TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
KHOA ĐIỆN TỬ VIỄN THÔNG

NGUYỄN NGỌC QUANG

BÁO CÁO BÀI TẬP LỚN
BỘ MÔN: HỆ THỐNG LOGIC MỜ

ĐỀ TÀI:
ỨNG DỤNG KỸ THUẬT ĐIỀU KHIỂN MỜ TRONG THIẾT KẾ BỘ CÂN BẰNG SỬ DỤNG ĐỘNG CƠ ENCODER

Giảng viên hướng dẫn : TS.Nguyễn Thị Thanh Vân
Lớp môn học : ELT3111_59

Hà Nội, tháng 12 năm 2024

MỤC LỤC

DANH MỤC HÌNH ẢNH

I. LỜI MỞ ĐẦU

Lời đầu tiên em xin chân thành cảm ơn TS.Nguyễn Thị Thanh Vân đã tạo điều kiện để chúng em có thể ứng dụng các kiến thức đã được giảng dạy trong việc giải quyết các yêu cầu, bài toán thực tiễn. Đây là cơ hội quý giá để chúng em vận dụng và hiểu rõ hơn về cách thiết kế một bộ điều khiển mờ trong thực tế.

Đề tài thiết kế bộ cân bằng được em lựa chọn vì đây là thiết bị có thể được ứng dụng trong nhiều lĩnh vực và có khả năng ứng dụng kỹ thuật điều khiển mờ để giải quyết một số hạn chế khi sử dụng các kỹ thuật điều khiển truyền thống.

Trong nội dung bài tập lớn lần này, em sẽ thực hiện thiết kế, xây dựng một bộ chống rung ứng dụng kỹ thuật điều khiển mờ và nhận xét tính hiệu quả khi ứng dụng kỹ thuật điều khiển mờ trong điều khiển bộ cân bằng

II. GIỚI THIỆU

1. Giới thiệu về bộ cân bằng.

Bộ cân bằng là bộ phận đảm bảo góc hoặc hướng của thiết bị không bị dịch chuyển trong quá trình thiết bị chuyển động. Ví dụ trong lĩnh vực điện ảnh, trong quá trình quay phim, người ghi hình có thể phải di chuyển để thực hiện các góc quay chuyển động phục vụ cho ý tưởng của đạo diễn, nếu không có dụng cụ chuyên dụng, quá trình người ghi hình di chuyển có thể khiến các cảnh quay được bị rung, giảm chất lượng của cảnh quay thu được. Vì vậy, khi quay các cảnh phim yêu cầu máy quay di chuyển, người quay phim phải sử dụng thêm thiết bị chuyên dụng (Gimbal) chống rung lắc trong quá trình chuyển động.

Bộ cân bằng thường được thiết kế gồm 3 thành phần chính, là bộ điều khiển, động cơ, và cảm biến góc. Với các ứng dụng thực tế, bộ cân bằng thường được thiết kế để có thể chống rung từ 3 hướng (ROLL, PITCH, YAW).

Hình II-1: Gimble sử dụng trong điện ảnh và các hướng quay

Ngoài lĩnh vực điện ảnh, bộ cân bằng còn được ứng dụng trong nhiều lĩnh vực như:

- Quân sự: Ổn định và chống rung cho hệ thống nhắm bắn và cảm biến quang học.
- Hàng không, hàng hải: Chống rung cho các thiết bị, hàng hóa nhạy cảm với chấn động được vận chuyển qua đường không và đường biển.
- Chế tạo robot, drone: Ổn định cho các hệ thống thị giác máy

2. Giới thiệu về hệ thống điều khiển mờ được tích hợp trong bộ cân bằng.

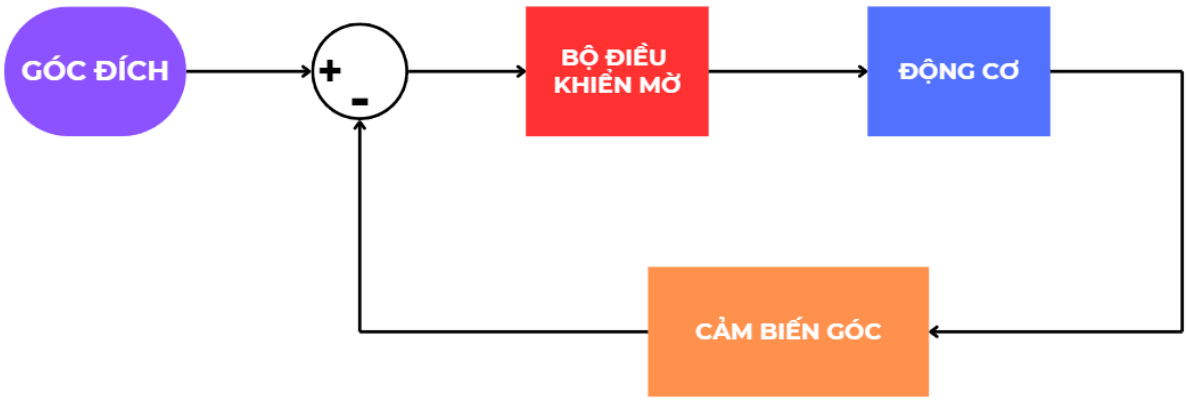
Kỹ thuật điều khiển mờ thường được sử dụng khi quan hệ lỗi vào và lối ra không được xác định rõ ràng và có thể bị ảnh hưởng bởi nhiều biến số khác khiến cho việc ứng dụng các kỹ thuật điều khiển truyền thống không đáp ứng được hiệu quả mong muốn trong hoạt động thực tế. Với bộ cân bằng, quan hệ giữa góc đích (lối vào) và hoạt động của động cơ (lối ra) khó được xác định do tải trọng của thiết bị được căn chỉnh bởi bộ cân bằng có thể bị thay đổi. Các tham số khác có thể kể tới là sai số của cảm biến góc, ảnh hương từ môi trường (gió, sức cản không khí). Bộ điều khiển mờ có thể được thiết kế mà không cần quá quan tâm đến quan hệ giữa lối vào và lối ra. Vậy nên kỹ thuật điều khiển mờ là kỹ thuật phù hợp để điều khiển bộ cân bằng.

III. Ý TƯỞNG THIẾT KẾ

1. Thiết kế hệ thống

Như đã giới thiệu, một bộ cân bằng thường được thiết kế gồm 3 thành phần chính là động cơ, cảm biến góc và bộ điều khiển. Bộ cân bằng cúng thường có khả năng chống rung theo 3 hướng, tuy nhiên, vì cơ chế cân bằng cho cả 3 hướng là tương tự nhau, nên trong nội dung bài tập lớn lần này, em sẽ tập trung thực hiện trên một hướng (YAW).

Về mô hình hoạt động, từ yêu cầu về hướng (góc) đích và thông tin về hướng hiện tại của thiết bị (do cảm biến góc thu thập), ta có thể xác định độ sai lệch giữa vị trí hiện tại và góc đích. Thông tin về độ sai lệch trên được bộ điều khiển mờ tiếp nhận và xử lý, sau quá trình xử lý, bộ điều khiển sẽ đưa ra yêu cầu điều khiển tới động cơ để động cơ thực hiện chuyển động với hướng đích. Động cơ chuyển động có thể làm thay đổi hướng của thiết bị, cảm biến góc sẽ lại thu thập thông tin về hướng hiện tại và lặp lại các quy trình ở trên, tạo thành một hệ thống điều khiển vòng kín.



Hình III-1: Mô hình hệ thống bộ cân bằng

2. Thiết kế bộ điều khiển mờ

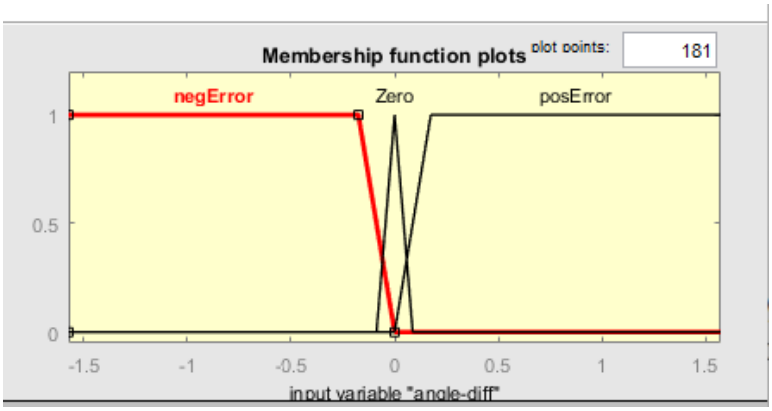
Để thuận tiện cho việc thiết kế bộ điều khiển mờ, em đã sử dụng công cụ Fuzzy Logic Designer của Matlab và khảo sát trước đặc tính vào/ra của bộ điều khiển do em thiết kế.

a. Các biến ngôn ngữ vào/ra

Lối vào của bộ điều khiển mờ được xác định là độ sai lệch giữa hướng đích và hướng hiện tại. Tùy vào giá trị sai khác này mà bộ điều khiển sẽ xác định giá trị điều khiển dành cho động cơ tại lối ra. Để xây dựng bộ điều khiển mờ cần phải xây dựng tập mờ và các biến ngôn ngữ cho lối vào và lối ra:

Lối vào: góc lệch giữa góc đích và góc hiện tại (ang_diff) theo thang radian:

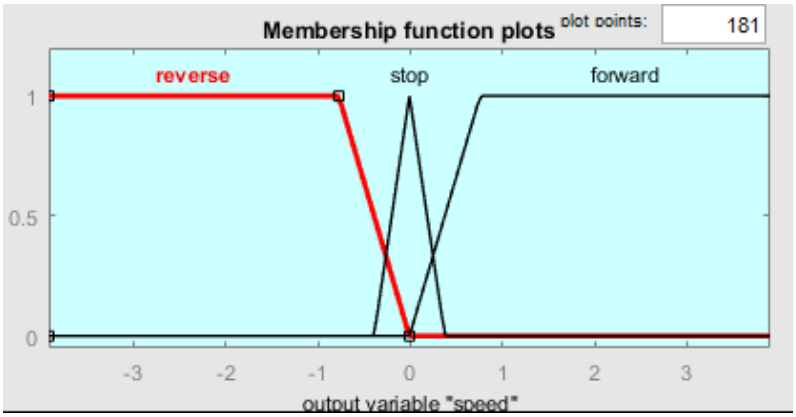
- Lệch dương (posError) : [-1.571 -1.571 -0.1745 0]
- Lệch âm (negError) : [0 0.1745 1.571 1.571]
- Không lệch (zero): [-0.08726 0 0.08726]



Hình III-2: Tập mờ của lối vào

Lối ra: tốc độ điều khiển (speed) theo thang rad/s:

- Quay ngược chiều kim đồng hồ (reverse): [-3.92 -3.92 -0.7767 0]
- Quay thuận chiều kim đồng hồ (forward): [0 0.7767 3.92 3.92]
- Dừng (stop): [-0.3833 0 0.3833]

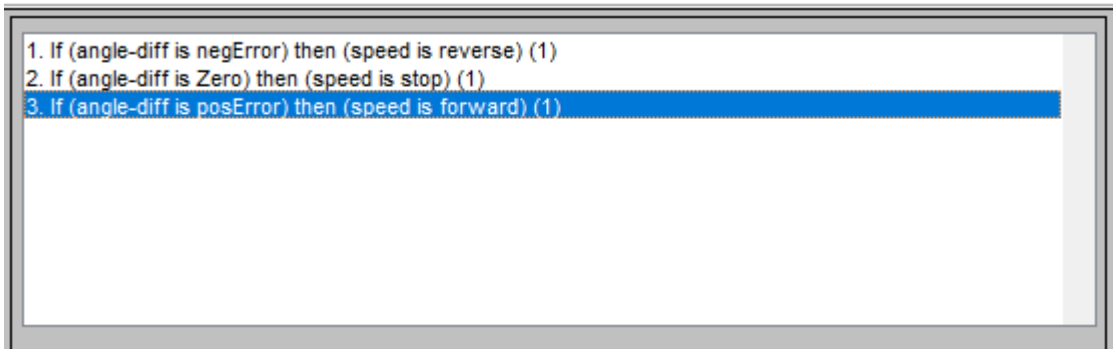


Hình III-3: Tập mờ lối ra

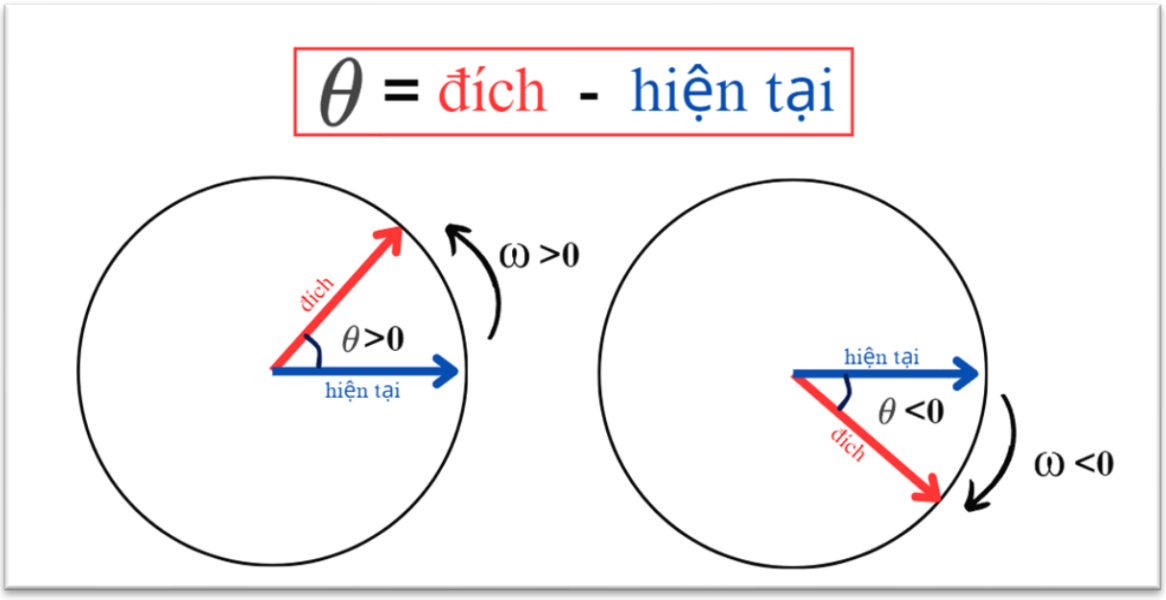
b. Luật điều khiển

Trước khi đặt luật, cần có một số quy ước về hoạt động quay của động cơ. Tại đây em quy định chiều tăng dần của giá trị góc là ngược chiều kim đồng hồ và ngược lại. Điều này có nghĩa rằng, khi góc đích lớn hơn góc hiện tại thì động cơ cần quay ngược chiều kim đồng hồ, khi góc đích bé hơn góc hiện tại thì động cơ cần quay thuận chiều kim đồng hồ. Đồng thời, em cũng quy ước chiều quay ngược (âm) của động cơ là thuận chiều kim đồng hồ, chiều quay thuận (dương) của động cơ là ngược chiều kim đồng hồ từ đó ta có các luật điều khiển như sau:

- Nếu *ĐỘ LỆCH GÓC* là **dương (posError)** thì *ĐỘNG CƠ* là **thuận (forward).**
- Nếu *ĐỘ LỆCH GÓC* là **âm (negError)** thì *ĐỘNG CƠ* là **ngược (reverse).**
- Nếu *ĐỘ LỆCH GÓC* là **không (Zero)** thì *ĐỘNG CƠ* là **dừng (stop).**



Hình III-4: Các luật điều khiển được thiết lập

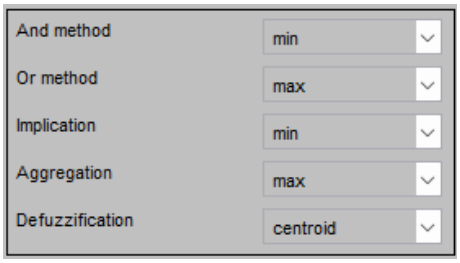


Hình III-5: Quy ước về góc và chiều quay của động cơ

c. Thiết bị hợp thành và phương pháp giải mờ

Lựa chọn thiết bị hợp thành và phương pháp giải mờ, em sử dụng các chế độ mặc định của Fuzzy Logic Designer bao gồm:

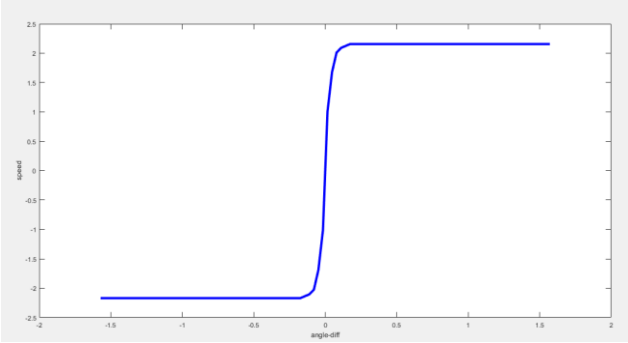
- Quy tắc hợp thành : Max – Min.
- Phương pháp giải mờ: Phương pháp điểm trọng tâm



Hình III-6: Các chế độ xử lý mờ theo mặc định của công cụ

d. Tối ưu bộ điều khiển

Từ các thiết lập trên của Fuzzy Logic Designer em có thể quan sát đặc tính truyền đạt của bộ điều khiển và đưa ra một số đánh giá sơ bộ cho bộ điều khiển.



Hình III-7: Đặc tính truyền đạt của bộ điều khiển được thiết kế

Sơ bộ có thể thấy đặc tính truyền đạt có hình dạng đúng yêu cầu thiết kế như tại luật điều khiển. Tại cửa sổ Rule Viewer của công cụ, thử nghiệm một số giá trị lối vào và quan sát giá trị lối ra:

INPUT (θ)	OUTPUT (ω)	Nhận xét
-1.047 (-π/6) ~ -30°	- 2.16	Xoay ngược
-0.01745 (-π/180) ~ -1°	- 1.06	Xoay ngược, giảm tốc
0 ~ 0°	0	Dừng
0.01745 (π/180) ~ 1°	1.06	Xoay thuận
1.047 (π/6) ~ 30°	2.16	Xoay thuận, tăng tốc

IV. QUÁ TRÌNH THỰC HIỆN

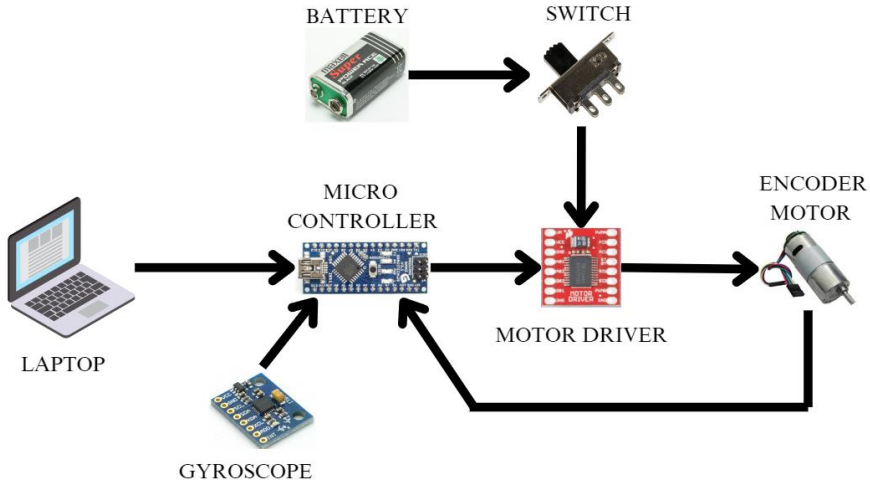
1. Chế tạo thiết bị

Thiết bị em xây dựng sử dụng các linh kiện:

- Cảm biến gia tốc: MPU6050
- Vi điều khiển : Arduino Nano
- Động cơ Encoder và driver điều khiển động cơ TB6612
- Nguồn nuôi động cơ: Pin 9V

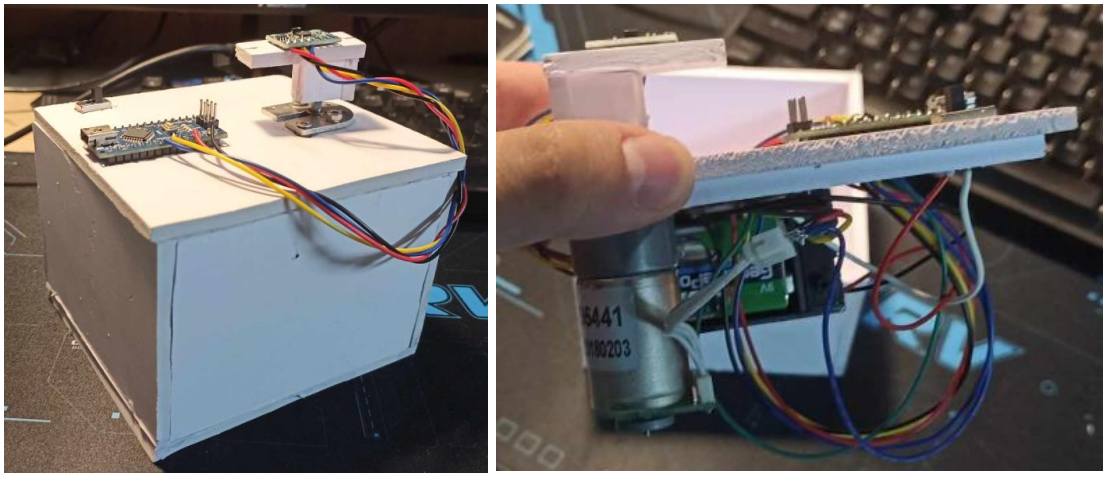
Kết nối giữ các linh kiện, thiết bị được thục hiện như sau:

- Cảm biến gia tốc được kết nối với vi điều khiển qua giao thức I2C.
- Vi điều khiển kết nối với driver điều khiển động cơ để có thể đưa ra các yêu cầu chuyển động tới động cơ.
- Driver được kết nối với pin 9V làm nguồn nuôi động cơ. Ngoài ra để có thể ngắt động cơ trong một số tình huống khẩn cấp (ví dụ như bị cuốn dây), em đã bổ sung thêm một công tắc gạt kết nối giữa pin 9v với driver động cơ.
- Driver được kết nối với động cơ để thực hiện yêu cầu điều khiển mà vi điều khiển đưa ra.
- Động cơ được kết nối với vi điều khiển để truyền thông tin về trạng thái chuyển động (ENCODER)
- Vi điều khiển được kết nối với Laptop để có thể theo dõi trạng thái hoạt động của hệ thống và đưa ra các yêu cầu điều khiển từ bàn phím.



Hình IV-1: Sơ đồ kết nối hệ thống

Sau khi hàn và kết nối các linh kiện thông qua hệ thống dây dẫn, em đã thiết kế một hộp đựng dùng để bảo vệ các kết nối cũng như hỗ trợ khảo sát hoạt động của hệ thống



Hình IV-2: Bộ cân bằng đơn trục được triển khai trong thử nghiệm

2. Chương trình vận hành thiết bị

Trước khi triển khai bộ điều khiển mờ vào chương trinh của vi điều khiển, vi điều khiển cần được lập trình để thao tác cơ bản với các thành phần linh kiện khác

a. Nhận và xử lý thông tin từ cảm biến gia tốc

Để nhận thông tin về góc từ cảm biến gia tốc, em đã sử dụng thư viện hỗ trợ dành cho vi điều khiển Arduino.

```
1   #include <MPU6050_tockn.h>

//--------------------GYRO ZONE-------------------------
MPU6050 mpu6050(Wire);
float prev_ang = 0;
float target_ang = 0;

// calculate differential in angle
float current_ang_diff(){
  mpu6050.update();

  float current = mpu6050.getAngleZ();
  float diff = target_ang - current;

  // Serial.print("Recent angle: ");
  // Serial.print(current);
  // Serial.print("\tAngle_diff: ");
  // Serial.print(diff);

  return diff;
}
//------------------END GYRO ZONE---------------------
```

Hình IV-3: Thư viện và phần chương trình thực hiện lấy thông tin về góc và tính toán sai lệch giữa góc hiện tại và góc đích

b. Điều khiển động cơ

Thông qua driver TB6612 ta có thể thực hiện điều khiển động cơ

```
#define PULSE_PER_CYCLE 200
#define MAX_SPEED 255.0
#define MAX_PWM 255

#define ANG_TOLRATE 0

// Khai báo chân điều khiển động cơ
const int AIN1 = 3; // Chân AIN1
const int AIN2 = 4; // Chân AIN2
const int PWMA = 5; // Chân PWMA (PWM)
```

Hình IV-4: Các tham số khởi tạo cho động cơ

```
void motor_init(){
  // Cấu hình chân TB6612
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(PWMA, OUTPUT);

  // Cấu hình chân Hall sensor
  pinMode(HallSensor, INPUT_PULLUP);

  // Gắn ngắt ngoài để đọc xung từ encoder
  attachInterrupt(digitalPinToInterrupt(HallSensor), countPulse, FALLING);
}
```

Hình IV-5: Chương trình con khởi tạo động cơ

```
void set_motor_speed(float speed){
  unsigned long time_diff = millis() - t_motor_run;
  t_motor_run = millis();

  int pwm = (int) ((speed / MAX_SPEED) * MAX_PWM);
  driveMotor(pwm);
}
```

Hình IV-6: Chương trình con chuyển đổi tốc độ thành tham số cho bộ điều khiển PWM

```
// Hàm điều khiển động cơ
void driveMotor(int speed) {
  // Serial.print("\tControled speed: ");
  // Serial.print(speed);
  if (speed > 0) {
    // Quay thuận
    digitalWrite(AIN1, HIGH);
    digitalWrite(AIN2, LOW);

  // Điều chỉnh tốc độ bằng PWM
  analogWrite(PWMA, constrain(speed, 0, 255));
  } else if (speed < 0) {
    // Quay ngược
    digitalWrite(AIN1, LOW);
    digitalWrite(AIN2, HIGH);
    speed = -speed; // Đảo chiều tốc độ
    // Serial.print("\tRev controled speed: ");
    // Serial.print(speed);

    // Điều chỉnh tốc độ bằng PWM
    analogWrite(PWMA, constrain(speed, 0, 255));
  } else {
    // Dừng động cơ
    digitalWrite(AIN1, LOW);
    digitalWrite(AIN2, LOW);
    // Điều chỉnh tốc độ bằng PWM
    analogWrite(PWMA, constrain(speed, 0, 255));
  }

}
```

Hình IV-7: Chương trình thực hiện điều khiển động cơ quay đúng chiều với tốc độ theo tham số PWM

c. Giao tiếp với Laptop

```
//----------------SERIAL PROCESS ZONE-----------------

#define LOG_CYCLE 500
unsigned long log_marked;
float pi=3.14159265358979323846;

float toRad(float num){
  return (num/360)*(2*pi);
}

void status_log(float ang_diff, float speed){
  if (millis() - log_marked > LOG_CYCLE)
  {
    //print status
    Serial.print("Target angle: ");
    Serial.print(toRad(target_ang));
    Serial.print("rad ;\tCurrent angle: ");
    Serial.print(toRad(mpu6050.getAngleZ()));
    Serial.print("rad ;\tAngle difference: ");
    Serial.print(toRad(ang_diff));
    Serial.print("rad ;\tMotor speed: ");
    Serial.print(-1.0*toRad(speed));
    Serial.println(" rad/s ");

    //update time mark
    log_marked=millis();
  }
}

void command_listen(){
  if(Serial.available()){
    // String str = Serial.readStringUntil('\n'); // To read string

    // Read Serial into char type buffer
    char buffer[5] ;
    uint8_t size = Serial.readBytesUntil('\n',buffer,5);
    buffer[size]='\0';

    // COMMENT/UNCOMMENT for define input as angle or speed

    // keyboard_motor_speed = atof(buffer);
    // Serial.print("\tSpeed get from buffer: ");
    // Serial.print(keyboard_motor_speed);

    target_ang = atof(buffer);
    // Serial.print("\tAngle get from buffer: ");
    // Serial.print(target_ang);
  }
}
//----------------END SERIAL PROCESS ZONE----------------
```

Hình IV-8: Phần chương trình thực hiện giao tiếp với Laptop

d. Triển khai bộ điều khiển mờ.

Để triển khai bộ điều khiển mờ em đã sử dụng thư viện Fuzzy.h để hỗ trợ cho việc triển khai bộ điều khiển mờ trên Arduino.

```
#include <Fuzzy.h>

//-----------------FUZZY LOGIC ZONE-------------------
Fuzzy *fuzzy = new Fuzzy();

void setup_fuzzy_logic() {
  // Input: Angle Difference
  FuzzySet *negError = new FuzzySet(-90, -90, -10, 0);
  FuzzySet *zero = new FuzzySet(-5, 0, 0, 5);
  FuzzySet *posError = new FuzzySet(0, 10, 90, 90);
  FuzzyInput *angleDiff = new FuzzyInput(1);
  angleDiff->addFuzzySet(negError);
  angleDiff->addFuzzySet(zero);
  angleDiff->addFuzzySet(posError);
  fuzzy->addFuzzyInput(angleDiff);

  // Output: Motor Speed
  FuzzySet *reverse = new FuzzySet(-225, -225, -50, 0);
  FuzzySet *stop = new FuzzySet(-25, 0, 0, 25);
  FuzzySet *forward = new FuzzySet(0, 50, 225, 225);
  FuzzyOutput *motorSpeed = new FuzzyOutput(1);
  motorSpeed->addFuzzySet(reverse);
  motorSpeed->addFuzzySet(stop);
  motorSpeed->addFuzzySet(forward);
  fuzzy->addFuzzyOutput(motorSpeed);

  // Rules
  FuzzyRuleAntecedent *ifNegError = new FuzzyRuleAntecedent();
  ifNegError->joinSingle(negError);
  FuzzyRuleConsequent *thenReverse = new FuzzyRuleConsequent();
  thenReverse->addOutput(reverse);
  fuzzy->addFuzzyRule(new FuzzyRule(1, ifNegError, thenReverse));

  FuzzyRuleAntecedent *ifZero = new FuzzyRuleAntecedent();
  ifZero->joinSingle(zero);
  FuzzyRuleConsequent *thenStop = new FuzzyRuleConsequent();
  thenStop->addOutput(stop);
  fuzzy->addFuzzyRule(new FuzzyRule(2, ifZero, thenStop));

  FuzzyRuleAntecedent *ifPosError = new FuzzyRuleAntecedent();
  ifPosError->joinSingle(posError);
  FuzzyRuleConsequent *thenForward = new FuzzyRuleConsequent();
  thenForward->addOutput(forward);
  fuzzy->addFuzzyRule(new FuzzyRule(3, ifPosError, thenForward));
}

float fuzzy_speed_control(float ang_diff) {
  fuzzy->setInput(1, ang_diff);
  fuzzy->fuzzify();
  return fuzzy->defuzzify(1);
}
//----------------END FUZZY LOGIC ZONE----------------
```

Hình IV-9: Phần chương trinh thiết lập và sử dụng bộ điều khiển mờ

e. Setup() và Loop()

Tại hàm Setup(), em gọi lại các hàm thực hiện thiết lập như thiết lập động cơ, bộ điều khiển mờ, cảm biến gia tốc và giao tiếp với máy tính thông qua Serial Monitor.

```
void setup() {
  // Giao tiếp Serial để hiển thị thông tin
  Serial.begin(115200);

  motor_init();

  Wire.begin();
  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);
  mpu6050.update();
  prev_ang = mpu6050.getAngleZ();

  setup_fuzzy_logic();
  log_marked=millis();
  // t_spd_measure=millis();
}
```

Hình IV-10: Hàm Setup().

Tại hàm Loop(), em đã gọi các hàm thực hiện hoạt độ của bộ điều khiển mờ bao gồm: xác định độ sai lệch giữa góc hiện tại với góc đích; thực thi bộ điều khiển mờ để xác định tốc độ điều khiển động cơ; yêu cầu động cơ chuyeenr động với tốc độ được xác định. Các chu trình lặp lại này thực hiện như với mô hình thiết kế tại *Hình III-1*. Ngoài ra , trong hàm Loop() em còn gọi hàm thực hiện báo cáo các trạng thái hoạt động của hệ thống.

```
void loop() {
  // Điều khiển động cơ (quay theo chiều kim đồng hồ)
  float ang_diff= current_ang_diff();

  //simlple test
  // float speed = choose_speed(ang_diff);

  //fuzzy implement
  float speed = fuzzy_speed_control(ang_diff);

  set_motor_speed(speed);

  command_listen();

  status_log(ang_diff,speed);

  // motor_proc();

  // Serial.println();
  // delay(50);
}
```

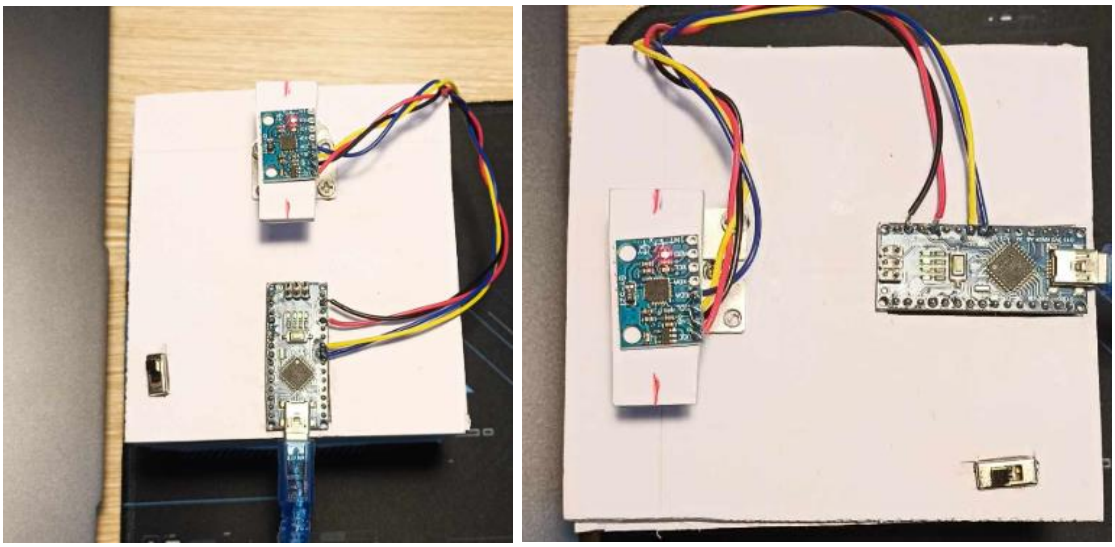
Hình IV-11: Hàm Loop()

V. THỬ NGHIỆM VÀ NHẬN XÉT

Sau khi đã hoàn thành hương trình cho vi điều khiển, em đã tiến hành nạp chương trình và kiểm tra các hoạt động của thiết bị. Khi khởi động thiết bị cần phải chờ thiết bị khởi động trong một khoảng thời gian để cảm biến góc tính toán các sai số trong gia tốc góc, trong quá trình này, thiết bị không được phép di động, nếu không quá trình tính toán sẽ xảy ra sai sót và hệ thống sẽ không hoạt động như được thiết kế.

1. Kiểm tra và nhận xét khả năng cân bằng của thiết bị

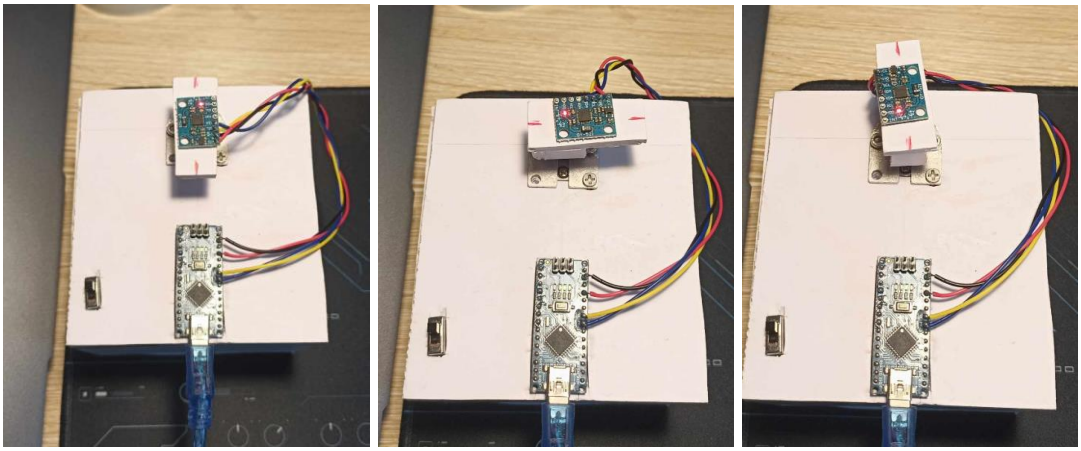
Em đã tiến hành xoay hộp thiết bị và nhận thấy động cơ đã hoạt động và điều khiển đầu chỉ hướng quay về hướng cũ. Khi xoay với tốc độ thấp, đầu chỉ hướng được điều khiển để giữ nguyên hướng chỉ định. Khi quay nhanh hộp, đầy chỉ hướng có phần không kịp chuyển động để giữ hướng và cần một khoảng thời gian ngắn để quay về vị trí ban đầu. Nguyên nhân của vấn đề này có thể do giới hạn của tốc độ động cơ sử dụng trong thử nghiệm và cũng có thể do bộ điều khiển chưa đáp ứng đủ nhanh với thay đổi trong thời gian ngắn.



Hình V-1: Thử nghiệm khả năng cân bằng của thiết bị

2. Kiểm tra và nhận xét khả năng thực thi yêu cầu điều khiển quay góc xác định

Để kiểm tra khả năng điều hướng của thiết bị tới hướng chỉ định từ bàn phím Laptop, em đã thực hiện đưa ra các yêu cầu về góc đích từ bàn phím máy tính thông qua của sổ Serial Monitor. Sau khi gửi yêu cầu động cơ lập tức được điều khiển để quay đầu chỉ hướng tới góc được xác định ở yêu cầu.



Hình V-2: Thử nghiệm điều khiển đầu chỉ hướng quay tới góc chỉ định (từ góc 0° đến góc 90° và góc 180°)

VI. ĐÁNH GIÁ ĐỀ TÀI

Trong nội dung bài tập lớn lần này em đã hoàn thành xây dựng một bộ cân bằng đơn trục từ thiết kế phần cứng, viết chương trình cho vi điều khiển và ứng dụng kỹ thuật điều khiển mờ cho hệ thống. Sau quá trình thực hiện em đã nắm được rõ và hiểu hơn về quy trình, cách thiết kế một hệ thống sử dụng kỹ thuật điều khiển mờ cũng như cách chế tạo một bộ cân bằng đơn giản.