TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
1338 Arlegui Street, Quiapo, Manila
DEPARTMENT OF COMPUTER ENGINEERING

Technological Institute of the Philippines
Manila Campus

Hand Gesture Controlled Robot using Arduino Nano

Submitted by:

Hizola, Justine
Marasigan, Jullianne Yen
Mendoza, Carl Justine
Mutuc, John Carlo
Perico Jr., Ricardo
Ramos, Anton Gerard

Bachelor of Science in Computer Engineering

Submitted To:

Phd. Mon Arjay Malbog

Submitted to the Computer Engineering Department
Technological Institute of the Philippines – Manila

TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
1338 Arlegui Street, Quiapo, Manila
DEPARTMENT OF COMPUTER ENGINEERING

December 13, 2024

I. Overview

The Arduino Nano serves for gesture detection, an Arduino Uno for robot movement, and a gyroscope sensor to make the control of a hand gesture robot possible by following the movements of a human hand. The Arduino Nano interprets the gestures by processing data about the hand orientation and movement through the gyroscope. A transmitter-receiver pair is used to wirelessly relay these orders, and the Arduino Uno on the robot's side interprets the signals to regulate its movements. The robot has DC motors running off a battery so that it can move in response to orders. This configuration offers a simple yet efficient method of using hand gestures to intuitively control a robot.

II. Introduction

A hand-gesture-controlled robot is a system that uses the user's hand movements to control the robot's movement. It replaces buttons, joysticks, or any common way of controlling this device. The system utilized sensors for real-time interpretation of the user's hand gestures, hand gestures like waving back, waving right, waving left, or waving down will dictate how the robot will move. This advanced approach to controlling a moving device provides us with an interactive way of controlling moving devices like robots, helping students educate themselves on how to use sensors in different ways.

III. Methodology

Component Setup:

- To determine hand orientations and motions, connect the Arduino Nano to the gyroscope.

- For wireless communication, connect the Arduino Nano to the transmitter module.
- To receive transmitted signals, attach the Arduino Uno to the receiver module on the robot.
- To control the robot's movement, connect DC motors to the Arduino Uno using a motor driver circuit.
- Use a battery to power the system, making sure that every component has the right voltage regulation.

Gesture Detection:

- Assign particular hand motions to predetermined commands (such as forward, backward, left, right, and stop) by programming the Arduino Nano to read and interpret gyroscope data.

Signal Transmission:

- Wirelessly send the Arduino Nano's gesture-based commands to the robot's receiver module.

Command Processing:

- Set up the Arduino Uno to interpret commands and generate the proper DC motor control signals.

Robot Movement:

- Based on the control signals, the Arduino Uno turns on the DC motors, allowing the robot to move in the appropriate direction.

Testing and Calibration:

- To guarantee precise gesture recognition and responsive robot movement, test the system.
- Adjust the gyroscope's calibration to increase gesture detection's sensitivity and precision.

Validation:

- To guarantee dependable operation and make any required system improvements, assess the robot's performance in a variety of circumstances.

IV. Components

1. Input Subsystem (Gesture Detection)

- **Gyroscope Sensor:**
  - Detects rotational movements and orientation based on angular velocity.
  - Outputs data for X, Y, and Z axes.
  - Connects to the microcontroller on the transmitter side.

---

2. Processing Unit (Transmitter Side)

- **Microcontroller (Arduino Nano):**
  - Reads digital signals from the gyroscope.
  - Processes the gyroscope data to interpret gestures.
  - Converts gestures into commands for the robot's movement.
  - Sends the commands wirelessly to the robot via the communication module.
- **Wireless Module (NRF24L01):**
  - Transmits the processed commands from the transmitter to the receiver module on the robot.
- **Power Source:**
  - A 9V battery powers the transmitter system.

---

3. Communication Subsystem

- **Wireless Communication:**
  - The NRF24L01 module establishes a communication link between the transmitter and receiver.
  - Sends gesture commands to the receiver on the robot.

---

4. Output Subsystem (Robot Control)

- **Receiver:**
  - The NRF24L01 module on the robot receives the wireless commands.
  - Decodes the commands and passes them to the microcontroller for further action.
- **Processing Unit (Receiver Side):**
  - Microcontroller (Arduino Nano):
    - Processes received commands.
    - Generates control signals for the motor driver to execute movements.

---

5. Motor Control Subsystem

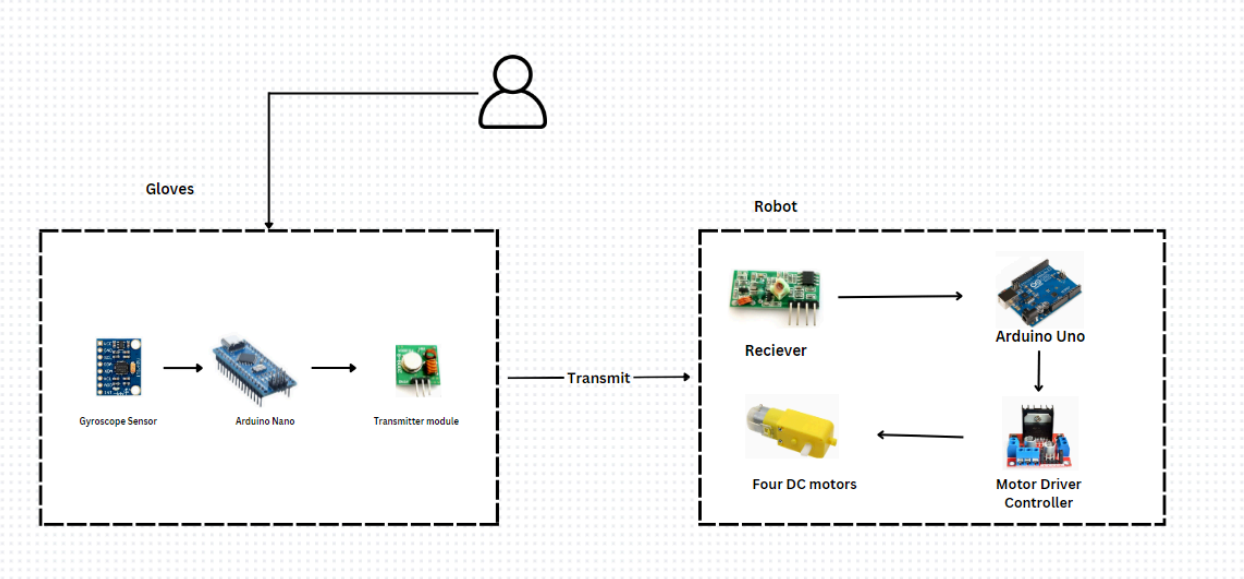
- **Motor Driver (L298N):**
  - Drives four DC motors connected to the robot's wheels.
  - Receives control signals from the Arduino to regulate speed and direction.
- **Motors:**
  - Four DC motors provide the robot with motion in the desired direction (forward, backward, left, or right).

---

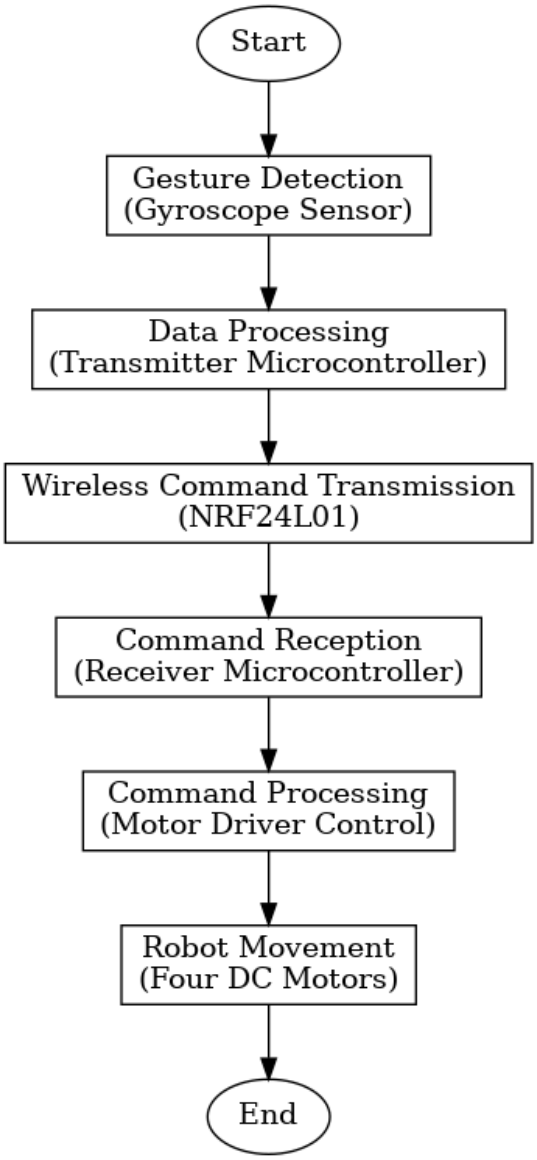
6. Power System

- **Battery Pack:**
  - Two 18650 batteries supply power to the motor driver and receiver circuit.

V. System Architecture

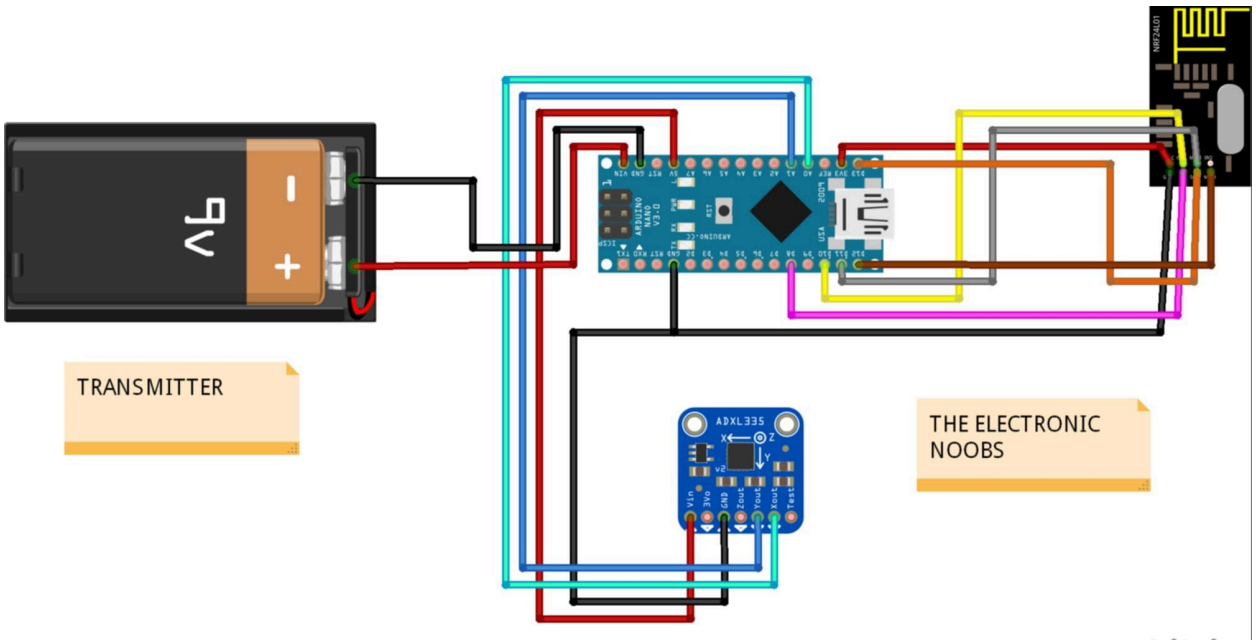


VI. System Flowchart:

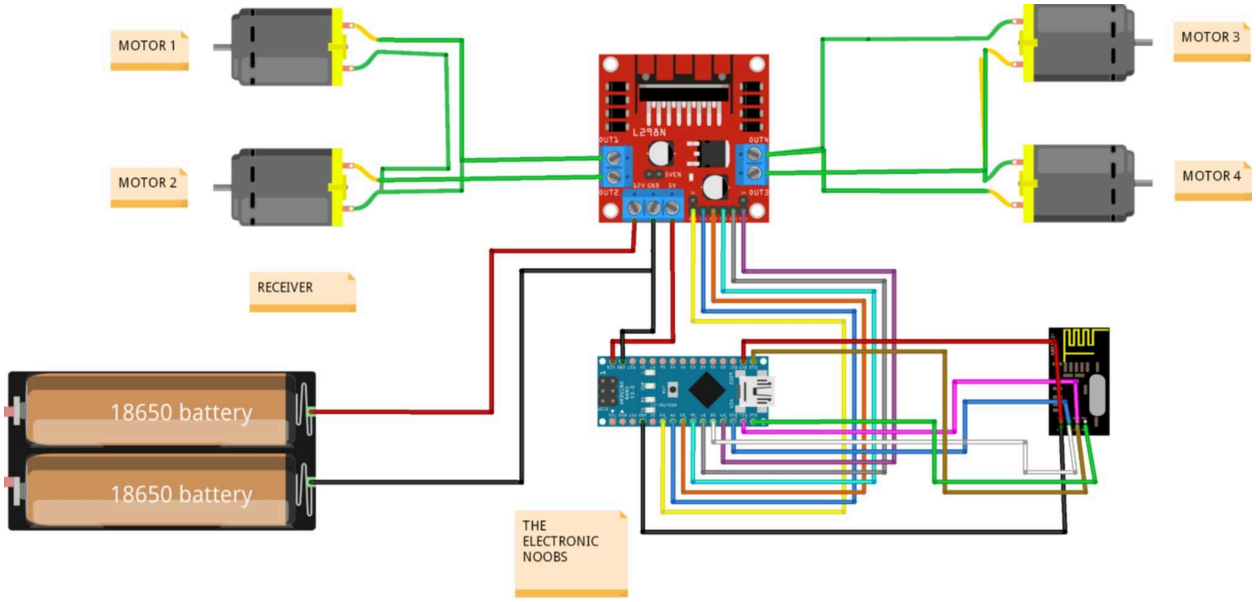


VII. Schematic:

Transmitter:



Receiver:



VIII. Data and Results

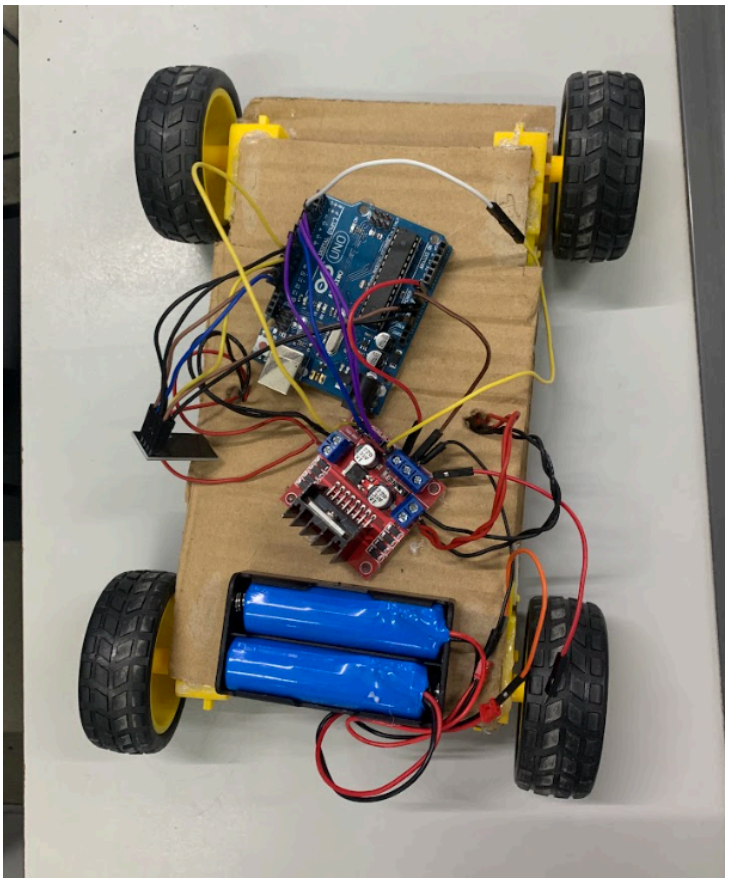
Test	Action	Result
1	Move Forward	Successful
2	Move Left	Successful
3	Move Backward	Successful
4	Move Forward	Successful
5	Move Left	Successful
6	Move Right	Successful
7	Move Right	Successful
8	Move Forward	Successful
9	Move Left	Successful
10	Move Left	Successful

11	Move Backward	Successful
12	Move Backward	Successful
13	Move Left	Successful
14	Move Left	Successful
15	Move Left	Successful
16	Stopped	Successful
17	Move Forward	Delay
18	Forward	Successful
19	Move Right	Successful
20	Move Left	Successful
21	Stopped	Successful
22	Move Forward	Delay
23	Move Right	Successful
24	Move Right	Successful
25	Move Right	Successful
26	Stopped	Successful
27	Move Forward	Delay
28	Move Forward	Successful
29	Move Left	Successful
30	Stopped	Successful

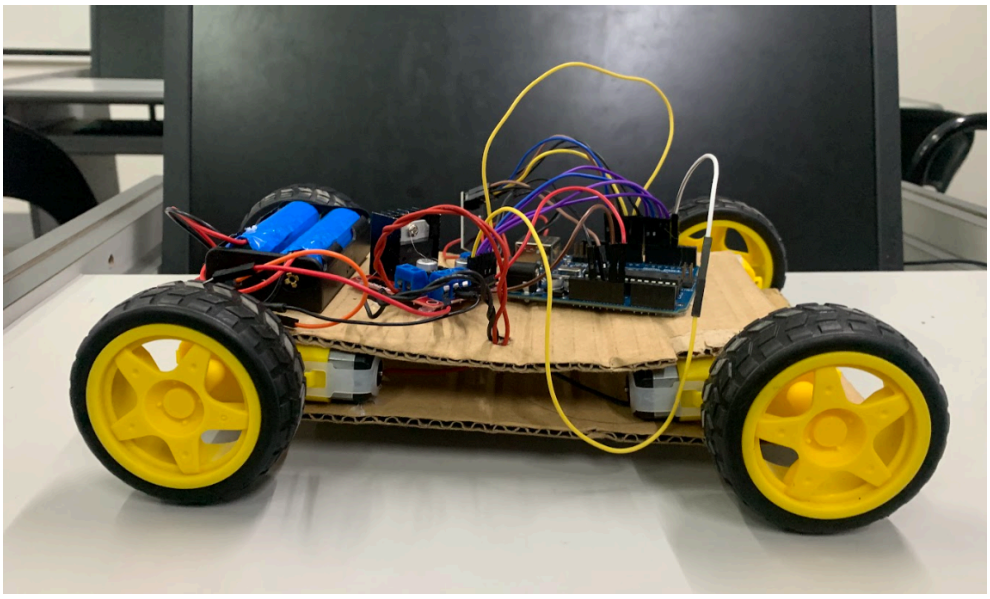
IX. Image of Prototype & Testing

Without case:

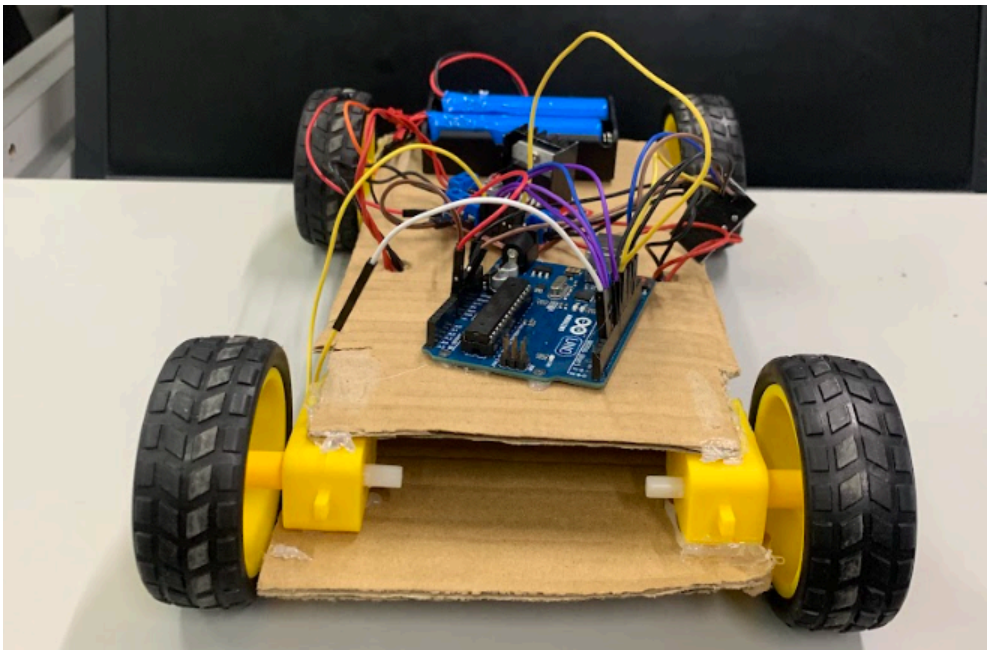
Top view:



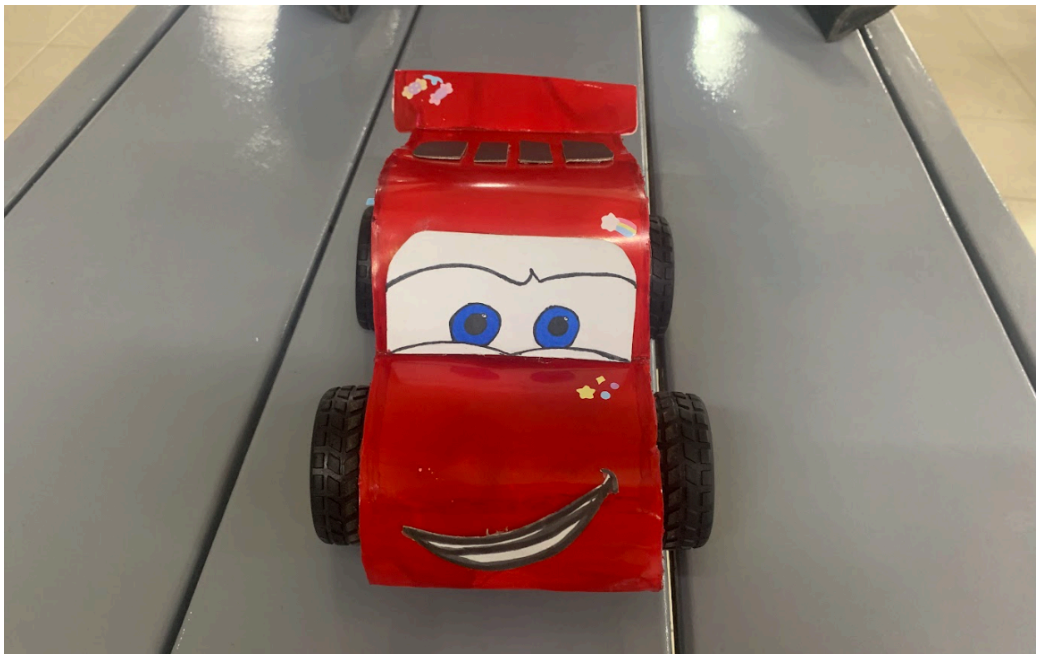
Side view:



Back View:



With Case:



TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
1338 Arlegui Street, Quiapo, Manila
DEPARTMENT OF COMPUTER ENGINEERING

X. Source Code:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include "Wire.h"
#include "MPU6050.h"

#define CSN_GPIO 10
#define CE_GPIO 8

// RF24 Configuration
RF24 radio(CE_GPIO, CSN_GPIO);
const byte Address[6] = "00001";

// MPU6050 Configuration
MPU6050 accelgyro;

// Pins and constants
#define TX_ENABLE_KEY 2
#define TX_LED 3

int16_t ax, ay, az;
int16_t gx, gy, gz;
unsigned char Tx_command = 0, Speed_index = 0;
unsigned char Tx_Array[2];
bool Tx_Enable_Flag = false;

void setup() {
  Serial.begin(115200);
  pinMode(TX_ENABLE_KEY, INPUT_PULLUP);
  pinMode(TX_LED, OUTPUT);

  // Initialize RF24
  radio.begin();
  radio.openWritingPipe(Address);
  radio.setPALevel(RF24_PA_MAX);
  radio.stopListening();
```

```
  // Initialize MPU6050
  Wire.begin();
  accelgyro.initialize();
  if (!accelgyro.testConnection()) {
          Serial.println("MPU6050 connection failed");
          while (1);
  } else {
          Serial.println("MPU6050 connected");
  }
}

void loop() {
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  // Determine direction and speed
  if (ay <= -4000) {
          Tx_command = 1;  // Forward
          Speed_index = map(ay, -16384, -4000, 5, 1);
          Serial.println("Moving Forward");
  } else if (ay >= 4000) {
          Tx_command = 2;  // Backward
          Speed_index = map(ay, 4000, 16384, 1, 5);
          Serial.println("Moving Backward");
  } else if (ax <= -4000) {
          Tx_command = 3;  // Left
          Speed_index = map(ax, -16384, -4000, 5, 1);
          Serial.println("Turning Left");
  } else if (ax >= 4000) {
          Tx_command = 4;  // Right
          Speed_index = map(ax, 4000, 16384, 1, 5);
          Serial.println("Turning Right");
  } else {
          Tx_command = 0;  // Stop
          Speed_index = 0;
          Serial.println("Stopped");
  }

  // Print additional details for debugging
  Serial.print("Command: ");
  Serial.print(Tx_command);
```

```
  Serial.print(", Speed Index: ");
  Serial.println(Speed_index);

  // Transmit data
  Tx_Array[0] = Tx_command;
  Tx_Array[1] = Speed_index;
  radio.write(&Tx_Array, sizeof(Tx_Array));

  delay(100);
}
```

Receiver:

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

// Motor driver pins
#define RIGHT_FORWARD 7
#define RIGHT_BACKWARD 6
#define LEFT_FORWARD 5
#define LEFT_BACKWARD 4
#define enA 3  // PWM pin for left motor
#define enB 9  // PWM pin for right motor

// RF24 Configuration
#define CSN_GPIO 10
#define CE_GPIO 8
RF24 radio(CE_GPIO, CSN_GPIO);
const byte Address[6] = "00001";

// Variables for received data
unsigned char Received_Command = 0, Speed_index = 0;
unsigned char Rx_Array[2];

void setup() {
  Serial.begin(115200);

  // Initialize motor driver pins
  pinMode(RIGHT_FORWARD, OUTPUT);
```

```
  pinMode(RIGHT_BACKWARD, OUTPUT);
  pinMode(LEFT_FORWARD, OUTPUT);
  pinMode(LEFT_BACKWARD, OUTPUT);
  pinMode(enA, OUTPUT);  // Enable pin for left motor
  pinMode(enB, OUTPUT);  // Enable pin for right motor

  // Initialize RF24
  radio.begin();
  radio.openReadingPipe(0, Address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();

  Serial.println("Receiver Ready");
}

void loop() {
  if (radio.available()) {
    // Read the data received
    radio.read(&Rx_Array, sizeof(Rx_Array));
    Received_Command = Rx_Array[0];
    Speed_index = Rx_Array[1];

    Serial.print("Command: ");
    Serial.print(Received_Command);
    Serial.print(", Speed: ");
    Serial.println(Speed_index);

    // Execute the received command
    executeCommand(Received_Command, Speed_index);
  }
}

void executeCommand(unsigned char command, unsigned char speed) {
  // Map speed index (1 to 5) to PWM values (100 to 255)
  int motorSpeed = map(speed, 1, 5, 100, 255);

  switch (command) {
    case 1:  // Forward
      moveForward(motorSpeed);
      break;
```

```
    case 2:  // Backward
      moveBackward(motorSpeed);
      break;
    case 3:  // Left
      turnLeft(motorSpeed);
      break;
    case 4:  // Right
      turnRight(motorSpeed);
      break;
    case 0:  // Stop
    default:
      stopMotors();
      break;
  }
}

void moveForward(int speed) {
  digitalWrite(LEFT_FORWARD, HIGH);
  digitalWrite(RIGHT_FORWARD, HIGH);
  digitalWrite(LEFT_BACKWARD, LOW);
  digitalWrite(RIGHT_BACKWARD, LOW);
  analogWrite(enA, speed);
  analogWrite(enB, speed);
}

void moveBackward(int speed) {
  digitalWrite(LEFT_BACKWARD, HIGH);
  digitalWrite(RIGHT_BACKWARD, HIGH);
  digitalWrite(LEFT_FORWARD, LOW);
  digitalWrite(RIGHT_FORWARD, LOW);
  analogWrite(enA, speed);
  analogWrite(enB, speed);
}

void turnLeft(int speed) {
  digitalWrite(LEFT_BACKWARD, HIGH);
  digitalWrite(RIGHT_FORWARD, HIGH);
  digitalWrite(LEFT_FORWARD, LOW);
  digitalWrite(RIGHT_BACKWARD, LOW);
  analogWrite(enA, speed / 2);  // Slow one motor for turning
```

```
  analogWrite(enB, speed);
}

void turnRight(int speed) {
  digitalWrite(LEFT_FORWARD, HIGH);
  digitalWrite(RIGHT_BACKWARD, HIGH);
  digitalWrite(LEFT_BACKWARD, LOW);
  digitalWrite(RIGHT_FORWARD, LOW);
  analogWrite(enA, speed);
  analogWrite(enB, speed / 2);  // Slow one motor for turning
}

void stopMotors() {
  digitalWrite(LEFT_FORWARD, LOW);
  digitalWrite(LEFT_BACKWARD, LOW);
  digitalWrite(RIGHT_FORWARD, LOW);
  digitalWrite(RIGHT_BACKWARD, LOW);
  analogWrite(enA, 0);
  analogWrite(enB, 0);
}
```

Testing Video Link: **Testing .MOV** and **Testing2.MOV**

XI. Conclusion

This project that our group created, which is a Hand Gesture Controlled Robot, helps us to utilize what we have learned from our Microprocessor course. From interfacing a simple arduino uno to run a DC motors to learning how to use and calibrate a gyroscope for it to be accurate at sending data that will be interpreted by the receiver and send to arduino uno and send a signal to the four DC motors that will serve as the movement mechanism of the robot. Doing all these we've learned the in's and out's of developing a microprocessor device and developed our critical thinking and problem solving that will help us transition to the professional world.