SPECULATION DOCUMENT

GESTURE CONTROLLED UNMANNED GROUND VEHICLE

DESCRIPTION:

Some applications necessitate the use of semi-autonomous or human-controlled robots. Despite the fact that numerous controlled robots use user commands or self-controlled robots that use GPS and sensors, the demand for gesture-controlled robots is increasing for disabled people, military applications, and other reasons.

In recent years, tremendous research has been conducted worldwide to develop robots for surveillance. Military operations and hazardous places surveillance and navigation. A Hand Gesture Controlled Robot is one of the most commonly used motion-controlled robots.

In this project we mainly focus on the controlling of the ground vehicle , efficiently and effortlessly using gesture control technique instead of controlling it through a wireless remote handler.

Prerequisites:

- Understanding of working of DC motors
- Knowledge of L298N motor driver
- Arduino software and Arduino IDE
- Knowledge of MPU6050 sensor

- Knowledge of ESP8266 (Node MCU)
- Integration of L298N with Microcontroller.
- Transmission and receiving of of data through Node MCU

Setting Up Arduino & Processing IDE :

Here are the details about how we set up the environment:

> To program your boards, you need an IDE to write your code. For beginners, we recommend using Arduino IDE. While it's not the best IDE, it works well and is simple and intuitive to use for beginners.

Downloading Arduino IDE:

To download the Arduino IDE, visit the following URL:

- https://www.arduino.cc/en/Main/Software

Running Arduino IDE

Grab the folder you've just downloaded and unzip it. Run the executable file called *arduino.exe*

Installing the ESP8266 NodeMCU in Arduino IDE

To be able to program the ESP8266 NodeMCU using Arduino IDE, you need to add support for the ESP8266 boards. Follow the next steps:

1. Go to **File** > **Preferences**.
2. Enter the following into the "*Additional Board Manager URLs*" field.

https://arduino.esp8266.com/stable/package_esp8266com_index.json

3. Open the **Boards Manager**. Go to **Tools** > **Board** >**Boards Manager…**
4. Search for **ESP8266** and install the "**ESP8266 by ESP8266 Community**".

That's it. It will be installed after a few seconds.

After this, restart your Arduino IDE.

Then, go to **Tools** > **Board** and check that you have ESP8266 boards available.

Now, you're ready to start programming your ESP8266 using Arduino IDE.

Materials Required:

- 2 NodeMCU modules
- MPU6050 sensor
- L298N motor driver
- 2 DC motors
- Caster wheel
- Chassis
- Battery

Construction:

1. First, connect the MPU6050 sensor to the sender NodeMCU module. The sensor will measure the angles of the hand gestures and transmit them to the sender NodeMCU.
2. Connect the sender NodeMCU module to the receiver NodeMCU module using a wireless communication protocol such as WiFi.
3. Connect the receiver NodeMCU module to the L298N motor driver. The motor driver will control the speed and direction of the DC motors.
4. Mount the two DC motors onto the chassis and connect them to the motor driver.
5. Connect a caster wheel to the front of the chassis to provide stability.
6. Connect a battery to power the UGV.

Working:

A hand gesture-controlled robot is created in this project using the MPU6050, which is a 3-axis accelerometer and 3-axis gyroscope sensor.

The project is separated into two sections: transmitter and receiver.

Transmitter: An MPU6050 Sensor and a Node MCU is used at the transmitter end. We have used NodeMCU as our microcontroller Instead of Arduino because of its WiFi capabilities. The I2C interface is used to communicate between the microcontroller and the MPU6050 Sensor.

The MPU6050 IMU has both 3-Axis accelerometer and 3-Axis gyroscope integrated on a single chip.The gyroscope measures rotational velocity or

rate of change of the angular position over time, along the X, Y and Z axis. The outputs of the gyroscope are in degrees per second, so in order to get the angular position we just need to integrate the angular velocity. On the other hand, the MPU6050 accelerometer measures acceleration, it can measure gravitational acceleration along the 3 axes and using some trigonometry math we can calculate the angle at which the sensor is positioned. So, if we fuse, or combine the accelerometer and gyroscope data we can get very accurate information about the sensor orientation.

Node MCU transmits the data i.e. calculated angles over WiFi to the Receiver end. This eliminates the requirement of an additional RF transmitter.

Receiver: On the receiver end, there is a chassis with two DC motors that are controlled using an L298N motor driver. The L298N chip is equipped with two standard H-bridges, which are suitable for driving a pair of DC motors, making it an excellent choice for constructing a two-wheeled robotic platform. With a capacity of 2A continuous current per channel, the L298N motor driver is well-suited for most DC motors. The direction of the motors is controlled by digital output signals, while the speed is adjusted using PWM signals. The NodeMCU, situated on the chassis, receives data from the transmitter and sends appropriate signals to the motor driver based on the received data thus controlling the speed and direction of rotation of motors.

Arduino Code :

Code to get Mac Address:

```
#include "WiFi.h"

void setup(){

  Serial.begin(115200);

  WiFi.mode(WIFI_MODE_STA);

  Serial.println(WiFi.macAddress());

}

void loop(){

}
```

Sender's code :

```
#include<Wire.h>

#include <ESP8266WiFi.h>

#include <espnow.h>

uint8_t broadcastAddress1[] = {0xDC, 0x4F, 0x22, 0x76, 0x0D, 0xBC};

const int MPU = 0x68;

float AccX, AccY, AccZ;

float GyroX, GyroY, GyroZ;

float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;

float roll, pitch, yaw;

float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;

float elapsedTime, currentTime, previousTime;
```

```
typedef struct test_struct {
float r;
float p;
float y;
} test_struct;
test_struct Angles;


void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  char macStr[18];
  Serial.print("Packet to:");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
         mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4],
mac_addr[5]);
  Serial.print(macStr);
  Serial.print(" send status: ");
  if (sendStatus == 0){
    Serial.println("Delivery success");
  }
  else{
    Serial.println("Delivery fail");
  }
}


void setup() {
```

```
  Serial.begin(9600);

  Wire.begin();

  Wire.beginTransmission(MPU);

  Wire.write(0x6B);

  Wire.write(0x00);

  Wire.endTransmission(true);



   /*

  // We can Configure Accelerometer Sensitivity - Full Scale Range
(default +/- 2g)

  Wire.beginTransmission(MPU);

  Wire.write(0x1C);

  Wire.write(0x10);

  Wire.endTransmission(true);

  // We can Configure Gyro Sensitivity - Full Scale Range (default +/- 250
deg/s)

  Wire.beginTransmission(MPU);

  Wire.write(0x1B);

  Wire.write(0x10);

  Wire.endTransmission(true);

  delay(20);

  */



  WiFi.mode(WIFI_STA);
```

```
  WiFi.disconnect();
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);
  esp_now_register_send_cb(OnDataSent);
  esp_now_add_peer(broadcastAddress1, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);
}
void loop() {
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true);
  AccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
  AccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
  AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
  accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI);
  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 /
PI);
  previousTime = currentTime;
  currentTime = millis();
  elapsedTime = (currentTime - previousTime) / 1000;
```

```
  Wire.beginTransmission(MPU);
  Wire.write(0x43);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true);
  GyroX = (Wire.read() << 8 | Wire.read()) / 131.0;
  GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
  GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;
  gyroAngleX = gyroAngleX + GyroX * elapsedTime;
  gyroAngleY = gyroAngleY + GyroY * elapsedTime;
  yaw =  yaw + GyroZ * elapsedTime;
  roll = 0.96 * gyroAngleX + 0.04 * accAngleX;
  pitch = 0.96 * gyroAngleY + 0.04 * accAngleY;
  Angles.r = roll;
  Angles.p = pitch;
  Angles.y = yaw;
  esp_now_send(0, (uint8_t *) &Angles, sizeof(Angles));
  Serial.println(roll);
  Serial.println(pitch);
  Serial.println(yaw);
  delay(1000);
}
```

Receiver's code :

```
#include <SPI.h>

#include <ESP8266WiFi.h>

#include <espnow.h>

uint8_t pin1 = 5;

uint8_t pin2 = 4;

uint8_t pin3 = 0;

uint8_t pin4 = 2;

uint8_t pin5 = 12;

uint8_t pin6 = 13;


typedef struct  test_struct {

  float r;

  float p;

  float y;

} test_struct;


test_struct Angles;


void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {

  memcpy(&Angles, incomingData, sizeof(Angles));

}
```

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  esp_now_set_self_role(ESP_NOW_ROLE_SLAVE);
  esp_now_register_recv_cb(OnDataRecv);

  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);
  pinMode(pin3, OUTPUT);
  pinMode(pin4, OUTPUT);
  pinMode(pin5, OUTPUT);
  pinMode(pin6, OUTPUT);

  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, LOW);
```

```
}
void right() {
 digitalWrite(pin1, HIGH);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, LOW);
  delay(500);
}
void backward() {
  digitalWrite(pin2, HIGH);
  digitalWrite(pin1, LOW);
  digitalWrite(pin4, HIGH);
  digitalWrite(pin3, LOW);
  for (int i = 0; i < 256; i++) {
    analogWrite(pin5, i);
    analogWrite(pin6, i);
    delay(5);
  }
  delay(200);
}
void forward() {
  digitalWrite(pin1, HIGH);
  digitalWrite(pin2, LOW);
```

```
  digitalWrite(pin3, HIGH);
  digitalWrite(pin4, LOW);
  for (int i = 0; i < 256; i++) {
    analogWrite(pin5, i);
    analogWrite(pin6, i);
    delay(5);
  }
  delay(200);
}
void left() {
 digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, HIGH);
  digitalWrite(pin4, LOW);
  delay(500);
}
void stop() {
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, LOW);
  delay(1000);
}
```

```
void loop() {
if (Angles.p < -17) {
   forward();
 }
 if (Angles.p > 20) {
   backward();
 }
 if (Angles.r > 30) {
   left();
 }
 if (Angles.r < -30) {
   right();
 }
 if(Angles.p >= -17 && Angles.p <= 20 && Angles.r <= 30 && Angles.r >= -30
)
 {
   stop();
 }
Serial.println(Angles.r);
Serial.println(Angles.p);
}
```

Challenges Faced

- **Wireless Communication Issues**: We were not able to send and receive data from NRF24L01 transmitter and receiver as originally planned and hence we shifted to NodeMCU thus eliminating the need of Arduino boards from the project.
- **Calibration of MPU6050**: The MPU6050 sensor requires proper calibration for accurate measurement of angles. To get more accurate angles, we fused the data from the accelerometer and gyroscope. Unfortunately, even after employing this technique, the desired outcomes were not achieved with the MPU.
- **NodeMCU Microcontroller**: The board installation may be hectic sometimes as it is not automatically detected on ports and needs to be reset.

References:

https://randomnerdtutorials.com/esp-now-esp8266-nodemcu-arduino-ide/

https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

Future Evolution

This bot can be integrated with LIDAR and can be used for navigation and mapping. The aim of the mapping would be to explore the hazardous places or the surveillance. This can also help in disaster affected remote areas.