Assignment 1

สมาชิกกลุ่ม LOCALHOST

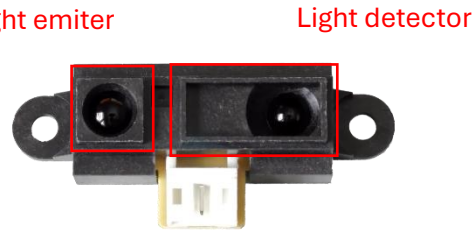
1. **6610110425 คีตศิลป์ คงสี**
2. **6610110034 คุณานนต์ หนูแสง**
3. **6610110327 สิรวิชญ์ น้อยผา**
4. **6610110341 สุธินันท์ รองพล**

Sensor

การติดตั้ง sensor

ผู้รับผิดชอบ: 6610110341 นายสุธินันท์ รองพล

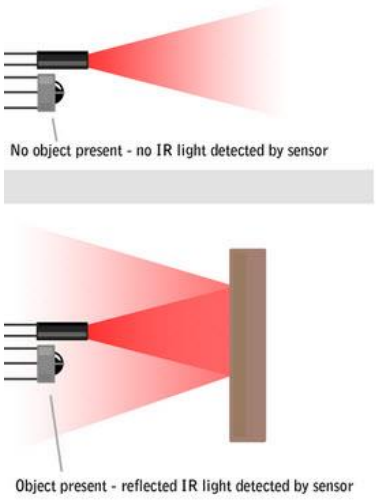
Sharp infrared distance sensor คือเซ็นเซอร์วัดระยะด้วยอินฟราเรด โดยมี output เป็นสัญญาณอนาล็อก สามารถวัดระยะได้ในช่วง 4 - 30 cm (รุ่น GP2Y0A51SK0F)



รูปที่1 Sharp infrared distance sensor

หลักการทำงานของ Infrared distance sensor

หลักการทำงานของ Infrared distance sensor ประกอบด้วย Light emitter (ตัวปล่อยแสง) และ Light detector (ตัวรับแสง) ซึ่งใช้ในการตรวจจับและวัดระยะทางโดยอาศัยการสะท้อนกลับของแสงอินฟราเรด



รูปที่ 2 หลักการทำงานของ Sharp infrared distance sensor

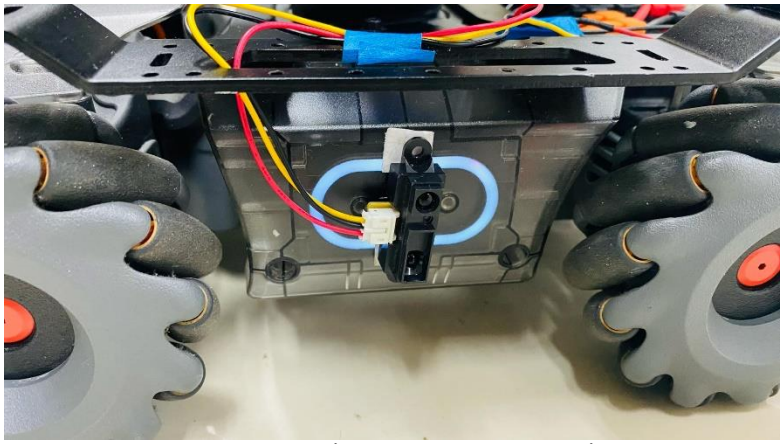
Port การเชื่อมต่อ

- แดง(VCC) --> 5V
- ดำ(GND)
- เหลือง(OUT)

ตำแหน่งการติดตั้งเซ็นเซอร์บนตัวหุ่น

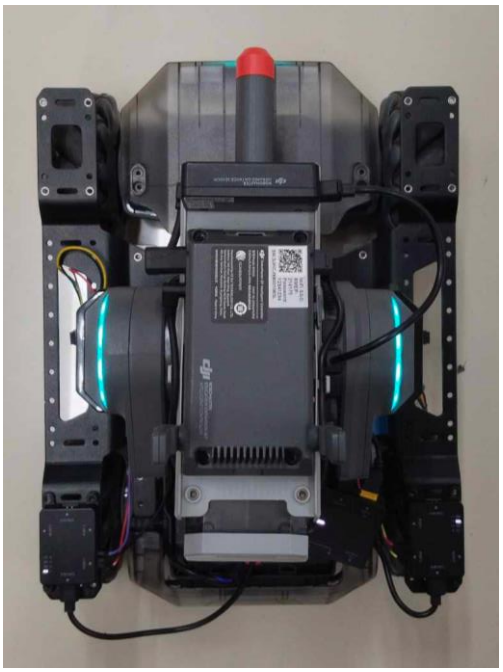
การติดตั้ง sharp sensor ได้ลองติดตั้ง 2 แบบ ติดตั้งเซ็นเซอร์ในแนวนอน และติดตั้งเซ็นเซอร์ในแนวตั้ง

ได้ผลสรุปว่า การติดตั้งเซ็นเซอร์ในแนวตั้งวัดค่าได้แม่นยำขึ้น



รูปที่ 3: การติดตั้งเซ็นเซอร์ Sharp ในแนวตั้ง

เซ็นเซอร์ Sharp แต่ละตัวจะเชื่อมต่อกับฮับ โดยฝั่งซ้ายเชื่อมต่อกับ PORT 1 และฝั่งขวาเชื่อมต่อกับ PORT 2



รูปที่ 4: มุมมองจากข้างบน

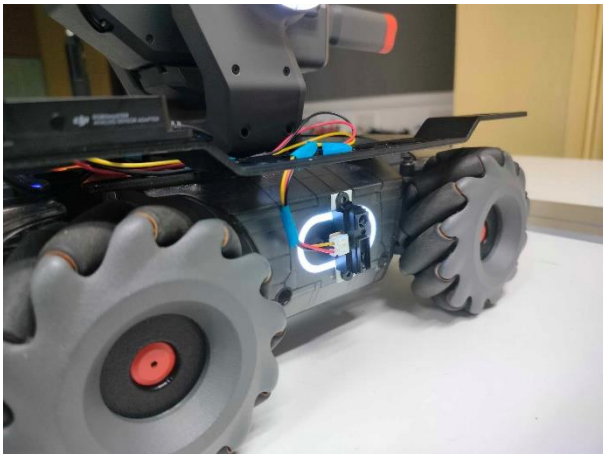
การติดตั้งเซ็นเซอร์ด้านข้างของหุ่นยนต์ RoboMaster ทั้งสองด้านบริเวณ LED hit sensor มีข้อดีดังนี้:

1. Chassis Extension Platform ช่วยป้องกันความเสียหายจากการชนหรือกระแทกกับวัตถุภายนอก

2. Infrared distance sensor ที่ใช้งานมีระยะการวัด 4 – 30 cm ซึ่งเมื่อระยะน้อยกว่า 4 cm อาจเกิดความคลาดเคลื่อนที่ทำให้โปรแกรมคำนวณผิดพลาดได้ โดย Chassis Extension Platform มีความยาวเพียงพอที่จะรองรับและเก็บระยะก่อนถึง 4 cm ได้

ข้อควรระวัง

ในการติดตั้งเซ็นเซอร์ Sharp ต้องระวังการต่อสายไฟ และควรเก็บสายไฟให้เรียบร้อยเพื่อไม่ให้ไปรบกวนสิ่งต่างๆ ที่อาจทำให้เกิดข้อผิดพลาดในขณะทดสอบ เช่น ตัวล้อ Gimbal



รูปที่ 5: วิธีการเก็บสายไฟของเซ็นเซอร์ Sharp

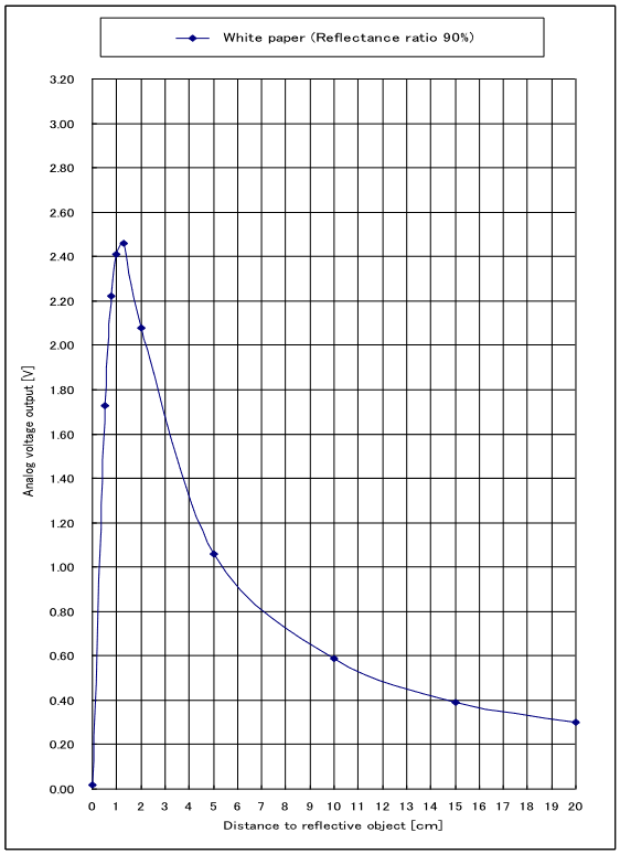
ปัญหาที่พบ

เซ็นเซอร์ Sharp แต่ละตัวมีประสิทธิภาพแตกต่างกัน บางตัววัดระยะได้แม่นยำ ขณะที่บางตัวอาจมีคลาดเคลื่อน และบางตัวสามารถวัดระยะทางได้แม้ไม่มีสิ่งกีดขวาง

Sharp Sensor & Filter

ผู้รับผิดชอบ: 6610110034 คุณานนต์ หนูแสง

ค่าระยะทางที่ได้จาก Sharp Sensor (GP2Y0A51SK0F) เป็นสัญญาณที่ถูกแปลงจากสัญญาณแอนะล็อคเป็นสัญญาณดิจิทัล หรือค่า ADC (A-D converter) การรับค่า ADC ในโปรแกรมนั้นจะต้องเรียกใช้ฟังก์ชัน sub_adapter() ที่ได้จากเอาต์พุตของเซ็นเซอร์เพื่อส่งค่า ADC ไปยังฟังก์ชันที่ใช้ในการแปลงเป็นค่า Voltage และใช้ในการแปลงเป็นระยะทางในหน่วยเซนติเมตร (cm)



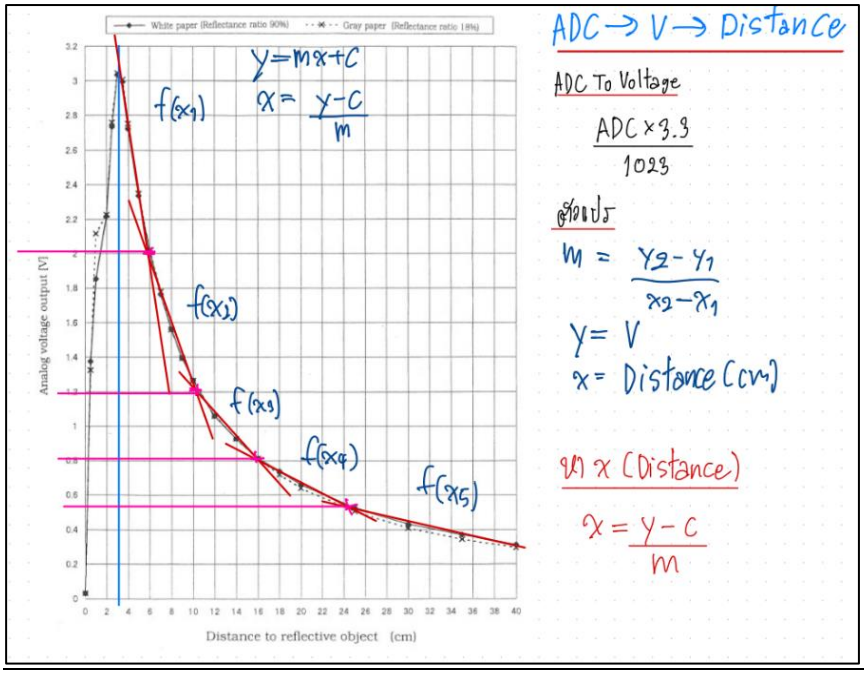
รูปที่ 6 กราฟนี้แสดงความสัมพันธ์ระหว่างแรงดันไฟฟ้าที่เอาต์พุตจากเซ็นเซอร์ (แกน Y) กับระยะห่างจากวัตถุสะท้อนแสง (แกน X)

จากความสัมพันธ์ระหว่างแรงดันไฟฟ้า (Voltage ในแนวแกน Y) กับระยะห่างจากวัตถุสะท้อนแสง (หน่วยเซนติเมตร ในแนวแกน X) สามารถใช้ในการเทียบจากแรงดันไฟฟ้าเป็นระยะทางได้โดยการใช้เส้นตรงในการแบ่งเป็นช่วงๆในกราฟดังกล่าวเพื่อประมาณค่าระยะทางที่ได้ไปใช้กับการรักษาระยะห่างระหว่างหุ่น Robomaster กับกำแพงโฟมในการเดินในเขาวงกต รวมไปถึงการ

ตรวจสอบว่ามีกำแพงโฟมหรือไม่ทั้งกรณีมีกำแพงโฟม หรือไม่มีกำแพงโฟมก็จะส่งผลต่อการตัดสินใจในการเดินในเขาวงกตของ Robomaster

ขั้นตอนการแบ่งกราฟเป็นช่วงๆโดยใช้เส้นตรง

กำหนดเส้นตรงให้ครอบคลุมเส้นโค้งเพื่อเป็นการกำหนดช่วงจาก



รูปที่ 7 รูปภาพแสดงการแบ่งเป็นช่วงๆโดยใช้เส้นตรงในการแบ่ง

จากกราฟได้แบ่งไว้เป็น 5 ช่วง

สมการเส้นตรง

y = mx + c โดย y : voltage, m : ความชันในช่วงนั้นๆ, x : distance, c : จุดตัดที่แกน y ในช่วงนั้นๆ

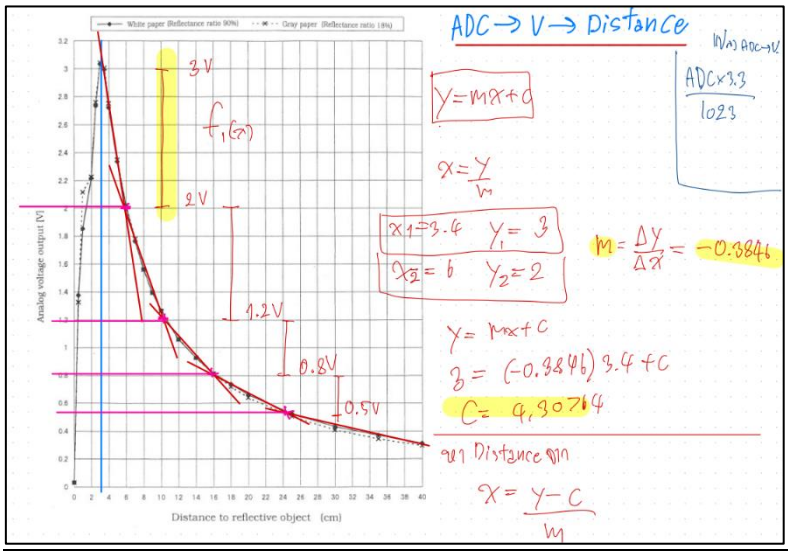
เมื่อลองย้ายข้างสมการเพื่อหาระยะทาง

x = y – c / m

เริ่มต้นจากการแปลงค่าจาก ADC เป็น Voltage โดยการใช้บัญญัติไตรยางศ์ในการเปรียบเทียบ เนื่องจากจำนวน ADC ที่วัดได้อยู่ที่ 0 – 1023 และ Voltage อยู่ระหว่าง 0V – 3.3V จะได้เป็น

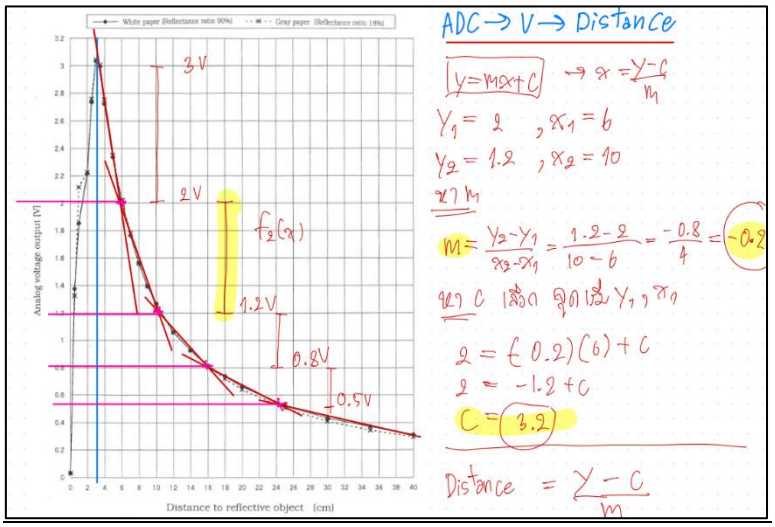
ADC x 3.3 / 1023

ช่วงที่1 : จาก 3V ถึง 2V



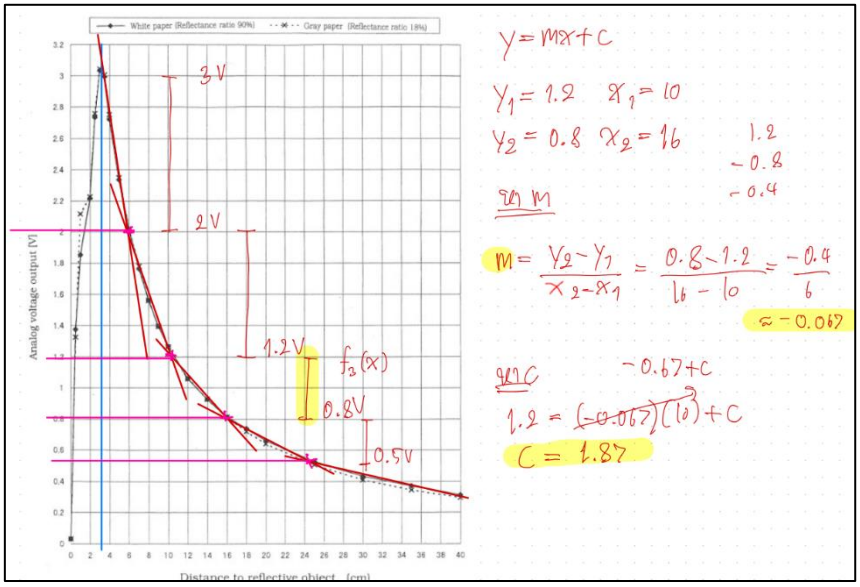
รูปที่ 8 การแบ่งช่วง 3V ถึง 2V

ช่วงที่2 : จาก 2V ถึง 1.2V



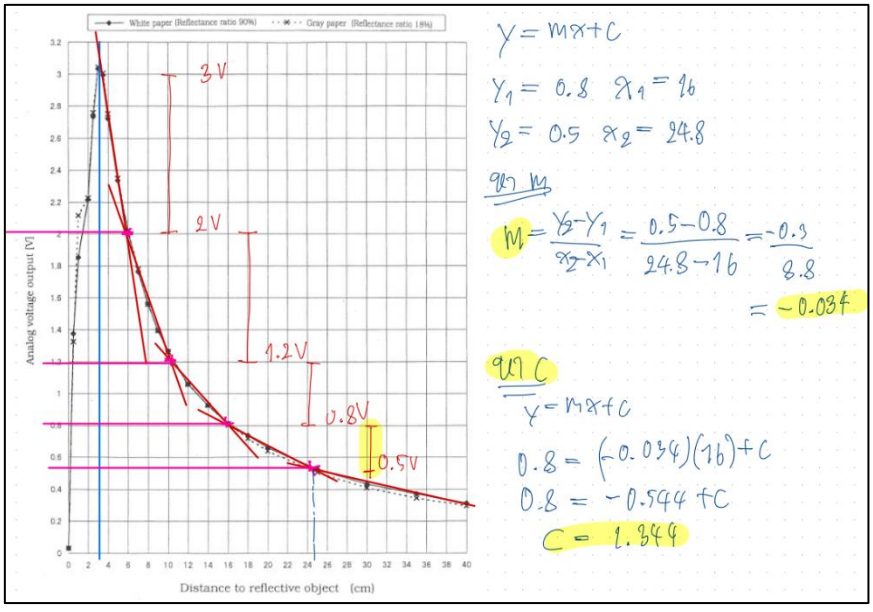
รูปที่ 9 การแบ่งช่วง 2V ถึง 1.2V

ช่วงที่3 : จาก 1.2V ถึง 0.8V



รูปที่ 10 การแบ่งช่วง 1.2V ถึง 0.8V

ช่วงที่4 : จาก 0.8V ถึง 0.5V



รูปที่ 11 การแบ่งช่วง 0.8V ถึง 0.5V

ช่วงที่5 : จาก 0.5V เป็นต้นไปจะไม่ถูกนำมาใช้เนื่องจากค่าที่ได้จากการวัดนั้นคลาดเคลื่อนกับในความเป็นจริงเป็นอย่างมาก

ในแต่ละช่วงที่ได้แบ่งโดยใช้เส้นตรง ใช้ในการหาค่า m (ความชัน)ในช่วงนั้นๆ โดยหาได้จากการนำค่าความดันไฟฟ้าล่าสุด – ค่าความดันไฟฟ้าเริ่มต้น / ระยะทางล่าสุด - ระยะทางเริ่มต้นดังสมการ

$$m = y2 - y1 / x2 - x1$$

และใช้ในการหาค่า c (จุดตัดที่แกน y)ในช่วงนั้นๆ โดยหาได้จาก

$$c = y - mx$$

เมื่อได้ค่า m และ ค่า c ในช่วงต่างๆ แล้วสามารถนำไปใช้ต่อในการหาค่า x (ระยะทาง) โดยมี Input เป็นค่า y (แรงดันไฟฟ้า) เป็นตัวกำหนดช่วงที่ต้องใช้ในการหาระยะทาง ตามการทำงานของฟังก์ชัน sub_data_handler

```
def sub_data_handler(sub_info):
    io_data, ad_data = sub_info


    distances = []
    for adc_value in ad_data:
        voltage = adc_value * 3.3 / 1023


        if 2.2 <= voltage < 3.2:
            distance = (voltage - 4.30764) / -0.3846
        elif 1.4 <= voltage < 2.2:
            distance = (voltage - 3.2) / -0.2
        elif 0.8 <= voltage < 1.4:
            distance = (voltage - 1.87) / -0.067
        elif 0.4 <= voltage < 0.8:
            distance = (voltage - 1.344) / -0.034
        else:
            if voltage >= 3.2:
                distance = (voltage - 4.30764) / -0.3846
```

```
        elif voltage < 0.4:
            distance = 50.0

    distances.append(distance)

left = sum(distances[0:2]) / 2
right = sum(distances[2:4]) / 2
left_data.append(left)
right_data.append(right)
   print(f"port1 left: {left}, port2 right: {right}")

return distances
```

<u>คำอธิบายหลักการทำงาน</u>

- ใช้ for ลูปในการลูปค่า list ของ ad_data เพื่อแปลงค่าจาก ADC เป็น Voltage
- ใช้เงื่อนไขในการเช็คว่าค่า Voltage ที่ได้จากการวัดควรใช้ค่า m และ c ในช่วงใดบางมีตั้งแต่ช่วงที่ 1 ถึง 4
- เนื่องจากค่าที่ได้จากการวัดด้วย Sharp Sensor ประกอบไปด้วย 2 ค่าใน 1 เซ็นเซอร์ ยกตัวอย่างเช่น port1(left) : [153, 154], port2(right) : [243, 239] จึงต้องหาค่าเฉลี่ยทั้งสองค่านี้เพื่อให้ได้มาซึ่งค่าของระยะห่างจากกำแพงทางด้านซ้ายและด้านขวา
- เก็บค่าระยะห่างจากกำแพงทางด้านซ้ายและด้านขวาไว้ใน list ของ distances
- Return ค่า distances ออกไปเพื่อใช้งานในส่วนของ Logic

การประยุกต์หลักการ First-order Infinite Impulse Response มาใช้ใน Sharp Sensor

First-order Infinite Impulse Response (IIR) Filter หรือที่เรียกอีกอย่างว่า Recursive Filter ใช้ในการลดสัญญาณรบกวน (Noise) หลักการทำงานนี้คือจะเป็นการคำนวณค่า Output y[n] จากค่า Input x[n] และค่า Outputก่อนหน้า y[n-1] โดยมีตัวแปร α ที่เป็นค่าสัมประสิทธิ์ของ Filter ที่ควบคุมการตอบสนองของ Filter นี้ โดยมีสูตรที่ใช้คือ

$$y[n] = \alpha \cdot y[n-1] + x[n]$$

ผลของ α :

- ถ้า α มีค่าใกล้เคียงกับ 1 จะทำให้การตอบสนองของฟิลเตอร์ช้าลง และลดสัญญาณรบกวนได้มาก เนื่องจากค่า y[n] จะขึ้นอยู่กับค่าผลลัพธ์ก่อนหน้ามากกว่า x[n]
- ถ้า α มีค่าใกล้เคียงกับ 0 จะทำให้การตอบสนองของฟิลเตอร์เร็วขึ้น และลดสัญญาณรบกวนได้น้อยลง เนื่องจากค่า y[n] ขะขึ้นอยู่กับค่า x[n] มากกว่า

เนื่องจากการทดสอบจริง Filter นี้อาจช่วยลดสัญญาณรบกวนได้ไม่ดีพอจึงต้องมีการควบคุมความสมดุลระหว่างสัดส่วนค่าผลลัพธ์เก่า y[n-1] และ x[n] ที่เป็น Input จึงเพิ่มในส่วนของ 1 – α ในสมการ

$$y[n] = \alpha \cdot y[n-1] + (1-\alpha) \cdot x[n]$$

สูตรนี้ช่วยให้ค่าที่ได้มีความสมดุลระหว่างข้อมูลเดิม (ค่าเก่า) และข้อมูลใหม่ (ค่า ADC ที่อ่านได้ล่าสุด) ทำให้ผลลัพธ์มีความราบรื่นและไม่เกิดการเปลี่ยนแปลงอย่างฉับพลันเกินไป

ฟังก์ชัน filter_adc_value ใช้ในการ Filter ค่า ADC เพื่อลด Noise ที่เกิดขึ้น

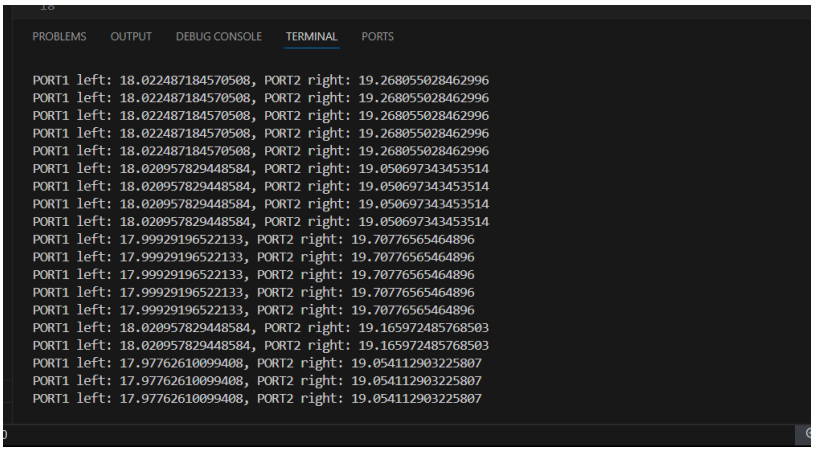
```
def filter_adc_value(ad_data):
    y_filtered = []
    alpha = 0.1  # ตั้งค่า alpha
    y_prev = 0  # ค่าเริ่มต้นของ y[n-1]
    for adc_value in ad_data: # 153, 153, 203, 203
        y_current = alpha * y_prev + (1 - alpha) * adc_value
        y_filtered.append(y_current)
        y_prev = y_current

    return y_filtered # 150, 150, 200, 200
```

<u>สรุปการใช้ Filter</u>

การใช้ Filter ช่วยในการลดสัญญาณรบกวนที่เกิดขึ้นได้จริง ซึ่งช่วยให้ค่าที่ได้จากการวัดระยะห่างจากหุ่น Robomaster ให้มีความคลาดเคลื่อนที่น้อยลง

ผลการทดสอบ

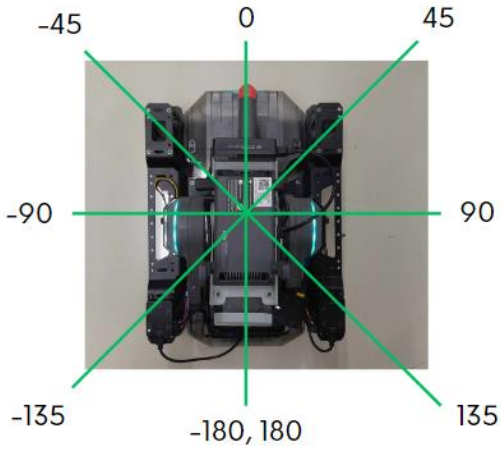
รูปที่ 12 วิธีการวัดระยะของเซ็นเซอร์ Sharp



รูปที่ 13 ผลการวัดระยะของเซ็นเซอร์ Sharp

Adjust angle

ผู้รับผิดชอบ : สิรวิชญ์ น้อยผา 6610110327



รูปที่ 14 ภาพจำลองมุมของหุ่น robomaster

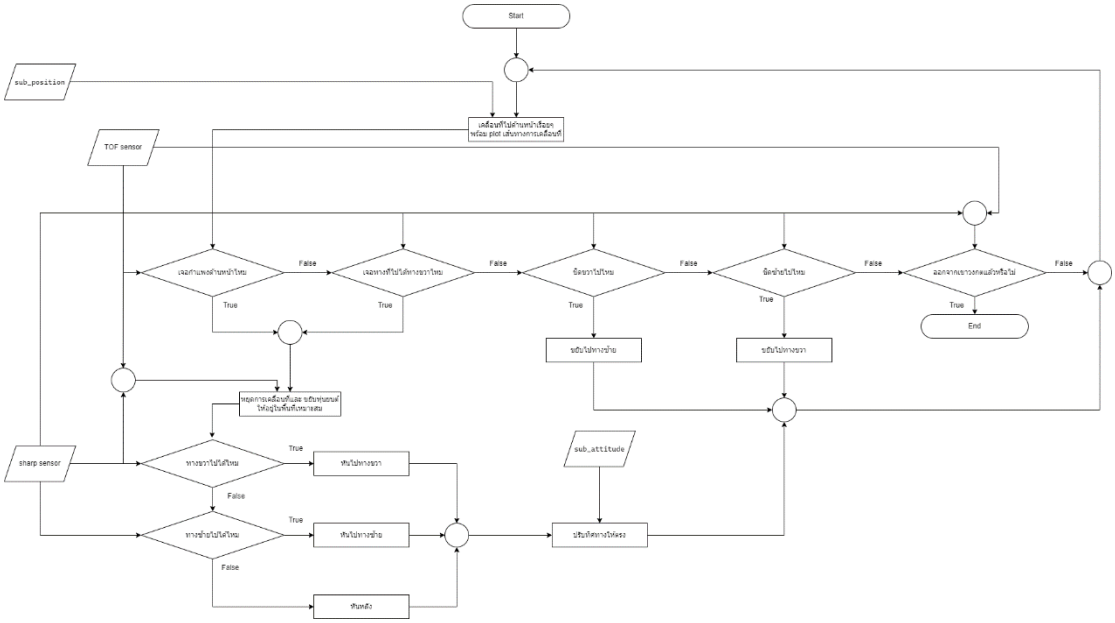
ฟังก์ชันนี้มีใช้สำหรับการปรับให้ตำแหน่งรถตั้งตรง เมื่อรถมีการเอียงหลังจากการเคลื่อนที่ โดยการทำงานของฟังก์ชันนี้จะใช้ sensor sub_attitude ในการอ่านค่าตำแหน่งมุมของรถ เพื่อรับค่ามุมองศาของรถที่อ่านค่าได้ ณ ปัจจุบันหลังจากการหยุดเพื่อปรับมุมรถก่อนที่จะเคลื่อนที่ไปต่อในขั้นต่อไป โดยที่จะตรวจสอบเงื่อนไขว่าค่ามุมที่อ่านค่าได้นั้นอยู่ในช่วงของมุมเป้าหมายที่เราจะทำให้รถตั้งตรงหรือไม่ ซึ่งมุมเป้าหมายที่เรากำหนดว่ารถนั้นจะตั้งตรงหรือไม่ ก็จะมีมุมเป้าหมายเป็น

- มุม -180 องศา หากมุมที่อ่านค่าได้อยู่ในช่วงมุม -180 องศา ถึง มุม -135 องศา
- มุม -90 องศา หากมุมที่อ่านค่าได้อยู่ในช่วงมุม -135 องศา ถึง มุม -45 องศา
- มุม 0 องศา หากมุมที่อ่านค่าได้อยู่ในช่วงมุม -45 องศา ถึงมุม 45 องศา
- มุม 90 องศา หากมุมที่อ่านค่าได้อยู่ในช่วงมุม 45 องศา ถึง มุม 135 องศา
- มุม 180 องศา หากมุมที่อ่านค่าได้อยู่ในช่วงมุม 135 องศา ถึง มุม 180 องศา

สาเหตุที่กำหนดมุมเช่นนี้ เป็นเพราะว่า sensor sub_attitude จะอ่านค่ามุมของหุ่นยนต์ (ค่า z) ตั้งแต่ค่า -180 จนถึง 180 องศา โดยการปรับมุมองศาของรถเพื่อเข้าสู่มุมเป้าหมายจะใช้ฟังก์ชัน move ของ ep_chassis โดยปรับที่ค่า z โดยที่ตั้งค่าเป็น (มุมที่อ่านได้ ณ ปัจจุบัน – มุมเป้าหมาย)

Logic

ผู้รับผิดชอบ: 6610110425 นายคีตศิลป์ คงสี



รูปที่ 15 Flowchart Logic การทำงานของหุ่นยนต์

- หุ่นยนต์จะเคลื่อนที่ไปข้างหน้าเรื่อยๆ และบันทึกเส้นทางการเคลื่อนที่ โดยเก็บข้อมูลตำแหน่งปัจจุบันจาก sub_position ไว้ในlist พร้อมplot เส้นทางการเคลื่อนที่แบบreal time

- เจอกำแพงด้านหน้าไหม?
  - อ่านค่าระยะห่างจากกำแพงทางด้านหน้า จากTOF เจอกำแพงไหม(ค่าจากTOF น้อยกว่า300mm.)
  - True: หุ่นยนต์หยุดการเคลื่อนที่และ ขยับหุ่นยนต์ให้อยู่ในพื้นที่เหมาะสม ไม่ใกล้กำแพงมากเกินไป โดยอ่านค่าระยะห่างจากกำแพงทั้ง3ด้าน จากSharpทั้ง2ข้าง และTOF
    - ทางขวาไปได้ไหม? อ่านค่าระยะห่างจากกำแพงทางด้านขวา จากSharpข้างขวา ไม่เจอกำแพงใช่ไหม(ค่าจากsharp มากกว่าช่วงที่สามารถวัดได้)

- True: หันไปทางขวา (เมื่อเข้าเงื่อนไขนี้แล้ว จะเคลื่อนที่ไปด้านหน้าในขั้นตอนต่อไป จะไม่เข้าเงื่อนไข“เจอทางที่ไปได้ทางขวาไหม?”เนื่องจาก มีค่าstatus)
- False: อยู่นิ่งไม่เคลื่อนที่ พร้อมเช็คเงื่อนไขถัดไป

- ทางซ้ายไปได้ไหม? อ่านค่าระยะห่างจากกำแพงทางด้านซ้าย จากSharpข้างซ้าย ไม่เจอกำแพงใช่ไหม(ค่าจากsharp มากกว่าช่วงที่สามารถวัดได้)
  - True: หันไปทางซ้าย
  - False: หันหลัง
- หุ่นยนต์จะปรับทิศทาง(yaw) ตามข้อมูลจากsub_attitude เพื่อเคลื่อนที่ไปในทิศทางที่ เหมาะสม (ไม่เอียง)

- False: เคลื่อนที่ไปข้างหน้าเรื่อยๆ พร้อมเช็คเงื่อนไขถัดไป

➢ เจอทางที่ไปได้ทางขวาไหม?

- อ่านค่าระยะห่างจากกำแพงทางด้านขวา จากSharpข้างขวา ไม่เจอกำแพงใช่ไหม(ค่าจากsharp มากกว่าช่วงที่สามารถวัดได้)
- True: หุ่นยนต์หยุดการเคลื่อนที่และ ขยับหุ่นยนต์ให้อยู่ในพื้นที่เหมาะสม ไม่ใกล้กำแพงมากเกินไป โดยอ่านค่าระยะห่างจากกำแพงทั้ง3ด้าน จากSharpทั้ง2ข้าง และTOF
  - ทางขวาไปได้ไหม? อ่านค่าระยะห่างจากกำแพงทางด้านขวา จากSharpข้างขวา ไม่เจอกำแพงใช่ไหม(ค่าจากsharp มากกว่าช่วงที่สามารถวัดได้)
    - True: หันไปทางขวา (เมื่อเข้าเงื่อนไขนี้แล้ว จะเคลื่อนที่ไปด้านหน้าในขั้นตอนต่อไป จะไม่เข้าเงื่อนไข“เจอทางที่ไปได้ทางขวาไหม?”เนื่องจาก มีค่าstatus)
    - False: อยู่นิ่งไม่เคลื่อนที่ พร้อมเช็คเงื่อนไขถัดไป
  - ทางซ้ายไปได้ไหม? อ่านค่าระยะห่างจากกำแพงทางด้านซ้าย จากSharpข้างซ้าย ไม่เจอกำแพงใช่ไหม(ค่าจากsharp มากกว่าช่วงที่สามารถวัดได้)
    - True: หันไปทางซ้าย
    - False: หันหลัง

- หุ่นยนต์จะปรับทิศทาง(yaw) ตามข้อมูลจากsub_attitude เพื่อเคลื่อนที่ไปในทิศทางที่ เหมาะสม (ไม่เอียง)

- False: เคลื่อนที่ไปข้างหน้าเรื่อยๆ พร้อมเช็คเงื่อนไขถัดไป

➢ ชิดขวาไปไหม?

- อ่านค่าระยะห่างจากกำแพงทางด้านขวา จากSharpข้างขวา น้อยกว่า10cm.ไหม
- True: หุ่นยนต์จะเคลื่อนที่ไปทางซ้าย
- False: เคลื่อนที่ไปข้างหน้าเรื่อยๆ พร้อมเช็คเงื่อนไขถัดไป

➢ ชิดซ้ายไปไหม?

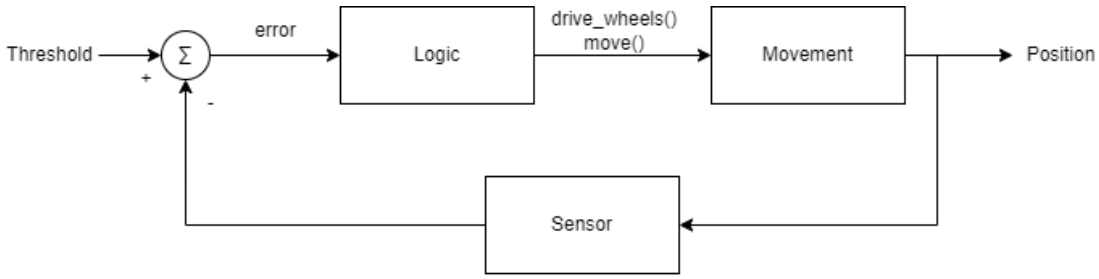
- อ่านค่าระยะห่างจากกำแพงทางด้านซ้าย จากSharpข้างซ้าย น้อยกว่า10cm.ไหม
- True: หุ่นยนต์จะเคลื่อนที่ไปทางขวา
- False: เคลื่อนที่ไปข้างหน้าเรื่อยๆ พร้อมเช็คเงื่อนไขถัดไป

➢ ออกจากเขาวงกตแล้วหรือไม่?

- อ่านค่าระยะห่างจากกำแพงทางด้านซ้ายและขวา จากSharpข้างซ้ายและขวา พร้อมอ่านค่าระยะห่างจากกำแพงทางด้านหน้า จากTOF หากทั้ง3ค่ามากเกินไป(มากกว่าค่าที่ตั้งไว้ ซึ่งไม่ใช่3แยก) ถือว่าออกจากเขาวงกต
- True: หุ่นยนต์จะหยุดการเคลื่อนที่ และหยุดการทำงาน(End)
- False: เคลื่อนที่ไปข้างหน้าเรื่อยๆ พร้อมกลับไปเช็คเงื่อนไขแรก

Block Diagram Closed Loop Control System

ผู้รับผิดชอบ: 6610110425 นายคีตศิลป์ คงสี



รูปที่ 16 Block Diagram Closed Loop Control System

- Input:
  - เกณฑ์ระยะห่างขั้นต่ำจากกำแพงเพื่อป้องกันไม่ให้หุ่นยนต์เข้าใกล้เกินไป
  - เกณฑ์ในการปรับเพื่อให้หุ่นยนต์อยู่ในทิศทางทั้ง4 (0, 90, -90, 180 องศา)
- Error:
  - ระยะทางที่หุ่นยนต์สามารถเคลื่อนที่ไปได้ภายใต้ข้อจำกัด
  - ค่าองศาที่ต้องปรับ(yaw)
- Controller:
  - Logic การเคลื่อนที่ของหุ่นยนต์
- Actuating:
  - Function “ep_chassis.drive_wheels()” สำหรับควบคุมความเร็วของล้อทั้งสี่
  - Function “move()”สำหรับปรับทิศทางแกนz ของหุ่นยนต์(yaw)
- Plant:
  - การเคลื่อนที่ไปข้างหน้า, เลี้ยวซ้ายหรือขวา, ถอยหลัง, สไลด์ไปทางซ้ายหรือขวา
  - ปรับทิศทางของหุ่นยนต์(yaw) เพื่อเคลื่อนที่ไปในทิศทางที่เหมาะสม(ไม่เอียง)
- Feedback:
  - ข้อมูลที่หุ่นยนต์วัดได้จากsensorต่าง ๆ (TOF, Sharp, sub_position, sub_attitude)
    - ระยะห่างจากกำแพงด้านหน้า ซ้ายและขวา
    - ทิศทางของหุ่นยนต์(yaw)

- Output:
    - ตำแหน่งที่หุ่นยนต์เคลื่อนที่ไป

ภาคผนวก

Code

```
import time

import matplotlib.pyplot as plt

from robomaster import robot



# กำหนดlist เพื่อเก็บข้อมูล และกำหนดตัวแปรค่าเริ่มต้น

position_data = []

tof_data = []

left_data = []

right_data = []

current_x = 0.0

current_y = 0.0

current_left = 0.0

current_right = 0.0

count = 0

yaw = None

status = True









# Function Callback สำหรับจัดการข้อมูลตำแหน่งจากเซ็นเซอร์
```

```
def sub_position_handler(position_info):
    x, y, z = position_info
    position_data.append((round(x,2), (round(y,2)), (round(z,2))))
    print("chassis position: x:{:.2f}, y:{:.2f}, z:{:.2f}".format(x, y, z))

# Function Callback สำหรับจัดการข้อมูลท่าทาง (yaw, pitch, roll) ของหุ่นยนต์
def sub_attitude_info_handler(attitude_info):
    global yaw
    yaw, pitch, roll = attitude_info
    print("chassis attitude: yaw:{0}, pitch:{1}, roll:{2} ".format(yaw, pitch, roll))

# Function Callback สำหรับจัดการข้อมูลจากเซ็นเซอร์ TOF
def sub_tof_handler(tof_info):
    tof_data.append(tof_info[0])
    print(f'TOF: {tof_info[0]}')

# Function กรองข้อมูลจาก ADC เพื่อให้ค่าเสถียรมากขึ้น
def filter_adc_value(ad_data):
    y_filtered = []
    alpha = 0.1  # ค่าอัลฟาสำหรับการกรอง
    y_prev = 0  # กำหนดค่าเริ่มต้นของ y_prev


    for adc_value in ad_data:
```

```
        y_current = alpha * y_prev + (1 - alpha) * adc_value
        y_filtered.append(y_current)
        y_prev = y_current

    return y_filtered

# Function Callback สำหรับจัดการข้อมูลจากเซ็นเซอร์ Sharp
def sub_data_handler(sub_info):
    global current_left, current_right
    io_data, ad_data = sub_info

    y_filtered = filter_adc_value(ad_data)

    distances = []
    for adc_filtered in  y_filtered:
        voltage = adc_filtered * 3.3 / 1023

        # การคำนวณระยะทางจากแรงดันไฟฟ้า
        if 2.2 <= voltage < 3.2:
            distance = (voltage - 4.30764) / -0.3846
        elif 1.4 <= voltage < 2.2:
            distance = (voltage - 3.2) / -0.2
        elif 0.8 <= voltage < 1.4:
            distance = (voltage - 1.87) / -0.067
```

```
    elif 0.4 <= voltage < 0.8:
        distance = (voltage - 1.344) / -0.034
    else:
        if voltage >= 3.2:
            distance = (voltage - 4.30764) / -0.3846
        elif voltage < 0.4:
            distance = (voltage - 1.344) / -0.034

    distances.append(distance)

# การคำนวณค่าเฉลี่ยของเซ็นเซอร์Sharpซ้ายและขวา
sharp_left = sum(distances[0:2]) / 2
sharp_right = sum(distances[2:4]) / 2
left_data.append(sharp_left)
right_data.append(sharp_right)

# ปรับค่า sharp_left และ sharp_right ตามเกณฑ์ที่กำหนด
if sharp_left >= 13:
    sharp_left += 2

if sharp_left >= 20:
    sharp_left = 50
if sharp_right >= 26:
    sharp_right = 50
```

```
    current_right = sharp_right
    current_left = sharp_left

    print(f"PORT1 left: {sharp_left}, PORT2 right: {sharp_right}")
    return distances

# Function เคลื่อนที่ไปข้างหน้าจนกว่าจะเจอกำแพงด้านหน้า หรือเจอทางทางขวาที่ไปได้
def move_forword(ep_chassis, threshold_distance, overall_start_time, time_data, list_current_x):
    global current_left, current_right, count, status

    while True:
        # แสดงผลเส้นทางการเคลื่อนที่ของหุ่นยนต์แบบเรียลไทม์
        if position_data:
            x_vals = [pos[0] for pos in position_data]
            y_vals = [pos[1] for pos in position_data]
            ax.clear()
            ax.plot(y_vals, x_vals ,'*-', label="Robot Path")
            ax.set_xlabel('Y Position (cm)')
            ax.set_ylabel('X Position (cm)')
            ax.set_title('Real-time Robot Path')
            ax.grid(True)
            plt.draw()
            plt.pause(0.01)
```

```
# หยุดหุ่นยนต์หาก TOF น้อยกว่าค่า threshold (เจอกำแพงด้านหน้า)
if tof_data and tof_data[-1] < threshold_distance:
    ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)
    time.sleep(0.05)
    count = 0
    break

# หยุดหุ่นยนต์หากค่า current_right มากกว่า 49 (ทางขวาไปได้)
elif status and current_right >= 49:
    ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)
    time.sleep(0.05)
    break

# current_right หรือ current_left ต่ำกว่า 10 ให้ขยับรถเข้ากลาง เพื่อไม่ให้ชนกำแพง
elif current_right <= 10 :
    ep_chassis.drive_wheels(w1=15, w2=-15, w3=15, w4=-15)

elif current_left <= 10:
    ep_chassis.drive_wheels(w1=-15, w2=15, w3=-15, w4=15)

# เคลื่อนที่ไปด้านหน้า
else:
    ep_chassis.drive_wheels(w1=50, w2=50, w3=50, w4=50)
```

```
        # ปรับค่า count และ status
        if count == 8:
            status = True
            count = 0

        if count >= 1:
            count += 1

        list_current_x.append((current_x, current_y))
        time_data.append(time.time() - overall_start_time)

        time.sleep(0.1)

    ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)
    time.sleep(0.1)

# Functionหมุนหุ่นยนต์ 180 องศา
def rotate_180_degrees(ep_chassis):
    ep_chassis.move(x=0, y=0, z=180, z_speed=80).wait_for_completed()
    time.sleep(0.1)

# Functionหมุนหุ่นยนต์ไปทางซ้าย 90 องศา
def rotate_left(ep_chassis):
    ep_chassis.move(x=0, y=0, z=90, z_speed=80).wait_for_completed()
    time.sleep(0.1)
```

```
# Functionหมุนหุ่นยนต์ไปทางขวา 90 องศา
def rotate_right(ep_chassis):
    ep_chassis.move(x=0, y=0, z=-90, z_speed=80).wait_for_completed()
    time.sleep(0.1)

# Function ปรับมุม yaw ของหุ่นยนต์ให้ตรงตามที่กำหนด
def adjust_angle(yaw):
    target_yaw = 0
    correction = yaw

    if -135 < yaw <= -45:
        target_yaw = -90
        ep_chassis.move(x=0, y=0, z=correction-target_yaw, z_speed=20).wait_for_completed()
    elif 45 < yaw < 135:
        target_yaw = 90
        ep_chassis.move(x=0, y=0, z=correction-target_yaw, z_speed=20).wait_for_completed()
    elif -45 < yaw <= 45:
        target_yaw = 0
        ep_chassis.move(x=0, y=0, z=correction, z_speed=20).wait_for_completed()
    elif -180 <= yaw < -135 :
        target_yaw = -180
        ep_chassis.move(x=0, y=0, z=correction-target_yaw, z_speed=20).wait_for_completed()
    elif 135 < yaw <= 180:
```

```
        target_yaw = 180
        ep_chassis.move(x=0, y=0, z=correction-target_yaw, z_speed=20).wait_for_completed()



if __name__ == '__main__':
    # เริ่มต้นการเชื่อมต่อกับหุ่นยนต์
    ep_robot = robot.Robot()
    ep_robot.initialize(conn_type="ap")
    ep_chassis = ep_robot.chassis
    ep_sensor = ep_robot.sensor
    ep_sensor_adaptor = ep_robot.sensor_adaptor
    ep_gimbal = ep_robot.gimbal

    # สมัครสมาชิกเพื่อรับข้อมูลจากเซ็นเซอร์ต่าง ๆ
    ep_chassis.sub_position(freq=10, callback=sub_position_handler)
    ep_chassis.sub_attitude(freq=10, callback=sub_attitude_info_handler)
    ep_sensor.sub_distance(freq=10, callback=sub_tof_handler)
    ep_sensor_adaptor.sub_adapter(freq=10, callback=sub_data_handler)
    time.sleep(0.1)

    time_data, list_current_x = [], []
    overall_start_time = time.time()

    # รีเซ็ตตำแหน่ง gimbal ให้อยู่ในตำแหน่งกลาง
```

```
ep_gimbal.recenter().wait_for_completed()
time.sleep(0.05)


# เริ่มplotเส้นทาง
plt.ion()
fig, ax = plt.subplots()


while True:
    # เคลื่อนที่ไปข้างหน้าจนกว่า TOF จะน้อยกว่า 300 หรือเจอทางทางขวาที่ไปได้
    move_forword(ep_chassis, 300, overall_start_time, time_data, list_current_x)
    ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)
    time.sleep(0.1)


    while True:
        # เคลื่อนที่ถอยหลังหาก TOF น้อยกว่า 160
        if tof_data[-1] < 160:
            ep_chassis.drive_wheels(w1=-20, w2=-20, w3=-20, w4=-20)


        # เคลื่อนที่ไปข้างหน้าหาก TOF อยู่ในช่วง 310-410
        elif tof_data[-1] > 310 and tof_data[-1] < 410:
            ep_chassis.drive_wheels(w1=20, w2=20, w3=20, w4=20)


        # เคลื่อนที่ไปทางซ้าย หาก current_right น้อยกว่า 10
        elif current_right <= 10:
```

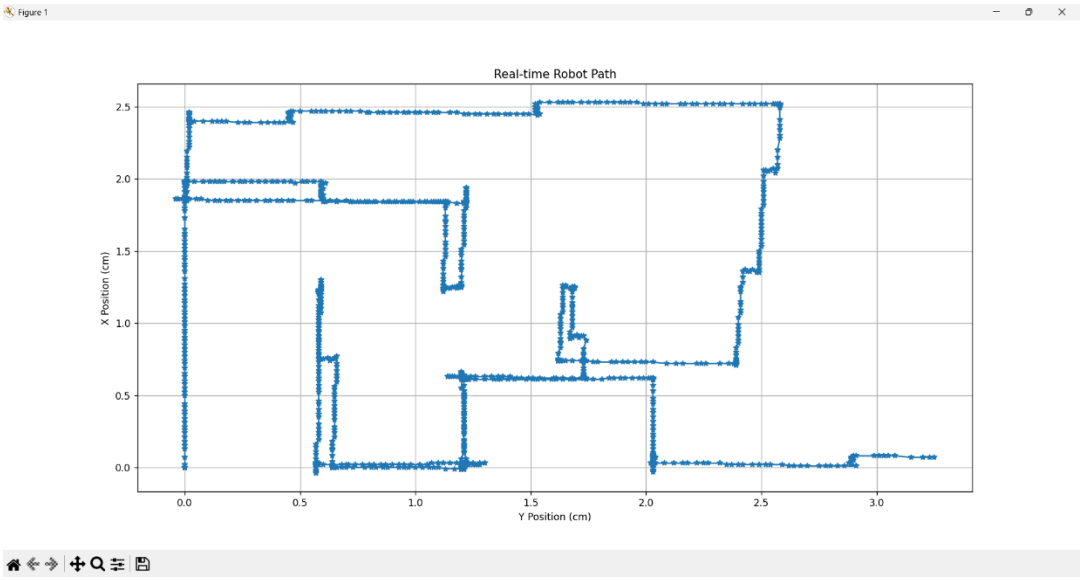
```
            ep_chassis.drive_wheels(w1=15, w2=-15, w3=15, w4=-15)


        # เคลื่อนที่ไปทางขวา หาก current_left น้อยกว่า 10
        elif current_left <= 10:
            ep_chassis.drive_wheels(w1=-15, w2=15, w3=-15, w4=15)


        else:
            # หยุดนิ่ง
            ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)
            break
    time.sleep(0.1)


    # หมุนหุ่นยนต์ไปทางขวา หากทางขวาไปได้
    if current_right >= 49:
            status = False
            count +=1
            rotate_right(ep_chassis)


    else:
        # หมุนหุ่นยนต์ไปทางซ้าย หากทางซ้ายไปได้
        if current_left >= 49:
            status = False
            count +=1
            rotate_left(ep_chassis)
```

```
        else:
            # หมุนหุ่นยนย์กลับหลัง เพราะเจอทางตัน
            rotate_180_degrees(ep_chassis)

    # รีเซ็ตตำแหน่ง gimbal
    ep_gimbal.recenter(yaw_speed=200).wait_for_completed()
    time.sleep(0.1)

    # ปรับมุม yaw ให้ตรง
    adjust_angle(yaw)
    time.sleep(0.2)

    # หยุดการทำงานหาก TOF มากกว่า 6000 และ current_left, current_right มากกว่า 49
    if tof_data and tof_data[-1] >= 6000 and current_left >= 49 and current_left >= 49:
        break

# ยกเลิกการสมัครสมาชิกเซ็นเซอร์และปิดการเชื่อมต่อหุ่นยนต์
ep_chassis.unsub_position()
ep_chassis.unsub_attitude()
ep_sensor.unsub_distance()
ep_sensor_adaptor.unsub_adapter()
ep_robot.close()
```

Video ตอนฝึกซ้อม

https://drive.google.com/file/d/1b7yzYoDv4Ihkk4-uzI9IVcRIqI_haUkK/view?usp=sharing

เส้นทางการเคลื่อนที่ขณะซ้อม จากสนามที่ใช้สอบ



รูปที่ 17 เส้นทางการเคลื่อนที่ของหุ่นยนต์ในเขาวงกต(สนามสอบ)

ปัญหาที่พบเจอ

ปัญหาที่พบเจอ	แนวทางการแก้ไขปัญหา
การเชื่อมต่อกับ Robomaster มักจะหลุดบ่อยก่อนที่จะสามารถทำงานได้	ตอนนี้ยังไม่ทราบสาเหตุของการหลุดการเชื่อมต่อ
เซ็นเซอร์ Sharp บางครั้งอาจตรวจจับทางที่สามารถไปได้ทางขวาได้เร็ว บางครั้งอาจช้าหรือไม่เสถียร ส่งผลให้เมื่อเลี้ยวอาจชนกำแพงได้	เมื่อเลี้ยว ควรเช็คและขยับให้อยู่ในพื้นที่ที่เหมาะสมก่อน เพื่อให้ไม่ชนเมื่อเคลื่อนที่ไปข้างหน้า
ข้อผิดพลาด TOF ทำให้ไม่สามารถรับค่าระยะได้ในขณะนั้น ซึ่งอาจส่งผลให้ชนกับกำแพงด้านหน้า	ใส่เงื่อนไขในโปรแกรมหากTOF ไม่รับค่าในขณะหนึ่งให้หุ่นยนต์หยุดการเคลื่อนที่ แล้วเคลื่อนที่ต่อเมื่อรับค่าได้แล้ว
เซ็นเซอร์ Sharp ตรวจพบกำแพงแต่ในความเป็นจริงในจุดนั้นไม่มีกำแพง	ควรเปิดไฟให้สว่างระหว่างฝึกซ้อม ฝึกซ้อมในช่วงกลางวัน(สภาพแวดล้อมมีผล)