



Stabilizer สำหรับผู้ป่วยรูมาตอยส์

จัดทำโดย

1. นายกันตพงศ์ เปรมโยธิน 65070502202
2. นายชุตินันท์ ลิขิตปริญญา 65070502204

เสนอ

อาจารย์ ชนาگانต์ 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

นายชุตินันท์ ลิขิตปริญญา รหัส 65070502204 MCE

Project Name: Stabilizer สำหรับผู้ป่วยโรครูมาตอยส์

ที่มาและความสำคัญ

เนื่องจากผู้ป่วยโรคข้ออักเสบรูมาตอยส์นั้นมักจะมีอาการเจ็บบริเวณข้อต่อของนิ้วมือและทำให้มือไม่มีแรงหยิบจับสิ่งของหรือเมื่อยมือแล้วอาจเกิดอาการสั่น ทางผู้จัดทำได้เล็งเห็นถึงปัญหานี้ จึงได้ออกแบบเครื่องมือสำหรับผู้ที่มือมีอาการสั่นเพื่อให้สามารถจับช้อนได้อย่างมั่นคง

วัตถุประสงค์

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

ขอบเขต

1. ทั้ง 3 แกนเอียงไม่เกิน 180 องศา

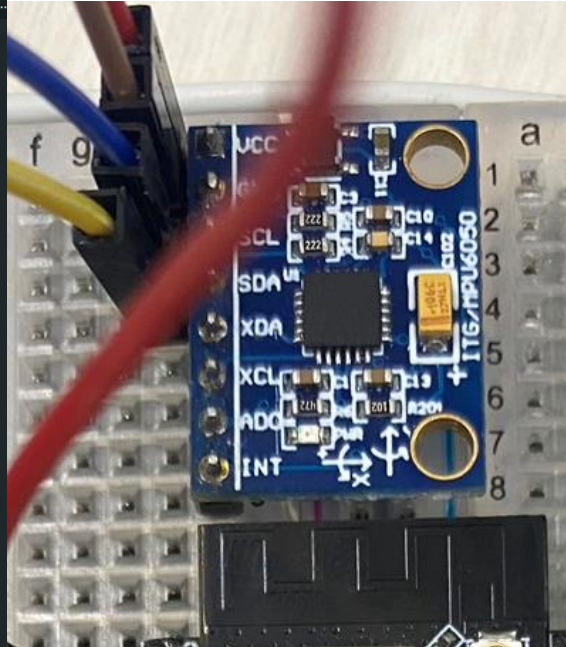
วัสดุและอุปกรณ์ที่ใช้ในการสร้าง Stabilizer

- | | |
|------------------------------|---------|
| 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

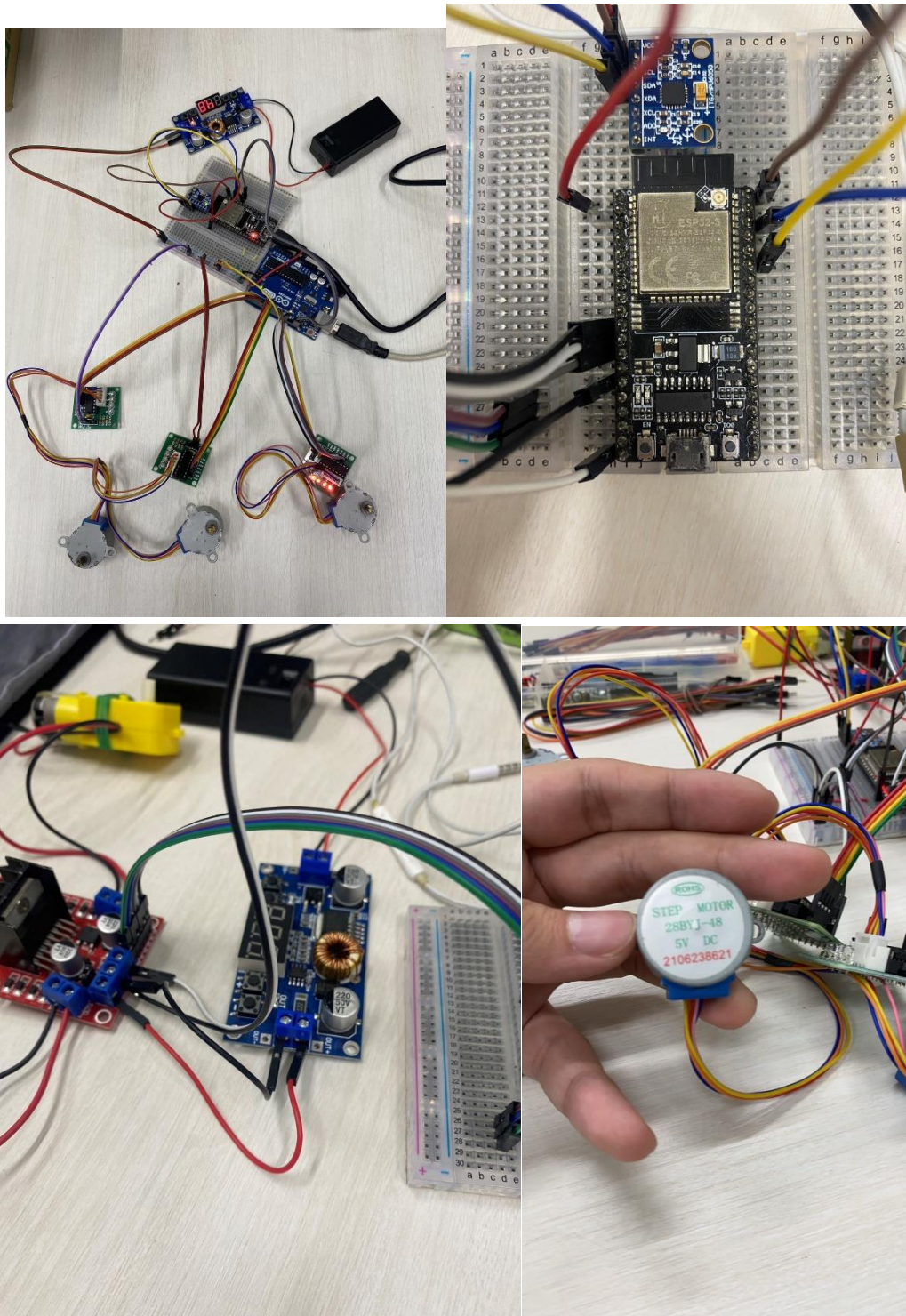
```
Gyro.ino
1 #include "I2Cdev.h"
2 #include "MPU6050_6Axis_MotionApps20.h"
3
4 #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
5 #include "Wire.h"
6 #endif
7
8 MPU6050 mpu;
9
10 #define OUTPUT_READABLE_YAWPITCHROLL
11
12 #define INTERRUPT_PIN 2
13 #define LED_PIN 13
14 bool blinkState = false;
15
16 bool dmpReady = false;
17 uint8_t mpuIntStatus;
18 uint8_t devStatus;
19 uint16_t packetSize;
20 uint16_t fifoCount;
21 uint8_t fifoBuffer[64];
22
23 Quaternion q;
24 VectorInt16 aaReal;
25 VectorInt16 aaWorld;
26 VectorFloat gravity;
27 float ypr[3];
28
29 volatile bool mpuInterrupt = false;
30
31 void dmpDataReady() {
32   mpuInterrupt = true;
33 }
34
35 void setup() {
36   #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
37     Wire.begin();
38     Wire.setClock(400000);
39   #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
40     Fastwire::setup(400, true);
41   #endif
42   Serial.begin(115200);
43   mpu.initialize();
44   pinMode(INTERRUPT_PIN, INPUT);
45 }
```



```
Gyro.ino
41 #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
42   Fastwire::setup(400, true);
43 #endif
44 Serial.begin(115200);
45 mpu.initialize();
46 pinMode(INTERRUPT_PIN, INPUT);
47 devStatus = mpu.dmpInitialize();
48
49 mpu.setXGyroOffset(220);
50 mpu.setYGyroOffset(76);
51 mpu.setZGyroOffset(-85);
52 mpu.setZAccelOffset(1788);
53
54 if (devStatus == 0) {
55   mpu.CalibrateAccel(6);
56   mpu.CalibrateGyro(6);
57   mpu.PrintActiveOffsets();
58   mpu.setDMPEnabled(true);
59   attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
60   mpuIntStatus = mpu.getIntStatus();
61   dmpReady = true;
62   packetSize = mpu.dmpGetFIFOPacketSize();
63 }
64
65 void loop() {
66   if (!dmpReady) return;
67   if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {
68     mpu.dmpGetQuaternion(&q, fifoBuffer);
69     mpu.dmpGetGravity(&gravity, &q);
70     mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
71     Serial.print("ypr:");
72     Serial.print(ypr[2] * 180/M_PI);
73     Serial.print("\t");
74     Serial.print(ypr[1] * 180/M_PI);
75     Serial.print("\t");
76     Serial.print(ypr[0] * 180/M_PI);
77     Serial.println();
78   }
79 }
```



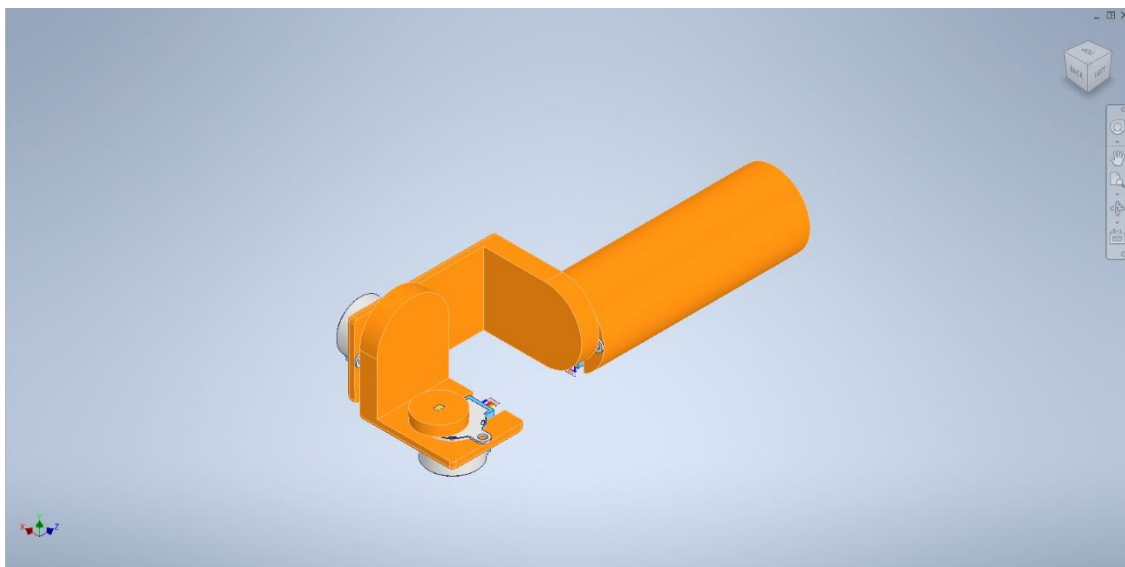
2. ทดสอบการทำงานร่วมกันของ MPU6050 และ Stepper Motor โดยการสื่อสารผ่าน I2C Communication ซึ่งมี ESP32 เป็น MASTER และ Arduino UNO R3 กับ MPU6050 เป็น SLAVE



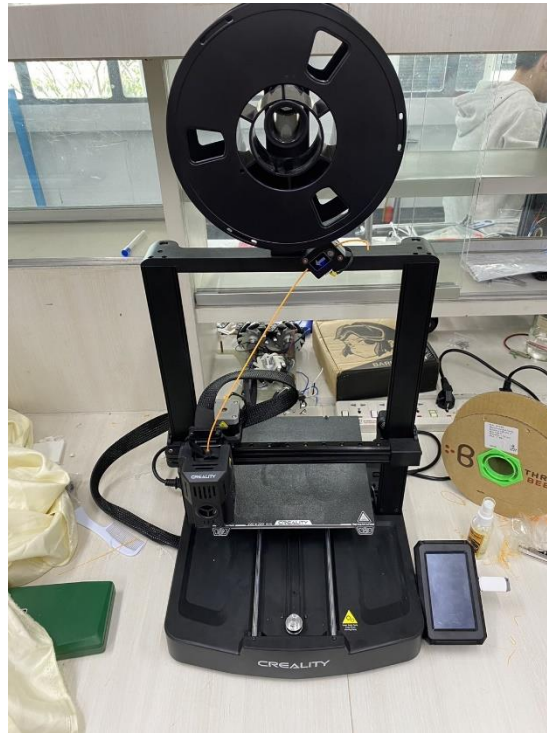
3. ออกแบบหน้า Interface ของโปรแกรม Blynk เพื่อใช้ในการตั้งค่าองศาของอุปกรณ์ Stabilizer



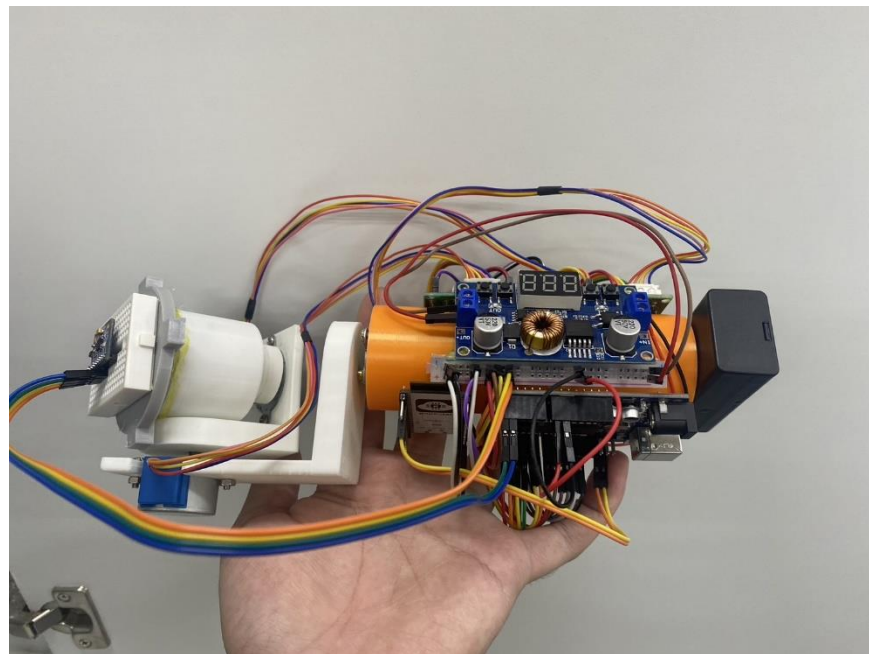
4. ออกแบบโครงสร้างของอุปกรณ์และลองประกอบด้วยโปรแกรม Autodesk Inventor Professional 2024



5. ปริ้นท์ชิ้นงานที่ออกแบบด้วยเครื่องปริ้นท์ 3 มิติ



6. ประกอบชิ้นงานทั้งหมดกับชิ้นงานที่ปริ้นท์มา



*หมายเหตุ โค้ดทั้งหมดสามารถดาวน์โหลด หรือดูได้จากลิงก์ตามนี้

[illegible]

```

67  /*[[]][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
68  // #define MOTORA_EN 2 // PWM pin for controlling motor ROW axis speed
69  // #define MOTORA_IN1 4 // Direction control pin +
70  // #define MOTORA_IN2 2 // Direction control pin -
71  // #define MOTORB_EN 2 // PWM pin for controlling motor PITCH axis speed
72  // #define MOTORB_IN1 0 // Direction control pin +
73  // #define MOTORB_IN2 4 // Direction control pin -
74  // #define MOTORC_EN 2 // PWM pin for controlling motor YAW axis speed
75  // #define MOTORC_IN1 0 // Direction control pin +
76  // #define MOTORC_IN2 2 // Direction control pin -
77
78  /*[[]][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
79  [][][][][][][][][][][][][][] DEFINE FOR PID CONTROL [][][][][][][][][][][][]
80  [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
81  unsigned long Present_time, Previous_time;
82  float Delta_time;
83  int Present_state, Next_state;
84
85  float Present_error_ROW, Integral_error_ROW, Devirative_error_ROW, Previous_error_ROW;
86  float Present_error_PITCH, Integral_error_PITCH, Devirative_error_PITCH, Previous_error_PITCH;
87  float Present_error_YAW, Integral_error_YAW, Devirative_error_YAW, Previous_error_YAW;
88
89  float out_ROW;
90  float out_PITCH;
91  float out_YAW;
92
93  float set_point_ROW = 0;
94  float set_point_PITCH = 0;
95  float set_point_YAW = 0;
96  /*      ^^^
97  |      |      |
98  |      |      |
99  |      |      | For adjust the position when you don't want to connected with blynk slider object
100 */
101
102 float Min_out = -128;
103 float Max_out = 127;
104 float kp = 12;//10
105 float ki = 0.009;//0.008
106 float kd = 90;//75
107 /*      ^^^
108 |      |      |
109 |      |      |
110 |      |      | For Optimize PID CONTROL
111 */
112
113 void setup() {
114 //Define for MPU and I2C communication
115 Wire.begin();
116 Wire.setClock(400000);
117
118 Serial.begin(115200);
119
120 mpu.initialize();
121
122 devStatus = mpu.dmpInitialize();
123
124 mpu.setXGyroOffset(220);
125 mpu.setYGyroOffset(76);
126 mpu.setZGyroOffset(-85);
127 mpu.setZAccelOffset(1788);
128
129 if (devStatus == 0) {
130 mpu.CalibrateAccel(6);
131 mpu.CalibrateGyro(6);
132 mpu.PrintActiveOffsets();

```



```

132 mpu.setDMPEnabled(true);
133 mpuIntStatus = mpu.getIntStatus();
134 packetSize = mpu.dmpGetFIFOPacketSize();
135 }
136 }
137
138 //Connected with blynk
139 Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
140
141 //Define OUTPUT for 3-axis motor pin
142 // pinMode(MOTORA_EN, OUTPUT);
143 // pinMode(MOTORA_IN1, OUTPUT);
144 // pinMode(MOTORA_IN2, OUTPUT);
145 // pinMode(MOTORB_EN, OUTPUT);
146 // pinMode(MOTORB_IN1, OUTPUT);
147 // pinMode(MOTORB_IN2, OUTPUT);
148 // pinMode(MOTORC_EN, OUTPUT);
149 // pinMode(MOTORC_IN1, OUTPUT);
150 // pinMode(MOTORC_IN2, OUTPUT);
151 }
152
153 /*[RECEIVE VALUE FROM BLYNK SLIDER]*/
154 [RECEIVE VALUE FROM BLYNK SLIDER]
155 [RECEIVE VALUE FROM BLYNK SLIDER]
156 BLYNK_WRITE ( V1 ) {
157   set_point_ROW = param.asInt();
158 }
159
160 BLYNK_WRITE ( V2 ) {
161   set_point_PITCH = param.asInt();
162 }
163
164 BLYNK_WRITE ( V3 ) {
165   set_point_YAW = param.asInt();
166 }
167
168 void loop() {
169   Blynk.run();
170   if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) {
171     mpu.dmpGetQuaternion(&q, fifoBuffer);
172     mpu.dmpGetGravity(&gravity, &q);
173     mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
174     Serial.print("Row = ");
175     Serial.print(ypr[2] * 180 / M_PI);
176     Serial.print(" | Pitch = ");
177     Serial.print(ypr[1] * 180 / M_PI);
178     Serial.print(" | Yaw = ");
179     Serial.print(ypr[0] * 180 / M_PI);
180
181     ROW = ypr[2] * 180 / M_PI;
182     PITCH = ypr[1] * 180 / M_PI;
183     YAW = ypr[0] * 180 / M_PI;
184
185     /*[PID]*/
186     [PID]
187     [PID]
188     Present_time = micros();
189     Delta_time = (float)(Present_time - Previous_time);
190
191     //ROW PID
192     Present_error_ROW = set_point_ROW - ROW ;
193     Integral_error_ROW += Present_error_ROW * Delta_time;
194     Devirative_error_ROW = (Present_error_ROW - Previous_error_ROW)/Delta_time ;
195     if (Integral_error_ROW >= Max_out ) Integral_error_ROW = Max_out;
196     else if(Integral_error_ROW <= Min_out ) Integral_error_ROW = Min_out;
197     out_ROW = (kp*Present_error_ROW) + (ki*Integral_error_ROW) + (kd*Devirative_error_ROW) ;
198     if (out_ROW >= Max_out ) out_ROW = Max_out;

```

```

199     else if (out_ROW <= Min_out ) out_ROW = Min_out;
200     Previous_error_ROW = Present_error_ROW;
201
202 //PITCH PID
203 Present_error_PITCH = set_point_PITCH - PITCH ;
204 Integral_error_PITCH += Present_error_PITCH * Delta_time;
205 Devirative_error_PITCH = (Present_error_PITCH - Previous_error_PITCH)/Delta_time ;
206 if (Integral_error_PITCH >= Max_out ) Integral_error_PITCH = Max_out;
207 else if (Integral_error_PITCH <= Min_out ) Integral_error_PITCH = Min_out;
208 out_PITCH = (kp*Present_error_PITCH) + (ki*Integral_error_PITCH) + (kd*Devirative_error_PITCH) ;
209 if (out_PITCH >= Max_out ) out_PITCH = Max_out;
210 else if (out_PITCH <= Min_out ) out_PITCH = Min_out;
211 Previous_error_PITCH = Present_error_PITCH;
212
213 //YAW PID
214 Present_error_YAW = set_point_YAW - YAW ;
215 Integral_error_YAW += Present_error_YAW * Delta_time;
216 Devirative_error_YAW = (Present_error_YAW - Previous_error_YAW)/Delta_time ;
217 if (Integral_error_YAW >= Max_out ) Integral_error_YAW = Max_out;
218 else if (Integral_error_YAW <= Min_out ) Integral_error_YAW = Min_out;
219 out_YAW = (kp*Present_error_YAW) + (ki*Integral_error_YAW) + (kd*Devirative_error_YAW) ;
220 if (out_YAW >= Max_out ) out_YAW = Max_out;
221 else if (out_YAW <= Min_out ) out_YAW = Min_out;
222 Previous_error_YAW = Present_error_YAW;
223
224 Previous_time = Present_time;
225
226 //OUTPUT = out Print to check if it is in normal condition or not
227 Serial.print(" ||| OUT_Row = ");
228 Serial.print(out_ROW);
229 Serial.print(" ||| OUT_Pitch = ");
230 Serial.print(out_PITCH);
231 Serial.print(" ||| OUT_Yaw = ");
232 Serial.println(out_YAW);
233
234 /*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
235 [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
236 [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
237
238 dataArray[0] = out_ROW ;
239 dataArray[1] = out_PITCH ;
240 dataArray[2] = out_YAW ;
241 Wire.beginTransaction(slaveAddress); //Address is queued for checking if the slave is present
242
243     Wire.write(dataArray,3); //Data bytes are queued in local buffer
244
245 Wire.endTransmission(); //All the above queued bytes are sent to slave on ACK handshaking
246
247 /*[][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
248 [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]
249 [][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][][]*/
250 //FOR ROW AXIS
251 /*
252     if (out_ROW > 0 ) {
253         // Move the motor forward
254         digitalWrite(MOTORA_IN1, HIGH);
255         digitalWrite(MOTORA_IN2, LOW);
256         analogWrite(MOTORA_EN, abs(out_ROW));
257     }
258     if (out_ROW < 0 ) {
259         // Stop the motor
260         digitalWrite(MOTORA_IN1, LOW);
261         digitalWrite(MOTORA_IN2, HIGH);
262         analogWrite(MOTORA_EN, abs(out_ROW));
263     }
264     if (out_ROW == 0 ) {

```

```

265     // Stop the motor
266     digitalWrite(MOTORA_IN1, LOW);
267     digitalWrite(MOTORA_IN2, LOW);
268     analogWrite(MOTORA_EN, 0);
269 }
270
271 //FOR PITCH AXIS
272 if (out_PITCH > 0 ) {
273     // Move the motor forward
274     digitalWrite(MOTORB_IN1, HIGH);
275     digitalWrite(MOTORB_IN2, LOW);
276     analogWrite(MOTORB_EN, abs(out_PITCH));
277 }
278 if (out_PITCH < 0 ) {
279     // Stop the motor
280     digitalWrite(MOTORB_IN1, LOW);
281     digitalWrite(MOTORB_IN2, HIGH);
282     analogWrite(MOTORB_EN, abs(out_PITCH));
283 }
284 if (out_PITCH = 0 ) {
285     // Stop the motor
286     digitalWrite(MOTORB_IN1, LOW);
287     digitalWrite(MOTORB_IN2, LOW);
288     analogWrite(MOTORB_EN, 0);
289 }
290
291 //FOR YAW AXIS
292 if (out_YAW > 0 ) {
293     // Move the motor forward
294     digitalWrite(MOTORC_IN1, HIGH);
295     digitalWrite(MOTORC_IN2, LOW);
296     analogWrite(MOTORC_EN, abs(out_YAW));
297 }
298 if (out_YAW < 0 ) {
299     // Stop the motor
300     digitalWrite(MOTORC_IN1, LOW);
301     digitalWrite(MOTORC_IN2, HIGH);
302     analogWrite(MOTORC_EN, abs(out_YAW));
303 }
304 if (out_YAW = 0 ) {
305     // Stop the motor
306     digitalWrite(MOTORC_IN1, LOW);
307     digitalWrite(MOTORC_IN2, LOW);
308     analogWrite(MOTORC_EN, 0);
309 }
310 */
311 }
312 }

```

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

*หมายเหตุ โค้ดทั้งหมดสามารถดาวน์โหลด หรือดูได้จากลิงก์ตามนี้

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

```

/* This MCE242 project created by 65070502202 Kantapong Premyadin 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 <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);

```

```

67     }
68     if (raw_ROW < -gap_Degree) {
69         map_ROW = map(raw_ROW, -128, 0, -2, -8);
70     }
71     if (raw_ROW <= gap_Degree && raw_ROW >= -gap_Degree) {
72         map_ROW = 0;
73     }
74
75     if (raw_PITCH > gap_Degree) {
76         map_PITCH = map(raw_PITCH, 0, 127, 8, 2);
77     }
78     if (raw_PITCH < -gap_Degree) {
79         map_PITCH = map(raw_PITCH, -128, 0, -2, -8);
80     }
81     if (raw_PITCH <= gap_Degree && raw_PITCH >= -gap_Degree) {
82         map_PITCH = 0;
83     }
84
85     if (raw_YAW > gap_Degree) {
86         map_YAW = map(raw_YAW, 0, 127, 8, 2);
87     }
88     if (raw_YAW < -gap_Degree) {
89         map_YAW = map(raw_YAW, -128, 0, -2, -8);
90     }
91     if (raw_YAW <= gap_Degree && raw_YAW >= -gap_Degree) {
92         map_YAW = 0;
93     }
94
95     Serial.print(" -- mapRPY = ");
96     Serial.print(map_ROW);
97     Serial.print(" || ");
98     Serial.print(map_PITCH);
99     Serial.print(" || ");

```

```

100     Serial.print(map_YAW);
101
102     //For ROW axis
103     if (state == 0) {
104         //+ ROW axis
105         if (map_ROW > 0) {
106             Serial.println(" ||| +ROW");
107             digitalWrite(RmotorPin1, HIGH);
108             digitalWrite(RmotorPin2, LOW);
109             digitalWrite(RmotorPin3, LOW);
110             digitalWrite(RmotorPin4, LOW);
111             delay(abs(map_ROW));
112             digitalWrite(RmotorPin1, LOW);
113             digitalWrite(RmotorPin2, HIGH);
114             digitalWrite(RmotorPin3, LOW);
115             digitalWrite(RmotorPin4, LOW);
116             delay(abs(map_ROW));
117             digitalWrite(RmotorPin1, LOW);
118             digitalWrite(RmotorPin2, LOW);
119             digitalWrite(RmotorPin3, HIGH);
120             digitalWrite(RmotorPin4, LOW);
121             delay(abs(map_ROW));
122             digitalWrite(RmotorPin1, LOW);
123             digitalWrite(RmotorPin2, LOW);
124             digitalWrite(RmotorPin3, LOW);
125             digitalWrite(RmotorPin4, HIGH);
126             delay(abs(map_ROW));
127         }
128         //- ROW axis
129         if (map_ROW < 0) {
130             Serial.println(" ||| -ROW");
131             digitalWrite(RmotorPin1, LOW);
132             digitalWrite(RmotorPin2, LOW);

```

```

133     digitalWrite(RmotorPin3, LOW);
134     digitalWrite(RmotorPin4, HIGH);
135     delay(abs(map_ROW));
136     digitalWrite(RmotorPin1, LOW);
137     digitalWrite(RmotorPin2, LOW);
138     digitalWrite(RmotorPin3, HIGH);
139     digitalWrite(RmotorPin4, LOW);
140     delay(abs(map_ROW));
141     digitalWrite(RmotorPin1, LOW);
142     digitalWrite(RmotorPin2, HIGH);
143     digitalWrite(RmotorPin3, LOW);
144     digitalWrite(RmotorPin4, LOW);
145     delay(abs(map_ROW));
146     digitalWrite(RmotorPin1, HIGH);
147     digitalWrite(RmotorPin2, LOW);
148     digitalWrite(RmotorPin3, LOW);
149     digitalWrite(RmotorPin4, LOW);
150     delay(abs(map_ROW));
151 }
152 if (map_ROW == 0) {
153     state = 1;
154 }
155 }
156 //state = 1
157 //For PITCH axis
158 if (state == 1) {
159     //+ PITCH axis
160     if (map_PITCH > 0) {
161         Serial.println(" ||| +PITCH");
162         digitalWrite(PmotorPin1, HIGH);
163         digitalWrite(PmotorPin2, LOW);
164         digitalWrite(PmotorPin3, LOW);
165         digitalWrite(PmotorPin4, LOW);
166         delay(abs(map_PITCH));
167         digitalWrite(PmotorPin1, LOW);
168         digitalWrite(PmotorPin2, HIGH);
169         digitalWrite(PmotorPin3, LOW);
170         digitalWrite(PmotorPin4, LOW);
171         delay(abs(map_PITCH));
172         digitalWrite(PmotorPin1, LOW);
173         digitalWrite(PmotorPin2, LOW);
174         digitalWrite(PmotorPin3, HIGH);
175         digitalWrite(PmotorPin4, LOW);
176         delay(abs(map_PITCH));
177         digitalWrite(PmotorPin1, LOW);
178         digitalWrite(PmotorPin2, LOW);
179         digitalWrite(PmotorPin3, LOW);
180         digitalWrite(PmotorPin4, HIGH);
181         delay(abs(map_PITCH));
182     }
183     //- PITCH axis
184     if (map_PITCH < 0) {
185         Serial.println(" ||| -PITCH");
186         digitalWrite(PmotorPin1, LOW);
187         digitalWrite(PmotorPin2, LOW);
188         digitalWrite(PmotorPin3, LOW);
189         digitalWrite(PmotorPin4, HIGH);
190         delay(abs(map_PITCH));
191         digitalWrite(PmotorPin1, LOW);
192         digitalWrite(PmotorPin2, LOW);
193         digitalWrite(PmotorPin3, HIGH);
194         digitalWrite(PmotorPin4, LOW);
195         delay(abs(map_PITCH));
196         digitalWrite(PmotorPin1, LOW);
197         digitalWrite(PmotorPin2, HIGH);
198         digitalWrite(PmotorPin3, LOW);

```



```

199     digitalWrite(PmotorPin4, LOW);
200     delay(abs(map_PITCH));
201     digitalWrite(PmotorPin1, HIGH);
202     digitalWrite(PmotorPin2, LOW);
203     digitalWrite(PmotorPin3, LOW);
204     digitalWrite(PmotorPin4, LOW);
205     delay(abs(map_PITCH));
206 }
207 if (map_PITCH == 0) {
208     state = 2;
209 }
210 }
211 //state = 2
212 //For YAW axis
213 if (state == 2) {
214     //+ YAW axis
215     if (map_YAW > 0) {
216         Serial.println(" ||| +YAW");
217         digitalWrite(YmotorPin1, HIGH);
218         digitalWrite(YmotorPin2, LOW);
219         digitalWrite(YmotorPin3, LOW);
220         digitalWrite(YmotorPin4, LOW);
221         delay(abs(map_YAW));
222         digitalWrite(YmotorPin1, LOW);
223         digitalWrite(YmotorPin2, HIGH);
224         digitalWrite(YmotorPin3, LOW);
225         digitalWrite(YmotorPin4, LOW);
226         delay(abs(map_YAW));
227         digitalWrite(YmotorPin1, LOW);
228         digitalWrite(YmotorPin2, LOW);
229         digitalWrite(YmotorPin3, HIGH);
230         digitalWrite(YmotorPin4, LOW);
231         delay(abs(map_YAW));
232         digitalWrite(YmotorPin1, LOW);
233         digitalWrite(YmotorPin2, LOW);
234         digitalWrite(YmotorPin3, LOW);
235         digitalWrite(YmotorPin4, HIGH);
236         delay(abs(map_YAW));
237     }
238     //- PITCH axis
239     if (map_YAW < 0) {
240         Serial.println(" ||| -YAW");
241         digitalWrite(YmotorPin1, LOW);
242         digitalWrite(YmotorPin2, LOW);
243         digitalWrite(YmotorPin3, LOW);
244         digitalWrite(YmotorPin4, HIGH);
245         delay(abs(map_YAW));
246         digitalWrite(YmotorPin1, LOW);
247         digitalWrite(YmotorPin2, LOW);
248         digitalWrite(YmotorPin3, HIGH);
249         digitalWrite(YmotorPin4, LOW);
250         delay(abs(map_YAW));
251         digitalWrite(YmotorPin1, LOW);
252         digitalWrite(YmotorPin2, HIGH);
253         digitalWrite(YmotorPin3, LOW);
254         digitalWrite(YmotorPin4, LOW);
255         delay(abs(map_YAW));
256         digitalWrite(YmotorPin1, HIGH);
257         digitalWrite(YmotorPin2, LOW);
258         digitalWrite(YmotorPin3, LOW);
259         digitalWrite(YmotorPin4, LOW);
260         delay(abs(map_YAW));
261     }
262     if (map_YAW == 0) {
263         state = 0;
264     }

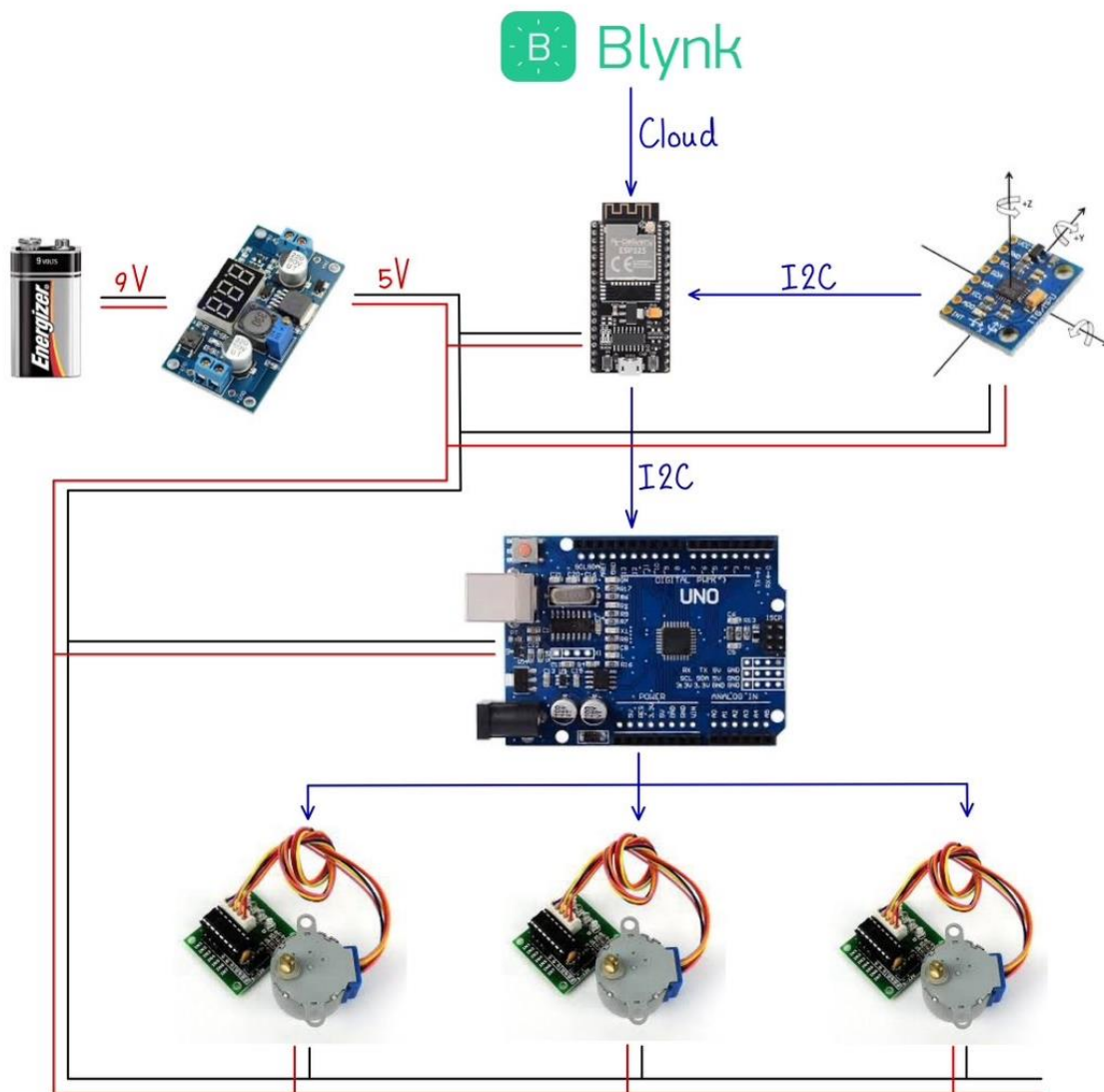
```

```

265 | }
266 //state = 3
267 if (map_ROW != 0 && state != 0) {
268 | state = 0;
269 }
270 if (map_PITCH != 0 && state == 2) {
271 | state = 1;
272 }
273 //Stop State
274 if (map_ROW == 0 && map_PITCH == 0 && map_YAW == 0){
275 | Serial.println(" ||| STOP");
276 }
277 Serial.print("state = ");
278 Serial.print(state);
279 }
280
281 void receiveEvent(int howmany) //howmany = Wire.write()executed by Master ESP
282 {
283 | while (1 < Wire.available()) {
284 | | for (int i = 0; i < howmany; i++) {
285 | | | dataArray[i] = Wire.read();
286 | | }
287 | }
288 }
289

```

แผนผังการทำงานของอุปกรณ์ต่างๆ



สรุปผลการทดลอง

Stabilizer สำหรับผู้ป่วยรูมาตอยส์ยังไม่สามารถใช้งานได้จริงเนื่องจากการตอบสนองของ Stepper Motor ที่ใช้ในปัจจุบันนี้มี 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/N8c2kVaJO_8

ไฟล์โครงงาน

<https://github.com/GapInwzaza/MCE242-Project-Stabilizer>