Stabilizer สำหรับผู้ป่วยรูมาตอยส์

จัดทำโดย

1. นายกันตพงศ์ เปรมโยธิน 65070502202
2. นายชุติพนธ์ ลิขิตปริญญา 65070502204

เสนอ

อาจารย์ ชนากานต์ แคล้วอ้อม

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา MCE 242 COMPUTER SYSTEMS AND INTERFACING

ภาคเรียนที่ 2 ปีการศึกษา 2566 สาขาวิชาวิศวกรรมเมคคาทรอนิกส์

ภาควิชาวิศวกรรมอุตสาหการและเมคคาทรอนิกส์

คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

MCE242 – 2024 Project Report

ผู้จัดทำ: นายกันตพงศ์ เปรมโยธิน รหัส 65070502202 MCE

นายชุติพนธ์ ลิขิตปริญญา รหัส 65070502204 MCE

Project Name: Stabilizer สำหรับผู้ป่วยโรครูมาตอยส์

ที่มาและความสำคัญ

เนื่องจากผู้ป่วยโรคข้ออักเสบรูมาตอยส์นั้นมักจะมีอาการเจ็บบริเวณข้อต่อของนิ้วมือและทำให้มือไม่มีแรงหยิบจับสิ่งของหรือเมื่อหยิบแล้วอาจเกิดอาการสั่น ทางผู้จัดทำได้เล็งเห็นถึงปัญหานี้ จึงได้ออกแบบเครื่องมือสำหรับผู้ที่มีอาการสั่นเพื่อให้สามารถจับช้อนได้อย่างมั่นคง

วัตถุประสงค์

1. แก้ปัญหาอาการมือสั่นของผู้ป่วยโรครูมาตอยส์
2. ศึกษาการทำงานของ MPU6050 และ Stepper motor
3. ศึกษาการทำงานของระบบควบคุมแบบ PID
4. ศึกษาการควบคุมระบบ ด้วยโปรแกรม BLYNK

ขอบเขต

1. ทั้ง 3 แกนเอียงไม่เกิน 180 องศา

วัสดุและอุปกรณ์ที่ใช้ในการสร้าง Stabilizer

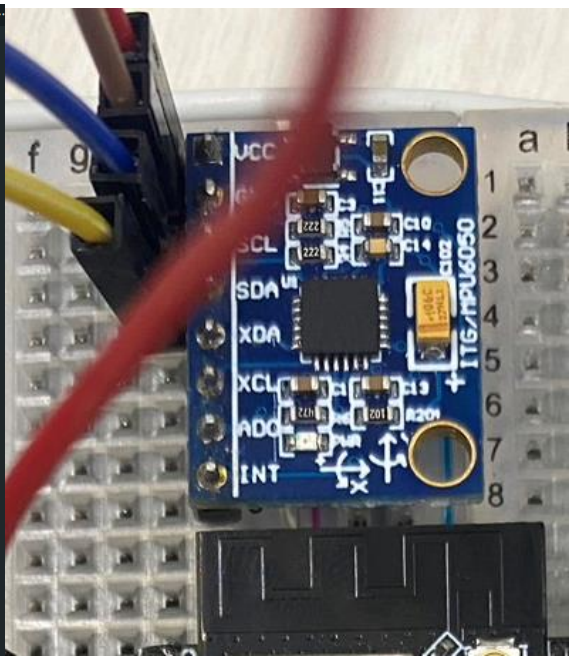
1.	ESP32s	1 ตัว
2.	Arduino UNO R3	1 ตัว
3.	Breadboard	1 บอร์ด
4.	MPU6050	1 ตัว
5.	Stepper Motor (28BYJ-48)	3 ตัว
6.	Step Down Voltage	1 ตัว
7.	9V Battery	1 ก้อน
8.	รางถ่าน 9V	
9.	สายไฟจัมป์เปอร์	
10.	PLA (สำหรับ 3D Printing)	

ขั้นตอนการดำเนินงาน

1. ศึกษาการทำงานของ MPU6050 (Gyroscope module) และ Stepper Motor

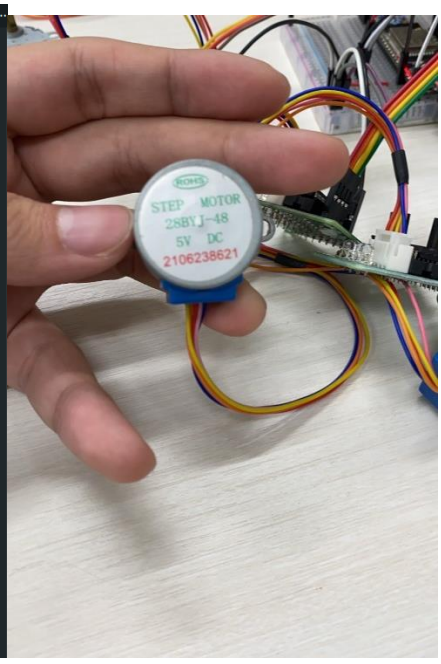
Gyro.ino

```ino
#include "I2Cdev.h"

#include "MPU6050_6Axis_MotionApps20.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

MPU6050 mpu;

#define OUTPUT_READABLE_YAWPITCHROLL

#define INTERRUPT_PIN 2
#define LED_PIN 13
bool blinkState = false;

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

Quaternion q;
VectorInt16 aa;
VectorInt16 aaReal;
VectorInt16 aaWorld;
VectorFloat gravity;
float ypr[3];

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(115200);

    mpu.initialize();
    pinMode(INTERRUPT_PIN, INPUT);
```

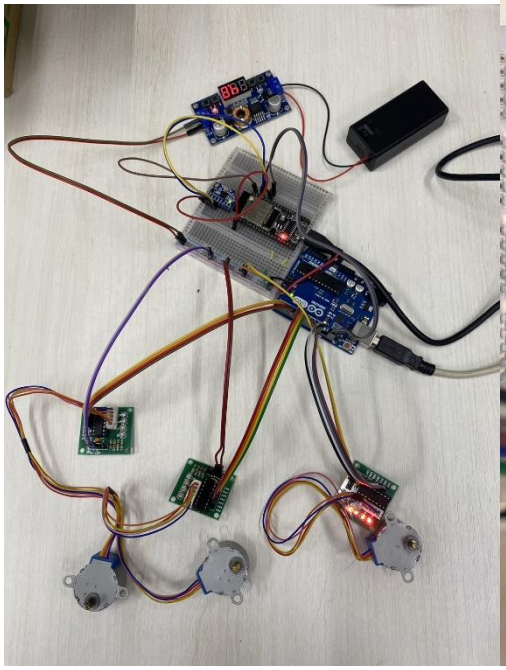


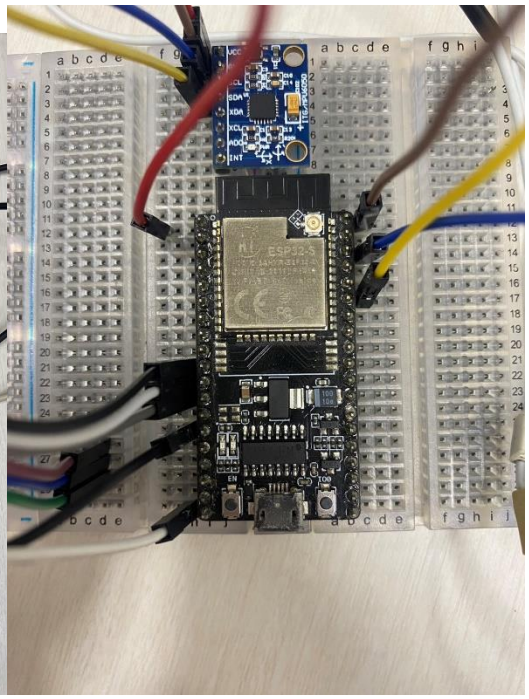
Gyro.ino

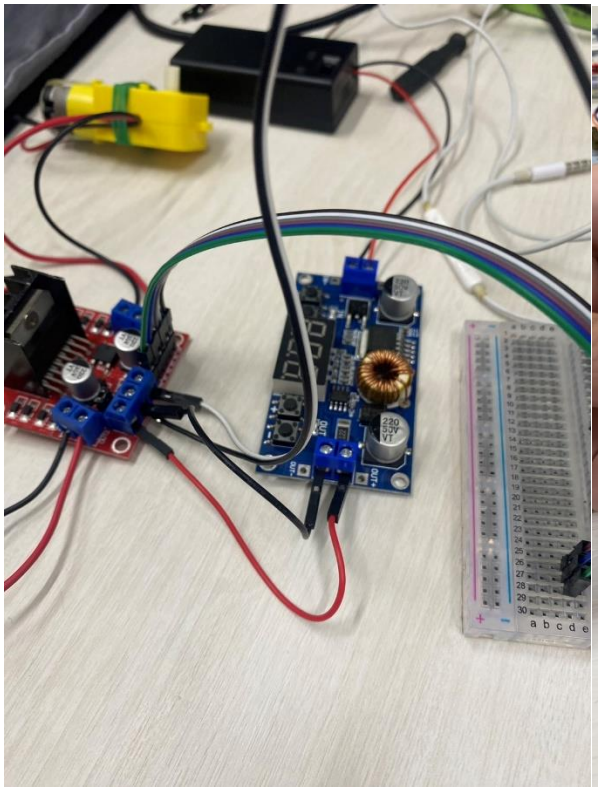
```ino
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif
    Serial.begin(115200);

    mpu.initialize();
    pinMode(INTERRUPT_PIN, INPUT);

    devStatus = mpu.dmpInitialize();

    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1788);


    if (devStatus == 0) {

        mpu.CalibrateAccel(6);
        mpu.CalibrateGyro(6);
        mpu.PrintActiveOffsets();

        mpu.setDMPEnabled(true);

        attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();

        dmpReady = true;

        packetSize = mpu.dmpGetFIFOPacketSize();
    }
}

void loop() {
    if (!dmpReady) return;
    if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {
            mpu.dmpGetQuaternion(&q, fifoBuffer);
            mpu.dmpGetGravity(&gravity, &q);
            mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
            Serial.print("ypr\t");
            Serial.print(ypr[2] * 180/M_PI);
            Serial.print("\t");
            Serial.print(ypr[1] * 180/M_PI);
            Serial.print("\t");
            Serial.println(ypr[0] * 180/M_PI);
    }
}
```



2. ทดสอบการทำงานร่วมกันของ MPU6050 และ Stepper Motor โดยการสื่อสารผ่าน I2C Communication ซึ่งมี ESP32 เป็น MASTER และ Arduino UNO R3 กับ MPU6050 เป็น SLAVE

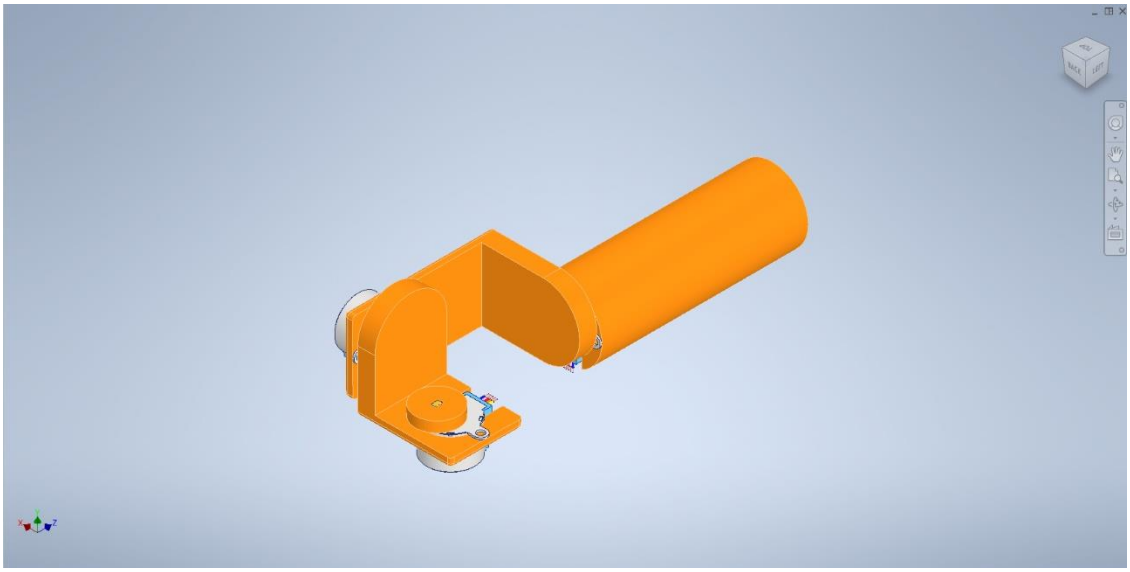




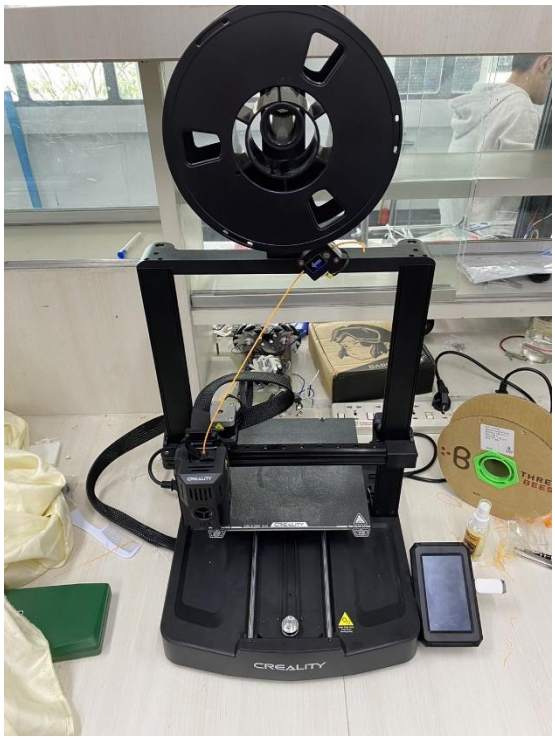


3. ออกแบบหน้า Interface ของโปรแกรม Blynk เพื่อใช้ในการตั้งค่าองศาของอุปกรณ์ Stabilizer

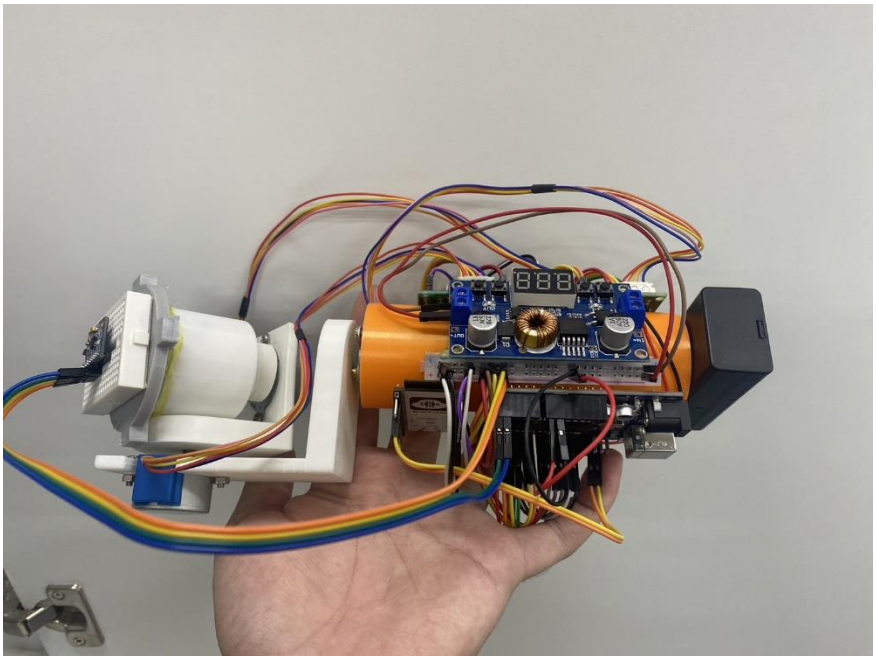
4. ออกแบบโครงสร้างของอุปกรณ์และลองประกอบด้วยโปรแกรม Autodesk Inventor Professional 2024



5. ปริ้นท์ชิ้นงานที่ออกแบบด้วยเครื่องปริ้นท์ 3 มิติ



6. ประกอบชิ้นงานทั้งหมดกับชิ้นงานที่ปริ้นท์มา



ESP32s (MASTER) & MPU6050 Code (SLAVE)

*หมายเหตุ โค้ดทั้งหมดสามารถดาวน์โหลด หรือดูได้จากลิงก์ตามนี้

https://github.com/Gaplnwzaza/MCE242-Project-Stabilizer

```
/*  This MCE242 project was created by 65070502202 Kantapong Premyodin and 65070502204 Chutipon Likitparinya
    The project name is 3-Axis stabilizer with adjust value and PID
    NOTE : First we use esp32 to drive motor but esp32 cannot hold that much voltage so we have to use another board
    and that board is Arduino UNO R3 by I2C communication */

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][]      INCLUDE LIBRARY FOR MPU6050     [][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][]   INCLUDE LIBRARY FOR I2C COMMUNICATION  [][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
#include <Wire.h>

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][]              INCLUDE LIBRARY AND DEFINE BLYNK INFO               [][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
#define BLYNK_TEMPLATE_ID "TMPL6zOYvoFza"
#define BLYNK_TEMPLATE_NAME "Quickstart Template"
#define BLYNK_AUTH_TOKEN "s2cs7kfO603Rcx9-3eNtWDw3zi-LvkDw"
#define BLYNK_PRINT Serial
#include <SPI.h>
#include <Ethernet.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

// Your WiFi credentials
// Set password to "" for open networks
char ssid[] = "GaplnwZaZa";
char pass[] = "gaplnwzaza";

/*    ^^^
      |
      |
      Set to 2.4 Ghz wifi by your | wifi name - ssid | wifi password - pass
*/

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][]      DEFINE FOR I2C COMMUNICATION     [][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
#define slaveAddress 0x08  //you have to assign an 8-bit address to Slave Arduino UNO R3
byte dataArray[3] = { 0 , 0 , 0 };  //To transmit data to Slave using I2C the data type must be byte data for stable transmission

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][]      DEFINE FOR MPU - QUATERNION      [][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL
bool dmpReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];
Quaternion q;
VectorInt16 aa;
VectorInt16 aaReal;
VectorInt16 aaWorld;
VectorFloat gravity;
float ypr[3] = { 0 , 0 , 0 };
float ROW, PITCH, YAW;

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][]      DEFINE FOR MOTOR CONTROL PIN     [][][][][][][][][][][][][][][]
```

```
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
// #define MOTORA_EN 2  // PWM pin for controlling motor ROW axis speed
// #define MOTORA_IN1 4  // Direction control pin +
// #define MOTORA_IN2 2  // Direction control pin -
// #define MOTORB_EN 2   // PWM pin for controlling motor PITCH axis speed
// #define MOTORB_IN1 0   // Direction control pin +
// #define MOTORB_IN2 4   // Direction control pin -
// #define MOTORC_EN 2   // PWM pin for controlling motor YAW axis speed
// #define MOTORC_IN1 0   // Direction control pin +
// #define MOTORC_IN2 2   // Direction control pin -

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][]       DEFINE FOR PID CONTROL       [][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
unsigned long Present_time, Previous_time;
float Delta_time;
int Present_state, Next_state;

float Present_error_ROW, Intergral_error_ROW, Devirative_error_ROW, Previous_error_ROW;
float Present_error_PITCH, Intergral_error_PITCH, Devirative_error_PITCH, Previous_error_PITCH;
float Present_error_YAW, Intergral_error_YAW, Devirative_error_YAW, Previous_error_YAW;

float out_ROW;
float out_PITCH;
float out_YAW;

float set_point_ROW = 0;
float set_point_PITCH = 0;
float set_point_YAW = 0;
/*    ^^^
      |
      |
      For adjust the position when you don't want to connected with blynk slider object
*/

float Min_out = -128;
float Max_out = 127;
float kp = 12;//10
float ki = 0.009;//0.008
float kd = 90;//75
/*    ^^^
      |
      |
      For Optimize PID CONTROL
*/

void setup() {
//Define for MPU and I2C communication
  Wire.begin();
  Wire.setClock(400000);

  Serial.begin(115200);

  mpu.initialize();

  devStatus = mpu.dmpInitialize();

  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788);

  if (devStatus == 0) {
    mpu.CalibrateAccel(6);
    mpu.CalibrateGyro(6);
    mpu.PrintActiveOffsets();
```

```
    mpu.setDMPEnabled(true);
    mpuIntStatus = mpu.getIntStatus();
    packetSize = mpu.dmpGetFIFOPacketSize();
  }

//Connected with blynk
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

//Define OUTPUT for 3-axis motor pin
  // pinMode(MOTORA_EN, OUTPUT);
  // pinMode(MOTORA_IN1, OUTPUT);
  // pinMode(MOTORA_IN2, OUTPUT);
  // pinMode(MOTORB_EN, OUTPUT);
  // pinMode(MOTORB_IN1, OUTPUT);
  // pinMode(MOTORB_IN2, OUTPUT);
  // pinMode(MOTORC_EN, OUTPUT);
  // pinMode(MOTORC_IN1, OUTPUT);
  // pinMode(MOTORC_IN2, OUTPUT);
}

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][]    RECIEVE VALUE FROM BLYNK SLIDER    [][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
BLYNK_WRITE ( V1 ) {
  set_point_ROW = param.asInt();
}

BLYNK_WRITE ( V2 ) {
  set_point_PITCH = param.asInt();
}

BLYNK_WRITE ( V3 ) {
  set_point_YAW = param.asInt();
}

void loop() {
  Blynk.run();
  if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    Serial.print("Row = ");
    Serial.print(ypr[2] * 180 / M_PI);
    Serial.print(" | Pitch = ");
    Serial.print(ypr[1] * 180 / M_PI);
    Serial.print(" | Yaw = ");
    Serial.print(ypr[0] * 180 / M_PI);

    ROW = ypr[2] * 180 / M_PI;
    PITCH = ypr[1] * 180 / M_PI;
    YAW = ypr[0] * 180 / M_PI;

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][]                PID                  [][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
  Present_time = micros();
  Delta_time = (float)(Present_time  - Previous_time);

//ROW PID
  Present_error_ROW   = set_point_ROW - ROW ;
  Intergral_error_ROW  += Present_error_ROW * Delta_time;
  Devirative_error_ROW  = (Present_error_ROW - Previous_error_ROW)/Delta_time ;
  if (Intergral_error_ROW >= Max_out ) Intergral_error_ROW = Max_out;
  else if(Intergral_error_ROW <= Min_out ) Intergral_error_ROW = Min_out;
  out_ROW = (kp*Present_error_ROW) + (ki*Intergral_error_ROW) + (kd*Devirative_error_ROW) ;
  if (out_ROW >= Max_out ) out_ROW = Max_out;
```

```
  else if (out_ROW <= Min_out ) out_ROW = Min_out;
  Previous_error_ROW = Present_error_ROW;

//PITCH PID
  Present_error_PITCH   = set_point_PITCH - PITCH ;
  Intergral_error_PITCH  += Present_error_PITCH * Delta_time;
  Devirative_error_PITCH  = (Present_error_PITCH - Previous_error_PITCH)/Delta_time ;
  if (Intergral_error_PITCH >= Max_out ) Intergral_error_PITCH = Max_out;
  else if(Intergral_error_PITCH <= Min_out ) Intergral_error_PITCH = Min_out;
  out_PITCH = (kp*Present_error_PITCH) + (ki*Intergral_error_PITCH) + (kd*Devirative_error_PITCH) ;
  if (out_PITCH >= Max_out ) out_PITCH = Max_out;
  else if (out_PITCH <= Min_out ) out_PITCH = Min_out;
  Previous_error_PITCH = Present_error_PITCH;

//YAW PID
  Present_error_YAW   = set_point_YAW - YAW ;
  Intergral_error_YAW  += Present_error_YAW * Delta_time;
  Devirative_error_YAW  = (Present_error_YAW - Previous_error_YAW)/Delta_time ;
  if (Intergral_error_YAW >= Max_out ) Intergral_error_YAW = Max_out;
  else if(Intergral_error_YAW <= Min_out ) Intergral_error_YAW = Min_out;
  out_YAW = (kp*Present_error_YAW) + (ki*Intergral_error_YAW) + (kd*Devirative_error_YAW) ;
  if (out_YAW >= Max_out ) out_YAW = Max_out;
  else if (out_YAW <= Min_out ) out_YAW = Min_out;
  Previous_error_YAW = Present_error_YAW;

  Previous_time  = Present_time;

//OUTPUT = out Print to check if it is in normal condition or not
  Serial.print(" ||| OUT_Row = ");
  Serial.print(out_ROW);
  Serial.print(" ||| OUT_Pitch = ");
  Serial.print(out_PITCH);
  Serial.print(" ||| OUT_Yaw = ");
  Serial.println(out_YAW);

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][]          Send data to Slave          [][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/

  dataArray[0] = out_ROW  ;
  dataArray[1] = out_PITCH  ;
  dataArray[2] = out_YAW  ;
  Wire.beginTransmission(slaveAddress); //Address is queued for checking if the slave is present

    Wire.write(dataArray,3);  //Data bytes are queued in local buffer

  Wire.endTransmission(); //All the above queued bytes are sent to slave on ACK handshaking

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][]            MOTOR CONTROL             [][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
//FOR ROW AXIS
/*
    if (out_ROW > 0 ) {
      // Move the motor forward
      digitalWrite(MOTORA_IN1, HIGH);
      digitalWrite(MOTORA_IN2, LOW);
      analogWrite(MOTORA_EN, abs(out_ROW));
    }
    if (out_ROW < 0 ) {
      // Stop the motor
      digitalWrite(MOTORA_IN1, LOW);
      digitalWrite(MOTORA_IN2, HIGH);
      analogWrite(MOTORA_EN, abs(out_ROW));
    }
    if (out_ROW = 0 ) {
```

```
    // Stop the motor
    digitalWrite(MOTORA_IN1, LOW);
    digitalWrite(MOTORA_IN2, LOW);
    analogWrite(MOTORA_EN, 0);
  }

//FOR PITCH AXIS
  if (out_PITCH > 0 ) {
    // Move the motor forward
    digitalWrite(MOTORB_IN1, HIGH);
    digitalWrite(MOTORB_IN2, LOW);
    analogWrite(MOTORB_EN, abs(out_PITCH));
  }
  if (out_PITCH < 0 ) {
    // Stop the motor
    digitalWrite(MOTORB_IN1, LOW);
    digitalWrite(MOTORB_IN2, HIGH);
    analogWrite(MOTORB_EN, abs(out_PITCH));
  }
  if (out_PITCH = 0 ) {
    // Stop the motor
    digitalWrite(MOTORB_IN1, LOW);
    digitalWrite(MOTORB_IN2, LOW);
    analogWrite(MOTORB_EN, 0);
  }

//FOR YAW AXIS
  if (out_YAW > 0 ) {
    // Move the motor forward
    digitalWrite(MOTORC_IN1, HIGH);
    digitalWrite(MOTORC_IN2, LOW);
    analogWrite(MOTORC_EN, abs(out_YAW));
  }
  if (out_YAW < 0 ) {
    // Stop the motor
    digitalWrite(MOTORC_IN1, LOW);
    digitalWrite(MOTORC_IN2, HIGH);
    analogWrite(MOTORC_EN, abs(out_YAW));
  }
  if (out_YAW = 0 ) {
    // Stop the motor
    digitalWrite(MOTORC_IN1, LOW);
    digitalWrite(MOTORC_IN2, LOW);
    analogWrite(MOTORC_EN, 0);
  }
*/
 }
}
```

Arduino UNO (SLAVE) & 28BYJ-48 Code (ACTUATOR)

*หมายเหตุ โค้ดทั้งหมดสามารถดาวน์โหลด หรือดูได้จากลิงก์ตามนี้

https://github.com/Gaplnwzaza/MCE242-Project-Stabilizer

```
/*  This MCE242 project was created by 65070502202 Kantapong Premyodin and 65070502204 Chutipon Likitparinya
    The project name is 3-Axis stabilizer with adjust value and PID
    NOTE : First we use esp32 to drive motor but esp32 cannot hold that much voltage so we have to use another board
    and that board is Arduino UNO R3 by I2C communication */

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][]     DEFINE FOR I2C COMMUNICATION     [][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
#include <Wire.h>
#define slaveAddress 0x08  //you have to assign an 8-bit address to Slave
int8_t dataArray[3] = { 0 , 0 , 0 };
long raw_ROW, raw_PITCH, raw_YAW;

/*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][]       DEFINE FOR MOTOR CONTROL       [][][][][][][][][][][][][][][]
[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
long map_ROW, map_PITCH, map_YAW;
int state = 0;
int gap_Degree = 15 ;

int RmotorPin1 = 2;
int RmotorPin2 = 3;
int RmotorPin3 = 4;
int RmotorPin4 = 5;
int PmotorPin1 = 6;
int PmotorPin2 = 7;
int PmotorPin3 = 8;
int PmotorPin4 = 9;
int YmotorPin1 = 10;
int YmotorPin2 = 11;
int YmotorPin3 = 12;
int YmotorPin4 = 13;

void setup() {
  Wire.begin(slaveAddress);
  Serial.begin(115200);
  Wire.onReceive(receiveEvent);  //You need to declre it in setup() to receive data from Master

  pinMode(RmotorPin1, OUTPUT);
  pinMode(RmotorPin2, OUTPUT);
  pinMode(RmotorPin3, OUTPUT);
  pinMode(RmotorPin4, OUTPUT);
  pinMode(PmotorPin1, OUTPUT);
  pinMode(PmotorPin2, OUTPUT);
  pinMode(PmotorPin3, OUTPUT);
  pinMode(PmotorPin4, OUTPUT);
  pinMode(YmotorPin1, OUTPUT);
  pinMode(YmotorPin2, OUTPUT);
  pinMode(YmotorPin3, OUTPUT);
  pinMode(YmotorPin4, OUTPUT);
}

void loop() {
  raw_ROW = dataArray[0];
  raw_PITCH = dataArray[1];
  raw_YAW = dataArray[2];

  Serial.print(" || RPY = ");
  Serial.print(dataArray[0], DEC);
  Serial.print(" || ");
  Serial.print(dataArray[1], DEC);
  Serial.print(" || ");
  Serial.print(dataArray[2], DEC);

  if (raw_ROW > gap_Degree) {
    map_ROW = map(raw_ROW, 0, 127, 8, 2);
```

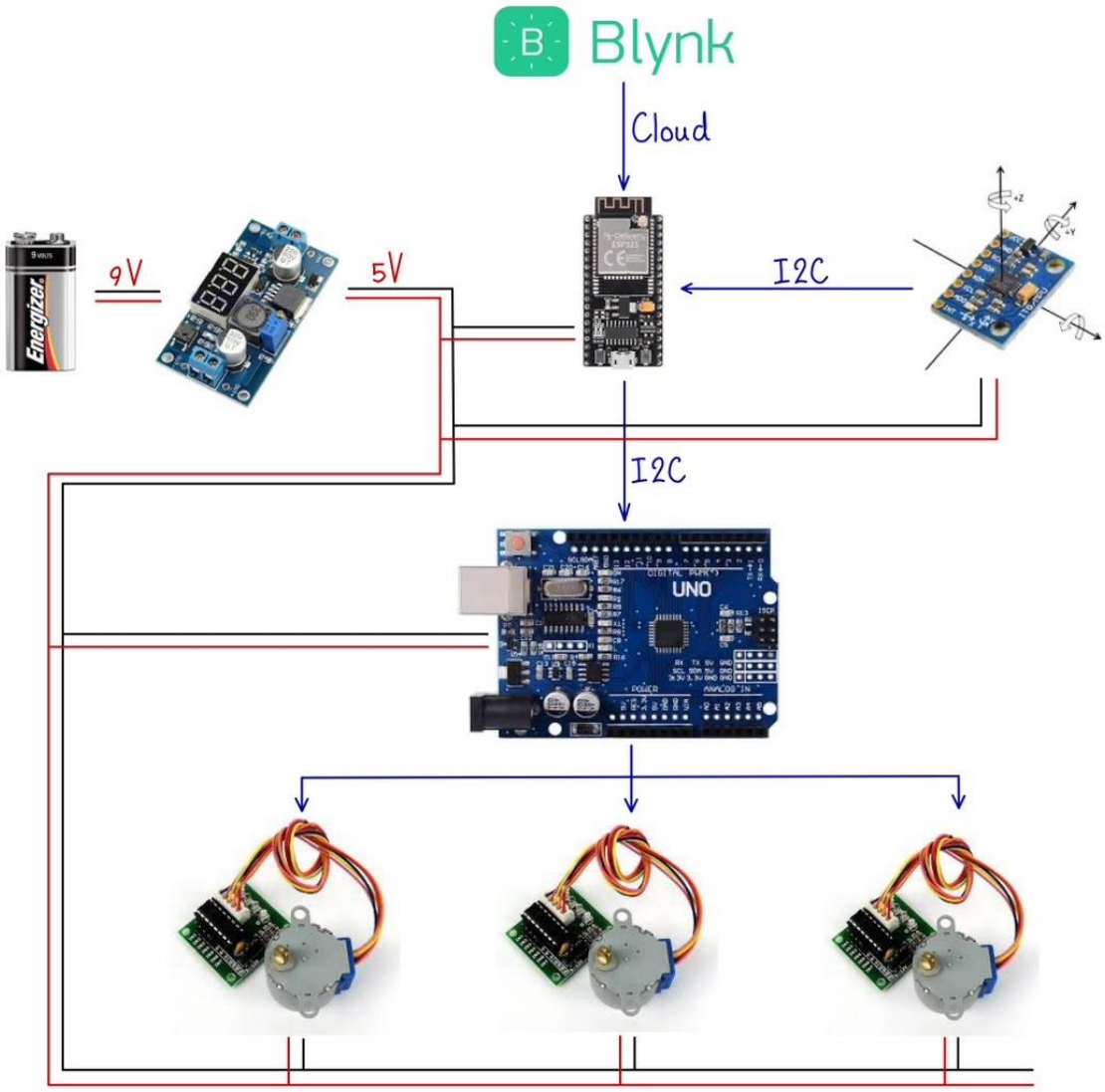
```
  }
  if (raw_ROW < -gap_Degree) {
    map_ROW = map(raw_ROW, -128, 0, -2, -8);
  }
  if (raw_ROW <= gap_Degree && raw_ROW >= -gap_Degree) {
    map_ROW = 0;
  }

  if (raw_PITCH > gap_Degree) {
    map_PITCH = map(raw_PITCH, 0, 127, 8, 2);
  }
  if (raw_PITCH < -gap_Degree) {
    map_PITCH = map(raw_PITCH, -128, 0, -2, -8);
  }
  if (raw_PITCH <= gap_Degree && raw_PITCH >= -gap_Degree) {
    map_PITCH = 0;
  }

  if (raw_YAW > gap_Degree) {
    map_YAW = map(raw_YAW, 0, 127, 8, 2);
  }
  if (raw_YAW < -gap_Degree) {
    map_YAW = map(raw_YAW, -128, 0, -2, -8);
  }
  if (raw_YAW <= gap_Degree && raw_YAW >= -gap_Degree) {
    map_YAW = 0;
  }

  Serial.print(" -- mapRPY = ");
  Serial.print(map_ROW);
  Serial.print(" || ");
  Serial.print(map_PITCH);
  Serial.print(" || ");
  Serial.print(map_YAW);

  //For ROW axis
  if (state == 0) {
    //+ ROW axis
    if (map_ROW > 0) {
      Serial.println(" ||| +ROW");
      digitalWrite(RmotorPin1, HIGH);
      digitalWrite(RmotorPin2, LOW);
      digitalWrite(RmotorPin3, LOW);
      digitalWrite(RmotorPin4, LOW);
      delay(abs(map_ROW));
      digitalWrite(RmotorPin1, LOW);
      digitalWrite(RmotorPin2, HIGH);
      digitalWrite(RmotorPin3, LOW);
      digitalWrite(RmotorPin4, LOW);
      delay(abs(map_ROW));
      digitalWrite(RmotorPin1, LOW);
      digitalWrite(RmotorPin2, LOW);
      digitalWrite(RmotorPin3, HIGH);
      digitalWrite(RmotorPin4, LOW);
      delay(abs(map_ROW));
      digitalWrite(RmotorPin1, LOW);
      digitalWrite(RmotorPin2, LOW);
      digitalWrite(RmotorPin3, LOW);
      digitalWrite(RmotorPin4, HIGH);
      delay(abs(map_ROW));
    }
    //- ROW axis
    if (map_ROW < 0) {
      Serial.println(" ||| -ROW");
      digitalWrite(RmotorPin1, LOW);
      digitalWrite(RmotorPin2, LOW);
```

```
      digitalWrite(RmotorPin3, LOW);
      digitalWrite(RmotorPin4, HIGH);
      delay(abs(map_ROW));
      digitalWrite(RmotorPin1, LOW);
      digitalWrite(RmotorPin2, LOW);
      digitalWrite(RmotorPin3, HIGH);
      digitalWrite(RmotorPin4, LOW);
      delay(abs(map_ROW));
      digitalWrite(RmotorPin1, LOW);
      digitalWrite(RmotorPin2, HIGH);
      digitalWrite(RmotorPin3, LOW);
      digitalWrite(RmotorPin4, LOW);
      delay(abs(map_ROW));
      digitalWrite(RmotorPin1, HIGH);
      digitalWrite(RmotorPin2, LOW);
      digitalWrite(RmotorPin3, LOW);
      digitalWrite(RmotorPin4, LOW);
      delay(abs(map_ROW));
    }
    if (map_ROW == 0) {
      state = 1;
    }
  }
//state = 1
  //For PITCH axis
  if (state == 1) {
    //+ PITCH axis
    if (map_PITCH > 0) {
      Serial.println(" ||| +PITCH");
      digitalWrite(PmotorPin1, HIGH);
      digitalWrite(PmotorPin2, LOW);
      digitalWrite(PmotorPin3, LOW);
      digitalWrite(PmotorPin4, LOW);
      delay(abs(map_PITCH));
      digitalWrite(PmotorPin1, LOW);
      digitalWrite(PmotorPin2, HIGH);
      digitalWrite(PmotorPin3, LOW);
      digitalWrite(PmotorPin4, LOW);
      delay(abs(map_PITCH));
      digitalWrite(PmotorPin1, LOW);
      digitalWrite(PmotorPin2, LOW);
      digitalWrite(PmotorPin3, HIGH);
      digitalWrite(PmotorPin4, LOW);
      delay(abs(map_PITCH));
      digitalWrite(PmotorPin1, LOW);
      digitalWrite(PmotorPin2, LOW);
      digitalWrite(PmotorPin3, LOW);
      digitalWrite(PmotorPin4, HIGH);
      delay(abs(map_PITCH));
    }
    //- PITCH axis
    if (map_PITCH < 0) {
      Serial.println(" ||| -PITCH");
      digitalWrite(PmotorPin1, LOW);
      digitalWrite(PmotorPin2, LOW);
      digitalWrite(PmotorPin3, LOW);
      digitalWrite(PmotorPin4, HIGH);
      delay(abs(map_PITCH));
      digitalWrite(PmotorPin1, LOW);
      digitalWrite(PmotorPin2, LOW);
      digitalWrite(PmotorPin3, HIGH);
      digitalWrite(PmotorPin4, LOW);
      delay(abs(map_PITCH));
      digitalWrite(PmotorPin1, LOW);
      digitalWrite(PmotorPin2, HIGH);
      digitalWrite(PmotorPin3, LOW);
```

```
      digitalWrite(PmotorPin4, LOW);
      delay(abs(map_PITCH));
      digitalWrite(PmotorPin1, HIGH);
      digitalWrite(PmotorPin2, LOW);
      digitalWrite(PmotorPin3, LOW);
      digitalWrite(PmotorPin4, LOW);
      delay(abs(map_PITCH));
    }
    if (map_PITCH == 0) {
      state = 2;
    }
  }
//state = 2
  //For YAW axis
  if (state == 2) {
    //+ YAW axis
    if (map_YAW > 0) {
      Serial.println(" ||| +YAW");
      digitalWrite(YmotorPin1, HIGH);
      digitalWrite(YmotorPin2, LOW);
      digitalWrite(YmotorPin3, LOW);
      digitalWrite(YmotorPin4, LOW);
      delay(abs(map_YAW));
      digitalWrite(YmotorPin1, LOW);
      digitalWrite(YmotorPin2, HIGH);
      digitalWrite(YmotorPin3, LOW);
      digitalWrite(YmotorPin4, LOW);
      delay(abs(map_YAW));
      digitalWrite(YmotorPin1, LOW);
      digitalWrite(YmotorPin2, LOW);
      digitalWrite(YmotorPin3, HIGH);
      digitalWrite(YmotorPin4, LOW);
      delay(abs(map_YAW));
      digitalWrite(YmotorPin1, LOW);
      digitalWrite(YmotorPin2, LOW);
      digitalWrite(YmotorPin3, LOW);
      digitalWrite(YmotorPin4, HIGH);
      delay(abs(map_YAW));
    }
    //- PITCH axis
    if (map_YAW < 0) {
      Serial.println(" ||| -YAW");
      digitalWrite(YmotorPin1, LOW);
      digitalWrite(YmotorPin2, LOW);
      digitalWrite(YmotorPin3, LOW);
      digitalWrite(YmotorPin4, HIGH);
      delay(abs(map_YAW));
      digitalWrite(YmotorPin1, LOW);
      digitalWrite(YmotorPin2, LOW);
      digitalWrite(YmotorPin3, HIGH);
      digitalWrite(YmotorPin4, LOW);
      delay(abs(map_YAW));
      digitalWrite(YmotorPin1, LOW);
      digitalWrite(YmotorPin2, HIGH);
      digitalWrite(YmotorPin3, LOW);
      digitalWrite(YmotorPin4, LOW);
      delay(abs(map_YAW));
      digitalWrite(YmotorPin1, HIGH);
      digitalWrite(YmotorPin2, LOW);
      digitalWrite(YmotorPin3, LOW);
      digitalWrite(YmotorPin4, LOW);
      delay(abs(map_YAW));
    }
    if (map_YAW == 0) {
      state = 0;
    }
```

```
  }
//state = 3
  if (map_ROW != 0 && state != 0) {
    state = 0;
  }
  if (map_PITCH != 0 && state == 2) {
    state = 1;
  }
//Stop State
  if (map_ROW == 0 && map_PITCH == 0 && map_YAW == 0){
    Serial.println(" ||| STOP");
  }
  Serial.print("state = ");
  Serial.print(state);
}

void receiveEvent(int howmany)  //howmany = Wire.write()executed by Master ESP
{
  while (1 < Wire.available()) {
    for (int i = 0; i < howmany; i++) {
      dataArray[i] = Wire.read();
    }
  }
}

```

แผนผังการทำงานของอุปกรณ์ต่างๆ



สรุปผลการทดลอง

Stabilizer สำหรับผู้ป่วยรูมาตอยส์ยังไม่สามารถใช้งานได้จริงเนื่องจากการตอบสนองของ Stepper Motor ที่ใช้ในปัจจุบันนั้นมี Response Time ที่ช้าเกินไป แต่หากการเปลี่ยนองศาไม่ได้เกิดขึ้นอย่างเฉียบพลัน Stabilizer ยังคงสามารถทำงานได้อย่างปกติ

ปัญหาที่พบระหว่างการทดลอง

1. ESP32s ไม่สามารถใช้ในการควบคุมมอเตอร์ 28BYJ-48 ได้
2. ความเร็วการตอบสนองของมอเตอร์ไม่เป็นไปตามที่คิด

วิธีแก้ไข

1. ใช้ ESP32s ในการรับข้อมูลจาก MPU6050 และส่งข้อมูลให้ Arduino UNO เพื่อใช้ในการควบคุมมอเตอร์ 28BYJ-48
2. เปลี่ยนมอเตอร์และโค้ดให้มี Response Time ที่ดีขึ้น

ข้อเสนอแนะ

1. ขณะเริ่มทำงานไม่ควรขยับอุปกรณ์ เพื่อให้ MPU6050 (Gyroscope) ได้ทำการ Calibrate ตัวเองก่อน
2. หากโปรแกรมหยุดทำงาน สังเกตได้จากการที่ Stepper motor หมุนไม่หยุด ให้ทำการกดปุ่ม RESET แล้วเริ่มการทำงานใหม่

วิดีโอสาธิตการทำงาน

https://youtu.be/N8c2kVaJQ_8

ไฟล์โครงงาน

https://github.com/Gaplnwzaza/MCE242-Project-Stabilizer