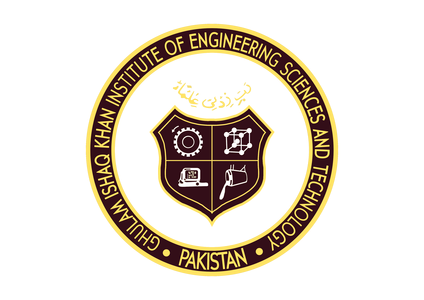
**Hand Gesture Controlled Robot**  **Microprocessor Interfacing**  
PROJECT REPORT



**Made by**Muhammad Affan Javaid 2017230

Zabhiullah 2017505

Waleed Khan 2017499

Shahid Khan 201742  
**Instructors  
Dr. Sajid Anwar**

**Engr Sanam Surreya**

**Table of Contents**

**Introduction………………………………………………………….. Page 3**

**Working Principle…………………………………………….……… Page 3**

**MPU6050 Sensor…………………………………………….………. Page 3**

**Block Diagram………………………………………………….…….. Page 7**

**Circuit Diagram of Transmitter ……………….…………………… Page 8**

**Transmitter Section…………………………………………….……. Page 8**

**Circuit Diagram of Receiver…………………………………….….. Page 9**

**What gestures will the robot recognize…………………….……. Page 11**

**Arduino Code…….…………………………………………………… Page 13**

**Applications .……………………………………………………….........Page 16**

**References………………………………………………………….... Page 16**

**Introduction**

Majority of the industrial robots are autonomous as they are required to operate at high speed and with great accuracy. But some applications require semi-autonomous or human controlled robots.

Some of the most commonly used control systems are voice recognition, tactile or touch controlled and motion controlled.

One of the frequently implemented motion controlled robot is a Hand Gesture Controlled Robot. In this project, a hand gesture controlled robot is developed using MPU6050, which is a 3-axis Accelerometer and 3-axis Gyroscope sensor and the controller part is Arduino Uno.

Instead of using a remote control with buttons or a joystick, the gestures of the hand are used to control the motion of the robot.

The project is based on wireless communication, where the data from the hand gestures is transmitted to the robot over RF link (RF Transmitter – Receiver pair).

The project is divided into transmitter and receiver section. The circuit diagram and components are explained separately for both transmitter and receiver sections.

### Working Principle

The first part is getting data from the MPU6050 Accelerometer Gyro Sensor by the Arduino. The Arduino continuously acquires data from the MPU6050 and based on the predefined parameters, it sends a data to the RF Transmitter.

The second part of the project is the Wireless Communication between the RF Transmitter and RF Receiver. The RF Transmitter, upon receiving data from Arduino (through the Encoder IC), transmits it through the RF Communication to the RF Receiver.

Finally, the third part of the project is decoding the Data received by the RF Receiver and sending appropriate signals to the Motor Driver IC, which will activate the Wheel Motors of the Robot.

MPU6050 is used only as an accelerometer to make it easy, and only the x and y parameter is taken.

### MPU-6050 Sensor

MPU-6050 is an IMU Sensor that contains a MEMS (Microelectromechanical System) Accelerometer and MEMS Gyroscope on a single chip.

Here, IMU Sensor, where IMU stands for Inertial Measurement Unit, is a device that measures the specific force using Accelerometer, angular rate using Gyroscope and magnetic field using Magnetometers.

IMU Sensors are used in self-balancing robots, aircrafts, mobile phones, tablets, spacecraft, satellites, drones, UAVs (unmanned aerial vehicles) etc. for guidance, position detection, orientation detection, motion tracking and flight control.

The two common IMUs are ADXL 335 Accelerometer and MPU-6050. The ADXL 335 contains a 3-axis Accelerometer.

In case of MPU-6050, it is a six-axis motion tracking device that combines a 3-axis Accelerometer and a 3-axis Gyroscope on a single chip. We will see more details about MPU6050 in the next section.

The MPU-6050 is a six-axis motion tracking device developed by InvenSense. The main features of the MPU6050 device are mentioned below.

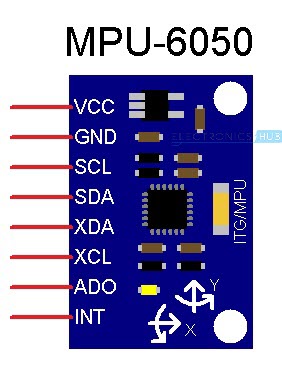
* Three – axis Accelerometer
* Three – axis Gyroscope
* Digital Output Temperature Sensor
* Six 16-bit ADC (three for Accelerometer and three for Gyro)
* Integrated Digital Motion Processor (DMP)
* 1024B FIFO Buffer

The six-axis MPU-6050 is some time called as a 6 DoF (six Degrees of Freedom) device, as it provides six output values (three from Accelerometer and three from Gyro). The MPU-6050 can communicate using I2C Protocol.

Digital Motion Processor or the DMP is an embedded processor that can reduce the computational load from the host processor, like an Arduino, by acquiring and processing data from Accelerometer, Gyroscope and an external Magnetometer.

### Interfacing MPU6050 with Arduino

### 



As mentioned earlier, the MPU6050 supports only I2C Communication and hence, it must be connected only to the I2C Pins of the Arduino. The I2C pins of Arduino are multiplexed with the analog input pins A4 and A5 i.e. A4 is SDA and A5 is SCL.

Coming to the MPU6050, we have used a normal breakout board that provided eight pins. The above image shows the schematic representation of the MPU6050 Breakout board.

In this, we will be using the SCL, SDA and the INT pins to connect with Arduino.

**Circuit Diagram MPU-605:**

**I2c Communication:**

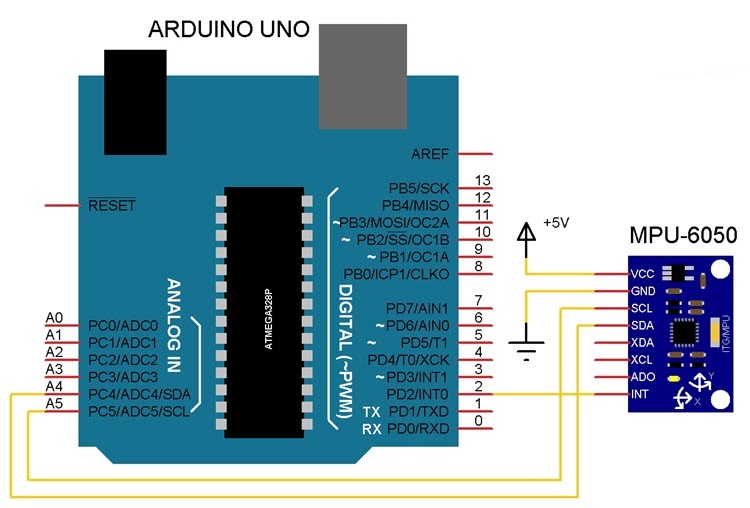
It is serial communication between master and many slaves with only two wires (SDA and SCL) .SDA is use to transfer and receive data between master and slave. SCL carries clock signal.

The following image shows the circuit diagram for interfacing MPU6050 with Arduino UNO. As mentioned earlier, the interface between MPU6050 and Arduino must be implemented using I2C Protocol.

Hence, the SCL Pin of the Arduino (A5) is connected to the SCL Pin of the MPU6050. Similarly, the SDA Pin of the Arduino (A4) is connected to the SDA Pin of the MPU6050 board.

Additionally, we will be using the Interrupt feature of the MPU6050 to indicate (or interrupt) Arduino when the 1024 Byte FIFO buffer is full. So, connect the INT pin of the MPU6050 to the external interrupt 0 (INT0) pin of Arduino UNO i.e. Pin 2.

**NOTE:** In I2C Communication, the MPU-6050 always acts as a slave.

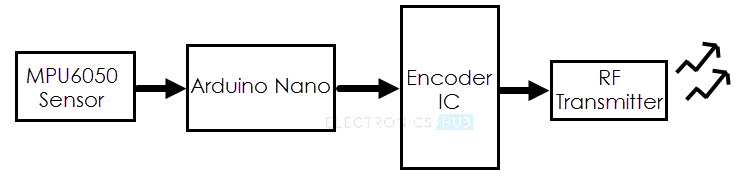


Install **MPU6050** and **12Cdev** libraries in Arduino.

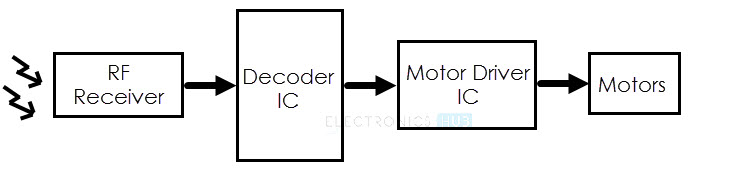
### Block Diagram of Hand Gesture Controlled Robot

The following images show the simple block diagram of Hand Gesture Controlled Robot for both Transmitter and Receiver Parts.

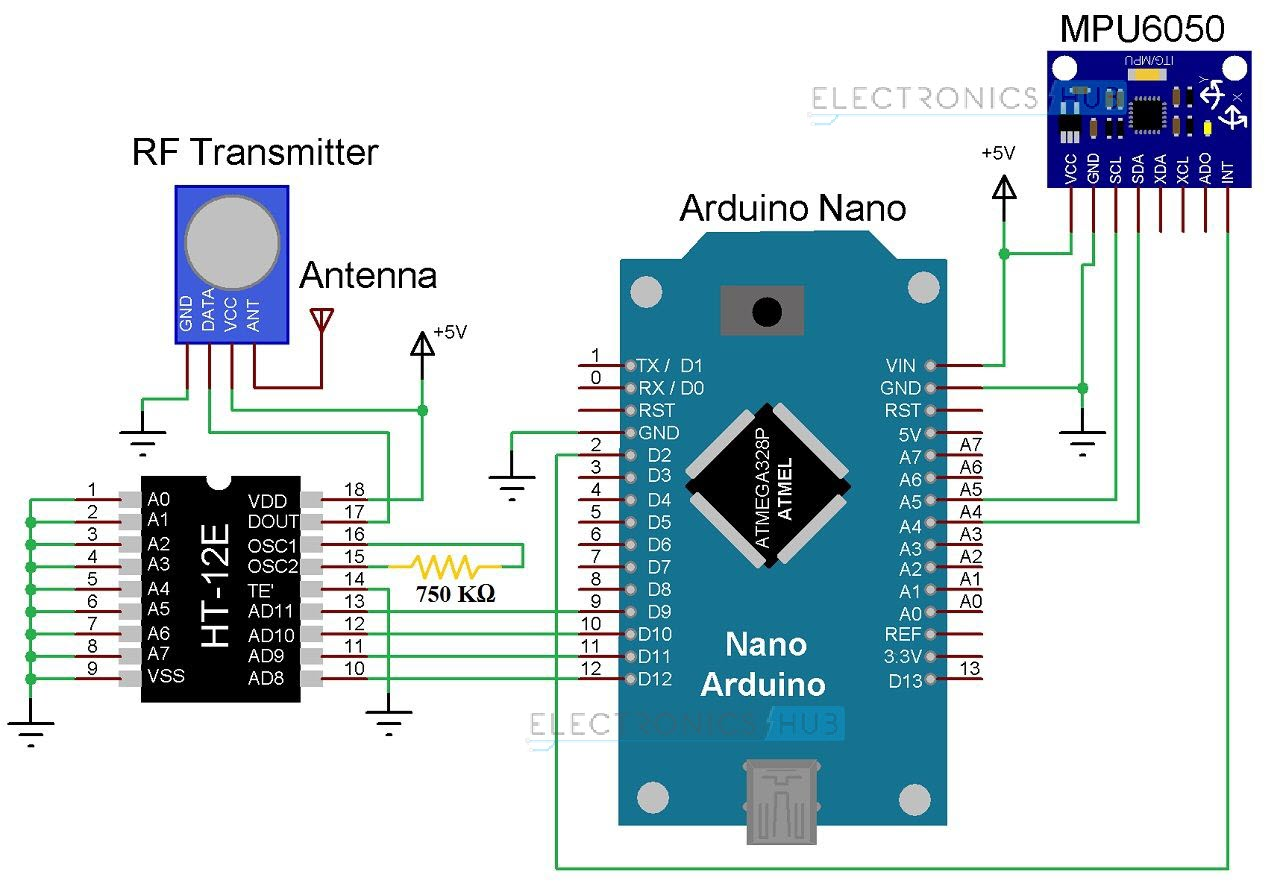
**Transmitter Block Diagram**



**Receiver Block Diagram**



### Circuit Diagram of the Transmitter Section

The following image shows the circuit diagram of the Transmitter part of the Hand Gesture Controlled Robot project.

#### Components for Transmitter Section

* Arduino Uno
* 434MHz RF Transmitter
* HT-12E Encoder IC
* MPU6050 Accelerometer/Gyroscope Sensor
* 750KΩ Resistor

#### **Transmitter Section**

The transmitter section of the robot consists of Arduino Uno board, MPU6050 Sensor, HT-12E Encoder IC and an RF Transmitter. The communication between Arduino and MPU6050 Sensor takes place through I2C Interface. Hence, the SCL and SDA pins of the MPU6050 Sensor are connected to A5 and A4 pins of the Arduino Uno.

Additionally, we will be using the interrupt pin of the MPU6050 and hence, it is connected to D2 of Arduino Uno.

HT-12E is an encoder IC that is often associated with RF Transmitter module. It converts the 12-bit parallel data to serial data. The 12-bit data is divided into address and data bits. A0 to A7 (Pin 1 to Pin8) are the address bits and they are used for secure transmission of the data. These pins can be either left open or connected to ground (Vss). In this circuit, Pin 1 to Pin 9 (A0 – A7 and Vss) of HT-12E are connected to ground.

Pins 10 to 13 (AD8, AD9, AD10 and AD11) are the data pins of HT-12E. They receive the 4 word parallel data from external source like a microcontroller (Arduino Uno in this case). They are connected to the pins D12, D11, D10 and D9 of Arduino Uno respectively.

TE’ is the transmission enable pin and it is an active low pin. The data is transmitted as long as the TE’ is low. Hence, Pin 14 (TE’) is also connected to ground.

The encoder IC has an internal oscillator circuit between the pins 16 and 15 (OSC1 and OSC2). A 750KΩ resistor is connected between these pins to enable the oscillator. Dout (Pin 17) is the serial data out pin. It is connected to the data in pin of the RF Transmitter.

Both Arduino Uno and MPU6050 have 3.3V Regulator. Hence, all the VCC pins are connected to a regulated 5V Supply.

### Circuit Diagram of the Receiver Section

 The receiver section of the robot consists of an RF Receiver, HT-12D Decoder IC, L293D Motor Driver IC and a robot chassis with four motors connected to wheels.

HT-12D is the decoder IC that is often associated with RF Receiver. It converts the serial data received by the RF link into parallel data. A0 to A7 (Pin 1 to Pin 8) are the address pins and must be matched with the address pins of the encoder.

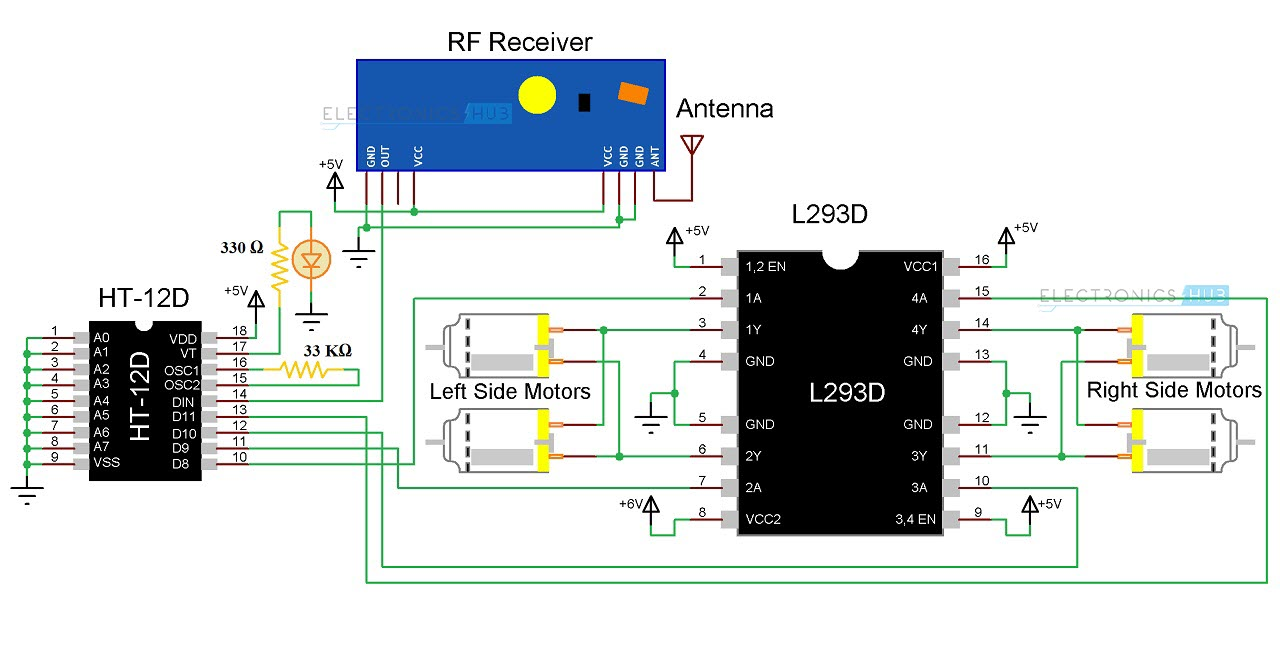
Since the address pins of encoder (HT-12E) are grounded, the address pins of decoder must also be grounded. Hence, pins 1 to 9 (A0 – A7 and Vss) are connected to ground. The serial data from the RF Receiver is given to Din (Pin 14) of the decoder IC.

HT-12D has an internal oscillator and an external resistor of 33KΩ is connected between OSC1 and OSC2 (Pins 16 and 15). Pin 17 (VT) indicates a valid transmission of data and this pin will be high when a valid data is present on the data pins. An LED in series with a 330Ω resistor is connected to this pin to indicate a valid data transmission.

Pins 10 to 13 (D8, D9, D10 and D11) of HT-12D are the parallel data out pins. They are connected to the input pins of the L293D motor driver IC (Pins 2, 7, 10 and 15 respectively).

L293D motor driver IC is used to provide the necessary current (for both forward and reverse directions) to the motors. Pins 1 and 9 are the enable pins and are connected to VCC (+5v) along with Pin 16 (which is the logic supply). Pins 3 – 6 and 11 – 14 are the outputs and are connected to the four motors.

Pin 8 is the Motor Supply Pin and is connected to a separate power supply. Hence, you will need two batteries in the Receiver Section; one for the Circuit and one for the motors.



#### **Components for Receiver Section**

* L293D Motor Driver IC
* HT-12D Decoder IC
* 434 MHz RF Receiver
* 33KΩ Resistor
* 330Ω Resistor
* LED
* 4 Geared Motors with Wheels
* Robot Chassis

#### **RF Transmitter and Receiver Modules**

The communication between transmitter and receiver is using RF modules. A 434 MHz transmitter and receiver pair are used in this project.

#### **HT-12E**

It is an encoder IC that converts the 4-bit parallel data into serial data in order to transmit over RF link.

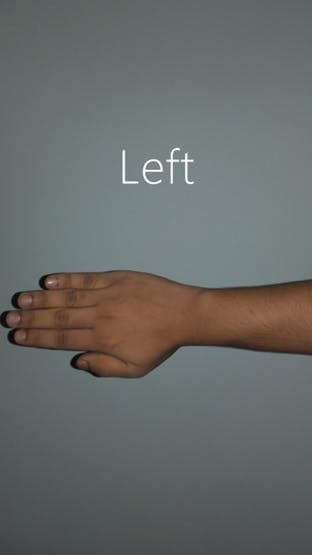
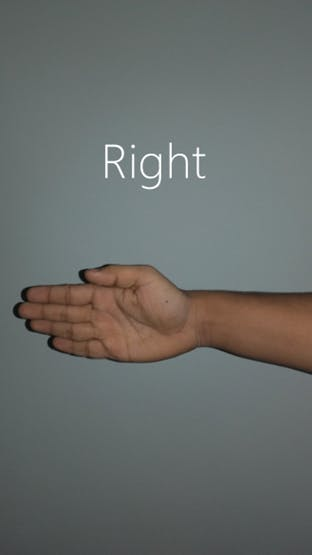
#### **HT-12D**

It is a decoder IC that converts the serial data received by the RF Receiver into 4-bit parallel data. This parallel data can be used to drive the motors.

### What gestures will the robot recognize?

This robot is designed for recognizing five sets of gestures: forward, backward, left, right and stop. You will get a better idea if you check the photos of the gestures given below.

We used x and y axis of accelerometer, when the accelerometer is flat or hand is stable the motors stop, it gets forward when the value of y is greater than 67, reverse when y is less than -50, right when x is greater than 66 and left when x is less than -45. For values in between the motors stops and are stationary.

### 

### Arduino Code

#include <Wire.h>

int lm1=10,lm2=11,rm1=5,rm2=6; //Output pins to be connected to 10, 11, 12, 13 of encoder IC

long accelX, accelY, accelZ;

float gForceX, gForceY, gForceZ;

void setup() {

  Serial.begin(9600);

  Wire.begin();// for MPU

  setupMPU();

  pinMode(lm1,OUTPUT); // for motor

  pinMode(lm2,OUTPUT);

  pinMode(rm1,OUTPUT);

  pinMode(rm2,OUTPUT);

}

void loop() {

  recordAccelRegisters();  // get values of accelerometer

  printData();

  if(gForceY>67)      // Change the value for adjusting sensitivity

     forward();

  else if(gForceY<-50) // Change the value for adjusting sensitivity

     backward();

  else if(gForceX>66) // Change the value for adjusting sensitivity

     right();

  else if(gForceX<-47) // Change the value for adjusting sensitivity

     left();

  else

  {

     stop\_();

  }

  delay(100);

}

////functions///////////////////////functions//////////////////////////////////functions//////////////////////////////////functions///////////////////////functions/////////////////////////

///////////////////////////////////////////////////////////////////function 1

void setupMPU(){    // used to set up the MPU for the first time

  Wire.beginTransmission(0b1101000); //This is the I2C address of the MPU (b1101000/b1101001 for AC0 low/high datasheet sec. 9.2)

  Wire.write(0x6B); //Accessing the register 6B - Power Management (Sec. 4.28)

  Wire.write(0b00000000); //Setting SLEEP register to 0. (Required; see Note on p. 9)

  Wire.endTransmission();

  Wire.beginTransmission(0b1101000); //I2C address of the MPU

  Wire.write(0x1C); //Accessing the register 1C - Acccelerometer Configuration (Sec. 4.5)

  Wire.write(0b00000000); //Setting the accel to +/- 2g

  Wire.endTransmission();

}

//////////////////////////////////////////////////////////////////////////////function 2

void recordAccelRegisters() {

  Wire.beginTransmission(0b1101000); //I2C address of the MPU

  Wire.write(0x3B); //Starting register for Accel Readings

  Wire.endTransmission();

  Wire.requestFrom(0b1101000,4); //Request Accel Registers (3B - 40)

  while(Wire.available() < 4);

  accelX = Wire.read()<<8|Wire.read(); //Store first two bytes into accelX

  accelY = Wire.read()<<8|Wire.read(); //Store middle two bytes into accelY

  processAccelData();

}

/////////////////////////////////////////////////////////////////////function 3

void processAccelData(){

  gForceX = (accelX / 16384.0)\*100;  // later used for left and right

  gForceY = (accelY / 16384.0)\*100;   // later used for forward and reverse

}

////////////////////////////////////////////////////////////////////////function 4

void printData() {

  Serial.print(" Accel (g)");

  Serial.print(" X=");

  Serial.print(gForceX);

  Serial.print(" Y=");

  Serial.println(gForceY);

}

////////////////////////////////////////////////////////////////////////////function 5

void stop\_()

{

  Serial.println("");

  Serial.println("STOP");

  digitalWrite(lm1,LOW);

  digitalWrite(lm2,LOW);

  digitalWrite(rm1,LOW);

  digitalWrite(rm2,LOW);

}

///////////////////////////////////////////////////////////////////////////function 6

void forward()

{

  Serial.println("");

  Serial.println("Forward");

  digitalWrite(lm1,HIGH);

  digitalWrite(lm2,LOW);

  digitalWrite(rm1,HIGH);

  digitalWrite(rm2,LOW);

}

/////////////////////////////////////////////////////////////////////function 7

void backward()

{

  Serial.println("");

  Serial.println("Backward");

  digitalWrite(lm1,LOW);

  digitalWrite(lm2,HIGH);

  digitalWrite(rm1,LOW);

  digitalWrite(rm2,HIGH);

}

//////////////////////////////////////////////////////////////////////////function 8

void left()

{

  Serial.println("");

  Serial.println("Left");

  digitalWrite(lm1,LOW);

  digitalWrite(lm2,HIGH);

  digitalWrite(rm1,HIGH);

  digitalWrite(rm2,LOW);

}

///////////////////////////////////////////////////////////////////////function 9

void right()

{

  Serial.println("");

  Serial.println("Right");

  digitalWrite(lm1,HIGH);

  digitalWrite(lm2,LOW);

  digitalWrite(rm1,LOW);

  digitalWrite(rm2,HIGH);

}

### Applications

* Wireless controlled robots are very useful in many applications like remote surveillance, military etc.
* Hand gesture controlled robot can be used by physically challenged in wheelchairs.
* Hand gesture controlled industrial grade robotic arms can be developed.
* Doctors can perform complex surgeries with maximum degree of freedom.

**References**

* <https://www.hackster.io/mayooghgirish/hand-gesture-controlled-robot-4d7587>
* https://www.electronicshub.org/hand-gesture-controlled-robot/