GESTURE CONTROLLED ROBOTIC CAR

ABSTRACT:

This project explores the design and development of a hand gesture-controlled robotic car, showcasing the seamless integration of wearable technology, embedded systems, and robotics. The primary innovation lies in using an MPU6050 gyroscope sensor and accelerometer sensor to detect hand tilt angles, which are processed by an Arduino Nano microcontroller. The processed data is transmitted wirelessly using HC-05 Bluetooth modules, enabling real-time communication between the user's hand and the robotic car.

At the receiver end, another Arduino Uno decodes the transmitted data and adjusts the car's speed and direction via an L298N motor driver module. The robotic car operates based on tilt values, providing an intuitive and interactive user experience. The system's modular design ensures flexibility, allowing the choice between Bluetooth or RF modules for communication.

This project demonstrates how gesture recognition can be effectively utilized in robotics for user-friendly control. Potential applications include assistive devices for differently-abled individuals, interactive toys, and innovative gaming platforms. By bridging wearable sensor technology and robotics, this project underscores the practicality and versatility of modern embedded systems.

COMPONENTS:

Transmitter Unit:

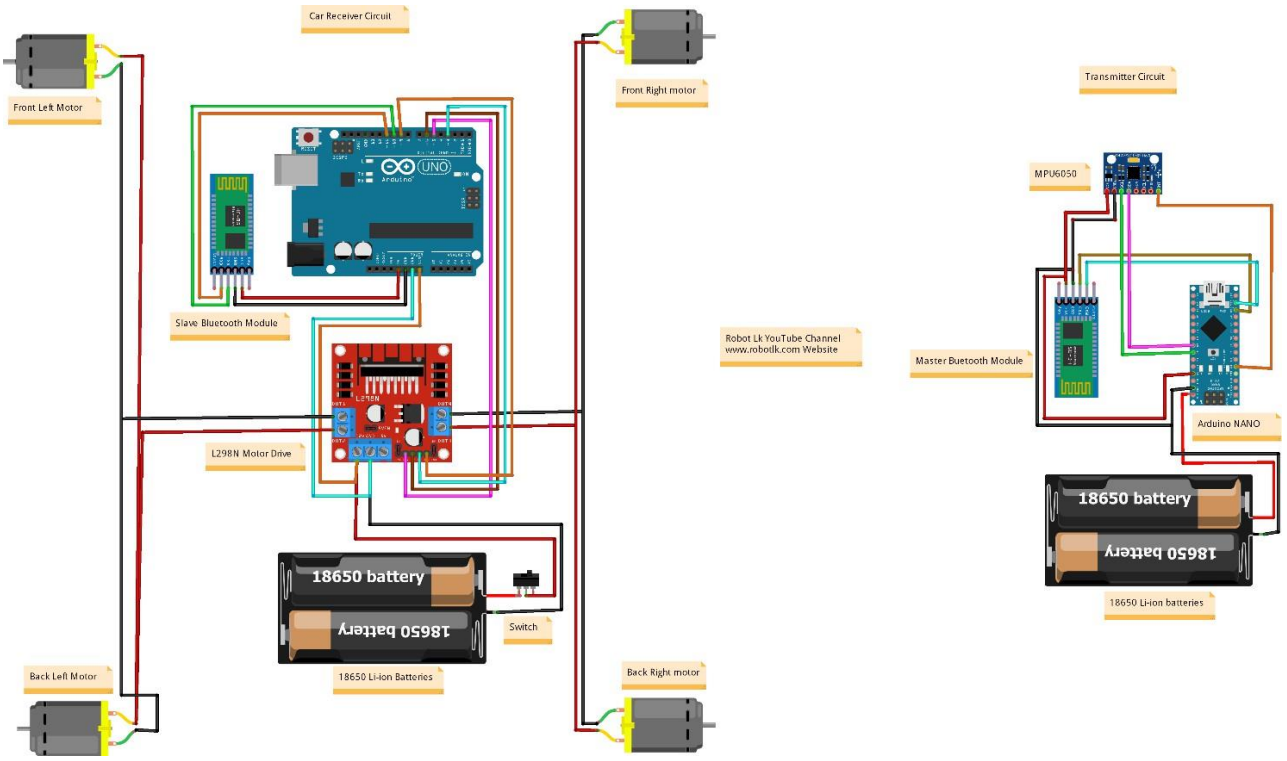
1. Arduino Nano
2. MPU6050 Module
3. HC-05 Bluetooth Module
4. Battery

5. Breadboard and Jumper Wires

Receiver Unit:

1. 2WD Car Kit
2. Arduino Uno
3. HC-05 Bluetooth Module
4. L298N Motor Driver Module
5. Battery
6. Breadboard and Jumper Wires

CIRCUIT DIAGRAM:



WORKFLOW:

1. Input Stage (Gesture Detection):

- The MPU6050 sensor detects hand movements by measuring tilt angles along the X, Y, and Z axes.
- The tilt data represents the orientation of the hand (e.g., forward tilt for moving forward, left tilt for turning left).
- This data is processed by the Arduino Nano on the transmitter side, converting the raw sensor values into gesture-specific commands.

2. Processing Stage (Command Generation):

- The Arduino Nano interprets the processed data from the MPU6050 module to recognize specific gestures.
- Each gesture is mapped to a predefined command. For instance:

  Forward tilt → Move forward

  Backward tilt → Move backward

  Left tilt → Turn left

  Right tilt → Turn right
- The gesture is translated into a digital command signal.

3. Transmission Stage (Wireless Communication):

- The command signal generated by the Arduino Nano is transmitted wirelessly using the HC-05 Bluetooth module to the receiver unit on the robotic car.

4. Reception Stage (Command Interpretation):

- The receiver unit's HC-05 Bluetooth module receives the transmitted commands and passes them to the Arduino Uno.

- The Arduino Uno interprets these commands and prepares motor control instructions.

5. Output Stage (Robotic Car Movement):

- The L298N motor driver module controls the car's motors based on the received commands. It adjusts:

  Direction: Whether the car moves forward, backward, left, or right.

  Speed: Controlled by varying the voltage and current supplied to the motors.

- The robotic car responds to the gestures in real-time, executing movements corresponding to the hand gestures.

CODE:

Transmitter Code:

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11); // CONNECT BT RX PIN TO ARDUINO 11 PIN | CONNECT BT TX PIN TO ARDUINO 10 PIN

#define OUTPUT_READABLE_YAWPITCHROLL
#define INTERRUPT_PIN 2
#define LED_PIN 13

MPU6050 mpu;
bool blinkState = false;
bool dmpReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];

// orientation/motion vars
Quaternion q;
VectorFloat gravity;
```

```
float ypr[3];
float pitch = 0;
float roll = 0;
float yaw = 0;
int x;
int y;
volatile bool mpuInterrupt = false;
void dmpDataReady() {
mpuInterrupt = true;
}
void setup() {

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  Wire.setClock(400000);
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif
  Serial.begin(38400);
  BTSerial.begin(9600);
  while (!Serial);
  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();
  pinMode(INTERRUPT_PIN, INPUT);
  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050
connection failed"));
  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize()
  mpu.setXGyroOffset(126);
  mpu.setYGyroOffset(57);
  mpu.setZGyroOffset(-69);
  mpu.setZAccelOffset(1869);
  if (devStatus == 0) {
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);
    Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();
    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;
    packetSize = mpu.dmpGetFIFOPacketSize();
    }
    else {
    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
    Serial.println(F(")"));
  }
  pinMode(LED_PIN, OUTPUT);
}
```

```
void loop() {
    if (!dmpReady) return;
  while (!mpuInterrupt && fifoCount < packetSize) {
  }
  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();
  fifoCount = mpu.getFIFOCount();

  if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
    mpu.resetFIFO();
    Serial.println(F("FIFO overflow!"));
  } else if (mpuIntStatus & 0x02) {
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
    mpu.getFIFOBytes(fifoBuffer, packetSize);
    fifoCount -= packetSize;

#ifdef OUTPUT_READABLE_YAWPITCHROLL
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    yaw = ypr[0] * 180 / M_PI;
    pitch = ypr[1] * 180 / M_PI;
    roll = ypr[2] * 180 / M_PI;

    if (roll > -100 && roll < 100)
      x = map (roll, -100, 100, 0, 100);

    if (pitch > -100 && pitch < 100)
      y = map (pitch, -100, 100, 100, 200);

    Serial.print(x);
    Serial.print("\t");
    Serial.println(y);

    if((x>=45 && x<=55) && (y>=145 && y <=155)){
      BTSerial.write('S');
    }else if(x>60){
      BTSerial.write('R');
    }else if(x<40){
      BTSerial.write('L');
    }else if(y>160){
      BTSerial.write('B');
    }else if(y<140){
      BTSerial.write('F');
    }
#endif
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
  }
}
```

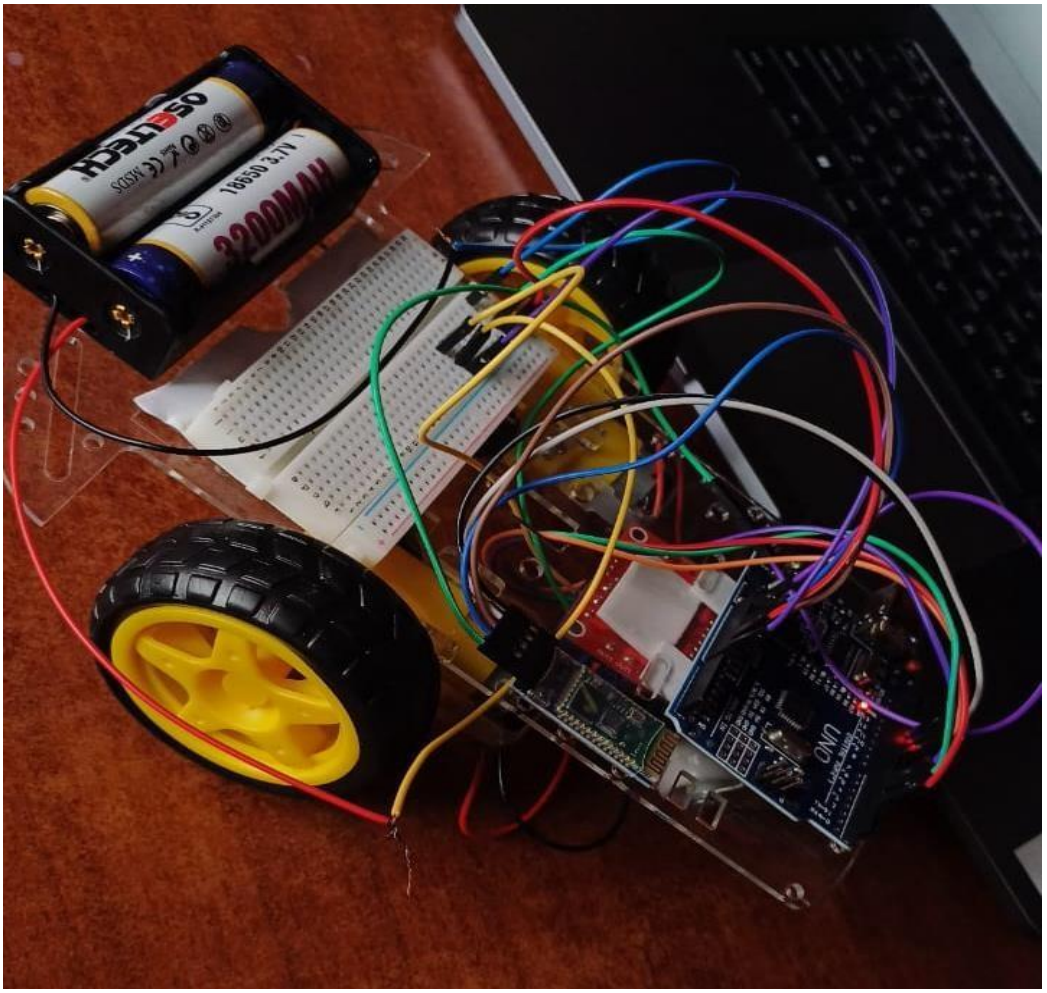
Receiver Code:

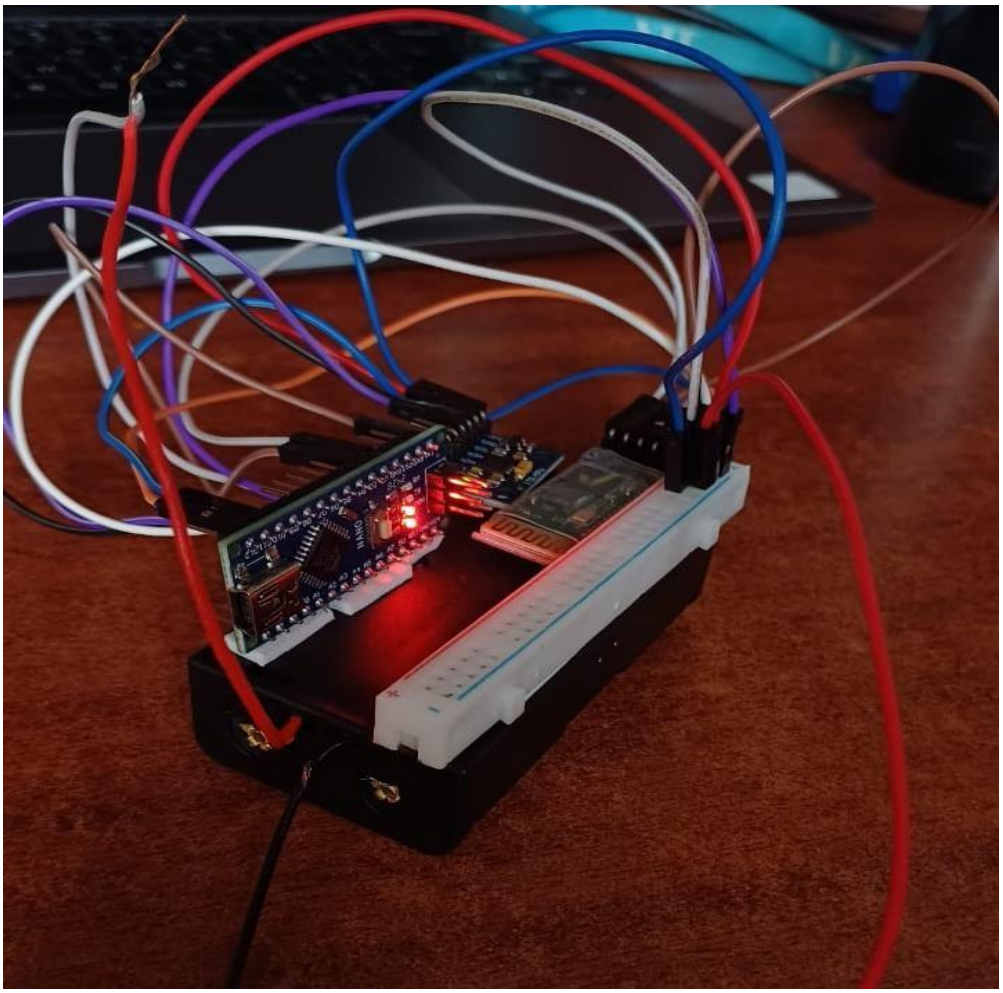
```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11); // CONNECT BT RX PIN TO ARDUINO 11 PIN |
CONNECT BT TX PIN TO ARDUINO 10 PIN

char tiltDirection;
int motorInput1 = 5;
int motorInput2 = 6;
int motorInput3 = 3;
int motorInput4 = 9;

void setup() {
  pinMode(motorInput1, OUTPUT);
  pinMode(motorInput2, OUTPUT);
  pinMode(motorInput3, OUTPUT);
  pinMode(motorInput4, OUTPUT);
  digitalWrite(motorInput1, LOW);
  digitalWrite(motorInput2, LOW);
  digitalWrite(motorInput3, LOW);
  digitalWrite(motorInput4, LOW);
  Serial.begin(38400);       // Serial communication is activated at 38400 baud/s.
  BTSerial.begin(9600);     // HC-05 default speed in AT command more
}
//Robot lk
void loop() {
  if (BTSerial.available()) {
    tiltDirection = BTSerial.read();
    if(tiltDirection == 'F'){
    Serial.println("Forward");
       reverse();
    }else if(tiltDirection == 'B'){
      Serial.println("Reverse");
      forward();
    }else if(tiltDirection == 'R'){
      Serial.println("Right");
      left();
    }else if(tiltDirection == 'L'){
      Serial.println("Left");
      right();
    }else if(tiltDirection == 'S'){
      Serial.println("Stop");
      stopCar();
    }
  }
}
//Robot lk
void forward()
{
```

```
  digitalWrite(motorInput1,  LOW);
  digitalWrite(motorInput2, HIGH);
  digitalWrite(motorInput3,  LOW);
  digitalWrite(motorInput4, HIGH);
}
void reverse()
{
  digitalWrite(motorInput1, HIGH);
  digitalWrite(motorInput2,  LOW);
  digitalWrite(motorInput3, HIGH);
  digitalWrite(motorInput4, LOW);
}
void right()
{
  digitalWrite(motorInput1, LOW);
  analogWrite(motorInput2, 150);
  analogWrite(motorInput3, 150);
  digitalWrite(motorInput4, LOW);
}
void left()
{
  analogWrite(motorInput1, 150);
  digitalWrite(motorInput2, LOW);
  digitalWrite(motorInput3, LOW);
  analogWrite(motorInput4, 150);
}
void stopCar() {
  digitalWrite(motorInput1, LOW);
  digitalWrite(motorInput2, LOW);
  digitalWrite(motorInput3, LOW);
  digitalWrite(motorInput4, LOW);
}
```


18650 3.7V
3200MAH
UNO



The hand gesture-controlled robotic car project effectively demonstrates the integration of wearable sensors, wireless communication, and microcontroller systems to enable intuitive control through simple hand movements. Using the MPU6050 accelerometer and HC-05 Bluetooth module, the system allows real-time control of the car, offering applications in assistive technologies, gaming, and robotics.

Future enhancements could include autonomous navigation with obstacle detection, voice command integration, and improved speed and precision controls. This project showcases the potential of wearable technology in creating user-friendly, efficient control systems, with opportunities for scalability and improvement in various fields.