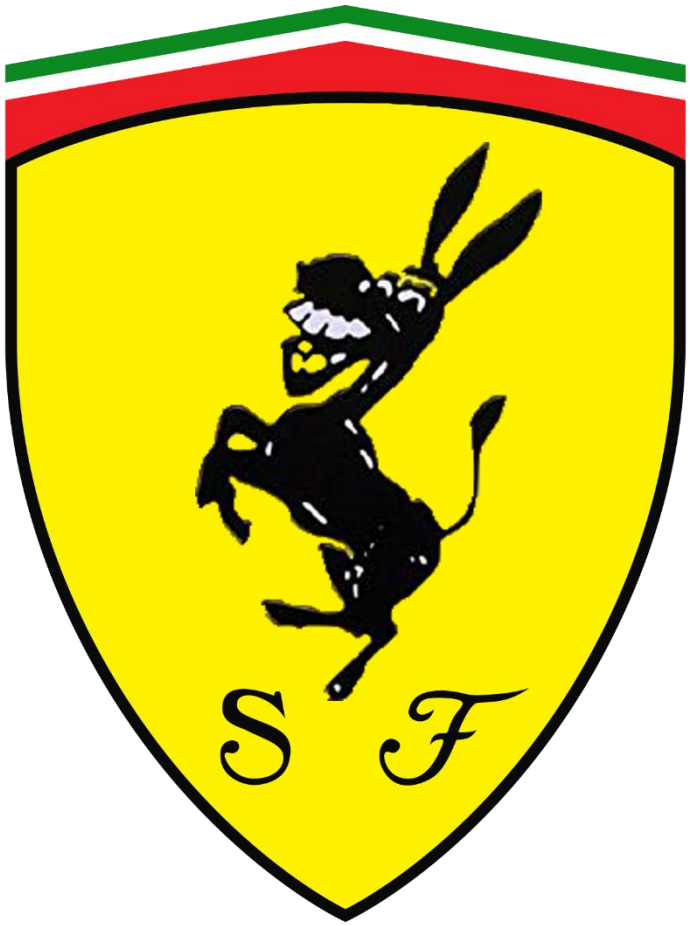


Scuderia

"Ferari"

Di Andrea Vaccaro, Michele Pedroni, Cheng Kai Liu Paluan, Matteo Braguzzi

"progetto telecomunicazione comunicazione wireless"

Itis E.Fermi MN 2022-2023

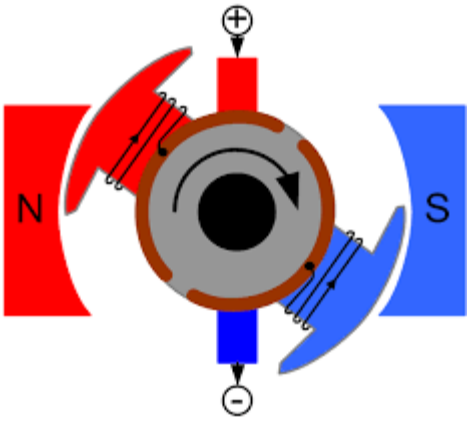
4Ciin gruppo 1

Indice:

Materiali Necessari

Motore DC

Un motore DC, o motore a corrente continua, è un dispositivo che converte l'energia elettrica in energia meccanica. Funziona applicando una corrente continua attraverso un avvolgimento o bobina all'interno del motore. Noi ne utilizzeremo 1 per ogni ruota della macchinina, per un totale di 4 motori.



Transistor

Un transistor è una specie di interruttore per la corrente, noi nello specifico usiamo i transistor di tipo NPN, e ne useremo 2, dove ognuno controlla i 2 motori di un lato della macchinina, le basi dei transistor verranno alimentate o meno dai pin logici dell'Arduino Uno, cosi da poter gestire l'andamento della macchinina a seconda dei comandi.

Diodo

Un diodo è un componente che rende possibile il passaggio della corrente in un solo verso, noi ne useremo 2, 1 per ognuno dei 2 circuiti, cosi da evitare il ritorno di corrente dei Motori DC, che provocherebbe il danneggiamento del transistor.

Resistenze

In questo circuito impiegheremo 2 resistenza da 330Ω l'una, ognuna delle 2 alla base di un transistor, cosi da evitarne il danneggiamento e poter permetterne il corretto impiego.

Arduino UNO (non originale)

L’ Arduino che andremo ad utilizzare non sarà originale, ma emula il funzionamento di un’Arduino Nano; lo utilizzeremo per controllare i 2 pin digitali, che a loro volta controlleranno i motori della macchinina secondo le esigenze; in più sarà colui che riceverà ed elaborerà i dati ricevuti attraverso il modulo bluetooth.

HC-05

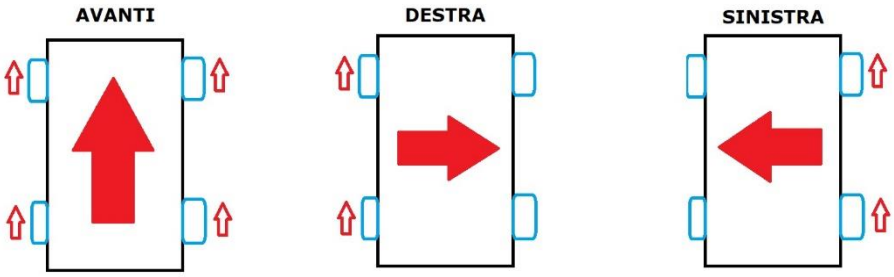
L’HC-05 è il modulo bluetooth che utilizzeremo per ricevere i dati in maniera wireless, servendoci per l’appunto della comunicazione bluetooth seriale.

Progettazione di funzionamento

Logica di movimenti base

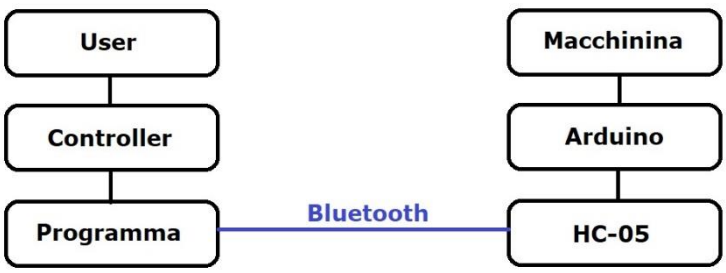
Avendo a disposizione 4 motori comandabili singolarmente, abbiamo deciso di dividerli in sezioni per lato (destra e sinistra), cosi che potessimo alimentare entrambi i lati insieme, oppure uno per volta; facendo ciò possiamo controllare il movimento della macchinina, attraverso i 3 principali movimenti:

- Avanti, dove entrambi i lati vengono alimentati e tutti insieme fanno muovere la macchinina in avanti.
- Sterzata a destra, dove solo il lato sinistro viene alimentato, cosicché il lato destro faccia da perno e ciò renda possibile la sterzata.
- Sterzata a sinistra, dove segue la stessa logica della sterzata a destra, ma invertendo l'alimentazione dei due lati, quindi verrà alimentato solo il lato destro.



Logica di controllo

Per il controllo della macchinina da remoto, abbiamo pensato ad un controller molto semplice, sia con GUI, che con acquisizione di tasti; il tutto doveva semplicemente fare da tramite tra User e macchinina.



Software

Sviluppo di controller remoto

Scelta del linguaggio di programmazione

Per quanta riguarda il linguaggio di programmazione, la nostra prima scelta era java, in quanto argomento del nostro anno scolastico; però dopo esserci resi conto della complessità di tale linguaggio per gestire la comunicazione bluetooth abbiamo deciso di provare altri linguaggi; siamo arrivati ad una soluzione, ovvero Python, e all' utilizzo della sua libreria "PySerial", che permette di comunicare in maniera seriale attraverso una porta "COM" del pc da noi specificata; tutto ciò ha reso la realizzazione (lato software) del progetto molto più semplice.

Logica dei dati da trasferire

Abbiamo cercato di rendere la comunicazione di dati tra controller ed Arduino la più comprensibile possibile, per far ciò abbiamo creato una specie di protocollo tra Master e Slave, dove:

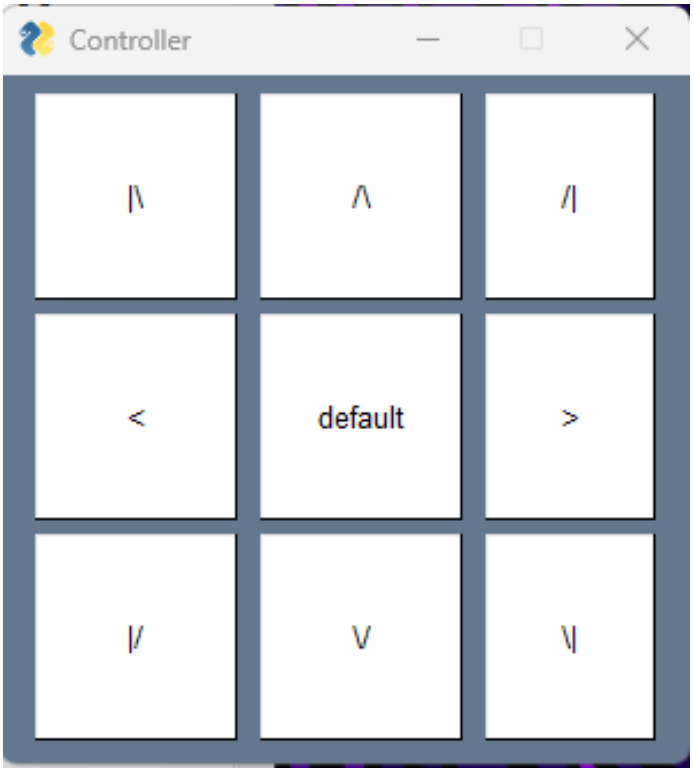
DATO INVIATO:	ISTRUZIONE:
1	Fermo
2	Avanti
3	Destra
4	Sinistra

Implementazione del codice

Per il codice sono state proposte più versioni, dove le prime presentavano un'interfaccia grafica abbozzata, mentre la definitiva utilizza una funzione per eseguire determinate istruzioni dipendentemente dal tasto che viene premuto (sulla tastiera), e logicamente noi abbiamo collegato i già canonici "wasd", con anche i caratteri "q" ed "e" per altre opzioni, le istruzioni poi utilizzavano dei metodi di "PySerial", che avviavano il trasferimento seriale di dati via bluetooth.

Prototipi e codice definitivo:

prima prova di controller:



Prima sperimentazione comunicazione seriale:

```
import serial
ser = serial.Serial("COM3", 9600, timeout = 1)
while(True):
    uInput = input("command:")
    ser.write(bytearray(uInput,'ascii'))
    if(uInput=="10"):
        ser.close()
```

Codice definitivo:

controller_definitivo_1.py - C:\Users\Lenovo\Desktop\Macchina-

File Edit Format Run Options Window Help

```
#controller definitivo
import serial
from tkinter import *
import sys

ser = serial.Serial("COM3", 9600, timeout=1)

window = Tk()

def avanti(event):
    print("avanti")
    ser.write(bytearray('2', 'ascii'))

def indietro(event):
    print("indietro")
    ser.write(bytearray('1', 'ascii'))

def sinistra(event):
    print("sinistra")
    ser.write(bytearray('3', 'ascii'))

def destra(event):
    print("destra")
    ser.write(bytearray('4', 'ascii'))

def destra_avanti(event):
    print("destra-avanti")
    ser.write(bytearray('6', 'ascii'))

def sinistra_avanti(event):
    print("sinistra-avanti")
    ser.write(bytearray('5', 'ascii'))

def none(event):
    print("none")
    ser.close()
    window.destroy()
    sys.exit(0)

window.bind("<w>", avanti)
window.bind("<s>", indietro)
window.bind("<a>", sinistra)
window.bind("<d>", destra)
window.bind("<e>", destra_avanti)
window.bind("<q>", sinistra_avanti)
window.bind("<c>", none)

window.mainloop()
```

Sviluppo del programma di controllo

Logica delle istruzioni

Come detto prima, per le varie istruzioni abbiamo adottato un "protocollo" molto semplice, quindi logicamente nel nostro arduino, all' arrivo del dato di valore "2", che stava ad indicare l' istruzione "avanti", venivano rilasciati valore di tensione "HIGH" nei digitalPin predestinati alla gestione dei motori, la stessa regola è stata applicati per le altre istruzioni, cosi mescolando la logica dei movimenti base e la logica delle istruzioni, siamo riusciti a ricavare il programma "Slave" dall' arduino.

Implementazione del codice

Servendoci dell' IDE Arduino, abbiamo sviluppato il codice per la gestione dei pin digitali:

```
#include <avr/wdt.h>
#include <SoftwareSerial.h>
#define VCCBLU 12  //pin per hc05
#define GNDBLU 11
SoftwareSerial mySerial(10, 9);

//pin dei motori

//dietro sinista
#define DX 10
#define SX 11

void setup() {
  Serial.begin(9600);    // inizializza la comunicazione seriale a 9600 bps
  mySerial.begin(9600);  //creo la mia seriale digitale
  pinMode(VCCBLU, OUTPUT);
  pinMode(GNDBLU, OUTPUT);
  digitalWrite(VCCBLU, HIGH);  //accendo hc05
  digitalWrite(GNDBLU, LOW);
  while (!Serial) { ; }  //attendo che si stabilisca la connessione
  wdt_enable(WDTO_1S);

  //setto i pin dei motiri come output
  pinMode(DX, OUTPUT);
  pinMode(SX, OUTPUT);
}

void loop() {
  wdt_reset();
```

```
  if (mySerial.available() > 0) {         // se ci sono dati disponibili sulla
seriale
    char incomingByte = mySerial.read();  // leggi un carattere dalla seriale
    if (incomingByte == '2') {            // se il carattere ricevuto è '2'
      avanti();                           //tutti accesi
      delay(200);
      wdt_reset();
      spenti();  //tutti i motori spenti
    } else if (incomingByte == '1') {
      spenti();  //motori spenti
      wdt_reset();
    } else if (incomingByte == '3') {
      destra();  //motori destra
      delay(200);
      wdt_reset();
      spenti();  //motori spenti
    } else if (incomingByte == '4') {
      sinistra();  //motori sinistra
      delay(200);
      wdt_reset();
      spenti();  //motori spenti
    } else {
      spenti();  //motori spenti
      wdt_reset();
    }
  }
  while (mySerial.available() > 0) {  //pulisco la seriale
    wdt_reset();
    char incomingByte = mySerial.read();
  }
}

void avanti() {
  digitalWrite(SX, HIGH);
  digitalWrite(DX, HIGH);
  return;
}

void sinistra() {
  digitalWrite(SX, LOW);
  digitalWrite(DX, HIGH);
  return;
}

void destra() {
  digitalWrite(SX, HIGH);
  digitalWrite(DX, LOW);
  return;
}
```

```
void spenti() {
  digitalWrite(SX, LOW);
  digitalWrite(DX, LOW);
  return;
}
```

Risoluzione delle criticità

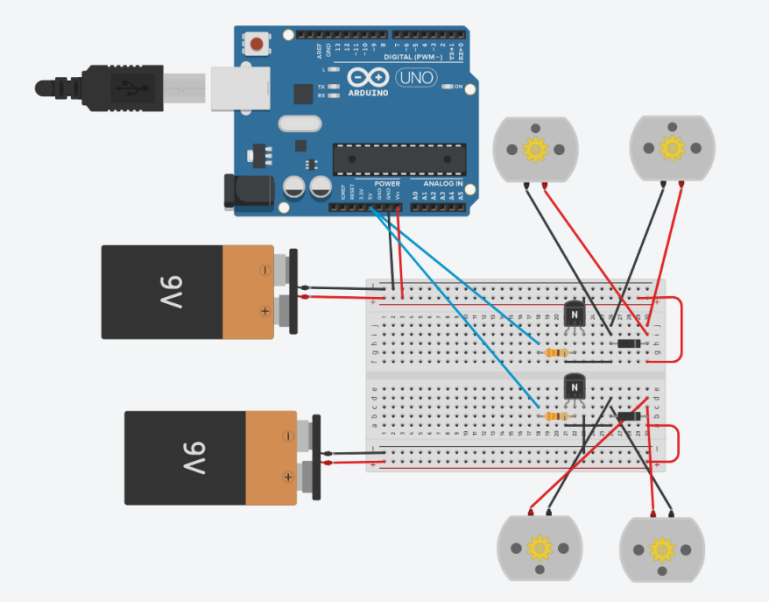
All'inizio abbiamo trovato molti problemi nella corretta interpretazione dei dati ricevuti dal programma Python, e ciò causava un loop infinito all'interno del nostro software arduino, perciò come prima cosa abbiamo implementato la libreria "avr/wdt.h", che ci permette di resettare l'arduino e farlo ripartire ogni qualvolta il timer (impostato da noi) si azzerasse, cosi da impossibilitare l'evento di loop infiniti; infine, dopo svariati test abbiamo capito che il vero problema risedeva nei troppi dati ricevuti, e ciò portava ad un rallentamento considerevole del programma nel passare da un' istruzione ad un'altra, perciò dopo qualsiasi istruzione eseguita, abbiamo messo un loop di while che ritornasse alla fine dei dati seriali, cosi da rendere il programma pronto all' ultima istruzione ricevuta.

```
while (mySerial.available() > 0) {  //pulisco la seriale
      wdt_reset();
      char incomingByte = mySerial.read();
}
```

Hardware

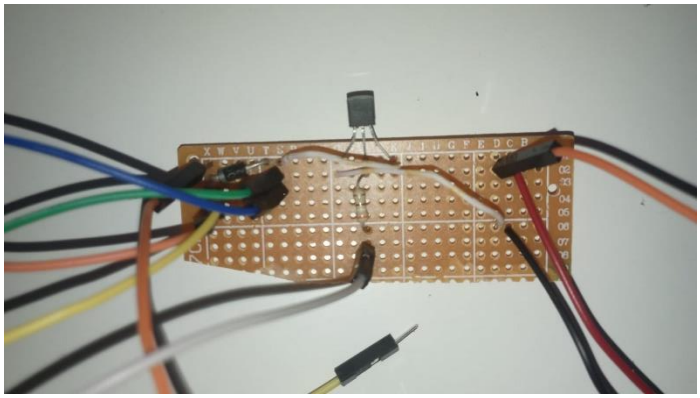
Sviluppo modello Tinkercad del circuito

Prima di creare il nostro circuito nell' effettivo, lo abbiamo realizzato su Tinkercad per testarlo, e verificarne il funzionamento.



Implementazione del circuito su millefori

Dopo aver creato i progetto in formato digitale, lo abbiamo realizzato su millefori, creando i corretti collegamenti tra i vari componenti; grazie alle ore di laboratorio siamo riusciti a usare il saldatore per lo stagno, cosi da rendere più sicuro il tutto.



Test del circuito fisico

Insieme al prof, dopo aver realizzato il circuito, lo abbiamo testato, con risultati positivi in laboratorio, grazie ad un generatore di tensione in continua a 5V, come ci aspettavamo infatti, al collegamento della base del transistor al collegamento positivo i 2 motori si azionavano.

Analisi criticità generali e risoluzione

Ritorno di corrente

Come detto prima nella sezione del “Diodo”, abbiamo accennato ad un ritorno di corrente, una criticità del circuito che ci ha dato molti problemi; infatti quando andavamo a spegnere il Motore DC questo causava un ritorno di corrente, che finiva per bruciare il transistor, danneggiando irreparabilmente il circuito.

Risoluzione

Dopo esserci resi conto di questo problema abbiamo installato un diodo tra il collegamento positivo e negativo del motore DC, cosi da rendere impossibile questo ritorno di corrente.

Criticità peso/potenza

Ovviamente un problema di cui tener conto è il peso della macchinina relativamente alla potenza sprigionata dai motori.

Risoluzione

Per questo abbiamo deciso di avere una batteria dedicata per ogni coppia di motori, cosi da acquisire peso con la batteria supplementare, ma da poterlo sostenere grazie alla maggiore potenza sprigionabile.

Criticità strutturali

Ovviamente anche la scocca della macchinina giocava un ruolo fondamentale per la realizzazione del progetto.

Risoluzione

Ovviamente abbiamo confrontato più materiali diversi tra di loro, ma le due opzioni più gettonate sono state alluminio (semplici lattine) e semplice cartone, inizialmente eravamo più orientati verso l'alluminio che alla fine si rivelò proprio la nostra scelta, utilizzando una lattina come scocca per la macchinina.

Considerazioni Finali

Il gruppo è stato molto produttivo durante la realizzazione del progetto, rispondendo in maniera positiva ai vari tentativi falliti, e non perdendo mai la motivazione; il risultato ottenuto non è all' altezza delle aspettative iniziali ma siamo comunque fieri di un lavoro completo, che abbia soddisfatto i requisiti fondamentali che ci siamo posti. Il saper lavorare in gruppo è stato fondamentale per il completamento del progetto, un nostro grande punto di forza è certamente stato la divisione dei lavori, in maniera che ognuno fosse partecipe al progetto, e la comunicazione tra i membri del gruppo.