RC Car Controlled With Hand Motion

Table of contents

01
Components

02
Circuit Diagram

03
Code

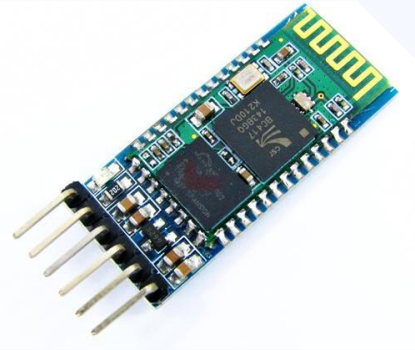
04
Git Hup

Components

RC car components

Arduino UNO

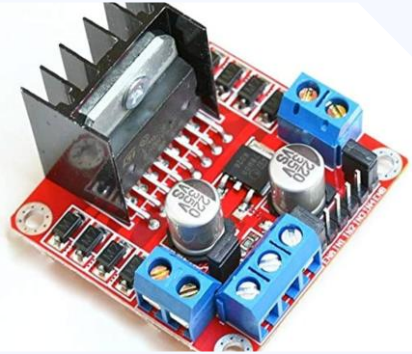
RC car kit



Bluetooth module HC05

RC car components

Motor

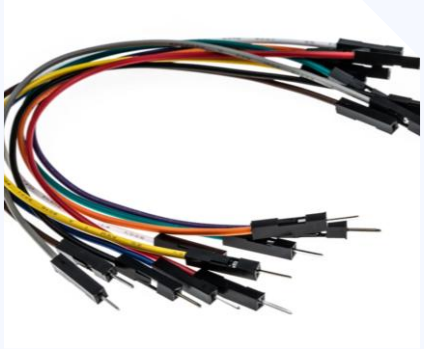


Motor driver L298N

3.7 volt battery x3

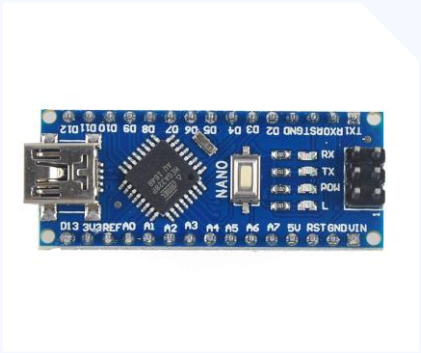
RC car components

Battery Holder

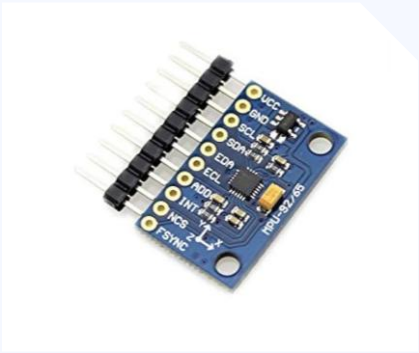


Jumper Wires

Glove components



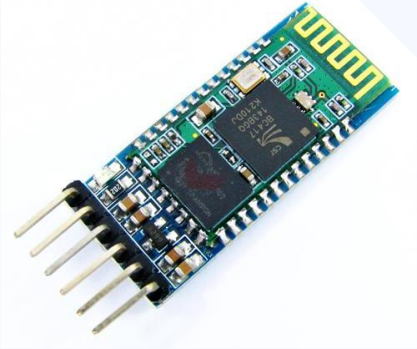
Arduino
Nano



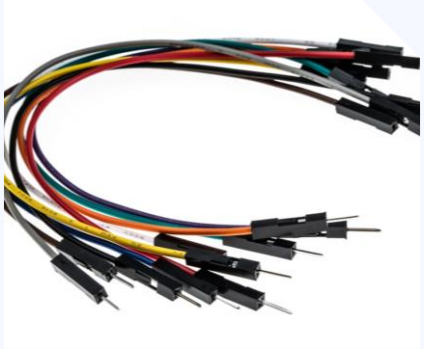
MPU9250

Flex Sensor

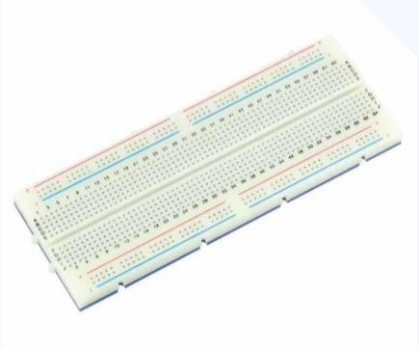
Glove components



Bluetooth module HC05



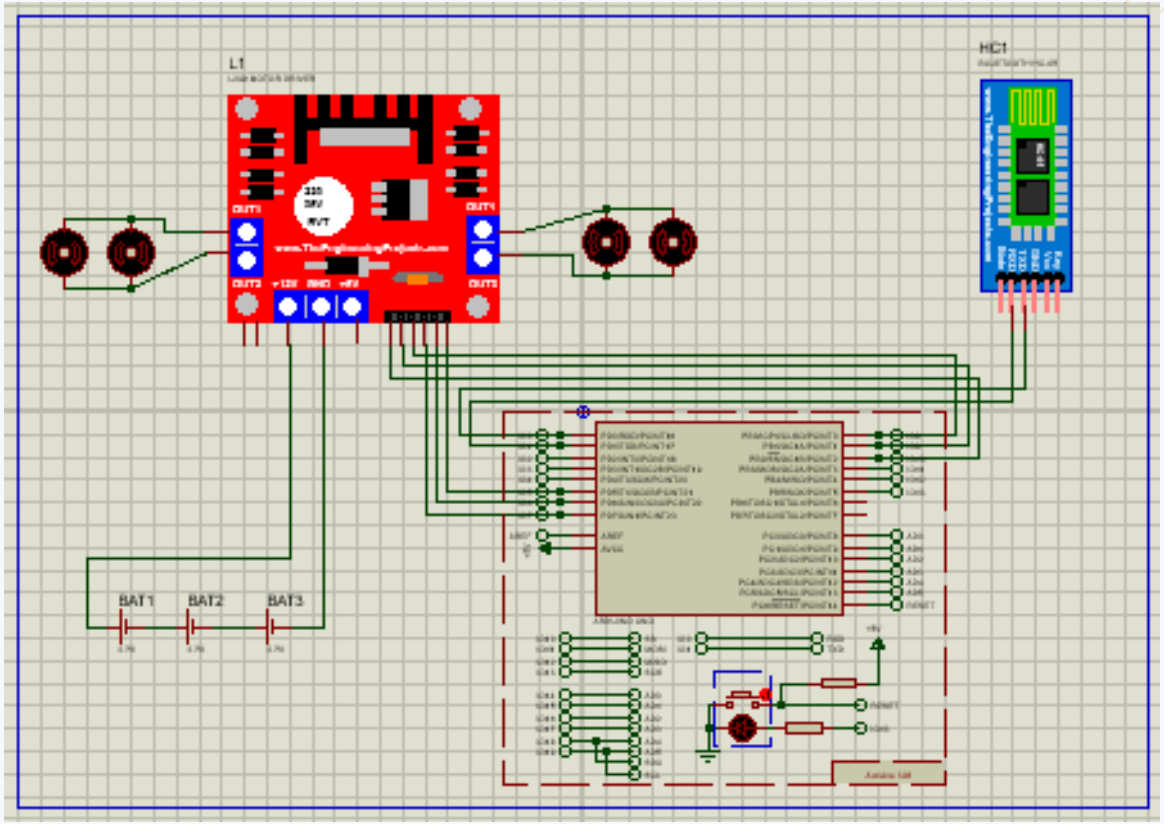
Jumper Wires



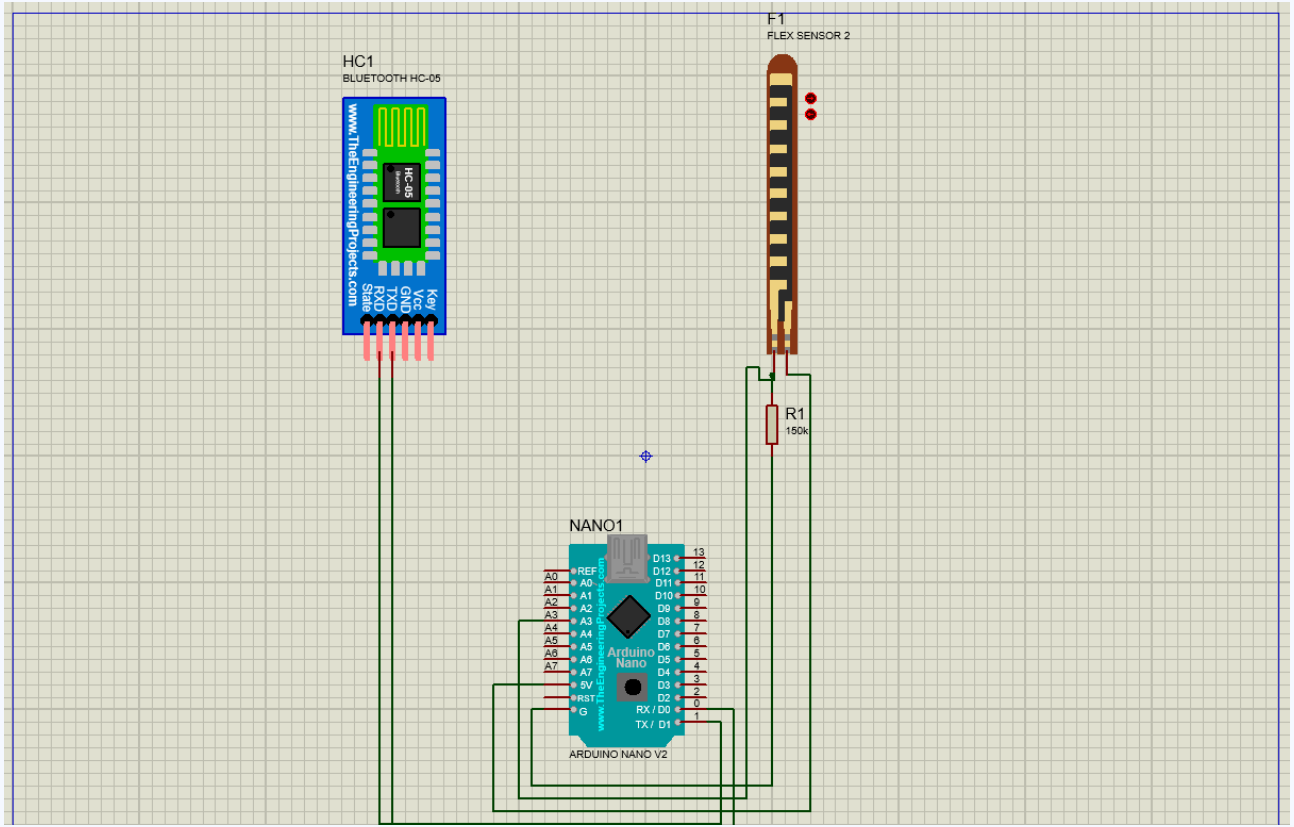
Bread Board

Circuit Diagram

RC car circuit



Glove Circuit



Code

Glove Code

Libraries used

```
#include <SoftwareSerial.h>
#include "MPU9250.h"
#include <Wire.h>
```

First one is for Bluetooth communication via UART.
Second one is used to get values from the MPU.
Third one is for the I2C communication that the MPU9250 works with.

Libraries used

```
SoftwareSerial btSerial(2, 3); // RX=2 (unused), TX=3 → HC-05 RX
MPU9250 mpu;

// Movement command codes
#define FORWARD_LEFT     0
#define FORWARD_RIGHT    1
#define BACKWARD_LEFT    2
#define BACKWARD_RIGHT   3
#define FORWARD          4
#define BACKWARD         5
#define STOP             6
#define FLEX             A3
int Direction = 0;
int Move = 0;
int speed = 0;
int last_spd = -1 ;
int last_cmd = -1 ;
```

Initializing UART and making MPU object for later use.
Initializing some macros that is used to send directions.
And some global variables to be kept always updated.

Setup()

```
void setup() {
  Serial.begin(38400);   // Debug monitor
  btSerial.begin(38400); // HC-05 data mode baud rate
  Serial.println("Nano BT Master ready @ 38400");
  pinMode (FLEX, INPUT);
  Wire.begin();
  delay(2000);

  if (!mpu.setup(0x68)) {
    while (1) {
      Serial.println("MPU connection failed.");
      delay(5000);
    }
  }
}
```

The setup is rather simple and self explanatory. Everything is just getting initialized!

Loop()

```
void loop() {
  speed = analogRead(FLEX);
  speed = map(speed, 720 , 900 , 0, 255); // scale 0-1023 → 0-255

  if (mpu.update()) {
    static uint32_t prev_ms = millis();
    if (millis() - prev_ms > 25) {
      int cmd = getMovementCommand();

   if (last_cmd != cmd || last_spd >= (speed + 15) || last_spd <= (speed - 15))
      {
      btSerial.write(cmd);   // command
      btSerial.write(speed); // speed
      Serial.print("Sent cmd: "); // just for debugging
      Serial.print(cmd);          // just for debugging
      Serial.print(" | speed: "); // just for debugging
      Serial.println(speed);      // just for debugging
      last_spd=speed;
      last_cmd=cmd;
      }
      prev_ms = millis();
    }
  }
}
```

Speed takes flex reading. And we mapped it from 0 to 255 for a certain reason coming later.
Every loop MPU gets updated with its new values.
cmd is used to get the movement direction command.
before sending we check if a new cmd has been recorded and also if speed got a new record with some sensitivity. If so then send it. If not then it returns and execute the loop from the start

GetMovmentCommand()

```
int getMovementCommand() {
  Direction = mpu.getRoll();
  Move = mpu.getPitch();
  int command = STOP;

  if (Move > 30) {
    if (Direction > 30) command = FORWARD_RIGHT;
    else if (Direction < -30) command = FORWARD_LEFT;
    else if (Direction > -10 && Direction < 10) command = FORWARD;
  }
  else if (Move < -30) {
    if (Direction > 30) command = BACKWARD_RIGHT;
    else if (Direction < -30) command = BACKWARD_LEFT;
    else if (Direction > -10 && Direction < 10) command = BACKWARD;
  }
  return command;
}
```

This function is used to process the values coming of the MPU. Move is (forward-backward) and Direction is (right-left).
We initially set command as stop then check MPU values..
after the function deduced the outcome it returns the command to check upon the command if it changed or not. (we don’t want to send the same value continuously)

Car Code

Libraries, macros, variables

```
#include <SoftwareSerial.h>

SoftwareSerial btSerial(10, 11); // RX=10 ← HC-05 TX, TX=11 unused
// Movement command codes
#define FORWARD_LEFT     0
#define FORWARD_RIGHT    1
#define BACKWARD_LEFT    2
#define BACKWARD_RIGHT   3
#define FORWARD          4
#define BACKWARD         5
#define STOP             6

//MOTOR I/O
#define IN1              7
#define IN2              6
#define IN3              4
#define IN4              5
#define ENA              3
#define ENB              9


 int flex_speed =100;
 int command = -1;
 int value = -1;
```

Same stuff as the Glove code her but slightly different… the Receiver is controlling motor direction and speed too! So we made more macros!
flex_speed is for the sent flex value.
command is for the direction command.
value is where either the flex value or the command is stored in first.

Setup()

```
void setup() {
  Serial.begin(38400);   // Debug monitor
  btSerial.begin(38400); // HC-05 data mode baud rate
  Serial.println("Uno BT Slave ready @ 38400");
pinMode (ENA, OUTPUT); // pwm signal en a     right tires
pinMode (ENB, OUTPUT); // pwm signal en b     left tires
pinMode (IN1, OUTPUT);   //in1 left tires
pinMode (IN2, OUTPUT);   //in2 left tires
pinMode (IN3, OUTPUT);   //in3 right tires
pinMode (IN4, OUTPUT);   //in4 right tires



}
```

Not much needed to be explained in the setup function… there are just some initializations.

Loop()

```
void loop() {
  while (btSerial.available()) {

      value = btSerial.read();

      if(value > 6){
        flex_speed = value;
      }else if(value <= 6 && value >= 0){
        command = value;
      }

      Serial.print("Received cmd: ");
      Serial.print(command);
      Serial.print(" | speed: ");
      Serial.println(flex_speed);

  }

  switch (command) {
    case FORWARD_LEFT:    left();     break;
    case FORWARD_RIGHT:   right();    break;
    case BACKWARD_LEFT:   left();     break;
    case BACKWARD_RIGHT:  right();    break;
    case FORWARD:         forward();  break;
    case BACKWARD:        back();     break;
    case STOP:            stop();     break;
    default: break;
  }
}
```

Firstly.. Value takes whatever sent to it.. We check if the value is larger than 6 then it is a flex sensor value.. If not then the other condtion is true and it's a command! (command macros are from 0-6)

After storing the command we check the command in a switch statement.. And execute the corresponding command.

Movement functions

```
void left()
{
  digitalWrite (IN1 , HIGH);
  digitalWrite (IN2 , LOW);
  digitalWrite (IN3 , LOW);
  digitalWrite (IN4 , HIGH);
  analogWrite (ENA , flex_speed );
  analogWrite (ENB , flex_speed );
}


void right()
{
  digitalWrite (IN1 , LOW);
  digitalWrite (IN2 , HIGH);
  digitalWrite (IN3 , HIGH);
  digitalWrite (IN4 , LOW);
  analogWrite (ENA , flex_speed );
  analogWrite (ENB , flex_speed );
}

void stop()
{
  digitalWrite (IN1 , LOW);
  digitalWrite (IN2 , LOW);
  digitalWrite (IN3 , LOW);
  digitalWrite (IN4 , LOW);
  analogWrite (ENA , flex_speed );
  analogWrite (ENB , flex_speed );
}
```

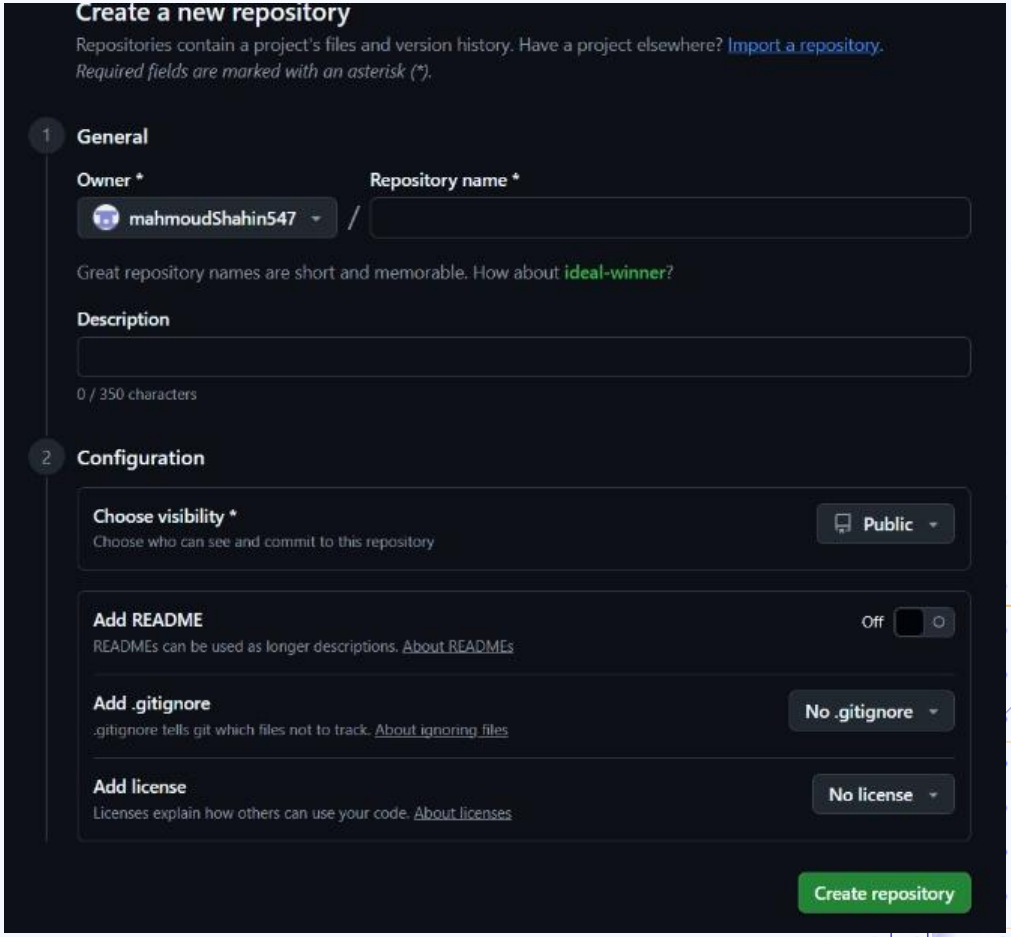
```
void forward()
{
  digitalWrite (IN1 , LOW);
  digitalWrite (IN2 , HIGH);
  digitalWrite (IN3 , LOW);
  digitalWrite (IN4 , HIGH);
  analogWrite (ENA , flex_speed );
  analogWrite (ENB , flex_speed );
}

void back()
{
  digitalWrite (IN1 , HIGH);
  digitalWrite (IN2 , LOW);
  digitalWrite (IN3 , HIGH);
  digitalWrite (IN4 , LOW);
  analogWrite (ENA , flex_speed );
  analogWrite (ENB , flex_speed );
}
```

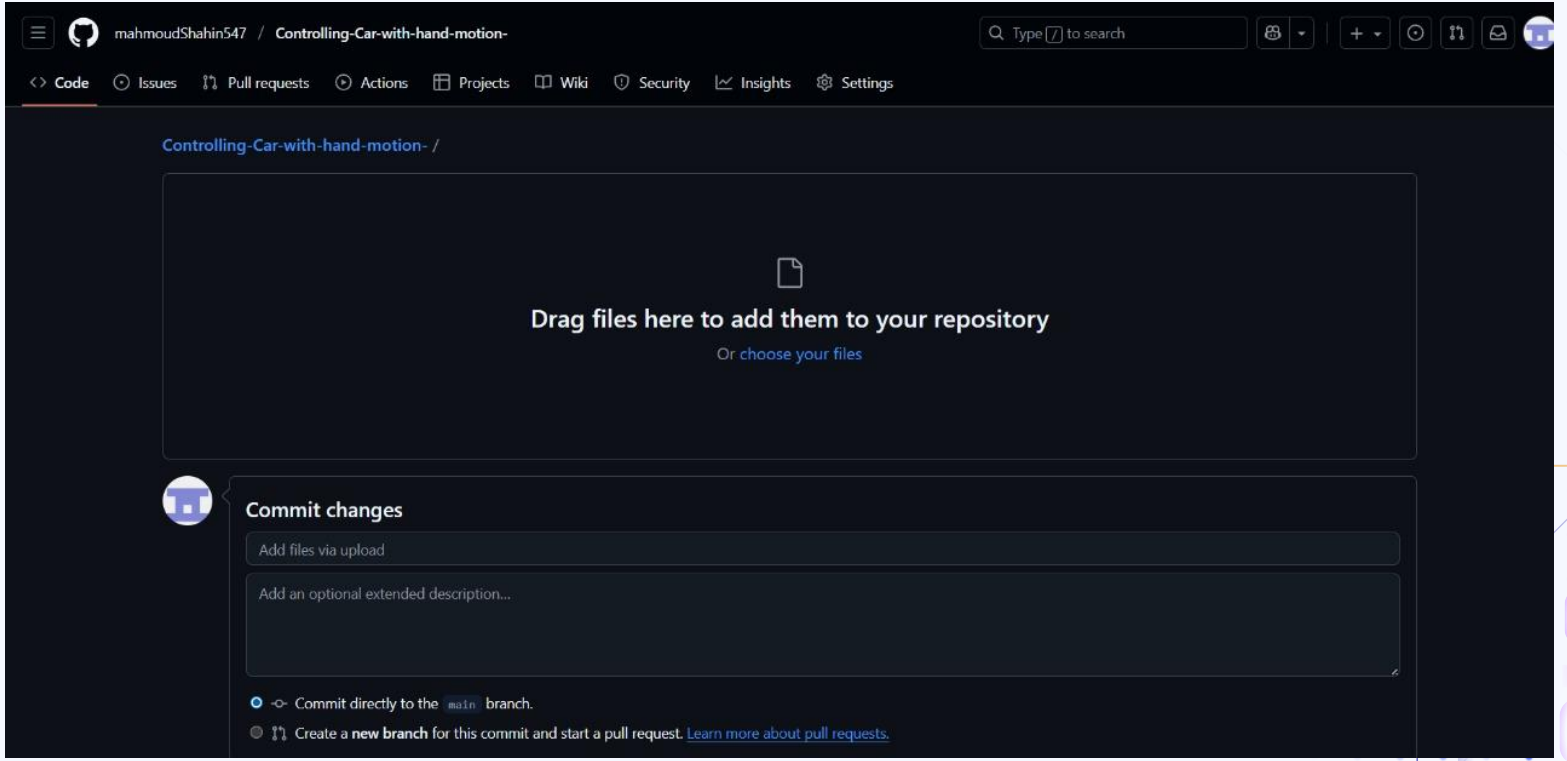
These are the functions that makes controls the direction and speed of the car motors.

Git Hub

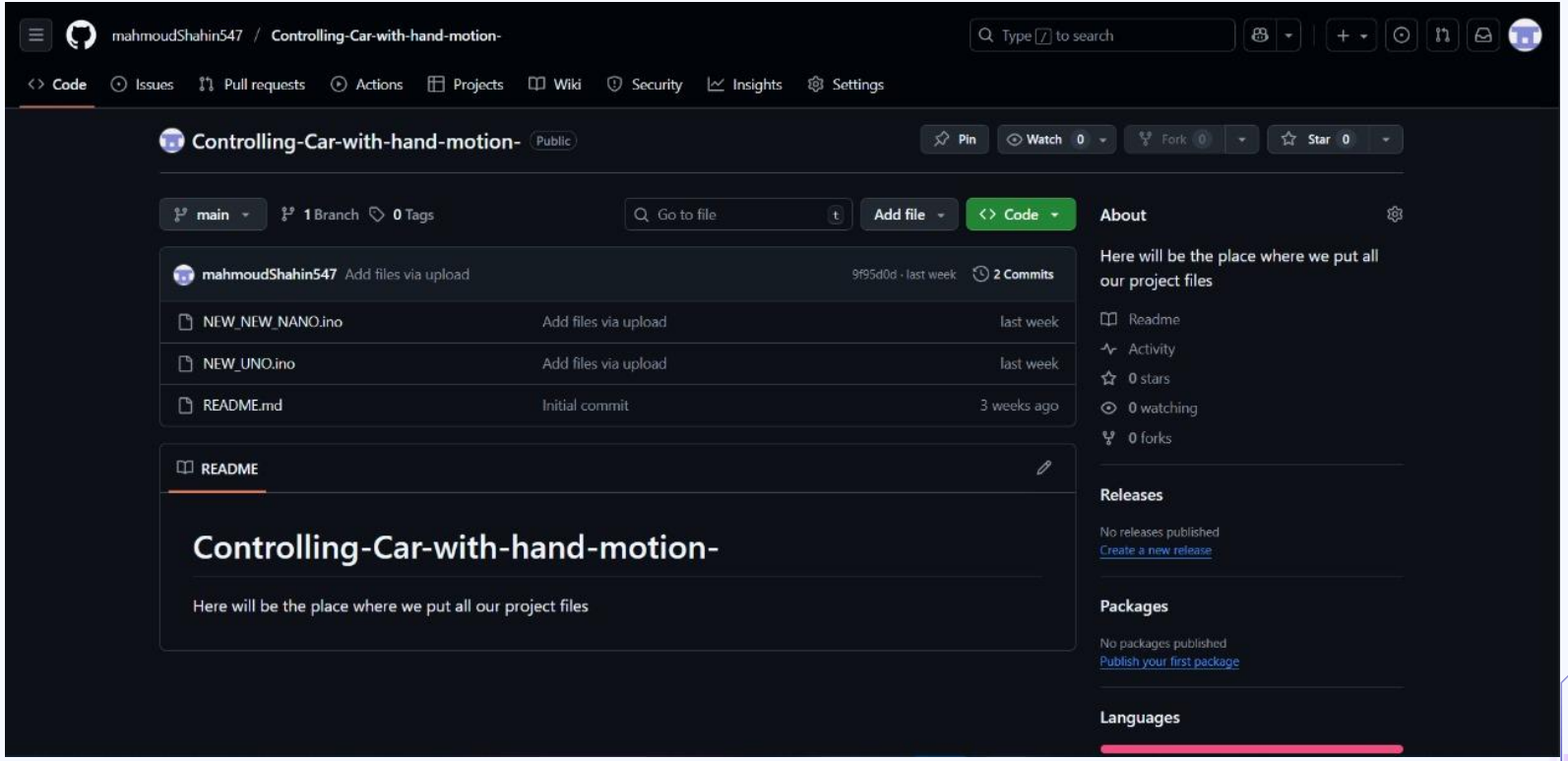
First of all, we had to create a new repository so that we can put our files and work in , so we got into the GitHub.com site , and had the following page loaded up where we got to put our project name (Repository name), the description of the project, and chose its visibility to be public so that everyone would have the access to the repository



Then we had to upload our files to the repository so we dragged them after we finished them and uploaded them here.



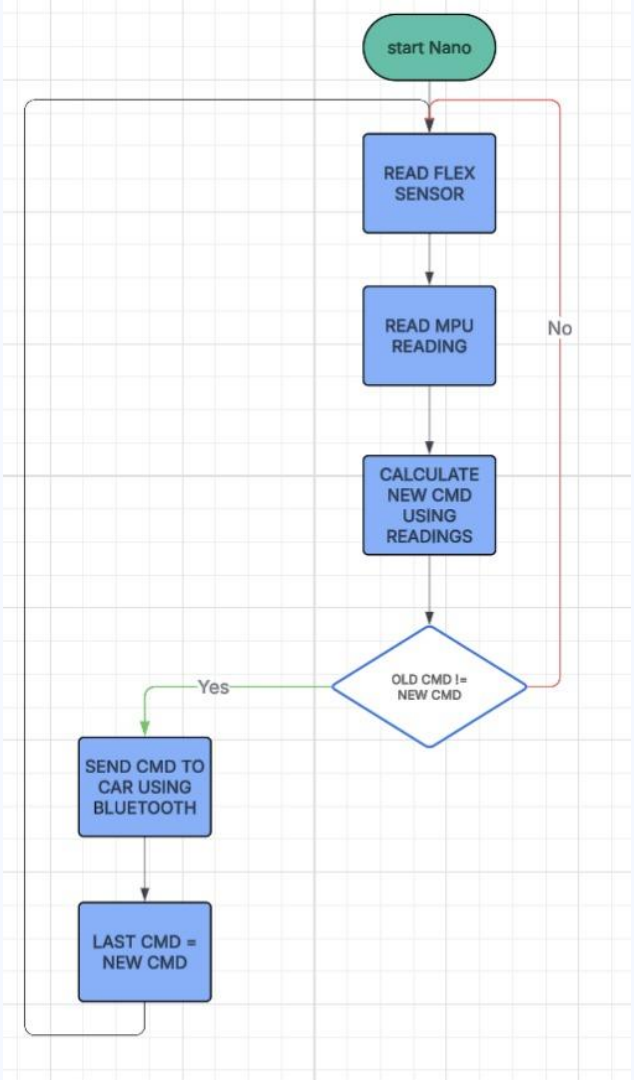
and last but not least this was our repository after uploading our files.

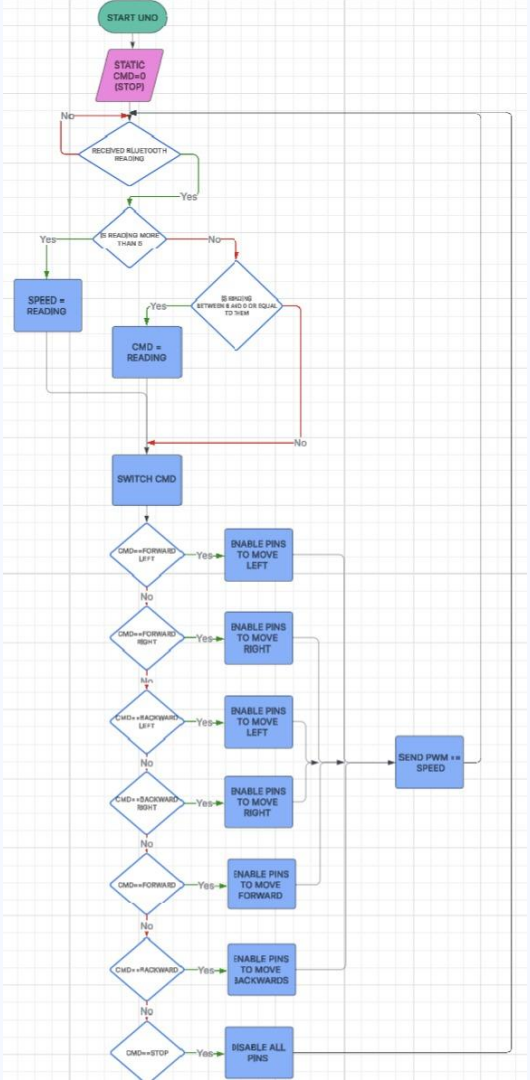


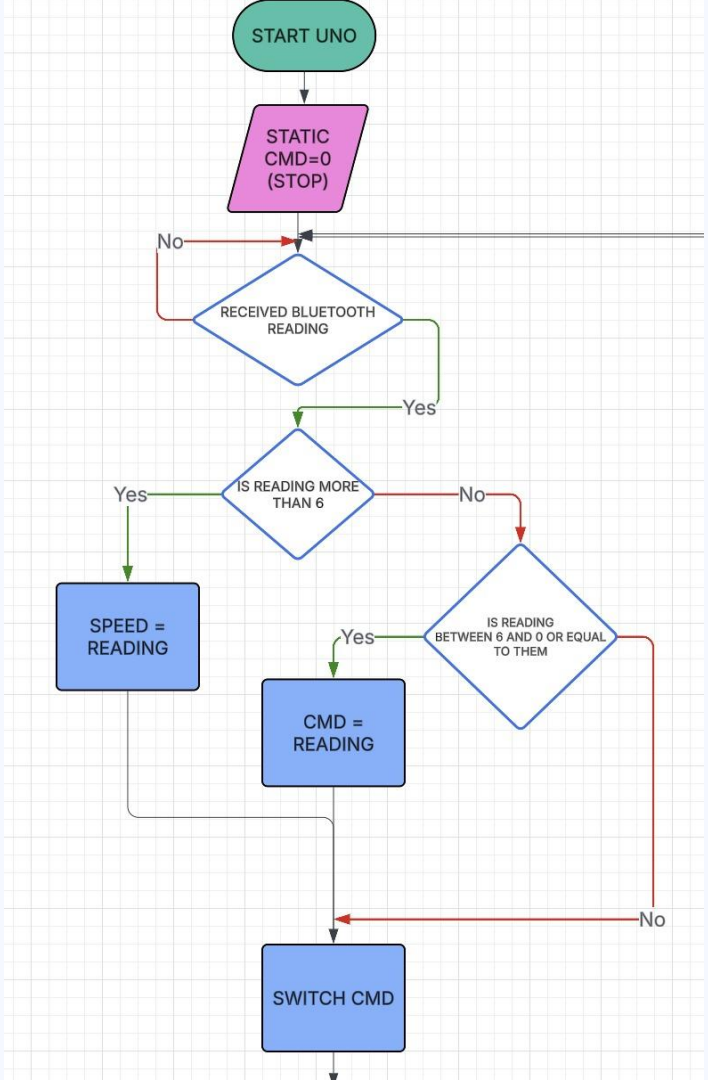
Github link

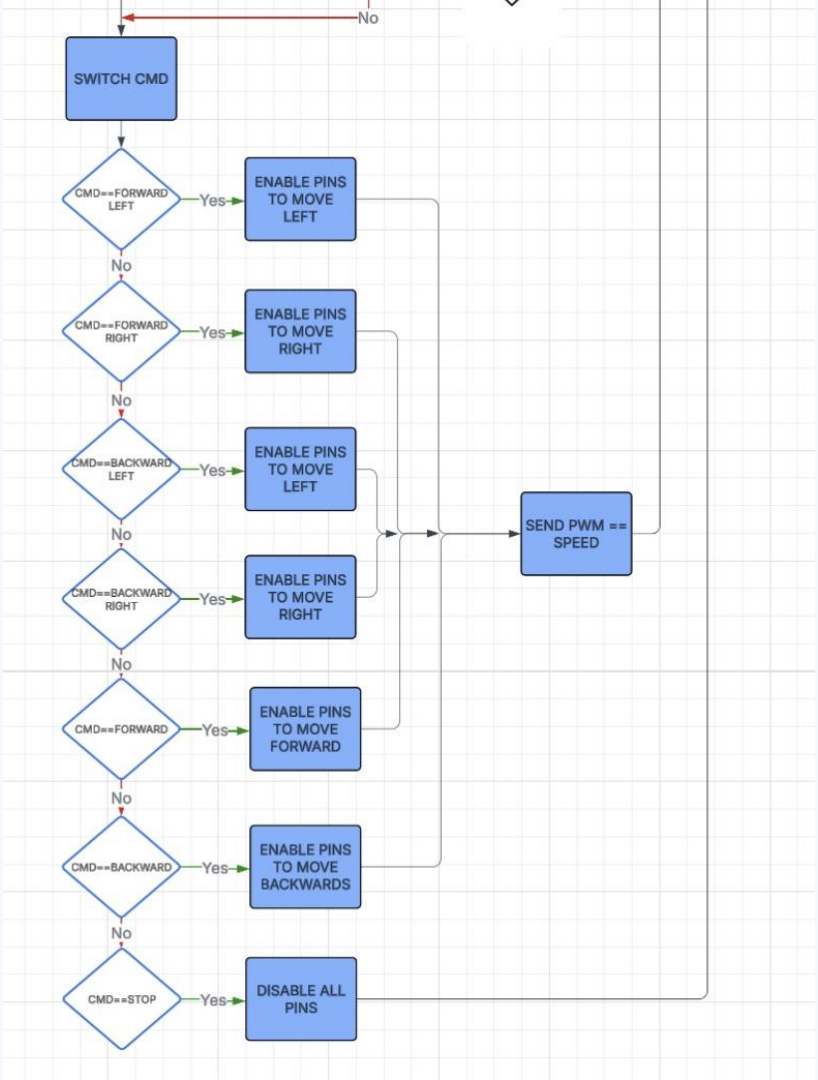
https://github.com/mahmoudShahin547/Controlling-Car-with-hand-motion-

Flowchart


start Nano
READ FLEX SENSOR
READ MPU READING
CALCULATE NEW CMD USING READINGS
OLD CMD != NEW CMD
No
Yes
SEND CMD TO CAR USING BLUETOOTH
LAST CMD = NEW CMD


START UNO
STATIC CMD=0 (STOP)
No
RECEIVED BLUETOOTH READING
Yes
IS READING MORE THAN 5
Yes
No
SPEED = READING
IS READING BETWEEN 6 AND 0 OR EQUAL TO THEM
Yes
CMD = READING
No
SWITCH CMD
CMD==FORWARD LEFT
Yes
ENABLE PINS TO MOVE LEFT
No
CMD==FORWARD RIGHT
Yes
ENABLE PINS TO MOVE RIGHT
No
CMD==BACKWARD LEFT
Yes
ENABLE PINS TO MOVE LEFT
No
CMD==BACKWARD RIGHT
Yes
ENABLE PINS TO MOVE RIGHT
No
CMD==FORWARD
Yes
ENABLE PINS TO MOVE FORWARD
No
CMD==BACKWARD
Yes
ENABLE PINS TO MOVE BACKWARDS
No
CMD==STOP
Yes
DISABLE ALL PINS
SEND PWM == SPEED


START UNO
STATIC CMD=0 (STOP)
No
RECEIVED BLUETOOTH READING
Yes
IS READING MORE THAN 6
Yes
No
SPEED = READING
IS READING BETWEEN 6 AND 0 OR EQUAL TO THEM
Yes
CMD = READING
No
SWITCH CMD


No
SWITCH CMD
CMD==FORWARD LEFT
Yes
ENABLE PINS TO MOVE LEFT
No
CMD==FORWARD RIGHT
Yes
ENABLE PINS TO MOVE RIGHT
No
CMD==BACKWARD LEFT
Yes
ENABLE PINS TO MOVE LEFT
No
CMD==BACKWARD RIGHT
Yes
ENABLE PINS TO MOVE RIGHT
No
CMD==FORWARD
Yes
ENABLE PINS TO MOVE FORWARD
No
CMD==BACKWARD
Yes
ENABLE PINS TO MOVE BACKWARDS
No
CMD==STOP
Yes
DISABLE ALL PINS
SEND PWM == SPEED

Team Members

FARES SAKR

Yahia Nabil

Mahmoud Shahin

Yousef Assar

Basant Hagag

THANKS