

**CARZ**

Team member: academic year:

1)Amer Fawzy term 5

2)Mahmoud Hamdy term 5

**Content:**

**1)abstract**

**2)purchased components**

**3)circuit designs**

**4)code**

**5)references and apps used**

Abstract :

The main idea of the controls of the motion of the robot car is MPU6050 to sense the orientation of the glove on your wrist and transmits it to the Arduino in digital value. The value range is from -40 to +30 for each axis. Module based on the NRF24L01 chip having two-way communication on the 2.4GHz band. (1)The circuit board has a built-in antenna. (2)The module communicates with microcontrollers via SPI reference. The range of this module in the theory is up to 100 meters. Also you can control the power of the transmitter to reduce power consumption. The motors are controlled by the L298N module powered by 2x 18650 Batteries.

Components Purchased:

i)Arduino nano

ii)nRF24L01+RFmodule

iii)L298 motor driver

iv)IMU(mpu6050)

v)TT-gear motor

vi)heavy rubber wheels

vii)18650 LI-ion cell 2500 MAH

viii)Battery holder

ix)nano shield

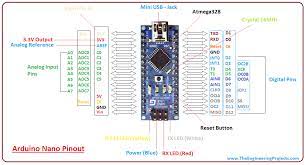
x)line follower

xi)acrylic robot

Components:

1)Arduino nano:

Small size, Cheapest microcontroller, based on the ATmega328 and  This will allow you to make larger programs than with the ArduinoUno (it has 50% more program memory), and with a lot more variables (the RAM is 200% bigger).



2)NRF24L01+RF Module: (SPI or 2.4MHz)

Low coast communication tiny and The nRF24L01+ transceiver module communicates over a 4-pin Serial Peripheral Interface (**SPI**) with a maximum data rate of **10Mbps and under max frequency 2.4MHz.** The SPI bus uses a concept of a MASTER and a SLAVE

#Master(Arduino nano):control SPI transmission, generates the clock and controls the slave select signal

\*CSN(chip signal not): enable pin for SPI bus and it’s active low, we always want to keep that pin high

except when we are sending a command or a data it is low to start listening on the SPI.

**\*SS**(Slave Select):Generated by the master to control which slave to talk to. It’s usually an active-low signal

.

SCK : clock generated by master go to slave

**#slave**

\*MOSI (Master Out Slave In):is SPI input to the nRF24L01.

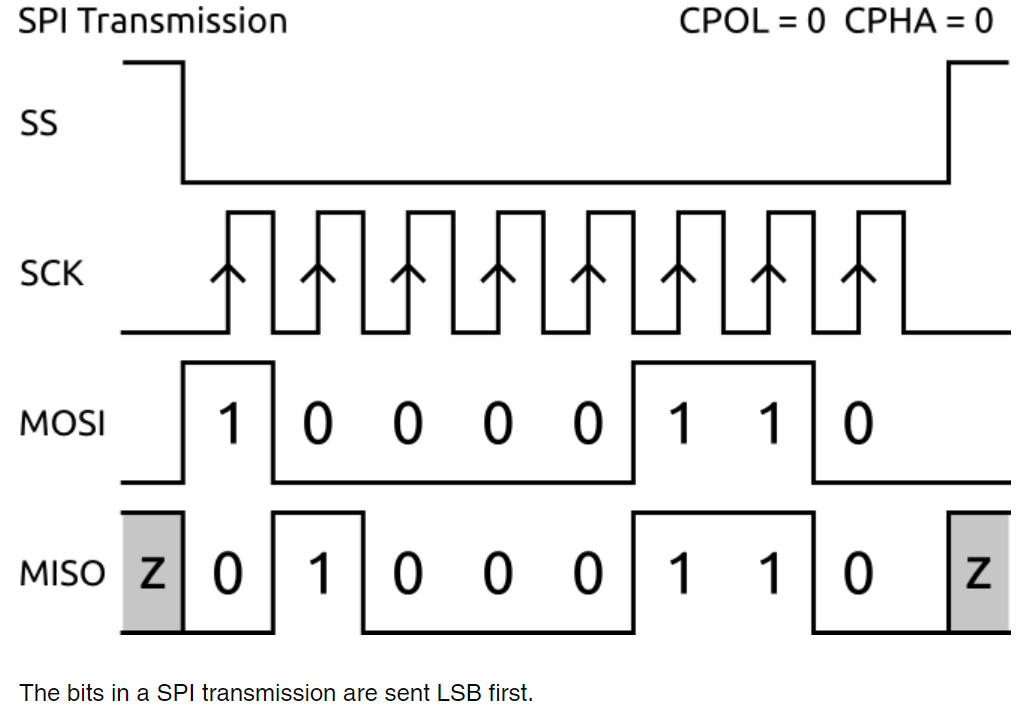
\*MISO (Master In Slave Out):is SPI output from the nRF2Ll01.

Note:

master SPI pull the **SS** signal low for the slave want to communicate with after using the **CSN** is the enable pin for SPI bus . The reference of 4 SPI bus standards is **SCK.** These 4 are branched into 2 **CPOL**(clock polarity) high or low and **CPHA**(clock phase) raising or falling **CPOL**=0 (low) and **CPHA**=0 (rising edge). SPI transfer is **full-duplex** but the slave cant pass a data to master without its request done by **MISO**(master send to slave) and **MOSI**(slave send to master) the backward function. The master kept **SS** low(raising edge are active) and toggling **SCK** master would send the command  at a certain address, Once **SS** returned high that is the end of reading data.

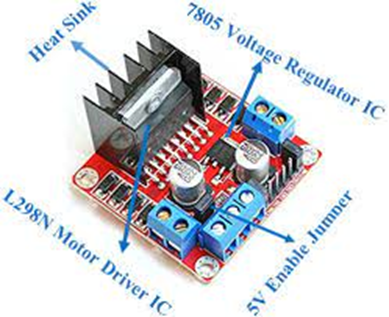
|  |  |  |
| --- | --- | --- |
| **SPI Mode** | **CPOL** | **CPHA** |
| 0 [00] | 0 | 0 |
| 1 [01] | 0 | 1 |
| 2 [10] | 1 | 0 |
| 3  [11] | 1 | 1 |





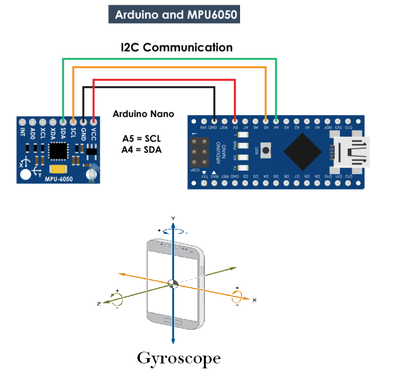
3)L298motor driver:

control both speed (PWM) and rotation Direction (H-Bridge). The module can drive 2 DC motors that have voltages between 5 and 35V, with a peak current up to 2A.

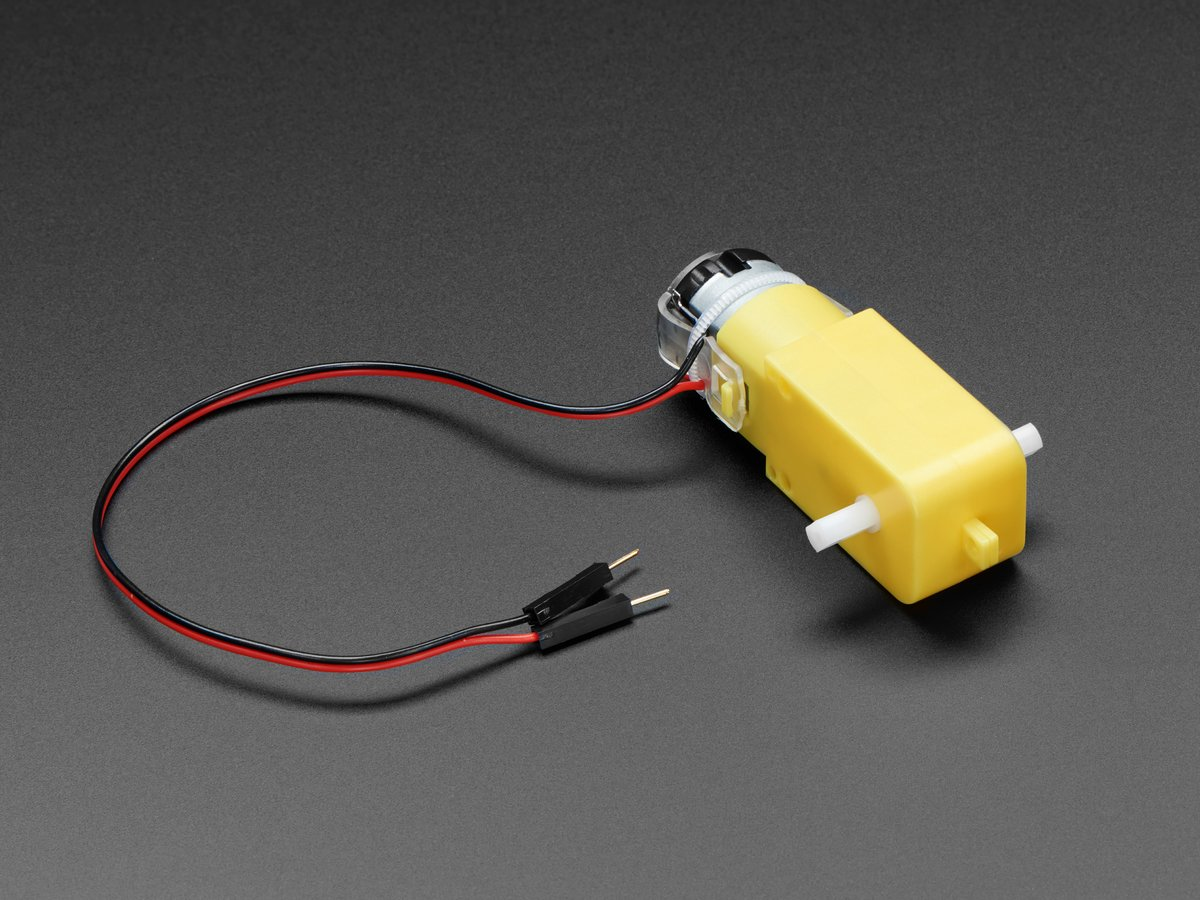


4)IMU(mpu6050):

3-Axis gyroscope integrated on a single chip. MPU6050 accelerometer measures acceleration by measuring gravitational acceleration along the 3 axes and using some trigonometry math we can calculate the angle at which the sensor is positioned so we can get very accurate information about the sensor orientation.



5)TT-Gear motors:

easy, low-cost way to get your projects moving. gearbox reduce the motor shaft speed (speed of rotation), and increase the torque output of the motor And that we need for our Application. 

6)Heavy Rubber wheels:

rubbertires make it easier to hold and stay stable and has low cost

.

7)18650 Li-ion battery 2500 MAH:

 its 18mm x 65mm size, rechargeable battery, This type of battery supply 4.2 Volt each.

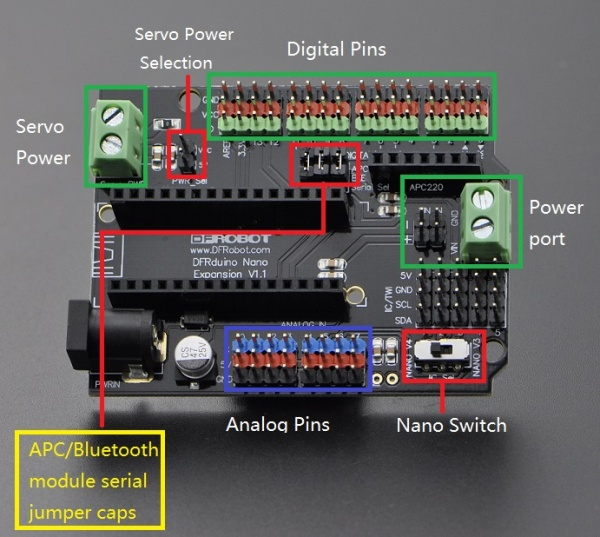
8)battery Holder:

The primary function of this holder is to keep batteries fixed in place safely



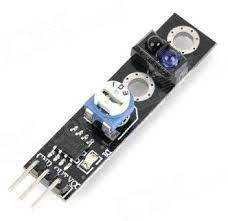
9)nano shield:

make connection to Arduino nano very easy as it separate each pin to pin id with + - that make the connection really easy for me . also I used it instead of making PCB because it is complex process.



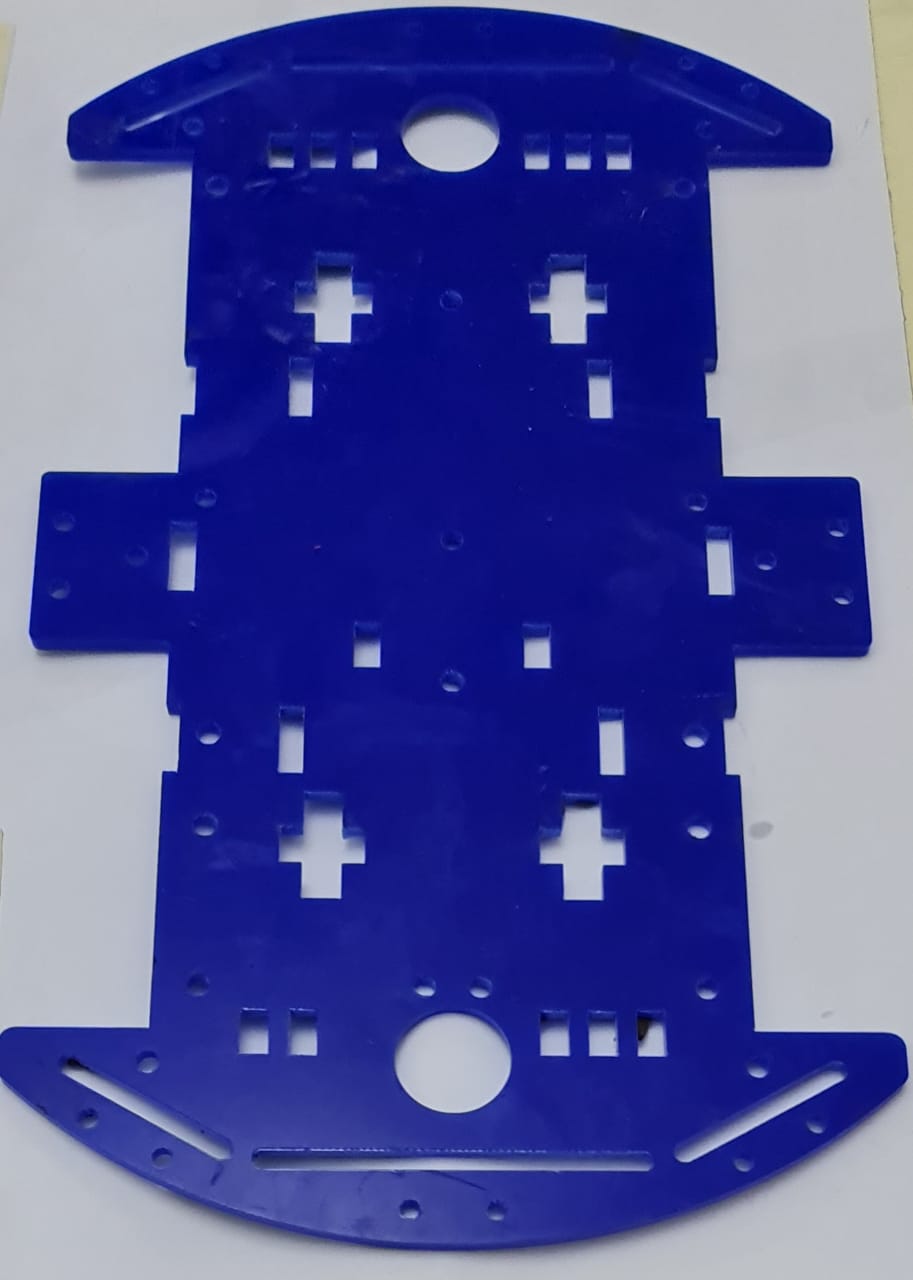
10) line follower sensor:

 works by detecting reflected light coming from the infrared led. When infrared rays emitted when not being reflected back so its shutdown state and output module is high. When it detect an object and detect range of the infrared is reflected back, so the output is low.

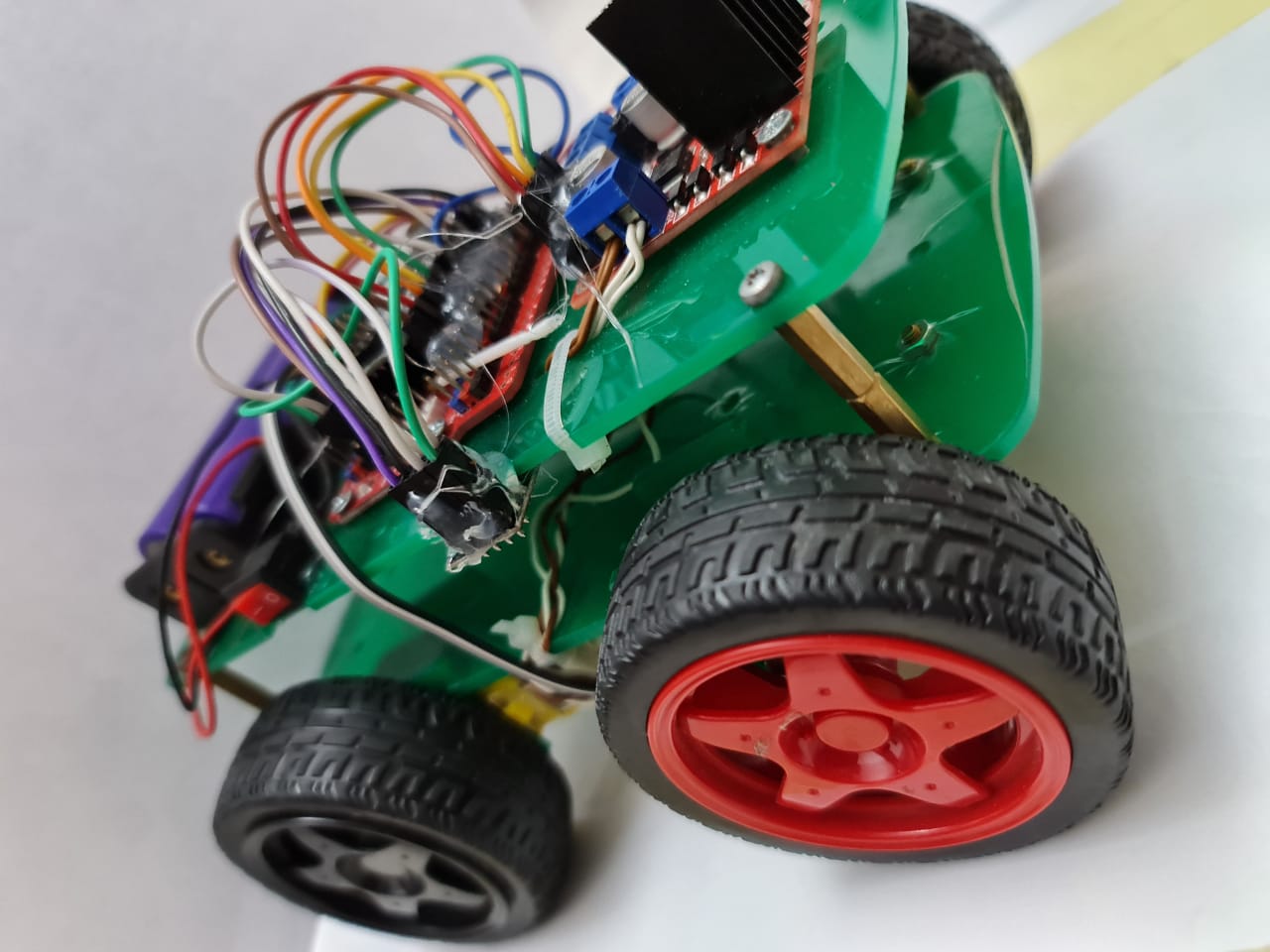
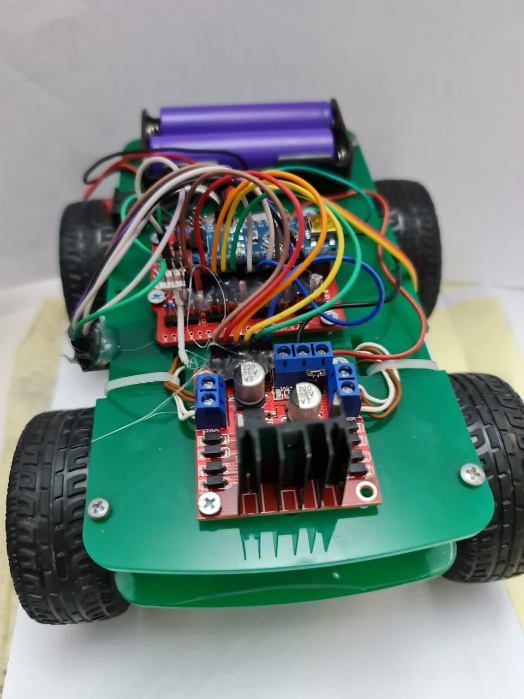


11)acrylic robot:

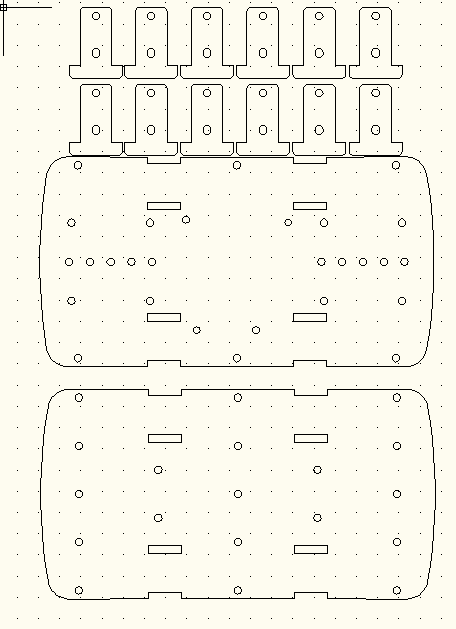
the market product was so much long(26x15) for super tiny components i chose.



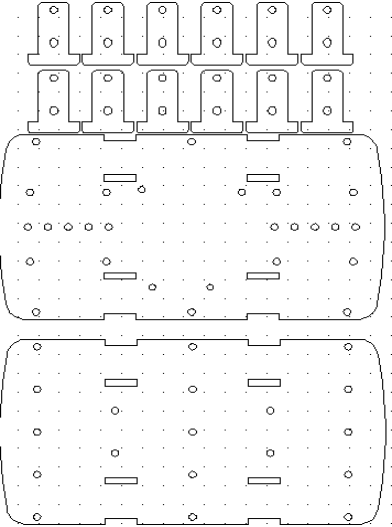
So, I tried to reduce that length to be (9x17cm) suitable for the components, easier in motion and smaller in size.

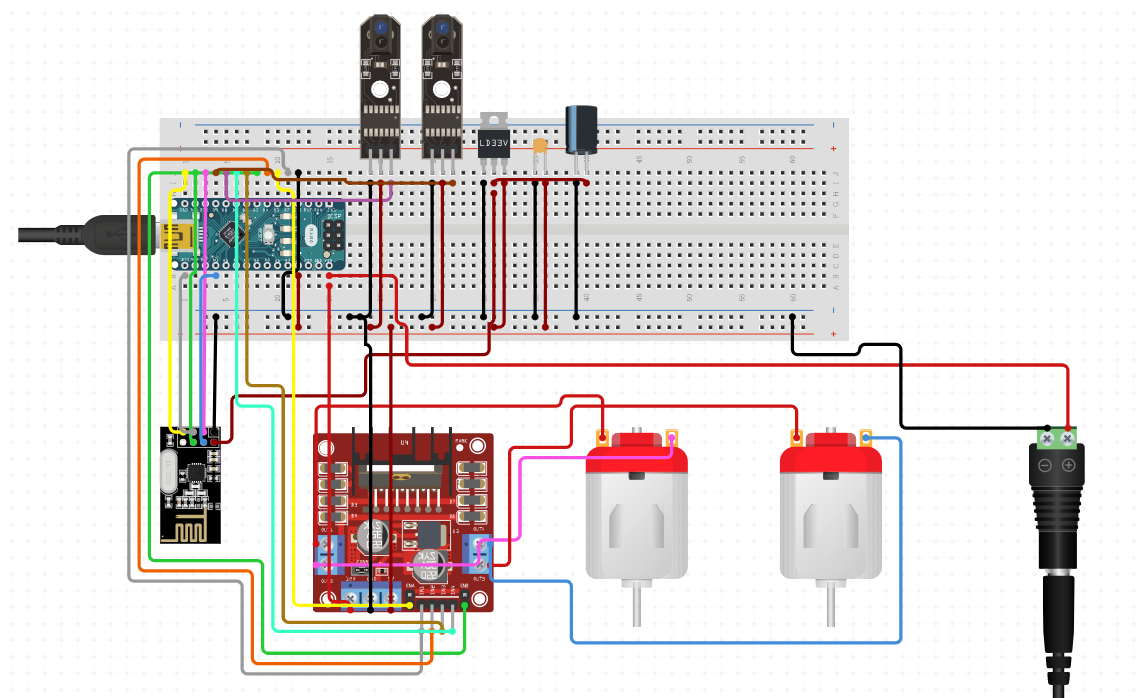
 

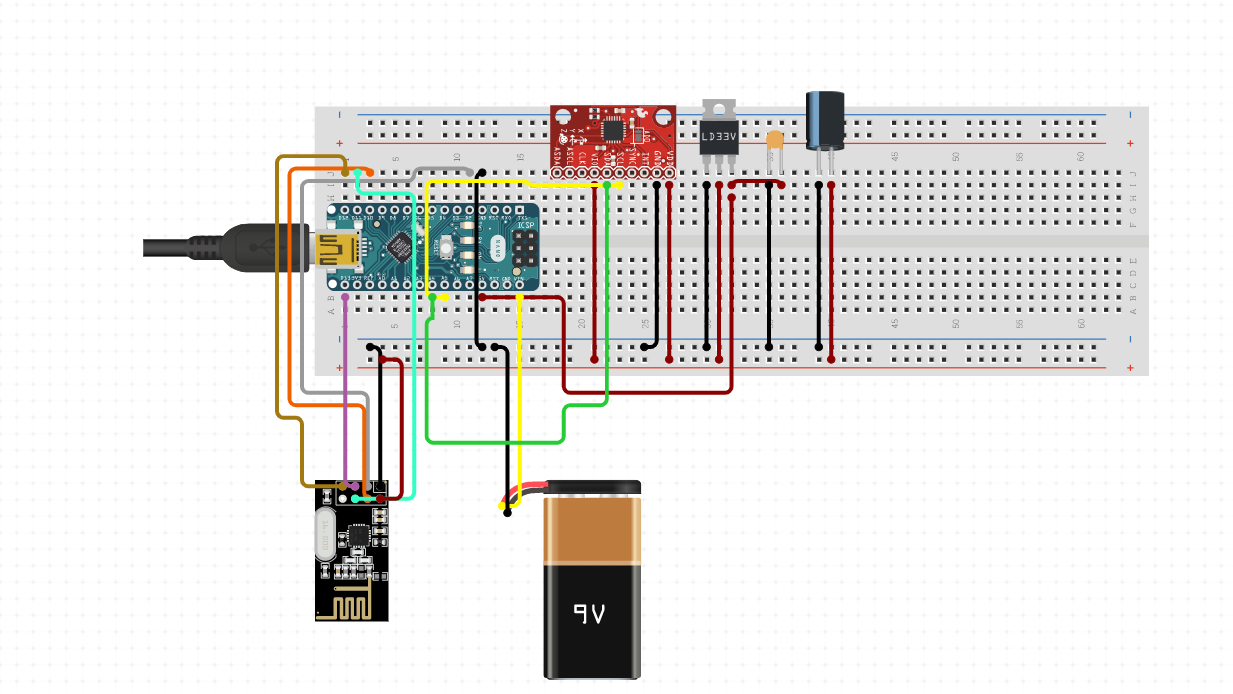
**Model:**



**Layout:**



#recieving circuit

#Transmitting Circuit

**##RECRIVING\_CODE**

#include<SPI.h>

#include<nRF24L01.h>

#include<RF24.h>

int ENA = 3;

int ENB = 9;

int MotorA1 = 4;

int MotorA2 = 5;

int MotorB1 = 6;

int MotorB2 = 7;

int LeftSensor = A0;

int RightSensor = A1;

int fastSpeed = 120, lowSpeed = 110;

RF24 radio(8, 10); // CE, CSN

const byte address[6] = "00001"; //array address through which the two modules will communicate.

struct data {

int xAxis;

int yAxis;

};

data receive\_data;

void setup() {

Serial.begin(9600);

radio.begin();

radio.openReadingPipe(0, address); //function communication between two modules

radio.setPALevel(RF24\_PA\_MIN);

radio.setDataRate(RF24\_250KBPS);

radio.startListening(); //function which sets module as receiver

pinMode(ENA, OUTPUT);

pinMode(ENB, OUTPUT);

pinMode(MotorA1, OUTPUT);

pinMode(MotorA2, OUTPUT);

pinMode(MotorB1, OUTPUT);

pinMode(MotorB2, OUTPUT);

pinMode(LeftSensor, INPUT);

pinMode(RightSensor, INPUT);

}

void loop() {

Serial.print("RS: ");

Serial.print(digitalRead(RightSensor));

Serial.print(" LS: ");

Serial.println(digitalRead(LeftSensor));

while (radio.available()) {

radio.read(&receive\_data, sizeof(data));

Serial.print("X: ");

Serial.print(receive\_data.xAxis);

Serial.print(" | Y: ");

Serial.println(receive\_data.yAxis);

if (receive\_data.yAxis > 25) { //Move Forward

if (digitalRead(LeftSensor) == true && digitalRead(RightSensor) == false) { //Move Right

left();

analogWrite(ENA, lowSpeed);

analogWrite(ENB, lowSpeed);

}

else if (digitalRead(LeftSensor) == false && digitalRead(RightSensor) == true) { //Move Right

right();

analogWrite(ENA, lowSpeed);

analogWrite(ENB, lowSpeed);

}

else {

forward();

analogWrite(ENA, lowSpeed);

analogWrite(ENB, lowSpeed);

}

}

else if (receive\_data.yAxis < -40) { //Move backward

backward();

analogWrite(ENA, fastSpeed);

analogWrite(ENB, fastSpeed);

}

else if (receive\_data.xAxis < -40) { //Move Right

right();

analogWrite(ENA, fastSpeed);

analogWrite(ENB, fastSpeed);

} else if (receive\_data.xAxis > 30) { //Move Left

left();

analogWrite(ENA, fastSpeed);

analogWrite(ENB, fastSpeed);

}

else {

stopp();

analogWrite(ENA, 0);

analogWrite(ENB, 0);

}

delay(50);

}

}

void forward() {

digitalWrite(MotorA1, HIGH);

digitalWrite(MotorA2, LOW);

digitalWrite(MotorB1, LOW);

digitalWrite(MotorB2, HIGH);

}

void backward() {

digitalWrite(MotorA1, LOW);

digitalWrite(MotorA2, HIGH);

digitalWrite(MotorB1, HIGH);

digitalWrite(MotorB2, LOW);

}

void left() {

digitalWrite(MotorA1, HIGH);

digitalWrite(MotorA2, LOW);

digitalWrite(MotorB1, HIGH);

digitalWrite(MotorB2, LOW);

}

void right() {

digitalWrite(MotorA1, LOW);

digitalWrite(MotorA2, HIGH);

digitalWrite(MotorB1, LOW);

digitalWrite(MotorB2, HIGH);

}

void stopp() {

digitalWrite(MotorA1, LOW);

digitalWrite(MotorA2, LOW);

digitalWrite(MotorB1, LOW);

digitalWrite(MotorB2, LOW);

}

**##TRANSMITTING\_CODE**

**#include<SPI.h>**

**#include<nRF24L01.h>**

**#include<RF24.h>**

**#include <Wire.h>**

**#include <MPU6050.h>**

**MPU6050 mpu;**

**RF24 radio(8, 10); //ce,csn**

**const byte address[6] = "00001";**

**struct data {**

**int xAxis;**

**int yAxis;**

**};**

**data send\_data;**

**void setup() {**

**radio.begin();**

**radio.openWritingPipe(address);**

**radio.setPALevel(RF24\_PA\_MIN);**

**radio.setDataRate(RF24\_250KBPS);**

**radio.stopListening();**

**Serial.begin(9600);**

**Serial.println("Initialize MPU6050");**

**while (!mpu.begin(MPU6050\_SCALE\_2000DPS, MPU6050\_RANGE\_2G))**

**{**

**Serial.println("Could not find a valid MPU6050 sensor, check wiring!");**

**delay(500);**

**}**

**}**

**void loop() {**

**// Read normalized values**

**Vector normAccel = mpu.readNormalizeAccel();**

**// Calculate Pitch & Roll**

**int roll = -(atan2(normAccel.XAxis, sqrt(normAccel.YAxis \* normAccel.YAxis + normAccel.ZAxis \* normAccel.ZAxis)) \* 180.0) / M\_PI;**

**int pitch = (atan2(normAccel.YAxis, normAccel.ZAxis) \* 180.0) / M\_PI;**

**send\_data.xAxis = roll;**

**send\_data.yAxis = pitch;**

**Serial.print(" \nPitch = ");**

**Serial.print(pitch);**

**Serial.print(" Roll = ");**

**Serial.print(roll);**

**radio.write(&send\_data, sizeof(data));**

**}**

**#acrylic sheet design done using:**

autoCAD

**#references:**

[***https://howtomechatronics.com/***](https://howtomechatronics.com/)

***https://www.instructables.com/circuits/***

***https://lastminuteengineers.com/***

[***https://github.com/jarzebski/Arduino-MPU6050***](https://github.com/jarzebski/Arduino-MPU6050)

[*https://www.circuito.io/*](https://www.circuito.io/)*.*

[***https://github.com/***](https://github.com/)

***https://deepbluembedded.com/***