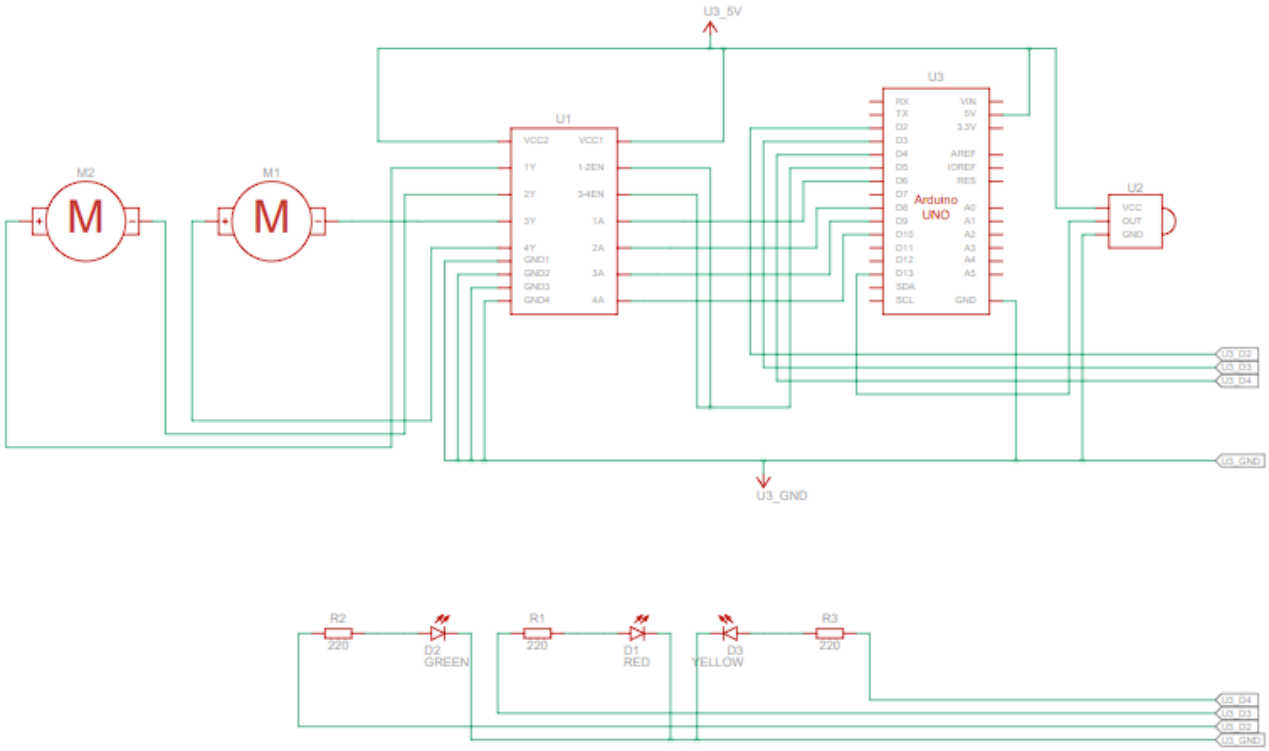
RELAZIONE N. 4 DEL: 13/05/2024

COGNOME PROGETTISTA: Emilii NOME PROGETTISTA: Nicolò

ESERCITAZIONE PONTE H

SCHEMA ELETTRICO

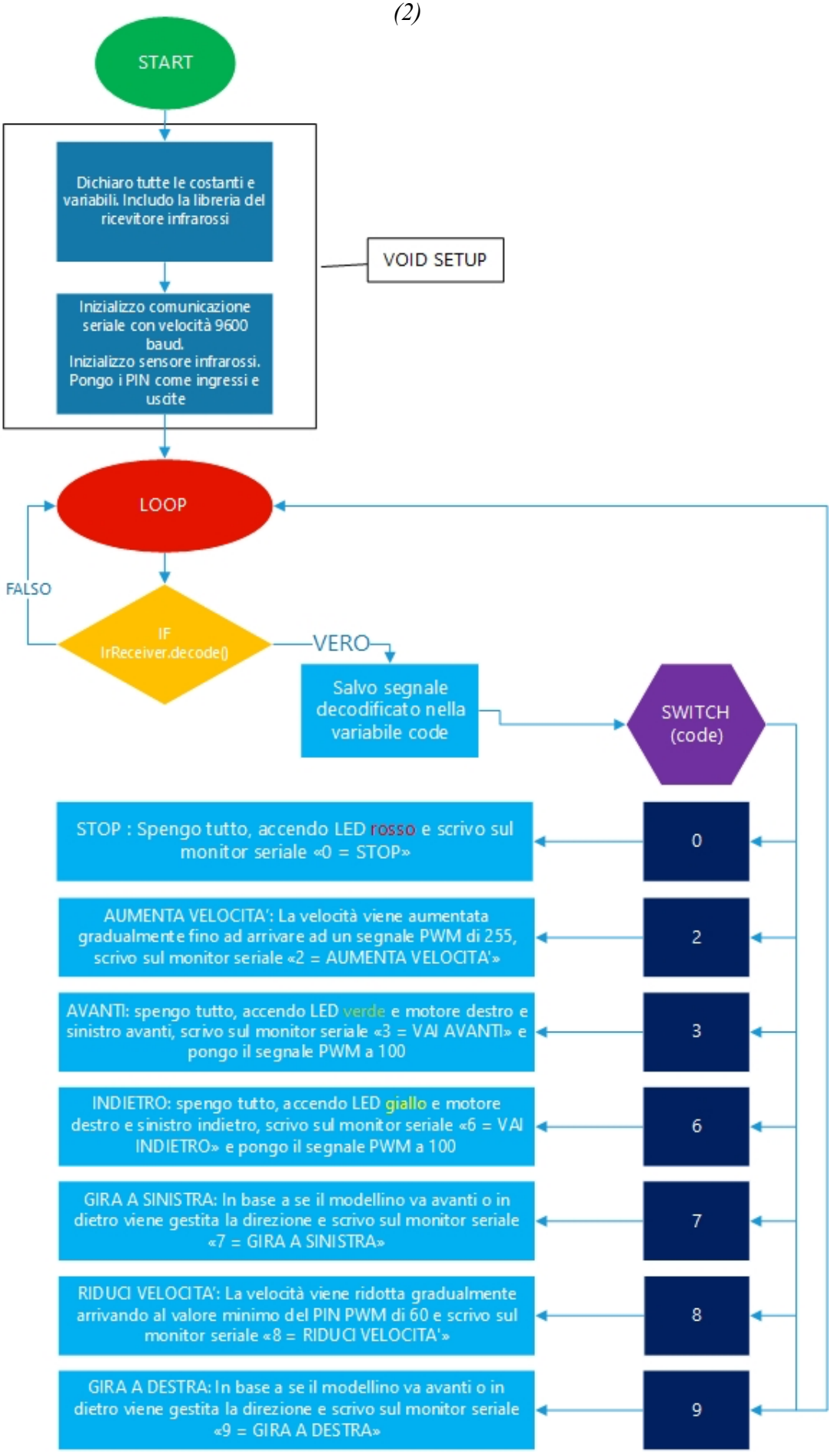
(1)



OGGETTO SOTTOPOSTO ALLA PROVA: L293D

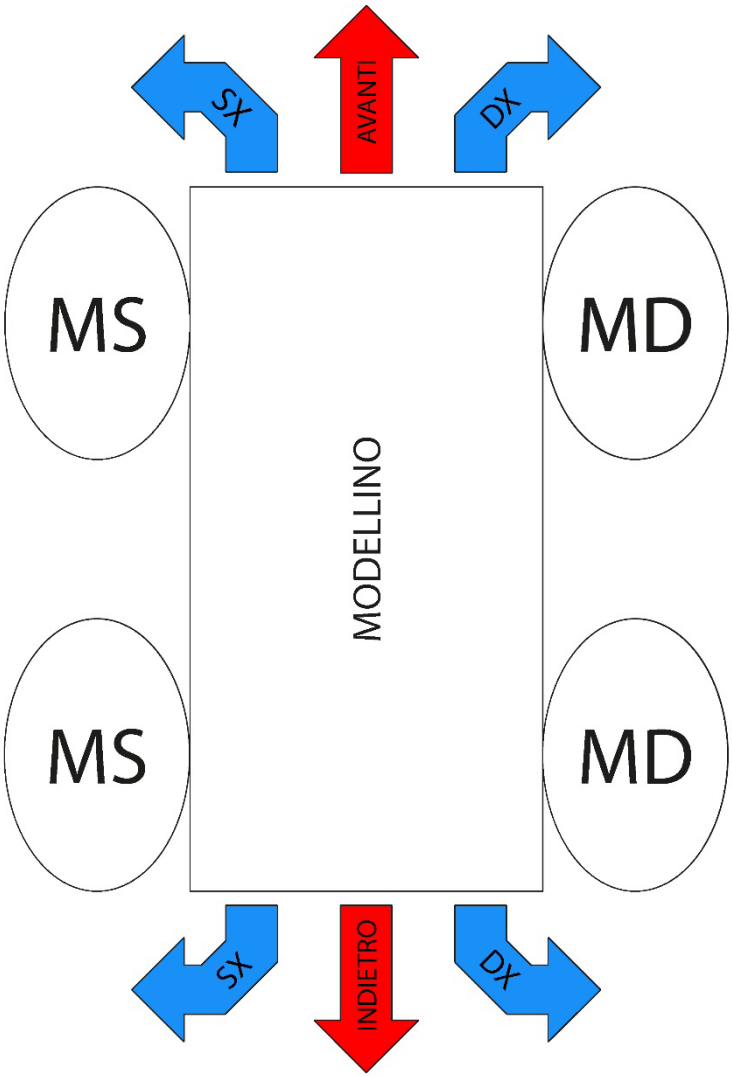
DIAGRAMMA DI FLUSSO

(2)



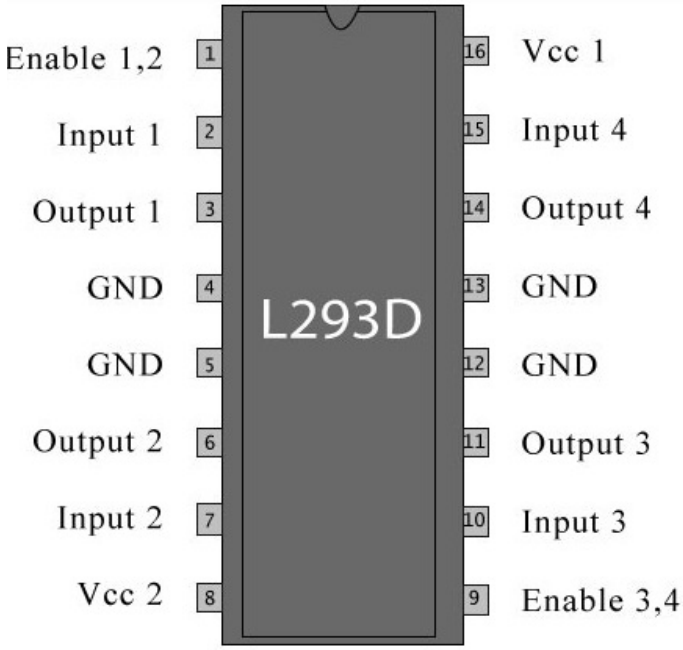
SCHEMA DI PRINCIPIO FUNZIONAMENTO MODELLINO

(3)



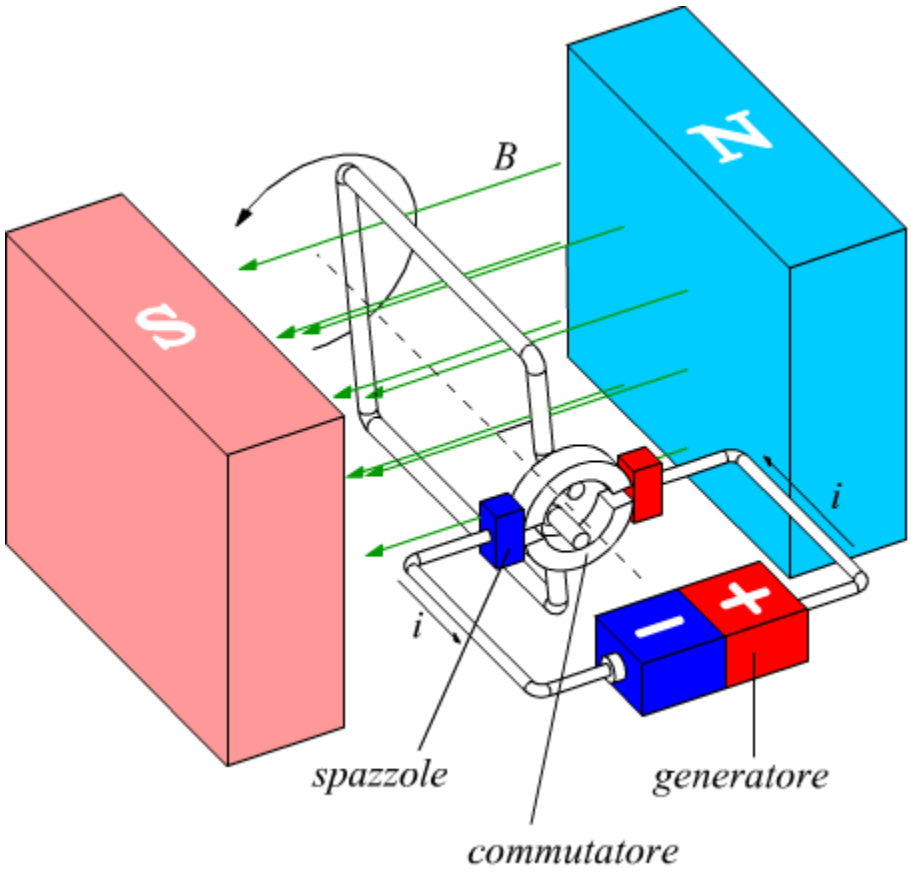
PIEDINATURA INTEGRATO L293D

(4)



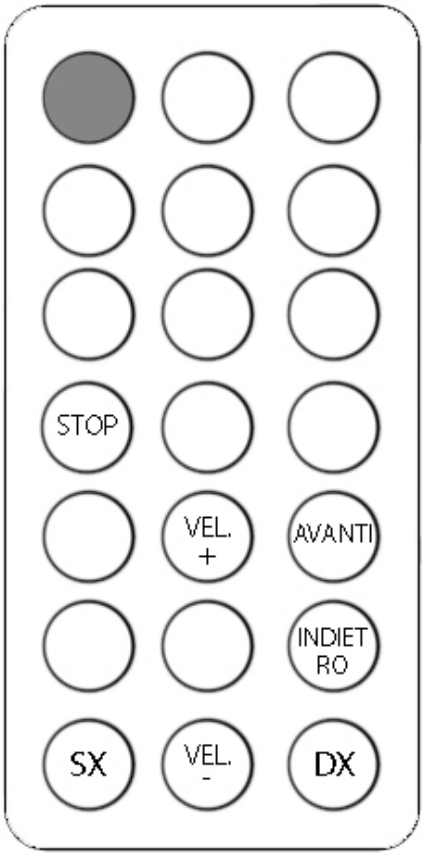
PRINCIPIO DI FUNZIONAMENTO MOTORE DC

(5)



DISPOSIZIONE TASTI

(6)



PROGRAMMA

(7)

```
#include <IRremote.h>

#define IR_RECEIVE_PIN 13
#define motoreD 6
#define motoreS 10
#define vel 5
#define LEDG 4
#define LEDR 3
#define LEDV 2
#define indietroD 8
#define indietroS 9
int veloc = 0;
char cont;

void setup() {
IrReceiver.begin(IR_RECEIVE_PIN);
Serial.begin(9600);
pinMode(motoreD, OUTPUT);
pinMode(motoreS, OUTPUT);
pinMode(indietroD, OUTPUT);
pinMode(indietroS, OUTPUT);
pinMode(vel, OUTPUT);
pinMode(LEDG, OUTPUT);
pinMode(LEDR, OUTPUT);
pinMode(LEDV, OUTPUT);

}

void loop() {

 if (IrReceiver.decode()) {

    unsigned long code = IrReceiver.decodedIRData.decodedRawData;

switch (code){

  case 0xF30CBF00: // STOP

digitalWrite(motoreD, LOW);
digitalWrite(motoreS, LOW);
digitalWrite(indietroD, LOW);
digitalWrite(indietroS, LOW);
digitalWrite(LEDG, LOW);
digitalWrite(LEDV, LOW);
digitalWrite(LEDR, HIGH);
analogWrite(vel, 0);
Serial.println("0 = STOP");
  break;
```

```
  case 0xEE11BF00: // VELOCITA' +

if(veloc<256){
veloc = veloc + 10;
analogWrite(vel, veloc);
Serial.println("2 = AUMENTA VELOCITA'");
break;

  case 0xED12BF00: // AVANTI

 digitalWrite(motoreD, HIGH);
digitalWrite(motoreS, HIGH);
digitalWrite(indietroD, LOW);
digitalWrite(indietroS, LOW);
digitalWrite(LEDG, LOW);
digitalWrite(LEDV, HIGH);
digitalWrite(LEDR, LOW);
cont = 'a';
Serial.println("3 = VAI AVANTI");
analogWrite(vel, 100);
break;

  case 0xE916BF00: // INDIETRO

digitalWrite(motoreD, LOW);
digitalWrite(motoreS, LOW);
digitalWrite(indietroD, HIGH);
digitalWrite(indietroS, HIGH);
digitalWrite(LEDG, HIGH);
digitalWrite(LEDV, LOW);
digitalWrite(LEDR, LOW);
cont = 'i';
Serial.println("6 = VAI INDIETRO");
analogWrite(vel, 100);
break;

  case 0xE718BF00: // GIRA A SINISTRA

 if(cont == 'a'){
digitalWrite(motoreD, HIGH);
digitalWrite(motoreS, LOW);
digitalWrite(indietroD, LOW);
digitalWrite(indietroS, LOW);
digitalWrite(LEDG, LOW);
digitalWrite(LEDV, HIGH);
digitalWrite(LEDR, LOW);
 }else if(cont == 'i'){
digitalWrite(motoreD, LOW);
digitalWrite(motoreS, LOW);
digitalWrite(indietroD, HIGH);
```

```
digitalWrite(indietroS, LOW);
digitalWrite(LEDG, HIGH);
digitalWrite(LEDV, LOW);
digitalWrite(LEDR, LOW);
 }
   Serial.println("7 = GIRA A SINISTRA");
}
break;

  case 0xE619BF00: // VELOCIA' -

   if(veloc>50){
    veloc = veloc - 10;
analogWrite(vel, veloc);
   }
    Serial.println("8 = RIDUCI VELOCITA'");
  break;

  case 0xE51ABF00: // GIRA A DESTRA

if(cont == 'a'){
digitalWrite(motoreD, LOW);
digitalWrite(motoreS, HIGH);
digitalWrite(indietroD, LOW);
digitalWrite(indietroS, LOW);
digitalWrite(LEDG, LOW);
digitalWrite(LEDV, HIGH);
digitalWrite(LEDR, LOW);
 }else if(cont == 'i'){
digitalWrite(motoreD, LOW);
digitalWrite(motoreS, LOW);
digitalWrite(indietroD, LOW);
digitalWrite(indietroS, HIGH);
digitalWrite(LEDG, LOW);
digitalWrite(LEDV, HIGH);
digitalWrite(LEDR, LOW);
 }
     Serial.println("9 = GIRA A DESTRA");
 break;
}



}
IrReceiver.resume();
  delay(10);
}
```

ELENCO COMPONENTI

(8)

- N.1 Arduino Uno
- N.1 Sensore infrarossi
- N.1 Integrato L293D
- N.1 Diodo LED Giallo
- N.1 Diodo LED Rosso
- N.1 Diodo LED Verde
- N.1 Breadboard
- N.2 Motori DC
- N.3 Resistori 220Ω

RELAZIONE

Il seguente programma ha lo scopo di controllare due motori (dx e sx) di un'ipotetica auto controllata con un telecomando a infrarossi. Lo schema di principio della stessa è visualizzabile nell'allegato *3*.

Il codice esegue i seguenti step:

1. **Inizializzazione dei pin e della comunicazione seriale**: Nel metodo ***'setup()'***, vengono inizializzati i pin utilizzati per controllare il motore, gli indicatori LED e il sensore IR. Inoltre, viene inizializzata la comunicazione seriale a una velocità di 9600 baud.

2. **Loop principale**: Nel metodo ***'loop()'***, il programma entra in un ciclo continuo in cui controlla costantemente se è stato ricevuto un segnale dal sensore IR.

3. **Ricezione dei comandi IR**: Utilizzando la funzione ***'IrReceiver.decode()'***, il programma verifica se è stato ricevuto un segnale IR. Se un segnale è stato ricevuto e decodificato correttamente, il programma procede ad eseguire le istruzioni corrispondenti al codice ricevuto.

4. **Switch-case per gestire i comandi**: Il codice utilizza uno statement switch-case per gestire diversi comandi IR. Ogni caso corrisponde a un determinato codice ricevuto dal sensore IR.

5. **Esecuzione dei comandi**: A seconda del codice ricevuto, vengono eseguite azioni specifiche come arrestare il motore, aumentare o diminuire la velocità, muovere il motore in avanti, indietro, a sinistra o a destra, e controllare gli indicatori LED. L'inversione di marcia può avvenire anche senza passare per lo stato di STOP dato che utilizzando motori di piccola taglia non è stata tenuta in considerazione l'inerzia del rotore dello stesso. Questa pratica a lungo andare può compromettere le prestazioni del motore.

6. **Controllo della velocità**: Il programma utilizza una variabile ***'veloc'*** per controllare la velocità del motore. Questa variabile viene modificata dai comandi per aumentare o diminuire la velocità.

7. **Gestione degli indicatori LED**: A seconda del comando ricevuto, vengono accesi o spenti gli indicatori LED per indicare lo stato del motore (ad esempio, avanti, indietro, stop).

8. **Comunicazione seriale**: In ogni caso, il programma stampa un messaggio sulla porta seriale per indicare quale comando è stato eseguito.

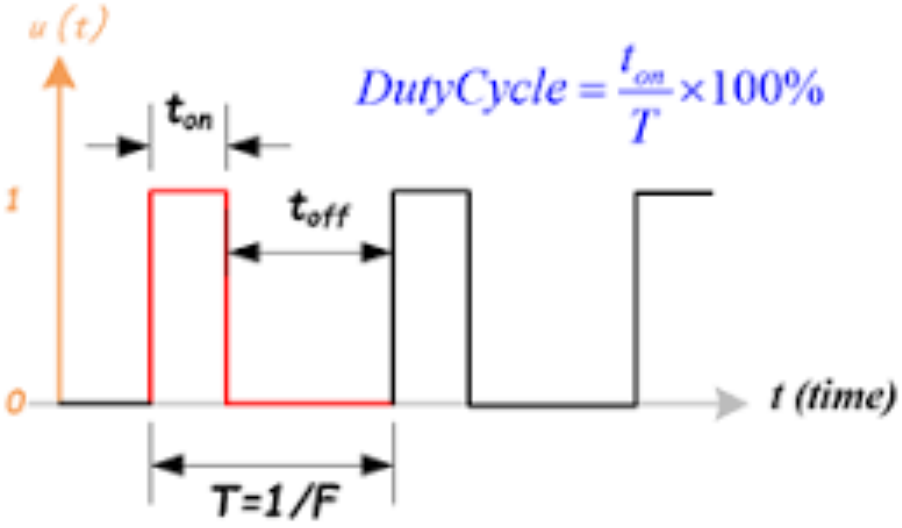
9. **Ripresa della ricezione IR**: Dopo aver gestito un comando IR, il programma riprende la ricezione per attendere il prossimo segnale IR.

Per il controllo dei motori è stato utilizzato un L293D che è un popolare chip driver di motori utilizzato per controllare la direzione e la velocità di un motore elettrico. È comunemente utilizzato in progetti di robotica e automazione. La sua piedinatura è visibile nell'allegato *4*.

Ecco alcune caratteristiche principali del L293D:

1. **Driver bidirezionale:** Il L293D è in grado di controllare la rotazione del motore in entrambe le direzioni (avanti e indietro).
2. **Driver di potenza:** È in grado di gestire correnti fino a 600 mA per canale (e picchi di corrente fino a 1,2 A) senza dissipatori di calore. Con dissipatori di calore appropriati, può gestire correnti più elevate.
3. **Controllo della velocità:** È possibile controllare la velocità del motore utilizzando la modulazione in larghezza di impulso (PWM). Questa funzionalità consente di variare la velocità del motore variando il ciclo di lavoro del segnale PWM.
4. **Protezione termica e di sovracorrente:** Il chip è dotato di funzionalità di protezione termica e di sovracorrente che lo proteggono da danni in caso di condizioni di carico elevate o di surriscaldamento.
5. **Interfaccia di controllo semplice:** È possibile controllare il L293D utilizzando segnali di controllo digitali provenienti da microcontrollori o da altri dispositivi a logica digitale.

Per la gestione della velocità come precedentemente citato è stato utilizzato un segnale PWM generato da una scheda Arduino Uno.



Un segnale PWM (Pulse Width Modulation) generato da Arduino Uno è un segnale digitale che oscilla tra due stati, HIGH (alto) e LOW (basso), con una frequenza costante, ma con la durata del tempo trascorso nello stato HIGH variabile, controllata tramite il parametro chiamato "ciclo di lavoro" o "duty cycle".

Ad esempio, se si desidera controllare la luminosità di un LED utilizzando PWM, si può impostare un valore di ciclo di lavoro che determina quanto tempo il segnale rimane HIGH rispetto al tempo totale di un periodo. Un ciclo di lavoro del 50% significa che il segnale è alto per la metà del tempo e basso per l'altra metà del tempo, così da ottenere una tenzione di uscita che è circa la metà della tenzione HIGH.

Arduino Uno offre 6 pin PWM, contrassegnati come 3, 5, 6, 9, 10 e 11. Tuttavia, è importante notare che Arduino Uno ha una risoluzione PWM di 8 bit, il che significa che il valore del ciclo di

lavoro può variare da 0 (nessun impulso, segnale sempre LOW) a 255 (segnale sempre HIGH). Questo fornisce una gamma di controllo di 256 livelli di luminosità o velocità.

Per la realizzazione dello schema elettrico (*allegato 1*), diagramma di flusso (*allegato 2*), schema di principio (*allegato 3*) e programma arduino (*allegato 7*) sono stati utilizzati in ordine i seguenti software:

- Autodesk Tinkercad
- Microsoft Visio Professional
- Adobe Illustrator, Adobe Photoshop
- Arduino IDE

Come si può notare dall'allegato *6* la disposizione dei tasti non è stata fatta in ordine casuale, ma seguendo una logica spaziale ben precisa, che prevede di posizionare i tasti per girare a destra e a sinistra nell'estremità inferiore nel lato di rotazione, di posizionare in una posizione più "alta" il tasto per andare avanti e vice versa per andare indietro (lo stesso principio è stato applicato anche per i tasti che regolano la velocità di rotazione dei motori). In fine il tasto di STOP ("emergenza") è stato posizionato nel punto sinistro più alto dato che è l'area con minor densità di tasti (abilitati) così che in caso di necessità la pressione accidentale di un tasto che non sia quello di STOP sarà minima.