HAND GESTURE CONTROLLED ROBOT USING ARDUINO

A Project Report submitted in the fulfilment of the requirement for

the degree of

INFORMATION TECHNOLOGY AND MANAGEMENT

Under

UTKAL UNIVERSITY

BHUBANESWAR, ODISHA

Submitted By:-

- ***ROHIT SHARMA***

Guided by honourable:-

Dr.AMRESH SAHU,

Mr.SANAT KUMAR SWAIN.

AJAY BIAY INSTITUTE OF TECHNOLOGY

CDA, CUTTACK

ABSTRACT:-

Gesture controlled car is a Robot which can be controlled by simple human gestures. Thus user just needs to wear a gesture device in which a sensor is included. The sensor will record the movement of hand in a specific direction which will result in the motion of the robot in the respective direction. The robot and the Gesture instrument are connected wirelessly through radio waves. We can control the car using accelerometer sensors connected to a hand glove. The sensors are intended to replace the remote control that is generally used to run the car. It will allow the user to control the forward, backward, leftward and rightward movements, while using the same accelerometer sensor to control the throttle of the car.

The mechanism involves the rotation of both forth & rear wheels of left and right side to move in the anticlockwise direction and the other pair is to rotate in the clockwise direction which makes the car to rotate about its own axis without any kind of forward and backward action. The main advantages of this mechanism is the car can take a sharp turn without any difficulty. The robotic arms is designed in such a way that is consists of four movable fingers, each finger with three linkages, an opposing thumb, a rotating wrist and an elbow. The robotic arm is made to imitate the human hand movements using a hand glove

OBJECTIVE:-

A Gesture controlled vehicle is a kind of robot which is capable of carrying complex actions automatically or under human supervision.

The main objective of this project is to control the car using human hand gesture and sensed with the help of an accelerometer. It is coded in such a way that the required actions for the human gesture are done. These sensed signals are processed and then transmitted to the robotic arm at the receiver section using RF transceiver module. Thus the vehicle performs the required movement.

- **For the hardware, customised Arduino will give the control over the motors that are used to run the robotic vehicle.**
- **This system uses an RF transceiver and receiver module for the wireless communication to change the speed and the direction of the vehicle.**
- **This project helps us to control robotic arms using gesture control system and has been widely used under AI (Artificial Intelligence)**
- **This proposed model will be helpful and avoid danger for the people working in hazardous areas.**

Introduction:-

Nowadays, robotics are becoming one of the most advanced in the field of technology. A Robot is an electro-mechanical system that is operated by a computer program. Robots can be *autonomous* or *semi-autonomous*. An autonomous robot is not controlled by human and acts on its own decision by sensing its environment. 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. *A Gesture Controlled robot is a kind of robot which can be controlled by your hand gestures not by old buttons.* You just need to wear a small transmitting device in your hand which included an acceleration meter. This will transmit an appropriate command to the robot so that it can do whatever we want. The transmitting device included an ADC for analog to digital conversion and an encoder IC which is use to encode the four bit data and then it will transmit by an RF Transmitter module. At the receiving end an RF Receiver module receive's the encoded data and decode it by and decoder IC. This data is then processed by a microcontroller and finally our motor driver to control the motors. Now it's time to break the task in different module's to make the task easy and simple any project become easy or error free if it is done in different modules.

As our project is already divided into two different part *transmitter* and *receiver*. The applications of robotics mainly involve in automobiles, medical, construction, defense and also used as a fire fighting robot to help the people from the fire accident. But, controlling the robot with a remote or a switch is quite complicated. So, a new project is developed that is, an accelerometer based gesture control robot. The main goal of this project is to control the movement of the robot with hand gesture using accelerometer.

SCOPE OF THE PROJECT:

- **Connect and communicate with physical devices: IoT facilities the communication between human and machine.**
- **Faster and smart innovation: speed is very crucial aspect of any tool. Because of the use of sensors in IoT devices the required output is given in a good speed with great accuracy.**
- **Smart sensing capabilities: sensors such as accelerometer can sense very minute vibration, so the device works very precisely and can be used for such works where errors must be minimised.**

- Convenience: we can manifest very little movement on very large scale. In this way we can do maximum work which requires minimum human energy.
- This will bring efficiency along with comfort and convenience.

ARDUINO BASED GESTURE CONTROLLED VEHICLE:

The idea behind this car is to read the hand tiled angles using MPU6050 Gyroscope module and send the data through NRF transceiver based on these angles values car speed and direction will be controlled using L298N motor driver module.

COMPONENTS AND SPECIFICATIONS REQUIRED:

HARDWARE COMPONENTS:

Transmitter components:-

1. Arduino Nano,
2. NRF24LO1+ module,
3. NRF Adapter,
4. MPU6050 module,
5. 7-12v DC Battery,
6. Breadboard.

Receiver components:-

1. Arduino Nano
2. NRF24LO1+module with antenna,
3. NRF Adapter,
4. L293D motor driver,
5. 7-12v DC Battery,
6. 4*TT Gear motors with wheels and chassis,
7. Breadboard.

SYSTEM SPECIFICATIONS:-

OS: - Windows 11 HOME

OS Architecture: - 64bit,

RAM: - 4GB,

HDD: - 1 TB,

SSD:-256 GB,

Processor: - Intel i3_10th gen.

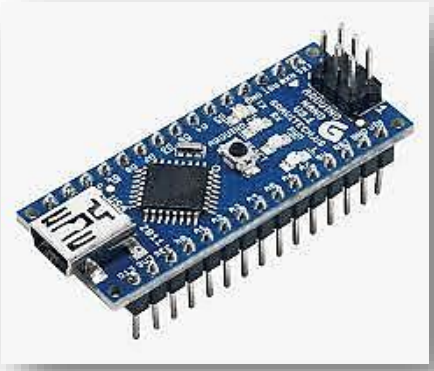
Software used:

1.Arduino IDE.

Apparatus used:

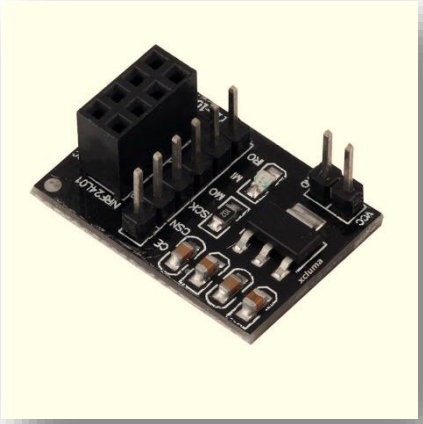
Arduino Nano:-

Arduino Nano are mainly used to build electronic projects embedded systems, robotics etc. But the Nano boards are mainly introduced for the beginners who are not from the technical background.



The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x).It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini –B USB cable instead of a standard one.

NRF Adapter:-



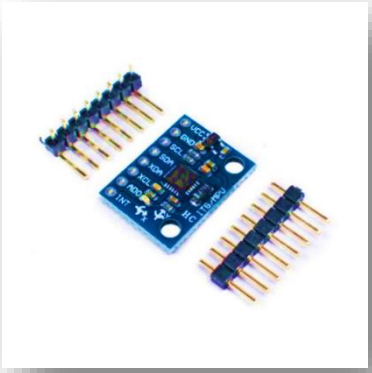
NRF serial Adapter board allows you to plug in your NRF modules and communicate through serial communication. These modules allow very reliable and simple communication between microcontrollers, computers, and system, really anything with a serial port. By default these modules are in broadest mode.

Transceiver module- NRF24L021 PA/LNA:-

The NRF24L01+ is a2.4GHz ISM band The transceiver having range from 800-1000 meters line of sight. Includes a board support components and the RFX2401C which is a PA(power amplifier) and a LNA (Low Noise Amplifier)

t is nRF24L01 pa

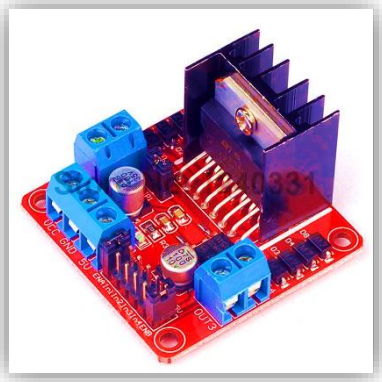
MPU6050 Module:-



MPU- 6050 Six-Axis (Gyro+ Accelerometer) MEMS Motion Tracking Device.

The MPU-6050 parts are the world's first Motion Tracking devices designed for the low power, low cost, and high-performance requirements of smartphones, tablets and wearable sensors

L298N Motor Driver:-

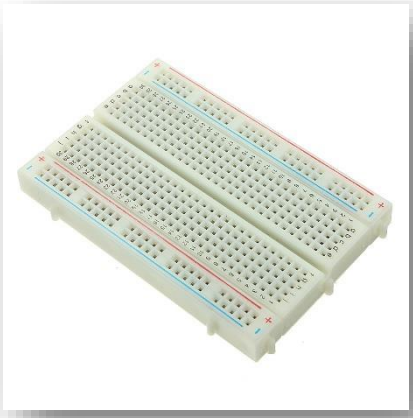


The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.

Gear Motors and wheels:-

Plastic wheels and Gearbox motors (also known as "TT" motors) are an easy, low-cost way to get your projects moving. This is a TT DC Gearbox Motor with a gear ratio of 1:48, and it comes with 2*200mm wires with breadboard-friendly 0.1 male connectors. Perfect for plugging into a breadboard or terminal blocks

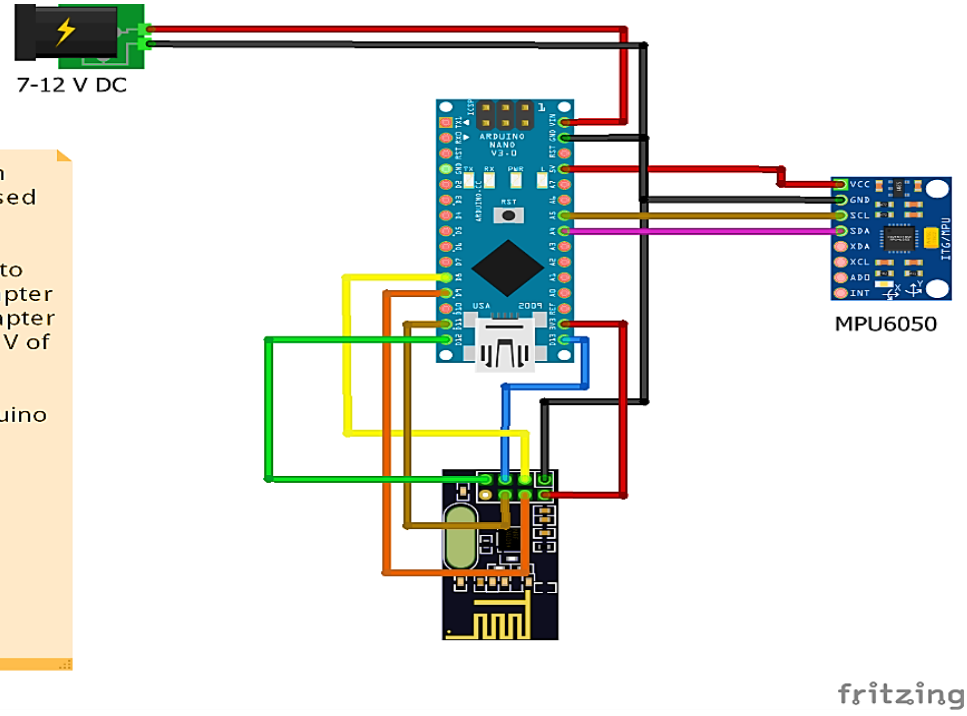
BREADBOARD:-



A breadboard (sometimes called a plugblock) is used for building temporary circuits. It is useful to designers because it allows components to be removed and replaced easily. It is useful to the person who wants to build a circuit to demonstrate its action, then to reuse the components in another circuit.

CIRCUIT DIAGRAM REPRESENTING PROJRCT MODEL:

TRANSMITTER MODULE:

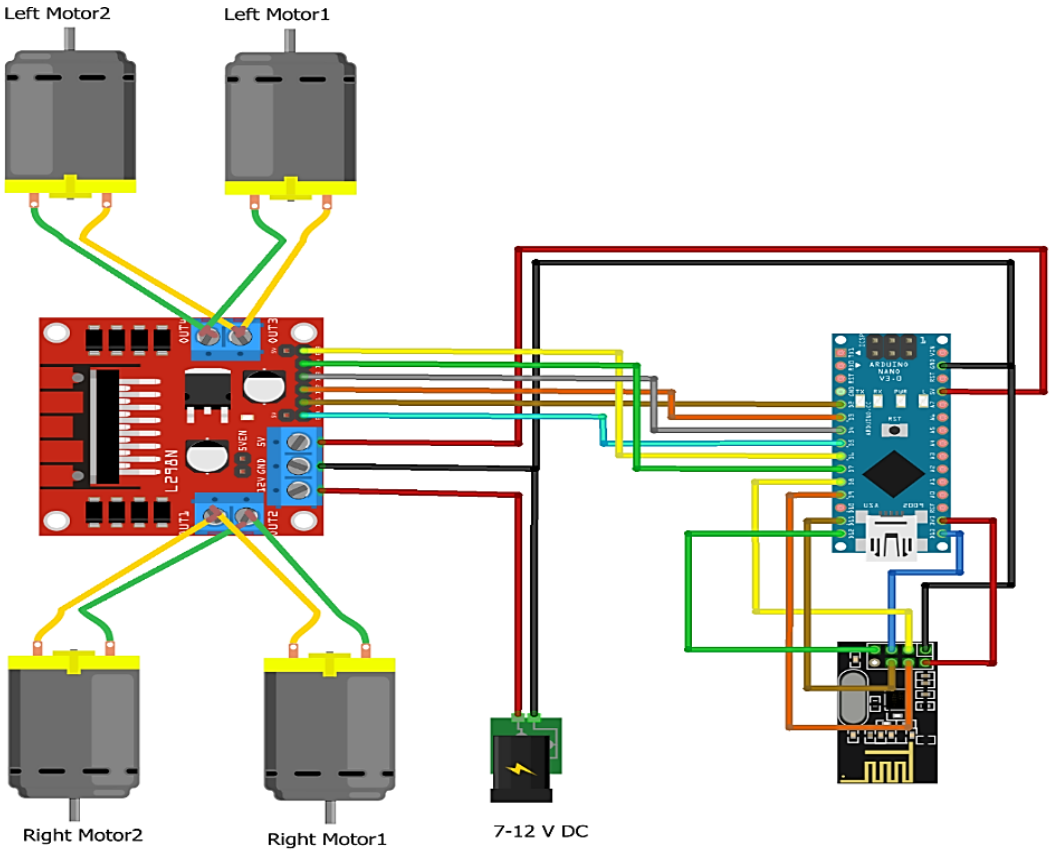


Please note that in this diagram we used nrf24l01+ for connections.
However we need to conect the nrf adapter to arduino. For adapter connect VCC to +5 V of arduino.

NRF24L01+ -> Arduino

VCC -> +3.3V
GND -> GND
CE -> 8
CS -> 9
CLK -> 13
MOSI -> 11
MISO -> 12

RECEIVER MODULE :

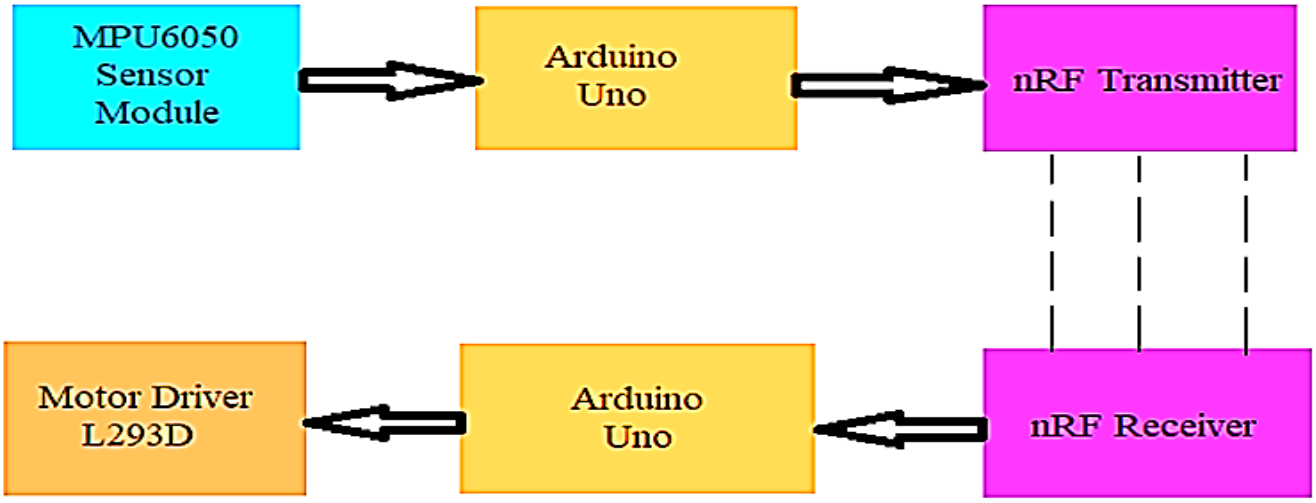


Please note that in this diagram we used nrf24l01+ for connections.
However we need to conect the nrf adapter to arduino. For adapter connect VCC to +5 V of arduino.

NRF24L01+ -> Arduino

VCC -> +3.3V
GND -> GND
CE -> 8
CS -> 9
CLK -> 13
MOSI -> 11
MISO -> 12

BLOCK DIAGRAM REPRESENTING PROJRCT MODEL:



NOTE: WE ARE USING ARDUINO NANO INSTEAD OF ARDUINO UNO.

WORKING:

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 if 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 wheels of the Robot/Car.

PROJECT CODE TO BE INTEGRATE INSIDE ARDUINO:

STEPS:-

First we have to download and install the –I2CDEV, MPU6050 RF24 Library so that we can include these libraries onto the code that going to be uploaded in Arduino.

Transmitter code:-

```
//These are needed for MPU
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
//These are need for RF handling
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
//#define PRINT_DEBUG //Uncomment this line if you want to print the MPU6050 initialization information
 on serial monitor
// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif
// MPU control/status vars
MPU6050 mpu;
bool dmpReady = false; // set true if DMP init was successful
uint8_t devStatus; // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint8_t fifoBuffer[64]; // FIFO storage buffer
```

```
Quaternion q; // [w, x, y, z] quaternion container

VectorFloat gravity; // [x, y, z] gravity vector

float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector

// RF control

const uint64_t pipeOut = 0xF9E8F0F0E1LL; //IMPORTANT: The same as in the receiver 0xF9E8F0F0E1LL

RF24 radio(8, 9); // select CE,CSN pin

struct PacketData

{

byte xAxisValue;

byte yAxisValue;

} data;

void setupRadioTransmitter()

{

radio.begin();

radio.setDataRate(RF24_250KBPS);

radio.openWritingPipe(pipeOut);

radio.stopListening(); //start the radio comunication for Transmitter


data.xAxisValue = 127; // Center

data.yAxisValue = 127; // Center

}

void setupMPU()

{

// join I2C bus (I2Cdev library doesn't do this automatically)

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE

Wire.begin();

Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties

#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
```

```
Fastwire::setup(400, true);
#endif
#ifdef PRINT_DEBUG
// initialize serial communication
Serial.begin(115200);
while (!Serial); // wait for Leonardo enumeration, others continue immediately
// initialize device
Serial.println(F("Initializing I2C devices..."));
#endif
mpu.initialize();

#ifdef PRINT_DEBUG
// verify connection
Serial.println(F("Testing device connections..."));
Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050
 connection failed"));
// wait for ready
Serial.println(F("\nSend any character to begin DMP programming and demo: "));
while (Serial.available() && Serial.read()); // empty buffer
while (!Serial.available()); // wait for data
while (Serial.available() && Serial.read()); // empty buffer again
// load and configure the DMP
Serial.println(F("Initializing DMP..."));
#endif
devStatus = mpu.dmpInitialize();
// make sure it worked (returns 0 if so)
if (devStatus == 0)
{

// Calibration Time: generate offsets and calibrate our MPU6050
```

```
mpu.CalibrateAccel(6);


mpu.CalibrateGyro(6);



#ifdef PRINT_DEBUG

mpu.PrintActiveOffsets();

// turn on the DMP, now that it's ready

Serial.println(F("Enabling DMP..."));

#endif

mpu.setDMPEnabled(true);

// set our DMP Ready flag so the main loop() function knows it's okay to use it

#ifdef PRINT_DEBUG

Serial.println(F("DMP ready! Waiting for first interrupt..."));

#endif

dmpReady = true;

// get expected DMP packet size for later comparison

packetSize = mpu.dmpGetFIFOPacketSize();

}

else

{

// ERROR!

// 1 = initial memory load failed

// 2 = DMP configuration updates failed

// (if it's going to break, usually the code will be 1)

#ifdef PRINT_DEBUG

Serial.print(F("DMP Initialization failed (code "));

Serial.print(devStatus);


Serial.println(F(")"));
```

```
#endif
}
}
void setup()
{
//This is to set up radio transmitter for rf
setupRadioTransmitter();
//This is to set up MPU6050 sensor
setupMPU();
}
void loop()
{
// if programming failed, don't try to do anything
if (!dmpReady) return;
// read a packet from FIFO. Get the Latest packet
if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer))
{
// display Euler angles in degrees
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
int xAxisValue = constrain(ypr[2] * 180/M_PI, -90, 90);
int yAxisValue = constrain(ypr[1] * 180/M_PI, -90, 90);
data.xAxisValue = map(xAxisValue, -90, 90, 0, 254);
data.yAxisValue = map(yAxisValue, -90, 90, 254, 0);
radio.write(&data, sizeof(PacketData));
#ifdef PRINT_DEBUG
Serial.println(xAxisValue);
```

```
Serial.println(yAxisValue);

#endif

}

}
```

Receiver code:-

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>


//#define PRINT_DEBUG //Uncomment this line if you want to print the values on
 serial monitor

#define SIGNAL_TIMEOUT 500 // This is signal timeout in mills seconds.

const uint64_t pipeIn = 0xF9E8F0F0E1LL;
RF24 radio(8, 9);
unsigned long lastRecvTime = 0;


struct PacketData
{
byte xAxisValue; 0
byte yAxisValue;
} receiverData;


//Right motor
int enableRightMotor=5;
int rightMotorPin1=2;
int rightMotorPin2=3;


//Left motor
int enableLeftMotor=6;
int leftMotorPin1=4;
int leftMotorPin2=7;
void setup()
```

```
{
pinMode(enableRightMotor,OUTPUT);


pinMode(rightMotorPin1,OUTPUT);
pinMode(rightMotorPin2,OUTPUT);


pinMode(enableLeftMotor,OUTPUT);
pinMode(leftMotorPin1,OUTPUT);
pinMode(leftMotorPin2,OUTPUT);


rotateMotor(0,0);


radio.begin();
radio.setDataRate(RF24_250KBPS);
radio.openReadingPipe(1,pipeIn);
radio.startListening(); //start the radio receiver


#ifdef PRINT_DEBUG
Serial.begin(115200);
#endif
}


void loop()
{
int rightMotorSpeed=0;
int leftMotorSpeed=0;
// Check if RF is connected and packet is available
if(radio.isChipConnected() && radio.available())
{
radio.read(&receiverData, sizeof(PacketData));
int mappedYValue = map(receiverData.yAxisValue, 0, 254, -255, 255);
int mappedXValue = map(receiverData.xAxisValue, 0, 254, -255, 255);
int motorDirection = 1;
if (mappedYValue < 0)
{
motorDirection = -1;
}
```

```
{
rightMotorSpeed = abs(mappedYValue) - mappedXValue;
leftMotorSpeed = abs(mappedYValue) + mappedXValue;


rightMotorSpeed = constrain(rightMotorSpeed, 0, 255);
leftMotorSpeed = constrain(leftMotorSpeed, 0, 255);


rotateMotor(rightMotorSpeed * motorDirection, leftMotorSpeed * motorDirection);


lastRecvTime = millis();


#ifdef PRINT_DEBUG
Serial.println(receiverData.xAxisValue);
Serial.println(receiverData.yAxisValue);
#endif
}
Else
{
//Signal lost. Reset the motor to stop
unsigned long now = millis();
if ( now - lastRecvTime > SIGNAL_TIMEOUT )
{
rotateMotor(0, 0);
}
}
}

void rotateMotor(int rightMotorSpeed, int leftMotorSpeed)
{
if (rightMotorSpeed < 0)
{
digitalWrite(rightMotorPin1,LOW);
digitalWrite(rightMotorPin2,HIGH);
}
else if (rightMotorSpeed > 0)
{
digitalWrite(rightMotorPin1,HIGH);

digitalWrite(rightMotorPin2,LOW);
}
```

```
Else
{
digitalWrite(rightMotorPin1,LOW);
digitalWrite(rightMotorPin2,LOW);
}


if (leftMotorSpeed < 0)
{
digitalWrite(leftMotorPin1,LOW);
digitalWrite(leftMotorPin2,HIGH);
}
else if (leftMotorSpeed > 0)
{
digitalWrite(leftMotorPin1,HIGH);
digitalWrite(leftMotorPin2,LOW);
}
Else
{
digitalWrite(leftMotorPin1,LOW);
digitalWrite(leftMotorPin2,LOW);
}


analogWrite(enableRightMotor, abs(rightMotorSpeed));
analogWrite(enableLeftMotor, abs(leftMotorSpeed));

}
```

APPLICATIONS OF GETURE CONTROLLED VEHICLE:-

- **Wireless controlled robots are very useful in many applications like remote surveillance, military etc.**
- **Hand gesture controlled robots can be used by physically challenged in wheelchairs.**
- **Hand gesture controlled industrial grade robotic arms can be developed**

ADVANTAGES:

- Gesture reorganisation technology that uses sensors to read and interpret hand movements as commands.
- This mechanism can take a sharp turn without any difficulty.
- The design and implementation of a gesture control robotic arm using flex sensor is proposed.

In the automotive industry, this capability allows drivers and passengers to interact with the infotainment system without touching any buttons or screens .eg:-used in BMW 7 Series in 2015

CONCLUSION:

- Gesture recognition is technology that uses sensors to read and interpret hand movements as commands
- In this project using Arduino Gesture control system was implemented.
- This project helps us to control robotic arms using gesture control system and has been widely used under AI (Artificial Intelligence) and upcoming technologies.
- In Automotive industry, this capability allows drivers and passengers to interact with the vehicle –usually to control the infotainment system without touching any buttons or screens.
- In the future it can be modified better to understand wireless futuristic technologies to communicate between robots and humans.

REFERENCE:-

Codes & Diagram link:-

https://github.com/un0038998/Gesture.C...

Libraries – I2CDev and MPU6050:

https://github.com/jrowberg/i2cdevlib

❄..... END OF THE DOCUMENTATION .....❄