

# **Sunflower Solar - Tacker**



# **Sunflower Solar - Tacker**



# T

# E

# A

# M



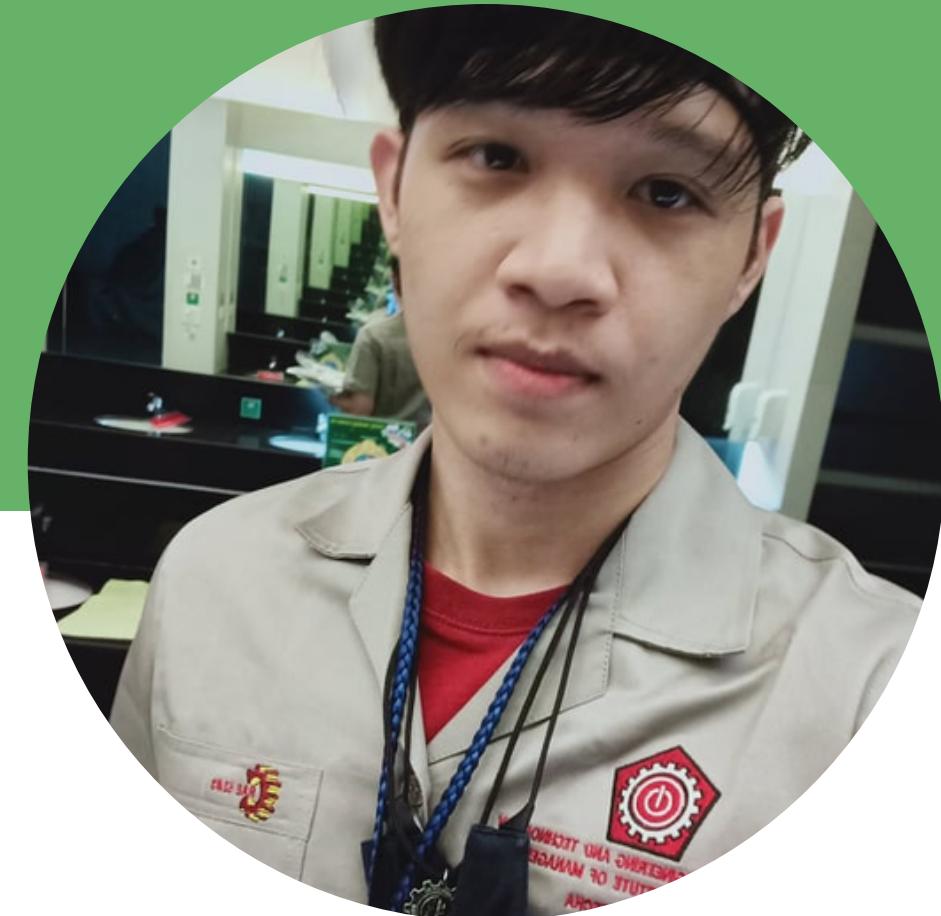
## Team Work

นายศิวฤกษ์ กล่าวโภจนาณ  
6252500020



## Team Work

นายภูริต เหล่ารุ่งเกียรติ  
6252500127



## Team Work

นายพงษ์ปรีชา รัตนสร้อย  
6252500283

# T E A M



Team Work

นายศิวฤกษ์ กล่าวโภจนาณ  
6252500020



Team Work

นายภูริต เหล่ารุ่งเกียรติ  
6252500127



Team Work

นายพงษ์ปรีชา รัตนสร้อย  
6252500283

# T

# E

# A

# M

## Team Work

นายชินดันัย ไสรบุตร  
6252500526



## Team Work

นายจิรสิน เทศรุ่งเรือง  
6252500569



T  
E  
A  
M



Team Work

นายชินดันัย ไสรบุตร  
6252500526



Team Work

นายจิรสิน เทศรุ่งเรือง  
6252500569

# Design

การออกแบบหุ่นยนต์ที่เลียนแบบสิ่งมีชีวิตหรือทำงานร่วมกับสิ่งมีชีวิต



# Simulator

วิดีโอจำลองการทำงานของ Solar - Tracker

Sole\_Tracker | Arduino 1.8.10

File Edit Sketch Tools Help

Serial Monitor COM1

```

Sole_Tracker.ino:1:1: fatal error: ESP32Servo.h: No such file or directory
 #include <ESP32Servo.h>
          ^~~~~~
compilation terminated.
```

15:09:40.637 -> ets Jun 8 2016 00:22:57
15:09:40.637 ->
15:09:40.637 -> rst:0x1 (POWERON\_RESET),boot:0x13 (SPI\_FAST\_FLASH\_BOOT)
15:09:40.637 -> configsip: 0, SPIWP:0xee
15:09:40.637 -> clk\_drv:0x00,q\_drv:0x00,d\_drv:0x00,cs0\_drv:0x00,hd\_drv:0x00,wp\_drv:0x
15:09:40.637 -> mode:DIO, clock div:1
15:09:40.637 -> load:0x3fff0030,len:1184
15:09:40.637 -> load:0x40078000,len:12804
15:09:40.637 -> ho 0 tail 12 room 4
15:09:40.637 -> load:0x40080400,len:3032
15:09:40.637 -> entry 0x400805e4
15:09:40.776 -> Servo\_X:91,Servo\_Y:90,UT:-14283.79,UT\_y:inf
15:09:40.869 -> Servo\_X:92,Servo\_Y:91,UT:-2117.76,UT\_y:14653.60
15:09:40.963 -> Servo\_X:94,Servo\_Y:92,UT:-5491.58,UT\_y:17473.40
15:09:41.010 -> Servo\_X:96,Servo\_Y:93,UT:-6583.19,UT\_y:17303.80
15:09:41.056 -> Servo\_X:97,Servo\_Y:94,UT:-5375.99,UT\_y:14923.40
15:09:41.149 -> Servo\_X:98,Servo\_Y:95,UT:-10097.19,UT\_y:16702.40
15:09:41.241 -> Servo\_X:99,Servo\_Y:96,UT:-3168.19,UT\_y:16454.00

Autoscroll  Show timestamp

No line ending 115200 baud Clear output

Camera

15:09 28°C 05/22/2022

# Theory

อธิบายหลักการทำงานที่เลียนแบบการทำงานที่มีอยู่ในทางชีวภาพ

## ชีวภาพของพืช

พืชส่วนใหญ่นั้น ก็มีแนวโน้มที่จะเคลื่อนที่ไปในทิศทางของดวงอาทิตย์ซึ่งชาวกรีกสมัยโบราณเรียกปรากฏการนี้ว่า "Heliotropium" เพราะฉะนั้นเราจึงเรียกพฤติกรรมนี้ว่า Heliotropism เช่นเดียวกับมนุษย์ (biological clock) พืชก็มีนาฬิกาชีวิตภายใน "Circadian Rhythm" ซึ่งการเต้นของหัวใจมนุษย์ทำให้เกิดการเปลี่ยนแปลง ทางสรีรวิทยาและทางเคมีหลายอย่างในขณะเดียวกันพืชก็มีการตอบสนองใน 24 ช.ม.

# ชีวภาพของพืช

พืชส่วนใหญ่นั้น ก็มีแนวโน้มที่จะเคลื่อนที่ไปในทิศทางของดวงอาทิตย์ซึ่งชาวกรีกสมัยโบราณเรียกปรากฏการนี้ว่า "Heliotropium" เพราะฉะนั้นเราจึงเรียกพุตติกรรมนี้ว่า Heliotropium เช่นเดียวกับมนุษย์ (biological clock) พืชก็มีนาฬิกาชีวิตภายใน "Circadian Rhythm" ซึ่งการเต้นของหัวใจมนุษย์ทำให้เกิดการเปลี่ยนแปลง ทางสรีรวิทยาและทางเคมีหลายอย่างในขณะเดียวกันพืชก็มีการตอบสนองใน 24 ช.ม.

# ชีวภาพของดอกงานตะวัน

ก่อนรุ่งแสง ดอกงานตะวันจะหันหน้าไปทางทิศตะวันออกในทางที่ดวงอาทิตย์ขึ้นและเมื่อดวงอาทิตย์เคลื่อนตัวไปยังทิศตะวันตกดอกงานตะวันก็จะหันเช่นกันและเมื่อตัววันตกลบดอกงานตะวันก็จะกลับไปยังทางทิศตะวันออกเพื่อเริ่มนิรภัยครั้งในวันต่อๆไป นักวิจัยเชื่อว่า ดอกงานตะวัน แสดงให้ถึงปรากฏการณ์ *Heliotropism* เนื่องจาก ลำต้นของมันยึดอุปกรณ์อัตราที่ต่างกัน ในแต่ละช่วงเวลาซึ่งในช่วงเวลาของดวงอาทิตย์ขึ้นในทิศตะวันออกไปทิศตะวันตก ด้านตะวันออกลำตัวของมันจะเติบโตเร็วกว่าข้างทิศตะวันตก เนื่องจากการเจริญเติบโตทั้ง 2 ข้างไม่เท่ากัน ดอกงานตะวันเลยโค้งงอไปในทิศของดวงอาทิตย์นั้นๆ เพราะถ้าหากลักษณะของดอกงานตะวันจะเจริญเติบโตไม่เท่ากันเนื่องจาก *AUXINS* (ฮอร์โมนพืชที่กระตุ้นการเจริญเติบโต) ไม่เท่ากันทั้ง 2 ข้าง และเมื่อดอกงานตะวันต้นนั้นแก่ลง การเจริญเติบโตก็จะลดลงไปเรื่อยๆ และจะไม่หันไปอีก ดอกงานตะวันถ้าหันไปทางทิศตะวันออกจะดึงดูดผึ้งได้ดีให้ผสมเกสร

# ชีวภาพของดอกงานตะวัน

ก่อนรุ่งแสง ดอกงานตะวันจะหันหน้าไปทางทิศตะวันออกในทางที่ดวงอาทิตย์ขึ้นและเมื่อดวงอาทิตย์เคลื่อนตัวไปยังทิศตะวันตกดอกงานตะวันก็จะหันเช่นกันและเมื่อตัววันตกดินดอกงานตะวันก็จะกลับไปยังทางทิศตะวันออกเพื่อเริ่มนิรภัยครั้งใหม่ต่อๆไป นักวิจัยเชื่อว่า ดอกงานตะวัน แสดงให้ถึงปรากฏการณ์ *Heliotropism* เนื่องจาก ลำต้นของมันยึดอุปโภคในอัตราที่ต่างกัน ในแต่ละช่วงเวลาซึ่งในช่วงเวลาของดวงอาทิตย์ขึ้นในทิศตะวันออกไปทิศตะวันตก ด้านตะวันออกลำตัวของมันจะเติบโตเร็วกว่าข้างทิศตะวันตก เนื่องจากการเจริญเติบโตทั้ง 2 ข้างไม่เท่ากัน ดอกงานตะวันเลยโค้งงอไปในทิศของดวงอาทิตย์นั้นๆ เพราะถ้าหากลักษณะของดอกงานตะวันจะเจริญเติบโตไม่เท่ากันเนื่องจาก *AUXINS* (ฮอร์โมนพืชที่กระตุ้นการเจริญเติบโต) ไม่เท่ากันทั้ง 2 ข้าง และเมื่อดอกงานตะวันตัวนั้นแก่ลง การเจริญเติบโตก็จะลดลงไปเรื่อยๆ และจะไม่หันไปอีก ดอกงานตะวันถ้าหันไปทางทิศตะวันออกจะดึงดูดผึ้งได้ดีให้ผสมเกสร

# ระบบหรืออัลกอริทึมที่เกี่ยวข้อง

เราจะเปรียบเทียบค่าความสว่าง ( lux ) กับอุณหภูมิที่ได้จากการตัววันได้รับมาแล้วน้อยถึง ยิ่งค่าความสว่างนั้นสูง อุณหภูมิก็จะยิ่งสูงตามไปด้วย ซึ่งสามารถเขียนเป็นสมการความสัมพันธ์ได้คือ

$$\text{ความสว่าง} (\text{ Lux: } \emptyset) \propto \text{อุณหภูมิ} (\text{ Temperature : T })$$

ซึ่งในการศึกษาข้อมูลดังกล่าวทำให้เรานำ sunflower optimization (SFO) algorithm มาใช้

# sunflower optimization (SFO) algorithm

1. พฤติกรรมทางชีวภาพและธรรมชาติ จะเคลื่อนที่เข้าหาแสงที่มีความเข้มแสงมากที่สุดโดยเราสามารถคำนวณได้โดย ใช้สูตรทางคณิตศาสตร์ ดังนี้

$$I = \frac{P}{(4\pi d^2)}$$

เป็นสูตรในการหาความเข้มของแสงโดยเฉพาะ โดยที่  
| คือความเข้มของการแผ่รังสีด้วงอาทิตย์  
P คือพลังของดวงอาทิตย์  
d คือระยะห่างระหว่างดวงอาทิตย์กับดอกรากันตัววันแต่ละดวง

# sunflower optimization (SFO) algorithm

2. การปรับทิศทางดอ�팡ตัววัน ดอ�팡ตัววันแต่ละดอจะปรับทิศทางของมันไปที่ ดวงอาทิตย์ ดังแสดงในสมการต่อไปนี้

$$S(i) = \frac{X^* - X_i}{|X^* - X_i|} ; i = 1, 2, \dots, n$$

โดยที่

$X^*$  คือค่าตอบที่ดีที่สุดโดยรวม

$X_i$  คือค่าตอบปัจจุบัน

ทคือขนาดประชากร

# sunflower optimization (SFO) algorithm

3. องศาของดอกรากันตะวันที่จะหันเข้าหาดวงอาทิตย์ มีสมการดังนี้

$$d_i = Y(\mu_m) \times \pi(||X_i + X_{i-1}||)$$

โดยจะมี

$Y(\mu_m)$  = ความเอื้อยของดอกรากันตะวัน

$\pi(||X_i + X_{i-1}||)$  = ความน่าจะเป็นของการผสมเกสรก่อนการตะวันแต่ละชั้นของ  
ดอกรากันตะวันกับดอกรากันตะวันที่อยู่ใกล้กันที่สุดเพื่อสร้างดอกรากันตะวันใหม่ในตำแหน่งใหม่โดย  
พิจารณาจากระยะห่างระหว่างดอกรากันตะวันทั้งสองตัว

# sunflower optimization (SFO) algorithm

4. การกำหนดลิมิตของดอกรากวนแต่ละขั้นถูกจำกัดไม่ให้เกินค่าต่อไปนี้  
ด้วยสูตร

$$d_{(max)} = \frac{\|X_{(max)} - X_{(min)}\|}{2n}$$

โดยที่

$X_{(max)}$  และ  $X_{(min)}$  เป็นลิมิตที่มากที่สุดและลิมิตที่น้อยที่สุด  
ทั้งคือ จำนวนประชากรของดอกรากวน

# sunflower optimization (SFO) algorithm

## 5. อัพเดตตำแหน่งดอกร้านตะวัน

ดอกร้านตะวันแต่ละชีบีด มีการเปลี่ยนตำแหน่งเพื่อผลิตดอกร้านตะวันรุ่นใหม่ตามทิศทาง ( ขี้นตก Sun ) และขึ้นตอนของดอกร้านตะวันที่มุ่งสู่ดวงอาทิตย์ ( การหันหน้าเข้าหาดวงอาทิตย์ ) ดังนี้

โดยใช้สูตร

$$X(i+1) = X(i) + (d(i) \times S(i))$$

โดยที่

$X(i+1)$  คือตำแหน่งของดอกร้านตะวันที่สร้างขึ้นใหม่ (รายบุคคล)

# การนำมาใช้และการเปรียบเทียบกับ Solar-Tacker

- การนำมาใช้

1.	ตั้งค่าเริ่มต้นสำหรับอัตราการตาย $m$ , ขนาดประชากร $n$ , อัตราการผสมเกสร $P$ และจำนวนการวนซ้ำสูงสุด $mazitr$ เช่น $m = 1, n = 50, P = 20, mazitr = 80$
2.	ตั้งค่า $t = 0$
3.	สร้างประชากร $X(i) \in [L, U]$ แบบสุ่ม $i = 1, 2, 3, \dots, n$ จะคล้ายกับในส่วนของ Genetic หรือเราสามารถปรับได้เป็น $population = n$

4.

การคำนวณ Fitness Function รายตัวของดอก  
กานตัวนั้น ในประชากรที่เรากำหนดไว้ก็จะมีค่า  
สมรรถภาพร่างกายของบุคคล (กานตัวนั้น) ในประชากร 1,.. ,  
 $n = S(X)$

5.

เลือกตัวแทนของเซลล์ที่ใกล้กับแสงที่สุดจากโดยรวมของ  
ประชากร X

6.

ปรับบุคคลก็จะมีค่า (ดอกกานตัวนั้น) ปรับตัวแทนไปทางดัง  
อาคิตรี ดังแสดงในสมการที่ 5

7

คำนวณกิจกรรมการเคลื่อนที่ ของค่า Fitness เป็นรูปแบบเวกเตอร์  
ให้กับทุกประชากร

ลบดอกไม้ที่มีค่าແຍ່ງສุดออกไป ค่า  $m\%$

คำนวณขั้นตอนของบุคคลเข้าหาดวงอาทิตย์ดังในข้อที่ 6

ให้ปุ่ยดอกกานตะวันที่ดีที่สุดรอบดวงอาทิตย์

ตรวจสอบค่าที่มากที่สุด

อัพเดตข้อมูล

ประเมินบุคคลใหม่

ยอมรับบุคคลใหม่หากค่าความพิเศษของดอกใหม่ดีกว่าค่าปัจจุบัน

ตั้ง  $t = t+1$

8.

1. ถ้า  $t > mazitr$  ให้ออกจากลูป

9.

1. แสดงผลลัพธ์ที่ได้ที่สุดอ่อนมา

# การนำมาใช้และการเปรียบเทียบกับ Solar-Tacker

- การเปรียบเทียบกับ Solar - Tacker

โดยในการนำ sunflower optimization (SFO) algorithm มาใช้นั้นเราได้นำสมการต่างๆมาดัดแปลงเป็นฟังก์ชัน ดังนี้

- การหาค่าความสว่าง ( lux ) โดยใช้ฟังก์ชันแทนสมการ คือ

```
//ค่าที่ใช้ในการแปลง Lux      for (int i = 0; i < N ; i++) { //sizeof(sensor_LDR)
float LDR = 0;
float Volt = 0;
float Lux = 0;
    if (i != 2)
    {
        LDR = analogRead(sensor_PIN[i]);
        Volt = (3.3 / 4096) * (LDR);
        Lux = 42.0 * pow(Volt, 3.15);
        sensor_LDR[i] = int(Lux);
    }
    else
    {
        LDR = analogRead(sensor_PIN[i]);
        Volt = (3.3 / 4096) * (LDR+500);
        Lux = 42.0 * pow(Volt, 3.15);
        sensor_LDR[i] = int(Lux);
    }
}
```

# การนำมาใช้และการเปรียบเทียบกับ Solar-Tacker

- การเปรียบเทียบกับ Solar - Tacker

## 2. การปรับทิศทางดอกราบริวัต โดยใช้ฟังก์ชันแทนสมการ คือ

```
int P = 1;
int N = 4;
int y_Update = 0;
int x_Update = 0;
int Count(int si_value[], int length) {
    int count = 0;
    for (int i = 0; i < length ; i++) {
        if (si_value[i] == 0 ) {
            count += 1;
        }
    }
    return count;
}

for (int i = 0; i < N ; i++) {
    if (MaxVal == sensor_LDR[i]) {
        si_value[i] = int(0);
    } else {
        int si = ( MaxVal - sensor_LDR[i] / abs(MaxVal - sensor_LDR[i]) );
        si_value[i] = int(si);
    }
    int di = ( rad * (P * ((sensor_LDR[i] + sensor_LDR[i - 1]) * abs(sensor_LDR[i] + sensor_LDR[i - 1])))) % 180;
    di_value[i] = int(di);
    int X = sensor_LDR[i] + di_value[i] * si_value[i];
    X_value[i] = int(X) % 180;
}
int num = Count(si_value, N);
if ((num > 2 && si_value[1] == 0) || si_value[1] == 0) {
    x_Update = ((abs(X_value[1] - X_value[2]) / 4) * (-1));
}
else {
    x_Update = ((abs(X_value[1] - X_value[2]) / 4));
}

if ((num > 2 && si_value[3] == 0) || si_value[3] == 0) {
    y_Update = ((abs(X_value[0] - X_value[3]) / 4) * (-1));
}
else {
    y_Update = ((abs(X_value[0] - X_value[3]) / 4));
}
```

# การนำมาใช้และการเปรียบเทียบกับ Solar-Tacker

- การเปรียบเทียบกับ Solar - Tacker

- องศาขององค์การตัววันที่จะหันเข้าหาดวงอาทิตย์และการกำหนดลิมิตขององค์การตัววัน โดยใช้พังค์ชั่นแทนสมการ คือ

```
int MaxVal = 0; int MaxValue(int sensor_LDR[] , int length) {  
    int MinVal = 0;     int Max = 0;  
    int rad = 0;         for (int i = 0; i < length ; i++) {  
        if (sensor_LDR[i] > Max ) {  
            Max = int(sensor_LDR[i]);  
        }  
    }  
    return Max;  
}  
  
int MinValue(int sensor_LDR[] , int length) {  
    int Min = 10000;  
    for (int i = 0; i < length ; i++) {  
        if (sensor_LDR[i] < Min ) {  
            Min = int(sensor_LDR[i]);  
        }  
    }  
    return Min;  
}  
  
int radian(int MaxVal , int MinVal , int N) {  
    int d_max = (abs((MaxVal * 1000 / 57296) + (MinVal * 1000 / 57296)) / (2 * N) );  
    int rads = (d_max % 180) * 1000 / 57296;  
    return rads;  
}
```

# การนำมาใช้และการเปรียบเทียบกับ Solar-Tacker

- การเปรียบเทียบกับ Solar - Tacker

## 4. อัพเดตตำแหน่งดอกร้านตะวัน โดยใช้ฟังก์ชันแทนสมการ คือ

```
int y_Update = 0;
int x_Update = 0;
int Count(int si_value[], int length) {
    int count = 0;
    for (int i = 0; i < length ; i++) {
        if (si_value[i] == 0 ) {
            count += 1;
        }
    }
    return count;
}
int num = Count(si_value, N);
if ((num > 2 && si_value[1] == 0) || si_value[1] == 0) {
    x_Update = ((abs(X_value[1] - X_value[2]) / 4) * (-1));
} else {
    x_Update = ((abs(X_value[1] - X_value[2]) / 4));
}

if ((num > 2 && si_value[3] == 0) || si_value[3] == 0) {
    y_Update = ((abs(X_value[0] - X_value[3]) / 4) * (-1));
} else {
    y_Update = ((abs(X_value[0] - X_value[3]) / 4));
}
```

# การนำมาใช้และการเปรียบเทียบกับ Solar-Tacker

- การเปรียบเทียบกับ Solar - Tacker

## 4. อัพเดกตำแหน่งดอกร้านตะวัน โดยใช้ฟังก์ชันแทนสมการ คือ ( ต่อ )

```
pos_x = pos_x+map(x_Update,-43,43,-2,2);
pos_y = pos_y+map(y_Update,-43,43,-2,2);

if(pos_x>180)
{
    pos_x = 180;
}
else if(pos_x<0)
{
    pos_x = 0;
}

if(pos_y>180)
{
    pos_y = 180;
}
else if(pos_y<0)
{
    pos_y = 0;
}
```

# Test & Compare

การทดสอบและเปรียบเทียบระหว่าง PID , Classical Control และ SFO Algorithm

# การทดสอบและปรับเทียบ

- Classical Control

```
#include <ESP32Servo.h>
Servo myservo, myservoy;
int servoPin = 17;
int servoPiny = 16;
int pos_x = 0;
int pos_y = 0;

#define LDR1 A0 ///y
#define LDR2 A3 ///x
#define LDR3 A6 ///x
#define LDR4 A7 ///y
void setup()
{
    Serial.begin(9600);
    myservo.setPeriodHertz(50);
    myservo.attach(servoPin, 500, 2400);
    myservoy.setPeriodHertz(50);
    myservoy.attach(servoPiny, 500, 2400);
    pos_x = 90;
    pos_y = 120;
    myservo.write(pos_x);
    myservoy.write(pos_y);
}
```

```
float LDR_sensor(String Name, int LDR_Pin)
{
    float LDR = 0;
    float Volt = 0.0;
    float Lux = 0;
    if (Name != "3") {
        float LDR = analogRead(LDR_Pin);
        float Volt = (3.3 / 4096) * LDR;
        float Lux = 42.0 * pow(Volt, 3.15);
        // Serial.print("Lux ");
        // Serial.print(Name);
        // Serial.print(":");
        // Serial.println(Lux);
        return Lux;
    }
    else
    {
        float LDR = analogRead(LDR_Pin);
        float Volt = (3.3 / 4096) * (LDR + 300);
        float Lux = 42.0 * pow(Volt, 3.15);
        // Serial.print("Lux ");
        // Serial.print(Name);
        // Serial.print(":");
        // Serial.println(Lux);
        return Lux;
    }
}
```

# การทดสอบและปรับเทียบ

- Classical Control

```
void loop()
{
    float Lux_1 = LDR_sensor("1", LDR1);
    float Lux_2 = LDR_sensor("2", LDR2);
    float Lux_3 = LDR_sensor("3", LDR3);
    float Lux_4 = LDR_sensor("4", LDR4);
    if (abs(Lux_1 - Lux_4) > 100 || abs(Lux_2 - Lux_3) > 100 )
    {
        if (Lux_1 > Lux_2 && Lux_1 > Lux_3 && Lux_1 > Lux_4)
        {
            pos_y += map(Lux_1-Lux_4,-1800,1800,0,3);
            if (pos_y > 180)
            {
                pos_y = 180;
            }
            myservoy.write(pos_y);
        }

        else if (Lux_2 > Lux_1 && Lux_2 > Lux_3 && Lux_2 > Lux_4)
        {
            pos_x -= map(Lux_2-Lux_3,-1800,1800,0,3);
            if (pos_x < 0)
            {
                pos_x = 0;
            }
            myservo.write(pos_x);
        }
    }
}
```

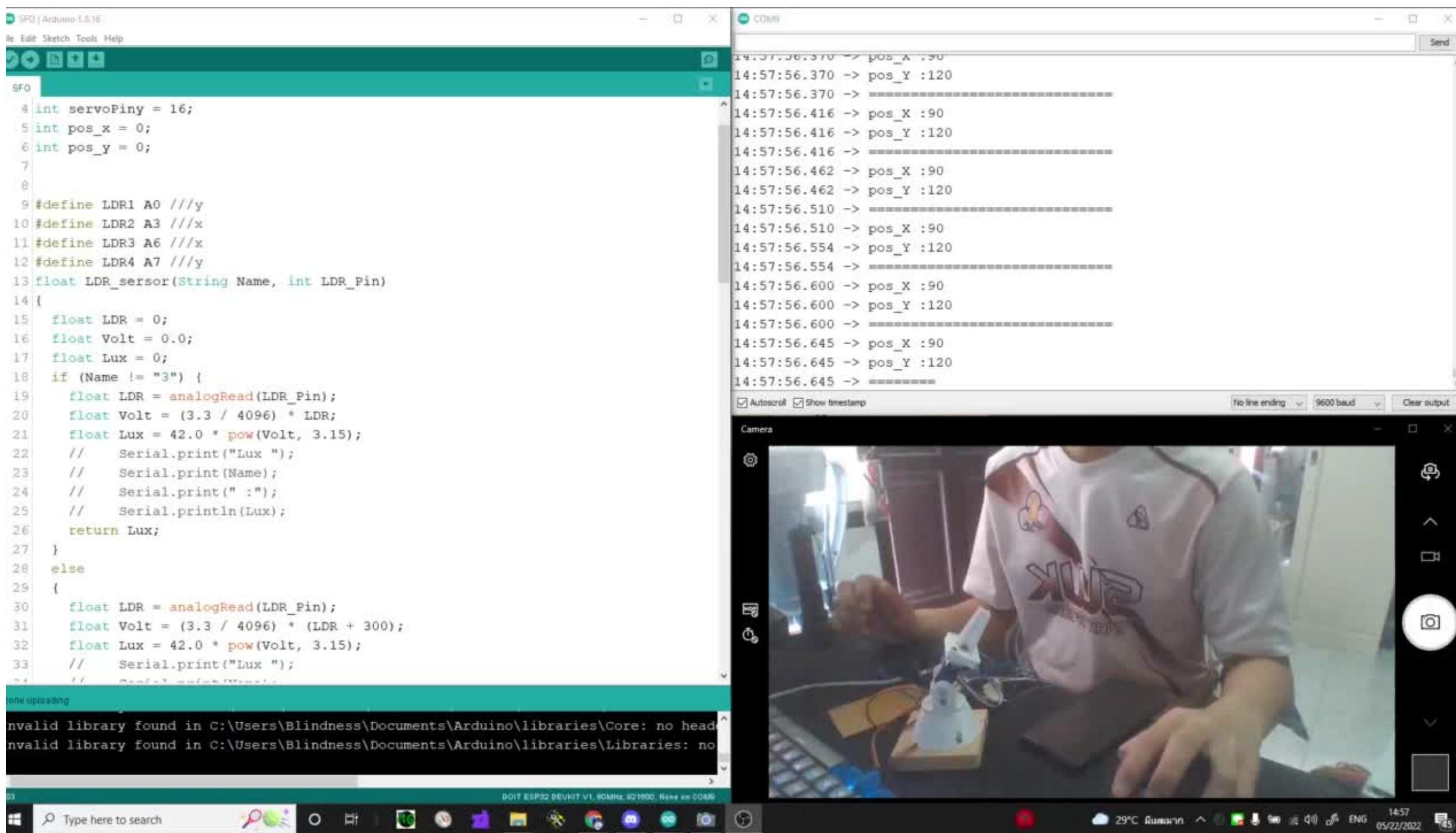
```
else if (Lux_3 > Lux_1 && Lux_3 > Lux_2 && Lux_3 > Lux_4)
{
    pos_x += map(Lux_3-Lux_2,-1800,1800,0,3);
    if (pos_x > 180)
    {
        pos_x = 180;
    }
    myservo.write(pos_x);
}

else if (Lux_4 > Lux_1 && Lux_4 > Lux_2 && Lux_4 > Lux_3)
{
    pos_y -= map(Lux_4-Lux_1,-1800,1800,0,3);
    if (pos_y < 0)
    {
        pos_y = 0;
    }
    myservoy.write(pos_y);
}

// Serial.print("pos_X :");
// Serial.println(pos_x);
// Serial.print("pos_Y :");
// Serial.println(pos_y);
// Serial.println("=====");
delay(50);
}
```

# การทดสอบและปรับเทียบ

- Classical Control



# การทดสอบและปรับเทียบ

- PID

```
#include <ESP32Servo.h>
Servo myservo,myservoy;
int servoPin = 17;
int servoPiny = 16;
double P = 25; //25
double I = 8; //8
double D = 1; //1
double Max = 84017.39; //72375
double Min = -95756.30; //-72375
double Max_y = 83364.40; //72375
double Min_y = -89044.80; //-72375
void Task1(void *pvParameter);
void Task2(void *pvParameter);
void Task3(void *pvParameter);
void Task4(void *pvParameter);
SemaphoreHandle_t xMutex;
#define LDR1 A0 ///y
#define LDR2 A3 ///x
#define LDR3 A6 ///x
#define LDR4 A7 ///y
///////////////////x//////////////////////
```

```
unsigned long currentTime, previousTime;
double dt = 0;
double Error = 0;
double d_Error = 0;
double p_Error = 0;
double i_Error = 0;
double ut = 0;
double Kp = P; //25
double Ki = I; //8
double Kd = D; //1
int valueLDR1 = 0;
int valueLDR2 = 0;
int servovalue1 = 90;
int Errorut1 = 0;
///////////////////y//////////////////////
unsigned long currentTime_y, previousTime_y;
double dt_y = 0;
double Error_y = 0;
double d_Error_y = 0;
double p_Error_y = 0;
double i_Error_y = 0;
double ut_y = 0;
double Kp_y = P;
double Ki_y = I;
double Kd_y = D;
int valueLDR3 = 0;
int valueLDR4 = 0;
int Errorut2 = 0;
int servovalue2 = 90;

void setup() {
    // put your setup code here, to run once:
    xMutex = xSemaphoreCreateMutex();
    Serial.begin(115200);
    myservo.setPeriodHertz(50);
    myservo.attach(servoPin, 500, 2400);
    myservoy.setPeriodHertz(50);
    myservoy.attach(servoPiny, 500, 2400);
    xTaskCreatePinnedToCore(Task1, "Task1", 1024, NULL, 50, NULL, 0);
    xTaskCreatePinnedToCore(Task2, "Task2", 1024, NULL, 10, NULL, 0);
    xTaskCreatePinnedToCore(Task3, "Task3", 1024, NULL, 5, NULL, 0);
    xTaskCreatePinnedToCore(Task4, "Task4", 1024, NULL, 0, NULL, 0);
    // myservo.write(servovalue1);
    // myservoy.write(servovalue2);
    currentTime = millis();
    previousTime = currentTime;
    currentTime_y = millis();
    previousTime_y = currentTime_y;
}
void loop() {
    // put your main code here, to run repeatedly:
    // Serial.print(analogRead(LDR1));
    // Serial.print(" ");
    // Serial.print(analogRead(LDR2));
    // Serial.print(" ");
    // Serial.print(analogRead(LDR3));
    // Serial.print(" ");
    // Serial.println(analogRead(LDR4));
}
```

# การทดสอบและปรับเทียบ

- PID

```
void Task1(void *pvParameter)
{
    while (1)
    {
        if (xSemaphoreTake(xMutex, portMAX_DELAY) == pdTRUE)
        {
            currentTime = millis();
            dt = (currentTime - previousTime) * 0.001;
            previousTime = currentTime;
            valueLDR1 = analogRead(LDR2);
            valueLDR2 = analogRead(LDR3) + 500;
            xSemaphoreGive(xMutex);
        }
        vTaskDelay(50);
    }
}

void Task2(void *pvParameter)
{
    while (1)
    {
        if (xSemaphoreTake(xMutex, portMAX_DELAY) == pdTRUE)
        {
            Error = valueLDR1 - valueLDR2;
            d_Error = (Error - p_Error) / dt;
            p_Error = Error;
            i_Error += Error * dt;
            ut = Kp * Error + Ki * i_Error + Kd * d_Error;
            Serial.print("X : ");
            Serial.print(valueLDR1);
            Serial.print(" ");
            Serial.print(valueLDR2);
            Serial.print(" ");
            Serial.print(Error);
            Serial.print(" ");

            Errorut1 = map(ut, Min, Max, 3, -3);
            servovalue1 += Errorut1;
            if (servovalue1 < 0)
                servovalue1 = 0;
            else if (servovalue1 > 180)
                servovalue1 = 180;
            myservo.write(servovalue1);
            Serial.print(Errorut1);
            Serial.print(" ");
            Serial.print(servovalue1);
            Serial.print("x: ");
            xSemaphoreGive(xMutex);
        }
        vTaskDelay(50);
    }
}
```

# การทดสอบและปรับเทียบ

- PID

```
void Task3(void *pvParameter)
{
    while (1)
    {
        if (xSemaphoreTake(xMutex, portMAX_DELAY) == pdTRUE)
        {
            currentTime_y = millis();
            dt_y = (currentTime_y - previousTime_y) * 0.001;
            previousTime_y = currentTime_y;
            valueLDR3 = analogRead(LDR1);
            valueLDR4 = analogRead(LDR4);
            xSemaphoreGive(xMutex);
        }
        vTaskDelay(50);
    }
}
```

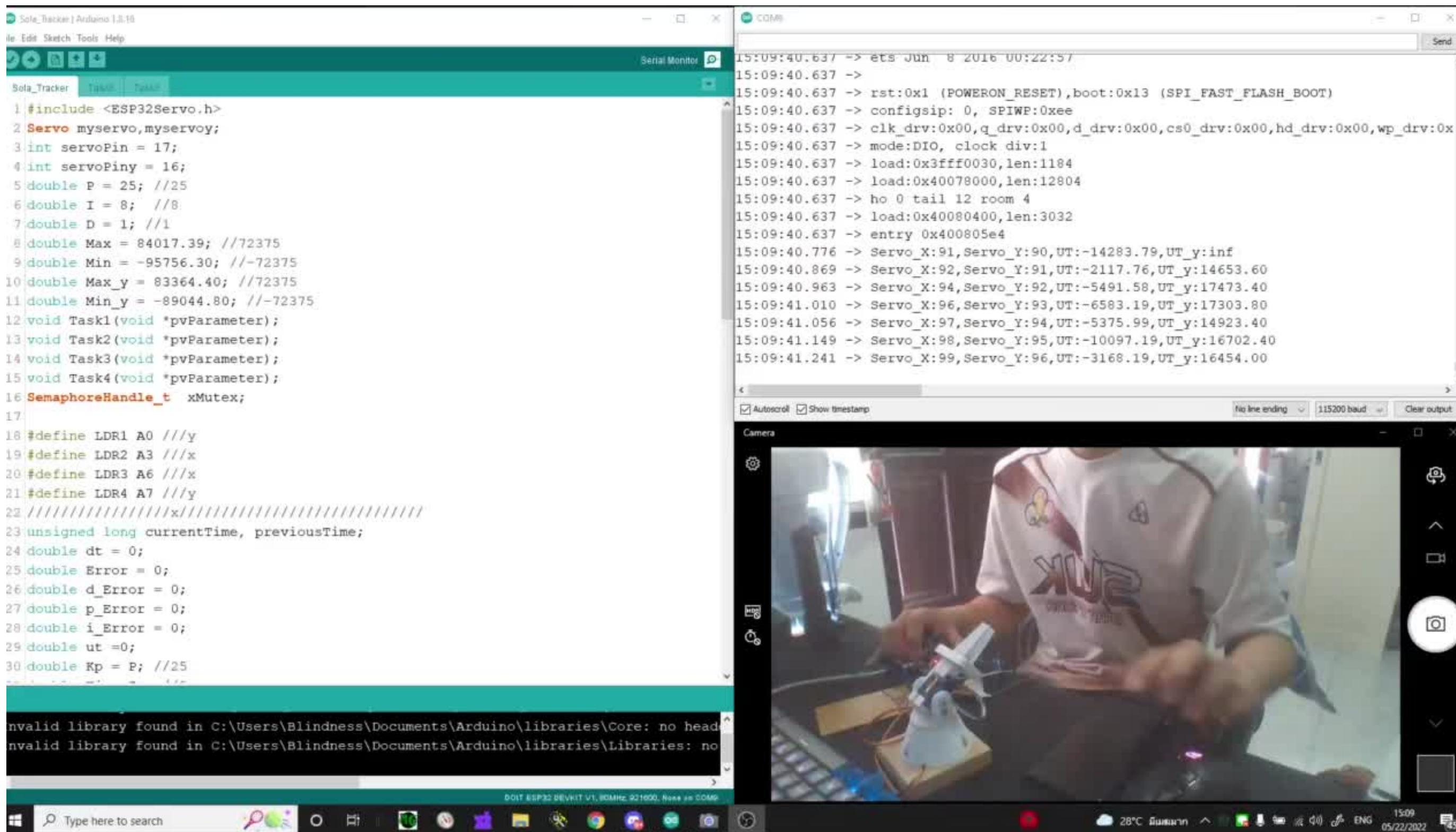
```
void Task4(void *pvParameter)
{
    while (1)
    {
        if (xSemaphoreTake(xMutex, portMAX_DELAY) == pdTRUE)
        {
            Error_y = valueLDR3 - valueLDR4;
            d_Error_y = (Error_y - p_Error_y) / dt_y;
            p_Error_y = Error_y;
            i_Error_y += Error_y * dt_y;
            ut_y = Kp_y * Error_y + Ki_y * i_Error_y + Kd_y * d_Error_y;
            // Serial.print("Y : ");
            // Serial.print(valueLDR3);
            // Serial.print(" ");
            // Serial.print(valueLDR4);
            // Serial.print(" ");
            // Serial.print(Error_y);
            // Serial.print(" ");

            Errorut2 = map(ut_y, Min_y, Max_y, -3, 3);
            servovalue2 += Errorut2;
            if (servovalue2 < 0)
                servovalue2 = 0;
            else if (servovalue2 > 180)
                servovalue2 = 180;

            myservoy.write(servovalue2);
            // Serial.print(Errorut2);
            // Serial.print(" ");
            // Serial.print(servovalue2);
            Serial.print("Servo_X:");
            Serial.print(servovalue1);
            Serial.print(',');
            Serial.print("Servo_Y:");
            Serial.print(servovalue2);
            Serial.print(',');
            Serial.print("UT:");
            Serial.print(ut);
            Serial.print(',');
            Serial.print("UT_y:");
            Serial.print(ut_y);
            Serial.println();
            xSemaphoreGive(xMutex);
        }
        vTaskDelay(50);
    }
}
```

# การทดสอบและปรับเทียบ

- PID



# การทดสอบและปรับเทียบ

## ● SFO Algorithm

```
#include <ESP32Servo.h>
Servo myservo, myservoy;
//Pin motor
int servoPin = 17;
int servopiny = 16;
int sensor_PIN[4] = {A0, A3, A6, A7};

//ค่าต่ำแหน่งขององค์ความอเดอร์
int pos_x = 0;
int pos_y = 0;

//ค่าที่ใช้เล่น
int sensor_LDR[4] = {0, 0, 0, 0};
int si_value[4] = {0, 0, 0, 0};
int di_value[4] = {0, 0, 0, 0};
int X_value[4] = {0, 0, 0, 0};
int MaxVal = 0;
int MinVal = 0;
int rad = 0;
int P = 1;
int N = 4;
int y_Update = 0;
int x_Update = 0;

//ค่าที่ใช้ในการแปลง LUX
float LDR = 0;
float Volt = 0;
float Lux = 0;
```

```
void setup() {
    Serial.begin(9600);
    myservo.setPeriodHertz(50);
    myservo.attach(servoPin, 500, 2400);
    myservoy.setPeriodHertz(50);
    myservoy.attach(servopiny, 500, 2400);
    pos_x = 90;
    pos_y = 120;
    myservo.write(pos_x);
    myservoy.write(pos_y);
}

int MaxValue(int sensor_LDR[] , int length) {
    int Max = 0;
    for (int i = 0; i < length ; i++) {
        if (sensor_LDR[i] > Max ) {
            Max = int(sensor_LDR[i]);
        }
    }
    return Max;
}

int MinValue(int sensor_LDR[] , int length) {
    int Min = 10000;
    for (int i = 0; i < length ; i++) {
        if (sensor_LDR[i] < Min ) {
            Min = int(sensor_LDR[i]);
        }
    }
    return Min;
}

int radian(int MaxVal , int MinVal , int N) {
    int d_max = (abs((MaxVal * 1000 / 57296) + (MinVal * 1000 / 57296)) / (2 * N));
    int rads = (d_max % 180) * 1000 / 57296;
    return rads;
}

int Count(int si_value[] , int length) {
    int count = 0;
    for (int i = 0; i < length ; i++) {
        if (si_value[i] == 0 ) {
            count += 1;
        }
    }
    return count;
}
```

# การทดสอบและปรับเทียบ

- SFO Algorithm

```
void loop() {  
    for (int i = 0; i < N ; i++) {  
        if (i != 2)  
        {  
            LDR = analogRead(sensor_PIN[i]);  
            Volt = (3.3 / 4096) * (LDR);  
            Lux = 42.0 * pow(Volt, 3.15);  
            sensor_LDR[i] = int(Lux);  
        }  
        else  
        {  
            LDR = analogRead(sensor_PIN[i]);  
            Volt = (3.3 / 4096) * (LDR+500);  
            Lux = 42.0 * pow(Volt, 3.15);  
            sensor_LDR[i] = int(Lux);  
        }  
    }  
  
    MaxVal = MaxValue( sensor_LDR , N );  
    MinVal = MaxValue( sensor_LDR , N );  
    rad = radian( MaxVal, MinVal, N );  
  
    for (int i = 0; i < N ; i++) {  
        if (MaxVal == sensor_LDR[i]) {  
            si_value[i] = int(0);  
        }  
        else {  
            int si = ( MaxVal - sensor_LDR[i] / abs(MaxVal - sensor_LDR[i]) ) ;  
            si_value[i] = int(si);  
        }  
        int di = ( rad * (P * ((sensor_LDR[i] + sensor_LDR[i - 1]) * abs(sensor_LDR[i] + sensor_LDR[i - 1]))) ) % 180;  
        di_value[i] = int(di);  
        int X = sensor_LDR[i] + di_value[i] * si_value[i];  
        X_value[i] = int(X) % 180;  
    }  
    int num = Count(si_value, N);  
    if ((num > 2 && si_value[1] == 0) || si_value[1] == 0) {  
        x_Update = ((abs(X_value[1] - X_value[2]) / 4) * (-1));  
    }  
    else {  
        x_Update = ((abs(X_value[1] - X_value[2]) / 4));  
    }  
  
    if ((num > 2 && si_value[3] == 0) || si_value[3] == 0) {  
        y_Update = ((abs(X_value[0] - X_value[3]) / 4) * (-1));  
    }  
    else {  
        y_Update = ((abs(X_value[0] - X_value[3]) / 4));  
    }  
}
```

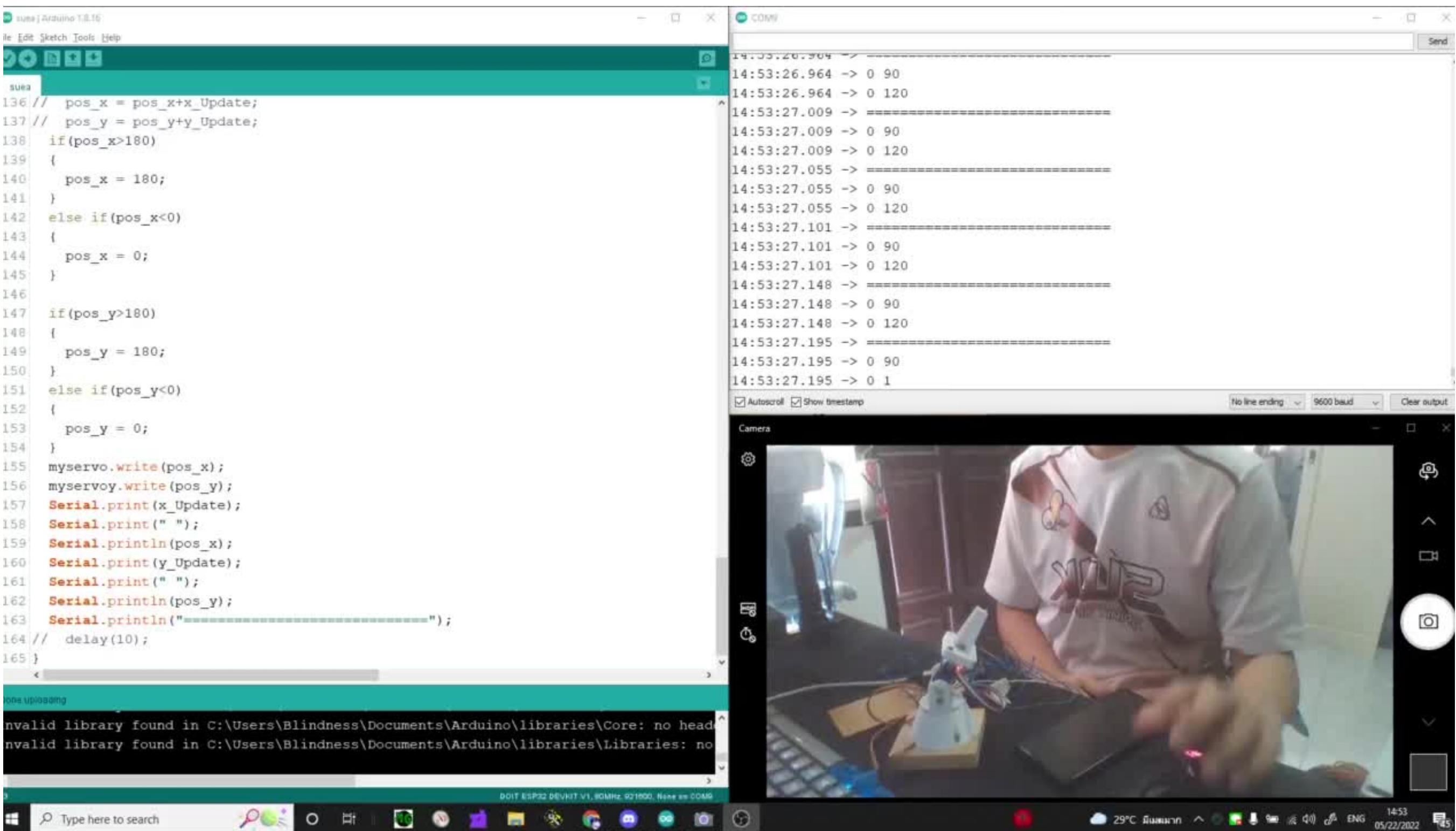
# การทดสอบและปรับเทียบ

- SFO Algorithm

```
// if (si_value[1] == 0 || si_value[3] == 0 || (num > 2 && ( si_value[1] == 0 || si_value[3] == 0)) || (num < 2 && ( si_value[1] == 0 || si_value[3] == 0))) {  
//     y_Update = ((abs(X_value[0] - X_value[3]) / 4) * (-1));  
//     x_Update = ((abs(X_value[1] - X_value[2]) / 4) * (-1));  
// }  
// else {  
//     y_Update = ((abs(X_value[0] - X_value[3]) / 4));  
//     x_Update = ((abs(X_value[1] - X_value[2]) / 4));  
// }  
pos_x = pos_x+map(x_Update,-43,43,-2,2);  
pos_y = pos_y+map(y_Update,-43,43,-2,2);  
// pos_x = pos_x+x_Update;  
// pos_y = pos_y+y_Update;  
if(pos_x>180)  
{  
    pos_x = 180;  
}  
else if(pos_x<0)  
{  
    pos_x = 0;  
}  
  
if(pos_y>180)  
{  
    pos_y = 180;  
}  
else if(pos_y<0)  
{  
    pos_y = 0;  
}  
  
myservo.write(pos_x);  
myservoy.write(pos_y);  
Serial.print(x_Update);  
Serial.print(" ");  
Serial.print(pos_x);  
Serial.print(y_Update);  
Serial.print(" ");  
Serial.println(pos_y);  
Serial.println("=====");  
// delay(10);  
}
```

# การทดสอบและปรับเทียบ

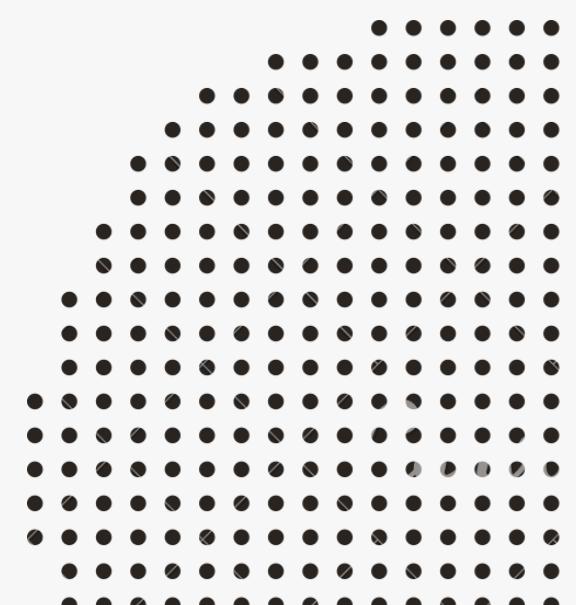
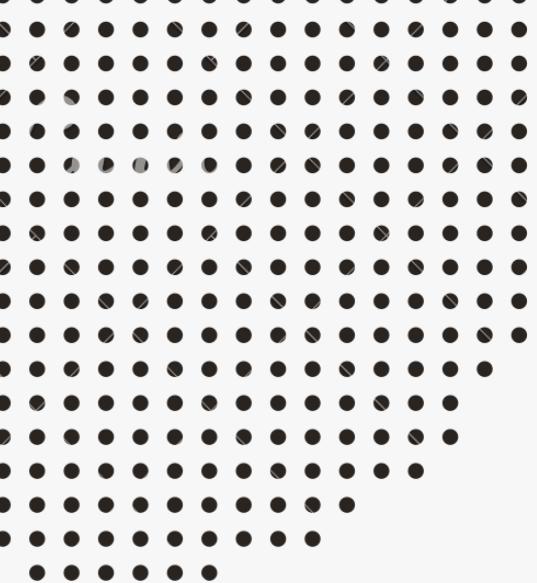
- SFO Algorithm



# การทดสอบและเปรียบเทียบ

## • ตารางการเปรียบเทียบ

ประเภท หัวข้อความแตกต่าง	Classical Control	PID	SFO Algorithm
1.ความซับซ้อน	มีความซับซ้อนต่ำและใช้งานง่าย	มีความซับซ้อนสูงและเข้าใจได้ยาก	มีความซับซ้อนปานกลางแต่ต้องมีความเข้าใจใน Algorithm
2.รูปแบบการทำงาน	ทำงานได้ดี แต่ไม่สามารถคิดเองได้ในบางสถานการณ์	ทำงานได้ดี สามารถทำงานได้ด้วยเอง มีความเสถียรสูง	ทำงานได้ดี สามารถทำงานได้เองในเกือบทุกสถานการณ์ มีความเสถียรสูง
3.การควบคุม	ควบคุมผ่านตัวแปรที่มีค่ามากที่สุด ดังนั้นจึงเลือกได้เพียงค่าในการแบบเข้าหาแสงในแต่ละครั้ง	ควบคุมผ่านสมการ PID ซึ่งจะลู่เข้าสู่ค่าที่มีความเสถียรที่สุดเสมอ ไม่ตাযตัว	ควบคุมโดยตัวแปรต่างๆ กี เกี่ยวข้องทำให้สามารถเปลี่ยนแปลงการแบบเข้าหาแสงได้ตลอดเวลาและจะทำงานไปเรื่อยจนกว่าจะไม่มีแสง



# Question & Thank You