ระบบควบคุมคานสมดุลแบบไร้สาย

WIRELESS BALANCE BEAM CONTROL SYSTEM

พรพิชชา ปิ่นทอง

วิทวัส แซ่โค้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2564

WIRELESS BALANCE BEAM CONTROL SYSTEM

PORNPICHA PINTHONG
WITTHAWAT SAEKHOW

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN CONTROL ENGINEERING
SCHOOL OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2021

ปริญญานิพนธ์ปีการศึกษา 2564

ภาควิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมคานสมดุลแบบไร้สาย

Wireless Balance Beam Control System

ผู้จัดทำ นางสาวพรพิชชา ปิ่นทอง 61010693

นายวิทวัส แซ่โค้ว 61010967

.....อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร.วรรณดี เพชรมณีล้ำค่า)

ระบบควบคุมคานสมดุลแบบไร้สาย

โดย
นางสาวพรพิชชา ปิ่นทอง 61010963
นายวิทวัส แซ่โค้ว 61010967

อาจารย์ที่ปรึกษา
รองศาสตราจารย์ ดร.วรรณดี เพชรมณีล้ำค่า

ปีการศึกษา 2564

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการออกแบบระบบควบคุมคานสมดุลแบบไร้สาย ให้มีคุณสมบัติที่ตอบสนองต่อความต้องการของผู้ใช้งาน โดยนำเทคโนโลยี Internet of Things (IoT) มาประยุกต์ใช้เพื่อพัฒนาต่อยอดระบบควบคุมคานสมดุลแบบไร้สายให้เข้ากับเหตุการณ์ที่เกิดขึ้นในปัจจุบัน เนื่องจากสถานการณ์การแพร่ระบาดของเชื้อไวรัส โควิด-19 ส่งผลกระทบทำให้การปฏิบัติงานในห้องปฏิบัติการเป็นไปได้อย่างยากลำบาก โดยระบบควบคุมคานสมดุลแบบไร้สายที่พัฒนาขึ้น ผู้ใช้งานสามารถรับส่งข้อมูลเพื่อสั่งงานและควบคุมระบบคานสมดุลแบบไร้สาย โดยเลือกฟังก์ชันการทำงานได้ รวมถึงการแสดงผลผ่านระบบแอปพลิเคชันบนสมาร์ทโฟนหรือแท็บเล็ตได้

ขั้นตอนในการดำเนินงานเริ่มจากศึกษาการทำงานของระบบควบคุมคานสมดุลแบบไร้สาย ศึกษาและออกแบบวงจรควบคุมการทำงาน ประกอบด้วยบอร์ดควบคุม ESP32 วงจรควบคุมการทำงานของมอเตอร์ และมอเตอร์กระแสตรง ทำการเขียนโปรแกรมด้วย Arduino เพื่อควบคุมการทำงาน จากนั้นเขียนโปรแกรมด้วยแอปพลิเคชัน NETPIE เพื่อทำการเชื่อมต่อกับระบบเครือข่ายแบบไร้สายและสามารถใช้งานบนสมาร์ทโฟนได้ สุดท้ายทดสอบระบบการทำงานทั้งหมดและปรับแก้ไขให้ทำงานตามที่ออกแบบไว้

WIRELESS BALANCE BEAM CONTROL SYSTEM

By

Miss Pornpicha Pinthong 61010963

Mr. Witthawat Saekhow 61010967

Advisor

Assoc.Prof.Dr. Wandee Petchmaneelumka

Academic Year 2021

ABSTRACT

This thesis presents a wireless balance beam control system design that has functions for user requirements. Internet of Things (IoT) technology is implemented in wireless balance beam control systems. Due to the COVID-19 epidemic situation, it is difficult to work in the laboratory. The operation of the developed system can send and receive data to control the balance beam control system wirelessly. The operating function including display can be selected through the application system on smartphones or tablets.

The procedure starts with studying the operation of a wireless balance beam control system. Then, the circuit for control is studied and designed including an Arduino ESP32 control board, motor control circuit, and a DC motor. Arduino IDE is used for programming to control the system. After that NETPIE application is used to connect to a wireless network via smartphone. Finally, the entire system is tested and modified to work as designed.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ประสบผลสำเร็จลุล่วงไปได้ด้วยดี ด้วยความกรุณาช่วยเหลือ แนะนำให้คำปรึกษา และตรวจสอบแก้ไขข้อบกพร่องต่างๆ ด้วยความเอาใจใส่อย่างดียิ่ง จากอาจารย์ที่ปรึกษา รองศาสตราจารย์ ดร.วรรณดี เพชรมณีล้ำค่า ภาควิชาวิศวกรรมการวัดและควบคุม ตลอดระยะเวลาในการศึกษาจนทำให้ปริญญานิพนธ์ฉบับนี้สามารถดำเนินงานให้เป็นไปตามวัตถุประสงค์ได้อย่างลุล่วงสมบูรณ์

ขอขอบพระคุณ ศาสตราจารย์ ดร.วันชัย ริ้วรุจา ภาควิชาวิศวกรรมการวัดและควบคุม ที่กรุณาเป็นผู้เชี่ยวชาญด้านวงจรอิเล็กทรอนิกส์และระบบควบคุม ให้คำแนะนำ และให้คำปรึกษา อันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.อภินัย ฤกษ์รัตน์ ภาควิชาวิศวกรรมการวัดและควบคุม ที่กรุณาเป็นผู้เชี่ยวชาญด้านวงจรอิเล็กทรอนิกส์และเครื่องมือวัด ให้คำปรึกษาและตรวจสอบการทำงานของระบบ จนกระทั่งสามารถดำเนินงานสำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณ อาจารย์ท่านอื่นๆ ในหลักสูตรวิศวกรรมระบบควบคุม ภาควิชาวิศวกรรมการวัดและควบคุม ในความอนุเคราะห์ให้ช่วยเหลือในการทำโครงงานนี้

ขอขอบคุณเพื่อนๆ ทุกคนที่ช่วยเหลือและให้คำแนะนำในการทำปริญญานิพนธ์ฉบับนี้ จึงทำให้สำเร็จลุล่วงไปได้ด้วยดี

สุดท้ายนี้ขอกราบพระคุณบิดา มารดา ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้ และถ่ายทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้า

คณะผู้จัดทำ
พรพิชชา ปิ่นทอง
วิทวัส แซ่โค้ว

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญรูป VII

สารบัญตาราง X

บทที่ 1 บทนำ **1**

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์ 1

1.2 วัตถุประสงค์ของปริญญานิพนธ์ 1

1.3 ขอบเขตของโครงงาน 1

1.4 ประโยชน์ที่คาดว่าจะได้รับ 2

1.5 รายละเอียดของปริญญานิพนธ์ 2

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง **3**

2.1 กฎการเคลื่อนที่ของนิวตัน 3

2.2 มวลและน้ำหนัก 4

2.3 แรงโน้มถ่วงของโลก 5

2.4 แรงยก 5

2.5 สภาพสมดุล 7

2.6 โมเมนต์ของแรง 7

2.7 พลังงานลมและอากาศพลศาสตร์ 8

2.8 ระบบควบคุม 8

2.8.1 องค์ประกอบพื้นฐานของระบบควบคุม 9

2.8.2 พื้นฐานของระบบควบคุมแบบวงเปิดและวงปิด 9

2.8.3 ระบบควบคุม PID 10

2.8.4 คุณสมบัติของระบบควบคุมแบบ PID 14

2.8.5 การปรับจูนค่าคงที่ของตัวควบคุม 15

สารบัญ (ต่อ)

	หน้า
บทที่ 3 วิธีการดำเนินงาน	**18**
3.1 ขั้นตอนการดำเนินงาน	18
3.2 อุปกรณ์ที่เกี่ยวข้อง	18
3.2.1 บอร์ด Arduino ESP32	18
3.2.2 ชุดระบบควบคุมคานสมดุล	19
3.2.3 วงจรคงค่าแรงดัน (Voltage regulator)	19
3.2.4 Optocoupler หรือ Opto-Isolator	20
3.2.5 IC ULN2003	20
3.2.6 AC/DC Adapter 9V 3A	21
3.2.7 Motor	21
3.3 ซอฟต์แวร์ที่เกี่ยวข้อง	22
3.3.1 Arduino IDE	22
3.3.2 Visual Studio	23
3.3.3 NETPIE	23
3.3.4 Fritzing	24
3.4 การออกแบบและการวางแผนการทำงาน	24
3.4.1 การออกแบบและการวางแผนทางด้านฮาร์ดแวร์	24
3.4.2 การออกแบบและการวางแผนทางด้านซอฟต์แวร์	24
3.5 วิธีการดำเนินงาน	25
3.5.1 การทดสอบการทำงานของเซนเซอร์	25
3.5.2 การทดสอบการทำงานของใบพัดมอเตอร์	27
3.5.3 การเขียนโปรแกรม Arduino	27
บทที่ 4 ผลการดำเนินงาน	**36**
4.1 แผนผังการทำงานของระบบควบคุมคานสมดุลแบบไร้สาย	36
4.2 ชุดอุปกรณ์ของระบบควบคุมคานสมดุลแบบไร้สาย	36
4.3 ผลการออกแบบหน้าแสดงผลผ่าน NETPIE	38
4.4 การทดสอบการทำงาน	39
4.4.1 ทดสอบการรับ-ส่งข้อมูล โหมดการปรับด้วยมือ (Manual mode)	39
4.4.2 ทดสอบการรับ-ส่งข้อมูล โหมดอัตโนมัติ (Automation mode)	40

สารบัญ (ต่อ)

หน้า

4.5 ผลการทดสอบการควบคุมระบบ 41

4.5.1 ผลการทดสอบการควบคุมระบบด้วยตัวควบคุมแบบ P Controller 41

4.5.2 ผลการทดสอบการควบคุมระบบด้วยตัวควบคุมแบบ PI Controller 43

4.5.3 ผลการทดสอบการควบคุมระบบด้วยตัวควบคุมแบบ PID Controller 45

บทที่ 5 สรุปผลการดำเนินงานและข้อเสนอแนะ 47

5.1 สรุปผลการดำเนินงาน 47

5.2 ปัญหาและอุปสรรคในการดำเนินงาน 47

5.3 แนวทางการแก้ไขปัญหา 48

5.4 การพัฒนาในอนาคต 48

เอกสารอ้างอิง 49

ภาคผนวก 50

ภาคผนวก ก วงจรในชุดทดลองควบคุมคานสมดุล 51

ภาคผนวก ข บอร์ด ARDUINO ESP32 52

ภาคผนวก ค โปรแกรมควบคุมการทำงาน การส่ง-ข้อมูล และการแสดงผลของระบบ 54

สารบัญรูป

รูปที่	หน้า
2.1 แรงกิริยาและแรงปฏิกิริยา	4
2.2 มวลและน้ำหนัก	5
2.3 แรงกระทำที่เกิดขึ้นในภาคตัดขวางของปีก (1)	6
2.4 แรงกระทำที่เกิดขึ้นในภาคตัดขวางของปีก (2)	6
2.5 บล็อกไดอะแกรมของระบบควบคุมแบบวงเปิด	9
2.6 บล็อกไดอะแกรมของระบบควบคุมแบบวงปิด	10
2.7 บล็อกไดอะแกรมของการควบคุมแบบPID	11
2.8 แผนภูมิ Process Value ต่อเวลา กำหนดค่า K_p เป็น 0.5, 1 และ 2	12
2.9 แผนภูมิ Process Value ต่อเวลา กำหนดค่า K_i เป็น 0.5, 1 และ 2	13
2.10 แผนภูมิ Process Value ต่อเวลา กำหนดค่า K_d เป็น 0.5, 1 และ 2	13
2.11 กราฟผลการตอบสนองของ PV (Process Value)	16
2.12 กราฟของ Quarter-amplitude	16
2.13 การปรับจูนระบบโดยใช้วิธี Ziegler-Nichols Close-Loop tuning	17
3.1 บอร์ด Arduino ESP32	19
3.2 LM7805	20
3.3 PC817	20
3.4 ULN2003	21
3.5 AC/DC Adapter 9V 3A	21
3.6 Motor ที่ใช้งาน	22
3.7 สัญลักษณ์โปรแกรมArduino IDE	22
3.8 สัญลักษณ์โปรแกรม Visual Studio	23
3.9 สัญลักษณ์โปรแกรม NETPIE	23
3.10 สัญลักษณ์โปรแกรม Fritzing	24
3.11 การต่อโมดูลเข้ากับบอร์ด ESP 32	25
3.12 กราฟความสัมพันธ์ระหว่างแรงดันเอาต์พุตและตำแหน่งของคาน (Feedback)	26
3.13 กราฟความสัมพันธ์ระหว่างค่า Setpoint และ ค่าแรงดันเอาต์พุต	26
3.14 หน้าต่างโปรแกรม Arduino (1)	27

สารบัญรูป (ต่อ)

รูปที่	หน้า
3.15 หน้าต่างโปรแกรม Arduino (2)	28
3.16 หน้าต่างโปรแกรม Arduino (3)	28
3.17 หน้าต่างโปรแกรม Arduino (4)	29
3.18 หน้าต่างโปรแกรม Arduino (5)	29
3.19 หน้าต่างโปรแกรม Arduino (6)	30
3.20 หน้าต่างโปรแกรม Arduino (7)	30
3.21 หน้าต่างโปรแกรม Arduino (8)	31
3.22 หน้าต่างโปรแกรม Arduino (9)	31
3.23 หน้าต่างโปรแกรม Arduino (10)	32
3.24 หน้าต่างโปรแกรม Arduino (11)	33
3.25 หน้าต่างโปรแกรม Arduino (12)	33
3.26 หน้าต่างโปรแกรม Arduino (13)	34
3.27 หน้าต่างโปรแกรม Arduino (14)	34
3.28 หน้าต่างโปรแกรม Arduino (15)	35
3.29 ฐานเก็บข้อมูลบน NETPIE	35
4.1 แผนผังการทำงานของระบบควบคุมคานสมดุลแบบไร้สาย	36
4.2 ชุดอุปกรณ์ระบบควบคุมคานสมดุลแบบไร้สาย	37
4.3 ระบบควบคุมคานสมดุลแบบไร้สายที่สมบูรณ์	37
4.4 วงจรอิเล็กทรอนิกส์ภายในชุดระบบควบคุมคานสมดุลแบบไร้สาย	37
4.5 วงจรอิเล็กทรอนิกส์ภายในที่เชื่อมต่อกับ Arduino ESP32	38
4.6 หน้าแสดงผลบน NETPIE	39
4.7 การควบคุมระบบผ่าน NETPIE โหมดการปรับมือ	40
4.8 การควบคุมระบบผ่าน NETPIE โหมดอัตโนมัติ	40
4.9 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ P controller ที่ Kp = 6 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 50 %	41

สารบัญรูป (ต่อ)

รูปที่	หน้า
4.10 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ P controller ที่ Kp = 6 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 74 %	42
4.11 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ P controller ที่ Kp = 15 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 50 %	42
4.12 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ P controller ที่ Kp = 15 โหลดอยู่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 74 %	43
4.13 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ PI controller ที่ Kp = 40 Ki = 2 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 50 %	44
4.14 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ PI controller ที่ Kp = 40 Ki = 2 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 74 %	44
4.15 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ PID controller ที่ Kp = 10 Ki = 0.12 Kd = 40 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 50 %	45
4.16 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ PID controller ที่ Kp = 10 Ki = 0.12 Kd = 40 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 74 %	46

สารบัญตาราง

ตารางที่	หน้า
2.1 ผลของการเพิ่มค่าตัวแปรอิสระ	15
2.2 พารามิเตอร์การปรับ Ziegler-Nichols Closed loop	17

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ปัจจุบันมีงานวิจัยมากมายที่ให้ความสนใจเกี่ยวกับการทำงานที่ผ่านระบบเครือข่ายไร้สาย Wi-Fi เพื่ออำนวยความสะดวกให้กับผู้ปฏิบัติงานในห้องควบคุม อีกทั้งในปัจจุบันยังมีสถานการณ์ของโรคโควิด 19 (COVID-19) ที่ยังแพร่ระบาดอย่างต่อเนื่องจึงส่งผลให้การเข้าไปปฏิบัติงานในห้องปฏิบัติการนั้นเป็นไปได้ยาก ดังนั้นผู้จัดทำจึงมองหาวิธีการที่จะลดต้นทุนของห้องปฏิบัติการและเข้าถึงห้องปฏิบัติการได้สะดวกโดยการใช้ประโยชน์จากอินเทอร์เน็ตมาประยุกต์ใช้กับระบบควบคุม เนื่องจากในทุกวันนี้อินเทอร์เน็ตมีบทบาทสำคัญในการดำรงชีวิตของผู้คนมากขึ้น คณะผู้จัดทำจึงเล็งเห็นว่าสามารถนำอินเทอร์เน็ตมาประยุกต์ใช้เพื่อช่วยลดค่าใช้จ่ายในการใช้ห้องปฏิบัติการลงและยังสามารถทำให้ผู้ที่ใช้งานเข้าถึงชุดปฏิบัติการการควบคุมระบบได้สะดวก โดยไม่จำเป็นต้องเข้าไปปฏิบัติการในห้องปฏิบัติการและยังคงสามารถศึกษาเกี่ยวกับระบบได้โดยการใช้เทคโนโลยี Internet of Things (IoT) มาใช้เพื่อเป็นการพัฒนาและต่อยอดเกี่ยวกับระบบควบคุมคานสมดุลแบบไร้สายนำไปประยุกต์ใช้กับห้องปฏิบัติการออนไลน์เพื่อการศึกษาให้เกิดประโยชน์และอำนวยความสะดวกมากที่สุด

ระบบควบคุมคานสมดุลแบบไร้สายเป็นกระบวนการที่ตัวคานถูกยกขึ้นโดยไม่พึ่งกลไกทางกล ระบบควบคุมคานสมดุลแบบไร้สายที่นำเสนอในโครงงานนี้จึงมีการออกที่เรียบง่ายไม่ซับซ้อน เนื่องจากใช้ต้นทุนที่ต่ำ และง่ายต่อการเรียนรู้ รวมไปถึงการนำไปประยุกต์ใช้ ซึ่งชุดอุปกรณ์มีความสามารถที่เพียงพอต่อการทำงานเป็นห้องปฏิบัติการระยะไกล เพียงเชื่อมต่อบอร์ดกับเครือข่ายไร้สาย(Wi-Fi) รวมถึงยังสามารถแสดงผลและควบคุมผ่านแอปพลิเคชันบนสมาร์ทโฟนหรือผ่านเว็บบราวเซอร์เพื่อความสะดวกของผู้ใช้งาน

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ออกแบบและสร้างระบบควบคุมคานสมดุลแบบไร้สาย
2. ควบคุมระบบและแสดงผลผ่านแอปพลิเคชันบนโทรศัพท์มือถือและผ่านเว็บบราวเซอร์

1.3 ขอบเขตของโครงงาน

1. ประยุกต์ใช้หลักการทำงานของอุปกรณ์อิเล็กทรอนิกส์ที่เกี่ยวข้อง

2. ประยุกต์ใช้ Internet of thing (IoT) ของการเชื่อมต่ออุปกรณ์ด้วย IP Network และโปรแกรมที่นำมาใช้ประมวลกับระบบควบคุม

3. ออกแบบและพัฒนาการทำงานของระบบควบคุมคานสมดุลแบบไร้สายโดยสามารถใช้งานฟังก์ชันต่างๆ บนสมาร์ทโฟนและเว็บบราวเซอร์ได้

4. ออกแบบส่วนแสดงผลต่างๆ ที่เกี่ยวกับระบบ

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถสร้างชุดระบบควบคุมคานสมดุลได้

2. สามารถควบคุมระบบโดยการรับ-ส่งค่าตำแหน่งของคาน และแสดงผลบนโทรศัพท์มือถือและเว็บบราวเซอร์ผ่านเครือข่ายไร้สาย Wi-Fi

3. สามารถนำระบบควบคุมคานสมดุลไร้สายไปประยุกต์ใช้งานกับระบบควบคุมอื่นๆ ที่เชื่อมต่อเครือข่ายไร้สาย Wi-Fi เพื่อใช้งานควบคุมระยะไกลหรือออนไลน์ได้

1.5 รายละเอียดของปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย 5 บท และ 3 ภาคผนวก ซึ่งมีรายละเอียดดังต่อไปนี้

บทที่ 1 บทนำ เป็นการกล่าวที่มาและความสำคัญของปริญญานิพนธ์ วัตถุประสงค์ของการทำปริญญานิพนธ์ ขอบเขตของการทำงาน ประโยชน์ที่คาดว่าจะได้รับและรายละเอียดของปริญญานิพนธ์

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง เป็นการกล่าวถึงทฤษฎีการเคลื่อนของนิวตัน ความหมายของมวลและน้ำหนัก ค่าของแรงโน้มถ่วงโลก แรงยก สภาพสมดุล โมเมนต์ของแรง พลังงานลมและอากาศพลศาสตร์ และระบบควบคุม

บทที่ 3 ขั้นตอนการดำเนินงาน เป็นการอธิบายขั้นตอนการทำงานโดยละเอียดทั้งในส่วนของฮาร์ดแวร์และซอฟต์แวร์

บทที่ 4 ผลการดำเนินงาน เป็นการแสดงผลการทำงานของระบบควบคุมคานสมดุลแบบไร้สายผ่านเว็บบราวเซอร์ของ NETPIE

บททที่ 5 ผลสรุปและข้อเสนอแนะ เป็นบทสรุปของระบบควบคุมคานสมดุลแบบไร้สายรวมถึงสิ่งที่จะพัฒนาต่อไป

ภาคผนวก ก วงจรในชุดทดลองควบคุมคานสมดุล

ภาคผนวก ข บอร์ด Arduino ESP32

ภาคผนวก ค โปรแกรมควบคุมการทำงาน การส่ง-ข้อมูล และการแสดงผลของระบบ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 กฎการเคลื่อนที่ของนิวตัน

กฎการเคลื่อนที่ของนิวตันเป็นกฎทางกายภาพสามข้อที่เป็นรากฐานของกลศาสตร์ดั้งเดิม ใช้สำหรับการอธิบายความสัมพันธ์ระหว่างวัตถุกับแรงที่กระทำต่อวัตถุนั้น และการเคลื่อนที่เนื่องจากแรงเหล่านั้น โดยถูกค้นพบโดย Isaac Newton

กฎการเคลื่อนที่ข้อที่ 1 ของนิวตัน

กฎข้อที่หนึ่งของนิวตันหรือ กฎของความเฉื่อย กล่าวว่า “วัตถุจะรักษาสภาวะอยู่นิ่ง หรือสภาวะเคลื่อนที่อย่างสม่ำเสมอในแนวเส้นตรง นอกจากมีแรงลัพธ์มากระทำ” กล่าวคือ ถ้าวัตถุนั้นนิ่งอยู่ไม่เคลื่อนไหวก็ยังนิ่งอยู่อย่างนั้น แต่ถ้าวัตถุนั้นกำลังเคลื่อนที่ด้วยความเร็วคงที่ (a = 0) ก็ยังคงเคลื่อนที่ด้วยความเร็วคงที่ต่อไปตราบใดที่ไม่มีแรงภายนอกมากระทำ เขียนเป็นความสัมพันธ์ได้ดังนี้

$$\sum F = 0 \qquad (1)$$

กฎการเคลื่อนที่ข้อที่ 2 ของนิวตัน

กฎข้อที่ 2 ของนิวตันบางที่เรียกว่า กฎความเร่ง กฎข้อนี้กล่าวว่า “ความเร่งของอนุภาคเป็นปฏิภาคโดยตรงกับแรงลัพธ์ที่กระทำต่ออนุภาค โดยมีทิศทางเดียวกัน และเป็นปฏิภาคผกผันกับมวลของอนุภาค”ตามกฎข้อที่ 2 ของนิวตัน เนื่องจากความเร่งเป็นสัดส่วนตรงกับแรง ดังนั้น อัตราส่วนของแรงกับความเร่งจะเป็นค่าคงที่ ซึ่งตรงกับมวล m ของวัตถุ เขียนเป็นความสัมพันธ์ได้ดังนี้

$$\sum F = ma \qquad (2)$$

เมื่อ $\sum F$ คือ ผลรวมของแรงหรือแรงลัพธ์
m คือ มวลของวัตถุ
a คือ ความเร่งของวัตถุ

กฎการเคลื่อนที่ข้อที่ 3 ของนิวตัน

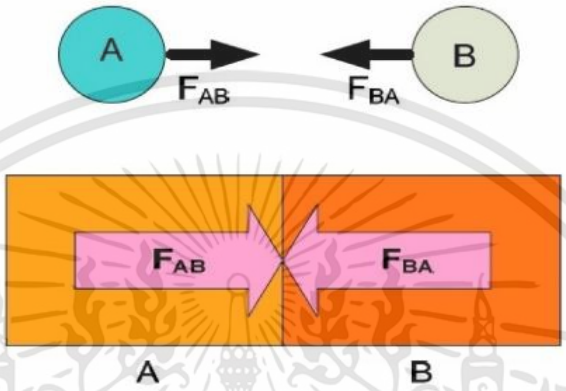
กฎการเคลื่อนที่ข้อที่ 3 ของนิวตันกล่าวว่า “ทุกแรงกิริยาย่อมมีแรงปฏิกิริยาซึ่งมีขนาดเท่ากันแต่มีทิศทางตรงข้ามกันเสมอ กฎข้อนี้เรียกว่า กฎของกิริยาและปฏิกิริยา (Law of action and reaction) โดยแรงกิริยาและแรงปฏิกิริยาหมายถึง แรงกระทำและแรงกระทำตอบ โดยเป็นแรงซึ่ง

กระทำต่อมวลที่ต่างกัน และเกิดดขึ้นพร้อมกันเป็นคู่เสมอ โดยที่มวลอาจไม่สัมผัสกันเขียนเป็นความสัมพันธ์ได้ดังนี้

$$F_{AB} = F_{BA} \quad (3)$$

เมื่อ F_{AB} คือ แรงกิริยาที่วัตถุชิ้นที่ A กระทำต่อวัตถุชิ้นที่ B

F_{BA} คือ แรงกิริยาที่วัตถุชิ้นที่ B กระทำต่อวัตถุชิ้นที่ A



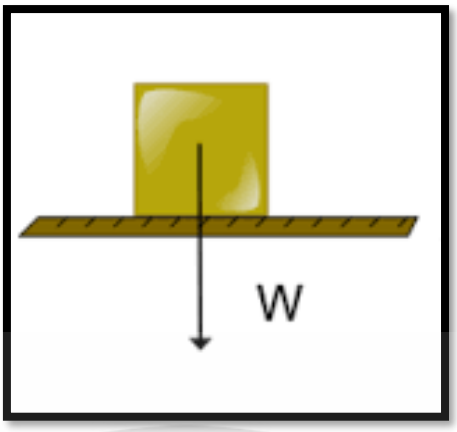
รูปที่ 2.1 แรงกิริยาและแรงปฏิกิริยา [7]

2.2 มวลและน้ำหนัก

มวล (Mass) เป็นสมบัติของก้อนสสารที่บ่งบอกถึงค่าความต้านทานในการเปลี่ยนสภาพการเคลื่อนที่ หรือเป็นปริมาณที่แปรผันตรงกับค่าความต้านทานต่อการเกิดความเร่งเมื่อถูกแรงกระทำ หรือมวล m ของวัตถุ หมายถึง ความเฉื่อยต่อการเคลื่อนที่ มวลมีหน่วยเป็นกิโลกรัม

น้ำหนัก (Weight) หมายถึง แรงที่เกิดจากความเร่งโน้มถ่วงของโลกกระทำต่อวัตถุดังนั้น ถ้าปล่อยให้วัตถุมวล m ตกลงมาอย่างอิสระ แรงสุทธิที่กระทำต่อวัตถุคือ น้ำหนักของมวล m คูณกับความเร่งโน้มถ่วงของโลก g โดยน้ำหนักมีหน่วยเป็นนิวตัน จาก F = ma จะได้

$$w = mg \quad (4)$$



รูปที่ 2.2 มวลและน้ำหนัก [8]

2.3 แรงโน้มถ่วงของโลก

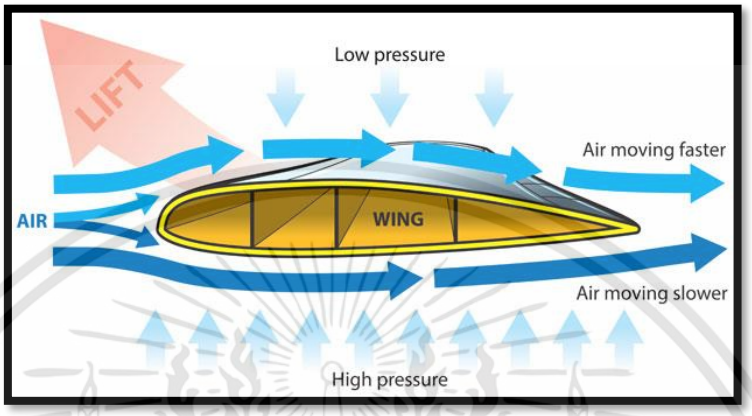
แรงโน้มถ่วงของโลก (Gravity) คือ แรงดึงดูดของโลกหรือแรงของโลกที่กระทำต่อมวลของวัตถุทุกชนิดบนโลกและวัตถุที่อยู่ใกล้ผิวโลก โดยจะดึงดูดวัตถุซึ่งกันและกันเข้าสู่ศูนย์กลางของโลกทำให้วัตถุต่างๆ ตกลงสู่พื้นโลกเสมอและทำให้วัตถุมีน้ำหนัก โดยแรงโน้มถ่วงมีทิศทางเข้าสู่ศูนย์กลางโลก เป็นแรงที่ยึดเหนี่ยววัตถุให้ติดอยู่กับพื้นโลก มิฉะนั้นวัตถุหรือแม้กระทั้งบรรยากาศจะหลุดปลิวไปในอากาศ

วัตถุที่อยู่ในสนามโน้มถ่วงของโลกจะถูกโลกดึงดูด ดังนั้น เมื่อปล่อยวัตถุให้ตกบริเวณใกล้ผิวโลก แรงดึงดูดของโลกจะทำให้วัตถุเคลื่อนที่เร็วขึ้น นั่นคือวัตถุมีความเร่งการตกของวัตถุที่มีมวลต่างกันในสนามโน้มถ่วงวัตถุเคลื่อนที่ด้วยความเร่งคงตัว เรียกว่า ความเร่งโน้มถ่วง (Gravitational acceleration) มีทิศทางเข้าสู่ศูนย์กลางของโลก ความเร่งโน้มถ่วงที่ผิวโลกมีค่าต่างกันตามตำแหน่งทางภูมิศาสตร์ในการตกของวัตถุ วัตถุจะเคลื่อนที่ลงด้วยความเร่งโน้มถ่วง 9.8 เมตรต่อวินาทียกกำลังสอง ซึ่งหมายความว่าความเร็วของวัตถุจะเพิ่มขึ้นวินาทีละ 9.8 เมตรต่อวินาที แต่ถ้าเกิดโยนวัตถุขึ้นในแนวดิ่ง วัตถุในสนามโน้มถ่วงจะเคลื่อนที่ ขึ้นด้วยความเร่งโน้มถ่วง g โดยมีทิศเข้าสู่ศูนย์กลางโลกทำให้วัตถุซึ่งเคลื่อนที่ขึ้นมีความเร็วลดลงวินาทีละ 9.8 เมตรต่อวินาทีจนกระทั่งความเร็วสุดท้ายเป็นศูนย์ จากนั้นแรงโน้มถ่วงจะดึงวัตถุให้ตกกลับสู่ผิวโลก

2.4 แรงยก

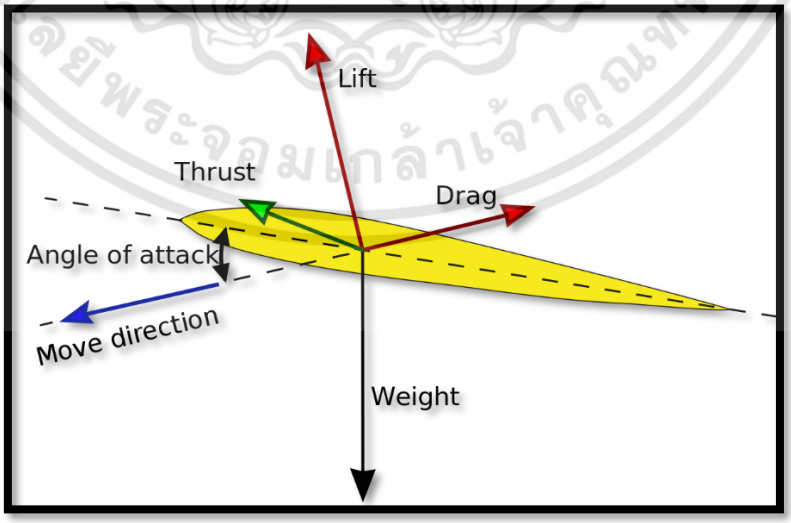
แรงยก คือ แรงทางอากาศพลศาสตร์ (Aerodynamic force) เป็นแรงที่ทำให้เครื่องบินสามารถลอยอยู่บนอากาศได้ โดยแรงยกของเครื่องบินเกิดจากผลของมุมปะทะของปีกที่ติดตั้งกับลำตัวของเครื่องบิน และผลของความแตกต่างระหว่างความดันที่กระทำที่พื้นผิวของปีกเครื่องบิน

ด้านบนและปีกเครื่องบินด้านล่าง ความดันที่กระทำกระจายตัวอยู่บนพื้นผิวดังกล่าวจะสร้างแรงกระทำลงบนพื้นผิวนั้นๆ เมื่อพื้นผิวด้านล่างมีความดันกระทำสูงกว่า จึงมีแรงกระทำเกิดขึ้นสูงกว่าที่พื้นผิวด้านบน โดยแรงยกก็คือผลต่างที่เหลืออยู่ของแรงทั้งสอง ซึ่งสามารถใช้หลักการเรื่องของไหลของแบร์นูลลี (Bernoulli) และ กฎของนิวตัน (Newton law)



รูปที่ 2.3 แรงกระทำที่เกิดขึ้นในภาคตัดขวางของปีก (1) [10]

สาเหตุที่พื้นผิวด้านล่างของปีกและพื้นผิวด้านบนของปีกมีความดันกระทำไม่เท่ากัน เป็นเพราะความเร็วของอากาศที่ไหลผ่านพื้นผิวทั้งสองฝั่งมีค่าไม่เท่ากัน ซึ่งการออกแบบรูปร่าง รูปทรงของปีก รวมทั้งมุมปะทะของการติดตั้งปีกเข้ากับลำตัวของเครื่องบิน เป็นปัจจัยสำคัญในการทำให้เกิดปรากฏการณ์นี้ การที่เครื่องบินจะลอยได้นั้น แรงยกที่มีทิศทางยกลำตัวเครื่องบินขึ้นต้องมีขนาดมากกว่าแรงกระทำทั้งหมดที่มีทิศทางตรงกันข้าม หรือทิศทางลงสู่พื้น ซึ่งแรงกระทำดังกล่าวได้แก่ น้ำหนักของเครื่องบิน แรงกระทำทางอากาศพลศาสตร์ และแรงอื่นๆ



รูปที่ 2.4 แรงกระทำที่เกิดขึ้นในภาคตัดขวางของปีก (2) [10]

2.5 สภาพสมดุล

สภาพสมดุล (Equilibrium) คือ สภาพที่วัตถุสามารถรักษาสภาพการเคลื่อนที่ให้คงเดิม เช่น อยู่นิ่งไม่มีการเลื่อนตำแหน่งและไม่หมุน หรือเคลื่อนที่ในสภาพเดิม คือ จุดศูนย์กลางมวลเลื่อนด้วยอัตราเร็วคงที่และหมุนรอบแกนผ่านจุดศูนย์กลางมวลด้วยอัตราเร็วเชิงมุมคงที่ หรือกล่าวได้อีกนัยหนึ่งว่า สมดุลที่เกิดขึ้นในขณะที่วัตถุอยู่ในสภาพนิ่ง หรือเคลื่อนที่ด้วยความเร็วคงตัว ถ้าแรงลัพธ์ที่กระทำต่อวัตถุมีค่าเป็นศูนย์

ประเภทของเสถียรสภาพของสมดุล

1. สภาพสมดุลเสถียร (Stable equilibrium) คือ สมดุลที่วัตถุวางอยู่ได้ถ้าถูกแรงกระทำให้เอียงเล็กน้อย ก็จะกลับเข้าสู่สภาพเดิมได้ โดยเมื่อวัตถุได้รับแรงกระทำแล้วทำให้จุด Center of gravity สูงกว่าเดิม จะมีแนวโน้มกลับสู่สภาพสมดุล

2. สภาพสมดุลไม่เสถียร (Unstable equilibrium) คือ สมดุลที่วัตถุไม่สามารถวางเหมือนเดิมได้ ถ้าถูกแรงกระทำเพียงเล็กน้อย โดยเมื่อวัตถุได้รับแรงกระทำแล้วทำให้จุด Center of gravity ต่ำกว่าเดิม จะมีแนวโน้มสู่สภาพสมดุลไม่เสถียร

3. สภาพสมดุลสะเทิน (Neutral equilibrium) คือ สมดุลที่วัตถุถูกแรงกระทำเพียงเล็กน้อยแล้ววัตถุจะเปลี่ยนตำแหน่งไป แต่ยังคงสภาวะเดิมได้ โดยเมื่อวัตถุได้รับแรงกระทำแล้วทำให้จุด Center of gravity อยู่ในระดับเดิม จะทำให้วัตถุมีสภาพสมดุลสะเทิน

2.6 โมเมนต์ของแรง

โมเมนต์ของแรง (Moment of force) หมายถึง ผลของแรงที่กระทำต่อวัตถุหมุนไปรอบจุดหมุน ดังนั้น ค่าโมเมนต์ของแรง ก็คือ ผลคูณของแรงนั้นกับระยะตั้งฉากจากแนวแรงถึงจุดหมุน (มีหน่วยเป็น นิวตัน-เมตร)

โมเมนต์ (นิวตัน-เมตร) = แรง (นิวตัน) * ระยะตั้งฉากจากแนวแรงถึงจุดหมุน (5)

โดยโมเมนต์ของแรงแบ่งตามทิศทางการหมุนได้เป็น 2 แบบ คือ โมเมนต์ทวนเข็มนาฬิกา เป็นโมเมนต์ของแรงที่ทำให้วัตถุหมุนทวนเข็มนาฬิกา และ โมเมนต์ตามเข็มนาฬิกา เป็นโมเมนต์ของแรงที่ทำให้วัตถุหมุนตามเข็มนาฬิกาโดยสองแรงมีความสัมพันธ์กันดังนี้

ผลรวมของโมเมนต์ทวนเข็มนาฬิกา = ผลรวมของโมเมนต์ตามเข็มนาฬิกา (6)

$$M_{\text{ทวน}} = M_{\text{ตาม}}$$

2.7 พลังงานลมและอากาศพลศาสตร์

อากาศพลศาสตร์ เป็นพลศาสตร์สาขาหนึ่งซึ่งว่าด้วยการเคลื่อนที่ของอากาศและแก๊สอื่นๆ หรือที่เกี่ยวข้องกับแรงต่างๆ ที่กระทำต่อวัตถุที่กำลังเคลื่อนที่ผ่านไปในอากาศ เรื่องของพลังงานลมและหลักการของอากาศพลศาสตร์นั้นมีความสัมพันธ์โดยตรง และเป็นสิ่งที่มีความจำเป็นต้องเรียนรู้หลักการต่างๆ

แรงต้านอากาศ หมายถึงแรงอุปสรรคอากาศของวัตถุที่เคลื่อนย้ายวัตถุที่เคลื่อนที่ด้วยแรงโดยมีทิศทางตรงข้ามกับแรงที่ใช้ในการเคลื่อนที่ของวัตถุ และค่าสัมประสิทธิ์แรงต้านอากาศมีผลจากรูปร่างของวัตถุ

2.8 ระบบควบคุม

ระบบควบคุม (Control system) หมายถึง การควบคุมระบบหรือสิ่งที่ผู้ออกแบบต้องการควบคุมให้ได้ค่าผลลัพธ์ในรูปแบบของเอาต์พุตที่ต้องการซึ่งทำได้โดยการป้อนค่าอินพุตให้กับระบบโดยนิยามศัพท์พื้นฐานของงานระบบควบคุมมีดังนี้

1. อินพุต (Input) หมายถึง การสัญญาณเข้าที่ต้องการป้อนให้กับระบบรับรู้ซึ่งอาจแสดงในรูปแบบของสัญญาณทางไฟฟ้า

2. ระบบ (System) หมายถึงสิ่งที่ต้องการหรือระบบที่ต้องการควบคุม ประกอบด้วยชุดควบคุมกระบวนการ (Process) ซึ่งอาจเป็นเครื่องมืออุปกรณ์หรือเครื่องจักร

3. ระบบควบคุมวงเปิด (Open-loop control system) หมายถึงระบบควบคุมที่ไม่ได้ใช้สัญญาณจากเอาต์พุต มาบ่งบอกถึงลักษณะการควบคุม

4. ระบบควบคุมวงปิด (Closed-loop control system) หรือระบบป้อนกลับ (Feedback control system) หมายถึงระบบควบคุมที่ใช้สัญญาณจากเอาต์พุตมาบ่งบอกหรือคำนวณค่าที่เหมาะสมสำหรับการควบคุม

5. เอาต์พุต (Output) หมายถึง ผลของการทำงานของระบบที่ผ่านการควบคุมซึ่งจะแสดงในรูปแบบ ผลตอบสนองทางกล (Mechanical response) และผลตอบสนองทางไฟฟ้า (Electrical response)

6. สิ่งรบกวน (Disturbance) หมายถึง สิ่งที่เป็นสาเหตุทำให้เกิดผลลัพธ์ทางเอาต์พุตของระบบเปลี่ยนไปซึ่งอาจอยู่ในรูปของสัญญาณรบกวน (Noise signal) ที่ปนมากับอินพุตระบบ

2.8.1 องค์ประกอบพื้นฐานของระบบควบคุม

1. กำหนดเป้าหมายของการควบคุม (Setpoint) คือการกำหนดค่าเป้าหมายหรือค่าอ้างอิง (Reference input) ของการควบคุมงานที่ต้องการ

2. ชุดควบคุม (Controller) คือ ส่วนที่ทำหน้าที่ควบคุมการทำงานและประมวลผลระบบซึ่งจะประกอบด้วยชุดฮาร์ดแวร์ที่เป็นวงจรอิเล็กทรอนิกส์และชุดซอร์ฟแวร์ที่เป็นโปรแกรมคำสั่งเพื่อควบคุมการทำงานของระบบ โดยมีเป้าหมายให้เกิดการตอบสนองค่าเอาต์พุตที่ต้องการซึ่งมีทั้งระบบควบคุมที่เป็นแอนะล็อกและระบบควบคุมดิจิตอล

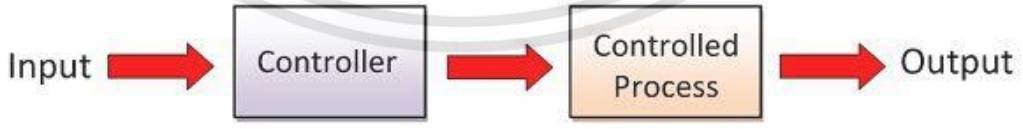
3. ชุดกระตุ้นระบบ (Actuator) คือส่วนที่ทำหน้าที่เปลี่ยนสัญญาณควบคุมให้อยู่ในรูปของสัญญาณที่สามารถปรับเปลี่ยนได้ โดยอุปกรณ์ที่ทำหน้าที่เปลี่ยนสัญญาณควบคุมไปเป็นพลังงานต่างๆ ที่ระบบต้องการ

4. กระบวนการ (Process) คือ ส่วนที่ทำหน้าที่ดำเนินการ (Operation) เมื่อได้รับสัญญาณจากชุดกระตุ้น

5. ผลการควบคุมระบบ (Output system) คือ ส่วนที่ทำหน้าที่แสดงผลของการควบคุมระบบซึ่งอาจแสดงในรูปแบบผลตอบสนองของระบบ ทำให้ทราบค่าเสถียรภาพ (Stability) และค่าความคาดเคลื่อนของระบบ (Error) เพื่อใช้ในการพิจารณาประสิทธิภาพของระบบควบคุม

2.8.2 พื้นฐานของระบบควบคุมแบบวงเปิดและวงปิด

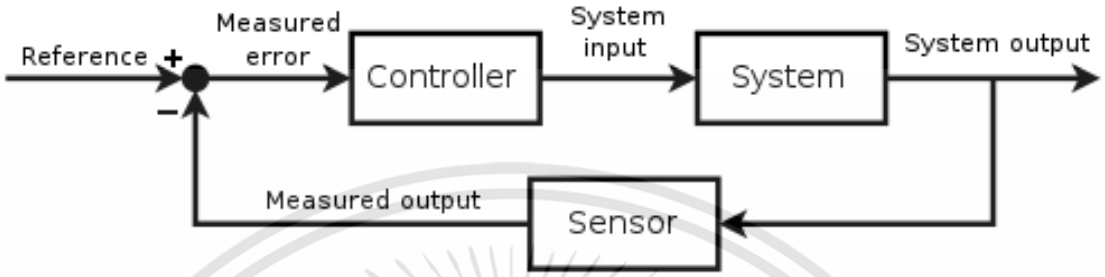
ระบบควบคุมพื้นฐานเมื่อมีค่าอินพุตเข้ามา ตัวควบคุมจะส่งสัญญาณ ไปยังกระบวนการที่ถูกควบคุมและให้ผลลัพธ์เป็นเอาต์พุตตามที่ต้องการโดยถ้าค่าอินพุตคงงที่ ผลลัพธ์ทางเอาต์พุตจะไม่มีการเปลี่ยนแปลง แต่เอาต์พุตจะเบี่ยงเบนได้เมื่อถูกรบกวนจากปัจจัยภายนอก ระบบควบคุมแบบนี้จะเรียกว่า ระบบควบคุมแบบวงเปิด (Open-loop control system) การปรับค่าเอาต์พุตที่เบี่ยงเบนให้กลับสู่ค่าจะต้องปรับด้วยมือเท่านั้น



รูปที่ 2.5 บล็อกไดอะแกรมของระบบควบคุมแบบวงเปิด [14]

ระบบที่มีการนำค่าเอาต์พุตที่ได้มาป้อนกลับมาเปรียบเทียบกับค่าอินพุตเพื่อปรับปรุงสัญญาณในการควบคุมให้มีค่าเหมาะสมที่จะกระตุ้นให้กระบวนการที่ควบคุมให้เอาต์พุตที่ถูกต้องอยู่ตลอดเวลาเรียกระบบนี้ว่า ระบบควบคุมป้อนกลับหรือระบบควบคุมป้อนกลับโดยการเปรียบเทียบ

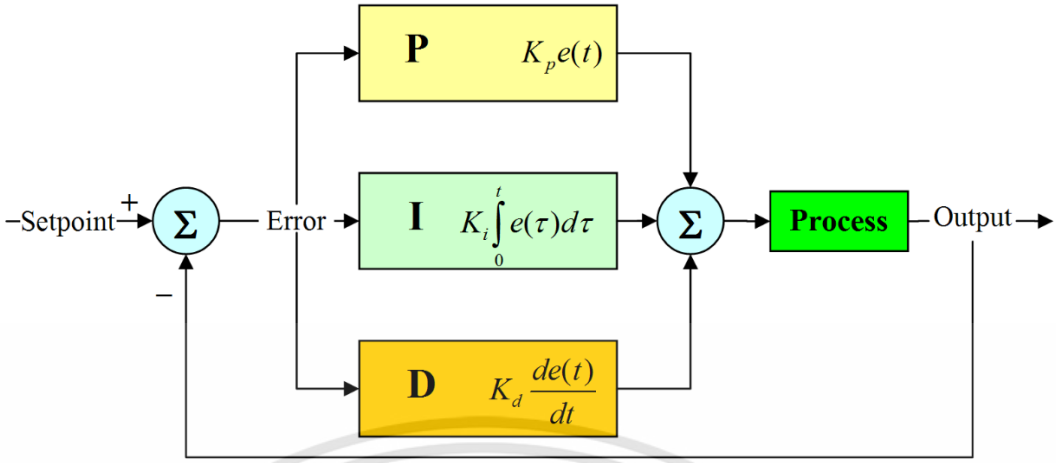
สัญญาณค่าผิดพลาดที่เกิดขึ้นจากผลต่างของค่าอินพุตและค่าเอาต์พุตจะถูกส่งไปยังตัวควบคุม เพื่อสร้างสัญญาณควบคุม ไปกระตุ้นกระบวนการที่ถูกควบคุมและให้เอาต์พุตออกมาตามที่ตั้งไว้ถ้าเกิดค่าเอาต์พุตมีการเบี่ยงเบนไปจากผลกระทบที่มาจากปัจจัยภายนอก ส่งผลให้เอาต์พุตถูกปรับปรุงแก้ไขให้กลับมายังค่าที่ต้องการตามเดิมได้ด้วยวงรอบการทำงานของระบบควบคุมเอง



รูปที่ 2.6 บล็อกไดอะแกรมของระบบควบคุมแบบวงปิด [14]

2.8.3 ระบบควบคุม PID

ระบบควบคุมแบบ PID (Proportional Integral and Derivative Controller : PID) เป็นระบบควบคุมแบบป้อนกลับ ซึ่งค่าที่นำไปใช้ในการคำนวณเป็นค่าความผิดพลาดที่หามาจากความแตกต่างของตัวแปรในระบบและค่าที่ต้องการ ตัวควบคุมจะพยายามลดค่าผิดพลาดให้เหลือน้อยที่สุด ด้วยการปรับค่าพารามิเตอร์ของกระบวนในแต่ละส่วนของตัวควบคุม PID ในการปรับค่าพารามิเตอร์นั้น โดยวิธีคำนวณของ PID ขึ้นอยู่กับสามตัวแปรคือค่าสัดส่วน (Proportional: P) ปริพันธ์ (Integral: I) และอนุพันธ์ (Derivative: D) ค่าสัดส่วนกำหนดจากผลของความผิดพลาดในปัจจุบัน ค่าปริพันธ์กำหนดจากผลบนพื้นฐานของผลรวมความผิดพลาดที่ผ่านพ้นไป และค่าอนุพันธ์กำหนดจากผลบนพื้นฐานของอัตราการเปลี่ยนแปลงของค่าความผิดพลาด น้ำหนักที่เกิดจากการรวมกันของทั้งสามเทอมนี้จะใช้ในการปรับระบบโดยการปรับค่าคงที่ใน PID ตัวควบคุมสามารถปรับรูปแบบการควบคุมให้เหมาะกับที่ระบบต้องการได้ การตอบสนองของตัวควบคุมจะอยู่ในรูปของการไหวตัวของตัวควบคุมจนถึง ค่าความผิดพลาด ค่าโอเวอร์ชูต (Overshoot) และ ค่าแกว่งของระบบ (Oscillation)



รูปที่ 2.7 บล็อกไดอะแกรมของการควบคุมแบบ PID [14]

จากรูปที่ 2.7 เมื่อนำค่าสัญญาณขาออกของเทอมสัดส่วน, สัญญาณขาออกปริพันธ์ และสัญญาณขาออกอนุพันธ์ มารวมกันกลายเป็นสัญญาณขาออกของการควบคุมแบบ PID โดยกำหนดให้ $u(t)$ เป็นสัญญาณขาออก จะได้สมการ PID ดังนี้

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \quad (7)$$

K_p คือ อัตราขยายสัดส่วน, ตัวแปรปรับค่าได้

K_i คือ อัตราขยายปริพันธ์, ตัวแปรปรับค่าได้

K_d คือ อัตราขยายอนุพันธ์, ตัวแปรปรับค่าได้

e คือ ค่าความผิดพลาด (Setpoint - Process value)

t คือ เวลา

τ คือ ตัวแปรปริพันธ์

ตัวควบคุมแบบสัดส่วน (Proportional: P)

เทอมของสัดส่วนจะเปลี่ยนแปลงเป็นสัดส่วนของค่าความผิดพลาด การตอบสนองของสัดส่วนสามารถทำได้โดยการคูณค่าความผิดพลาดด้วยค่าคงที่ K_p หรือที่เรียกว่าอัตราขยายสัดส่วน ดังรูปที่ 2.8 แสดงให้เห็นว่าความแตกต่างของการกำหนดพารามิเตอร์แบบสัดส่วนที่แตกต่างกันที่ K_p= 0.5, 1 และ 2 เมื่อ K_i = 1 และ K_d = 1 เทอมของสัดส่วนจะเป็นไปตามสมการที่

$$P_{out} = K_p e(t) \quad (8)$$

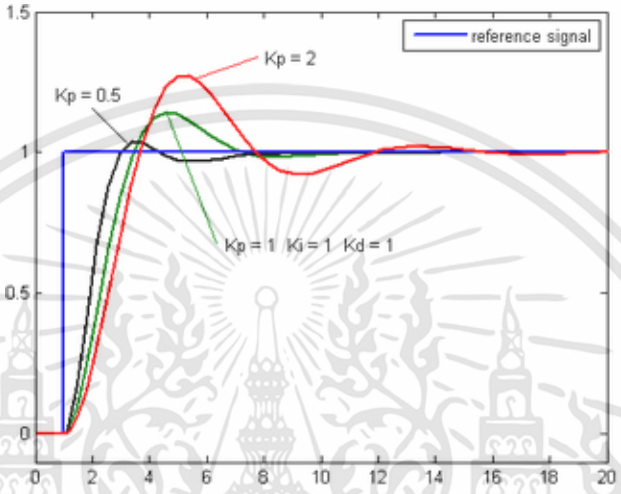
เมื่อ P_{out} คือ สัญญาณขาออกของเทอมสัดส่วน

K_p คือ อัตราขยายสัดส่วน, ตัวแปรปรับค่าได้

e คือ ค่าความผิดพลาด (Setpoint - Process value)

t คือ เวลา

ผลอัตราขยายสัดส่วนที่สูง ค่าความผิดพลาดก็จะเปลี่ยนแปลงมากเช่นกัน แต่ถ้าสูงเกินไประบบจะไม่เสถียรได้ ในทางตรงกันข้ามผลอัตราขยายสัดส่วนที่ต่ำ ระบบความจะมีผลตอบสนองต่อกระบวนการน้อยตามไปด้วย



รูปที่ 2.8 แผนภูมิ Process value ต่อเวลา กำหนดค่า K_p เป็น 0.5, 1 และ 2 (K_i=1 และ K_d=1) [14]

ตัวควบคุมปริพันธ์ (Integral: I)

คุณสมบัติของตัวควบคุมปริพันธ์ในการกำจัดความคลาดเคลื่อนหรือออฟเซต (Offset) ผลจากเทอมปริพันธ์เป็นสัดส่วนของความผิดพลาด และระยะเวลาของความผิดพลาด ผลรวมของความผิดพลาดในทุกช่วงเวลาจะให้ออฟเซตสะสมที่ควรจะเป็นในก่อนหน้า ความผิดพลาดสะสมจะถูกคูณโดยอัตราขยายปริพันธ์ขนาดของผลของเทอมปริพันธ์จะกำหนดโดยอัตราขยายปริพันธ์ K_i เทอมปริพันธ์จะเป็นไปตามสมการที่

$$I_{out} = K_i \int_0^t e(\tau)d\tau \quad (9)$$

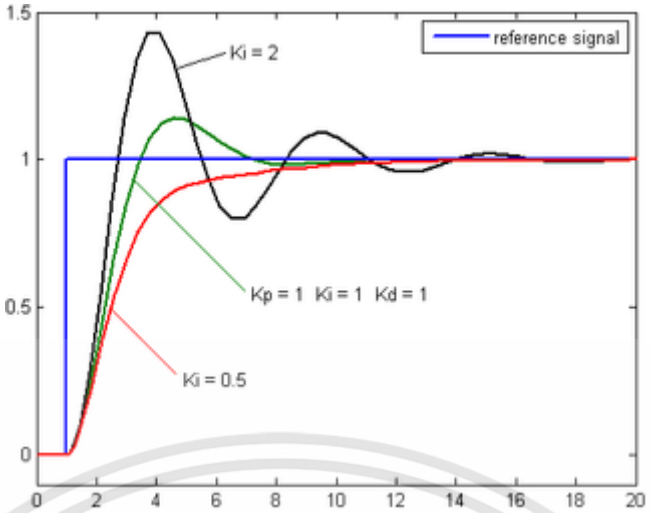
เมื่อ I_{out} คือ สัญญาณขาออกของเทอมปริพันธ์

K_i คือ อัตราขยายปริพันธ์, ตัวแปรปรับค่าได้

e คือ ความผิดพลาด (Setpoint - Process value)

t คือ เวลา

τ คือ ตัวแปรปริพันธ์

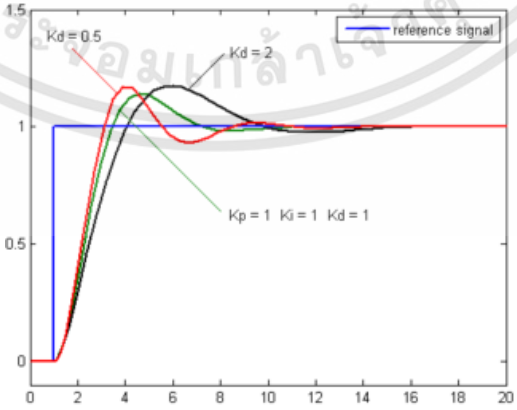


รูปที่ 2.9 แผนภูมิ Process value ต่อเวลา กำหนดค่า K_i เป็น 0.5, 1 และ 2 (K_p=1 และ K_d=1) [14]

เทอมปริพันธ์ (เมื่อรวมกับเทอมสัดส่วน) จะเร่งกระบวนการให้เข้าสู่จุดที่ต้องการและขจัดความผิดพลาดที่เหลืออยู่ที่เกิดจากการใช้เพียงเทอมสัดส่วน เทอมปริพันธ์เป็นการตอบสนองต่อความผิดพลาดสะสมในอดีต จึงสามารถทำให้เกิดโอเวอร์ชูตได้

ตัวควบคุมอนุพันธ์ (Derivative: D)

ตัวควบคุมอนุพันธ์ คือตัวควบคุมที่ก่อให้เกิดผลตรงข้ามกับตัวอินทิกรัล ดังนั้นจึงใช้ในการปรับปรุงกระบวนการที่มีการล้าหลังทางเวลา (Time lag) มากๆ ทำให้ผลตอบสนองรวดเร็วขึ้นและช่วงเวลาการแกว่งที่สั่นลง ข้อเสียของตัวอนุพันธ์คือ มีความไวต่อสัญญาณรบกวนเป็นอย่างมากเพราะมีผลตอบสนองโดยตรง ต่ออัตราการเปลี่ยนแปลงของสัญญาณที่วัดได้ ดังนั้นแม้สัญญาณรบกวนจะมีขนาดเล็ก แต่ก็อาจก่อให้เกิดการเปลี่ยนแปลงต่อสัญญาณออกของตัวควบคุม



รูปที่ 2.10 แผนภูมิ Process value ต่อเวลา กำหนดค่า K_d เป็น 0.5, 1 และ 2 (K_p=1 และ K_i =1) [14]

รูปที่ 2.10 แสดงให้เห็นการเปลี่ยนแปลงพารามิเตอร์ K_d = 0.5, 1 และ 2 เมื่อ K_p = 1 และ K_i = 1 เทียบกับอินพุตที่เป็นสัญญาณขั้นบันไดอัตราการเปลี่ยนแปลงของความผิดพลาดจากกระบวนการนั้นคำนวณหาจากความชันของความผิดพลาดทุก ๆ เวลาและคูณด้วยอัตราขยายอนุพันธ์ K_d ขนาดของผลของเทอมอนุพันธ์ขึ้นกับอัตราขยายอนุพันธ์ K_d เทอมอนุพันธ์เป็นไปตามสมการที่

$$D_{out} = K_d \frac{d}{dt} e(t) \quad (11)$$

เมื่อ D_{out} คือ สัญญาณขาออกของเทอมอนุพันธ์

K_d คือ อัตราขยายอนุพันธ์, ตัวแปรปรับค่าได้

e คือ ความผิดพลาด (Setpoint - Process value)

t คือ เวลา

ตัวควบคุมอนุพันธ์มีหลักการทำงาน คือตัวควบคุมตอบสนองต่ออัตราการเปลี่ยนแปลงของความคลาดเคลื่อน ถึงแม้ว่าความคลาดเคลื่อนยังมีค่าเล็กอยู่ สัญญาณออกของตัวอนุพันธ์ไม่ได้สัมพันธ์กับขนาดของความคลาดเคลื่อน แต่ขึ้นอยู่กับการเปลี่ยนแปลงของความคลาดเคลื่อน ถ้าความคลาดเคลื่อนมีค่าคงที่ตัวอนุพันธ์จะให้สัญญาณออกเป็นศูนย์ มีผลดีทำให้ผลตอบสนองที่เกิดก่อนที่ความคลาดเคลื่อนจะเพิ่มมากขึ้น และทำให้ระบบมีผลตอบสนองที่เร็วขึ้น

2.8.4 คุณสมบัติของระบบควบคุมแบบ PID

ระบบควบคุมโดยทั่วไป สามารถแบ่งคุณสมบัติของระบบได้เป็น 4 แบบ สามารถสรุปรายละเอียดได้ดังนี้

1. ระบบควบคุมแบบ P (P controller) จะมีคุณสมบัติ
   - ลดค่า Rise time (tr) ทำให้ระบบทำงานเร็วขึ้นในช่วงแรก
   - เพิ่ม Overshoot (Mp) ทำให้ระบบแกว่งในช่วงเริ่มต้น แต่ส่งผลให้ค่า Steady state error (ess) ลดลง
2. ระบบควบคุมแบบ PD (PD controller) จะมีคุณสมบัติ
   - ลด Overshoot (Mp) ทำให้ระบบมีการแกว่งน้อยลงในช่วงเริ่มต้น
   - ลด Settling time (ts) ทำให้ระบบทำงานถึงจุดคงที่ (Steady state) เร็วขึ้น
3. ระบบควบคุมแบบ PI (PI Controller) จะมีคุณสมบัติ
   - ลดค่า Rise time (tr) ทำให้ระบบทำงานเร็วขึ้นในช่วงแรก
   - กำจัดค่า Steady state error (ess) เพื่อให้ค่าเอาต์พุตเทียบเท่ากับค่าอินพุต

4. ระบบควบคุมแบบ PID (PID Controller) เป็นการนำคุณสมบัติของระบบควบคุมทั้ง 3 แบบ คือ P, PD และ PI มารวมกัน โดยสามารถกำหนดค่าเกนทั้ง 3 แบบ คือ P, I และ D ได้อย่างอิสระ ทำให้สามารถออกแบบระบบควบคุมให้เป็นไปตามที่ต้องการได้ โดยทำการปรับค่าเกนทั้ง 3 แบบ และนำมาพิจารณาผลตอบสนองที่ได้ เมื่อปรับค่าเกนจนได้ผลตอบสนองตามต้องการแล้ว จึงนำค่าที่ได้ไปติดตั้งหรือออกแบบเป็นวงจรใช้งานต่อไป

ตารางที่ 2.1 ผลของการเพิ่มค่าตัวแปรอิสระ

ตัวแปร	Rise time (tr)	Overshoot (Mp)	Settling time (ts)	Steady-state error (ess)	เสถียรภาพ
K_p	ลด	เพิ่ม	เปลี่ยนแปลงน้อย	ลด	ลด
K_i	ลด	เพิ่ม	เพิ่ม	กำจัด	ลด
K_d	ลดลงเล็กน้อย	ลดลงเล็กน้อย	ลดลงเล็กน้อย	ตามทฤษฎีไม่มีผล	ดีขึ้นถ้า K_d มีค่าน้อย

2.8.5 การปรับจูนค่าคงที่ของตัวควบคุม

การปรับจูนค่าคงที่ของตัวควบคุม เป็นการปรับการทำงานของระบบให้เป็นไปตามความต้องการของผู้ใช้งานโดยการปรับจูนนี้จะส่งผลถึงค่าความผิดพลาด ค่าโอเวอร์ชูต (Overshoot) และการแกว่งของระบบ (Oscillation) ซึ่งสมการ PID มีค่าคงที่อยู่ 3 ตัวที่สามารถปรับได้คือ K_p, K_i และ K_d ที่ต้องกำหนดค่าเข้าไป การเลือกค่าที่เหมาะสมจะทำให้ระบบสมดุลและได้ผลลัพธ์ตามความต้องการมากที่สุด ซึ่งก็คือการจูน PID ส่วนวิธีการจูน สามารถแบ่งได้สองแบบใหญ่ๆ คือ Close-loop tuning และ Open-loop tuning

Trial & error close-loop tuning

วิธีนี้คือการทดลองป้อน K_p, K_i และ K_d เข้าระบบ แล้วสังเกตการเปลี่ยนแปลงของระบบและหาค่าที่ทำให้ระบบสมดุลมากที่สุด โดยมีขั้นตอนการปรับจูน ดังนี้

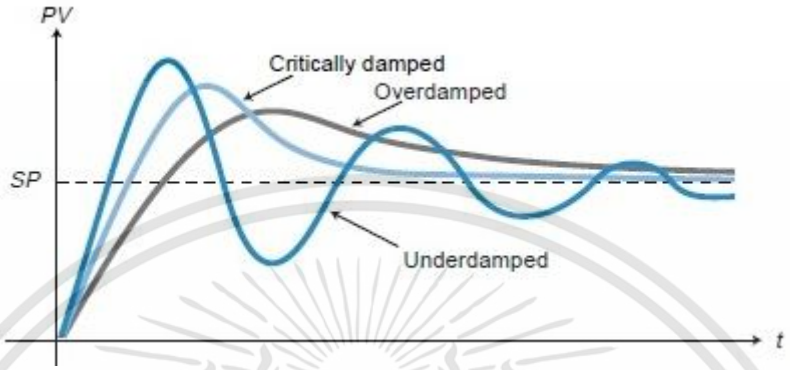
1. เริ่มพล็อตกราฟของ Process variable (PV)

2. ปรับให้ K_i และ K_d เป็นศูนย์

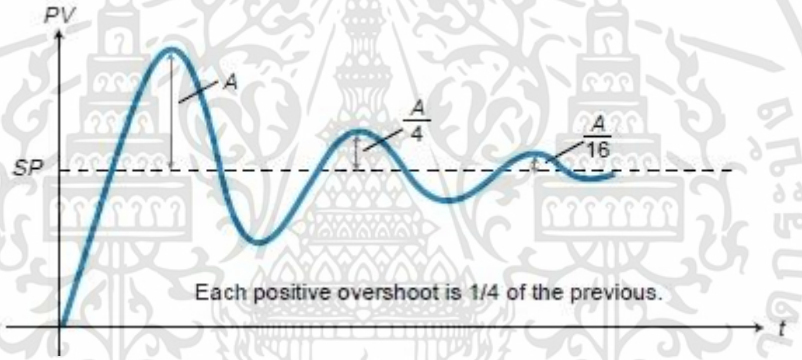
3. เริ่มปรับค่า K_p ค่าน้อยๆ

4. เปลี่ยน PID Controller ให้อยู่ในโหมด Auto

5. ทดสอบระบบโดยเปลี่ยนค่า Setpoint แล้วสังเกต Damping ของระบบ ปรับค่า K_p เพิ่มขึ้นเรื่อยๆ จนระบบมี Damping ตามต้องการ หรือให้เป็น Quarter amplitude decay ดังรูปที่ 2.11



รูปที่ 2.11 กราฟผลการตอบสนองของ Process value [15]



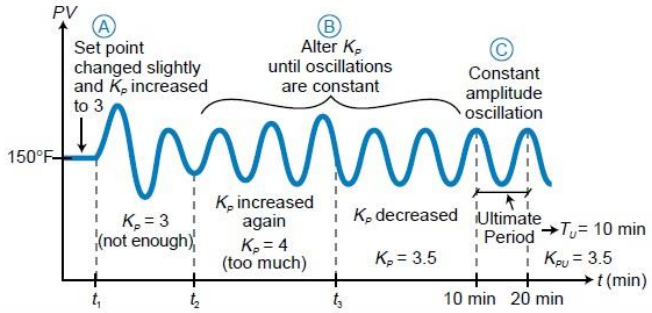
รูปที่ 2.12 กราฟของ Quarter-amplitude [15]

6. ถ้าระบบยังมี Offset ระหว่าง Setpoint และ Process Value ให้ปรับค่า K_i จนระบบอยู่ในความต้องการ

7. ถ้าระบบมีการเกิด Overshoot และ Undershoot ให้ปรับค่า K_d

Ziegler-Nichols Close-Loop Tuning

หลักของวิธีนี้คือหาค่าเกนที่ทำให้ระบบเกิด Oscillation แบบแอมปริจูดคงที่ซึ่งเรียกว่า Ultimate proportional gain (K_{pu}) และคาบการสั่นซึ่งเรียกว่า Ultimate period (T_u) จากนั้นนำทั้งสองค่านี้ไปคำนวณหา K_p, K_i และ K_d ตามตารางด้านล่าง



รูปที่ 2.13 การปรับจูนระบบโดยใช้วิธี Ziegler-Nichols close-loop tuning [15]

ตารางที่ 2.2 พารามิเตอร์การปรับ Ziegler-Nichols closed loop

ชนิดของตัวควบคุม	K_p	T_i	T_d
P	$\frac{K_{pu}}{2}$	∞	0
PI	$\frac{K_{pu}}{2.2}$	$\frac{T_u}{1.2}$	0
PID	$\frac{K_{pu}}{1.7}$	$\frac{T_u}{2}$	$\frac{T_u}{8}$

โดยมีขั้นตอนการจูนดังนี้

1. เริ่มพล็อตกราฟของ Process variable
2. กำหนด K_i และ K_d เป็นศูนย์
3. ปรับ PID Controller ให้อยู่ในโหมด Auto
4. ปรับค่า K_p เริ่มจากค่าน้อยๆ
5. จดค่า K_p และคาบเวลาที่ทำให้เกิด Oscillation แบบแอมปริจูดคงที่
6. คำนวณค่า K_p, K_i และ K_d จากตาราง

บทที่ 3
วิธีการดำเนินงาน

3.1 ขั้นตอนการดำเนินงาน

ขั้นตอนของการดำเนินงานได้วางแผนไว้ดังนี้

1. ศึกษาค้นคว้าทฤษฎีและเอกสารที่เกี่ยวข้อง
2. ศึกษาหลักการของเซนเซอร์และการทำงานของชุดควบคุมคานสมดุล
3. ออกแบบและวางแผนการสร้างชุดอุปกรณ์ของระบบควบคุมคานสมดุล
4. สั่งซื้อบอร์ด Arduino และ อุปกรณ์ของชุดควบคุมคานสมดุล
5. ทดสอบการทำงานของชุดควบคุมที่นำมาใช้
6. เขียนโปรแกรมทดสอบการทำงานของเซนเซอร์ และทำการ Normalization
7. เขียนโปรแกรมควบคุมการทำงานของมอเตอร์
8. เขียนโปรแกรมควบคุมการทำงานของระบบควบคุม โดยใช้การควบคุมแบบเปิด
9. เขียนโปรแกรมควบคุมการทำงานของระบบควบคุม โดยใช้การควบคุมแบบ PID
10. เขียนโปรแกรมการส่งค่าจากเซนเซอร์ไปแสดงผลบนโทรศัพท์มือถือและเว็บบราวเซอร์ผ่านเครือข่าย Wi-Fi บนแอปพลิเคชัน NETPIE
11. ออกแบบส่วน GUI (Graphical user interface) ผ่านแอปพลิเคชัน NETPIE สำหรับการแสดงผล
12. ทดสอบชุดอุปกรณ์ของระบบควบคุมคานสมดุลกับแอปพลิเคชัน NETPIE ผ่านเครือข่ายไร้สาย Wi-Fi
13. สรุปผลและจัดทำเอกสารรายงานการวิจัย

3.2 อุปกรณ์ที่เกี่ยวข้อง

3.2.1 บอร์ด Arduino ESP32

ESP32 เป็นชิปไอซีไมโครคอนโทรลเลอร์ 32 บิต ที่มี Wi-Fi และบลูทูธเวอร์ชั่น 4.2 ในตัว ซึ่งเป็นรุ่นต่อของชิปไอซี ESP8266 รุ่นยอดนิยม ผลิตโดยบริษัท Espressif จากประเทศจีน รองรับการเขียนโปรแกรมโดยใช้โปรแกรม Arduino IDE และรองรับไลบรารีส่วนใหญ่ของ Arduino ทำให้สามารถใช้งานได้ง่าย โดย CPU ใช้สถาปัตยกรรม Tensilica LX6 จำนวน 2 core สัญญาณนาฬิกา 240 MHz สามารถแยกการทำงานระหว่างโปรแกรมจัดการ Wi-Fi และแอพพลิเคชั่นออกจากกันได้

ทำให้มีสเถียรภาพเพิ่มขึ้นมาก มีแรม 520 KB มาในตัว นอกจากนี้ยังมี GPIO 32 ช่อง และมีช่อง ADC เพิ่มขึ้นเป็น 12 ช่อง จากเดิม ESP8266 มีเพียงช่องเดียว ใช้แรงดันไฟฟ้า 3.3 V ใช้กระแสไฟฟ้า 2.5 uA ซึ่งถือว่าเป็นบอร์ดที่คุ้มค่าต่อการลงทุนนำมาใช้งาน

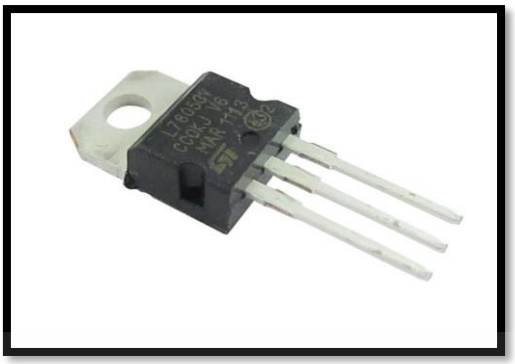
รูปที่ 3.1 บอร์ด Arduino ESP32 [5]

3.2.2 ชุดระบบควบคุมคานสมดุล

เป็นชุดการทดลองควบคุมคานโดยฝั่งขวาและซ้ายของคานจะเป็นใบพัดมอเตอร์ที่ทำหน้าที่เป็นตัวยกคานขึ้นลงโดยฝั่งขวาจะเป็นใบพัดมอเตอร์ที่ทำหน้าที่คล้ายกับโหลดคงที่โดยการให้ใบพัดหมุนในความเร็วคงที่ก็จะได้แรงยกที่คงที่โดยการทำงานของชุดระบบควบคุมนี้จะเป็นการทำงานแบบ Closed loop control system มีตัวต้านทานปรับค่าได้สองตัวที่ทำหน้าที่ในการรับค่า Setpoint variable และค่า Feedback variable เพื่อทำการส่งค่านี้ไปที่ Microcontroller เพื่อทำการประมวลผลในการคำนวณหาค่า Output ที่เป็นสัญญาณ PWM ที่เหมาะสมในการปรับสมดุลของคาน แล้วส่งค่าไปยังวงจรขับมอเตอร์เพื่อนำไปขับมอเตอร์ที่มีใบพัดเพื่อทำการยกคานให้อยู่ในที่ที่สมดุล

3.2.3 วงจรคงค่าแรงดัน (Voltage regulator)

วงจรคงค่าแรงดันเป็น IC ที่ทำหน้าที่เปลี่ยนแรงดันสูง (Vin) ให้เป็นแรงดันต่ำและเรียบคงที่ (Vout) โดยตัว IC7805 จะมีค่าแรงดันเอาต์พุตอยู่ที่ 5V โดยความแตกต่างของค่าแรงดันไฟฟ้าอินพุต (Vin) และ แรงดันไฟฟ้าเอาต์พุต (Vout) จะถูกขับออกมาในรูปแบบของความร้อน ดังนั้นหากค่าความแตกต่างระหว่างแรงดันไฟฟ้าอินพุตและแรงดันไฟฟ้าเอาต์พุตสูง จึงต้องมีแผ่นระบายความร้อน (Heat sink) เพื่อไม่ให้วงจรร้อนจนเกินไปจนเกิดความผิดพลาด



รูปที่ 3.2 LM7805 [1]

3.2.4 Optocoupler หรือ Opto-Isolator

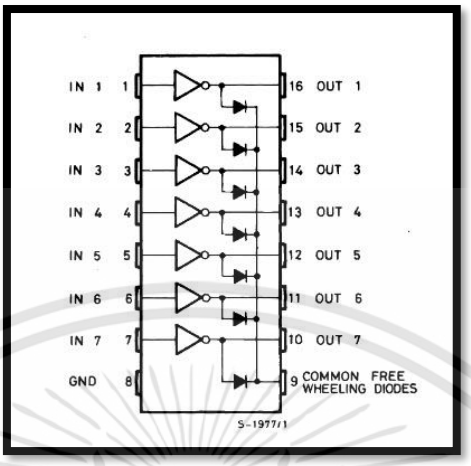
Optocoupler เป็นอุปกรณ์อิเล็กทรอนิกส์สำหรับการเชื่อมต่อทางแสง โดยการเปลี่ยนสัญญาณไฟฟ้าให้เป็นแสง แล้วเปลี่ยนกลับเป็นสัญญาณไฟฟ้าตามเดิม นิยมใช้สำหรับการเชื่อมต่อสัญญาณระหว่างสองวงจรและต้องการแยกกันทางไฟฟ้าโดยเด็ดขาก เพื่อป้องกันการรบกวนกันทางไฟฟ้าระหว่างวงจรสองวงจร ภายในของอุปกรณ์ประกอบด้วยไดโอดเปล่งแสง (LED) ซึ่งทำหน้าที่เป็นตัวส่งแสง (Optical transmitter) และตัวรับแสง (Optical receiver) ซึ่งมักใช้โฟโต้ทรานซิสเตอร์ (Phototransistor) เป็นตัวรับ โดยในระบบควบคุมคานสมดุลนี้จะใช้เป็น PC817

รูปที่ 3.3 PC817 [2]

3.2.5 IC ULN2003

ULN2003 เป็น IC ที่ภายในบรรจุอินเวอร์เตอร์ 7 ตัว มีรูปแบบการจัดขาและวงจรภายในดังรูปที่ 3.4 ใช้กับแรงดันไฟฟ้าสูงสุด 30 V กระแสไฟฟ้าเอาต์พุตสูงสุดในแต่ละขาเท่ากับ 500 mA ทั้งนี้ขึ้นอยู่กับความสามารถในการจ่ายกระแสไฟของแหล่งจ่ายไฟด้วย นอกจากนั้น ULN2003 ยังมีการต่อ

ไดโอดป้องกันแรงดันย้อนกลับจากอุปกรณ์เอาต์พุตที่มีโครงสร้างเป็นขดลวดไว้ที่ทุกขาเอาต์พุต ทำให้ใช้ขับโหลดที่เป็นขดลวด เช่น รีเลย์ หรือมอเตอร์ไฟตรงขนาดเล็กถึงขนาดกลางได้



รูปที่ 3.4 ULN2003 [3]

3.2.6 AC/DC Adapter 9V 3A

Adapter สำหรับแปลงแรงดันจาก กระแสสลับ AC 220V แปลงเป็น กระแสตรง DC 9V สามารถสร้างกระแสได้สูงสุด 3A

รูปที่ 3.5 AC/DC Adapter 9V 3A [3]

3.2.7 มอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงหรือ DC Motor ที่ใช้ไฟ DC ในการทำงานและมีแปรงถ่านอยู่ในตัว โดยแรงดันที่มอเตอร์ใช้ในการทำงาน (Forward voltage) อยู่ที่ตั้งแต่ 3-12 V แต่ควรใช้งานในช่วงไม่เกิน 6 V เพื่ออายุการใช้งานที่นานขึ้น ส่วนกระแสที่มอเตอร์ใช้ในการทำงานขณะที่เพลาไม่มีโหลดติดอยู่ (Forward แcurrent) ที่แรงดัน 5 V จะใช้กระแสประมาณ 250 mA และกระแสที่มอเตอร์ใช้เมื่อมีโหลด (Stall current) อยู่ในช่วง 800 mA ถึง 1 A

รูปที่ 3.6 มอเตอร์ที่ใช้งาน [4]

3.3 ซอฟต์แวร์ที่เกี่ยวข้อง

3.3.1 Arduino IDE

IDE คือซอฟต์แวร์เครื่องมือสำหรับพัฒนาโปรแกรม ซึ่งมีสิ่งอำนวยความสะดวกต่างๆ เช่น RUN, Compile, Debug ซึ่งมี GUI ที่ถูกออกแบบมาให้มีสภาวะแวดล้อม (Environment) เหมาะสมการพัฒนาโปรแกรม โดยหน้าที่หลักของ IDE คือการเขียนไฟล์ เปิดไฟล์ บันทึกไฟล์ ทดสอบการทำงาน จัดเตรียมข้อมูล รวมถึงจัดการ Directory สำหรับภาษานั้นๆ ที่รองรับ ดังนั้น Arduino IDE คือ ซอฟต์แวร์เครื่องมือสำหรับพัฒนาโปรแกรมด้วยภาษา C/C++ สำหรับควบคุมบอร์ด Arduino และเครื่องมือต่าง ๆ ที่จำเป็นสำหรับใช้งานบอร์ด Arduino เช่น Serial Monitor, Compile, Libraries ฯลฯ

รูปที่ 3.7 สัญลักษณ์โปรแกรมArduino IDE

3.3.2 Visual Studio

Visual Studio คือ โปรแกรมที่เป็นเครื่องมือที่ช่วยพัฒนาซอฟต์แวร์และระบบต่างๆ ซึ่งสามารถติดต่อสื่อสารพูดคุยกับคอมพิวเตอร์ในระดับหนึ่ง แต่ยังไม่สามารถพัฒนาเป็นระบบเองได้ และได้รวบรวมเครื่องมือพัฒนาต่างๆ ที่ใช้ในการพัฒนาโปรแกรมตั้งแต่ หน้าจอที่ใช้พัฒนาโปรแกรม (Development interface) เครื่องมือในการตรวจหาจุดผิดในโปรแกรม (Debugging tool) ตัวช่วยอัตโนมัติในการเขียนโปรแกรม (Wizard tool) ตัวจัดการฐานข้อมูล (Database management) และส่วนประกอบอื่นๆ ที่จำเป็นในการพัฒนาโปรแกรม

รูปที่ 3.8 สัญลักษณ์โปรแกรม Visual Studio

3.3.3 NETPIE

NETPIE เป็น IoT (Internet of Things) Cloud Platform ที่พัฒนาขึ้นโดยทีมงานวิจัยและยังเป็น Middleware ที่มี Distributed MQTT Brokers ซึ่งเป็นเสมือนจุดนัดพบให้สิ่งต่างๆ (Things) มาติดต่อสื่อสารและทำงานร่วมกันผ่านวิธีการส่งข้อความแบบ Publish/Subscribe NETPIE มีโครงสร้าง สถาปัตยธรรมเป็นคลาวด์อย่างแท้จริงในทุกองค์ประกอบ ทำให้สามารถขยายตัวได้อย่างอัตโนมัติ (Auto-scale) สามารถดูแลและซ่อมแซมตัวเองได้อัตโนมัติเมื่อส่วนหนึ่งส่วนใดในระบบมีปัญหา (Self-healing. Self-recovery) โดยไม่ต้องพึ่งดูแลระบบ การบริหารจัดการระบบเป็นแบบ Plug-and-Play ไม่ต้อง Configure หรือปรับแต่ง

รูปที่ 3.9 สัญลักษณ์โปรแกรม NETPIE

3.3.4 Fritzing

Fritzing เป็นโปรแกรมที่ใช้ในการออกแบบวงจรและแผ่นปริ้นให้กับบอร์ดวงจรต่างๆ เช่น Raspberry pi, Arduino และบอร์ดต่างๆ อีกมากมายเป็นโปรแกรมขนาดเล็กติดตั้งง่ายและใช้งานง่ายเหมาะมากสำหรับคนที่ต้องการเรียนรู้ เป็นตัวช่วยในการออกแบบวงจรลงบน BreadBoard ตัวโปรแกรมรองรับทั้ง Widows, MacOS, Linux

รูปที่ 3.10 สัญลักษณ์โปรแกรม Fritzing

3.4 การออกแบบและการวางแผนการทำงาน

3.4.1 การออกแบบและการวางแผนทางด้านฮาร์ดแวร์

1. ศึกษาและทำความเข้าใจเกี่ยวกับระบบ
2. ศึกษาและเลือกไมโครคอนโทรลเลอร์ที่เหมาะสมกับงาน
3. ทดสอบผลการวัดและทำการ Normalization เซนเซอร์
4. เลือกอุปกรณ์สำหรับแหล่งจ่ายไฟ และอุปกรณ์อื่นๆ ที่ต้องใช้ในกาต่อร่วมกัน
5. ออกแบบชุดอุปกรณ์ระบบควบคุมคานสมดุลแบบไร้สาย และระบบควบคุม
6. ประกอบอุปกรณ์ต่างๆ ตามรูปแบบที่ออกแบบไว้

3.4.2 การออกแบบและการวางแผนทางด้านซอฟต์แวร์

1. ศึกษาและทำความเข้าใจเกี่ยวกับภาษาที่ใช้ในการเขียนบอร์ดไมโครคอนโทรลเลอร์
2. กำหนดรูปแบบการแสดงผล
3. เขียนโปรแกรม Arduino เพื่อรับค่าจากเซนเซอร์ ควบคุมการทำงานของระบบ และส่งค่าไปแสดงผลตามรูปแบบทที่กำหนด
4. เขียนโปรแกรม Arduino IDE เพื่อเชื่อมต่อการรับ-ส่งค่าบนแอพพลิเคชันผ่านสมาร์ทโฟน

5. สร้าง GUI (Graphical user interface) และจอแสดงผลผ่านแอพพลิเคชัน NETPIE

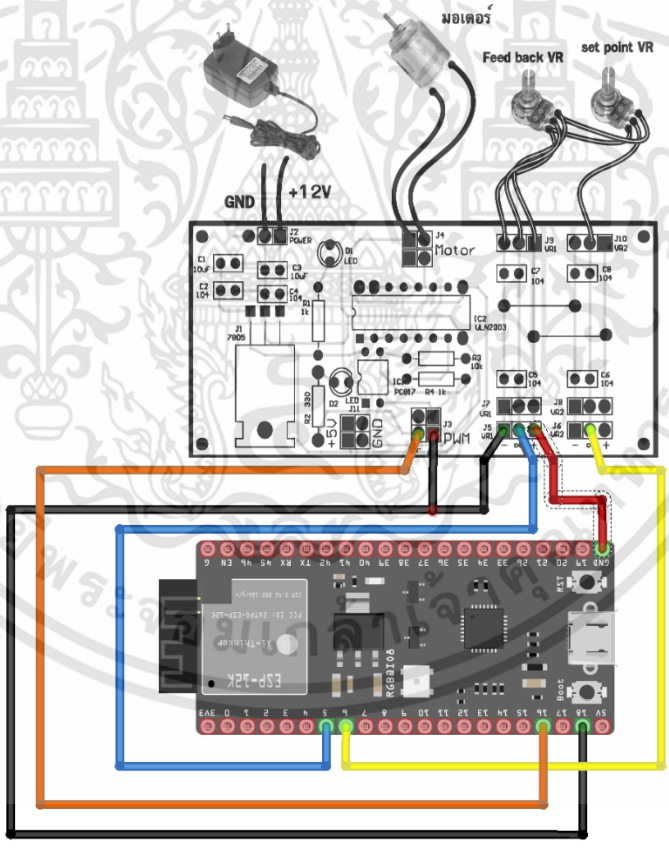
6. ทดสอบการทำงานร่วมกันระหว่างชุดอุปกรณ์ของระบบควบคุมและชุดการทำงานบนแอพพลิเคชัน NETPIE ผ่านเครือข่ายไร้สาย WiFi

3.5 วิธีการดำเนินงาน

3.5.1 การทดสอบการทำงานของเซนเซอร์

การทดสอบการทำงานของตัวต้านทานปรับค่าได้มีสองตัวคือ Setpoint และ Feedback

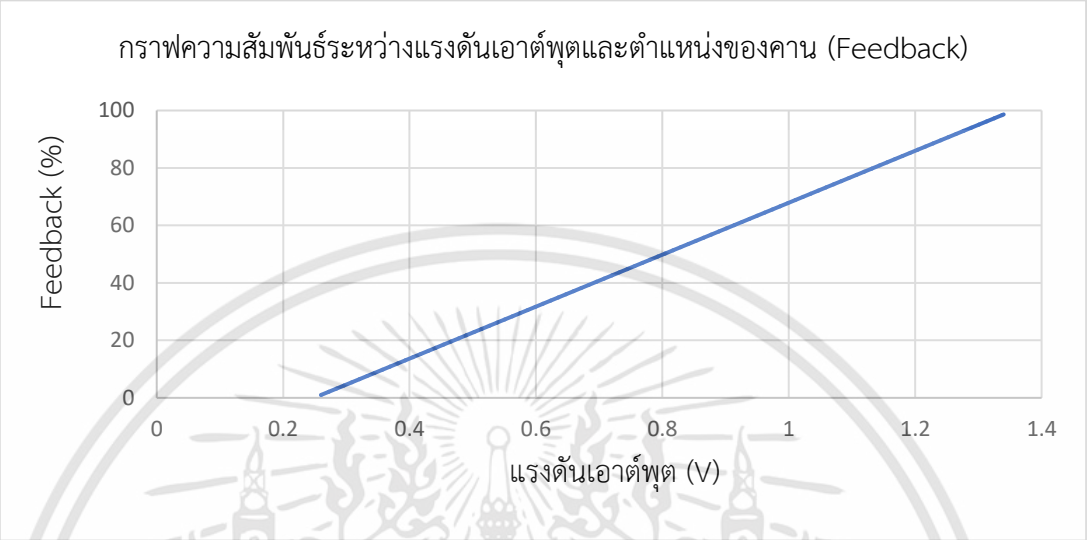
ทดสอบเพื่อดูการเปลี่ยนแปลงของค่าแรงดันเอาต์พุตของเซนเซอร์ เมื่อมีการเปลี่ยนแปลงของตำแหน่ง โดยจะนำโมดูลเซนเซอร์ต่อเข้ากับบอร์ดไมโครคอนโทรลเลอร์ Arduino ESP 32 โดยการต่อจะเป็นการต่อในเส้นสีน้ำเงินที่เป็นตัวต้านทานของ Feedback สีเหลืองเป็นตัวต้านทานของ Setpoint สีแดงเป็นไฟ 3.3 V และ สีดำเป็นกราวด์



รูปที่ 3.11 การต่อโมดูลเข้ากับบอร์ด ESP 32

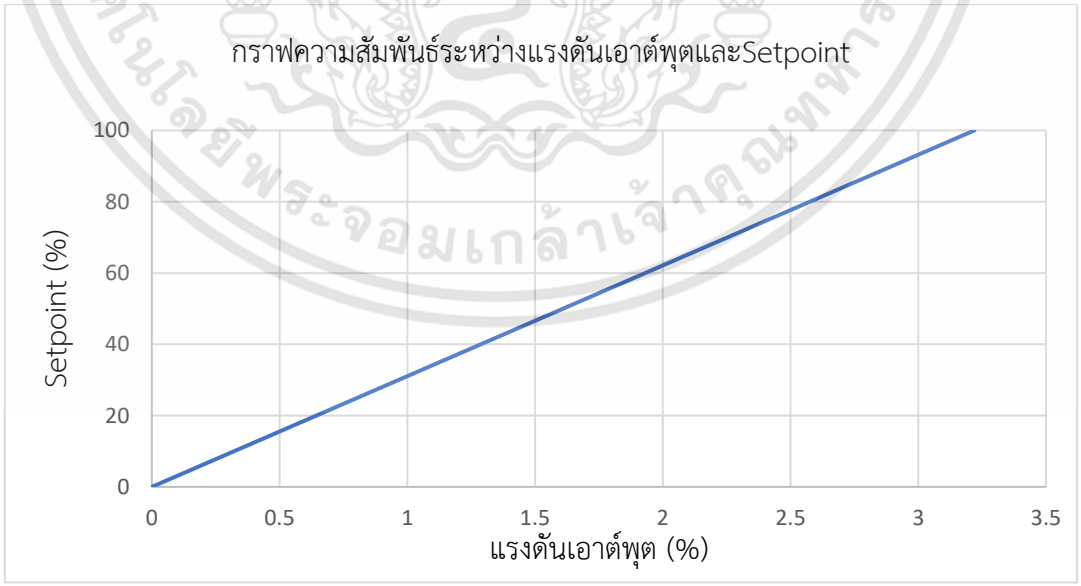
ในการทดสอบเซนเซอร์ Feedback จะทำการต่อคานสมดุลเข้ากับตัวต้านทานปรับค่าได้แล้วกดคานให้ต่ำที่สุดในฝั่งขวาแล้วทำการวัดค่าแรงดันจากวงจรแบ่งแรงดันโดยใช้ตัวต้านปรับค่าได้

หลังจากนั้นทำการยกคานขึ้นทุกๆ 15 องศาแล้วทำการวัดค่าแรงดันจนคานยกขึ้นสุดและอีกฝั่งของคานกดลงต่ำสุด โดยค่าแรงดันจะเพิ่มขึ้นทุกๆ ครั้งที่คานยกขึ้นโดยความสัมพันธ์ของแรงดันเอาต์พุตและตำแหน่งของคาน (Feedback) ดังกล่าวจะแสดงในรูปที่ 3.12



รูปที่ 3.12 กราฟความสัมพันธ์ระหว่างแรงดันเอาต์พุตและตำแหน่งของคาน (Feedback)

ในการทดสอบเซนเซอร์ Setpoint จะทำการปรับปุ่มหมุนให้อยู่ในจุดที่ต่ำสุดแล้วค่อยปรับปุ่มหมุนให้เพิ่มขึ้นเรื่อยๆ ตามลำดับแล้ววัดค่าแรงดันเอาต์พุตที่ตัวต้านทานปรับค่าได้จะพบว่าค่าแรงดันเอาต์พุตจะมีค่าเพิ่มขึ้นเรื่อยๆ ตามปุ่มหมุนที่หมุนเพื่อเพิ่มค่าขึ้นเลยเรื่อยโดยความสัมพันธ์ดังกล่าวจะแสดงในรูปที่ 3.13



รูปที่ 3.13 กราฟความสัมพันธ์ระหว่างค่า Setpoint และ ค่าแรงดันเอาต์พุต

หลังจากได้ช่วงการทำงานที่ต้องการแล้ว จึงได้ทำการหาค่าแฟคเตอร์เพื่อทำการเปรียบเทียบค่าเอาต์พุต (Output Scaling factor) เพื่อทำการ Normalization ค่าของเอาต์พุตให้เหมาะกับการใช้งาน โดยการเขียนโปรแกรมผ่าน Arduino IDE map ได้ Feedbcak = map(fed,240,1660,0,100); และ Setpoint = map(set,0,4095,0,100);

3.5.2 การทดสอบการทำงานของใบพัดมอเตอร์

ทดสอบเพื่อดูการทำงานของใบพัดมอเตอร์ที่เอาต์พุตค่าต่างๆ เพื่อหาช่วงการทำงานเริ่มต้นและช่วงสูงสุดการหมุนของใบพัดที่จะนำมาใช้ จะทำให้ใบพัดหมุนเร็วขึ้น โดยจะนำใบพัดมอเตอร์ต่อเข้ากับบอร์ดไมโครคอนโทรเลอร์ Arduino ESP 32 โดยสายที่ใช้ต่อจะเป็นสายสีส้มที่เป็นค่า PWM และ สายสีดำที่เป็นสายกราวด์ ดังรูปที่ 3.11

หลังจากการทดสอบจะพบว่าช่วงที่เหมาะสมกับการใช้งาน ค่าเอาต์พุตจะอยู่ในช่วง 0-125 และเมื่อมีการเปลี่ยนแปลงของค่า error ความเร็วของพัดลมจะเปลี่ยนแปลงตาม โดยค่า error มีความสัมพันธ์แบบผันตรงกับความเร็วมอเตอร์ กล่าวคือเมื่อค่า error เป็นลบความเร็วของมอเตอร์จะลดลงและหากค่า error เป็นบวกความเร็วของมอเตอร์จะหมุนเร็วขึ้น

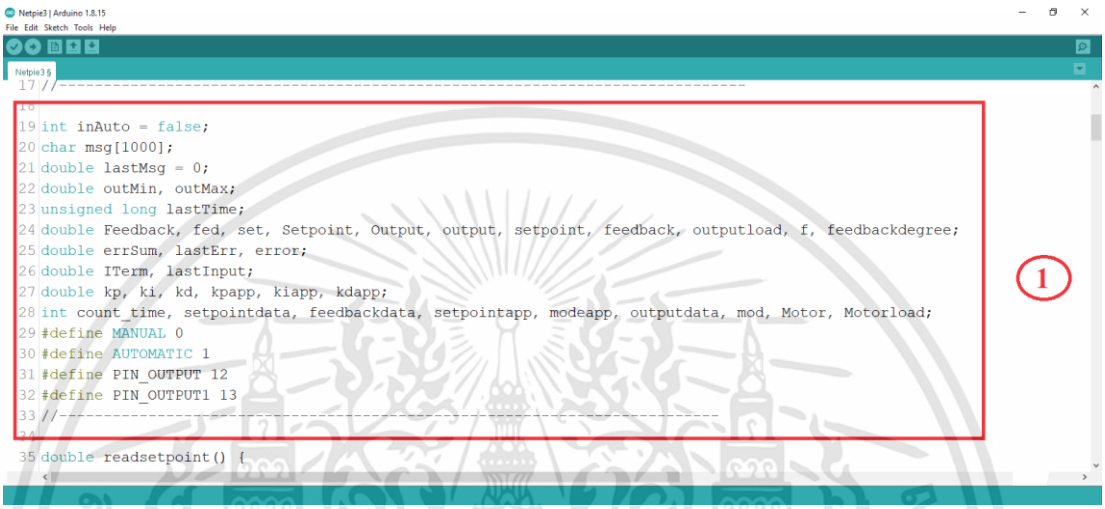
3.5.3 การเขียนโปรแกรม Arduino

ขั้นตอนนี้เป็นขั้นตอนการเขียนโปรแกรมเพื่อรับค่าจากเซนเซอร์มาประมวลผล และส่งค่าไปเพื่อควบคุมความเร็วในการหมุนของมอเตอร์

รูปที่ 3.14 หน้าต่างโปรแกรม Arduino (1)

จากรูปที่ 3.14 จะประกอบด้วย 2 ส่วน โดยส่วนที่ 1 คือ ส่วนของการเรียกใช้ไลบรารีที่จำเป็นต่อการเรียกใช้คำสั่งเพื่อเขียนคำสั่งระบบในการประมวลผล รับ-ส่งข้อมูล เชื่อมต่อ Wi-Fi และ

การเชื่อมต่อโปรโตคอล MQTT เพื่อเชื่อมต่อ IOT Platform NETPIE ที่ใช้เป็นตัวกลางในการส่งข้อมูลไปยังโทรศัพท์และคอมพิวเตอร์ และส่วนที่ 2 คือ คำสั่งที่ใช้ในการเชื่อมต่อ Wi-Fi ซึ่งในที่นี้คือตัวแปรที่มีชื่อว่า (ssid) และรหัสผ่าน (password) อีกทั้งในส่วนนี้ยังเป็นการเชื่อมต่อเซิร์ฟเวอร์ของ NETPIE โดยใช้ MQTT โดยใช้ตัวแปรเซิร์ฟเวอร์ที่มีชื่อว่า (mqtt_server) พอร์ตในการเชื่อมต่อ (mqtt_port) ผู้ใช้ (mqtt_Client) ชื่อ (mqtt_username) และรหัสผ่าน (mqtt_password)



รูปที่ 3.15 หน้าต่างโปรแกรม Arduino (2)

จากรูปที่ 3.15 จะประกอบด้วย 1 ส่วน คือส่วนสำหรับการประกาศตัวแปรต่างๆ ที่ใช้ในการเขียนคำสั่งเพื่อควบคุมการทำงานของระบบ รับ-ส่งข้อมูลระหว่าง Arduino ESP32 กับ NETPIE

รูปที่ 3.16 หน้าต่างโปรแกรม Arduino (3)

จากรูปที่ 3.16 จะประกอบด้วย 1 ส่วน คือส่วนที่เป็นฟังก์ชันในการรับค่าจากเซนเซอร์ทั้ง 2 ตัว ได้แก่ Setpoint และ Feedback จากตัวระบบแล้วนำมาทำการ Normalization โดยใช้คำสั่ง map แล้วส่งค่าออกเพื่อใช้ในการคำนวณต่อไป

รูปที่ 3.17 หน้าต่างโปรแกรม Arduino (4)

จากรูปที่ 3.17 จะประกอบด้วย 1 ส่วน คือ ฟังก์ชัน Set ค่า PID คือจะทำการส่งค่า Kp, Ki และ Kd เข้าไปในฟังก์ชันและทำการนำตัวแปร Global เข้ามารับค่าเพื่อที่จะนำค่า Kp, Ki และ Kd ไปคำนวณหาค่า Output ที่เหมาะสมต่อไป

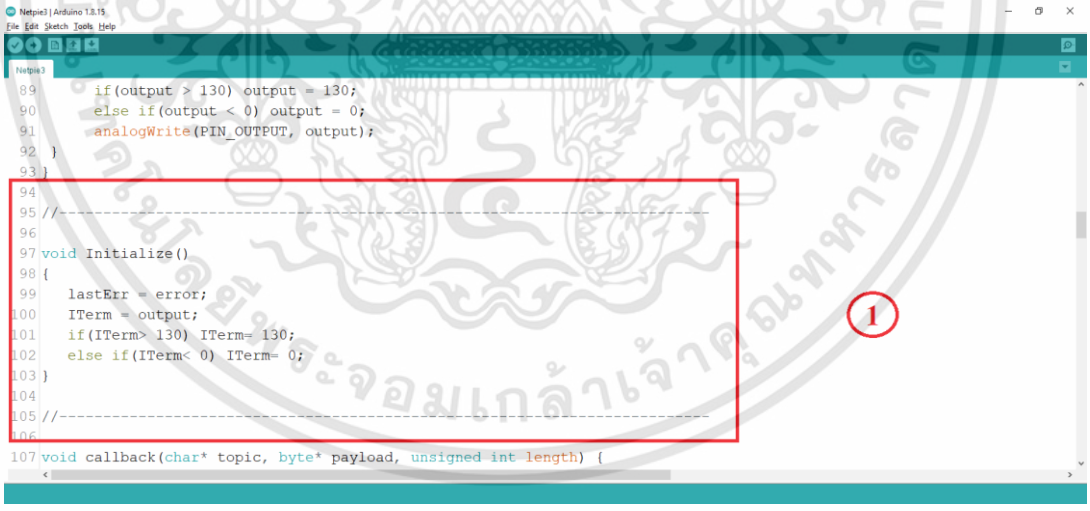
รูปที่ 3.18 หน้าต่างโปรแกรม Arduino (5)

จากรูปที่ 3.18 จะประกอบด้วย 1 ส่วน คือ ฟังก์ชันในการคำนวณหาค่า Output ของระบบที่เหมาะสมโดยจะคิดหาค่า error จาก Setpoint – Feedback แล้วนำค่า error ที่ได้มาคำนวณรวม

กับค่า Kp Ki และ Kd โดยมีสมการหาค่า Output คือ (kp*error) + (ki*errSum) + (kd*(error-lastErr))

รูปที่ 3.19 หน้าต่างโปรแกรม Arduino (6)

จากรูปที่ 3.19 จะประกอบด้วย 1 ส่วน คือ ฟังก์ชันในการควบคุมระบบว่าเป็น Manual หรือ Auto โดยในระบบ Manual สามารถควบคุมความเร็วมอเตอร์ได้ด้วยมือ ส่วนในระบบ Auto นั้นจะเข้าไปสู่ฟังก์ชัน Set ค่าเริ่มต้นของ พจน์ Ki เพื่อไม่เกิดการ error ในระบบ



รูปที่ 3.20 หน้าต่างโปรแกรม Arduino (7)

จากรูปที่ 3.20 ประกอบด้วย 1 ส่วน คือ ฟังก์ชันการทำงานเริ่มต้นของพจน์ Ki เมื่อมีการเปลี่ยนโหมดจากระบบ Manual มาเป็นระบบ Auto เพื่อไม่ให้การทำงานของระบบนั้นเกิดความผิดพลาดจึงต้องทำให้ค่าของพจน์ Ki นั้นเป็นค่าเริ่มต้น

รูปที่ 3.21 หน้าต่างโปรแกรม Arduino (8)

จากรูปที่ 3.21 ประกอบด้วย 2 ส่วน โดยส่วนที่ 1 และ 2 อยู่ในฟังก์ชันเดียวกันคือฟังก์ชันที่รับข้อมูลจาก NETPIE โดยส่วนที่ 1 คือ การแปลงข้อมูลที่ได้รับจาก NETPIE ซึ่งเป็นข้อมูลที่อยู่ในรูปของ JASON แปลงเป็น string โดยใส่ในตัวแปร message เพื่อนำไปใช้งานต่อ และส่วนที่ 2 คือ เมื่อมีข้อมูลที่มีหัวข้อว่า status ที่ส่งมาจาก NETPIE ก็จะทำการกำหนดค่าให้กับตัวแปร mod โดยเมื่อข้อมูลที่ส่งมานั้นมีข้อความว่า “on” ตัวแปรที่มีชื่อว่า modก็จะมีค่าเท่ากับ 1 และอัพเดทบนฐานข้อมูลของ NETPIE และ เมื่อมีข้อมูลที่ส่งมานั้นมีข้อความว่า “off” ตัวแปรที่มีชื่อว่า mod ก็จะมีค่าเท่ากับ 0 และอัพเดทบนฐานข้อมูลของ NETPIE

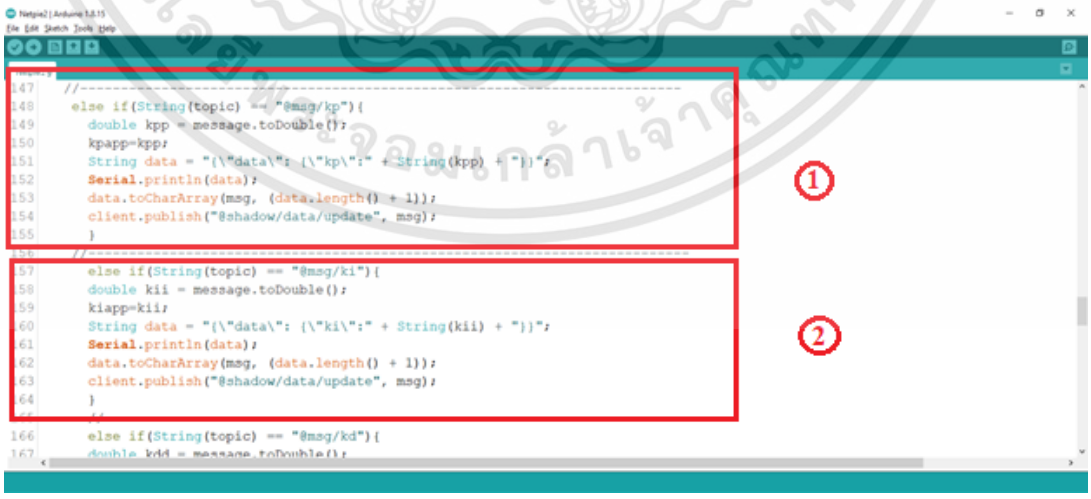
รูปที่ 3.22 หน้าต่างโปรแกรม Arduino (9)

จากรูปที่ 3.22 ประกอบด้วย 2 ส่วน โดยส่วนที่ 1 และ 2 อยู่ในฟังก์ชันเดียวกันคือฟังก์ชันที่รับข้อมูลจาก NETPIE โดยส่วนที่ 1 คือ เมื่อมีข้อมูลที่มีหัวข้อว่า input จาก NETPIE (Setpoint slide bar) ก็จะทำการแปลงข้อความ message ซึ่งเป็น string แปลงเป็น integer เพื่อใช้ในการคำนวณ

ต่อไปและแปลงข้อมูลที่ได้ให้อยู่ในรูปของ Jason เพื่อทำการอัพเดทข้อมูลในฐานข้อมูลของ NETPIE และส่วนที่ 2 คือ เมื่อมีข้อมูลที่มีหัวข้อว่า motor จาก NETPIE (Motor control slide bar) ก็จะทำการแปลงข้อความ message ซึ่งเป็น string แปลงเป็น integer และทำการ map ให้อยู่ในการทำงานของมอเตอร์แล้วนำไปขับมอเตอร์และแปลงข้อมูลที่ได้ให้อยู่ในรูปของ Jason เพื่อทำการอัพเดทข้อมูลในฐานข้อมูลของ NETPIE

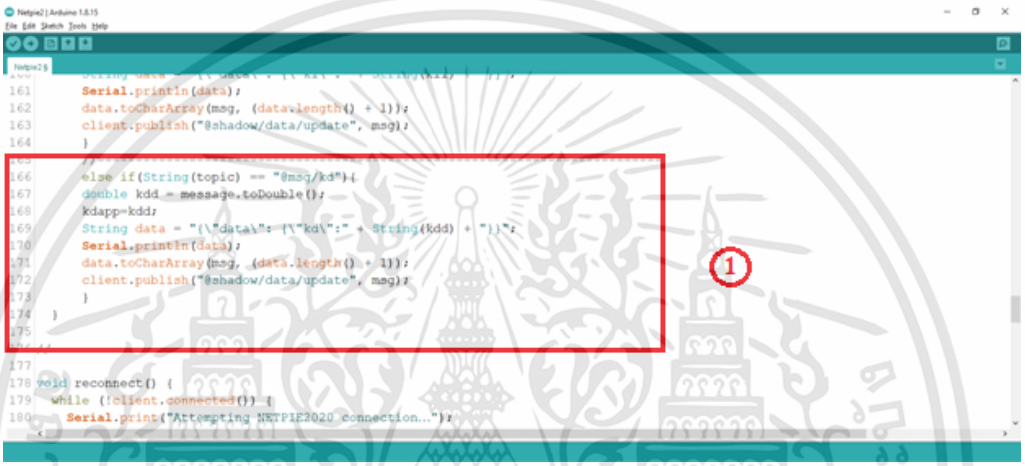
รูปที่ 3.23 หน้าต่างโปรแกรม Arduino (10)

จากรูปที่ 3.23 ประกอบด้วย 1 ส่วน คือ เมื่อมีข้อมูลที่มีหัวข้อว่า motorload จาก NETPIE (Motor control load slide bar) ก็จะทำการแปลงข้อความ message ซึ่งเป็น string แปลงเป็น integer และทำการ map ให้อยู่ในการทำงานของมอเตอร์แล้วนำไปขับมอเตอร์และทำการแปลงข้อมูลที่ได้ให้อยู่ในรูปของ Jason เพื่อทำการอัพเดทข้อมูลในฐานข้อมูลของ NETPIE



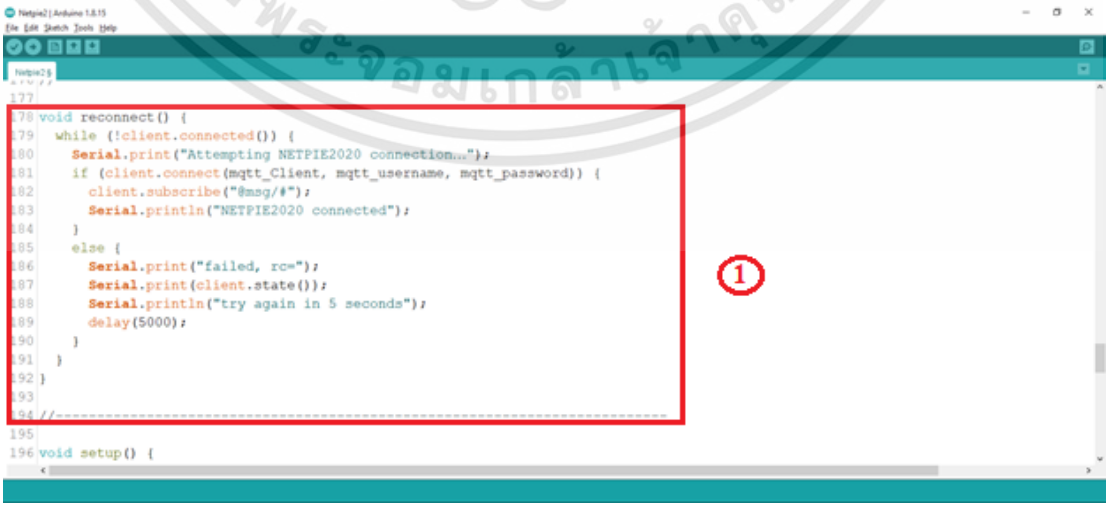
รูปที่ 3.24 หน้าต่างโปรแกรม Arduino (11)

จากรูปที่ 3.24 ประกอบด้วย 2 ส่วน โดยส่วนที่ 1 และ 2 อยู่ในฟังก์ชันเดียวกันคือฟังก์ชันรับข้อมูลจาก NETPIE โดยส่วนที่ 1 คือ เมื่อมีข้อมูลที่มีหัวข้อว่า kp จาก NETPIE (Kp Slide bar) ก็จะทำการแปลงข้อความ message ซึ่งเป็น string แปลงเป็น Double เพื่อใช้ในการคำนวณต่อไปและแปลงข้อมูลที่ได้ให้อยู่ในรูปของ Jason เพื่อทำการอัพเดทข้อมูลในฐานข้อมูลของ NETPIE และส่วนที่ 2 คือ เมื่อมีข้อมูลที่มีหัวข้อว่า ki จาก NETPIE (Ki Slide bar) ก็จะทำการแปลงข้อความ message ซึ่งเป็น string แปลงเป็น Double เพื่อใช้ในการคำนวณต่อไปและแปลงข้อมูลที่ได้ให้อยู่ในรูปของ Jason เพื่อทำการอัพเดทข้อมูลในฐานข้อมูลของ NETPIE



รูปที่ 3.25 หน้าต่างโปรแกรม Arduino (12)

จากรูปที่ 3.25 ประกอบด้วย 1 ส่วน คือ ส่วนหนึ่งที่อยู่ในฟังก์ชันรับข้อมูลเมื่อมีข้อมูลที่มีหัวข้อว่า kd จาก NETPIE (Kd Slide bar) ก็จะทำการแปลงข้อความ message ซึ่งเป็น string แปลงเป็น Double เพื่อใช้ในการคำนวณต่อไปและแปลงข้อมูลที่ได้ให้อยู่ในรูปของ Jason เพื่อทำการอัพเดทข้อมูลในฐานข้อมูลของ NETPIE



รูปที่ 3.26 หน้าต่างโปรแกรม Arduino (13)

จากรูปที่ 3.26 ประกอบด้วย 1 ส่วน คือ ฟังก์ชันที่ในการเชื่อมต่อ IOT Platform NETPIE และเริ่มรับข้อมูลจาก NETPIE โดยใช้คำสั่ง client.subscribe("@msg/#");

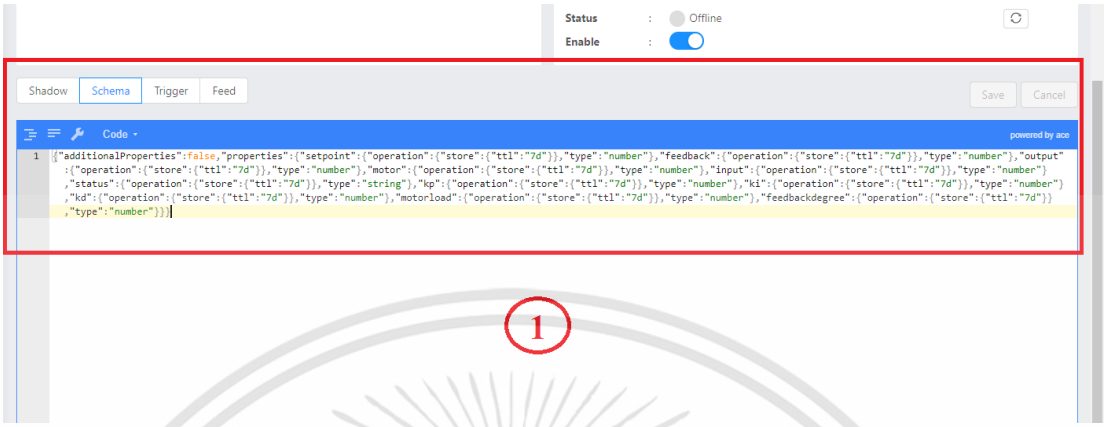
รูปที่ 3.27 หน้าต่างโปรแกรม Arduino (14)

จากรูปที่ 3.27 ประกอบด้วย 1 ส่วน คือ ฟังก์ชัน Setup เป็นฟังก์ชันเริ่มต้นของโปรแกรม โดยจะมีหน้าที่ในการเชื่อมต่อ WIFI และเมื่อทำการเชื่อมต่อ WIFI สำเร็จก็จะทำการเชื่อมต่อกับเซิฟเวอร์ของ NETPIE และรับข้อมูลจาก NETPIE โดยฟังก์ชัน callback

รูปที่ 3.28 หน้าต่างโปรแกรม Arduino (15)

จากรูปที่ 3.28 ประกอบด้วย 2 ส่วน โดยส่วนที่ 1 และ 2 จะอยู่ในฟังก์ชันเดียวกันนั้นคือ ฟังก์ชัน loop โดยในส่วนที่ 1 จะเป็นส่วนที่จะทำการตรวจสอบว่ามีการเชื่อมต่อไปยัง NETPIE หรือไม่และเป็นส่วนที่ใช้ในการทำงานของระบบและส่วนที่สองจะเป็นการส่งข้อมูลอัพเดทเกี่ยวกับตัว

แปร Setpoint Feedbeack และ Output ไปยังฐานข้อมูลของ NETPIE และแสดงผลบนคอมพิวเตอร์หรือโทรศัพท์มือถือ



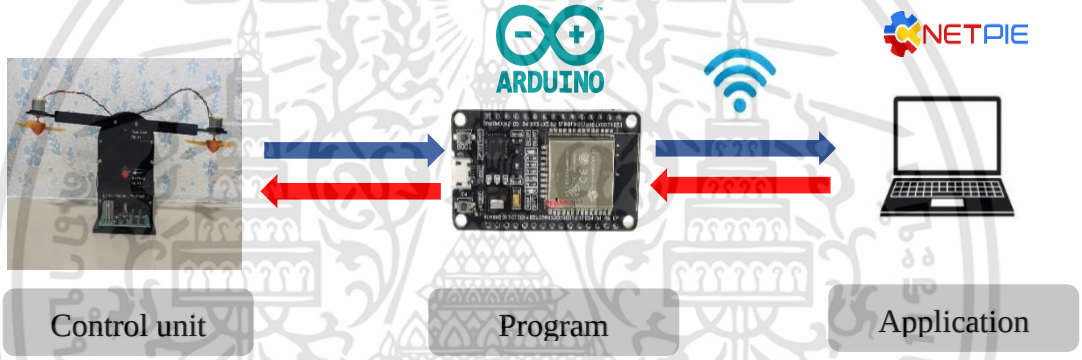
รูปที่ 3.29 ฐานเก็บข้อมูลบน NETPIE

จากรูปที่ 3.29 ประกอบด้วย 1 ส่วน คือ ส่วนที่ใช้เป็นฐานเก็บข้อมูลของ NETPIE ซึ่งเก็บข้อมูลของค่าสถานะของโหมดการทำงาน ค่าของ Setpoint, มุมของ Setpoint, Feedback, มุมของ Feedback, Kp, Ki, Kd, motor, motor ที่เป็นโหลด และ ค่า output ของระบบ

บทที่ 4
ผลการดำเนินงาน

4.1 แผนผังการทำงานของระบบควบคุมคานสมดุลแบบไร้สาย

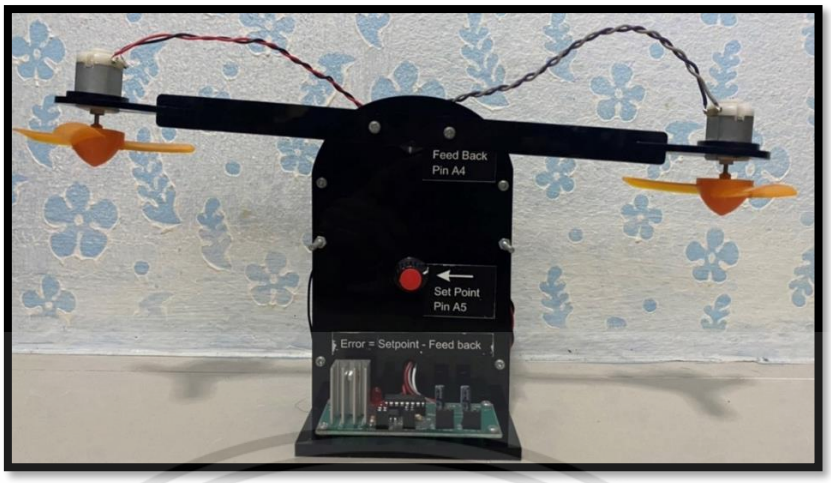
ชุดควบคุมคานสมดุลจะสื่อสารกับบอร์ด Arduino ESP32 ซึ่งทางบอร์ด Arduino ESP32 จำเป็นต้องมีการเขียนโค้ดเพื่อที่จะอัปโหลดโปรแกรมโค้ดใส่เข้าไปในตัวบอร์ด และผู้ใช้ต้องมีแอปพลิเคชัน NETPIE ถ้าเป็นการเชื่อมต่อบนโทรศัพท์ แต่ถ้าเป็นการเชื่อมต่อบนคอมพิวเตอร์จะเข้าทางเว็บไซต์ของ NETPIE โดยตรง เมื่อทางผู้ใช้สั่งการทำงานผ่านแอปพลิเคชัน ข้อมูลชุดคำสั่งจะส่งไปยังบอร์ด Arduino ESP32 ที่มีการเชื่อมต่อ Wi-Fi จากนั้นบอร์ด Arduino ESP32 จะทำการสั่งการทำงานของชุดระบบควบคุมคานสมดุลทันที



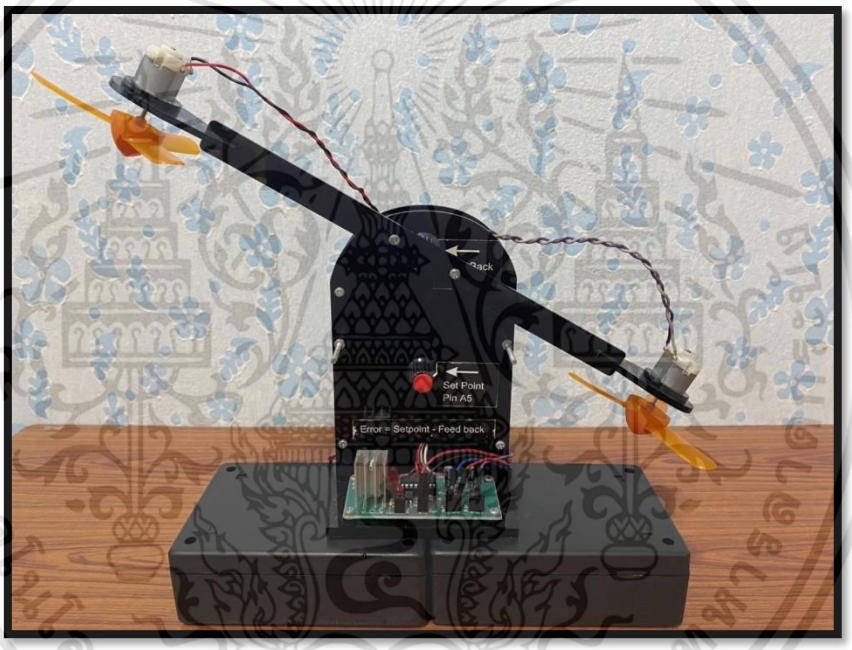
รูปที่ 4.1 แผนผังการทำงานของระบบควบคุมคานสมดุลแบบไร้สาย

4.2 ชุดอุปกรณ์ของระบบควบคุมคานสมดุลแบบไร้สาย

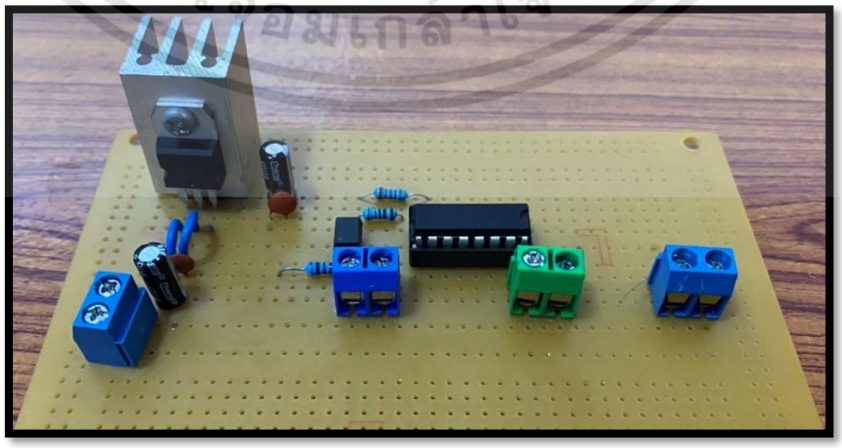
เนื่องจากตัวระบบควบคุมคานสมดุลมีวงจรอิเล็กทรอนิกส์สำหรับมอเตอร์ 1 ชุดมาเป็นที่เรียบร้อยแล้วจึงทำการเพิ่มวงจรอิเล็กทรอนิกส์สำหรับมอเตอร์อีก 1 ชุดเพิ่มเข้ามา แล้วทำการเชื่อมต่อสายจากตัวชุดระบบควบคุมคานสมดุลกับ Arduino ESP32 ดังรูปที่ 4.2 ระบบควบคุมคานสมดุลแบบไร้สายที่สมบูรณ์ ดังรูปที่ 4.3 และ วงจรอิเล็กทรอนิกส์ ดังรูปที่ 4.4 และ รูปที่ 4.5



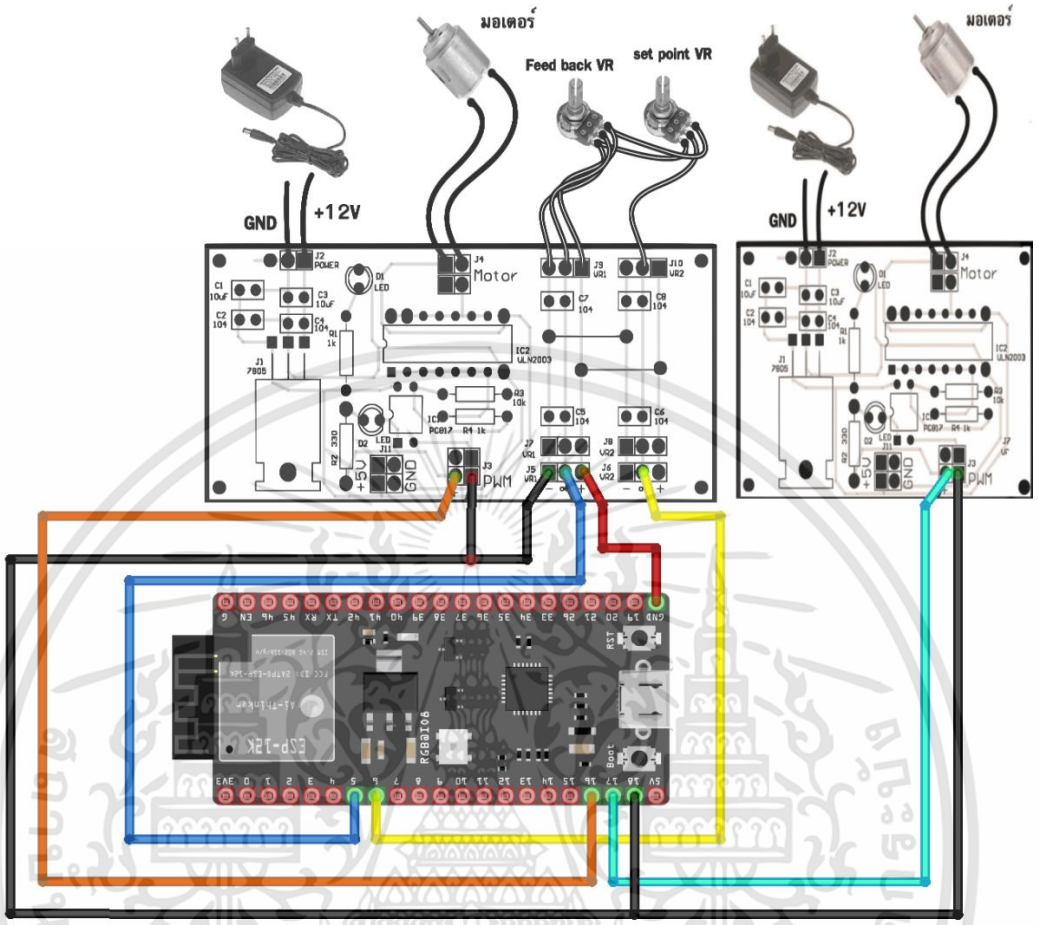
รูปที่ 4.2 ชุดอุปกรณ์ระบบควบคุมคานสมดุลแบบไร้สาย



รูปที่ 4.3 ระบบควบคุมคานสมดุลแบบไร้สายที่สมบูรณ์



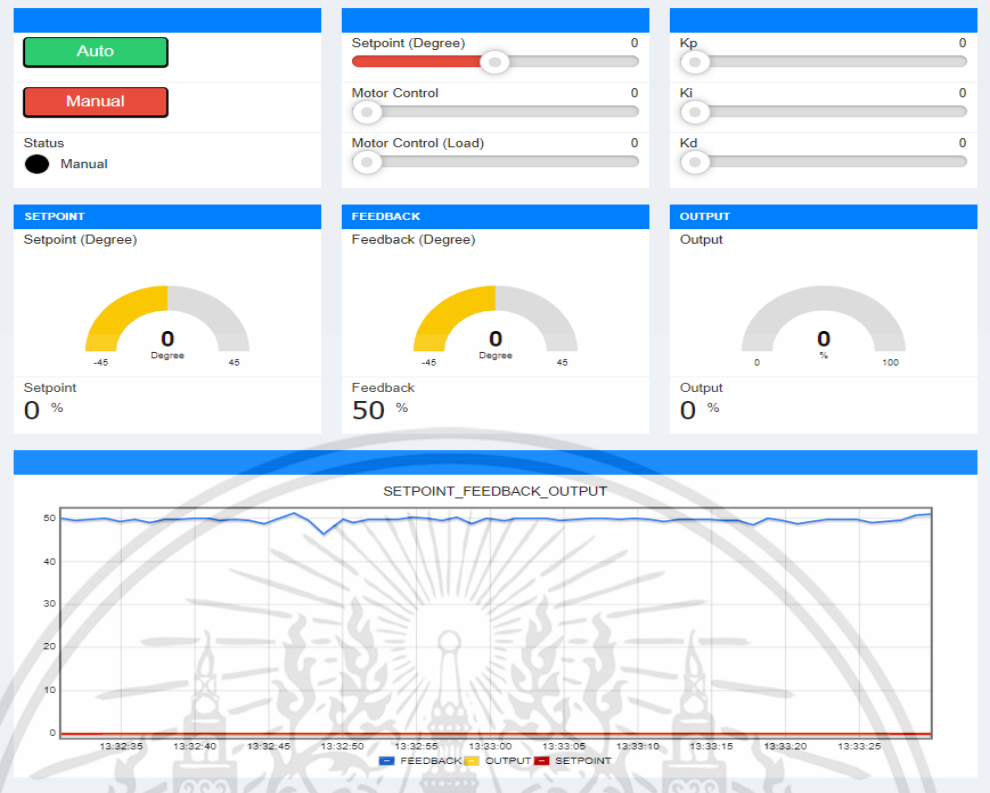
รูปที่ 4.4 วงจรอิเล็กทรอนิกส์ภายในชุดระบบควบคุมคานสมดุลแบบไร้สาย



รูปที่ 4.5 วงจรอิเล็กทรอนิกส์ภายในชุดระบบควบคุมคานสมดุลแบบไร้สาย
ที่เชื่อมต่อกับ Arduino ESP32

4.3 ผลการออกแบบหน้าแสดงผลผ่าน NETPIE

ส่วนของหน้าจอแสดงผลในส่วนนี้จะแสดงให้เห็นถึงองค์ประกอบของระบบซึ่งแสดงค่าผลลัพธ์ต่างๆ ได้แก่ กราฟแสดงผลและเกจแสดงผล ประกอบไปด้วยค่า Setpoint, Feedback และ Output ส่วนที่เป็น Slide bar สำหรับเพิ่ม-ลดเพื่อเปลี่ยนค่า Setpoint, Kp, Ki, Kd Controller และ Controller (load) รวมถึงส่วนสำหรับเลือกโหมดการทำงานของระบบซึ่งมีสองโหมดคือโหมดอัตโนมัติและโหมดปรับด้วยมือ ดังรูปที่ 4.5

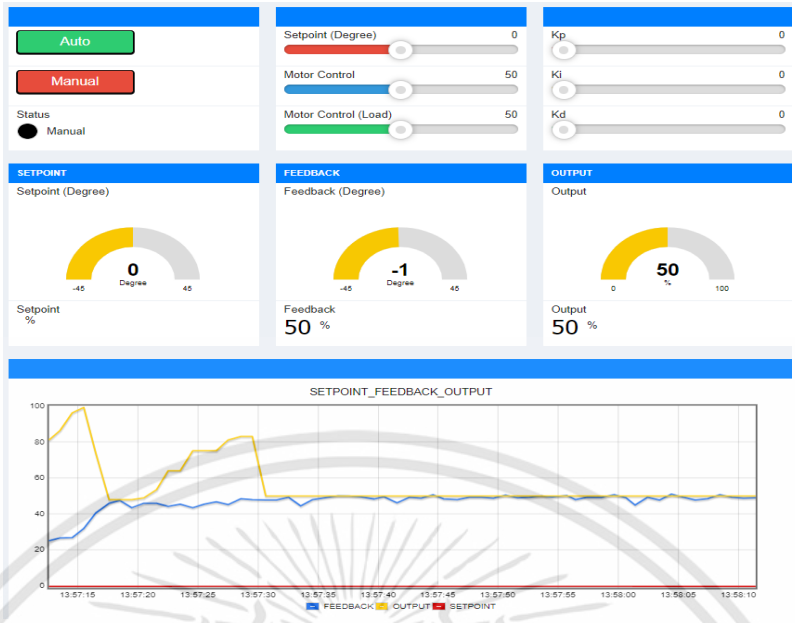


รูปที่ 4.6 หน้าแสดงผลบน NETPIE

4.4 การทดสอบการทำงาน

4.4.1 ทดสอบการรับ-ส่งข้อมูล และการแสดงผลผ่าน NETPIE โหมดการปรับด้วยมือ (Manual mode)

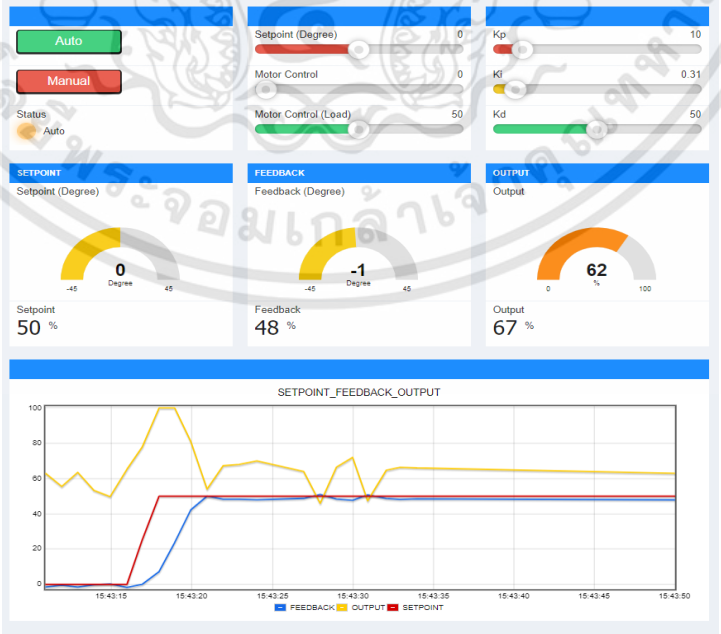
ทดสอบการรับ-ส่งข้อมูล และการแสดงผลระหว่างระบบควบคุมคานสมดุลแบบไร้สายและ NETPIE โหมดการปรับด้วยมือ (Manual mode) ซึ่งส่วนนี้จะแสดงให้เห็นถึงการทำงานของระบบและการควบคุมผ่าน NETPIE ในโหมดการปรับด้วยมือ โดยโหมดควบคุมนี้สามารถทำการปรับค่าของคอนโทรลเลอร์ได้ตามที่ผู้ใช้งานต้องการโดยจะส่งผลทำให้มอเตอร์นั้นหมุนได้ความเร็วที่ผู้ใช้งานปรับค่า และสามารถแสดงค่าต่างๆ ผ่าน NETPIE ดังรูปที่ 4.6



รูปที่ 4.7 การควบคุมระบบผ่าน NETPIE โหมดการปรับมือ

4.4.2 ทดสอบการรับ-ส่งข้อมูล และการแสดงผลผ่าน NETPIE โหมดอัตโนมัติ (Automation mode)

การทดสอบการรับ-ส่งข้อมูล และการแสดงผลระหว่างระบบควบคุมคานสมดุลแบบไร้สาย และ NETPIE ในโหมดอัตโนมัติ (Automation mode) ซึ่งส่วนนี้จะแสดงให้เห็นถึงการทำงานของระบบและการควบคุมผ่าน NETPIE โดยการปรับค่าพารามิเตอร์ Setpoint, Kp, Ki และ Kd ได้ตามที่ผู้ใช้งานต้องการ และแสดงค่าต่างๆ ผ่าน NETPIE ดังรูป 4.7



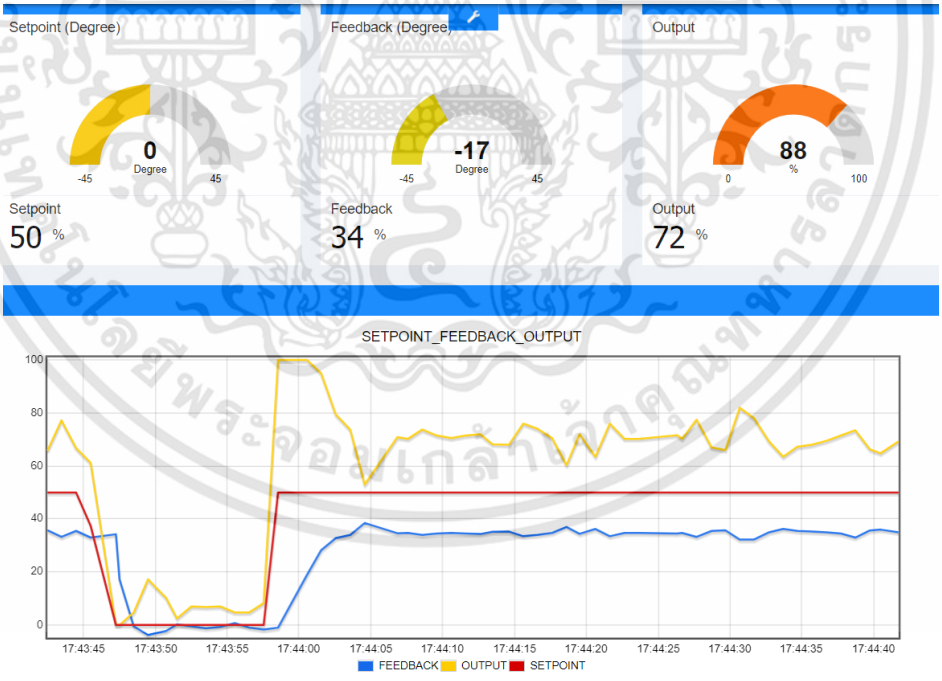
รูปที่ 4.8 การควบคุมระบบผ่าน NETPIE โหมดอัตโนมัติ

4.5 ผลการทดสอบการควบคุมระบบ

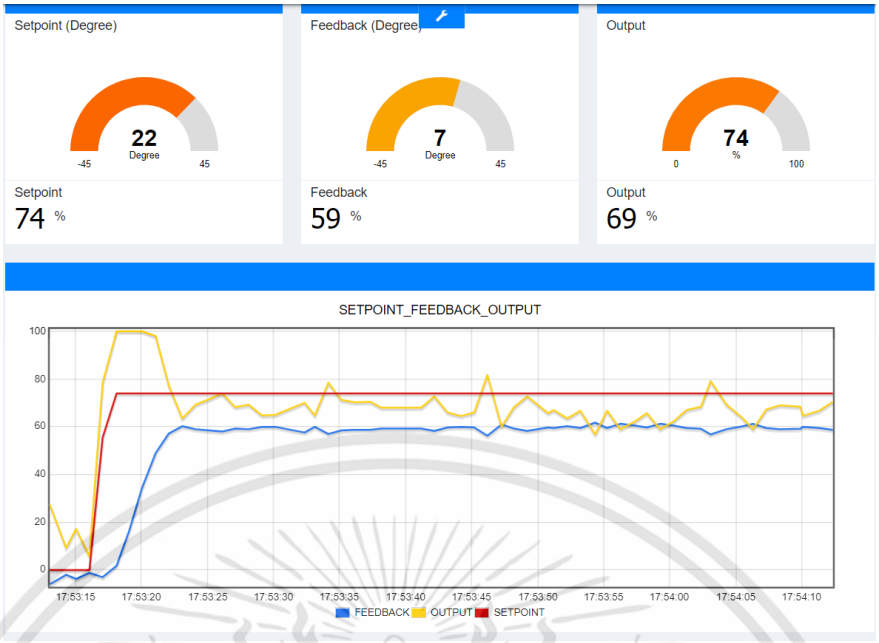
การทดสอบการควบคุมระบบ ซึ่งใช้วิธีการปรับจูนแบบ Trial & error close-loop tuning โดยวิธีนี้เป็นวิธีการทดลองป้อนค่า Kp, Ki, และ Kd เข้าระบบ แล้วสังเกตการเปลี่ยนแปลงของระบบ และหาค่าทำให้ระบบสมดุลมากที่สุด โดยมีขั้นตอนการปรับจูน ตามหัวข้อที่ 2.7.5 การปรับจูนค่าคงที่ของตัวควบคุม

4.5.1 ผลการทดสอบการควบคุมระบบด้วยตัวควบคุมแบบ P Controller

ผลการควบคุมแบบ P เมื่อมีการเปลี่ยนแปลงค่า Setpoint จาก 0 % เป็น 50 % และเปลี่ยนแปลงจาก 0% เป็น 74 % โดยมีค่า Kp = 6 และมีโหลดอยู่ที่ 50% ทันทีที่มีการเปลี่ยนแปลงค่า Setpoint สัญญาณ PWM จะเพิ่มขึ้นแทบจะในทันทีส่งผลทำให้ มอเตอร์ตอบสนองด้วยการหมุนใบพัดเร็วขึ้น ทำให้เกิดแรงยกเกิดการเปลี่ยนตำแหน่งของแกนใบพัดลู่เข้าสู่ Setpoint แต่ในขณะเดียวกันเมื่อตำแหน่งแกนใบพัดเข้าใกล้ Setpoint ทำให้ค่า error ลดลง และยังทำให้ค่า PWM ลดลง ส่งผลให้มอเตอร์หมุนช้าลง สูญเสียแรงยกจนเกิดค่าความผิดพลาด (Steady state error, ess) แต่ตัวควบคุมจะคำนวณค่าให้ PWM สามารถเลี้ยงแกนใบพัดให้อยู่ในตำแหน่งที่สมดุลระหว่างแรงยกและน้ำหนักแกน ดังรูปที่ 4.8 และ 4.9 ตามลำดับ

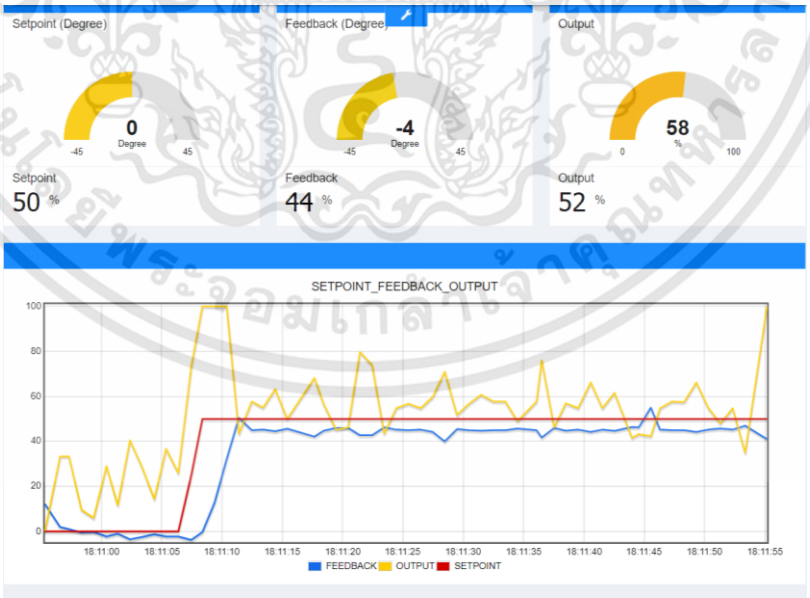


รูปที่ 4.9 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ P controller ที่ Kp = 6 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 50 %

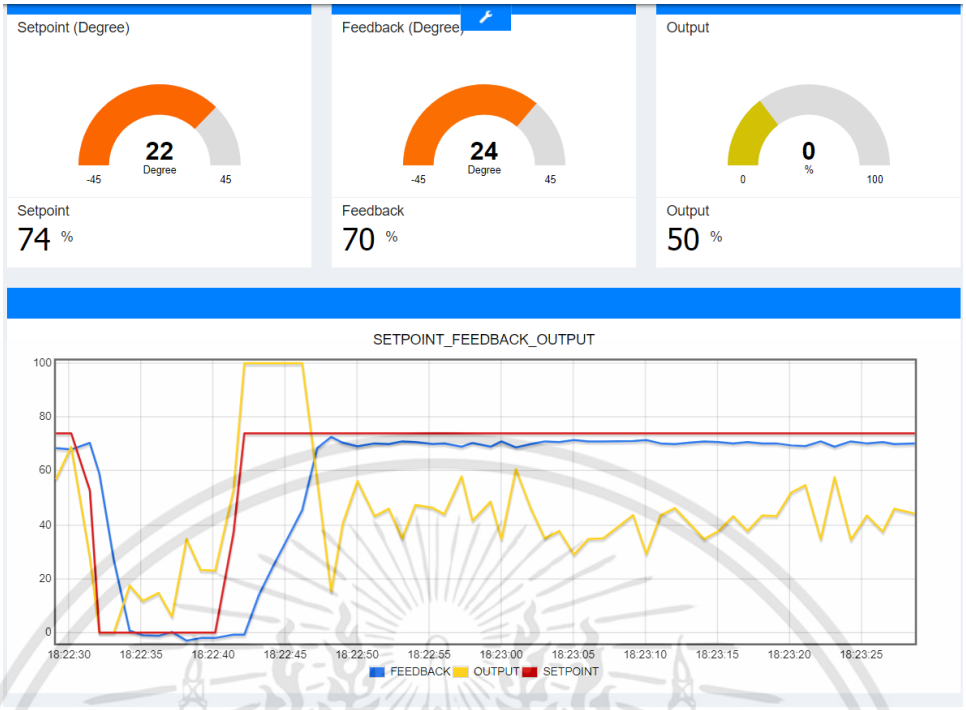


รูปที่ 4.10 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ P controller ที่ Kp = 6 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 74 %

และเมื่อทำการเพิ่มค่า Kp จาก 6 เป็น 15 ทำให้ผลตอบสนองวงปิดของระบบลดค่าความผิดพลาด (Steady state error, ess) ช่วยเพิ่มความเร็วของผลตอบสนองในช่วงแรก (Rise time, tr) มีการเกิดค่าพุ่งเกิน (Overshoots, Mp) และมีการเกิดดความถี่ในการแกว่ง (Oscillation) ดังรูปที่ 4.10 และ 4.11 ตามลำดับ



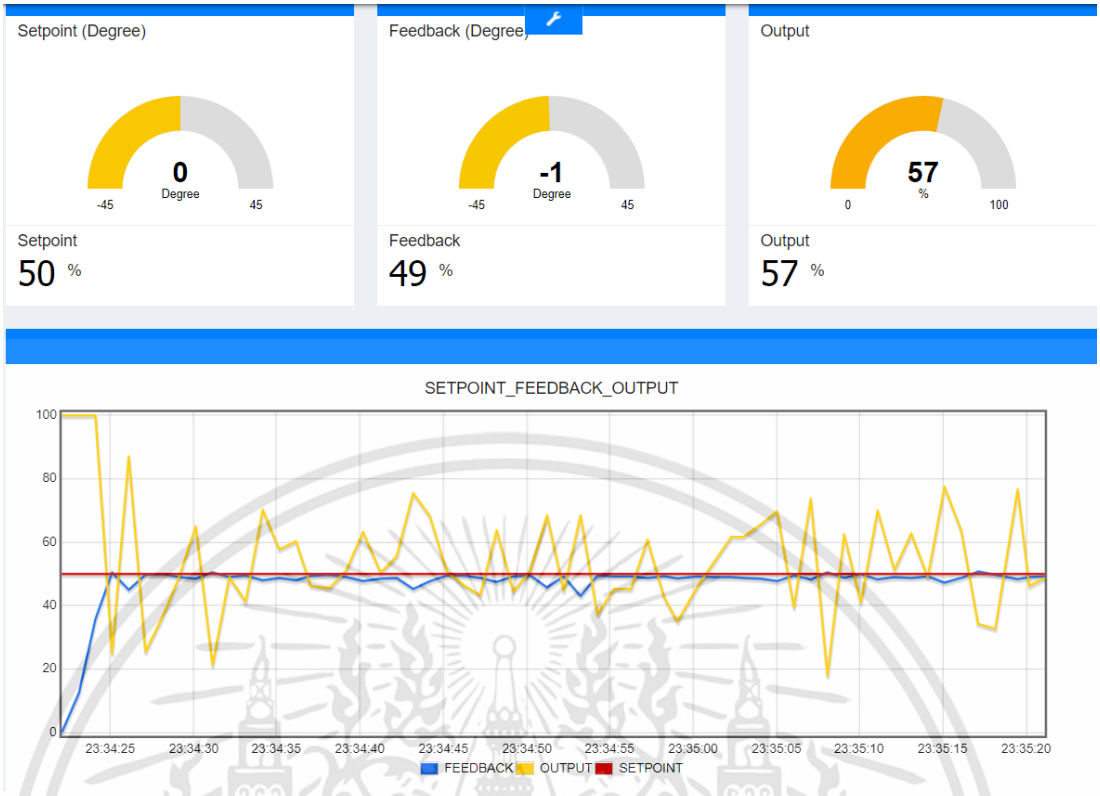
รูปที่ 4.11 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ P controller ที่ Kp = 15 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 50 %



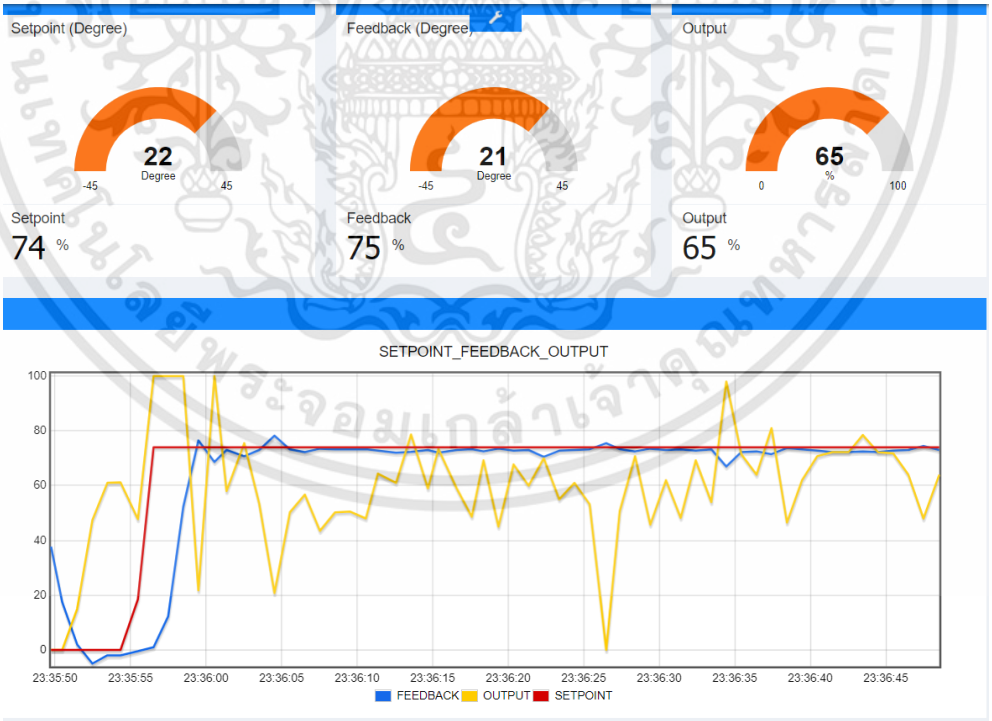
รูปที่ 4.12 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ P controller ที่ Kp = 15 โหลดอยู่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 74 %

4.5.2 ผลการทดสอบการควบคุมระบบด้วยตัวควบคุมแบบ PI Controller

ผลการควบคุมแบบ PI เมื่อมีการเปลี่ยนแปลงค่า Setpoint จาก 0 % เป็น 50 % และเปลี่ยนแปลงจาก 0% เป็น 74 % โดยมีค่า Kp = 40 Ki = 2 โหลดอยู่ที่ 50 % ทันทีที่มีการเปลี่ยนแปลงค่า Setpoint สัญญาณ PWM จะเพิ่มขึ้นแทบจะในทันทีส่งผลให้มอเตอร์ตอบสนองด้วยการหมุนใบพัดเร็วขึ้น ทำให้เกิดแรงยกเกิดการเปลี่ยนตำแหน่งของแกนใบพัดลู่เข้าสู่ Setpoint โดยผลตอบสนองวงปิดของระบบทำงานเร็วขึ้นในช่วงแรก (Rise time, tr) เป็นเพราะเทอม I ในสมการเป็นการรวมกันของค่า error ทำให้ในการทำงานช่วงแรกเกิดค่าพุ่งเกินเพิ่มขึ้น (Overshoot, Mp) เกิดความถี่ในการแกว่ง (Oscillation) และสามารถทำให้ตำแหน่งของแกนใบพัดลู่เข้าสู่ Setpoint โดยที่ไม่เกิดค่าความผิดพลาดได้ (Steady state error, ess) เนื่องเป็นการรวมกันของค่า error ในเทอม I จึงสามารถช่วยในการกำจัดค่าความผิดพลาดนี้ได้ ดังรูป 4.12 และ 4.13 ตามลำดับ



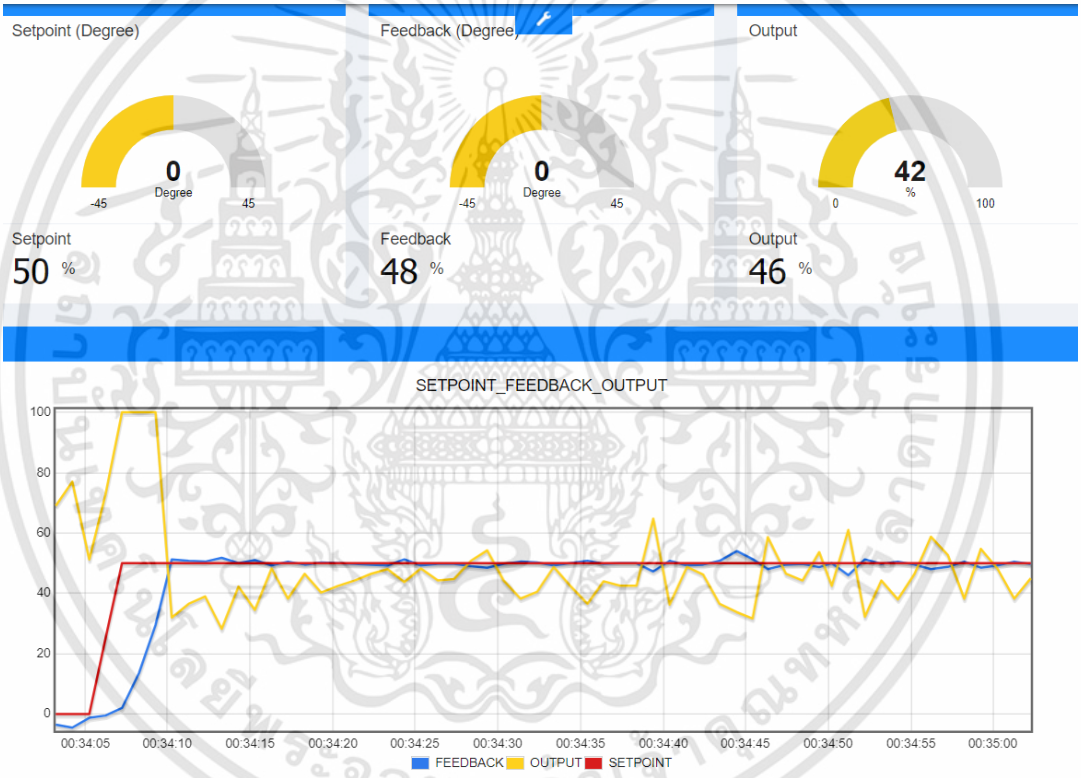
รูปที่ 4.13 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ PI controller ที่ Kp = 40 Ki = 2 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 50 %



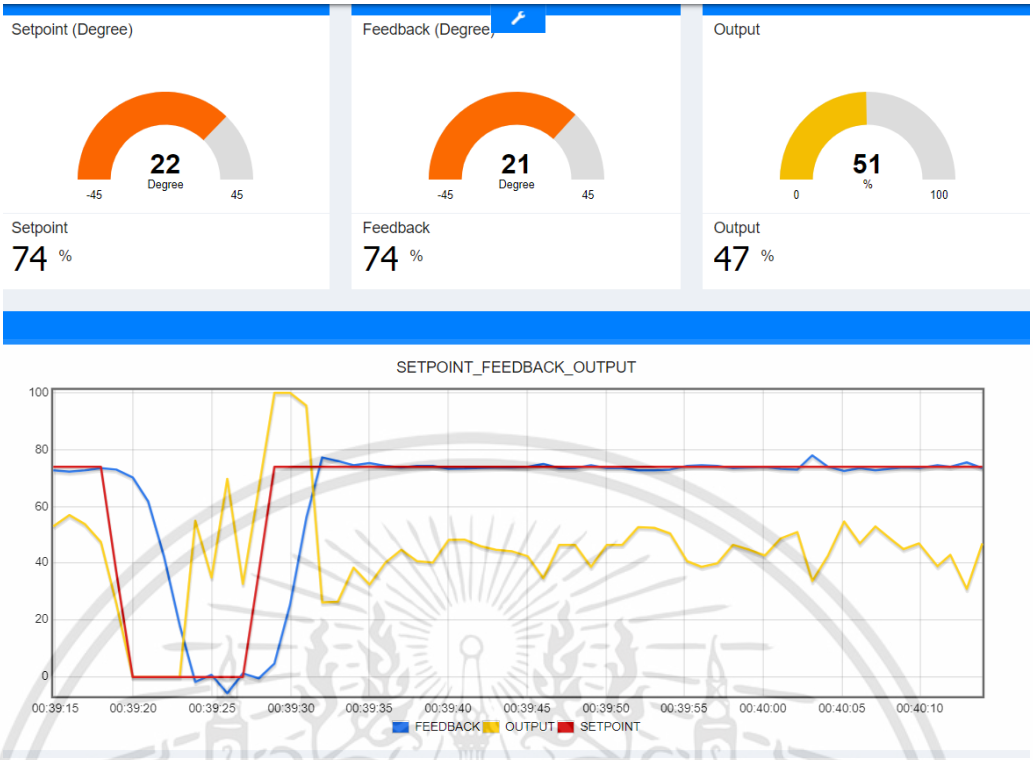
รูปที่ 4.14 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ PI controller ที่ Kp = 40 Ki = 2 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 74 %

4.5.3 ผลการทดสอบการควบคุมระบบด้วยตัวควบคุมแบบ PID Controller

ผลการควบคุมแบบ PID เมื่อมีการเปลี่ยนแปลงค่า Setpoint จาก 0 % เป็น 50 % และเปลี่ยนแปลงจาก 0% เป็น 74 % โดยมีค่า Kp = 10 Ki = 0.12 และ Kd = 40 โหลดอยู่ที่ 50 % ทันทีที่มีการเปลี่ยนแปลงค่า Setpoint สัญญาณ PWM จะเพิ่มขึ้นแทบจะในทันทีส่งผลให้มอเตอร์ตอบสนองด้วยการหมุนใบพัดเร็วขึ้น ทำให้เกิดแรงยกเกิดการเปลี่ยนตำแหน่งของแกนใบพัดลู่เข้าสู่ Setpoint โดยผลตอบสนองวงปิดของระบบทำงานถึงจุดคงที่ (Steady state) เร็วขึ้น และเกิดค่าพุ่งเกิน (Overshoots, Mp) ที่ลดลงเป็นผลมาจากการเพิ่มเทอม D เข้าไปในระบบเพราะเทอม D เป็นการนำค่า error ของเวลาปัจจุบันลบด้วยเวลาก่อนหน้าทำให้สามารถควบคุมค่าพุ่งเกินและยังทำให้ setting time ลดลง ดังรูปที่ 4.14 และ 4.15 ตามลำดับ



รูปที่ 4.15 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ PID controller ที่ Kp = 10 Ki = 0.12 Kd = 40 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 50 %



รูปที่ 4.16 ผลตอบสนองของระบบด้วยตัวควบคุมแบบ PID controller ที่ Kp = 10 Ki = 0.12 Kd = 40 โหลดอยู่ที่ 50 % Setpoint เปลี่ยนแปลงจาก 0 % เป็น 74 %

จากผลการทสอบ ผู้จัดทำได้เลือกใช้ตัวควบคุมแบบ PID controller ให้กับระบบควบคุมคานสมดุล เนื่องจากตัวควบคุมแบบ PID controller เป็นการรวมข้อดีของการควบคุมทุกรูปแบบทั้งการลดค่าพุ่งเกิน (Overshoot, Mp) การลด setting time การลดค่าความผิดพลาด (Steady state error, ess) ให้ใกล้เคียง 0 มากที่สุด การเพิ่มความเร็วการตอบสนองช่วงแรก (Rise time) และมีการลดความถี่ในการแกว่ง (Oscillation) ตัวควบคุมแบบ PID controller จึงเหมาะกับระบบควบคุมคานสมดุล ทั้งนี้การเลือกใช้ชนิดของตัวควบคุมขึ้นอยู่กับการเลือกค่าพารามิเตอร์ให้มีความเหมาะสมกับระบบ

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

จากการดำเนินงานตามขั้นตอนที่ได้กล่าวมา เริ่มจากขั้นตอนของการศึกษาทฤษฎีและหลักการพร้อมทำความเข้าใจกับขอบเขตของโครงงาน ดังนั้นจึงได้แบ่งส่วนของการดำเนินงานเป็นทั้งหมด 3 ส่วน คือ

1. ส่วนของงานทางด้านฮาร์ดแวร์ ในส่วนนี้ได้ศึกษาถึงหลักการทำงานของตัวระบบควบคุมคานสมดุล บอร์ดไมโครคอนโทรลเลอร์ วงจรอิเล็กทรอนิกส์ และอุปกรณ์ทางด้านฮาร์ดแวร์ต่างๆ รวมถึงการออกแบบชิ้นงานที่ให้ความสอดคล้องกับความต้องการ และวัตถุประสงค์ให้มากที่สุด

2. ส่วนของงานทางด้านซอร์ฟแวร์ เป็นส่วนของการเขียนโปรแกรมเพื่อทำการสื่อสารระหว่าง NETPIE และ ชุดอุปกรณ์ควบคุม ด้วยโปรแกรม Arduino IDE และเขียนฐานข้อมูลบน NETPIE และทำการออกแบบหน้าจอแสดงผลการใช้งานบน NETPIE ในการติดต่อระหว่างผู้ใช้กับชุดอุปกรณ์ควบคุมเพื่อให้ผู้ใช้เข้าใจและสามารถใช้งานได้โดยง่าย

3. เมื่อนำชุดอุปกรณ์มาทำการทดสอบด้วยการควบคุมการทำงานของระบบผ่าน NETPIE ด้วยคอมพิวเตอร์ พบว่าชุดอุปกรณ์สามารถควบคุมการทำงานได้จริง โดยบอร์ด Arduino ESP32 สามารถเชื่อมต่อ Wi-Fi ได้และรับ-ส่งข้อมูลได้แบบ Real-Time อีกทั้งยังเป็นตัวกลางในการสื่อสารระหว่างชุดอุปกรณ์ควบคุมและ NETPIE ได้

5.2 ปัญหาและอุปสรรคในการดำเนินงาน

1. มอเตอร์ที่ใช้ในระบบมีความไม่แน่นอนในเรื่องของการใช้กระแสไฟในการทำงานเนื่องจากข้อมูลที่ได้จาก data sheet และการนำไปใช้งานจริงมีความแตกต่างกันทำให้เกิดความผิดพลาดในการหาแหล่งจ่ายไฟให้กับระบบ

2. ความเร็วในการส่งข้อมูลสื่อสารของ ESP32 และ IOT Platform มีความเร็วที่มีการกำจัดความเร็วในการรับส่งข้อมูลทำให้ผลที่ได้มีความละเอียดที่ไม่มากเพียงพอกับระบบคานสมดุล

5.3 แนวทางการแก้ไขปัญหา

1. ทำการทดลองป้อนค่า PWM ที่น้อยที่สุดที่สามารถทำให้มอเตอร์ยกคานจนถึงจุดสูงสุดได้ แล้วทำการวัดค่ากระแสที่แล้วทำการหาแหล่งจ่ายที่เหมาะสม

2. ทำการค้นหาข้อมูลเกี่ยวกับการรับส่งข้อมูลของแต่ละ IOT Platform แล้วนำข้อมูลมาเปรียบเทียบว่า IOT Platform ตัวไหนมีความเหมาะสมกับระบบควบคุมคานสมดุลมากที่สุด

5.4 การพัฒนาในอนาคต

1. ประยุกต์การทำงานกับระบบอื่นๆ ที่มีความคล้ายคลึงกัน

2. สามารถนำชุดอุปกรณ์ของระบบควบคุมคานสมดุลแบบไร้สายไปประยุกต์ใช้งานสำหรับห้องปฏิบัติการระยะไกลหรือห้องปฏิบัติการออนไลน์ได้

เอกสารอ้างอิง

[1] “IC Voltage Regulator” เข้าถึงได้จาก: https://www.arduino4.com/article

[2] “การทำงานของ Optocoupler PC817” เข้าถึงได้จาก: https://tahobaza.ru/th/optopara-pc817-princip-raboty-i-ochen-prostaya-proverka-optopara-pc817-princip-raboty/

[3] “ULN2003 IC” เข้าถึงได้จาก: https://components101.com/ics/stepper-motor-driver-ic-uln2003-pinout-datasheet

[4] “Motor 130” เข้าถึงได้จาก: https://commandronestore.com/products/bc001.php

[5] “การเขียนโปรแกรม Arduino” เข้าถึงได้จาก: https://www.arduino.cc/en/Guide

[6] “การเขียนโปรแกรมรับ-ส่งข้อมูลผ่าน NETPIE” เข้าถึงได้จาก: https://docs.netpie.io/

[7] “กฎการเคลื่อนที่ของนิวตัน” เข้าถึงได้จาก: https://school.dek-d.com/blog/high-school/newtons-law/

[8] “มวลและน้ำหนัก” เข้าถึงได้จาก: https://www.ipst.ac.th/learning/3026/mass.html

[9] “แรงโน้มถ่วงของโลก” เข้าถึงได้จาก: https://ngthai.com/science/24097/gravitational-force/

[10] “แรงยก” เข้าถึงได้จาก: https://hmong.in.th/wiki/Lift_(force)

[11] “สภาพสมดุล” เข้าถึงได้จาก: https://www.scimath.org/lesson-physics/item/7226-equilibrium

[12] “โมเมนต์ของแรง” เข้าถึงได้จาก: http://cms575.bps.in.th/group13/moment-of-a-force2

[13] “พลังงานลมและอากาศพลศาสตร์” เข้าถึงได้จาก: http://fulltext.rmu.ac.th/fulltext/2553/86523/chapter%202.pdf

[14] “ระบบควบคุมพีไอดี” เข้าถึงได้จาก: https://th.jf-parede.pt/what-is-pid-controller

[15] “การปรับจูนพีไอดี” เข้าถึงได้จาก: https://thaicontrol.wordpress.com/2011/11/27/pid-tuning/

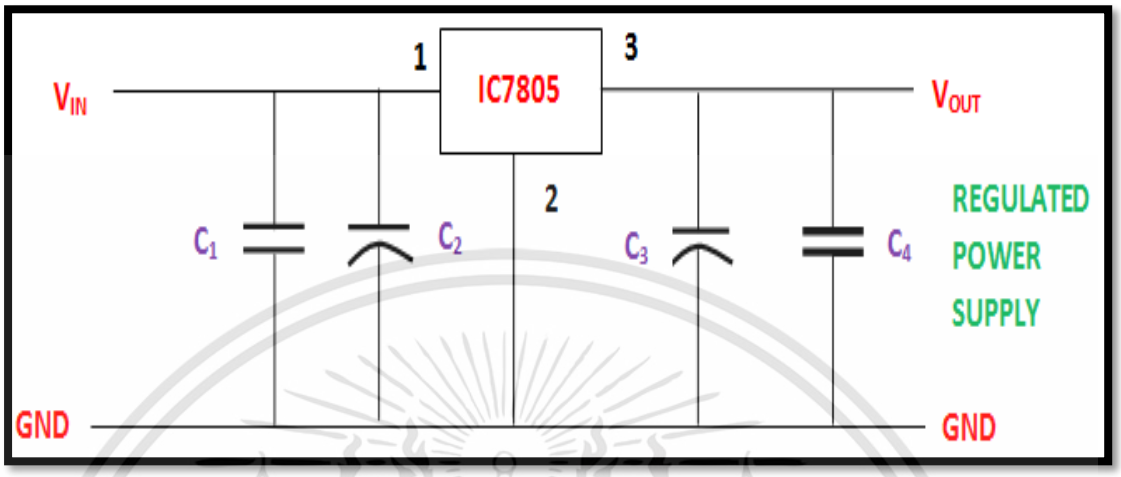
[16] “โปรแกรม Visual Studio” เข้าถึงได้จาก: https://visualstudio.microsoft.com/

[17] “โปรแกรม Fritzing” เข้าถึงได้จาก: https://fritzing.org/

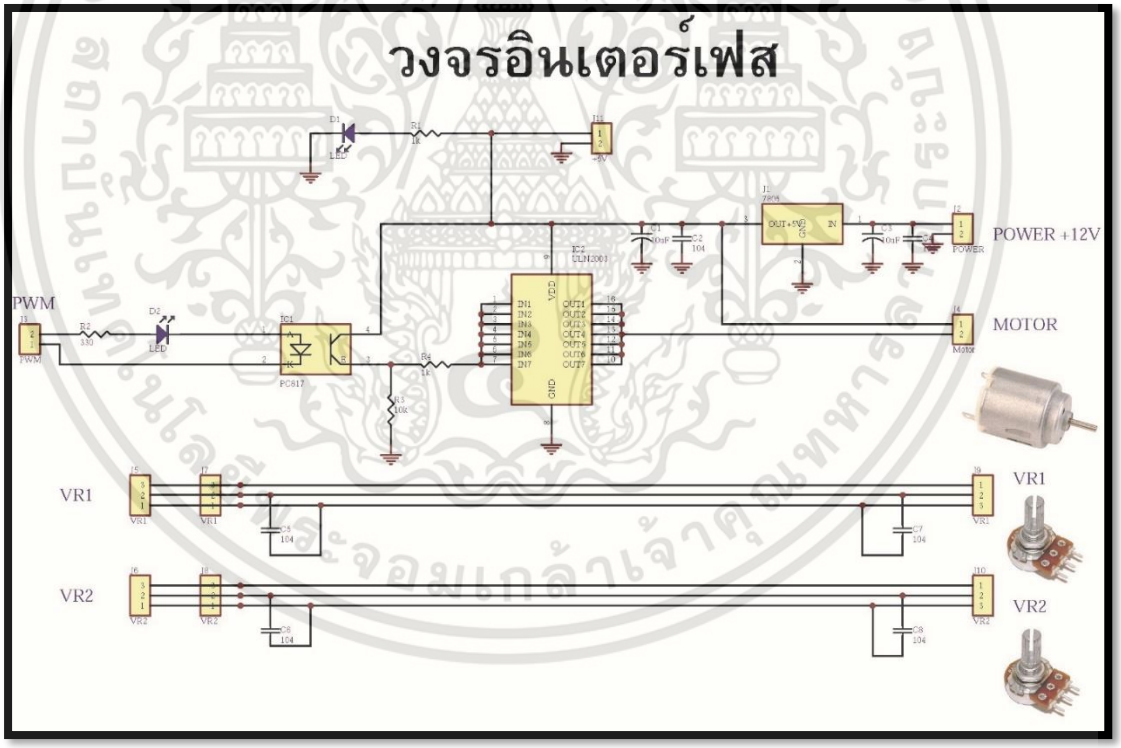
ภาคผนวก

ภาคผนวก ก

วงจรในชุดทดลองควบคุมคานสมดุล



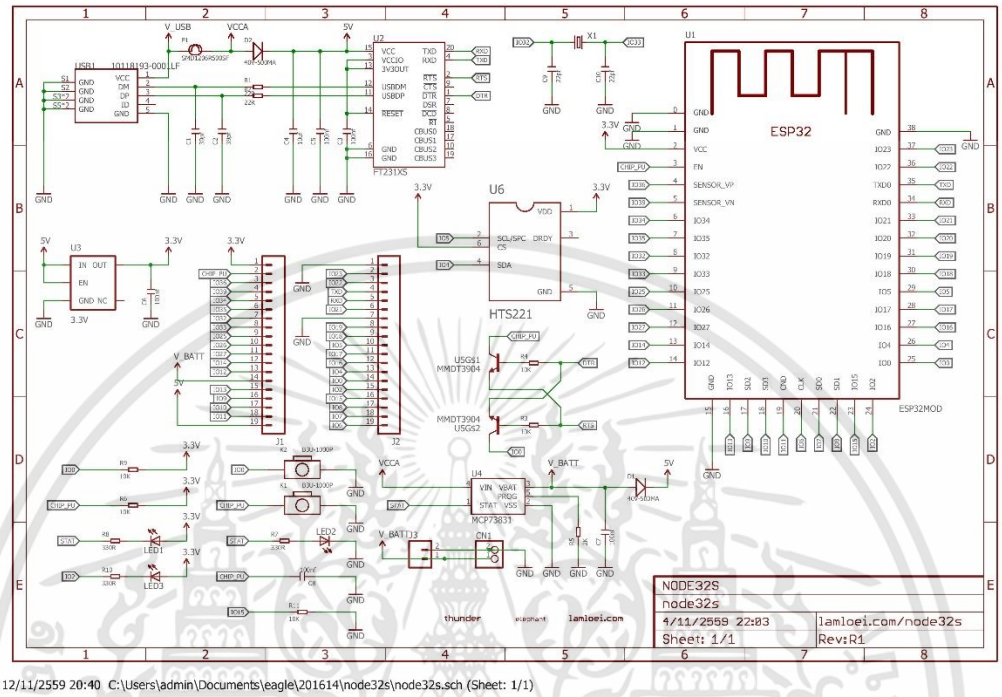
รูปที่ ก.1 วงจร 7805 Regulator Circuit



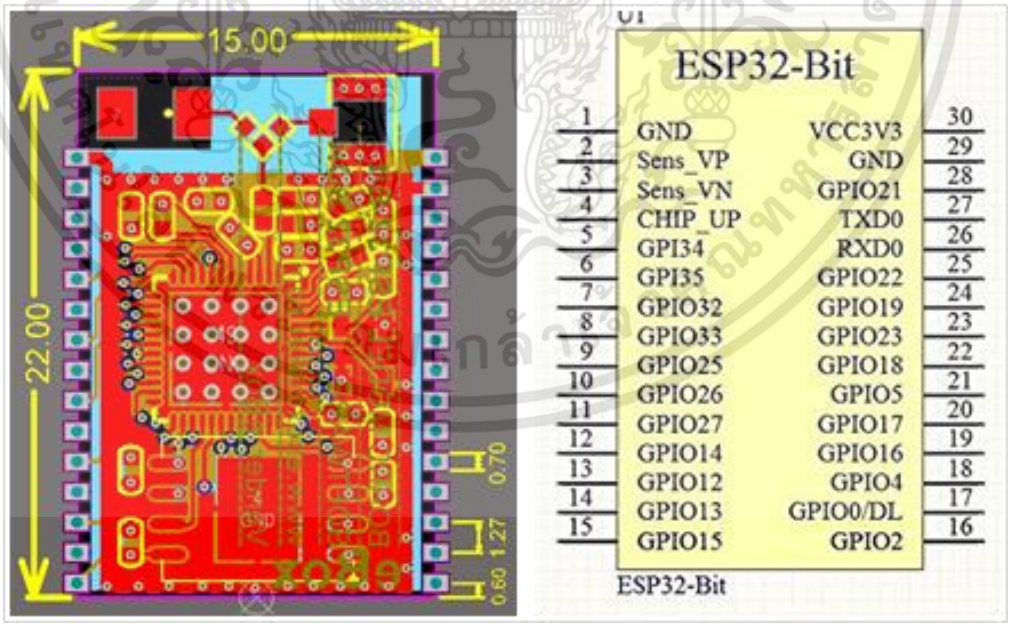
รูปที่ ก.2 วงจรรวมของชุดทดลองคานสมดุล

ภาคผนวก ข

บอร์ด Arduino ESP32



รูปที่ ข.1 โครงสร้างภายในบอร์ด Arduino ESP 32



รูปที่ ข.2 ตำแหน่งขาของบอร์ด Arduino ESP32

Table 2: Pin Definitions

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply
EN	3	I	Module-enable signal. Active high.
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8

รูปที่ ข.3 คุณสมบัติทั่วไปของบอร์ด Arduino ESP32

Name	No.	Type	Function
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Ground
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2*	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3*	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD*	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/CLK*	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0*	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1*	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Ground

รูปที่ ข.4 คุณสมบัติทั่วไปของ บอร์ด ESP32 (3)

ภาคผนวก ค

โปรแกรมควบคุมการทำงาน การส่ง-ข้อมูล และการแสดงผลของระบบ

```
#include <analogWrite.h>
#include <PubSubClient.h>
#include <WiFi.h>
//--------------------------------------------------------------------------
const char* ssid = "iPhone";
const char* password = "Jeawjieeiei";
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "39097ff8-9d26-489b-9207-a3685276748b";
const char* mqtt_username = "JKakfkKmydj2dbcMBR6qmD2xoGmCQXtu";
const char* mqtt_password = "Bmd$8unEArhME(cBq(CRjIlt06Ml_Kyt";
WiFiClient espClient;
PubSubClient client(espClient);
//--------------------------------------------------------------------------
int inAuto = false;
char msg[1000];
double lastMsg = 0;
double outMin, outMax;
unsigned long lastTime;
double Feedback, fed, set, Setpoint, Output, output, setpoint, feedback, outputload,
f, feedbackdegree;
double errSum, lastErr, error;
double ITerm, lastInput;
double kp, ki, kd, kpapp, kiapp, kdapp;
int count_time, setpointdata, feedbackdata, setpointapp, modeapp, outputdata, mod,
Motor, Motorload;
```

```
#define MANUAL 0
#define AUTOMATIC 1
#define PIN_OUTPUT 12
#define PIN_OUTPUT1 13
//-----------------------------------------------------------------------
double readsetpoint() {
  set = analogRead(34);
  Setpoint= map(set,0,4095,0,100);
  return Setpoint;
}
//-----------------------------------------------------------------------
double readfeedback() {
  fed = analogRead(35);
  Feedback= map(fed,240,1660,0,100);
  return Feedback;
}
//-----------------------------------------------------------------------
void SetTunings(double Kp, double Ki, double Kd){
  kp = Kp;
  ki = Ki;
  kd = Kd;
}
//-----------------------------------------------------------------------
void Compute(){
  if(mod==0) return;
  unsigned long now = millis();
  double timeChange = (double)(now-lastTime);
  double error =  Setpoint - (100-Feedback);
  errSum += (error*timeChange);
  ITerm = ki*errSum;
```

```
  if(ITerm<-255)errSum = (ITerm/ki)-(error*timeChange);
  if(ITerm>255)errSum = (ITerm/ki)-(error*timeChange);
  double dinput = (error - lastErr);
  output=(kp*error)+ITerm+(kd*dinput);
  if(output<0)output=0;
  if(output>130)output=130;
  analogWrite(PIN_OUTPUT, output);
  lastErr=error;
  lastTime=now;
}
//-------------------------------------------------------------------
void SetMode(int mod)
{
 //bool newAuto = (mod == 1);
 if(mod== 1)
 { //we just went from manual to auto
 Initialize();
 }
 if(mod == 0)
 {
      output=Motor;
      if(output > 130) output = 130;
      else if(output < 0) output = 0;
      analogWrite(PIN_OUTPUT, output);
 }
}
//-------------------------------------------------------------------------
void Initialize()
{
   lastErr = error;
```

```
  ITerm = output;
  if(ITerm> 130) ITerm= 130;
  else if(ITerm< 0) ITerm= 0;
}
//-------------------------------------------------------------------
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    String message;
    for (int i = 0; i < length; i++) {
      message = message + (char)payload[i];
    }
    Serial.println(message);
    if(String(topic) == "@msg/status") {
      if(message == "on"){
      mod = 1;
      client.publish("@shadow/data/update", "{\"data\":{\"status\":\"on\"}}");
      Serial.println("Auto");
      }
      else if(message == "off"){
      mod = 0;
      client.publish("@shadow/data/update", "{\"data\":{\"status\":\"off\"}}");
      Serial.println("Manual");
      }
    }
    //-----------------------------------------------------------------------
  else if(String(topic) == "@msg/input"){
      int setapp = message.toInt();
      setpoint = map(setapp,-45,45,0,100);
```

```
    String data = "{\"data\": {\"input\":" + String(setapp) + "}}";
    Serial.println(data);
    data.toCharArray(msg, (data.length() + 1));
    client.publish("@shadow/data/update", msg);
    }
//------------------------------------------------------------------------
else if(String(topic) == "@msg/motor"){
    int Motorapp = message.toInt();
    Motor=map(Motorapp,0,100,0,130);
    String data = "{\"data\": {\"motor\":" + String(Motorapp) + "}}";
    Serial.println(data);
    data.toCharArray(msg, (data.length() + 1));
    client.publish("@shadow/data/update", msg);
    }
//------------------------------------------------------------------------
else if(String(topic) == "@msg/motorload"){
    int Motorapp1 = message.toInt();
    Motorload=map(Motorapp1,0,100,125,130);
    String data = "{\"data\": {\"motorload\":" + String(Motorapp1) + "}}";
    Serial.println(data);
    data.toCharArray(msg, (data.length() + 1));
    client.publish("@shadow/data/update", msg);
    }
//------------------------------------------------------------------------
 else if(String(topic) == "@msg/kp"){
    double kpp = message.toDouble();
    kpapp=kpp;
    String data = "{\"data\": {\"kp\":" + String(kpp) + "}}";
    Serial.println(data);
    data.toCharArray(msg, (data.length() + 1));
```

```
      client.publish("@shadow/data/update", msg);
      }
    //------------------------------------------------------------------------
      else if(String(topic) == "@msg/ki"){
      double kii = message.toDouble();
      kiapp=kii;
      String data = "{\"data\": {\"ki\":" + String(kii) + "}}";
      Serial.println(data);
      data.toCharArray(msg, (data.length() + 1));
      client.publish("@shadow/data/update", msg);
      }
      //------------------------------------------------------------------
      else if(String(topic) == "@msg/kd"){
      double kdd = message.toDouble();
      kdapp=kdd;
      String data = "{\"data\": {\"kd\":" + String(kdd) + "}}";
      Serial.println(data);
      data.toCharArray(msg, (data.length() + 1));
      client.publish("@shadow/data/update", msg);
      }
  }
//------------------------------------------------------------------------
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting NETPIE2020 connection...");
    if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {
      client.subscribe("@msg/#");
      Serial.println("NETPIE2020 connected");
    }
    else {
```

```
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
  }
}
//--------------------------------------------------------------------
void setup() {
  Serial.begin(115200);
  Serial.println("Starting...");
  if (WiFi.begin(ssid, password)) {
    while (WiFi.status() != WL_CONNECTED) {
      delay(1000);
      Serial.print(".");
    }
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  outputload=Motorload;
  analogWrite(PIN_OUTPUT1, outputload);
```

```
  SetMode(mod);
  Setpoint = setpoint;
  Feedback = readfeedback();
  SetTunings(kpapp,kiapp*0.0001,kdapp*0.01);
  //SetTunings(5,0.001,100);
  Compute();
  Output=map(output,0,130,0,100);
  f=100-Feedback;
  feedbackdegree=map(f,0,100,-45,45);
  long now = millis();
  if (now - lastMsg > 250) {
    lastMsg = now;
    String data = "{\"data\": {\"setpoint\":" + String(Setpoint) + ", \"feedback\":" + String(f)
+ ", \"output\": " +  String(Output) + ", \"feedbackdegree\":" + String(feedbackdegree) +
"}}";
    Serial.println(data);
    data.toCharArray(msg, (data.length() + 1));
    client.publish("@shadow/data/update", msg);
  }
}
```