

Hand Gesture Control Car

FSSI-BEE Project.

29.01.2022

By

Dhruv Vyas.

Aryavardhan Sharma.

Deep Bhalsod.

Kushal Munjal.

Aim

To make a Hand gesture control car by using different sensors and modules.

Components

- Arduino UNO R3.
- 433 MHz RF Module.
- ADXL335 Accelerometer.
- L293D Motor Driver Module.
- Chassis.
- Motors.
- Wires.
- PCB Board.
- Arduino IDE.
- PC/Laptop.

Theory(About The Project)

As the name suggests , We will make a Hand Gesture Control Car. By moving our hand In different directions we will be able to move the car according to the hand gestures Like Forward, Backward, Left and Right.

Hand Directions To Control the Car:-

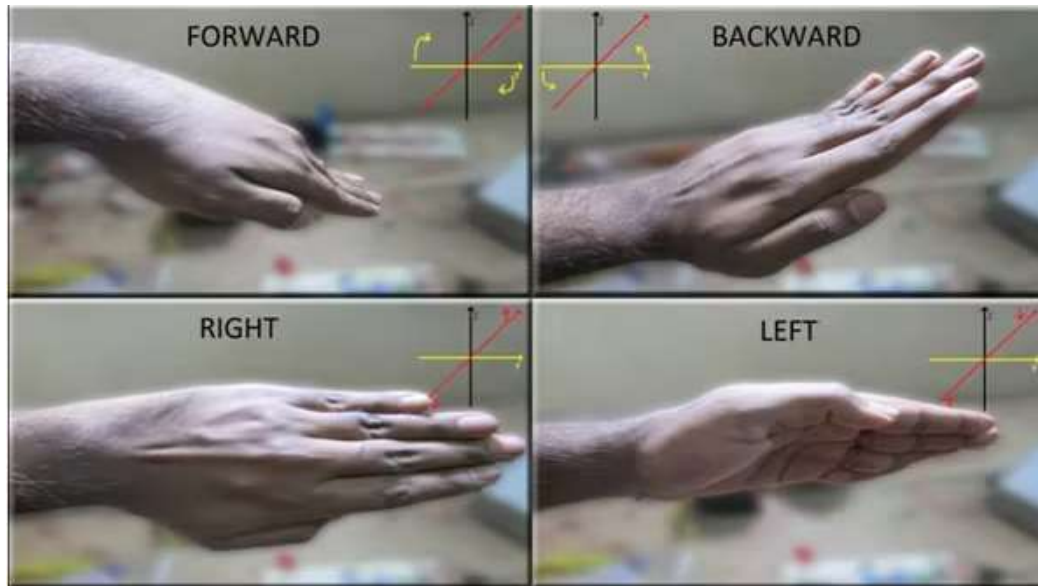
- Upward Direction For Moving Backward.
- Downward Direction For Moving Forward.
- Right Direction For Moving Towards Right.
- Left Direction For Moving Towards Left.

The gesture control robot is one of the most common types of projects made by many students to understand and implement microcontroller knowledge in a physical and practical project. The concept behind it is simple: the orientation of the palm controls the motion of the robot car. How is it possible?

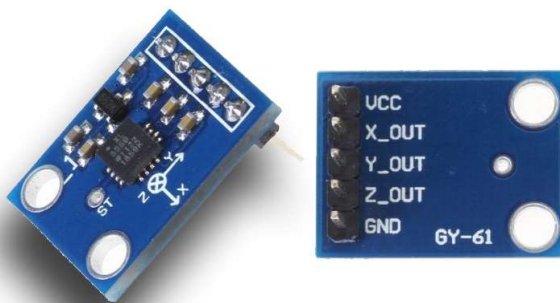
Let's understand that,we will move by understanding the role and function of each component and then combining them to achieve OUR AIM.

1.ADXL335 (Accelerometer)

The function of the accelerometer is quite simple : to sense the gesture made by hand of the wrist. The accelerometer measures acceleration including the acceleration due to gravity 'g' as well. Thus we can use the accelerometer to sense the hand gestures of the wrist by measuring the component of 'g' in any particular axis of ADXL335 as shown in figure below :



The ADXL335 can measure up to 3g of acceleration and is interfaced with Arduino by connecting it's axis pins to Analog pins of Arduino. The accelerometer outputs voltage values proportional to acceleration.

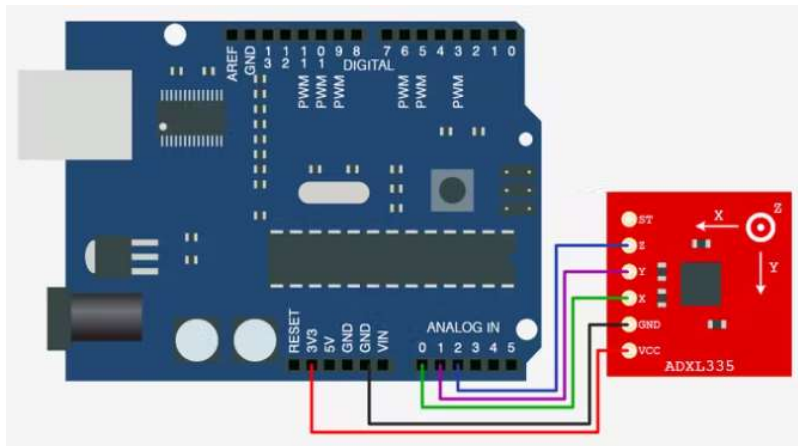


ElectronicWings.com

In this project, Accelerometer is connected to the Arduino Uno and is attached to the palm. The ADXL335 outputs voltage in range from 0 to Vcc (Applied voltage usually 3.3V) and is read by Arduino's Analog pins. Thus for the user, we get a value in range from 0 to 1024 (10-bit ADC). The different orientation yields a different

analog value for each axis which is then mapped to different robot movements.

The circuit diagram for the Accelerometer is:



Specifications:-

Operating Voltage	1.8V – 3.6V
Operating Current	350μA (typical)
Sensing Range	±3g (Full Scale)
Temperature Range	–40 to +85°C
Sensing axis	3 axis
Sensitivity	270 to 330mV/g (Ratiometric)
Shock Resistance	Up to 10,000g
Dimension	4mm x 4mm x 1.45mm

2. 433 MHz RF Transmitter and Receiver:

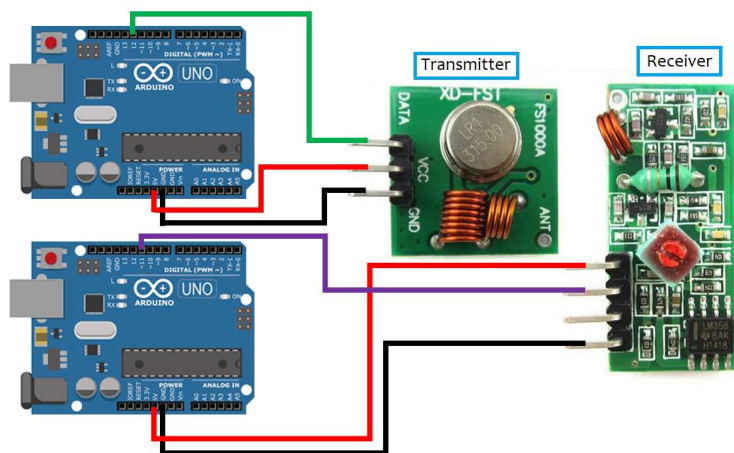
The function of the RF module is simple: to transmit command data from wrist Arduino Uno to the motor controlling Arduino Uno. The RF module uses Radio waves at 433hz frequency and thus the name RF-433. They use Amplitude Modulation to send data, but not getting into much technicalities and keeping things simple, they will be used to transmit commands to the robot, that is: move forward, backward, right or left. And in case of no data, stand still. They work well up to 10 meter range.



The receiver module receives the data in the form of a signal and sends it to the data pin. The data received by the module is always in an encoded form which can be decoded by either using the microcontroller or the decoder. The 433 MHz RF receiver module helps to receive the

433MHz frequency mainly. But it has a node also that can be changed by users for different frequencies. The variable node can be used to adjust the frequency from 315MHz to 433MHz. The receiver just receives the data but to decode and view this data, a microcontroller or decoder is required. The RF receiver module comprises an RF tuned circuit and a couple of Operational Amplifiers to amplify the received carrier wave from the transmitter. The amplified signal is then further fed to a PLL (Phase Lock Loop) later it is received by the decoder which decodes the output stream and gives better noise immunity.

Now to understand how to implement RF module in our project, let's work on the transmitter and receiver circuits sequentially.

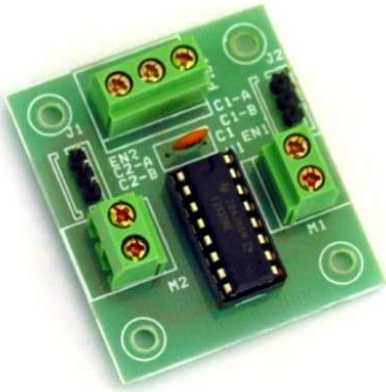


Specifications:-

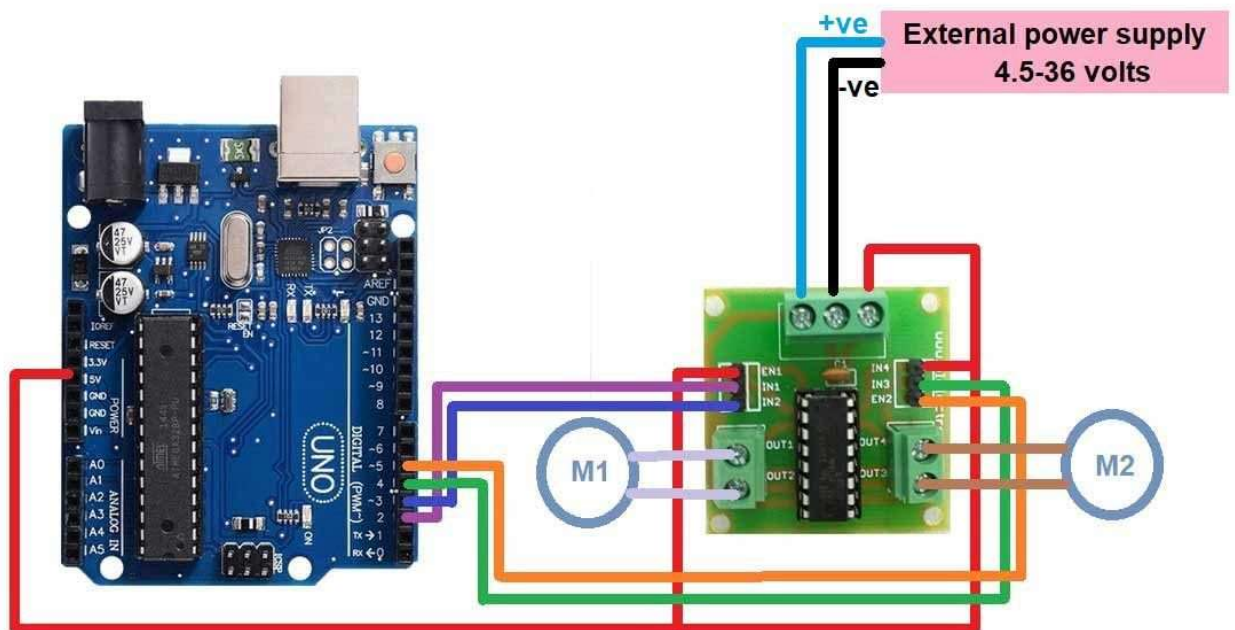
Max range with the antenna in normal Conditions	100 Meters
RX Receiver Frequency	433 MHz
RX Typical Sensitivity	105 Dbm
RX Supply Current	3.5 mA
RX IF Frequency	1MHz
RX Operating Voltage	5V
TX Frequency Range	433.92 MHz
TX Supply Voltage	3V ~ 6V

3.L293D Motor Driver Module

Using this L293D motor driver IC is very simple. As the name suggests it is mainly used to drive motors. A single **L293D IC** is capable of running two **DC motors** at the same time. The IC works on the principle of Half H-Bridge, let us not go too deep into what H-Bridge means, but for now just know that H bridge is a set up which is used to run motors both in clockwise and anti clockwise direction, also the direction of these two motors can be controlled independently. So for the motors which has operating voltage less than 36V and operating current less than 600mA, which are to be controlled by digital circuits like Op-Amp, digital gates or even Microcontrollers like Arduino, PIC, ARM etc..



The circuit diagram for the L293D Motor Driver Module is:



Specification:

Wide Supply-Voltage Range: 4.5 V to 36 V
Separate Input-Logic Supply
Internal ESD Protection
High-Noise-Immunity Inputs
Output Current 600 mA Per Channel
Peak Output Current 1.2 A Per Channel
Output Clamp Diodes for Inductive Transient Suppression
Operation Temperature 0°C to 70°C.
Automatic thermal shutdown is available

3.L298N Motor Driver Module

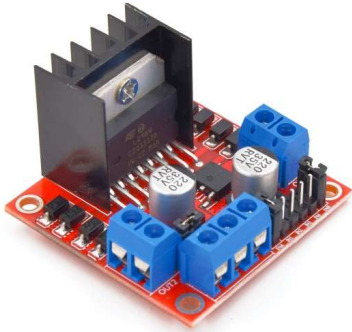
This L298N Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

Controlling a DC Motor

In order to have complete control over the DC motor, we have to control its speed and rotation direction. This can be achieved by combining these two techniques.

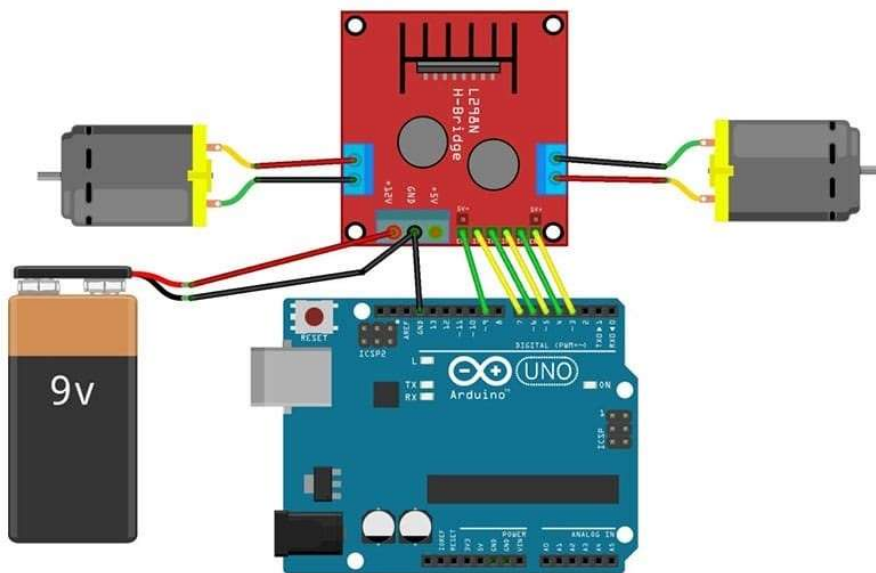
- PWM – For controlling speed
- H-Bridge – For controlling rotation direction

The heart of the module is the big, black chip with a chunky heat sink is an L298N.



The L298N is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors. That means it can individually drive up to two motors making it ideal for building two-wheel robot platforms.

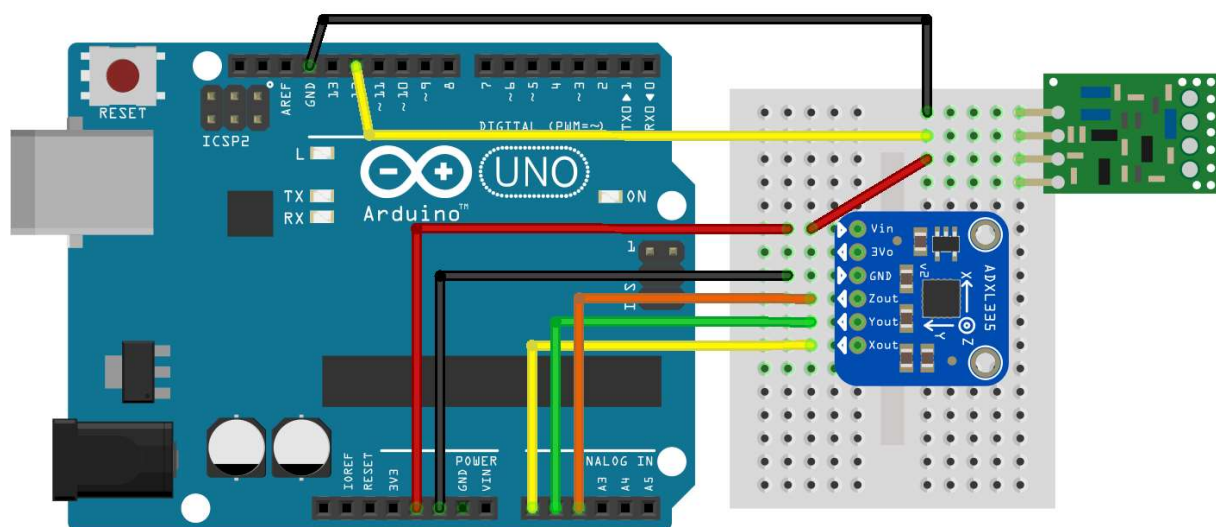
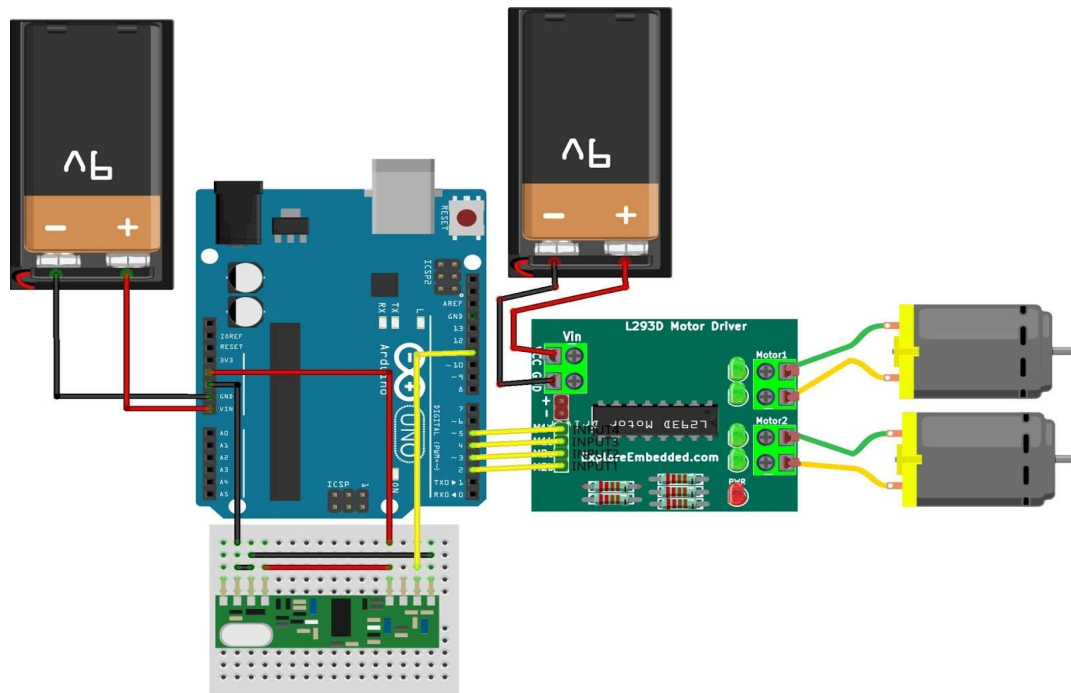
The circuit diagram for the L298N Motor Driver Module is:



L298 Module Features & Specifications:

Driver Model: L298N 2A
Driver Chip: Double H Bridge L298N
Motor Supply Voltage (Maximum): 46V
Motor Supply Current (Maximum): 2A
Logic Voltage: 5V
Driver Voltage: 5-35V
Driver Current: 2A
Logical Current: 0-36mA
Maximum Power (W): 25W
Current Sense for each motor
Heatsink for better performance
Power-On LED indicator

Interfacing Diagram(Schematic)



Code

TRANSMITTER CIRCUIT CODE:-

```
#include <VirtualWire.h>
```

```
//ADXL 335 pins which we have defined as the A0,A1,A2 pins.
```

```
#define x A0
```

```
#define y A1
```

```
#define z A2
```

```
//here we have defined all the variables which we have used ahead in the code
```

```
char *data;
```

```
int x_val;
```

```
int y_val;
```

```
int z_val;
```

```
int x_val2;
```

```
int y_val2;
```

```
int z_val2;
```

```
void setup()
```

```
{
```

```
  vw_set_tx_pin(12); //Declaring that RF transmitter digital pin is connected in pin  
  number 12
```

```
  vw_setup(2000); //Declaring communication speed and setting it to 2000 bits per  
  second
```

```
  pinMode(x, INPUT);
```

```
pinMode(y, INPUT); //Declaring that the pin x,y,z connected at A0,A1,A2 are input
pins
pinMode(z, INPUT);
Serial.begin(9600); //Declaring the Baud rate for serial communication
x_val2 = analogRead(x); //reading the value coming in pin A0 and storing it in
x_val2 only once for stable condition
y_val2 = analogRead(y); //reading the value coming in pin A1 and storing it in y_val2
only once for stable condition
z_val2 = analogRead(z); //reading the value coming in pin A2 and storing it in z_val2
only once for stable condition
}

void loop()
{
  x_val = analogRead(x); //reading the value coming in pin A0 and storing it in x_val
  everytime the loop runs
  y_val = analogRead(y); //reading the value coming in pin A1 and storing it in y_val
  everytime the loop runs
  z_val = analogRead(z); //reading the value coming in pin A2 and storing it in z_val
  everytime the loop runs

  int x_axis = x_val - x_val2; //Getting the difference of the (current reading-stable
  reading) storing it in x_axis
  int y_axis = y_val - y_val2; //Getting the difference of the (current reading-stable
  reading) storing it in y_axis
  int z_axis = z_val - z_val2; //Getting the difference of the (current reading-stable
  reading) storing it in z_axis

  if(y_axis >= 35)
  {
```

```
data="f";

vw_send((uint8_t *)data, strlen(data)); //Converting the data(f) into 8bit binary
data and then sending the data to RF receiver

vw_wait_tx(); //It will wait till the whole data is not sended
delay(500);

Serial.println("Forward"); //Printing forward in the Serial Monitor
}

else if(y_axis <= -35)
{
data="b";

vw_send((uint8_t *)data, strlen(data)); //Converting the data(b) into 8bit binary
data and then sending the data to RF receiver

vw_wait_tx();
delay(500);

Serial.println("Backward"); //Printing backward in the Serial Monitor
}

else if(x_axis >= 35)
{
data="r";

vw_send((uint8_t *)data, strlen(data)); //Converting the data(r) into 8bit binary
data and then sending the data to RF receiver

vw_wait_tx();
delay(500);

Serial.println("Right"); //Printing Right in the Serial Monitor
}

else if(x_axis <= -35)
{
data="l";
```



```

    vw_send((uint8_t *)data, strlen(data)); //Converting the data(l) into 8bit binary
data and then sending the data to RF receiver
    vw_wait_tx();
    delay(500);
    Serial.println("Left"); //Printing left in the Serial Monitor
}
else
{
    data="s";
    vw_send((uint8_t *)data, strlen(data)); //Converting the data(s) into 8bit binary
data and then sending the data to RF receiver
    vw_wait_tx();
    delay(500);
    Serial.println("Stop"); //Printing stop in the Serial Monitor
}
}

```

RECEIVING CIRCUIT CODE:-

```
#include <VirtualWire.h>
```

```
#define m1 2 //Declaring that m1 pin of motor driver is connected to digital pin 2
of the arduino
```

```
#define m2 3 //Declaring that m2 pin of motor driver is connected to digital pin 3
of the arduino
```

```
#define m3 4 //Declaring that m3 pin of motor driver is connected to digital pin 4
of the arduino
```


```
#define m4 5 //Declaring that m4 pin of motor driver is connected to digital pin 5
of the arduino
```

```
void setup()
{
    vw_set_rx_pin(11); //Declaring that the RF Receiver pin is connected to the pin 11
    vw_setup(2000); //Declaring the communication speed and setting it 2000 bits per
second
    pinMode(m1, OUTPUT); //Declaring that m1,m2,m3,m4 pins are output pins
    pinMode(m2, OUTPUT);
    pinMode(m3, OUTPUT);
    pinMode(m4, OUTPUT);
    vw_rx_start(); //To start the RF communication
    Serial.begin(9600); //Beginning the serial communication and setting the baud
rate
}

void loop()
{
    uint8_t buf[VW_MAX_MESSAGE_LEN]; //declaring variable to store the character
transmitted by the RF transmitter

    uint8_t buflen = VW_MAX_MESSAGE_LEN; //declaring variable to store the string
length transmitted by the RF transmitter

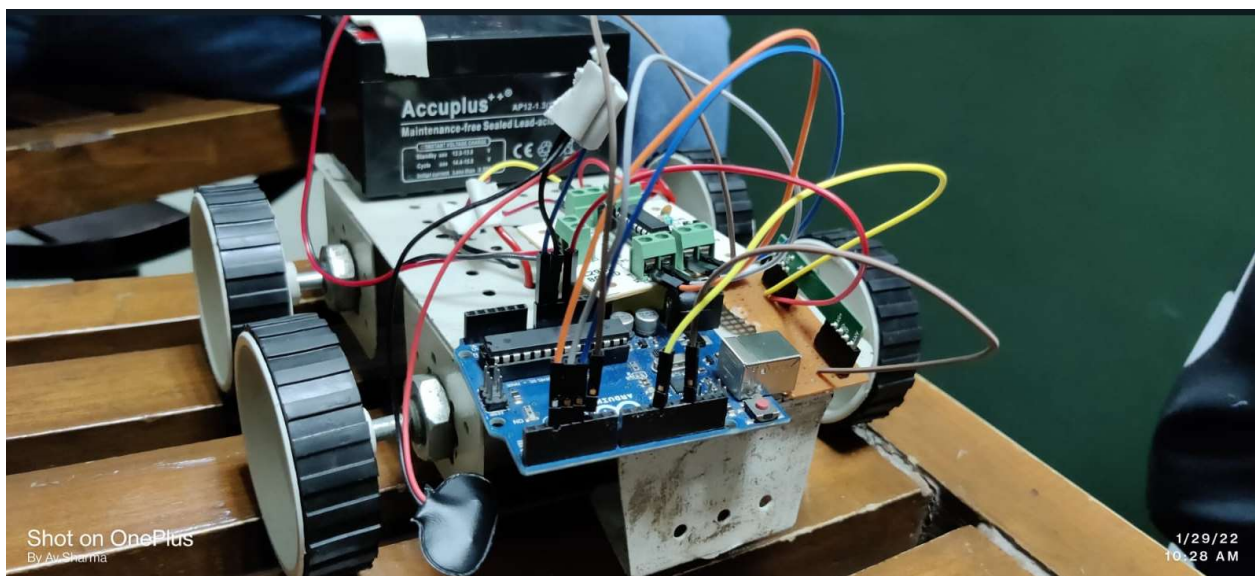
    if (vw_get_message(buf, &buflen)) //Whenever any message will come it will be
stored in buf and buflen
    {
        if(buf[0]=='f') //Condition if there is f on the zeroth position of string then go
inside
        {
            digitalWrite(m1,HIGH); //Declaring the logics of the motor pins to move forward
            digitalWrite(m2,LOW);
```

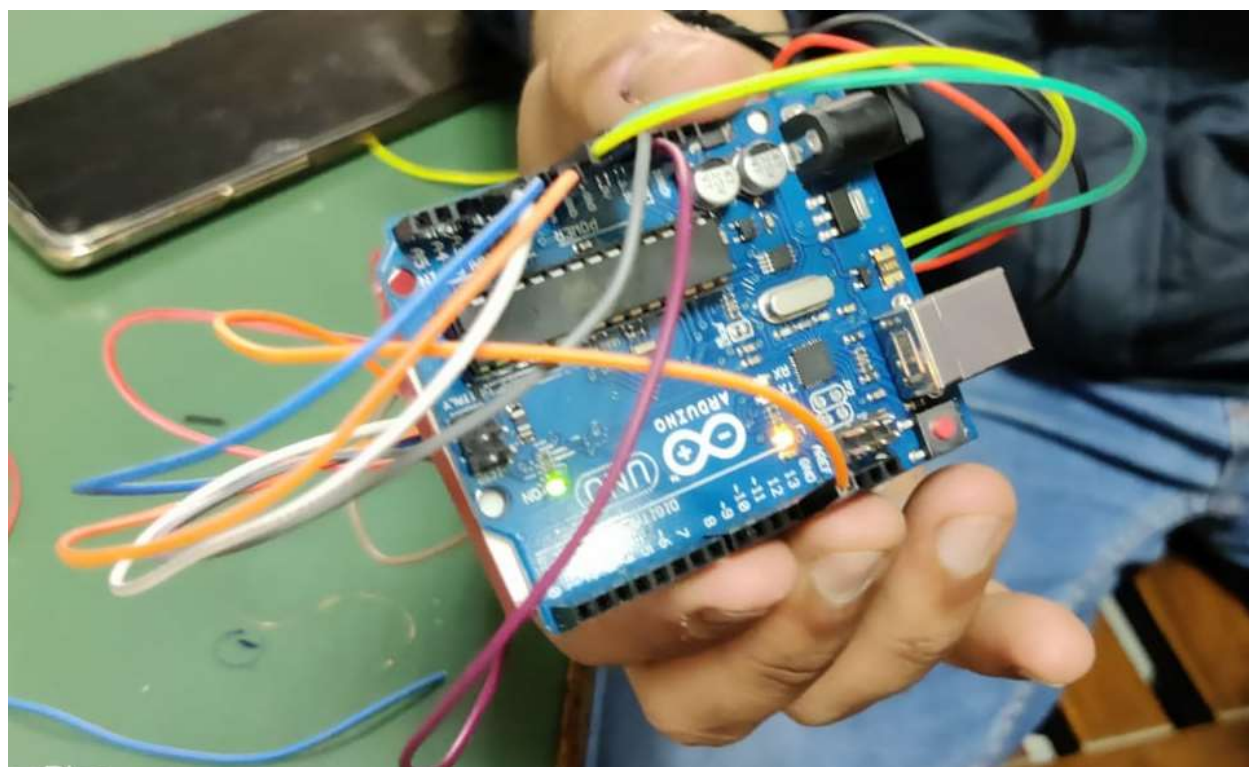


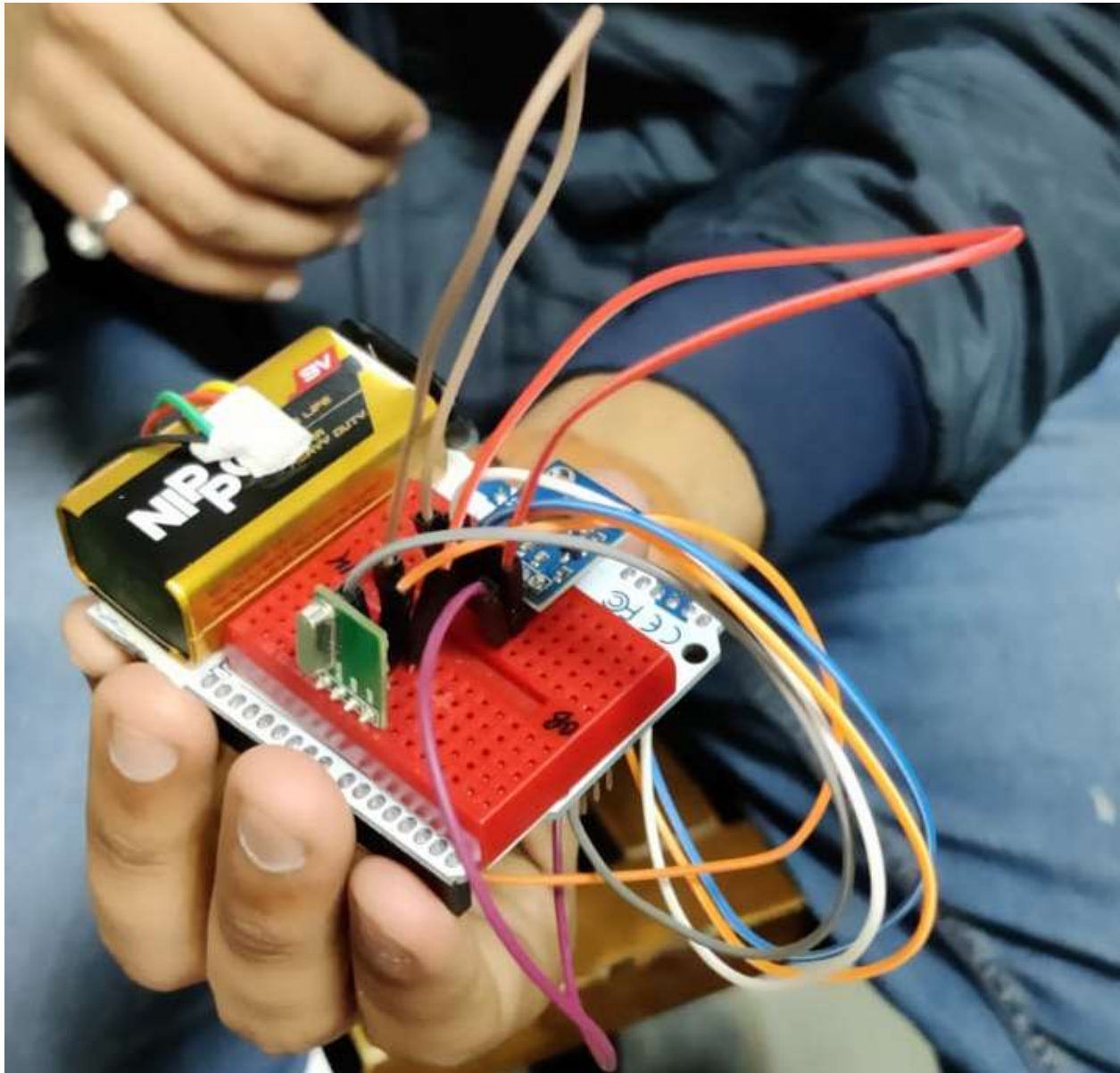
```
digitalWrite(m3,HIGH);
digitalWrite(m4,LOW);
Serial.println("Forward");
}
else if(buf[0]=='b') //Condition if there is b on the zeroth position of string then
go inside
{
digitalWrite(m1,LOW); //Declaring the logics of the motor pins to move
backward
digitalWrite(m2,HIGH);
digitalWrite(m3,LOW);
digitalWrite(m4,HIGH);
Serial.println("Backward");
}
else if(buf[0]=='r') //Condition if there is r on the zeroth position of string then go
inside
{
digitalWrite(m1,HIGH); //Declaring the logics of the motor pins to rotate right
digitalWrite(m2,LOW);
digitalWrite(m3,LOW);
digitalWrite(m4,LOW);
Serial.println("Right");
}
else if(buf[0]=='l') //Condition if there is l on the zeroth position of string then go
inside
{
digitalWrite(m1,LOW); //Declaring the logics of the motor pins to rotate left
digitalWrite(m2,LOW);
```

```
digitalWrite(m3,HIGH);  
digitalWrite(m4,LOW);  
Serial.println("Left");  
}  
else if(buf[0]=='s') //Condition if there is s on the zeroth position of string then  
go inside  
{  
digitalWrite(m1,LOW); //Declaring the logics of the motor pins to stop  
digitalWrite(m2,LOW);  
digitalWrite(m3,LOW);  
digitalWrite(m4,LOW);  
Serial.println("Stop");  
}  
}  
}
```

Output







Observations

Observations made while interfacing ADXL335 with the arduino:-

- 1) We started the experiment by interfacing the adxl335 and we observed that it consists of 5 pins in total.
- 2) 1 Vcc pin, 1 ground pin and 3 pins for the 3 axes that are x-axis, y-axis, z-axis.

- 3) We also observed that the angle measured by the Accelerometer was converted into analog and that analog value was getting printed in the serial monitor
- 4) Then we moved on the interfacing of the Rf transmitter.

Observations made while interfacing Rf transmitter with the arduino:-

- 1) In the RF module there are two pieces: the transmitter and the receiver.
- 2) The transmitter is the smaller one and the receiver is the larger one.
- 3) In the RF transmitter there are 4 pins.
- 4) 1 Vcc pin, 1 ground pin, 1 data pin and 1 pin for the antenna.
- 5) The transmitter has a range of 10 meters in normal conditions without the antenna.
- 6) And with the antenna attached the range becomes 100 metres.

Observations made while interfacing the motor driver with arduino:-

- 1) In the motor driver L298N there is a voltage regulator present on the module.
- 2) There is One channel for giving External DC power source and one output pin of 5V so we can also power our arduino board with our motor driver.
- 3) Then there are total 6 pins available on the board
- 4) They are Enable A, Input 1, Input 2, Input 3, Input 4, Enable B.
- 5) The L298N motor driver module works on the principle of the H-Bridge just like L293D.

Observations made while interfacing the RF receiver with arduino:-

- 1) In the RF receiver there are a total of 8 pins.
- 2) 3 ground pins, 2 data pins, 2 Vcc pins and 1 pin for the antenna.