforzarli.





Il servo del gomito e del polso hanno rispettivamente un'escursione da 45° a 178° e da 2° a 178°: il primo è limitato a 45° per evitare che i cavi che passano per il gomito si trancino o si ingroviglino nel movimento, il polso è invece limitato per gli stessi motivi dei servo della base precedentemente visti.

La pinza è invece formata da due mini servo, uno adibito alla rotazione e uno adibito all'apertura o alla chiusura della stessa. Il servo di rotazione è limitato da 15° a 173,5° perché aumentando l'escursione si incontravano problematiche meccaniche, mentre il servo della pinza è limitato ad un'escursione massima di 65°, dovuta al sistema di ingranaggi per il movimento e l'apertura della pinza.

6 ASSEMBLAGGIO

Come base per la struttura della macchina è stato scelto un telaio in ergal (derivato allumino) a cui sono collegate quattro ruote e le relative sospensioni. La particolarità dell'assetto fornito è la possibilità di usare un solo motore per muovere tutte e quattro le ruote, rendendo la macchina a tutti gli effetti un dispositivo a quattro ruote motrici. Prese le dimensioni della base è stato necessario trovare uno schema conservativo per posizionare le batterie, in quanto sono l'elemento più pesante. Sono state posizionate in due punti opposti in modo da bilanciare il tutto, inserendo il motore nello spazio rimasto tra le stesse.

6.1 MOTORE

Per quanto riguarda il motore è stato posizionato con l'albero rotante vicino al centro della struttura, in modo che fosse facilmente accessibile l'organo di trasmissione per il movimento delle ruote. Attraverso l'applicazione di un ingranaggio è stato possibile garantire un sistema in grado di fornire sufficiente coppia per permettere il movimento ad una buona velocità.

Dovendo essere collegato al pacco di batterie da 9V ciascuna, è stato possibile collegarlo attraverso il ponte, utilizzando come tramite uno switch per lo spegnimento. Le batterie sono state montate su una basetta millefori dotata di appositi connettori.

Durante i test in questa configurazione è sorto un problema fondamentale:

- L'ingranaggio del motore sembrava slittare senza riuscire a dare alla macchina sufficiente forza per farla muovere.
  Questo problema è stato risolto semplicemente attraverso l'utilizzo di alcune viti per il fissaggio del motore alla base, in modo da renderlo molto più stabile.

6.2 BATTERIE

Come asserito in precedenza, il pacco delle batterie da 9V è stato fissato mediante appositi connettori e sistemato in modo che potessero essere disabilitate fisicamente.

La medesima cosa è stata fatta per la batteria da 7.2V, quella che viene regolata per fornire l'alimentazione al circuito. Qui sorge però un secondo problema, forse più grave di quello precedente:

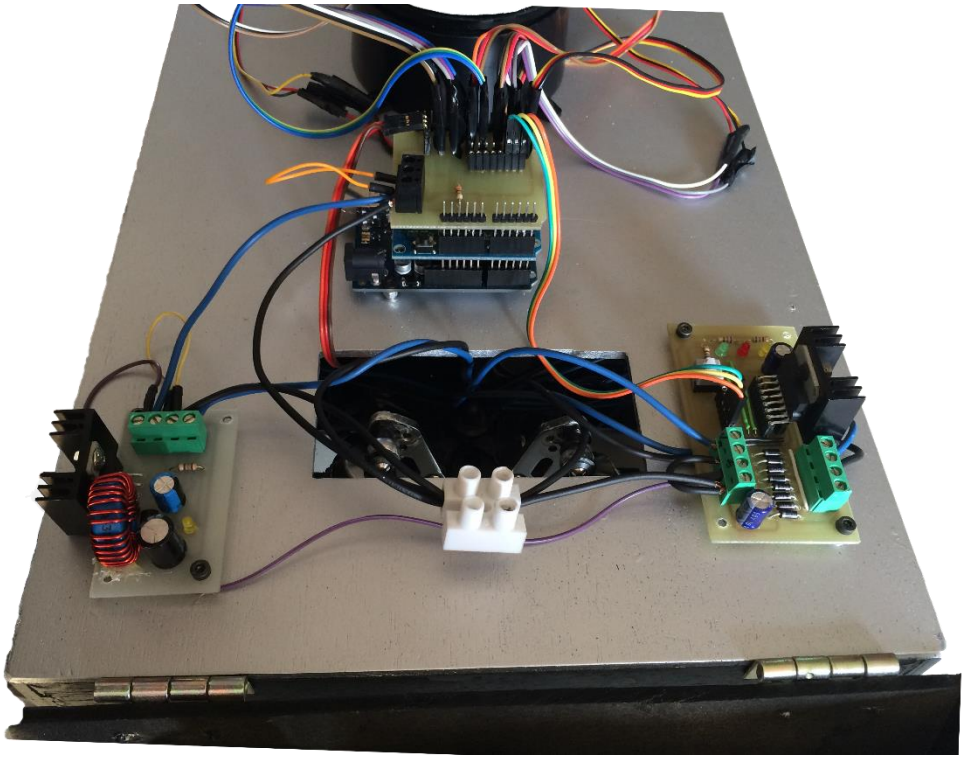
- Durante l'utilizzo del braccio, e quindi mentre il circuito dello step-down è messo sotto sforzo costantemente, risultavano esservi dei picchi di corrente purtroppo non gestibili che portavano il braccio a spegnersi. Questo avveniva soprattutto durante l'utilizzo dei due servomotori collegati in parallelo alla base, che sono infatti i più dispendiosi dal punto di vista dell'assorbimento della corrente. Il problema era evidentemente la batteria. Non era infatti lo switching ad essere il problema in quanto l'integrato garantisce 3A di corrente di spunto assorbita.
- Le soluzioni erano due e la nostra scelta si è spostata su quella più sicura ed economica. La prima prevedeva la sostituzione della batteria con una più resistente alle variazioni di corrente, ma sarebbe stata a questo punto necessaria qualche batteria molto più costosa ed avanzata. La soluzione che invece ci ha portato in poco tempo a risolvere il problema è stata l'applicazione di un condensatore di una capacità abbastanza elevata in parallelo alla batteria. Grazie ad un condensatore da $4.8mF$, si può garantire alla batteria una tensione che non subisce variazioni repentine e che fornisce anche una corrente dopo essersi precedentemente caricato.

Accanto alle batterie e al motore si trova il servomotore adibito allo sterzo delle ruote anteriori dell'auto.

Questi componenti si trovano in uno scomparto apposito tra il telaio e una struttura di legno creata appositamente. Questa struttura è stata realizzata in modo da garantire libertà di movimento alle ruote ma allo stesso tempo fornire una base d'appoggio alla circuiteria e al braccio. Si presenta come una scatola forata in corrispondenza delle sospensioni per permetterne l'adattamento alla macchina e per il passaggio dei cavi di alimentazione dalle batterie sottostanti. La

parte anteriore è coperta da un coperchio apribile al di sotto del quale è possibile trovare esposti i tre circuiti caratterizzanti della struttura:

- L'**alimentatore step-down**, al quale viene collegata direttamente la batteria e che fornisce l'alimentazione agli altri due circuiti;
- Il **ponte ad H,** collegato al motore sottostante, alle batterie da 9V per l'alimentazione del motore, allo step-down per l'alimentazione dell'integrato e con tre cavi di controllo collegati allo shield di adattamento di Arduino;
- **Arduino e i suoi shields,** il cuore pulsante dell'intero progetto, alimentato autonomamente dallo step-down utilizzando l'ultimo sketch caricato. È grazie al suo collegamento con la circuiteria e l'interfacciamento con il controller che si è in grado di controllare il tutto in modo semplice e intuitivo.



Infine la parte più appariscente della struttura, ovvero il **braccio**, è fissato posteriormente, nella zona sopra al motore, e i movimenti possibili sono di circa

180° coprendo sia i fianchi che il retro della macchina. Dopo aver fascettato i cavi in modo da non intralciare i movimenti del braccio, abbiamo collegato ogni servo al suo specifico controllo sullo shield di adattamento. Per far ciò in maniera più efficace era stata intagliata in precedenza una parte della copertura in modo da lasciare accessibili i collegamenti anche da chiusa. In questa posizione il braccio è in grado di raccogliere oggetti che si trovano posteriormente o lateralmente al furgone.

Come già spiegato la sua posizione del braccio è sicura in quanto prima di scollegarlo per passare al controllo macchina viene riportato nella posizione iniziale.