ออกแบบและพัฒนาการตรวจสอบสภาพแวดล้อมที่เหมาะสำหรับการเลี้ยงแมลงในระบบโรงเรือน

MONITORING SYSTEM DEVELOPMENT AND DESIGN FOR ENHANCING SUITABLE ENVIRONMENTS IN VERTICAL INSECT FARMING

โดย

นางสาวกนกวรรณ ทองม่วง

นายไกรวิชญ์ ชุนหคาม

นางสาวธนัชชา ปริญรัมย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคมและโครงข่าย

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2566

ออกแบบและพัฒนาการตรวจสอบสภาพแวดล้อมที่เหมาะสำหรับการเลี้ยงแมลงในระบบโรงเรือน

MONITORING SYSTEM DEVELOPMENT AND DESIGN FOR ENHANCING SUITABLE ENVIRONMENTS IN VERTICAL INSECT FARMING

โดย

นางสาวกนกวรรณ	**ทองม่วง**	**63010005**
นายไกรวิชญ์	**ชุนหคาม**	**63010096**
นางสาวธนัชชา	**ปริญรัมย์**	**63010431**

อาจารย์ที่ปรึกษา

ดร. พีระเมศร์ โชติกวีกิจญาดา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคมและโครงข่าย

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2566

ปริญญานิพนธ์ปีการศึกษา 2566

ภาควิชาวิศวกรรมโทรคมนาคมและโครงข่าย

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ออกแบบและพัฒนาการตรวจสอบสภาพแวดล้อมที่เหมาะสำหรับการเลี้ยงแมลงในระบบโรงเรือน

MONITORING SYSTEM DEVELOPMENT AND DESIGN FOR ENHANCING SUITABLE ENVIRONMENTS IN VERTICAL INSECT FARMING

ผู้จัดทำ

1. นางสาวกนกวรรณ ทองม่วง 63010005
2 นายไกรวิชญ์ ชุนหคาม 63010096
3. นางสาวธนัชชา ปริญรัมย์ 63010431

..... **อาจารย์ที่ปรึกษา**

ดร. พีระเมศร์ โชติกวีกิจญาดา

กิตติกรรมประกาศ

การดำเนินปริญญานิพนธ์ เรื่องออกแบบและพัฒนาการตรวจสอบภาพแวดล้อมที่เหมาะสำหรับการเลี้ยงแมลงในระบบโรงเรือน จะไม่สามารถสำเร็จลุล่วงไปได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และความอนุเคราะห์อย่างดียิ่งจากอาจารย์ที่ปรึกษาโครงงานคือ ดร. พีระเมศร์ โชติกวีกิจญาดา ที่กรุณาให้คำสั่งสอนและแนวทางการแก้ไขปัญหาตลอดระยะเวลาในการจัดทำโครงงานนี้ รวมทั้งสนับสนุนสถานที่ เครื่องมือและอุปกรณ์ต่างๆ ที่จำเป็นต้องใช้ในระหว่างการจัดทำโครงงานขอขอบพระคุณท่านในความห่วงใย และความหวังดีที่มีให้แก่คณะผู้จัดทำเป็นอย่างยิ่ง

ขอขอบคุณท่านอาจารย์ประจำภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้อบรมสั่งสอนและให้คำแนะนำวิชาความรู้ ให้แก่คณะผู้จัดทำ ขอขอบคุณผู้มีส่วนเกี่ยวข้องทุกท่านที่คอยสนับสนุนแนะนำและช่วยเหลือแก่คณะผู้จัดทำ จนกระทั่งโครงงานสำเร็จลุล่วงไปได้ด้วยดี

นางสาวกนกวรรณ ทองม่วง
นายไกรวิชญ์ ชุนหคาม
นางสาวธนัชชา ปริญรัมย์

ผู้จัดทำ

ออกแบบและพัฒนาการตรวจสอบภาพแวดล้อมที่เหมาะสำหรับการเลี้ยงแมลงในระบบโรงเรือน
MONITORING SYSTEM DEVELOPMENT AND DESIGN FOR ENHANCING SUITABLE ENVIRONMENTS IN VERTICAL INSECT FARMING

โดย นางสาวกนกวรรณ ทองม่วง 63010005
นายไกรวิชญ์ ชุนหคาม 63010096
นางสาวธนัชชา ปริญรัมย์ 63010431

อาจารย์ที่ปรึกษา ดร. พีระเมศร์ โชติกวีกิจญาดา

บทคัดย่อ

การสำรวจอวกาศอาจจะเป็นจุดเริ่มต้นของการพัฒนาความรู้ด้านต่างๆ ซึ่งสภาพความเป็นอยู่ของนักบินอวกาศเป็นเรื่องสำคัญอย่างยิ่ง โดยเฉพาะการตุนเสบียงอาหารสำหรับนักบินอวกาศ ทางนาซ่าได้มีโครงการพัฒนาระบบผลิตอาหารอัตโนมัติในสถานีสำหรับนักบินอวกาศ เพื่อที่จะทำให้สามารถผลิตอาหารที่สดใหม่พร้อมรับประทาน โดยแมลงจัดเป็นกลุ่มของสัตว์ที่มีจำนวนมากที่สุดในโลก และแมลงกลายเป็นสัตว์เศรษฐกิจสำคัญ โดยเฉพาะแมลงกินได้ (EDIBLE BUG) ที่มีโภชนาการจากแมลงประกอบด้วย โปรตีน ไขมัน ซึ่งให้พลังงานที่สามารถใช้ทดแทนและเป็นทางเลือกได้

ผู้จัดทำเล็งเห็นว่าถ้ามีระบบผลิตอาหารบนสถานีวิจัยอวกาศที่สามารถผลิตอาหารได้อย่างต่อเนื่อง ยั่งยืน และใช้พื้นที่น้อยที่สุดได้จริง ระบบนี้จะช่วยลดงบประมาณการเติมเสบียงไปยังอวกาศ อีกทั้งยังสามารถนำมาประยุกต์การใช้งานอื่นบนโลกได้อีกด้วย ทางผู้จัดทำได้ออกแบบการเลี้ยงแมลงในระบบโรงเรือนหรือกล่องจำลองการเลี้ยงแมลงโดยคำนึงถึง 2 ด้าน คือ 1. ด้านการเพาะเลี้ยงแมลง โดยจะออกแบบและพัฒนาระบบตรวจจับสภาพแวดล้อมภายในกล่องจำลองการเลี้ยงแมลงที่ส่งข้อมูลแบบไร้สาย 2. ด้านการกำจัดของเสียทางอากาศ ที่จะใช้ AI ช่วยในการทำงานและมีระบบแจ้งเตือนเมื่อเกิดความผิดปกติในระบบ ซึ่งในโครงงานนี้เป็นเพียงการจำลองสถานการณ์เพื่อตรวจจับสภาพแวดล้อมภายในกล่องและได้ทำการศึกษาทฤษฎีเกี่ยวกับวงจรชีวิตของแมลงเพื่อออกแบบกล่องจำลอง แต่ไม่ได้มีการเลี้ยงจริงเนื่องเป็นการรบกวนสิ่งมีชีวิต

ABSTRACT

SPACE EXPLORATION COULD BE THE BEGINNING POINT FOR GAINING KNOWLEDGE IN A VARIETY OF TOPICS. ASTRONAUTS' LIVING CONDITIONS ARE CRUCIAL, ESPECIALLY STOCKPILING FOOD SUPPLIES FOR ASTRONAUTS. AN AUTOMATED SYSTEM FOR PRODUCING FRESH, FOOD THAT CAN BE EATEN ON BOARD THE SPACE STATION IS BEING DEVELOPED BY NASA AS PART OF A PROJECT. INSECTS ARE THE WORLD'S MOST NUMEROUS ANIMAL GROUP. INSECTS HAVE BECOME IMPORTANT ECONOMIC ANIMALS. EDIBLE INSECTS INCLUDE INSECT NUTRITION IN THE FORM OF PROTEIN AND FAT, WHICH GIVE ENERGY THAT CAN BE USED AS A REPLACEMENT OR ALTERNATIVE.

THE INVENTORS THINK THAT IF THERE COULD BE A FOOD PRODUCTION SYSTEM ON THE SPACE RESEARCH STATION THAT COULD PRODUCE FOOD CONTINUOUSLY AND SUSTAINABLY WHILE REQUIRING THE LEAST AMOUNT OF SPACE, IT WOULD HELP REDUCE THE BUDGET FOR RESUPPLYING SPACE. IT CAN ALSO BE USED FOR DIFFERENT APPLICATIONS AROUND THE WORLD. THE ORGANIZERS DESIGNED INSECT REARING SYSTEMS IN GREENHOUSES OR BOXES TO IMITATE INSECT REARING, TAKING INTO ACCOUNT TWO FACTORS: 1. INSECT-RAISING. IT WILL DESIGN AND BUILD AN ENVIRONMENT SENSING SYSTEM WITHIN THE INSECT FARMING SIMULATOR BOX THAT SENDS DATA WIRELESSLY. 2. AI-ASSISTED AIR WASTE DISPOSAL. THERE IS ALSO A WARNING MECHANISM IN PLACE TO ALERT USERS WHEN PROBLEMS EMERGE IN THE SYSTEM. THIS PROJECT IS JUST A SIMULATION TO DETECT THE ENVIRONMENT WITHIN THE BOX, SO THE SIMULATION BOX WAS DESIGNED AFTER RESEARCHING INSECT LIFE CYCLE THEORY. HOWEVER, IT IS NOT TRULY RAISED BECAUSE IT DISTURBS LIVE CREATURES.

สารบัญ

		หน้า
กิตติกรรมประกาศ		I
บทคัดย่อ		II
สารบัญ		III
สารบัญรูป		XI
สารบัญตาราง		XIV
บทที่ 1	**บทนำ**	**1**
	1.1 ความเป็นมาและความสำคัญของปัญหา	1
	1.2 วัตถุประสงค์	2
	1.3 ขอบเขตของโครงงาน	2
บทที่ 2	**ทฤษฎีและหลักการที่เกี่ยวข้อง**	**3**
	2.1 ทฤษฎีที่เกี่ยวข้องกับฮาร์ดแวร์ภายในระบบ	3
	2.1.1 ESP-WROOM-32	3
	2.1.2 MAX485 Module	4
	2.1.3 Waterproof Temperature Humidity Sensor RS485	4
	2.1.4 H2S Hydrogen Sulfide Gas Sensor RS485	5
	2.1.5 Carbon dioxide Sensor RS485	6
	2.1.6 Oxygen Sensor RS485	6
	2.1.7 Soil Temperature Moisture PH and EC Sensor RS485	7
	2.1.8 Raspberry Pi 4 Model B	8
	2.1.9 Micro SD Card Module	9
	2.1.10 L298N Motor Driver Module	10
	2.1.11 พัดลมระบายอากาศ	11
	2.1.12 Micro DC Motor	12

2.1.13 Lithium-Ion Battery 12V 13
2.1.14 INA226 13
2.1.15 DC Relay module 14
2.1.16 Limit switch 15
2.1.17 LM2596 DC Voltage Regulator 15
2.1.18 มาตรฐาน IP 16
2.2 โปรโตคอลที่ใช้ในโครงงาน 17
2.2.1 Modbus RTU 17
2.2.2 TTL 18
2.2.3 RS485 18
2.2.4 MQTT 19
2.2.5 REST API 21
2.2.6 SMTP 21
2.3 โปรแกรมที่ใช้ภายในโครงงาน 22
2.3.1 Arduino IDE 22
2.3.2 PostgreSQL 22
2.3.3 ThingsBoard 22
2.3.4 Modbus poll 23
2.3.5 SketchUp 23
2.3.6 EasyEDA 23
2.4 ภาษาที่ใช้ภายในโครงงาน 24
2.4.1 C++ 24
2.4.2 JavaScript 24
2.4.3 VPL 25
2.5 วงจรชีวิตของแมลงกินได้ 25
2.5.1 ด้วงสาคู หรือ ด้วงงวงมะพร้าว 25
2.5.1.1 การแบ่งวงจรชีวิตเป็น 4 ระยะ 26

2.5.1.2 วิธีการเลี้ยงด้วงสาคู 26
2.5.1.3 ข้อควรระวังในการเลี้ยงด้วงสาคู 26
2.5.2 ไหม 27
2.5.2.1 การแบ่งวงจรชีวิตเป็น 4 ระยะ 27
2.5.2.2 วิธีการเลี้ยงไหม 27
2.5.2.3 ข้อควรระวังในการเลี้ยงไหม 28
2.5.3 จิ้งหรีด 28
2.5.3.1 การแบ่งวงจรชีวิตเป็น 3 ระยะ 28
2.5.3.2 วิธีการเลี้ยงจิ้งหรีด 28
2.5.3.3 ข้อควรระวังในการเลี้ยงจิ้งหรีด 29

บทที่ 3 การออกแบบและการจัดทำปริญญานิพนธ์ 30
3.1 การออกแบบโครงสร้างระบบโดยรวม 30
3.1.1 การออกแบบกล่องจำลองการเลี้ยงแมลง 32
3.1.2 การออกแบบการเชื่อมต่ออุปกรณ์กับ Microcontroller 33
3.1.2.1 การออกแบบการเชื่อมต่อเซนเซอร์ กับ โมดูล MAX 485 33
3.1.2.2 การออกแบบระบบกำจัดของเสียผ่าน ESP-WROOM-32 34
3.1.2.3 การออกแบบระบบฐานข้อมูลในรูปแบบออฟไลน์ 36
3.1.2.4 การออกแบบระบบตรวจสอบแรงดัน 37
3.1.3 การส่งชุดข้อมูลจากกล่องจำลองการเลี้ยงแมลง 38
3.1.3.1 การตั้งค่า Address ของเซนเซอร์ 38
3.1.3.2 การส่งชุดข้อมูลจากเซนเซอร์ ไป ESP-WROOM-32 39
3.1.3.3 การรับชุดข้อมูลโดย Raspberry Pi4 จาก ESP-WROOM-32 43
3.1.4 การสร้างและออกแบบระบบรายงานข้อมูล 45

3.1.4.1 การกำหนดพิกัด และการแสดงรายงานสภาพอากาศอัตโนมัติผ่าน REST API 45
3.1.4.2 การออกแบบระบบกำจัดของเสีย 45
3.1.4.3 การสร้างระบบแจ้งเตือน 46
3.2 เครื่องมือที่ใช้ในการทดลอง 46
3.2.1 ESP-WROOM-32 46
3.2.2 MAX 485 Module 46
3.2.3 Waterproof Temperature Humidity Sensor 47
3.2.4 H2S Hydrogen sulfide gas Sensor 47
3.2.5 Carbon dioxide Sensor 47
3.2.6 Oxygen Gas Module Sensor 47
3.2.7 Soil Temperature Moisture PH Sensor 47
3.2.8 Raspberry Pi 4 Model B 47
3.2.9 Micro SD card 47
3.2.10 L298N Motor Driver Module 47
3.2.11 พัดลมระบายอากาศ 47
3.2.12 Micro DC Motor 48
3.2.13 Lithium-Ion Battery 12V 48
3.2.14 INA 226 48
3.2.14 Limit switch 48
3.2.15 DC Relay module 48
3.2.16 Limit switch 48
3.2.17 LM2596 DC Voltage Regulator 48
3.3 การจัดเก็บผลการทดลอง 48
3.3.1 การทดสอบการรับข้อมูลจากเซนเซอร์ 48

3.3.2 การทดสอบการทำงาน ESP-WROOM-32 ในการควบคุมพัดลม 49
3.3.3 การทดสอบการทำงาน ESP-WROOM-32 ในการเปิด-ปิดประตูพัดลม 49
3.3.4 การจำลองการทดสอบระบบการทำงานในระยะเวลา 7 วัน 51
3.4 การประกอบชิ้นงานตามต้นแบบ 51
3.4.1 การออกแบบตัวกล่องชิ้นงาน 51
3.4.2 การออกแบบวงจรรวม 52
3.4.3 ระบบกำจัดของเสีย 53

บทที่ 4 ผลการทดลอง 55
4.1 การทดสอบการทำงานระหว่างอุปกรณ์ กับ Microcontroller 55
4.1.1 การทำงานระหว่างเซนเซอร์กับ โมดูล MAX485 55
4.1.2 การทำงานระหว่าง Arduino ESP32 กับ พัดลมระบายอากาศ 58
4.1.3 การทำงานระหว่าง Arduino ESP32 กับ ประตูระบายอากาศ 60
4.1.4 การจำลองระบบการทำงานของเซนเซอร์ 62
4.1.4.1 การเก็บค่าความชื้นในอากาศ และในดิน 62
4.1.4.2 การเก็บค่าอุณหภูมิในอากาศ และในดิน 63
4.1.4.3 การเก็บค่าคาร์บอนไดออกไซด์ 64
4.1.4.4 การเก็บค่าออกซิเจน 64
4.1.4.5 การเก็บค่า pH 65
4.1.4.6 การเก็บค่าไฮโดรเจนซัลไฟด์ 66
4.2 การทดสอบโดยรวมของชิ้นงาน 66
4.2.1 การส่งชุดข้อมูลจากกล่องจำลองการเลี้ยงแมลง 66
4.2.2 การสร้างระบบรับข้อมูลรายงานสภาพอากาศ 67
4.2.3 การแสดงค่าพิกัด และการรายงานสภาพอากาศจากกล่อง 68

จำลอง

4.2.4 การทำงานของระบบแจ้งเตือน 71

บทที่ 5 สรุปผลและข้อเสนอแนะ 72

5.1 สรุปผล 72

5.2 ข้อเสนอแนะ 73

สารบัญ (ต่อ)

		หน้า
บรรณานุกรม		**74**
ภาคผนวก	ESP-WROOM-32 Coding.	77

สารบัญรูป

รูปที่		หน้า
2.1	ESP-WROOM-32	3
2.2	โมดูล MAX485	4
2.3	เซนเซอร์ Waterproof Temperature Humidity RS485	5
2.4	เซ็นเซอร์ H2S Hydrogen Sulfide Gas Sensor RS485	5
2.5	เซนเซอร์ Carbon dioxide Sensor RS485	6
2.6	เซนเซอร์ Oxygen Sensor RS485	7
2.7	เซนเซอร์ Soil Temperature Moisture PH and EC Sensor RS485	8
2.8	Raspberry Pi 4 Model B	9
2.9	Micro DC Card	10
2.10	โมดูล L298N Motor Driver	11
2.11	พัดลมระบายอากาศ	12
2.12	Micro DC Motor	12
2.13	Lithium-Ion Battery 12V	13
2.14	INA 226	14
2.15	DC Relay module	14
2.16	Limit switch	15
2.17	LM2596 DC Voltage Regulator	16
2.18	การสื่อสารแบบอนุกรมด้วย RS – 485 สำหรับ Modbus RTU	17
2.19	ชุดข้อมูลสำหรับการสื่อสาร Modbus RTU	17
2.20	การเชื่อมต่อเซนเซอร์ 485 เข้ากับตัวแปลงสัญญาณ	19
2.21	การเชื่อมต่อโปรโตคอล MQTT	20
2.22	การการประมวลผลข้อมูลของ ThingsBoard	23
3.1	บล็อกไดอะแกรมภาพรวมของโครงงาน	31
3.2	ภาพรวมของการเชื่อมต่ออุปกรณ์ทั้งหมด	31
3.3	การออกแบบกล่องจำลองการเลี้ยงแมลงผ่านโปรแกรม SketchUp pro	32
3.4	ภาพฉายมุมที่ 1 ตามระบบ ISO-Method	33
3.5	การเชื่อมต่อระหว่างเซนเซอร์ กับ MAX 485	33

3.6	การเชื่อมต่อระหว่าง ESP-WROOM-32 กับ พัดลมระบายอากาศ	34
3.7	การเชื่อมต่อระหว่าง ESP-WROOM-32 กับ มอเตอร์	36
3.8	การเชื่อมต่อระหว่าง ESP-WROOM-32 กับ Micro SD card	37
3.9	การเชื่อมต่อระหว่าง ESP-WROOM-32 กับ โมดูล INA 226	38
3.10	ขั้นตอนการตั้งค่า Address ของเซนเซอร์	39
3.11	การทำงานการส่งชุดข้อมูลจากฝั่ง Client ไปยัง Server	41
3.12	การทำงานการส่งชุดข้อมูลจากฝั่ง Client ไปยัง Server	42
3.13	การทำงานการส่งชุดข้อมูลจากฝั่ง Client ไปยัง Server	42
3.14	การทำงานการรับชุดข้อมูลโดย Server จากฝั่ง Client	44
3.15	การทำงานของการระบุพิกัด และการรายงานสภาพอากาศจาก	45
3.16	แผนการทำงานของระบบกำจัดของเสีย	45
3.17	การออกแบบการสร้างระบบแจ้งเตือนผ่านอีเมล	46
3.18	การทดสอบการรับข้อมูลจากเซนเซอร์	49
3.19	การทำงานการปิด ของการเปิด-ปิดประตูพัดลม	50
3.20	การทำงานการเปิด ของการเปิด-ปิดประตูพัดลม	50
3.21	การเติมแคลเซียมคาร์บอเนตเพื่อปรับสภาพความเป็นกรดเบส	51
3.22	กล่องจำลองการเลี้ยงแมลง	52
3.23	วงจรรวม	53
3.24	ระบบกำจัดของเสีย	54
4.1	ค่าสัญญาณบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์	55
4.2	ถอดรหัสบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์(ไบต์ที่1-2)	56
4.3	ถอดรหัสบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์(ไบต์ที่3-5)	56
4.4	ถอดรหัสบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์(ไบต์ที่6-9)	57
4.5	ถอดรหัสบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์ทั้ง 9 ไบต์	57
4.6	การทำงานของพัดลมระบายอากาศที่ความเร็วปานกลาง	58
4.7	อุปกรณ์วัดความเร็วลมแบบใบพัดที่ความเร็วปานกลาง	59

4.8	การทำงานของพัดลมระบายอากาศที่ความเร็วสูงสุด	59
4.9	อุปกรณ์วัดความเร็วลมแบบใบพัดที่ความเร็วสูงสุด	60
4.10	ประตูระบายอากาศเปิด	60
4.11	ประตูระบายอากาศปิด	61
4.12	หน้าจอมอนิเตอร์แสดงการทำงานของประตูระบายอากาศที่ทำการเปิด	61
4.13	หน้าจอมอนิเตอร์แสดงการทำงานของประตูระบายอากาศที่ทำการปิด	62
4.14	กราฟการทดลองการเก็บค่าความชื้น	63
4.15	กราฟการทดลองการเก็บค่าอุณหภูมิ	63
4.16	กราฟการทดลองการเก็บค่าคาร์บอนไดออกไซด์	64
4.17	กราฟการทดลองการเก็บค่าออกซิเจน	65
4.18	กราฟการทดลองการเก็บค่า pH	65
4.19	กราฟการทดลองการเก็บค่าไฮโดรเจนซัลไฟด์	66
4.20	หน้าจอ serial monitor แสดงค่าข้อมูลที่วัดได้	67
4.21	หน้าจอแสดงผลแบบเรียลไทม์	67
4.22	อุณหภูมิ ความชื้นภายในอากาศ และดิน	68
4.23	ค่าความเป็นกรดด่างภายในดิน	68
4.24	ค่าแก๊สออกซิเจนภายในอากาศ	68
4.25	ค่าแก๊สไฮโดรเจนซัลไฟด์ในอากาศ	68
4.26	หน้าจอแสดงพิกัดของกล่องจำลองการเลี้ยงแมลง และรายงานสภาพอากาศจาก REST API	69
4.27	น้าจอแสดงพิกัดบนแมพ	69
4.28	การระบุพิกัดกล่องจำลองทางการเกษตร	70
4.29	หน้าจอแสดงค่าอุณหภูมิโดยรอบจาก REST API	70
4.30	หน้าจอแสดงความชื้นโดยรอบจาก REST API	70
4.31	หน้าจอแสดงผลเมื่อค่าที่ได้รับน้อยกว่าที่กำหนด	71
4.32	ระบบแจ้งเตือนผ่านอีเมลที่กำหนดไว้	71

สารบัญตาราง

ตารางที่		หน้า
2.1	GPIO pinout ของ Raspberry Pi 4	9
2.2	พินของโมดูล Micro SD Card	10
2.3	รายละเอียดรหัส IP	16
2.4	แนวความคิดและหลักการทำงานของ MQTT	20
3.1	การเชื่อมต่อของ PIN ในอุปกรณ์ ESP-WROOM-32 กับ MAX 485	34
3.2	การเชื่อมต่อของ PIN ในอุปกรณ์ ESP-WROOM-32 กับ พัดลมระบายอากาศ	35
3.3	การเชื่อมต่อของ PIN ในอุปกรณ์ ESP-WROOM-32 กับ มอเตอร์	36
3.4	การกำหนดช่วงการทำงานของระบบกำจัดของเสีย	37
3.5	การเชื่อมต่อของ PIN ในอุปกรณ์ ESP-WROOM-32 กับ โมดูล INA 226	38

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การสำรวจอวกาศเพื่อหาความเป็นไปได้ที่จะอพยพมนุษย์ไปตั้งถิ่นฐานที่อื่นหรือจะเป็นการยกระดับองค์ความรู้ทางดาราศาสตร์ และวิทยาศาสตร์ ที่อาจจะเป็นจุดเริ่มต้นของการพัฒนาความรู้ด้านต่างๆ โดยในปัจจุบันเทคโนโลยีที่ก้าวหน้ามากขึ้นของสถานีวิจัยอวกาศนานาชาติทำให้นักบินอวกาศออกไปสำรวจได้เป็นจำนวนมาก ซึ่งการออกไปสำรวจในอวกาศสภาพความเป็นอยู่ของนักบินอวกาศเป็นเรื่องสำคัญอย่างยิ่ง โดยเฉพาะการตุนเสบียงอาหารสำหรับนักบินอวกาศ ในรายงานงบประมาณปี 2022 [2] ชี้แจงเกี่ยวกับการขนส่งไปยังสถานีวิจัยอวกาศนานาชาติอยู่ที่ 1,716.9 ล้านเหรียญดอลลาร์สหรัฐฯ และมีแนวโน้มที่จะมากขึ้นในอนาคต ซึ่งทางนาซ่าได้มีโครงการพัฒนาระบบผลิตอาหารอัตโนมัติในสถานีสำหรับนักบินอวกาศเพื่อที่จะทำให้สามารถผลิตอาหารที่สดใหม่พร้อมรับประทาน นับเป็นปัจจัยสำคัญต่อสุขภาพกายและสุขภาพจิตของนักบินอวกาศที่จะสามารถสร้างสรรค์เมนูที่ชอบที่มีโภชนการสูงได้ในขณะการปฏิบัติหน้าที่ในอวกาศ

แมลงจัดเป็นกลุ่มของสัตว์ที่มีจำนวนมากที่สุดในโลก และแมลงกลายเป็นสัตว์เศรษฐกิจสำคัญ โดยเฉพาะแมลงกินได้ (Edible Bug) มีการคาดการณ์ว่าในอนาคต แมลงจะเป็นความหวังของมนุษยชาติเมื่อเทียบกับจำนวนประชากรที่จะมากขึ้นถึง 9700 ล้านคนในปี 2050 โดยแมลงมีคุณค่าทางอาหารสูง และเป็นสิ่งมีชีวิตที่ขยายพันธุ์ได้อย่างรวดเร็ว ซึ่งในภูมิภาคเอเชียตะวันออกเฉียงใต้ มีแมลงกินได้อยู่มากกว่า 100 ชนิด โดยโภชนาการจากแมลงประกอบด้วย โปรตีน ไขมัน ซึ่งให้พลังงานที่สามารถใช้ทดแทนและเป็นทางเลือกได้

ผู้จัดทำเล็งเห็นว่าถ้ามีระบบผลิตอาหารบนสถานีวิจัยอวกาศที่สามารถผลิตอาหารได้อย่างต่อเนื่อง ยั่งยืน และใช้พื้นที่น้อยที่สุดได้จริง ระบบนี้จะช่วยลดงบประมาณการเติมเสบียงไปยังอวกาศ และจะตอบโจทย์การสร้างสรรค์เมนูที่หลากหลายบนอวกาศที่มีคุณค่าทางอาหารสูงมากกว่าเนื้อสัตว์ปกติในประมาณที่เท่ากัน นอกจากนี้ระบบที่พัฒนาขึ้นจะไม่เป็นประโยชน์ในการใช้ชีวิตบนอวกาศเท่านั้น แต่ยังสามารถนำมาประยุกต์การใช้งานอื่นบนโลกได้อีกด้วย

โครงงานนี้จัดทำการเลี้ยงแมลงในระบบโรงเรือนหรือกล่องจำลองการเลี้ยงแมลงที่คำนึงถึง 2 ด้าน คือ 1. ด้านการเพาะเลี้ยงแมลง โดยจะออกแบบและพัฒนาระบบตรวจจับสภาพแวดล้อมภายในกล่องจำลองการเลี้ยงแมลงที่ส่งข้อมูลแบบไร้สายไปยังเว็บเซิร์ฟเวอร์บน Raspberry Pi4 เพื่อจัดทำจอแสดงผลรายงานข้อมูลเกี่ยวกับอุณหภูมิ ความชื้น ค่าความเป็นกรดเบส แก๊สไฮโดรเจนซัลไฟด์ แก๊สคาร์บอนไดออกไซด์ แก๊สออกซิเจน โดยข้อมูลที่ได้จะถูกเก็บในระบบ

ฐานข้อมูลเพื่อประเมินและวิเคราะห์สภาพอากาศภายในกล่อง 2. ด้านการกำจัดของเสียทางอากาศ โดยเมื่อมีปริมาณของแก๊สที่มากกว่าค่าที่กำหนดไว้ระบบพัดลมระบายอากาศจะทำงานโดยส่วนนี้จะทำงานแบบอัตโนมัติตัดสินใจผ่านชุดข้อมูลที่ได้รับ โดยระบบการทำงานรวมช่วยปรับปรุงความแม่นยำจากข้อมูลที่ได้รับ และมีแจ้งเตือนทางผ่านไลน์โนติเมื่อค่าข้อมูลสภาพแวดล้อมในกล่องจำลองการเลี้ยงแมลงผิดปกติ

1.2 วัตถุประสงค์

1) เพื่อออกแบบและพัฒนาระบบการตรวจจับสภาพแวดล้อมในกล่องจำลองการเลี้ยงแมลง
2) จัดทำเว็บแสดงผลรายงานสภาพแวดล้อมแบบเรียลไทม์ของระบบผ่านแพลตฟอร์ม Thingsboard พร้อมอกแบบระบบจัดเก็บฐานข้อมูลในรูปแบบออนไลน์ และออฟไลน์
3) ออกแบบระบบพัดลมระบายอากาศแก่กล่องจำลองการเลี้ยงแมลงเมื่อได้รับแจ้งเตือนว่าปริมาณข้อมูลภายในอากาศสูงเกินที่กำหนด
4) เพื่อเพิ่มประสิทธิภาพในการทำงานของกล่องจำลองเลี้ยงแมลง รวมถึงช่วยประเมินและวิเคราะห์ของข้อมูล
5) ออกแบบระบบตรวจวัดระดับของแหล่งจ่ายไฟ เพื่อประเมินการใช้งานของแหล่งพลังงาน

1.3 ขอบเขตของปริญญานิพนธ์

1) ออกแบบการจัดวางเซนเซอร์ภายในกล่องจำลองเลี้ยงแมลงเพื่อตรวจจับสภาพแวดล้อม อาทิเช่น อุณหภูมิ ความชื้นในดิน และในอากาศ ค่าความเป็นกรดด่าง แก๊สไฮโดรเจนซัลไฟด์ แก๊สคาร์บอนได้ออกไซด์ และแก๊สออกซิเจนภายในกล่อง
2) สร้างเว็บจอแสดงผลที่รายงานข้อมูลการตรวจจับสภาพแวดล้อม และระบบจัดเก็บฐานข้อมูลเพื่อนำไปวิเคราะห์ โดยเมื่อค่าข้อมูลที่ได้รับจากกล่องจำลองเลี้ยงแมลงสูงเกินกว่าที่กำหนดจะทำการแจ้งเตือนผ่านแพลตฟอร์ม ThingsBoard
3) เมื่อปริมาณของแก๊สอากาศภายในกล่องจำลองมีค่าสูงเกินกว่าที่กำหนดระบบพัดลมระบายอากาศจะเริ่มทำงานโดยใช้ระบบการตัดสินใจบนแพลตฟอร์ม ThingsBoard เพื่อส่งค่าข้อมูลกลับไปสั่งงานบน Arduino WROOM ESP32

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

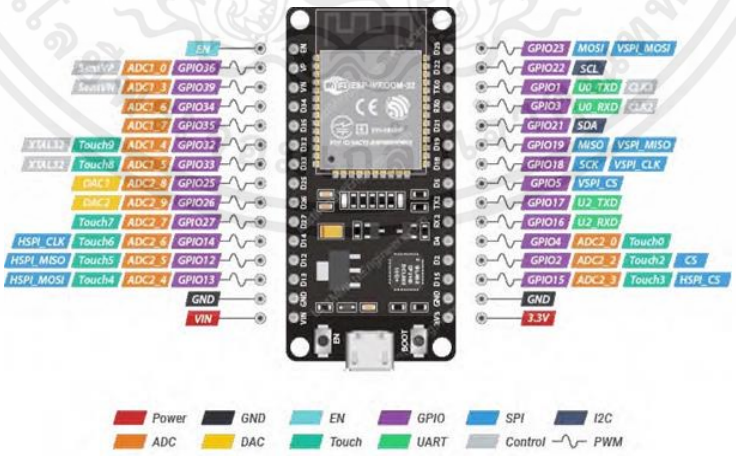
โครงงานเรื่องการออกแบบและพัฒนาการตรวจสอบสภาพแวดล้อมที่เหมาะสำหรับการเลี้ยงแมลงในระบบโรงเรือนโดยได้ทำการออกแบบระบบโรงเรือนให้อยู่ในรูปของกล่องจำลองการเลี้ยงแมลงซึ่งประยุกต์ใช้กับ ESP32 เพื่อส่งค่าไปยังแพลตฟอร์มบน Raspberry Pi4 เพื่อเชื่อมต่อกับผู้ใช้งาน โดยนำข้อมูลที่ได้มาจัดทำหน้าจอแสดงผลเพื่อรายงานข้อมูล และมีระบบฐานข้อมูล เมื่อค่าข้อมูลที่ได้รับมีค่าสูงกว่าที่กำหนดจะมีระบบแจ้งเตือนผ่านอีเมล ดังนั้นโครงงานที่นำเสนอจึงมีหลักการที่เกี่ยวข้อง ดังต่อไปนี้

2.1 ทฤษฎีที่เกี่ยวข้องกับฮาร์ดแวร์ภายในระบบ

2.1.1 ESP-WROOM-32

ESP-WROOM-32 เป็นชิป Wi-Fi และ Bluetooth ความเร็ว 2.4 GHz ออกแบบด้วยเทคโนโลยี 40 นาโนเมตรที่ใช้พลังงานต่ำของ TSMC ได้รับการออกแบบมาเพื่อให้ได้พลังงานและประสิทธิภาพ RF ที่ดีที่สุด

โมดูล Wi-Fi ESP-32 รุ่น ESP-WROOM-32 โมดูลWi-Fi + Bluetooth 4.2 + Touch/Temp Sensor ทำงานแบบ Dual Core ที่ความเร็ว 160 MHz มี มีขา GPIO 36 ขา ความละเอียดในการอ่านค่า ADC 12Bit แสดงได้ดังรูปที่ 2.1

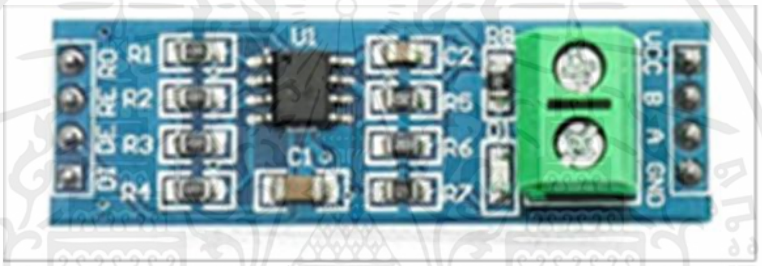


รูปที่ 2.1 ESP-WROOM-32

ESP-WROOM-32 มีคุณสมบัติดังนี้ มีมาตรฐานเครือข่ายไร้สาย 802.11 b/g/n ทำให้สามารถเชื่อมต่อกับเครือข่าย Wi-Fi ได้โดยรอบแบนด์วิดธ์ 2.4GHz มีการรองรับ Wi-Fi มัลติมีเดีย (WMM) ช่วยเรื่องการสื่อสารไร้สาย มีการลดการสูญเสียข้อมูลและเพิ่มความเร็วในการสื่อสารโดยการรวม Wi-Fi TX/RX A-MPDU และ RX A-MSDU มีการบล็อก ACK ทันทีเพื่อช่วยลดการส่งคำตอบยืนยัน และ รองรับโหมดการทำงาน SoftAP, Infrastructure Station และโหมด Promiscuous

2.1.2 MAX485 Module

เป็นโมดูลแปลงสัญญาณระดับ TTL เป็นสัญญาณ RS485 ใช้สำหรับการควบคุมหรือการสื่อสารระยะไกลๆ



รูปที่ 2.2 โมดูล MAX485

ชิป MAX485 แบบออนบอร์ด เป็นตัวรับส่งสัญญาณที่ใช้พลังงานต่ำที่ใช้สำหรับการสื่อสาร RS-485 ทำงานที่แหล่งจ่ายไฟ +5 โวลต์ และกระแสไฟที่กำหนดคือ 300 แอมแปร์ การใช้การสื่อสารแบบ half-duplex เพื่อใช้ฟังก์ชันการแปลงระดับ TTL เป็นระดับ RS 485 ทำให้สามารถบรรลุอัตราการส่งข้อมูลสูงสุดที่ 2.5Mbps ตัวรับส่งสัญญาณ MAX485 ดึงกระแสไฟจ่ายระหว่าง 120 μA และ 500 μA

2.1.3 Waterproof Temperature Humidity Sensor RS485

เป็นเซนเซอร์วัดอุณหภูมิ และความชื้นในอากาศ ทำงานโดยใช้ไฟเลี้ยง 12-24 โวลต์ บน Modbus RS485 โดยตั้งค่า Address ตั้งแต่ 1 ถึง 247 สามารถวัดอุณหภูมิได้ตั้งแต่ -30 ถึง 80 องศาเซลเซียส มีค่าคลาดเคลื่อนอยู่ที่ ± 5 % โดยการให้ค่าอินพุตเข้าระบบแต่ละครั้งน้อยกว่า 15 วินาที สามารถกันน้ำและฝุ่น ในมาตรฐาน IP67 แสดงได้ดังรูปที่ 2.3

รูปที่ 2.3 เซนเซอร์ Waterproof Temperature Humidity RS485

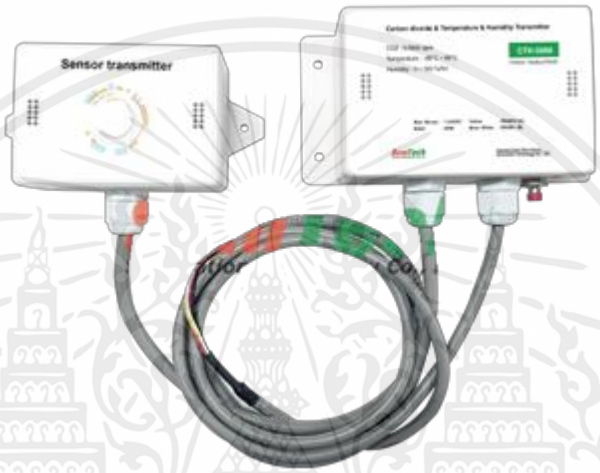
2.1.4 H2S Hydrogen Sulfide Gas Sensor RS485

แก๊สไฮโดรเจนซัลไฟด์ หรือแก๊สไข่เน่า (H2S) ไม่มีสีและเป็นพิษ เป็นแก๊สไวไฟมีกลิ่นเน่าเหม็นคล้ายไข่เน่า โดยเซ็นเซอร์แก๊สไฮโดรเจนซัลไฟด์ มีช่วงในการวัด 0 - 100 ppm ค่าความผิดพลาดในการวัดอยู่ที่ ± 3 % เวลาตอบสนองมีค่าน้อยกว่า 15 วินาที ทำงานโดยใช้ไฟเลี้ยง 12-24 โวลต์ เซนเซอร์ทำงานในอุณหภูมิ -30 ถึง 50 องศาเซลเซียส ความชื้นในการใช้งานสภาพแวดล้อมอยู่ที่ 0 - 100% RH เซนเซอร์สามารถทำงานบนความดันอากาศในระหว่าง 0.9-1.1 atm ทำงานบน IP65 แสดงดังรูปที่ 2.4

รูปที่ 2.4 เซ็นเซอร์ H2S Hydrogen Sulfide Gas Sensor RS485

2.1.5 Carbon dioxide Sensor RS485

เป็นเซนเซอร์วัดค่าแก๊สคาร์บอนไดออกไซด์ในอากาศ (CO2) ทำงานโดยใช้ไฟเลี้ยง 12-24 โวลต์ บน Modbus RS485 โดยตั้งค่า Address ตั้งแต่ 1 ถึง 247 สามารถวัดอุณหภูมิได้ถึง 5000 ppm มีค่าคาดเคลื่อนอยู่ที่ ± 3 % โดยสามารถกันน้ำและฝุ่น ในมาตรฐาน IP65 แสดงได้ดังรูปที่ 2.5



รูปที่ 2.5 เซนเซอร์ Carbon dioxide Sensor RS485

2.1.6 Oxygen Sensor RS485

เป็นเซนเซอร์วัดค่าแก๊สออกซิเจนในอากาศ ซึ่งระดับออกซิเจนในอากาศ โดยปกติอยู่ที่ 20.9% ระดับออกซิเจนจะเปลี่ยนแปลงไปขึ้นอยู่กับระดับความสูง-ต่ำจากพื้นดิน ส่งข้อมูลแบบ RS485 ทำงานโดยใช้ไฟเลี้ยง 10-30 โวลต์ บน Modbus RS485 โดยตั้งค่า Address ตั้งแต่ 1 ถึง 247 มีช่วงการวัดที่ 0-30% โดยสามารถกันน้ำและฝุ่น ในมาตรฐาน IP65 แสดงได้ดังรูปที่ 2.6

รูปที่ 2.6 เซนเซอร์ Oxygen Sensor RS485

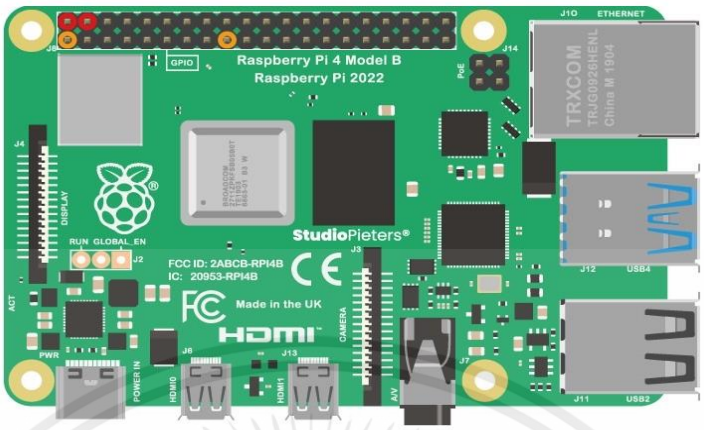
2.1.7 Soil Temperature Moisture PH and EC Sensor RS485

ใช้สำหรับวัดอุณหภูมิ ความชื้นของดิน PH และการนำไฟฟ้า (EC) ความแม่นยำสูง การตอบสนองที่รวดเร็ว และเสถียร ทนทาน และสามารถฝังอยู่ในดินระยะยาว โดยมีมาตรฐานอุตสาหกรรม IP68 ใช้ไฟเลี้ยง 4.5-30 โวลต์ บน Modbus RS485 โดยตั้งค่า Address ตั้งแต่ 1 - 247 สามารถวัดอุณหภูมิได้ตั้งแต่ -40 ถึง 80 องศาเซลเซียส วัดความชื้นได้ตั้งแต่ 0-100%RH วัด PHได้ตั้งแต่ 3-9 PH และวัดค่านำไฟฟ้าได้ตั้งแต่ 0-20000 us/cm มีค่าคาดเคลื่อนอยู่ที่ 5% โดยการให้ค่าอินพุตเข้าระบบแต่ละครั้งน้อยกว่า 15 วินาที แสดงได้ดังรูปที่ 2.7

รูปที่ 2.7 เซนเซอร์ Soil Temperature Moisture PH and EC Sensor RS485

2.1.8 Raspberry Pi 4 Model B

Raspberry Pi 4 Model B 8GB เป็นรุ่นพิเศษของคอมพิวเตอร์เล็กและ มี RAM จำนวน 8GB เพื่อให้มีความสามารถในการประมวลผลและรันแอปพลิเคชันที่มีขนาดใหญ่ขึ้นได้ โดยใช้ Broadcom BCM2711 Quad-Core ARM Cortex-A72 ความเร็ว 1.5 GHz เป็นหน่วยประมวลผลหลัก โดยมีขนาดของหน่วยความจำ LPDDR4-2400 ขนาด 8 GB และบอร์ดใช้ชิพ Wireless LAN แบบ Dual-Band รองรับ 2.4 GH และ 5 GHz พร้อมรองรับ Bluetooth 5.0 BLE มีพอร์ต LAN รองรับ Gigabit Ethernet พอร์ต USB 3.0 Host Type A จำนวน 2 พอร์ต และ USB 2.0 Host Type A จำนวน 2 พอร์ต มีพอร์ต micro-HDMI จำนวน 2 พอร์ต รองรับการเชื่อมต่อจอ 4K60P แสดงได้ดังรูปที่ 2.8



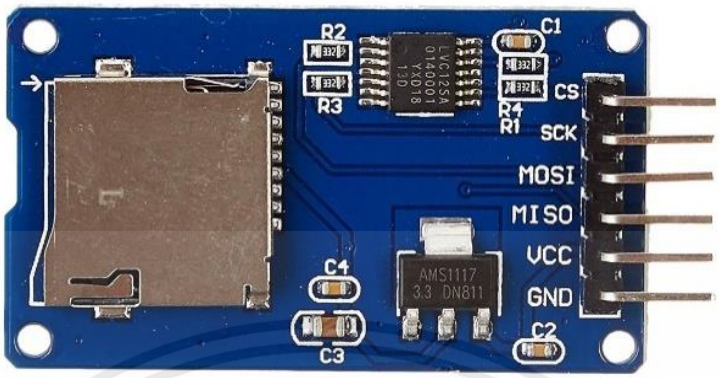
รูปที่ 2.8 Raspberry Pi 4 Model B

ตารางที่ 2.1 ตารางแสดง GPIO pinout ของ Raspberry Pi 4

Pin Type	GPIO Pins
PWM pins	GPIO12, GPIO13, GPIO18, GPIO19
SPI pins	SPI0: GPIO9 (MISO), GPIO10 (MOSI), GPIO11 (SCLK), GPIO8 (CE0), GPIO7 (CE1) SPI1: GPIO19 (MISO), GPIO20 (MOSI), GPIO21 (SCLK), GPIO18 (CE0), GPIO17 (CE1), GPIO16 (CE2)
I2C Pins	Data: (GPIO2), Clock: (GPIO3) EEPROM Data: (GPIO0), EEPROM Clock: (GPIO1)
UART Pins	TX: (GPIO14), RX: (GPIO15)

2.1.9 Micro SD Card Module

คืออุปกรณ์ที่ใช้สำหรับอ่านและเขียนข้อมูลบนการ์ดหน่วยความจำแบบ microSD ซึ่งเป็นรูปแบบขนาดเล็กของการ์ดหน่วยความจำแบบ SD (Secure Digital) เชื่อมต่อกับคอนโทรลเลอร์ผ่านสายพัฒนา (header) และควบคุมการอ่านและเขียนข้อมูลโดยสวิตช์หรือขาต่อเพื่อเปิดหรือปิด แสดงได้ดังรูปที่ 2.9



รูปที่ 2.9 โมดูล Micro SD Card

โมดูลประกอบด้วย 6 พิน สำหรับจ่ายไฟและสื่อสารกับคอนโทรลเลอร์ ด้านล่างอธิบายประเภทและบทบาทของแต่ละพิน

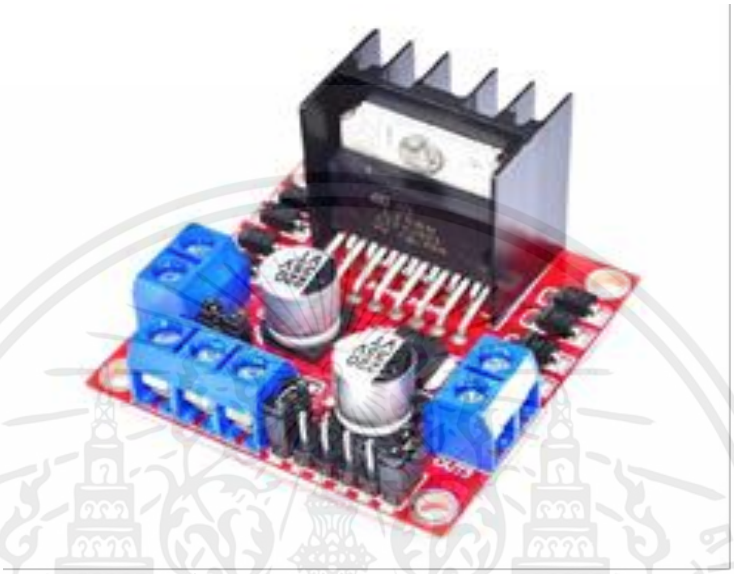
ตารางที่ 2.2 ตารางแสดงรายละเอียดพินของโมดูล Micro SD Card

Pin Type	Pin Description
GND	Ground
VCC	Voltage Input
MISO	Master In Slave Out (SPI)
MOSI	Master Out Slave In (SPI)
SCK	Serial Clock (SPI)
CS	Chip Select (SPI)

2.1.10 L298N Motor Driver Module

คืออุปกรณ์หรือวงจรอิเล็กทรอนิกส์ที่ใช้ในการควบคุมการเคลื่อนที่ของมอเตอร์ โดยปกติจะมีหน้าที่แปลงสัญญาณควบคุมจากคอนโทรลเลอร์เพื่อควบคุมมอเตอร์โดยมีความแม่นยำและ

ปลอดภัย จากแรงดันไฟฟ้าที่ไม่เหมาะสม โมดูลนี้ประกอบด้วย IC ไดรเวอร์มอเตอร์ L298 และตัวควบคุม 78M05 5V โมดูล L298N สามารถควบคุมมอเตอร์กระแสตรงได้สูงสุด 4 ตัว หรือมอเตอร์กระแสตรง 2 ตัวพร้อมการควบคุมทิศทางและความเร็ว แสดงได้ดังรูปที่ 2.10



รูปที่ 2.10 โมดูล L298N Motor Driver

2.1.11 พัดลมระบายอากาศ

PWM Fan หรือ Pulse Width Modulation Fan เป็นอุปกรณ์ที่ใช้เทคโนโลยี PWM เพื่อควบคุมความเร็วหมุนรอบของพัดลมได้อย่างมีประสิทธิภาพ โดยการควบคุมที่ใช้การเปิด-ปิดสัญญาณไฟฟ้าในช่วงเวลาสั้นๆ เพื่อควบคุมการทำงานของพัดลม จะทำงานด้วยสัญญาณพัลส์ที่เรียกว่า "Duty Cycle" โดยจะเปลี่ยนแปลงตามอัตราส่วนที่กำหนดไว้ซึ่งทำให้พัดลมหมุนรอบด้วยความเร็วที่ต่างกันไปตามอัตราส่วนนั้นๆ แสดงได้ดังรูปที่ 2.11

รูปที่ 2.11 พัดลมระบายอากาศ

2.1.12 Micro DC Motor

มอเตอร์ไมโคร DC เป็นอุปกรณ์ระบบเครื่องกลไฟฟ้าโดยใช้แรงดันไฟฟ้า 1.5 – 6 V ขนาดกะทัดรัดที่ออกแบบมาสำหรับการใช้งานขนาดเล็ก โดยทั่วไปแล้วจะขับเคลื่อนด้วยไฟฟ้ากระแสตรงแรงดันต่ำ (DC) ซึ่งจะแปลงพลังงานไฟฟ้าเป็นการเคลื่อนที่ทางกล มอเตอร์เหล่านี้มีลักษณะพิเศษคือมีขนาดเล็ก โครงสร้างน้ำหนักเบา และประสิทธิภาพสูง แสดงได้ดังรูปที่ 2.12

รูปที่ 2.12 Micro DC Motor

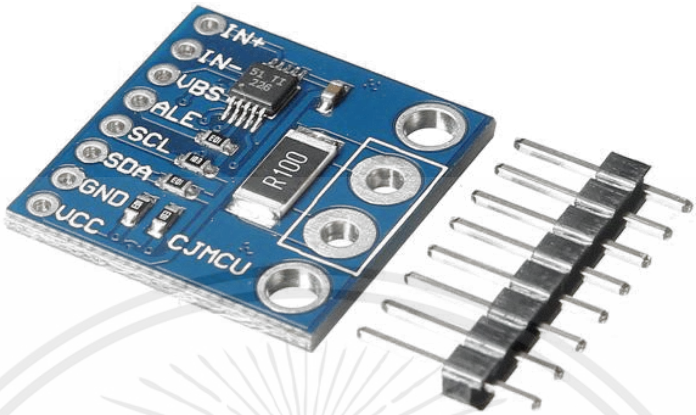
2.1.13 Lithium-Ion Battery 12V

แบตเตอรี่ลิเธียมไอออน 12V ประกอบด้วยหลายเซลล์ที่เชื่อมต่อกันแบบอนุกรม โดยแต่ละเซลล์มีแรงดันไฟฟ้าปกติที่ 3.6 - 3.7 V แบตเตอรี่เหล่านี้มีแหล่งพลังงานขนาดกะทัดรัดและน้ำหนักเบาพร้อมความหนาแน่นของพลังงานสูง โดยมีความจุเป็น 12 แอมแปร์-ชั่วโมง (Ah) มีอัตราการคายประจุเองค่อนข้างต่ำ และสามารถรองรับรอบการคายประจุได้หลายรอบ ทำให้เหมาะสำหรับอุปกรณ์อิเล็กทรอนิกส์แบบพกพาและการใช้พลังงานหมุนเวียน แสดงได้ดังรูปที่ 2.13

รูปที่ 2.13 Lithium-Ion Battery 12V

2.1.14 INA226

เป็นอุปกรณ์ตรวจสอบกระแสและพลังงานไฟฟ้าที่มีการเชื่อมต่อแบบ PCTM หรือ SMBUS ซึ่งสามารถตรวจสอบแรงดันไฟฟ้าตกและแรงดันไฟฟ้าของบัสได้ อุปกรณ์นี้สามารถอ่านค่ากระแสโดยตรงในหน่วยแอมแปร์และกำลังในวัตต์ และแรงดันไฟฟ้าบัสในโหมดทั่วไป โดยสามารถปรับเปลี่ยนแรงดันไฟฟ้าได้ตั้งแต่ 0 V ถึง 36 V โดยไม่ขึ้นอยู่กับแรงดันไฟฟ้าของแหล่งจ่ายไฟ อุปกรณ์นี้ทำงานบนแรงดันระหว่าง 2.7 V ถึง 5.5 V และใช้กระแสไฟจ่ายที่ระดับ 330 μA สามารถใช้งานในช่วงอุณหภูมิระหว่าง -40°C ถึง 125°C บนอินเทอร์เฟซที่รองรับ I^2C แสดงได้ดังรูปที่ 2.14



รูปที่ 2.14 INA226

2.1.15 DC Relay module

รีเลย์เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ทำหน้าที่เหมือนสวิตช์ไฟฟ้าในระบบควบคุมอัตโนมัติ โดยใช้แรงดันไฟฟ้าเพื่อเปิดและปิดอุปกรณ์ไฟฟ้าต่าง ๆ โดยการส่งกระแสไฟฟ้าผ่านขดลวด เมื่อกระแสไฟฟ้าไหลผ่านขดลวดนี้ เกิดพลังงานแม่เหล็กที่ทำให้ขดลวดดึงดูดหน้าสัมผัสเพื่อเปลี่ยนทิศทางการไหลของไฟฟ้า ซึ่งทำให้เป็นเหมือนสวิตช์ที่ควบคุมการไหลของไฟฟ้าเพื่อควบคุมอุปกรณ์ไฟฟ้าต่าง ๆ ในวงจรอัตโนมัติได้ แสดงได้ดังรูปที่ 2.15

รูปที่ 2.15 DC Relay module

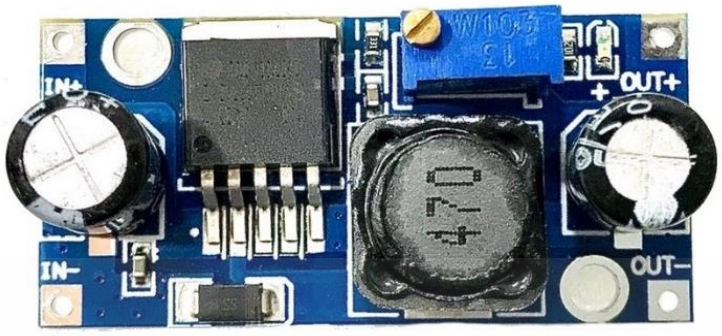
2.1.16 Limit switch

Limit switch เป็นอุปกรณ์ที่ใช้ในแอปพลิเคชันต่าง ๆ เพื่อตรวจจับการเคลื่อนไหวของวัตถุหรือชิ้นงานในสถานการณ์ที่มีความจำเป็นในการควบคุม และจำกัดการเคลื่อนของอุปกรณ์ที่ต้องการ โดยมีการส่งค่า logic ของสถานะของอุปกรณ์นี้สองสถานะ คือ ON (เปิด) และ OFF (ปิด) ผ่านสัญญาณไฟฟ้าแบบดิจิทัล แสดงได้ดังรูปที่ 2.16

รูปที่ 2.16 Limit switch

2.1.17 LM2596 DC Voltage Regulator

อุปกรณ์นี้ทำหน้าที่แปลงระดับแรงดันไฟฟ้าตรง จาก 4-35 V ให้อยู่ในช่วง 1.25-30 V โดยสามารถปรับค่าแรงดัน output ได้โดย Potentiometer ที่มีอยู่บนบอร์ด สามารถจ่ายกระแสได้ถึง 3A และใช้หลักการแปลงโดยวงจร Buck Converter ความถี่ Swithching 150 kHz ทำให้ทำงานเงียบ และแรงดันเรียบ แสดงได้ดังรูปที่ 2.15



รูปที่ 2.17 LM2596 DC Voltage Regulator

2.1.18 มาตรฐาน IP

มาตรฐาน IP ชื่อเต็ม International Protection Standard คือมาตรฐานที่บอกถึงระดับการป้องฝุ่นและน้ำของเครื่องจักร (mechanical casings) และอุปกรณ์ไฟฟ้า (electrical enclosures) ซึ่งถูกพัฒนาขึ้นโดย IEC (International Electrotechnical Commission)

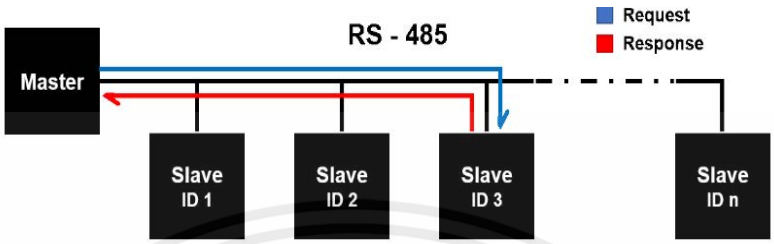
การบอกถึงระดับการป้องกันนั้นแสดงโดยตัวเลข 2 หลักคือ IPXX โดยหลักแรกจะหมายถึงระดับการป้องกันของฝุ่นหรือการสัมผัสโดยบังเอิญ ซึ่งจะมีระดับตั้งแต่ 0-6 ส่วนหลักที่สองจะหมายถึงระดับการป้องกันน้ำ ซึ่งจะมีระดับตั้งแต่ 0-9 ซึ่งความหมายของรหัส IP ตัวอื่นๆจะถูกเขียนไว้ในตารางที่อยู่ด้านล่าง

ตารางที่ 2.3 ตารางแสดงรายละเอียดรหัส IP

สัญลักษณ์ IP	การป้องกันของแข็ง	การป้องกันของเหลว	การทนต่อแรงกระแทก	การป้องกันอื่นๆ
IP	ตัวเลข 0-6	ตัวเลข 0-9	ตัวเลข 0-9	ตัวอักษร
จำเป็นต้องระบุ	จำเป็นต้องระบุ	จำเป็นต้องระบุ	ยกเลิกการใช้แล้ว	ไม่จำเป็นต้องระบุ

2.2 โปรโตคอลที่ใช้ในโครงงาน

2.2.1 Modbus RTU



รูปที่ 2.18 การสื่อสารแบบอนุกรมด้วย RS – 485 สำหรับ Modbus RTU

Modbus RTU จะใช้การสื่อสารในระดับกายภาพ (Physical Layer) Modbus RTU ใช้ serial communication แบบ asynchronous สำหรับการสื่อสารผ่านสายสัญญาณแบบ RS-232 หรือ RS-485 ข้อมูลในโพรโทคอล Modbus จะถูกเก็บ 4 รูปแบบ คือ Output coils, Input contacts, Input registers, Holding registers โดย Output coils และ Input contacts แต่ละแอดเดรสจะเก็บค่าเพียง 1 บิต หรือมีค่าได้แค่ “0” กับ “1” เปรียบเสมือนค่าการเปิดและปิดของอุปกรณ์รีเลย์และสวิตช์ที่พบได้ในระบบงานอัตโนมัติอุตสาหกรรม ในขณะที่ Input registers และ Holding registers สามารถเก็บค่าเป็นตัวเลขได้ถึง 16 บิต เปรียบเสมือนค่าที่มาจากอุปกรณ์ตรวจวัดที่ส่งข้อมูลแบบอนาล็อก (Analog)

การสื่อสารของข้อมูลในระบบ Modbus RTU จะรับส่งเป็นชุดข้อมูล โดยที่ใน 1 ชุดข้อมูลนั้นจะประกอบด้วยส่วน 6 ส่วน ดังแสดงในรูปที่ 2.17

Field Name	Bit length	Function
Start	28	At least 3.5 character times of silence (mark condition)
Address	8	Station address
Function	8	Indicates function code eg. read coils/holding registers
Data	n x 8	Data + length will be filled depending on message type
CRC	16	Cyclic Redundancy Check
End	28	At least 3.5 character times of silence between frames

รูปที่ 2.19 ชุดข้อมูลสำหรับการสื่อสาร Modbus RTU

ฟังก์ชันการทำงานสำหรับ Modbus RTU (Function code) คือชุดฟังก์ชันการทำงานสามารถแบ่งหน้าที่ต่างๆ ได้ตามรหัส หรือ Function code รายละเอียดแสดงดังรูปที่ 3 โดยหลักๆ แล้วจะมีฟังก์ชันการทำงานอยู่ 2 แบบ คือ การอ่าน (Read) และเขียน (Write) โดยสามารถเลือกที่จะอ่านหรือเขียนข้อมูลไปยัง Coils หรือ Contacts สำหรับข้อมูลแบบดิจิตอล (Digital) หรือ “0” กับ “1” และ Registers สำหรับอ่านหรือเขียนข้อมูลแบบอนาล็อก โดยมีขนาด 16 บิต หรือตั้งแต่ 0000 ถึง FFFF

2.2.2 TTL

TTL (Transistor-Transistor Logic) คือ การออกแบบลอจิกดิจิทัลที่ทรานซิสเตอร์สองขั้วทำหน้าที่กับพัลส์กระแสตรง โดยทั่วไปแล้วลอจิกเกต TTL จำนวนมากจะถูกสร้างขึ้นบนวงจรรวม (IC) มีลักษณะการส่งสัญญาณแบบดิจิทัลในระดับแรงดันต่ำ (low voltage level) และแรงดันสูง (high voltage level) ซึ่งในวงจร TTL จะมีการใช้ทรานซิสเตอร์เพื่อเปลี่ยนสัญญาณจากดิจิตอลเป็นรูปแบบแรงดันไฟฟ้าเพื่อที่เชื่อมกับไมโครคอนโทรลเลอร์

ข้อดี

1) ใช้งานง่าย และราคาถูก

2) ใช้งานกับไมโครคอนโทรลเลอร์ได้หลายแบบ

3) สัญญาณรบกวนต่ำ

ข้อเสีย

1) จำกัดความเร็ว และความถี่

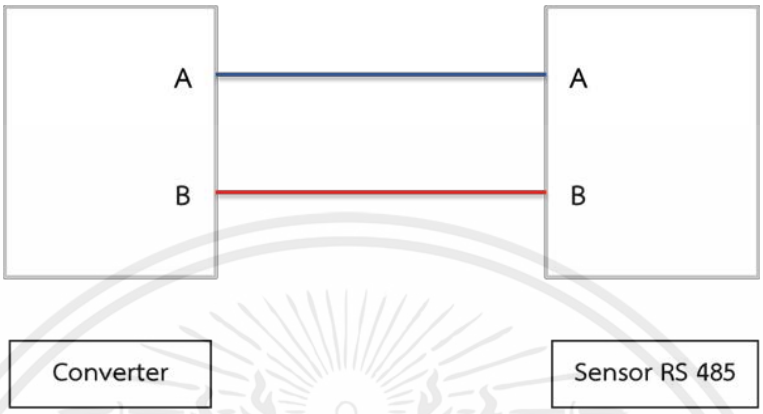
2) การใช้พลังงานที่สูงขึ้นที่ความถี่ที่สูงขึ้น

2.2.3 RS485

RS485 หรือ Recommended Standard no. 485 เป็นมาตรฐานการสื่อสารข้อมูลดิจิตอลแบบอนุกรม ถูกใช้อย่างแพร่หลาย ในโรงงานอุตสาหกรรม เนื่องจากสามารถส่งสัญญาณได้ไกลและยังสามารถส่งพร้อมๆ กันได้หลายจุดมาตรฐาน

RS485 เป็นมาตรฐานที่รับ และส่งข้อมูลในแบบที่เรียกว่า Half duplex ที่สามารถรับและส่งข้อมูลได้ ทีละอย่างเท่านั้นไม่สามารถทำทั้งสองอย่างได้ในเวลาเดียวกัน โดยการรับ และส่ง

ข้อมูลดิจิตอลแบบ RS485 นั้นจะส่งข้อมูลโดยใช้สายไฟเพียงแค่ 2 เส้นคือ A และ B เป็นตัวบอกค่ารหัสดิจิตอล (Digital code) แสดงได้ดังรูปที่ 2.18



รูปที่ 2.20 การเชื่อมต่อเซนเซอร์ 485 เข้ากับตัวแปลงสัญญาณ

โดยใช้ความแตกต่างของแรงดันไฟฟ้าระหว่างขั้ว A และ B เป็นตัวบอกดังนี้ เมื่อ Va - Vb ได้แรงดันไฟฟ้าน้อยกว่า -200 mV คือสัญญาณดิจิตอลเป็น 1 เมื่อ Va - Vb ได้แรงดันไฟฟ้ามากกว่า +200 mV คือสัญญาณดิจิตอลเป็น 0

ข้อดีของสัญญาณ RS485

1) สามารถส่งสัญญาณได้ไกลถึง 1200 เมตร
2) เชื่อมต่อเป็นเครือข่ายได้ในปริมาณมากแบบอนุกรม
3) สามารถสื่อสารกับคอมพิวเตอร์, PLC, HMI
4) รองรับการสื่อสารได้ทั้งแบบ MODBUS RTU หรือ MODBUS ASCII

ข้อเสียของสัญญาณ RS485

1) ต้องใช้ตัวแปลงสัญญาณในการเชื่อมต่อกับคอมพิวเตอร์
2) ความเร็วในการรับส่งข้อมูลลดลงเมื่อเชื่อมต่อเครือข่ายขนาดใหญ่

2.2.4 MQTT

โปรโตคอล MQTT (Message Queuing Telemetry Transport) คือโปรโตคอลสื่อสารที่ใช้หลักการแบบเผยแพร่/สมัครรับ (Publish/Subscribe) เพื่อการสื่อสารผ่านเครือข่ายแบบดั้งเดิม โดยมีองค์ประกอบหลักคือผู้เผยแพร่ (Publisher) และผู้สมัครรับ (Subscriber) ที่

สื่อสารผ่านโบรกเกอร์ข้อความ (Message Broker) ซึ่งมีหน้าที่จัดการการส่งข้อมูลระหว่างผู้เผยแพร่และผู้สมัครรับเพื่อให้การสื่อสารเกิดขึ้นอย่างมีประสิทธิภาพและเป็นระเบียบ แสดงได้ดังรูปที่ 2.19

รูปที่ 2.21 การเชื่อมต่อโปรโตคอล MQTT

ตารางที่ 2.4 สรุปแนวความคิดและหลักการทำงานของ MQTT

เผยแพร่ (Publish)	เป็นอุปกรณ์หรือแอปพลิเคชันที่สร้างข้อมูลที่ต้องการส่งไปยังผู้สมัครรับ
	ผู้เผยแพร่ส่งข้อมูลดังกล่าวไปยังโบรกเกอร์ข้อความโดยระบุหัวข้อ (Topic) ของข้อมูล
สมัครรับ (Subscribe)	ผู้สมัครรับเป็นอุปกรณ์หรือแอปพลิเคชันที่สนใจข้อมูลในหัวข้อ (Topic) ที่เปิดให้สมัครรับ
	สมัครรับการแจ้งเตือนหรือข้อมูลจากหัวข้อที่ตนสนใจจากโบรกเกอร์ข้อความ
โบรกเกอร์ข้อความ (Message Broker)	เป็นตัวกลางที่รับข้อมูลจากผู้เผยแพร่และกระจายไปยังผู้สมัครรับข้อมูลที่สนใจ
	มีหน้าที่ตรวจสอบและจัดการข้อมูลที่เผยแพร่ และส่งให้ผู้ที่ร้องขอ

โดยทั่วไปแล้ว หน้าที่ของ MQTT คือให้การสื่อสารแบบเผยแพร่/สมัครรับอย่างมีประสิทธิภาพ และการแยกแยะข้อมูลที่ส่งไปมาระหว่างผู้เผยแพร่และผู้สมัครรับในแต่ละหัวข้อ (Topic) เพื่อให้แต่ละอุปกรณ์หรือแอปพลิเคชันเข้าถึงข้อมูลที่ต้องการได้อย่างถูกต้อง การใช้งาน

MQTT มีประโยชน์ในระบบ IoT (Internet of Things) และแอปพลิเคชันที่ต้องการการสื่อสารแบบเรียลไทม์อย่างมีประสิทธิภาพในระยะไกลผ่านเครือข่ายอินเทอร์เน็ต

2.2.5 REST API

API หรือ Application Program Interface เป็นหนึ่งกลไกที่รวม 2 ส่วนเข้าด้วยกัน การใช้ API อยู่ในรูปแบบของ ‘ผู้ให้บริการ (Servers)’ กับ ‘ผู้ใช้บริการ (Clients)’ โดยฝ่ายที่ส่งคำขอเป็นผู้ใช้บริการ ขณะที่ฝ่ายที่ตอบรับคำขอเป็นผู้ให้บริการ ยกตัวอย่าง เช่นการดูพยากรณ์อากาศ ผู้ให้บริการ คือซอฟท์แวร์ที่รวบรวมข้อมูลสภาพอากาศ และผู้ขอใช้บริการที่อาจจะเป็นการดูพยากรณ์ผ่านโทรศัพท์ หรือการที่รายงานผ่านเว็บบนคอมพิวเตอร์ โดย API มี 4 รูปแบบ คือ

1.) SOAP API (Simple Object Access Protocol)

ผู้ให้บริการ และผู้ใช้บริการจะแลกเปลี่ยนข้อความโดยใช้ XML ซึ่งเป็น API ที่มีความยืดหยุ่นน้อย

2.) RPC API (Remote Procedure Call)

ผู้ใช้บริการจะขอการใช้งานฟังก์ชันหนึ่งๆ ที่ผู้ให้บริการ และผู้ให้บริการส่งผลลัพธ์กลับไปยังผู้ขอใช้บริการ

3.) WebSocket API

Web API สามารถรองรับ JSON ในการส่งข้อมูลระหว่างผู้ให้บริการ และผู้ใช้บริการ จึงทำให้มีประสิทธิภาพมากกว่า REST API

4.) REST API (Representational State Transfer)

ผู้ให้บริการจะส่งค่า เช่น GET, PUT, DELETE ฯลฯ ที่ผู้ใช้บริการสามารถใช้เพื่อเข้าถึงข้อมูลได้ โดยทั้งสองจะแลกเปลี่ยนข้อมูลโดยใช้ HTTP

2.2.6 SMTP

SMTP หรือ Simple Mail Transfer Protocol คือ Protocol แบบ TCP/IP ที่ใช้ในการส่งอีเมลในเครือข่ายอินเทอร์เน็ต ไปยังเครื่องบริการอื่นๆ ซึ่งสามารถส่งอีเมลไปยังผู้ใช้ได้ทั่วโลก มีข้อจำกัดในเรื่องของความสามารถในการส่งอีเมล ว่าสามารถทำได้แบบเป็นคิวเท่านั้น และ SMTP ส่วนใหญ่จะไม่ยอมให้คนนอกองค์กร หรือ IP ที่อยู่นอกองค์กรใช้งาน SMTP

2.3 โปรแกรมที่ใช้ภายในโครงงาน

2.3.1 Arduino IDE

Arduino IDE เป็นแพลตฟอร์มอิเล็กทรอนิกส์แบบโอเพนซอร์สบนพื้นฐานของฮาร์ดแวร์ และซอฟท์แวร์ที่ง่ายต่อการใช้งาน สามารถนำไปประยุกต์ใช้งานได้หลายอย่าง เช่น ควบคุมอุปกรณ์อิเล็กทรอนิกส์ขนาดเล็ก อ่านค่าเซ็นเซอร์วัดสภาพแวดล้อมต่างๆแล้วแสดงค่าที่เซ็นเซอร์สามารถอ่านได้ออกมาทางจอแสดงผล นำไปประยุกต์เข้าเป็นชิ้นงานทางอิเล็กทรอนิกส์เพื่ออำนวยความสะดวกในการใช้ชีวิตประจำวัน เป็นต้น

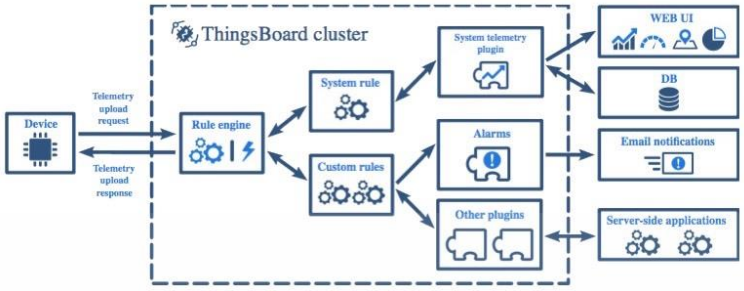
ปัจจุบัน Arduino ถือได้ว่าเป็นแพลตฟอร์มที่ได้รับความนิยมสูงจากทั่วโลก เนื่องมาจากราคาของตัวบอร์ด Arduino ไม่ค่อยสูงมาก เป็นโอเพนซอร์สทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์ คอมมิวนิตี้และฟอรัมน์ในการถามตอบเรื่องเกี่ยวกับการใช้งานร่วมกับอุปกรณ์อิเล็กทรอนิกส์ต่างๆ ตัวอย่างโปรแกรมเบื้องต้น และไลบรารี่สำหรับใช้งาน

2.3.2 PostgreSQL

PostgreSQL เป็นระบบการจัดการฐานข้อมูลแบบ ORDBMS (Object-Relational Database Management System) ที่สามารถใช้รูปแบบคำสั่งของภาษา SQL เกือบทั้งหมดได้. ระบบฐานข้อมูลแบบนี้นั้นเชื่อมโยงระบบการจัดการฐานข้อมูลเชิงวัตถุและระบบการจัดการฐานข้อมูลเชิงสัมพันธ์อย่างลงตัว ทำให้สามารถจัดเก็บข้อมูลแบบวัตถุและสร้างความสัมพันธ์ระหว่างข้อมูลได้อย่างยืดหยุ่น และรองรับการทำงานบนระบบปฏิบัติการที่หลากหลาย

2.3.3 ThingsBoard

ThingsBoard คือ IoT platform ตัวหนึ่งที่เป็น open source รองรับการเชื่อมต่อของ Devices ผ่าน MQTT CoAP และ HTTP อีกทั้งยังสามารถ integrate service พวก AWS IoT, Azure IoT และอื่นๆเข้ามาใช้งานได้ด้วย ในส่วนของการติดตั้ง support ตั้งแต่ Raspberry Pi



รูปที่ 2.22 การการประมวลผลข้อมูลของ ThingsBoard

จากรูปที่ 2.20 ใช้กฎกลไกระบบประมวลผลที่เชื่อถือได้และปรับขนาดได้ เปิดได้อย่างสะดวกเพื่อใช้ในการประมวลผลข้อมูลหรือส่งข้อมูลไปยังระบบภายนอกเพื่อการประมวลผลเพิ่มเติม แพลตฟอร์มนี้เป็นแบบอินเทอร์แอคทีฟและอนุญาตให้บุคคลและบริษัทอื่นๆ เชื่อมต่ออุปกรณ์ IoT ของตน และใช้คุณสมบัติที่เป็นเอกลักษณ์

2.3.4 Modbus poll

Modbus Poll เป็นโปรแกรมสำหรับใช้ติดต่อสื่อสารระหว่างคอมพิวเตอร์ PC กับอุปกรณ์ที่ใช้การสื่อสารผ่าน Modbus RTU ทางพอร์ตสื่อสารแบบ RS485 เป็นซอฟต์แวร์จำลองต้นแบบ Modbus (Modbus master simulator) Modbus slave หรืออื่น ๆ ที่ต้องการทดสอบและจำลองโปรโตคอล Modbus สามารถตรวจสอบ Modbus slaves และพื้นที่ข้อมูลหลาย ๆ ตัวได้ในเวลาเดียวกัน

2.3.5 SketchUp

SketchUp คือโปรแกรมสำหรับออกแบบโมเดล 3 มิติที่ได้รับความนิยมอย่างมากสำหรับการสร้างสถาปัตยกรรม วิศวกรรม การตกแต่งภายใน ไปจนถึงการออกแบบผลิตภัณฑ์ต่างๆ ถูกพัฒนาขึ้นมาเพื่อให้คนทั่วไปสามารถเข้าถึงการสร้างโมเดล 3 มิติได้ง่ายขึ้น มีฟังก์ชันพื้นฐานต่างๆ ครบถ้วน ใช้งานได้สะดวกสบายและเข้าใจง่าย ใช้ทรัพยากรเครื่องค่อนข้างน้อย

2.3.6 EasyEDA

EasyEDA เป็นเครื่องมือสำหรับงานด้าน Electronic Design Automation (EDA) ที่ใช้ออกแบบวงจรพิมพ์ PCB ทำงานออนไลน์บนเว็บเบราว์เซอร์ โดยขณะที่ออกแบบนั้นจำเป็นต้อง

เชื่อมต่อกับอินเตอร์เน็ต ผู้ใช้งานสามารถใช้งานโปรแกรมผ่านทางโปรแกรม google chrome, Microsoft Edge หรือโปรแกรมอื่น ๆ ที่ใช้เปิดเว็บ และยังสามารถดาวน์โหลดตัวโปรแกรมใช้งานมาติดตั้งบนเครื่องเพื่อทำงานได้เช่นกัน

2.4 ภาษาที่ใช้ภายในโครงงาน

2.4.1 C++

เป็นภาษาคอมพิวเตอร์เพื่อวัตถุประสงค์ทั่วไป ซึ่งสามารถเขียนโปรแกรมได้ทั้งแบบออบเจ็ค และการเขียนแบบปกติทั่วไป เป็นภาษาในการเขียนโปรแกรม ซึ่งถูกพัฒนาโดย Dr.Bjarne Stroustrup และยังมีเครื่องมืออำนวยความสะดวกในการจัดการและเข้าถึงระดับหน่วยความจำ นอกจากนี้มันยังถูกนำไปใช้ในการเขียนโปรแกรมแบบต่างๆ

1) ภาษาซีพลัสพลัสได้ถูกออกแบบมาเพื่อเป็นภาษาสำหรับการเขียนโปรแกรมทั่วไป สามารถรองรับการเขียนโปรแกรมในระดับภาษาเครื่องได้
2) ภาษาซีพลัสพลัสได้รับการออกแบบ เพื่อเข้ากันได้กับภาษาซีในเกือบทุกกรณี
3) มาตรฐานของภาษาซีพลัสพลัส ถูกออกแบบมา เพื่อไม่ให้มีการเจาะจงแพลตฟอร์มบนคอมพิวเตอร์
4) ภาษาซีพลัสพลัสถูกออกแบบมาให้รองรับรูปแบบการเขียนโปรแกรมที่หลากหลาย (multi-paradigm)

ข้อดี

1) ทำงานที่ค่อนข้างเร็วมากเมื่อเทียบกับภาษาอื่น และยังสามาถดำเนินการกับ Hardware ได้
2) เป็น Object Oriented Programming และStructure Programming

2.4.2 JavaScript

เป็นภาษาโปรแกรมที่ใช้ในการพัฒนาเว็บแอปพลิเคชันที่มีประสิทธิภาพและแบบอินเทอร์แอคทีฟตั้งแต่การรีเฟรชหน้าเว็บไซต์, การจัดการสื่อโซเชียล, การสร้างภาพเคลื่อนไหว, และแผนที่แบบอินเทอร์แอคทีฟ ภาษา JavaScript มีความสามารถในการปรับปรุงประสบการณ์ของผู้ใช้ในการใช้งานเว็บไซต์ และเป็นภาษาสคริปต์ฝั่งไคลเอ็นต์ที่สำคัญใน World Wide Web

ข้อดี

1) ทำงานที่ค่อนข้างเร็วมากเมื่อเทียบกับภาษาอื่น และยังสามาถดำเนินการกับ Hardware ได้
2) เรียนรู้และใช้งานง่าย
3) ได้รับความเป็นอิสระจากแพลตฟอร์ม
4) ลดโหลดของเซิร์ฟเวอร์
5) ปรับปรุงอินเทอร์เฟซผู้ใช้

2.4.3 VPL

VPL หรือ Visual Programming Language เป็นการเขียนโปรแกรมด้วยภาพ เป็นภาษาประเภทหนึ่งที่ใช้องค์ประกอบกราฟิก เช่น ไอคอน ปุ่ม และสัญลักษณ์ในรูปแบบของการเข้ารหัส ภาษาการเขียนโปรแกรมนี้ช่วยให้เห็นภาพแนวคิดการเข้ารหัสที่สร้างโดยคอมพิวเตอร์ ภาษาโปรแกรมประเภทนี้ช่วยให้ผู้ใช้ที่ไม่ใช่ด้านเทคนิคสามารถอธิบายแผนภูมิ และกระบวนการในลักษณะที่ผู้เริ่มต้นส่วนใหญ่สามารถเข้าใจได้ ภาษาการเขียนโปรแกรมแบบภาพยังช่วยให้ผู้ใช้สามารถใช้อินเทอร์เฟซแบบลาก และวางและทำงานได้อย่างมีประสิทธิภาพสูงสุดบนแพลตฟอร์มแบบ low code

2.5 วงจรชีวิตของแมลงกินได้

2.5.1 ด้วงสาคู หรือ ด้วงงวงมะพร้าว

ด้วงงวงมะพร้าว หรือด้วงสาคู (Red palm weevil) ชื่อวิทยาศาสตร์: Rhynchophorus ferrugineusเป็นด้วงงวงขนาดกลาง ตัวเต็มวัย ปีกมีสีน้ำตาลดำ อกมีสีน้ำตาลและมีจุดสีดำ มีขนาดลำตัวยาวประมาณ 25-28 มิลลิเมตร ทั้งตัวผู้และตัวเมียมีขนาดและลักษณะภายนอกคล้ายคลึงกัน ต่างกันที่ตัวผู้มีขนที่ด้านบนของงวงใกล้ส่วนปลาย ตัวหนอนมีสีเหลืองปนน้ำตาล ดักแด้เป็นปลอกทำด้วยเศษชิ้นส่วนจากพืชที่กินเป็นอาหาร ด้วงชนิดนี้ปัจจุบันเป็นแมลงเศรษฐกิจมีความสำคัญเป็นแมลงกินได้และแหล่งอาหารโปรตีน ประเทศไทยสามารถเลี้ยงด้วงสาคูได้ทุกภาค ทุกฤดูกาล สามารถเลี้ยงได้ทั้งแบบธรรมชาติและการเลี้ยงแบบประยุกต์ เช่นการเลี้ยงใน

กะละมัง โรงเรือน ตู้กระจก การใช้ทางมะพร้าวเลี้ยง ด้วงสาคูเพาะพันธุ์และเลี้ยงง่าย โตเร็ว ใช้เวลาเพียงเดือนกว่าก็สามารถเก็บเกี่ยวผลผลิตได้ ด้วงสาคูมีวงจรชีวิตอยู่ระหว่าง 150-259 วัน

2.5.1.1 การแบ่งวงจรชีวิตเป็น 4 ระยะ

1) ระยะไข่ 2-3 วันจะฟักมาเป็นตัวหนอน

2) ระยะตัวหนอน 35-39 วัน เป็นระยะเก็บผลผลิตด้วงสาคู

3) ระยะสร้างรัง (3-7 วัน)-อยู่ในรัง (6 วัน)

4) ระยะเข้าดักแด้ (9-10วัน) ออกดักแด้ (5-10 วัน)

5) ระยะตัวเต็มวัย มีอายุประมาณ 90-184 วัน

ด้วงสาคูมีแหล่งผลิตสำคัญทางภาคใต้เพราะมีสภาพอากาศเหมาะกับการเจริญเติบโตและการขยายพันธุ์เนื่องจากด้วงสาคูชอบอากาศแบบร้อนชื้น แต่สภาพแวดล้อมการเลี้ยงควรเป็นโรงเรือนมีหลังคาและตาข่ายมุ้งลวดโดยรอบโรงเรือน หรือเลี้ยงภายในระบบปิด

2.5.1.2 วิธีการเลี้ยงด้วงสาคู

1) เตรียมอาหารสำหรับด้วงสาคูนำกากน้ำตาล น้ำ และอาหารเลี้ยงสัตว์ตามอัตราส่วนที่ ผสมแช่น้ำเตรียมไว้ก่อน 20-30 นาที เพื่อให้อาหารเลี้ยงสัตว์ละลายเข้ากัน

2) นำพ่อแม่พันธุ์ด้วงสาคูปล่อยลงในท่อนสาคู อัตราตัวผู้ 2 ตัว ต่อตัวเมีย 4 ตัว จากนั้นปิดด้านบนของท่อนสาคูด้วยใยมะพร้าวสด

3) รดน้ำด้วยฝักบัว หรือสายยางรดน้ำ สัปดาห์ละ 2-3 ครั้ง ตั้งทิ้งไว้ประมาณ 40-45 วัน จะสามารถจับตัวหนอนด้วงสาคูออกจำหน่ายได้ โดยด้วงสาคูประมาณ 200 ตัว ให้ผลผลิตประมาณ 1-2 กิโลกรัม

2.5.1.3 ข้อควรระวังในการเลี้ยงด้วงสาคู

1) ควรเลี้ยงในพื้นที่ปิดเพื่อป้องกันตัวเต็มวัยเล็ดลอดไปในธรรมชาติ

2) ในโรงเรือนการเลี้ยงไม่ควรมีน้ำไม่ท่วมขัง และมีระบบการจัดการเรื่องความสะอาด

3) ภายในโรงเรือนไม่ควรเป็นพื้นที่แคบเพื่อสามารถใส่อาหารได้เพียงพอ หากไม่ได้รับสารอาหาร และความชื้นที่เพียงพอด้วงจะโตไม่เต็มที่ ตัวเล็กแคระแกรน และไม่ได้น้ำหนัก

4) ไม่ควรเปิดระบบเพื่อรบกวนการผสมพันธุ์ของด้วงสาคูภายใน 10-20 วันแรกของการเริ่มเลี้ยงด้วงสาคู

5) ควรวางโรงเรือนหรือภาชนะในการเลี้ยงด้วงสาคูให้พ้นจากแสงแดดและแมลงชนิดอื่นเพื่อป้องกันการบุกรุกด้วงสาคู

6) นำมาลวกน้ำร้อนก่อนนำไปผัดโดยไม่ต้องใส่น้ำมัน ให้น้ำมันสีดำๆ ในตัวหนอนด้วงออกให้หมด จนกว่าน้ำมันจากตัวหนอนด้วงเป็นสีใส แล้วนำไปล้างน้ำอีกครั้ง

2.5.2 ไหม

ไหม เป็นผีเสื้อกลางคืนชนิด Bombyx mori อยู่ในวงศ์ Bombycidae ตัวอ่อนเรียกว่า ตัวไหม หรือ หนอนไหม มีความสำคัญทางเศรษฐกิจในการปลูกหม่อนเลี้ยงไหม เนื่องจากมันสามารถให้เส้นใยเป็นเส้นไหม ผีเสื้อไหมไม่ปรากฏในป่าตามธรรมชาติ การสืบพันธุ์และดำรงชีวิตขึ้นอยู่กับการดูแลของมนุษย์เท่านั้น อาหารที่มันชอบก็คือใบหม่อนขาว ไหมมีวงจรชีวิตอยู่ระหว่าง 41-51 วัน

2.5.2.1 การแบ่งวงจรชีวิตเป็น 4 ระยะ

1) ระยะไข่ 10-12 เก็บไว้ในห้องที่ควบคุมอุณหภูมิ ความชื้น และแสง

2) ระยะตัวหนอน 19-25 จะแทะกินใบหม่อน หลังจากนั้นจึงลอกคราบ

3) ระยะเข้าดักแด้ 10-12 วัน

4) ระยะตัวเต็มวัย เมื่อผสมพันธุ์แล้วจะวางไข่ 400-500 ฟอง

2.5.2.2 วิธีการเลี้ยงไหม

1) เปิดกระดาษห่อไข่ไหม โรยสารโรยตัวไหมประมาณ 1 กรัมต่อตารางฟุต ทิ้งไว้ 10-15 นาที เพื่อป้องกันโรคไหม

2) โรยใบหม่อนที่พื้นที่ขนาด 0.5 x 0.5 ซ.ม. ประมาณ 40 กรัม ให้สม่ำเสมอ และทำการรักษาความชื้น

3) ให้อาหารไม่น้อยกว่าวันละ 3 มื้อโดยใช้หม่อนให้แก่ไหมตามช่วงอายุของไหม

2.5.2.3 ข้อควรระวังในการเลี้ยงไหม

1) ต้องระมัดระวังไม่ทำให้ใบหม่อนแห้งหรือเหี่ยว ในช่วงฤดูร้อน

2) ไม่เลี้ยงไหมวัยแก่ด้วยใบหม่อนที่อยู่บริเวณยอดอ่อนซึ่งมีความชื้นค่อนข้างสูง เพราะจะทำให้ระบบสรีระของหนอนไหมผิดปกติ หนอนไหมจะอ่อนแอและเชื้อโรคเข้าทำลายได้ง่าย

3) ปรับสภาพโรงเลี้ยงไหมให้เหมาะกับการเจริญเติบโตของหนอนไหม และระวังอย่าให้อุณหภูมิและ ความชื้นสูงซึ่งจะเหมาะกับการเจริญเติบโตของเชื้อโรค

4) ระมัดระวังมิให้มีแมลงศัตรูไหมติดใบหม่อนที่นำเข้าไปเลี้ยงไหม

2.5.3 จิ้งหรีด

จิ้งหรีด (Cricket) อยู่ในวงศ์ Gryllidae เป็นแมลงขนาดกลางถึงใหญ่ หนวดยาว ลักษณะปากเป็นแบบปากกัด มีตารวม ขาคู่หลังมีขนาดใหญ่และแข็งแรง เพศเมียปีกเรียวและมีอวัยวะวางไข่ยาวแหลมคล้ายเข็มยื่นออกมาจากส่วนท้อง เพศผู้มีปีกคู่หน้าย่นสามารถทำเสียงได้ การแบ่งวงจรชีวิตเป็น 3 ระยะ

2.5.3.1 การแบ่งวงจรชีวิตเป็น 3 ระยะ

1) ระยะไข่ 10-16 วัน เก็บไว้ในห้องที่ควบคุมอุณหภูมิ ความชื้น และแสง

2) ระยะตัวอ่อน 31-35 วัน หลังจากนั้นจึงลอกคราบ 8 ครั้ง

3) ระยะตัวเต็มวัย อายุตัวเต็มวัยของจิ้งหรีดประมาณ 45-50 วัน

2.5.3.2 วิธีการเลี้ยงจิ้งหรีด

1) หลังจากบ่มไข่แล้วเมื่อจิ้งหรีดเริ่มฟักตัวค่อยย้ายไปลงบ่อลูกจิ้งหรีดช่วงนี้จะขนาดตัวเล็กมากคล้ายๆ มด ต้องดูแลให้ดีทั้งการให้น้ำและอาหาร

2) เริ่มนำแผงไข่ที่เตรียมไว้ลงในบ่อ จิ้งหรีดจะลอกคราบประมาณ 8 ระยะ ช่วงลอกคราบจิ้งหรีดจะอ่อนแอมากและจะกินกันเอง ดังนั้นต้องจัดเตรียมน้ำและอาหารให้เพียงพอ เพิ่มหญ้าสดหรือหญ้าแห้งเพื่อเพิ่มพื้นที่หลบภัย

3) เมื่อถึงช่วงผสมพันธุ์จิ้งหรีดจะเริ่มส่งเสียงร้อง จิ้งหรีดเพศเมียจะเริ่มวางไข่ ภายใน 3-5 วันหลังจากผสมพันธุ์จิ้งหรีดจะกระวนกระวายหาที่วางไข่ หลังจากนั้นนำไข่ไปอบเพื่อเลี้ยง ส่วนจิ้งหรีดสามารถเก็บผลผลิตได้หรือจะเลี้ยงต่อเพื่อรองไข่อีกประมาณ 3-5 รอบ

2.5.3.3 ข้อควรระวังในการเลี้ยงจิ้งหรีด

1) อาหารสำหรับเลี้ยงจิ้งหรีดต้องไม่เสื่อมคุณภาพและไม่ส่งผลกระทบต่อสุขภาพของจิ้งหรีด

2) แหล่งน้ำที่ใช้ในฟาร์มต้องเป็นน้ำสะอาด ไม่ปนเปื้อนจากสิ่งที่เป็นอันตราย

3) โรงเรือนและอุปกรณ์ต้องสะอาด ถูกสุขลักษณะ มีการบำรุงรักษาโรงเรือนและอุปกรณ์ให้อยู่ในสภาพดี มีความปลอดภัยต่อจิ้งหรีด

4) กำจัดหรือจัดการขยะมูลฝอย ของเสีย และมูลจิ้งหรีดด้วยวิธีการเหมาะสมและถูกสุขลักษณะ เพื่อไม่ให้เกิดผลกระทบต่อสิ่งแวดล้อม

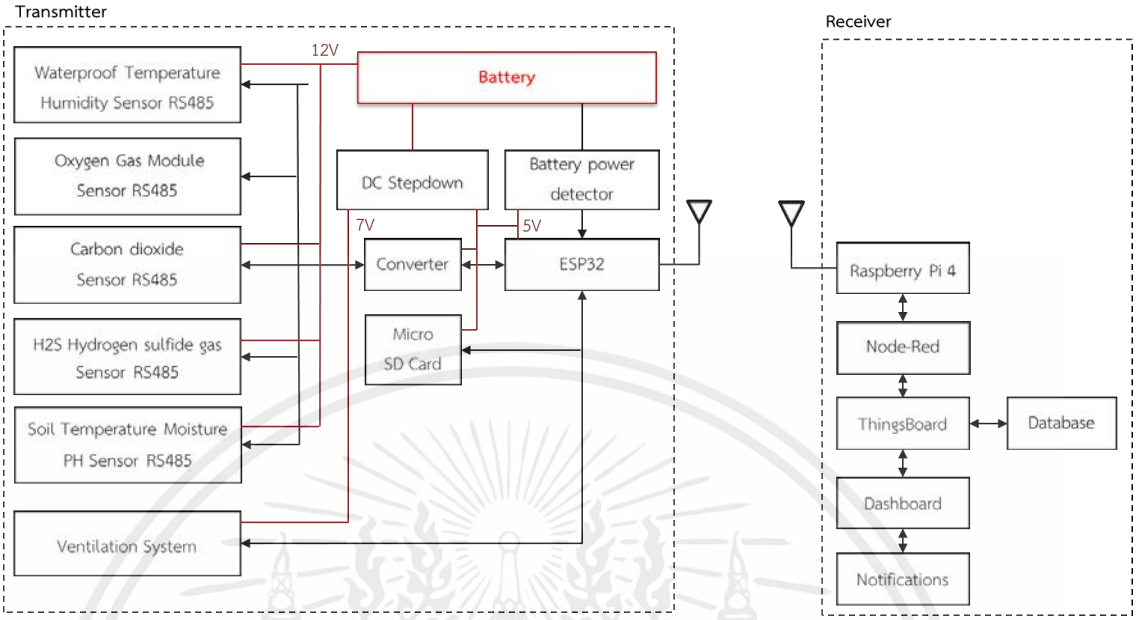
บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

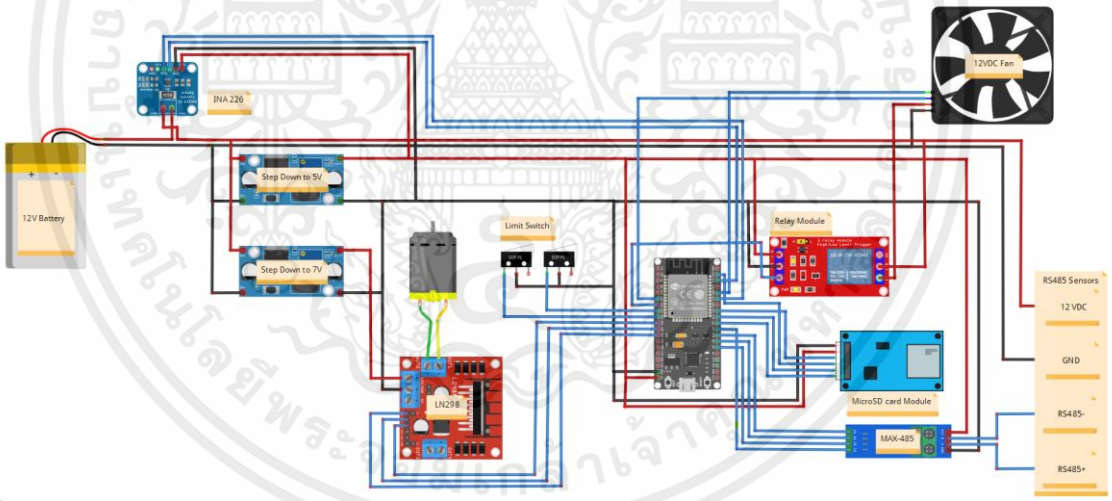
ระบบผลิตอาหารบนสถานีวิจัยอวกาศที่สามารถผลิตอาหารได้อย่างต่อเนื่อง ยั่งยืน และใช้พื้นที่น้อยที่สุด ผู้จัดทำจึงเล็งเห็นปัญหาเกิดเป็นโครงงานในครั้งนี้โดยได้จัดทำกล่องจำลองการเลี้ยงแมลงโดยคำนึงถึง 2 ด้าน คือ 1. ด้านการเพาะเลี้ยงแมลง 2. ด้านการกำจัดของเสียทางอากาศ ระบบนี้ถูกจัดทำขึ้นเพื่อแก้ไขปัญหาลดการเติมเสบียงบนสถานีวิจัยอวกาศ ซึ่งนอกจากนี้ระบบที่พัฒนาขึ้นจะไม่เป็นประโยชน์ในการใช้ชีวิตบนอวกาศเท่านั้น แต่ยังสามารถนำมาประยุกต์การใช้งานอื่นบนโลกได้อีกด้วย

3.1 การออกแบบโครงสร้างระบบโดยรวม

โครงงานนี้ได้จัดทำกล่องจำลองการเลี้ยงแมลงโดยจะออกแบบและพัฒนาระบบตรวจจับสภาพแวดล้อมภายในกล่องจำลองการเลี้ยงแมลงที่ส่งข้อมูลแบบไร้สายไปยังแพลตฟอร์มบน Raspberry Pi4 เพื่อจัดทำหน้าจอแสดงผลรายงานข้อมูลเกี่ยวกับอุณหภูมิ ความชื้น ค่าความเป็นกรดเบส แก๊สไฮโดรเจนซัลไฟด์ แก๊สคาร์บอนไดออกไซด์ แก๊สออกซิเจน โดยข้อมูลที่ได้จะถูกเก็บในระบบฐานข้อมูล และมีระบบการกำจัดของเสียทางอากาศ เมื่อมีปริมาณของแก๊สที่มากกว่าค่าที่กำหนดไว้ระบบพัดลมระบายอากาศจะทำงานแบบอัตโนมัติตัดสินใจผ่านชุดข้อมูลที่ได้รับ โดยระบบการทำงานรวมช่วยปรับปรุงความแม่นยำจากข้อมูลที่ได้รับและมีแจ้งเตือนทางผ่านแอปพลิเค-ชันไลน์ เมื่อค่าข้อมูลสภาพแวดล้อมในกล่องจำลองการเลี้ยงแมลงผิดปกติ

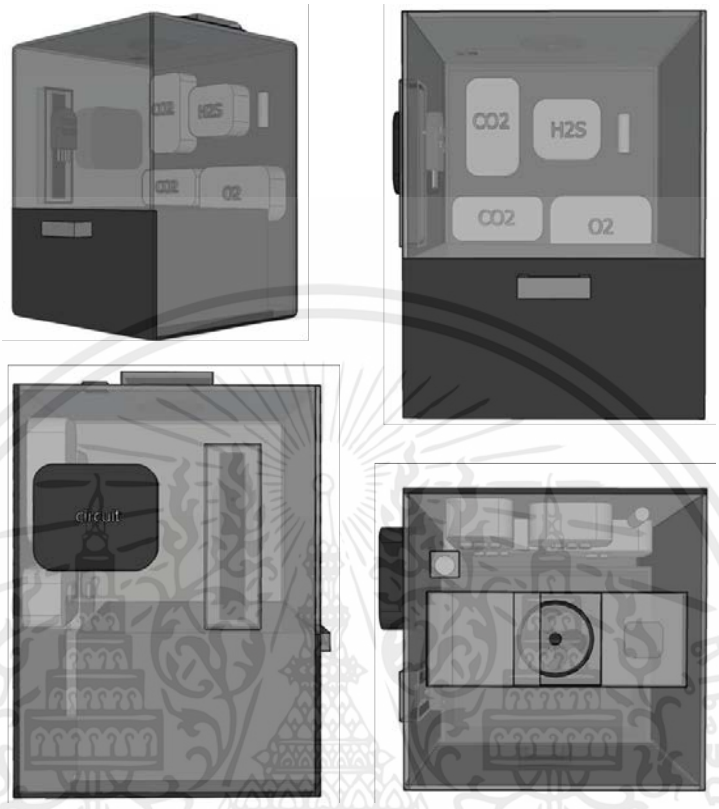


รูปที่ 3.1 บล็อกไดอะแกรมภาพรวมของโครงงาน



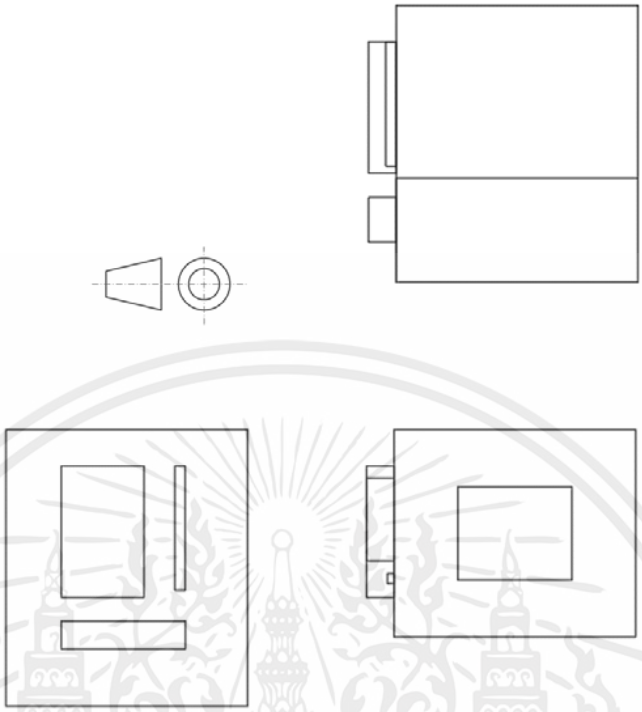
รูปที่ 3.2 ภาพรวมของการเชื่อมต่ออุปกรณ์ทั้งหมด

3.1.1 การออกแบบกล่องจำลองการเลี้ยงแมลง



รูปที่ 3.3 การออกแบบกล่องจำลองการเลี้ยงแมลงผ่านโปรแกรม SketchUp pro

จากรูปที่ 3.3 เป็นการออกแบบกล่องจำลองที่คล้ายกับกล่องทรงสี่เหลี่ยมผืนผ้ากว้าง 30 เซนติเมตร ยาว 35 เซนติเมตร สูง 40 เซนติเมตร ที่มีลิ้นชักเก็บภายในตัวกล่อง กล่องจำลองทางด้านหลังจะมีเซนเซอร์ทั้งหมด 4 ตัว โดยด้านหลังจะออกแบบให้เป็นประตู และ 1 ตัวที่ใช้ในบริเวณลิ้นชัก โดยจะมีตัวเก็บเซนเซอร์บริเวณด้านข้างของกล่อง ส่วนระบบกำจัดของเสียจะอยู่ด้านบนของกล่อง ประกอบไปด้วยพัดลม และประตูเปิด-ปิดที่ควบคุมผ่านมอเตอร์ ซึ่งวงจรรวมทั้งหมด และแบตเตอรี่จะถูกเก็บไว้ทางด้านข้างของกล่องจำลอง

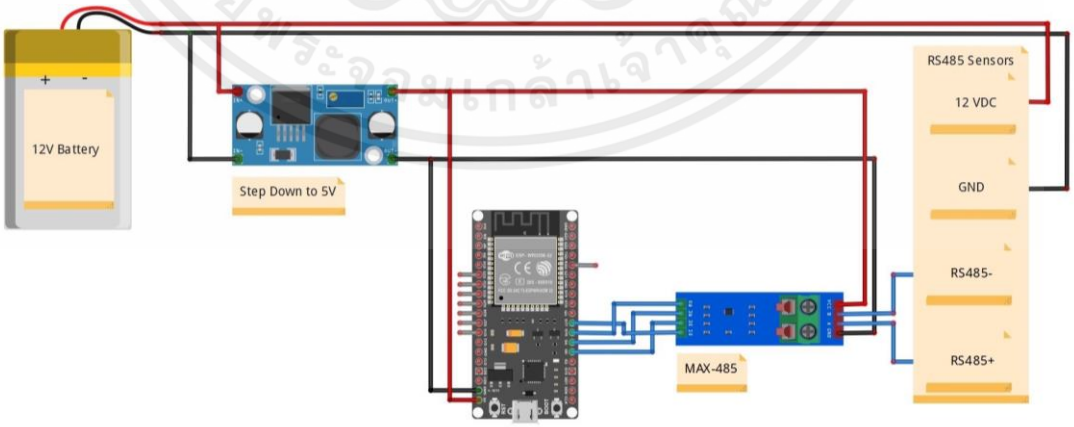


รูปที่ 3.4 ภาพฉายมุมที่ 1 ตามระบบ ISO-Method

3.1.2 การออกแบบการเชื่อมต่ออุปกรณ์กับ Microcontroller

3.1.2.1 การออกแบบการเชื่อมต่อเซนเซอร์ กับ โมดูล MAX 485

การออกแบบในส่วนการต่อเซนเซอร์ที่ทำงานบนมาตรฐาน RS485 ต่อเข้ากับโมดูล Max485 ที่ทำการแปลงสัญญาณจาก RS485 เป็นสัญญาณ TTL เพื่อจะทำให้เกิดการสื่อสารระหว่างเซนเซอร์กับ Arduino ESP 32 ซึ่งจะต่อวงจรดังรูปที่ 3.5



รูปที่ 3.5 การเชื่อมต่อระหว่างเซนเซอร์ กับ MAX 485

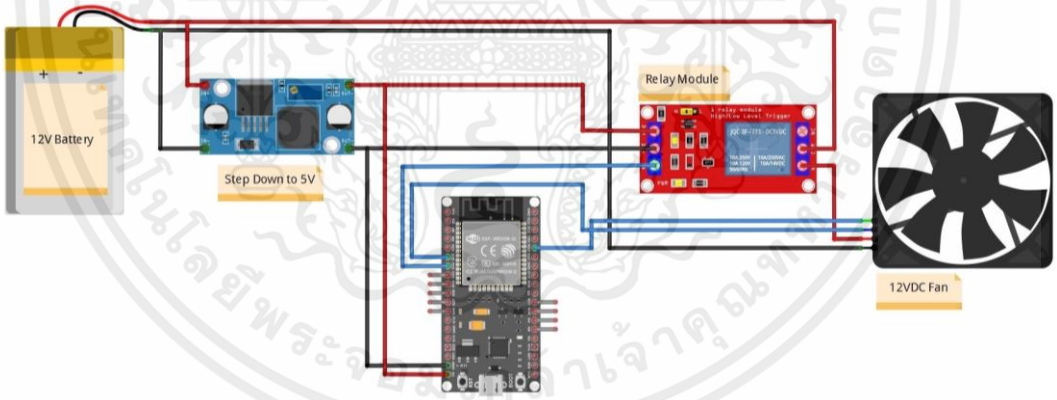
ตารางที่ 3.1 การเชื่อมต่อของ PIN ในอุปกรณ์ ESP-WROOM-32 กับ MAX 485

เซนเซอร์ RS485	MAX485
GND (สายสีดำ)	GND
A (สายสีเหลือง)	A
B (สายสีน้ำเงิน)	B
VCC (สายสีน้ำตาล)	VCC

3.1.2.2 การออกแบบระบบกำจัดของเสียผ่าน ESP-WROOM-32

1) การออกแบบระบบพัดลมระบายอากาศ

การออกแบบในส่วนที่เชื่อมต่อระหว่าง ESP-WROOM-32 กับพัดลม PWM หรือ Pulse Width Modulation Fan เป็นอุปกรณ์ที่ใช้เทคโนโลยี PWM เพื่อควบคุมความเร็วในการหมุนของใบพัดลม ซึ่งทำงานด้วยสัญญาณพัลส์ที่เรียกว่า “Duty Cycle” ด้วยไฟเลี้ยง 12 โวลต์ ซึ่งจะต่อวงจรดังรูปที่ 3.6



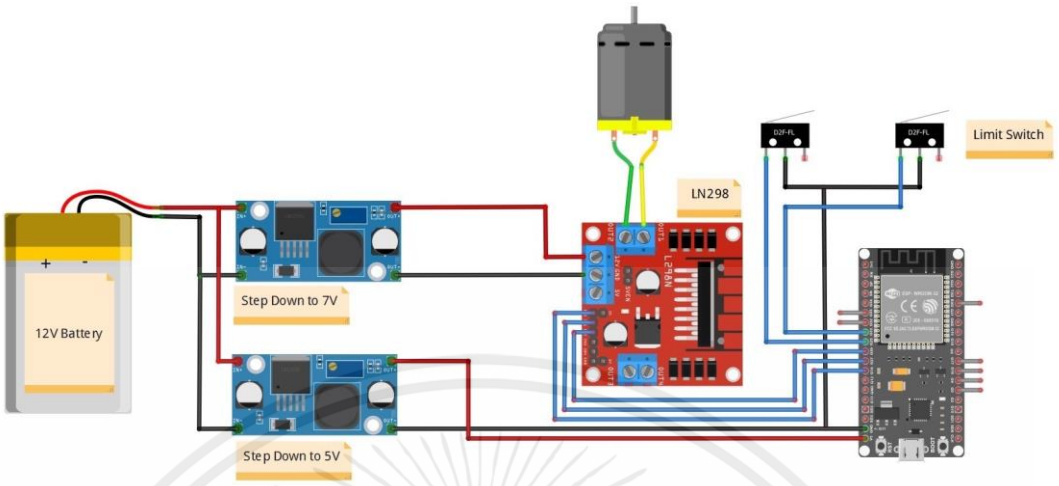
รูปที่ 3.6 การเชื่อมต่อระหว่าง ESP-WROOM-32 กับ พัดลมระบายอากาศ

ตารางที่ 3.2 การเชื่อมต่อของ PIN ในอุปกรณ์ ESP-WROOM-32 กับ พัดลมระบายอากาศ

PWM Fan	ESP-WROOM-32	DC Relay module	LM2596 DC Voltage Regulator
GND (สายสีดำ)	GND	GND	GND
	VCC	VCC	VCC
RPM (สายสีเหลือง)	GPIO 35		
PWM (สายสีน้ำเงิน)	GPIO 3		
VCC (สายสีแดง)		NO	
	GPIO 32	Signal trigger port	
		Common	VCC

2) การออกแบบระบบการเปิด-ปิดประตูระบายอากาศ

การออกแบบการทำงานในส่วนของประตูเปิด-ปิด ของระบบระบายอากาศที่เชื่อมต่อระหว่าง ESP-WROOM-32 เพื่อควบคุมมอเตอร์ใช้ในการเปิด-ปิดประตูระบายอากาศผ่านมอเตอร์ไดร์ฟเวอร์ โดยเมื่อปริมาณแก๊สไฮโดรเจนซัลไฟด์มีค่ามากกว่าหรือเท่ากับ 10 ถึง 50 ppm ESP-WROOM-32 จะส่งค่าไปที่มอเตอร์ไดร์ฟเวอร์เปิดการทำงานให้ขา IN2 เพื่อให้มอเตอร์หมุนเปิดประตู และเมื่อปริมาณแก๊สไฮโดรเจนซัลไฟด์ลดลงน้อยกว่า 10 ppm ESP-WROOM-32 จะรับค่าจากลิมิตสวิตช์เพื่อให้ ESP-WROOM-32 ปิดประตูการทำงานของมอเตอร์ผ่านขา IN1 โดยมีลิมิตสวิตช์ ที่ทำหน้าที่ในการหยุดการเลื่อนของประตูระบายอากาศซึ่งจะต่อวงจรดังรูปที่ 3.7



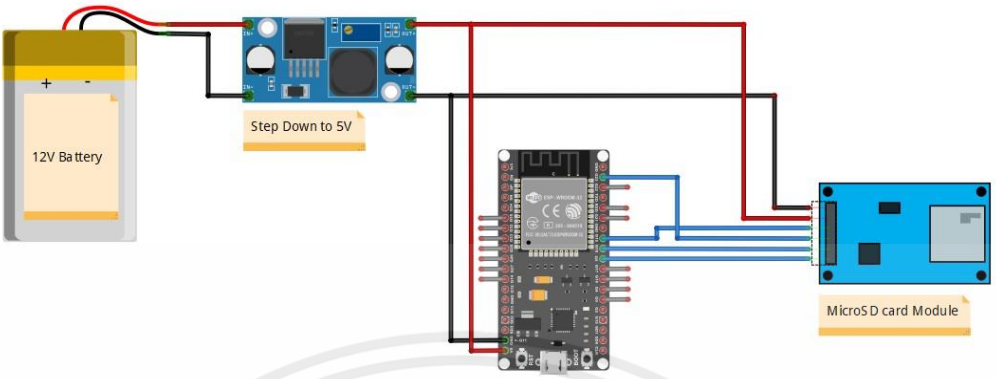
รูปที่ 3.7 การเชื่อมต่อระหว่าง ESP-WROOM-32 กับ มอเตอร์

ตารางที่ 3.3 การเชื่อมต่อของ PIN ในอุปกรณ์ ESP-WROOM-32 กับ มอเตอร์

Micro DC Motor	ESP-WROOM-32	L298N Motor Driver	Limit switch 1	Limit switch 2
positive		OUT1		
negative		OUT2		
	GPIO 14	ENA		
	GPIO 27	IN1		
	GPIO 26	IN2		
	GPIO 25		Common	
	GPIO 33			Common

3.1.2.3 การออกแบบระบบฐานข้อมูลในรูปแบบออฟไลน์

กล่องจำลองการเลี้ยงแมลงทำงานผ่านระบบแหล่งจ่ายแบบแบตเตอรี่ลิเทียม 12 โวลต์ เพื่อป้องกันการขาดหายของข้อมูลจึงได้ออกแบบวงจรใส่ Micro SD card เพื่อใช้เป็นระบบฐานข้อมูลแบบออฟไลน์ โดยออกแบบบนโปรแกรม Easy EDA ดังรูปที่ 3.8



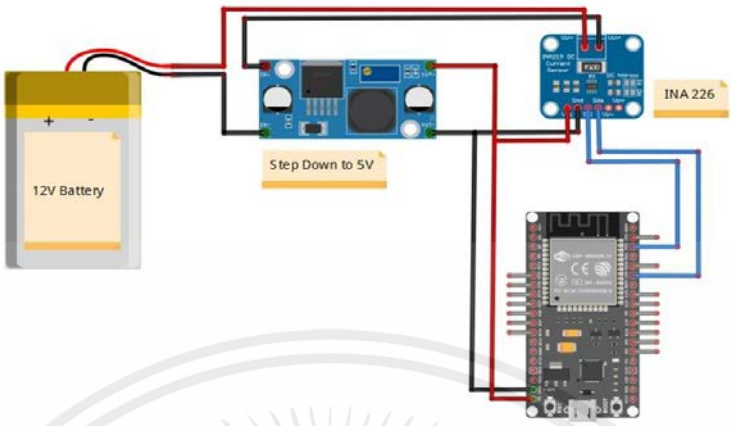
รูปที่ 3.8 การเชื่อมต่อระหว่าง ESP-WROOM-32 กับ Micro SD card

ตารางที่ 3.4 การเชื่อมต่อของ PIN ในอุปกรณ์ ESP-WROOM-32 กับ Micro SD card

ESP-WROOM-32	Micro SD card	LM2596 DC Voltage Regulator
GND	GND	GND
VCC	VCC	VCC
GPIO 19(MOSI)	MISO	
GPIO 23(MISO)	MOSI	
GPIO 18(SCK)	SCK	
GPIO 5	CS	

3.1.2.4 การออกแบบระบบตรวจสอบแรงดัน

การวัดแรงดันของอุปกรณ์ภายในระบบทำงานบนโมดูล INA 226 ที่วัดแรงดันและกระแสไฟฟ้าของเซนเซอร์อยู่ระหว่าง 0-36 โวลต์ ทำการออกแบบวงจรบนโปรแกรม Esay EDA ดังรูปที่ 3.9



รูปที่ 3.9 การเชื่อมต่อระหว่าง ESP-WROOM-32 กับ โมดูล INA 226

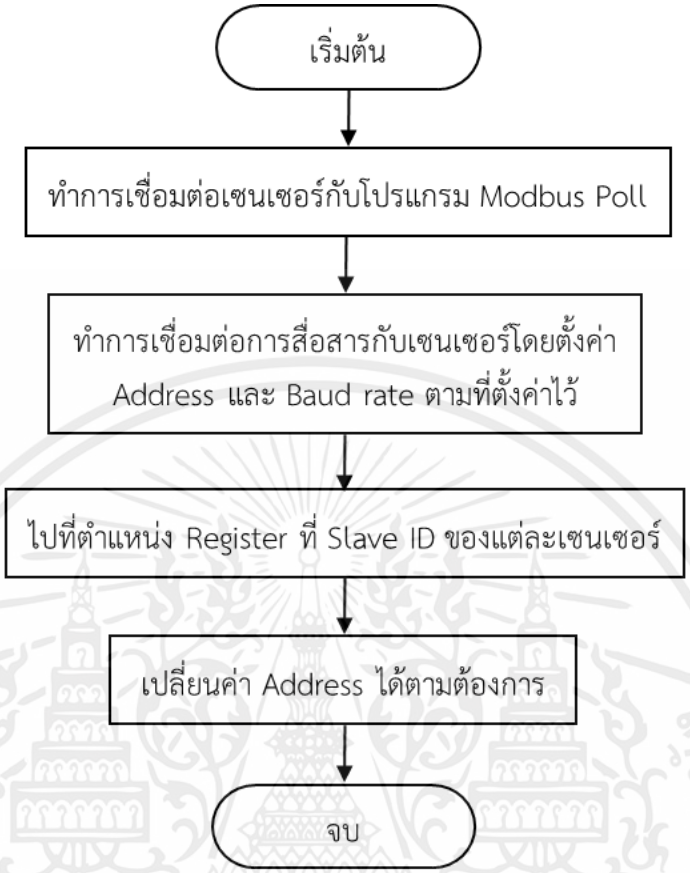
ตารางที่ 3.5 การเชื่อมต่อของ PIN ในอุปกรณ์ ESP-WROOM-32 กับ โมดูล INA 226

ESP-WROOM-32	โมดูล INA 226	LM2596 DC Voltage Regulator
GND	GND	GND
VCC	VCC	VCC
GPIO 22	SCL	
GPIO 21	SDA	

3.1.3 การส่งชุดข้อมูลจากกล่องจำลองการเลี้ยงแมลง

3.1.3.1 การตั้งค่า Address ของเซนเซอร์

ในส่วนนี้เป็นขั้นตอนการเปลี่ยน Address (Slave ID) โดยใช้โปรแกรม Modbus Poll ไปที่ Toolbar > Function > Read/Write Definition สามารถเปลี่ยน Address (Slave ID) ได้ตั้งแต่ 1 – 247 โดยจะทำการเปลี่ยนที่ Register 100H แสดงขั้นตอนการทำงานดังรูปที่ 3.10



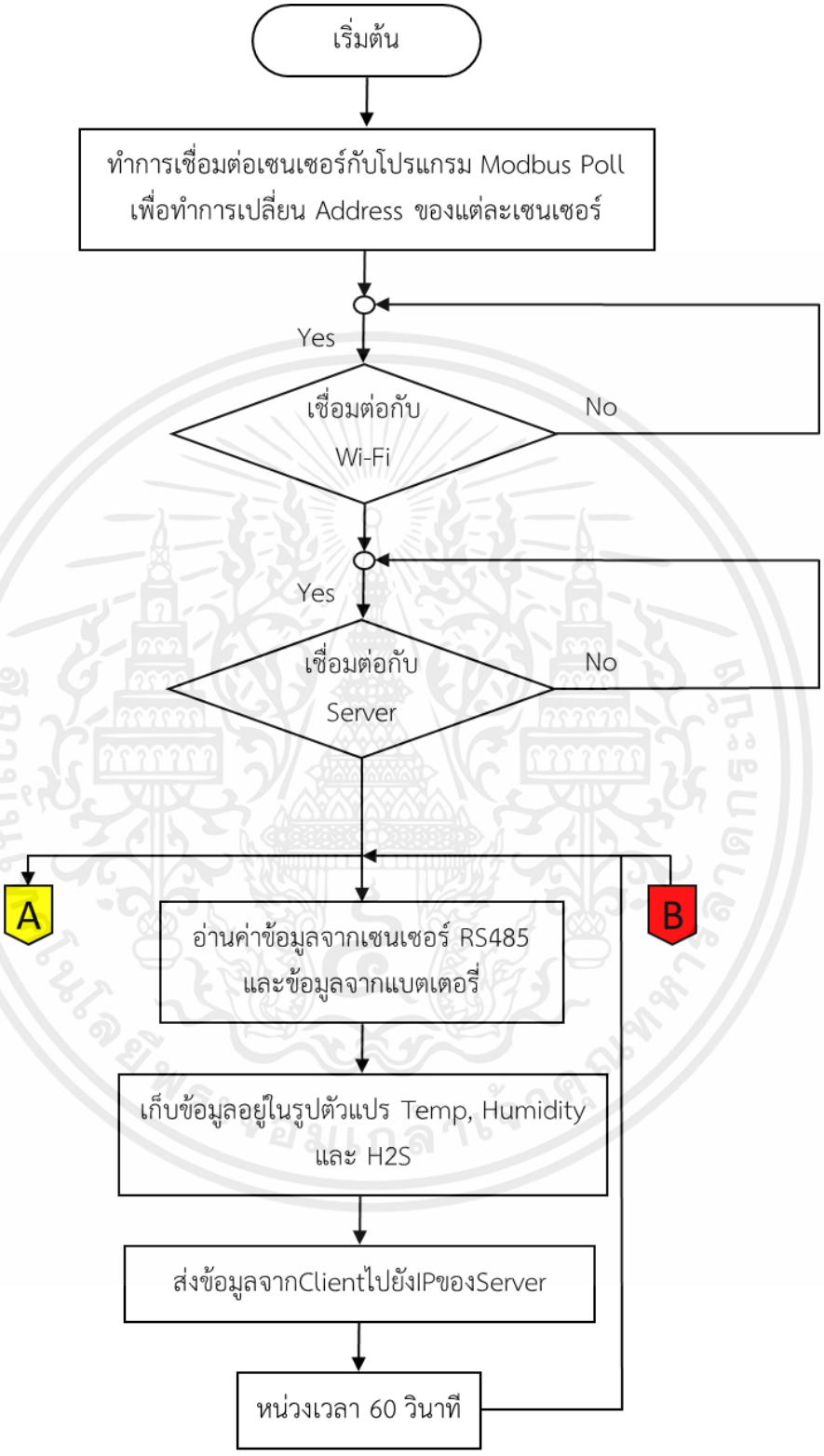
รูปที่ 3.10 ขั้นตอนการตั้งค่า Address ของเซนเซอร์

3.1.3.2 การส่งชุดข้อมูลจากเซนเซอร์ ไป ESP-WROOM-32

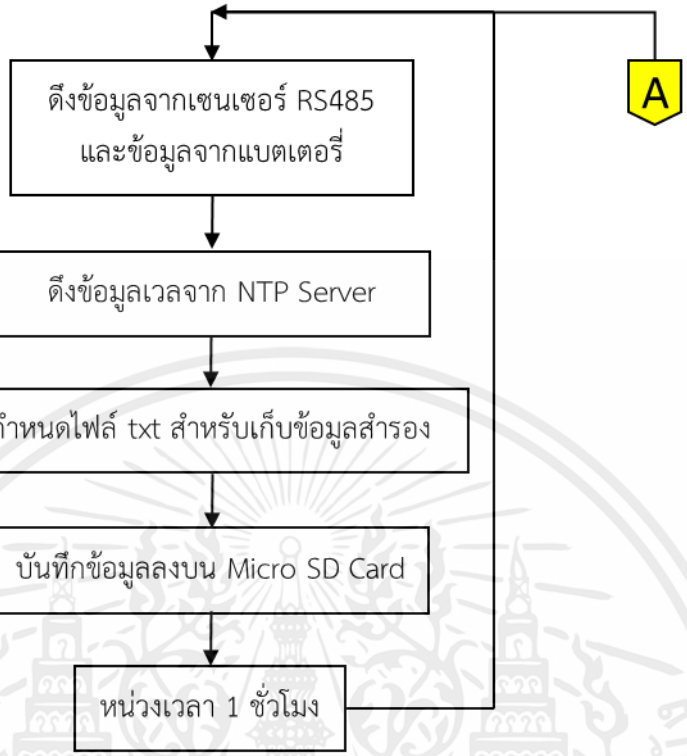
การส่งข้อมูลจากเซนเซอร์บนโปรโตคอล RS485 จะถูกต่อแบบอนุกรมเข้าตัวแปลงสัญญาณ (MAX485) เข้าขาดิจิตอลที่ ESP-WROOM-32 โดยรับค่าจาก Multiple Sensor โดยจะทำการส่ง Command เฉพาะแต่ละเซนเซอร์ไปยัง Multiple Sensor และรับค่าจากเซนเซอร์ที่มี Address ตามที่ทำการตั้งค่าไว้ก่อนหน้า โดยการส่งแต่ละ Command ไปยัง Multiple Sensor จะมีกำหนดค่าหน่วงเวลาไม่เท่ากันในโปรแกรมArduino IDE เพื่อป้องกันการชนกันของข้อมูล

ส่วนที่เป็นระบบการตรวจวัดสภาพแวดล้อมภายในกล่องเลี้ยงหนอนด้วงสาคูซึ่งใช้บอร์ด ESP-WROOM-32 ในการควบคุมการทำงานของเซนเซอร์ และระบบระบายอากาศโดยจะเชื่อมต่อกับเซิร์ฟเวอร์ผ่าน WI-FI บนโปรโตคอล MQTT ซึ่งจะดึงค่าข้อมูลจากเซนเซอร์ทุกๆ 1นาที เพื่อส่งไปยังเซิร์ฟเวอร์ แต่เนื่องจากเซนเซอร์เกรดอุตสาหกรรมที่มี IP 67 ขึ้นไป ใช้โปรโตคอล RS485 ทำให้ในการร้องขอในการดึงข้อมูลจากเซนเซอร์ในแต่ละตัวด้วยเส้นทางเดียวกัน จะต้องทำ

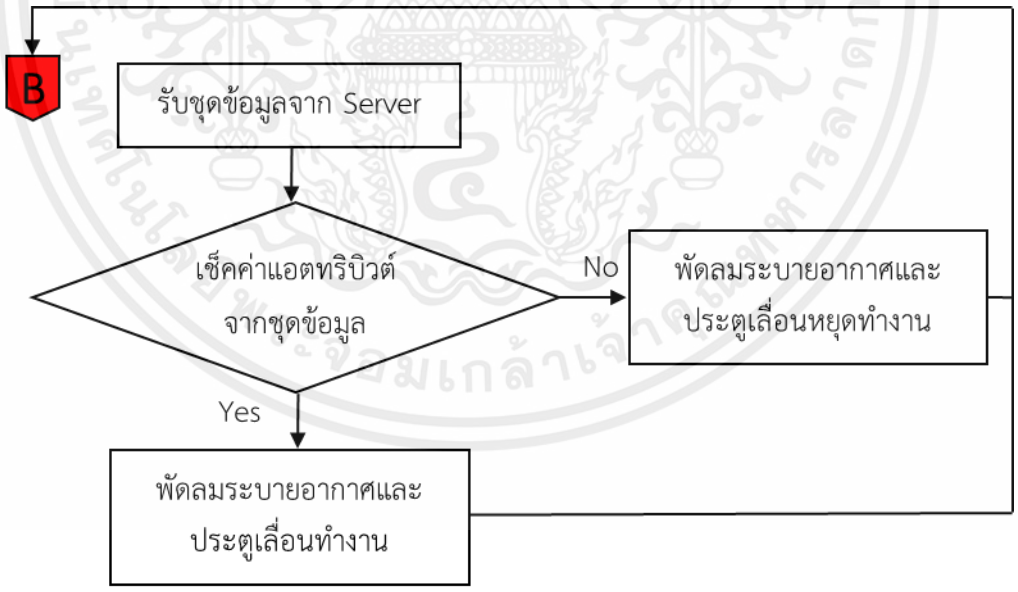
การเลื่อนเวลาให้เหลื่อมกันเล็กน้อย เพื่อป้องกันการชนกันของข้อมูลจากนั้น รอรับการประมวลผลกลับจากฝั่งเซิร์ฟเวอร์ในกรณีที่ต้องจัดการของเสีย เพื่อควบคุมการทำงานของระบบระบายอากาศ นอกจากนั้น ข้อมูลที่ได้จากเซนเซอร์ทุกตัวในทุกๆ 1 นาที จะบันทึกลงบน SD Card ด้วยเพื่อสำรองข้อมูล ในกรณีที่ไม่สามารถเชื่อมต่อกับเซิร์ฟเวอร์ผ่าน WI-FI ได้ดังรูปที่ 3.11



รูปที่ 3.11 การทำงานการส่งชุดข้อมูลจากฝั่ง Client ไปยัง Server



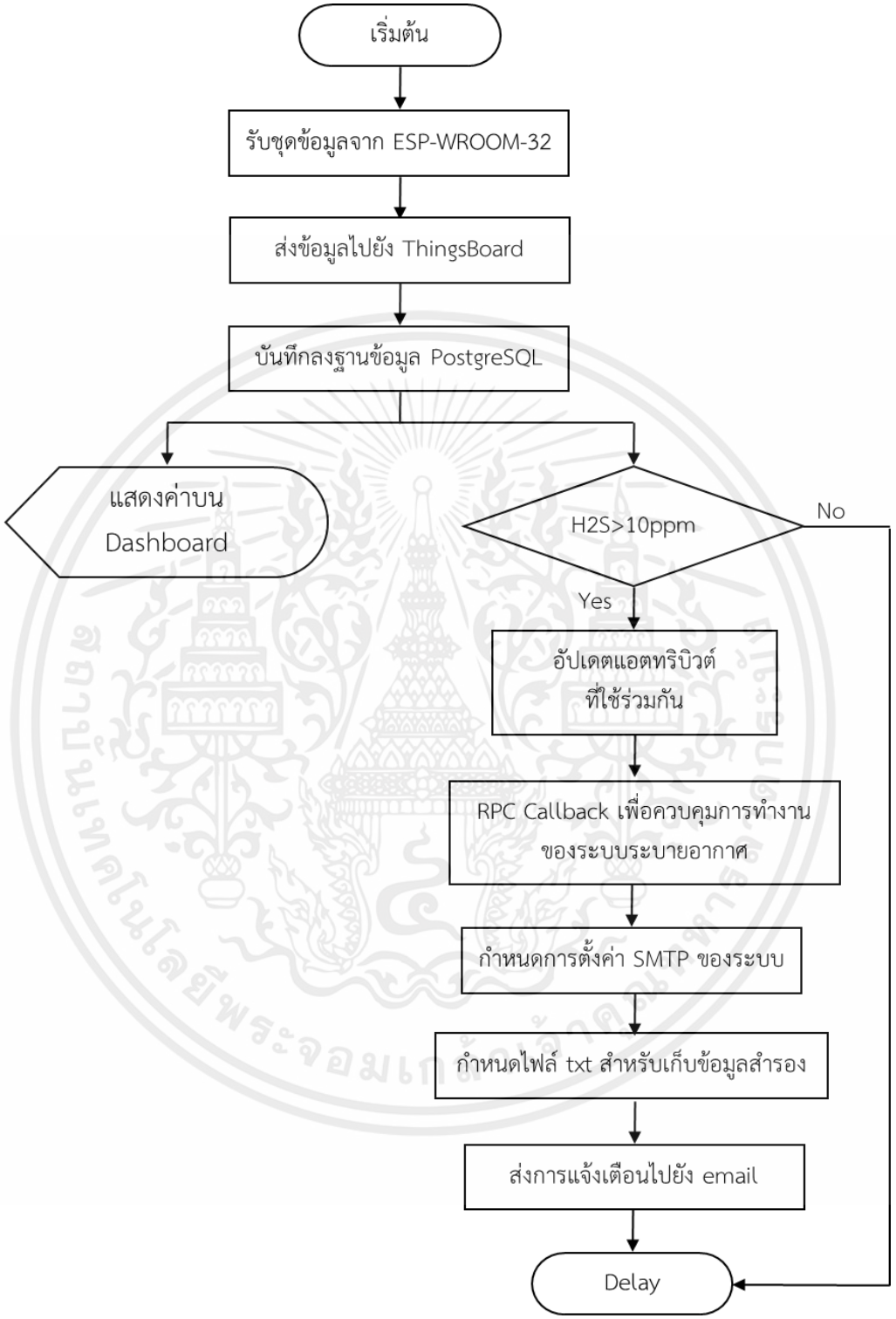
รูปที่ 3.12 การทำงานการส่งชุดข้อมูลจากฝั่ง Client ไปยัง Server



รูปที่ 3.13 การทำงานการส่งชุดข้อมูลจากฝั่ง Client ไปยัง Server

3.1.3.3 การรับชุดข้อมูลโดย Raspberry Pi4 จาก ESP-WROOM-32

ส่วนที่เป็นการทำงานฝั่งเซิร์ฟเวอร์บนบอร์ด Raspberry Pi4 โดยแพลตฟอร์ม ThingsBoard จะรอรับข้อมูลการตรวจวัดของเซนเซอร์ เพื่อแสดงค่าบนหน้าจอแสดงผล โดยระบบจะมีการประมวลผลข้อมูลและแจ้งเตือนในกรณีที่ข้อมูลสภาพแวดล้อมมีความผิดปกติซึ่งอาจส่งผลต่อคุณภาพในการเจริญเติบโตของแมลงกินได้ นอกเหนือจากนั้น ในระบบต้นแบบนี้ ได้พัฒนาระบบกำจัดของเสียด้วยการระบายอากาศ เมื่อสภาพอากาศภายในกล่องเลี้ยงมีค่าเกินกว่าที่กำหนด ได้ดังรูปที่ 3.14

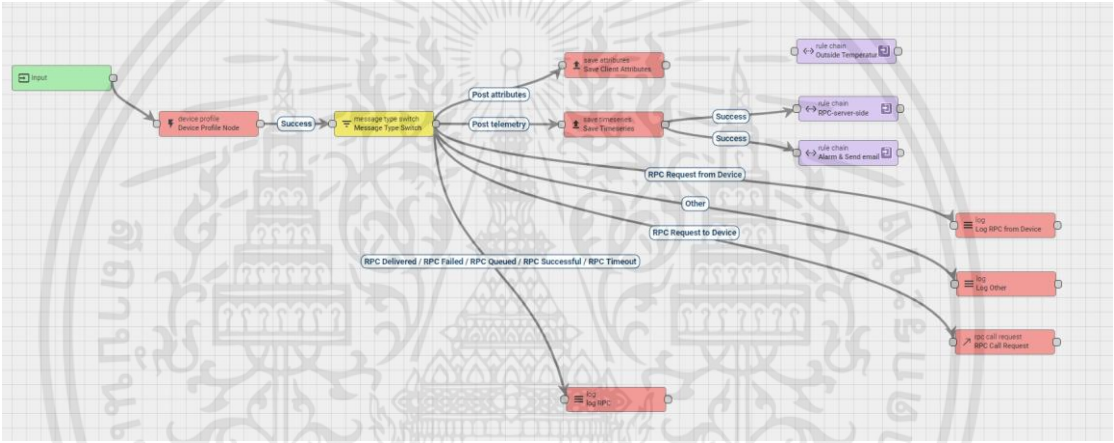


รูปที่ 3.14 การทำงานการรับชุดข้อมูลโดย Server จากฝั่ง Client

3.1.4 การสร้างและออกแบบระบบรายงานข้อมูล

3.1.4.1 การกำหนดพิกัด และการแสดงรายงานสภาพอากาศอัตโนมัติผ่าน REST API

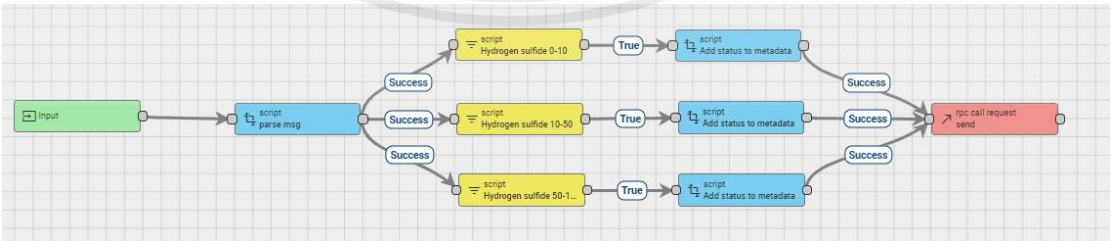
บนเว็บที่จัดทำรายงานจอแสดงผลข้อมูลจากกล่องจำลองการเลี้ยงแมลง หน้าแรกจัดทำรายงานพิกัดของกล่องจำลองโดยในกล่องต้นแบบยังไม่สามารถติดตามพิกัดได้เลยทำการระบุพิกัดบนแพลตฟอร์ม ThingsBoard และรายงานสภาพอากาศโดยรอบจากการใช้ REST API เข้ามาช่วยในการนำค่าข้อมูลจากเว็บรายงานสภาพอากาศมาจัดแสดงบนเว็บ โดยเขียนแผนการทำงานดังรูปที่ 3.15



รูปที่ 3.15 การทำงานของการระบุพิกัด และการรายงานสภาพอากาศจาก

3.1.4.2 การออกแบบระบบกำจัดของเสีย

การออกแบบระบบกำจัดของเสียจะทำการเขียนแผนการทำงานดังรูปที่ 3.16 โดยเมื่อรับค่าข้อมูลปริมาณแก๊สไฮโดรเจนซัลไฟด์จาก ESP-WROOM-32 ในปริมาณที่กำหนดไว้ระบบจะส่งค่ากลับไปที่ ESP32 เพื่อสั่งระบบระบายอากาศทำงาน เป็นไปตามตารางที่ 3.4



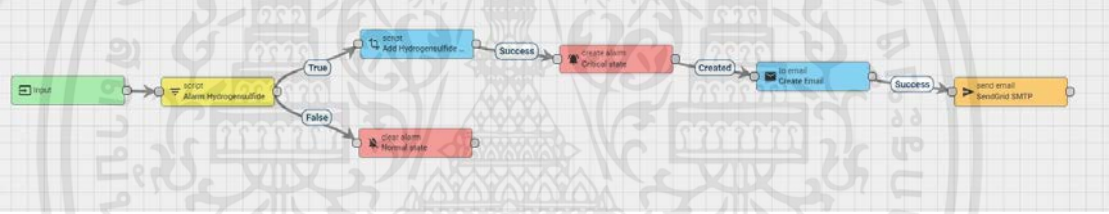
รูปที่ 3.16 แผนการทำงานของระบบกำจัดของเสีย

ตารางที่ 3.4 การกำหนดช่วงการทำงานของระบบกำจัดของเสีย

กรณีที่	ปริมาณแก๊สไฮโดรเจนซัลไฟด์ (ppm)	พัดลมระบายอากาศ	ระบบเปิด-ปิดประตู
1	0-10	ไม่ทำงาน	ปิด
2	10-50	ทำงานด้วยรอบความเร็วสูง	เปิด
3	50-10	ทำงานด้วยรอบความเร็วต่ำ	เปิด

3.1.4.3 การสร้างระบบแจ้งเตือน

เมื่อได้รับค่าข้อมูลจากกล่องจำลองการเลี้ยงแมลงตามหัวข้อที่ 3.1.4.1 ที่กำหนดช่วงของข้อมูลแต่ละชนิด เมื่อค่าข้อมูลที่เกินกว่าช่วงที่กำหนดไว้จะทำการแจ้งเตือนผ่านแอปพลิเคชันไลน์ โดยผ่านโปรโตคอล SMTP



รูปที่ 3.17 การออกแบบการสร้างระบบแจ้งเตือนผ่านอีเมล

3.2 เครื่องมือที่ใช้ในการทดลอง

ในโครงงานนี้มีอุปกรณ์ และเครื่องมือที่ใช้ในการทดลอง ดังนี้

3.2.1 ESP-WROOM-32

ใช้งานในการประมวลผลค่าข้อมูลจากเซนเซอร์ด้วยการรับส่งข้อมูลผ่าน Wi-Fi หรือบลูทูธ ตามรูปที่ 2.1

3.2.2 MAX 485 Module

เป็นโมดูลแปลงสัญญาณระดับ TTL เป็นสัญญาณ RS485485 ทำงานที่แหล่งจ่ายไฟ +5 โวลต์ และกระแสไฟที่กำหนดคือ 300 แอมแปร์ ตามรูปที่ 2.2

3.2.3 Waterproof Temperature Humidity Sensor

เป็นเซนเซอร์ที่สามารถวัดอุณหภูมิ และความชื้นในอากาศ ตามรูปที่ 2.3

3.2.4 H2S Hydrogen sulfide gas Sensor

เป็นเซนเซอร์ที่สามารถวัดแก๊สไฮโดรเจนซัลไฟด์ หรือแก๊สไข่เน่า ตามรูปที่ 2.4

3.2.5 Carbon dioxide Sensor

เป็นเซนเซอร์ที่สามารถวัดแก๊สคาร์บอนได้ออกไซด์ ตามรูปที่ 2.5

3.2.6 Oxygen Gas Module Sensor

เป็นเซนเซอร์ที่สามารถวัดแก๊สออกซิเจนตามรูปที่ 2.6

3.2.7 Soil Temperature Moisture PH Sensor

เป็นเซนเซอร์ที่สามารถวัดอุณหภูมิ ความชื้น ความเป็นกรดเบสในดิน และค่าความนำไฟฟ้าในดิน ตามรูปที่ 2.7

3.2.8 Raspberry Pi 4 Model B

เป็นเหมือนคอมพิวเตอร์เล็กและ มี RAM จำนวน 8GB เพื่อให้มีความสามารถในการประมวลผลและรันแอปพลิเคชันได้ โดยใช้ Broadcom BCM2711 Quad-Core ARM Cortex-A72 ตามรูปที่ 2.8

3.2.9 Micro SD card

ทำหน้าที่แปลงระดับแรงดันไฟฟ้าตรงลงจาก 4-35 โวลต์ ให้อยู่ในช่วง 1.25-30 โวลต์ โดยสามารถปรับค่าแรงดัน output ได้โดย Potentiometer ที่มีอยู่บนบอร์ด ตามรูปที่ 2.9

3.2.10 L298N Motor Driver Module

คืออุปกรณ์หรือวงจรอิเล็กทรอนิกส์ที่ใช้ในการควบคุมการเคลื่อนที่ของมอเตอร์ ตามรูปที่ 2.10

3.2.11 พัดลมระบายอากาศ

PWM Fan หรือ Pulse Width Modulation Fan เป็นอุปกรณ์ที่ใช้เทคโนโลยี PWM เพื่อควบคุมความเร็วในการหมุนของใบพัดลม ตามรูปที่ 2.11

3.2.12 Micro DC Motor

เป็นอุปกรณ์ระบบเครื่องกลไฟฟ้าโดยใช้แรงดันไฟฟ้า 1.5-6V ตามรูปที่ 2.12

3.2.13 Lithium-Ion Battery 12V

แบตเตอรี่ลิเธียมไอออน 12 โวลต์ ประกอบด้วยหลายเซลล์ที่เชื่อมต่อกันแบบอนุกรม โดยแต่ละเซลล์มีแรงดันไฟฟ้าปกติที่ 3.6-3.7 โวลต์ ตามรูปที่ 2.13

3.2.14 INA 226

ทำหน้าที่วัดแรงดันของอุปกรณ์ภายในระบบทำงานบนโมดูล INA 226 ที่วัดแรงดันและกระแสไฟฟ้าของเซนเซอร์อยู่ระหว่าง 0-36V ตามรูปที่ 2.14

3.2.15 DC Relay module

รีเลย์เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ทำหน้าที่เหมือนสวิตช์ไฟฟ้าในระบบควบคุมอัตโนมัติ โดยใช้แรงดันไฟฟ้าเพื่อเปิดและปิดอุปกรณ์ไฟฟ้าต่าง ๆ ตามรูปที่ 2.15

3.2.16 Limit switch

เป็นอุปกรณ์ที่ใช้ในแอปพลิเคชันต่าง ๆ เพื่อตรวจจับการเคลื่อนไหวของวัตถุหรือชิ้นงานในสถานการณ์ที่มีความจำเป็นในการควบคุม และจำกัดการเคลื่อนของอุปกรณ์ที่ต้องการ ตามรูปที่ 2.16

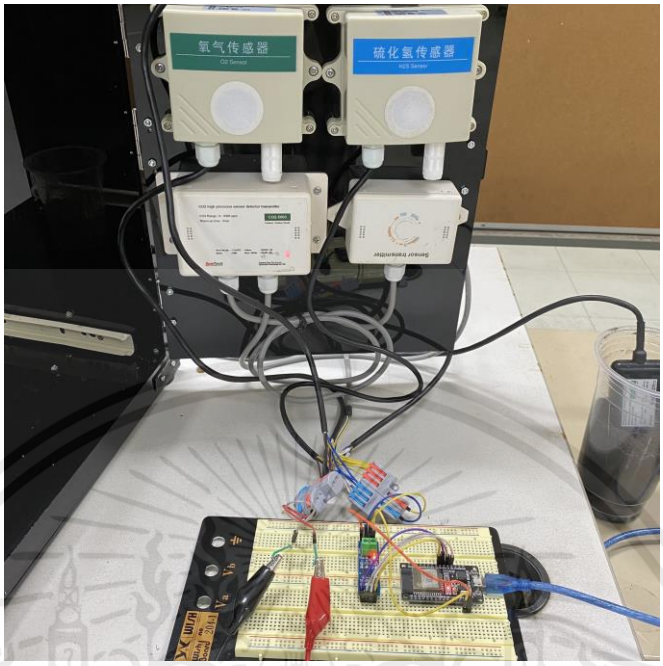
3.2.17 LM2596 DC Voltage Regulator

ทำหน้าที่แปลงระดับแรงดันไฟฟ้าตรงลงจาก 4-35 โวลต์ ให้อยู่ในช่วง 1.25-30 โวลต์ โดยสามารถปรับค่าแรงดัน output ได้โดย Potentiometer ที่มีอยู่บนบอร์ด ตามรูปที่ 2.17

3.3 การจัดเก็บผลการทดลอง

3.3.1 การทดสอบการรับข้อมูลจากเซนเซอร์

ทำการทดสอบการทำงานของเซนเซอร์ RS485 สัญญาณจากเซนเซอร์แบบ multiple sensors ที่ต่อแบบอนุกรมเข้าตัวแปลงสัญญาณ สัญญาณจากเซนเซอร์ทำการส่งค่าอินพุตเข้าตัวแปลงสัญญาณ เข้าขาดิจิตอลที่ ESP-WROOM-32 โดยภาครับจะรับสัญญาณเข้าระบบด้วยโปรแกรม Arduino IDE ตามรูปที่ 3.18



รูปที่ 3.18 การทดสอบการรับข้อมูลจากเซนเซอร์

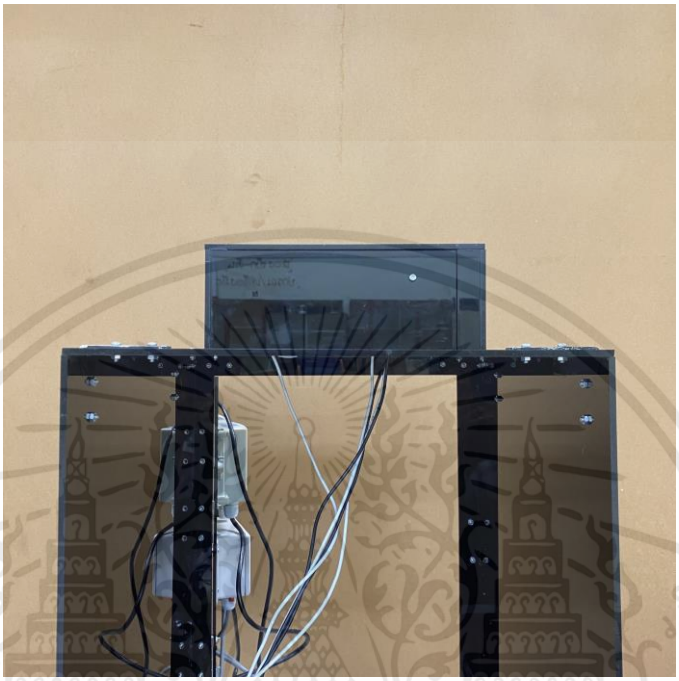
3.3.2 การทดสอบการทำงาน ESP-WROOM-32 ในการควบคุมพัดลม

ทำการทดสอบการทำงานของพัดลม PWM ด้วยการต่อพัดลม PWM กับ DC Relay module และ ESP-WROOM-32 เพื่อควบคุมการ เปิด-ปิด ในการทำงานของใบพัดลม ทำการจ่ายไฟเลี้ยง 12 โวลต์ โดยที่ระดับความเร็วของพัดลม จะแบ่งเป็น 2 ระดับ คือที่ระดับปานกลาง ซึ่งจะมี PWM = 127 คิดเป็นรูปแบบเปอร์เซ็นต์ของดิวตี้ไซเคิลได้ ดิวตี้ไซเคิลเท่ากับ 50 % และระดับความเร็วสูงสุด ซึ่งจะมี PWM = 255 คิดเป็นรูปแบบเปอร์เซ็นต์ของดิวตี้ไซเคิลได้ ดิวตี้ไซเคิลเท่ากับ 100 %

3.3.3 การทดสอบการทำงาน ESP-WROOM-32 ในการเปิด-ปิดประตูพัดลม

ทำการทดสอบการทำงานในส่วนของประตูเปิด-ปิด ประตูพัดลมในระบบระบายอากาศ จะทำการเชื่อมต่อมอเตอร์ไมโคร DC กับมอเตอร์ไดรเวอร์ L298N ฟังก์ชันนี้จะเริ่มต้นโหมดพินด้วยการตั้งค่าสถานะการเริ่มต้นของพินควบคุมมอเตอร์ กำหนดค่าลิมิตสวิตช์ และเริ่มการสื่อสารแบบอนุกรม ทำการตรวจสอบคำสั่งอนุกรมเพื่อเปิด-ปิดมอเตอร์ หากมอเตอร์เปิดอยู่และไม่มีการทริกเกอร์ LimitSwitch1Pin มันจะเปิดประตูจนกว่า LimitSwitch2Pin จะถูกทริกเกอร์ หากมอเตอร์ปิดอยู่ และไม่มีการทริกเกอร์ LimitSwitch2Pin มอเตอร์จะปิดจนกว่า LimitSwitch1Pin

จะถูกทริกเกอร์ และยังมีฟังก์ชัน Stopmotor() ที่ใช้เพื่อหยุดมอเตอร์โดยการตั้งค่าพินควบคุมทั้งสองไปเป็นค่า LOW



รูปที่ 3.19 การทำงานการปิด ของการเปิด-ปิดประตูพัดลม

รูปที่ 3.20 การทำงานการเปิด ของการเปิด-ปิดประตูพัดลม

3.3.4 การจำลองการทดสอบระบบการทำงานในระยะเวลา 7 วัน

ทำการจำลองการทดสอบระบบการทำงานเป็นระยะเวลา 7 วัน ในการเก็บค่าอุณหภูมิ ความชื้นทั้งในอากาศและในดิน ค่าความเป็นกรดเบส แก๊สไฮโดรเจนซัลไฟด์ แก๊สคาร์บอนไดออกไซด์ และแก๊สออกซิเจน โดยมีการเก็บค่าทุกๆ 5 นาที ในวันที่ 5 ของการทดลองจะมีการเติมแคลเซียมคาร์บอเนตเพื่อปรับสภาพความเป็นกรดเบส ดังรูปที่ 3.21

รูปที่ 3.21 การเติมแคลเซียมคาร์บอเนตเพื่อปรับสภาพความเป็นกรดเบส

3.4 การประกอบชิ้นงานตามต้นแบบ

3.4.1 การออกแบบตัวกล่องชิ้นงาน

ในส่วนของการออกแบบอุปกรณ์เพื่อให้ได้ชิ้นงานที่ใช้ได้จริงนั้น ผู้จัดทำได้คำนึงถึงการใช้งานง่ายต่อการทำความสะอาดหลังใช้งาน ที่สำคัญยังต้องสะดวกในการจัดเก็บข้อมูล การออกแบบครั้งนี้ผู้จัดทำได้ออกแบบในโปรแกรม SketchUp ร่างกล่องจำลองทางการเกษตรเพื่อให้เห็นภาพชัดเจน และลองจัดวางอุปกรณ์เพื่อให้ใช้งานได้จริงโดยอ้างอิงตามรูปที่ 3.3

รูปที่ 3.22 กล่องจำลองการเลี้ยงแมลง

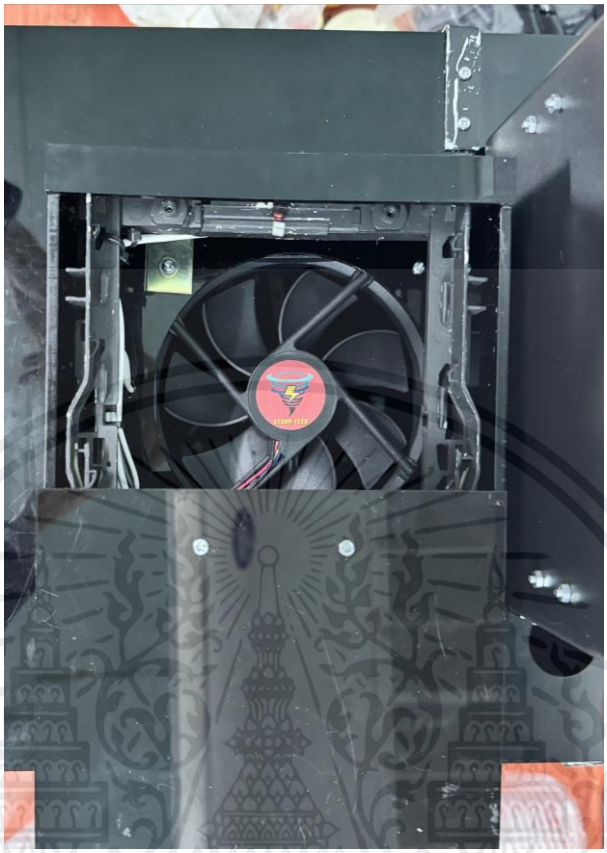
3.4.2 การออกแบบวงจรรวม

ในการออกแบบวงจรเพื่อเชื่อมต่ออุปกรณ์ภายในระบบเข้ากับระบบไฟเลี้ยง โดยบนบอร์ดประกอบด้วยไฟเลี้ยง วงจรแปลงแรงดัน อุณหภูมิ ความชื้นในดิน และในอากาศ ค่าความเป็นกรดด่าง แก๊สไฮโดรเจนซัลไฟด์ แก๊สคาร์บอนได้ออกไซด์ แก๊สออกซิเจนที่ทำงานบนโปรโตคอล RS485 โดยสัญญาณจากเซนเซอร์จะถูกต่อแบบอนุกรมทำการส่งค่าอินพุตเข้าตัวแปลงสัญญาณ (MAX485) เข้าขาดิจิตอลที่ ESP-WROOM-32 ตามรูปที่ 3.14

รูปที่ 3.23 วงจรรวม

3.4.3 ระบบกำจัดของเสีย

ในการออกแบบระบบกำจัดของเสียเพื่อทำการระบายอากาศเสียที่เกิดจากแก๊สไฮโดรเจนซัลไฟด์ หรือแก๊สไข่เน่า โดยใช้พัดลม PWM ช่วยในการถ่ายเท และระบายอากาศอากาศเสียภายในกล่องให้ค่าของแก๊สไฮโดรเจนซัลไฟด์ หรือแก๊สไข่เน่าลดลง โดยเมื่อค่าของแก๊สไฮโดรเจนซัลไฟด์ หรือแก๊สไข่เน่า มีค่าเพิ่มมากขึ้นจนถึงขีดจำกัดที่ตั้งไว้ ประตูระบายจะทำการเปิดออก และพัดลมระบายอากาศเริ่มทำงานทันที โดยที่พัดลมระบายอากาศจะมีการทำงาน 2 ระดับ คือ ระดับปานกลาง และระดับแรงสุด ตามปริมาณค่าของแก๊สไฮโดรเจนซัลไฟด์ หรือแก๊สไข่เน่าที่ได้ทำการตั้งค่าไว้ หลังจากที่ค่าของแก๊สไฮโดรเจนซัลไฟด์ หรือแก๊สไข่เน่า ลดลงจนถึงค่าที่ตั้งไว้ต่ำสุด พัดลมจะหยุดการทำงาน และประตูระบายอากาศจะทำการปิดอัตโนมัติ



รูปที่ 3.24 ระบบกำจัดของเสีย

บทที่ 4

ผลการทดลอง

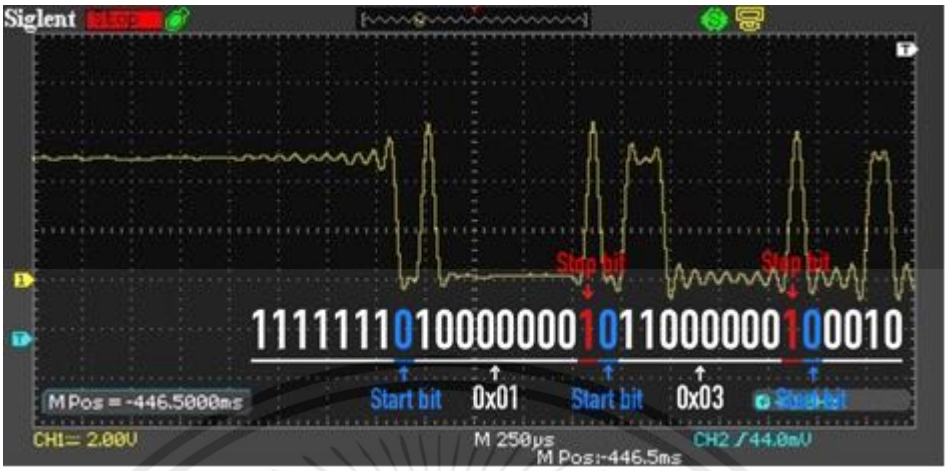
ผู้จัดทำได้ทำเก็บผลการทำงานของระบบ โดยการทดลองและจัดเก็บผลการทดลองเป็นส่วนๆ ดังต่อไปนี้

4.1 การทดสอบการทำงานระหว่างอุปกรณ์ กับ Microcontroller

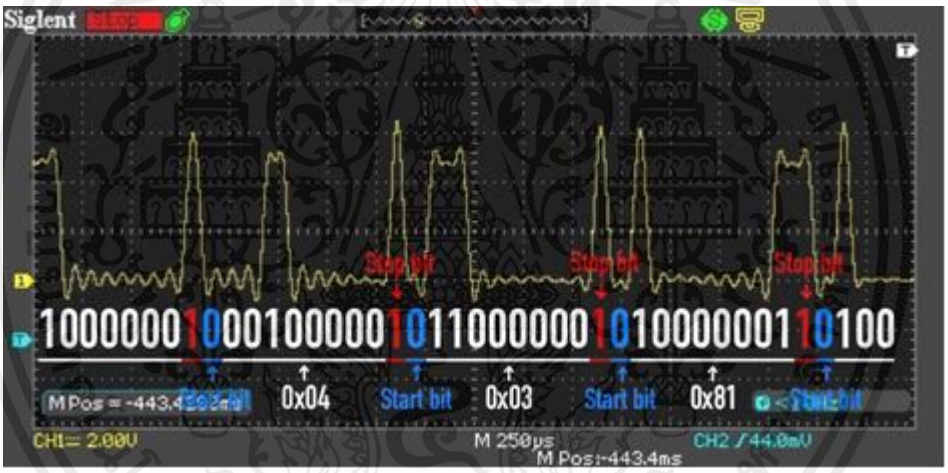
4.1.1 การทำงานระหว่างเซนเซอร์กับ โมดูล MAX485

จากการทดลองตามหัวข้อที่ 3.1.2.1 จะได้ค่าสัญญาณบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 โดยใช้งานร่วมกับโมดูล MAX485 และเซนเซอร์ที่มี Slave ID เท่ากับ 01 โดย ESP-WROOM-32 จะส่งชุดบิตคำสั่งเฉพาะไปยังเซนเซอร์โดยผ่านโมดูล MAX485 แสดงตามรูปที่ 4.1

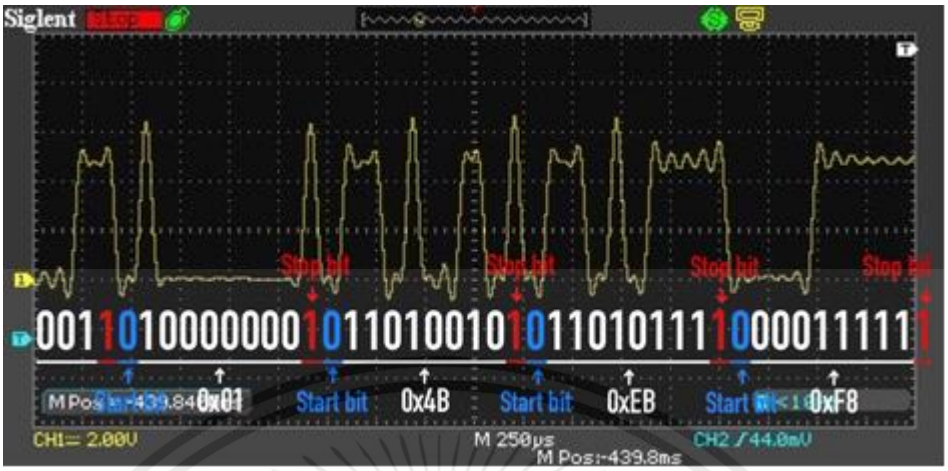
รูปที่ 4.1 ค่าสัญญาณบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์



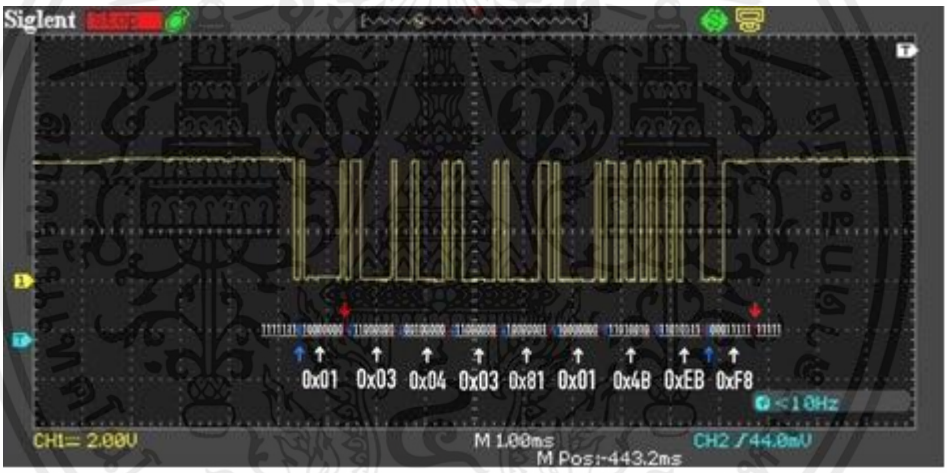
รูปที่ 4.2 ถอดรหัสบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์(ไบต์ที่1-2)



รูปที่ 4.3 ถอดรหัสบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์ (ไบต์ที่3-5)



รูปที่ 4.4 ถอดรหัสบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์ (ไบต์ที่6-9)



รูปที่ 4.5 ถอดรหัสบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 ร่วมกับโมดูล MAX485 และเซนเซอร์ทั้ง 9 ไบต์

จากรูปที่ 4.5 ค่าสัญญาณบิตข้อมูลที่ได้รับจาก ESP-WROOM-32 โดยใช้งานร่วมกับโมดูล MAX485 และเซนเซอร์ที่มี Slave ID เท่ากับ 01 โดย Arduino UNO R3 จะส่งชุดบิตคำสั่งเฉพาะไปยังเซนเซอร์โดยผ่านโมดูล MAX485 ทั้งหมดไบต์ที่ 9 ไบต์ คือ 0x01, 0x03, 0x04, 0x03, 0x81, 0x01, 0x4B, 0xEB, 0xFB โดยมี Start bit เท่ากับ 0 และ Stop bit เท่ากับ 1 จากนั้นวัดค่าสัญญาณบิตข้อมูลที่วัดได้จากขา Rx ของ ESP-WROOM-32 โดยใช้งานร่วมกับโมดูล MAX485 และเซนเซอร์วัดอุณหภูมิ ความชื้นในอากาศ จากเซนเซอร์ Waterproof Temperature Humidity Sensor RS485

โดยชุดข้อมูลที่ตอบวัดได้จากขา Rx จากเซนเซอร์ Waterproof Temperature Humidity Sensor RS485 ที่มีค่า Address เท่ากับ 1 จะนำค่าฐาน 16 ที่ได้ไปแปลงเป็นค่าอุณหภูมิ และความชื้นในอากาศดังนี้

1) No.1 Data (Humidity) = 0x03 0x81: ข้อมูลความชื้นมีค่าเท่ากับ 0x0381 ซึ่งหมายความว่าความชื้นเป็น 89.7%

2) No.2 Data (Temperature) = 0x01 0x4B: ข้อมูลอุณหภูมิมีค่าเท่ากับ 0x014B ซึ่งหมายความว่าอุณหภูมิเป็น 33.1 องศาเซลเซียส

4.1.2 การทำงานระหว่าง ESP-WROOM-32 กับ พัดลมระบายอากาศ

จากการทดลองการทำงานของพัดลมระบายอากาศพบว่า เมื่อแบ่งการทำงานของพัดลมระบายอากาศออกเป็น 2 ระดับ โดยแบ่งเป็นระดับแรกคือ พัดลมระบายอากาศมีความเร็วปานกลาง จะแสดงผลบนหน้าจอมอนิเตอร์เป็น “PWM Fan slow” โดยวัดเทียบกับอุปกรณ์วัดความเร็วลมแบบใบพัด ได้ค่า 1542 rpm ซึ่งเมื่อเทียบกับ PWM = 127 จะได้ Duty cycle = 49% คิดเป็นครึ่งหนึ่งของพัดลมระบายอากาศนี้ที่มีค่า RPM มากสุด 3000 rpm ดังรูปที่ 4.7 และอีกระดับคือ ความเร็วสูงสุด จะแสดงผลบนหน้าจอมอนิเตอร์เป็น “PWM Fan high” โดยวัดเทียบกับอุปกรณ์วัดความเร็วลมแบบใบพัด ได้ค่า 2252 rpm ซึ่งเมื่อเทียบกับ PWM = 255 จะได้ Duty cycle = 100% ดังนั้น RPM ก็ควรจะมีค่าเท่ากับ 3000 rpm แต่ในที่นี้พบว่า วัดเทียบกับอุปกรณ์วัดความเร็วลมแบบใบพัด ได้ค่า 2252 rpm ดังรูปที่ 4.6

รูปที่ 4.6 การทำงานของพัดลมระบายอากาศที่ความเร็วปานกลาง

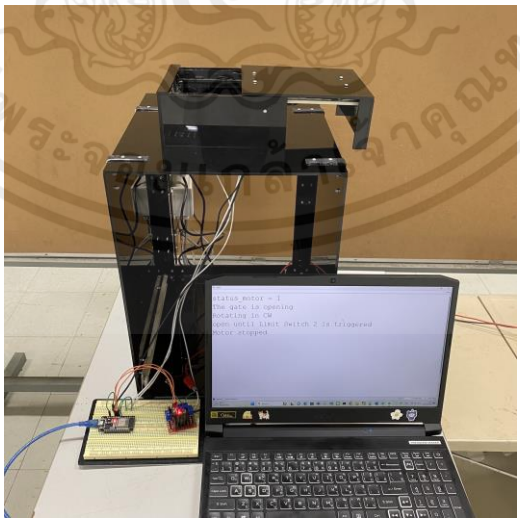
รูปที่ 4.7 อุปกรณ์วัดความเร็วลมแบบใบพัดที่ความเร็วปานกลาง

รูปที่ 4.8 การทำงานของพัดลมระบายอากาศที่ความเร็วสูงสุด

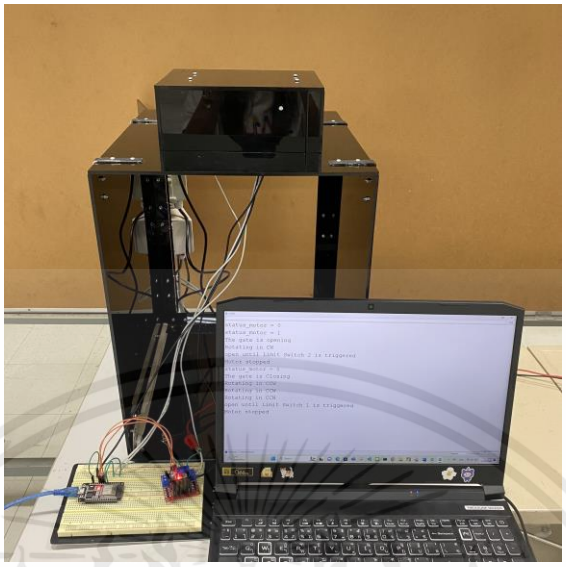
รูปที่ 4.9 อุปกรณ์วัดความเร็วลมแบบใบพัดที่ความเร็วสูงสุด

4.1.3 การทำงานระหว่าง ESP-WROOM-32 กับ ประตูระบายอากาศ

จากการทดลองการทำงานของประตูระบายอากาศผ่านมอเตอร์ไดร์ฟเวอร์ที่สั่งงานบน ESP-WROOM-32 โดยการทำงานของระบบจะเป็นไปตามปริมาณแก๊สไฮโดรเจนซัลไฟด์ กล่าวคือ เมื่อปริมาณแก๊สไฮโดรเจนซัลไฟด์มีค่าเพิ่มมากขึ้น ประตูระบายอากาศจะเริ่มทำการเปิดประตูออก แล้วพัดลมระบายอากาศจะสามารถถ่ายเท หรือระบายอากาศจากภายในกล่องที่เต็มไปด้วยแก๊สไฮโดรเจนซัลไฟด์ให้มีค่าลดน้อยลง ดังรูป 4.10 จากนั้นเมื่อแก๊สไฮโดรเจนซัลไฟด์ให้มีค่าลดน้อยลงจนถึงขีดที่ตั้งไว้ประตูระบายอากาศจะทำการปิดทันที ดังรูป 4.10

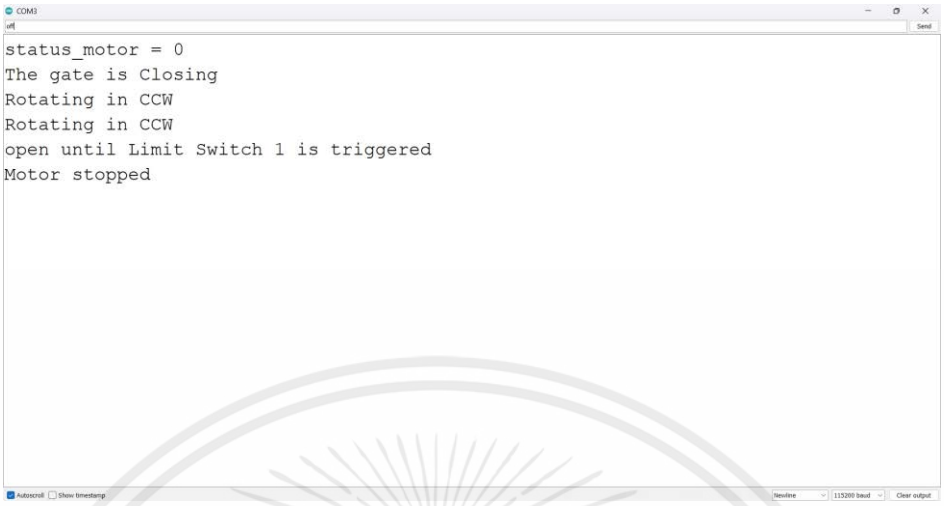


รูปที่ 4.10 ประตูระบายอากาศเปิด



รูปที่ 4.11 ประตูระบายอากาศปิด

รูปที่ 4.12 หน้าจอมอนิเตอร์แสดงการทำงานของประตูระบายอากาศที่ทำการเปิด

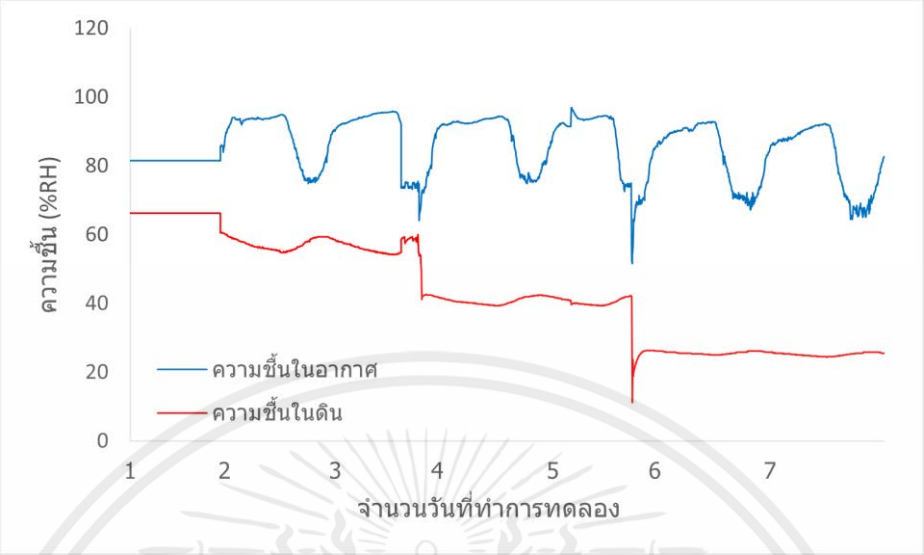


รูปที่ 4.13 หน้าจอมอนิเตอร์แสดงการทำงานของประตูระบายอากาศที่ทำการปิด

4.1.4 การจำลองระบบการทำงานของเซนเซอร์

4.1.4.1 การเก็บค่าความชื้นในอากาศ และในดิน

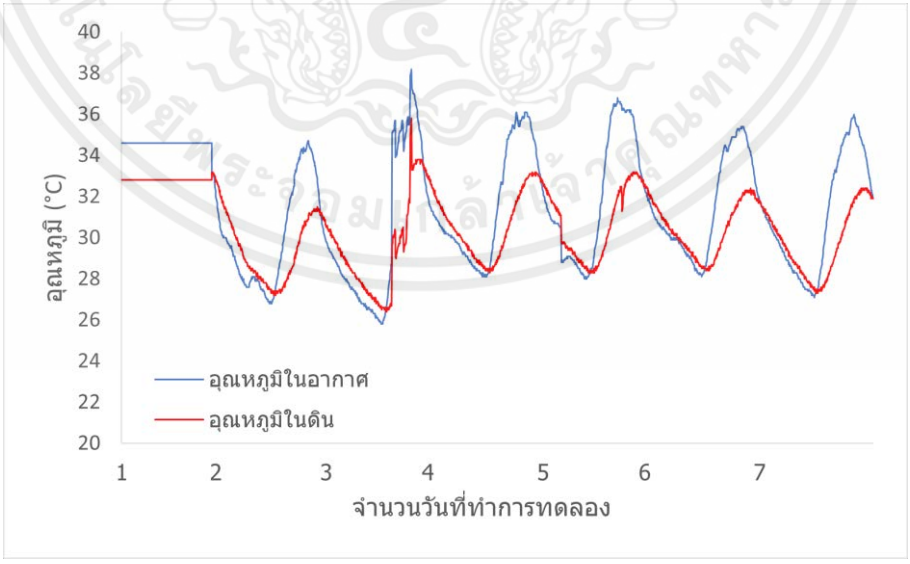
จากการทดลองระบบการทำงานจำลองในระยะเวลา 7 วัน โดยเริ่มเก็บค่าที่เวลา 12.00 นาฬิกา พบว่าค่าความชื้นเริ่มต้นในอากาศ และในดินมีค่าต่างกันเพียงเล็กน้อย เมื่อเวลาผ่านไปความชื้นในอากาศจะมีการเปลี่ยนแปลงในระหว่างวันมากกว่าความชื้นในดิน คือในช่วงกลางวันความชื้นจะมีค่าลดลง และเริ่มเพิ่มขึ้นในเวลากลางคืน แต่ความชื้นในดินจะค่อยๆ ลดลงอย่างต่อเนื่อง และลดลงมากในทุกๆ 2 วัน ในช่วงท้ายของวันที่ 5 ในการทดลอง ความชื้นในอากาศ และในดินจะลดลงอย่างรวดเร็วเนื่องจากเป็นช่วงที่มีการเติมแคลเซียมคาร์บอเนต(ปูนขาว) เพื่อปรับค่า pH ในดิน ดังรูปที่ 4.14



รูปที่ 4.14 กราฟการทดลองการเก็บค่าความชื้น

4.1.4.2 การเก็บค่าอุณหภูมิในอากาศ และในดิน

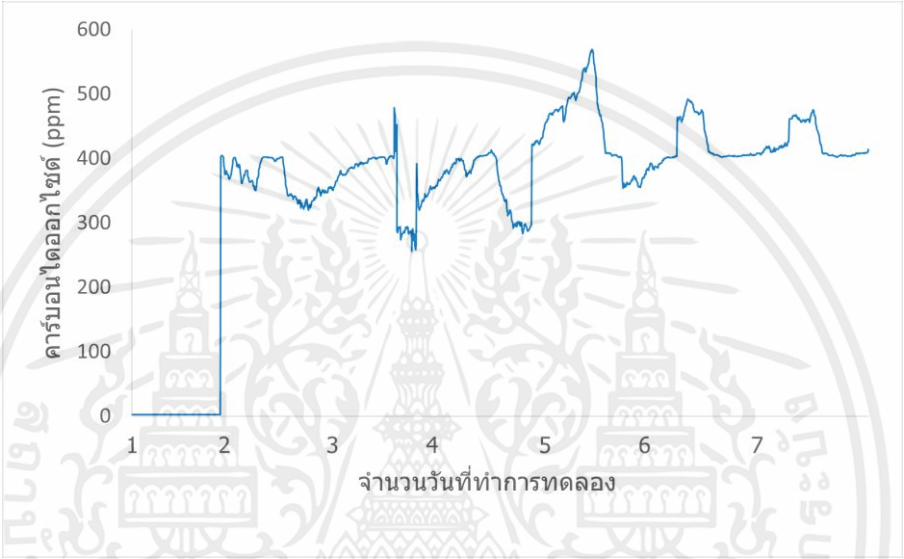
จากการทดลองระบบการทำงานจำลองในระยะเวลา 7 วัน โดยเริ่มเก็บค่าที่เวลา 12.00 นาฬิกา พบว่าค่าอุณหภูมิมีการเปลี่ยนแปลงในระหว่างวัน คือในช่วงกลางวันค่าอุณหภูมิจะมีค่าสูงสุดทั้งในอากาศและในดิน และในช่วงเวลากลางคืนอุณภูมิจะเริ่มลดลง โดยค่าอุณหภูมิของในอากาศ และในดิน จะอยู่ในช่วง 25 – 40 องศาเซลเซียส ตลอดการทดลอง และค่าอุณหภูมิในอากาศจะสูงกว่าค่าอุณภูมิในดินเพียงเล็กน้อย ดังรูปที่ 4.15



รูปที่ 4.15 กราฟการทดลองการเก็บค่าอุณหภูมิ

4.1.4.3 การเก็บค่าคาร์บอนไดออกไซด์

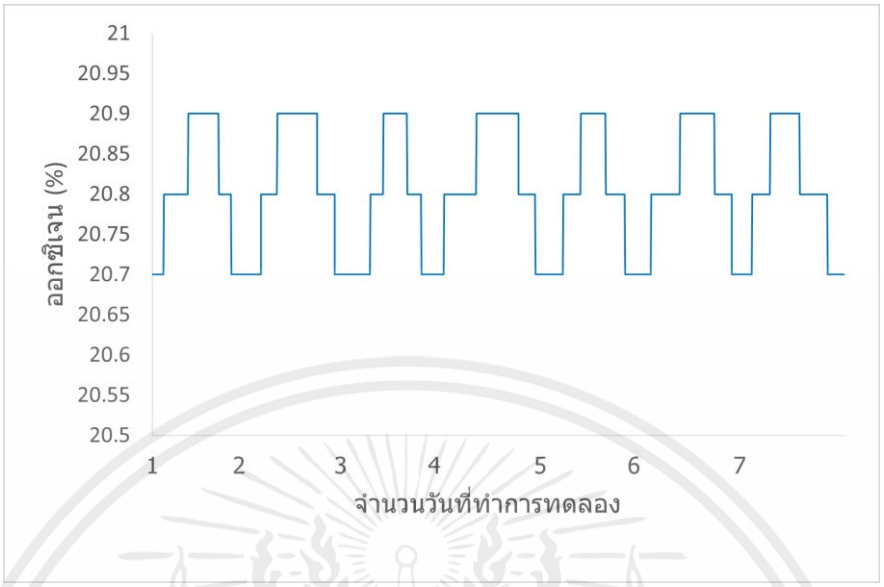
จากการทดลองระบบการทำงานจำลองในระยะเวลา 7 วัน โดยเริ่มเก็บค่าที่เวลา 12.00 นาฬิกา พบว่าค่าคาร์บอนไดออกไซด์เมื่อเริ่มต้นการทดลองจะมีค่าอยู่ที่ 2 ppm ซึ่งเกิดจากการที่แบตเตอรี่จ่ายไฟไม่เพียงพอ หลังจากนั้นเมื่อมีการเปลี่ยนแบตเตอรี่อีกตัวเพื่อจ่ายไฟแทนตัวก่อนหน้า ค่าคาร์บอนไดออกไซด์ขยับมาอยู่ในช่วง 200 – 600 ppm ดังรูปที่ 4.16



รูปที่ 4.16 กราฟการทดลองการเก็บค่าคาร์บอนไดออกไซด์

4.1.4.4 การเก็บค่าออกซิเจน

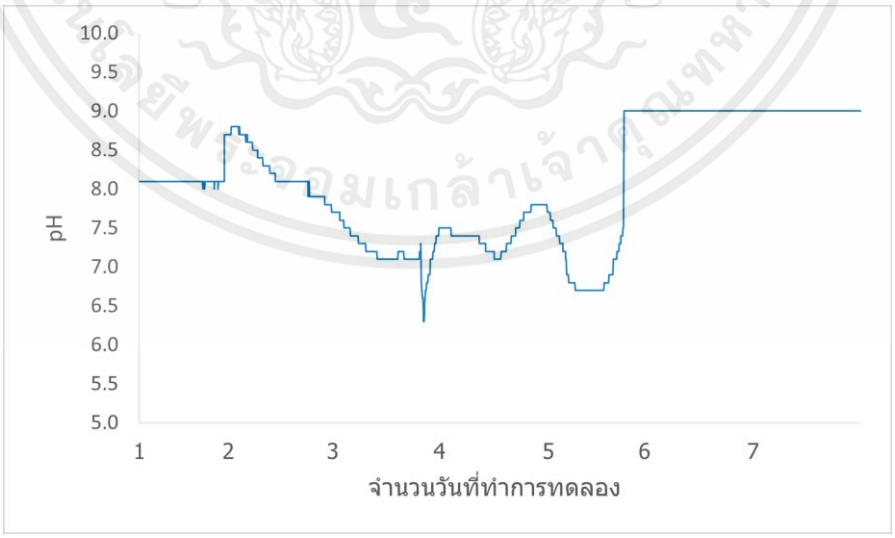
จากการทดลองระบบการทำงานจำลองในระยะเวลา 7 วัน โดยเริ่มเก็บค่าที่เวลา 12.00 นาฬิกา พบว่าค่าออกซิเจนมีการการเปลี่ยนแปลงค่อนข้างคงที่ตลอดการทดลอง คือในช่วงกวางวันค่าออกซิเจนลดลงเพียงเล็กน้อย และจะเพิ่มขึ้นเล็กน้อยในช่วงเวลากลางคืน ดังรูปที่ 4.17



รูปที่ 4.17 กราฟการทดลองการเก็บค่าออกซิเจน

4.1.4.5 การเก็บค่า pH

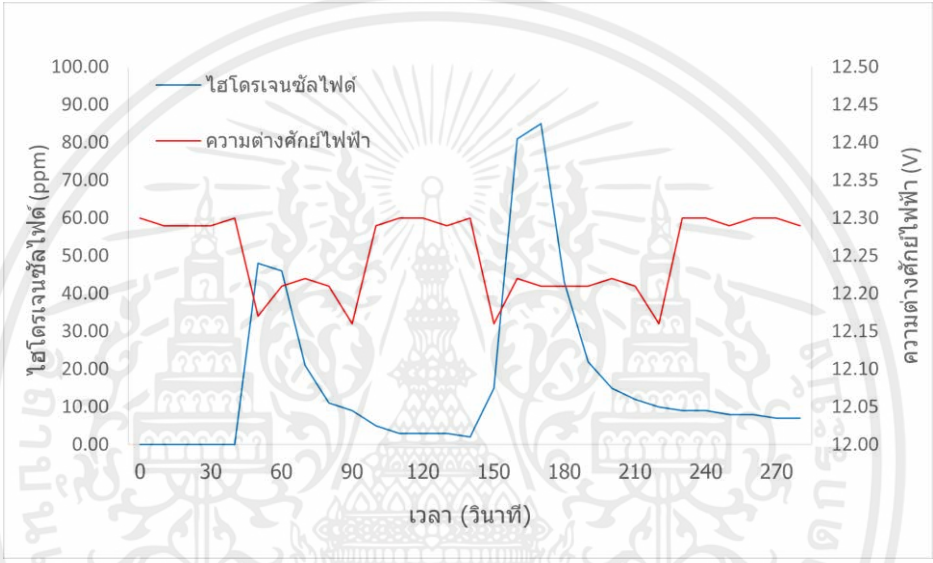
จากการทดลองระบบการทำงานจำลองในระยะเวลา 7 วัน โดยเริ่มเก็บค่าที่เวลา 12.00 นาฬิกา พบว่า pH ในช่วงเริ่มต้นการทดลองจะอยู่ที่ 8 และเพิ่มไป 9 เมื่อเวลาผ่านไป 1 วัน ค่า pH จะเริ่มลดลงอย่างต่อเนื่องจนกระทั่งดินมีความเป็นกรด หลังจากนั้นในช่วงท้ายของวันที่ 5 ในการทดลอง จึงได้ทำการเติมแคลเซียมคาร์บอเนต(ปูนขาว) เพื่อปรับสภาพดินให้มีความเป็นเบส และจะเห็นได้ว่ากราฟค่า pH จะกลับมาอยู่ที่ 9 อย่างคงที่จนจบการทดลอง ดังรูปที่ 4.18



รูปที่ 4.18 กราฟการทดลองการเก็บค่า pH

4.1.4.6 การเก็บค่าไฮโดรเจนซัลไฟด์

จากการทดลอง พบว่าเมื่อค่าไฮโดรเจนซัลไฟด์เพิ่มมากขึ้นเกินกว่าที่กำหนดไว้ ค่าความต่างศักย์ไฟฟ้าจะลดลง เนื่องจากปริมาณไฮโดรเจนซัลไฟด์มีค่าเพิ่มมากขึ้น พัดลมระบายอากาศจะเริ่มการทำงาน จากนั้นเมื่อปริมาณไฮโดรเจนซัลไฟด์ลดลงแล้ว พัดลมระบายอากาศจะหยุดการทำงาน ทำให้ค่าความต่างศักย์ไฟฟ้าเพิ่มขึ้นมาใกล้เคียงกับในช่วงเริ่มต้นของการทดลอง ดังรูปที่ 4.19

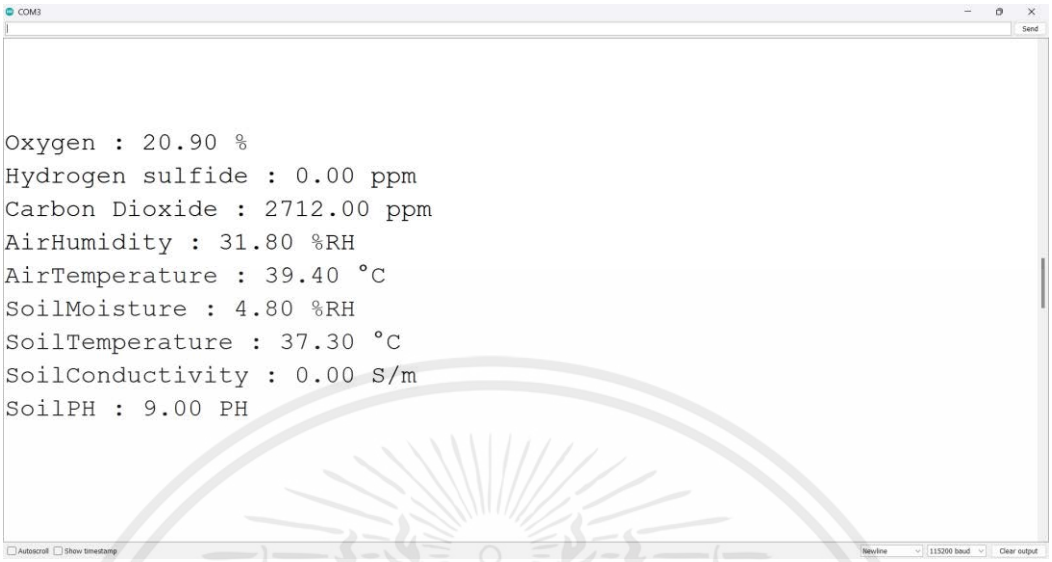


รูปที่ 4.19 กราฟการทดลองการเก็บค่าไฮโดรเจนซัลไฟด์

4.2 การทดสอบโดยรวมของชิ้นงาน

4.2.1 การส่งชุดข้อมูลจากกล่องจำลองการเลี้ยงแมลง

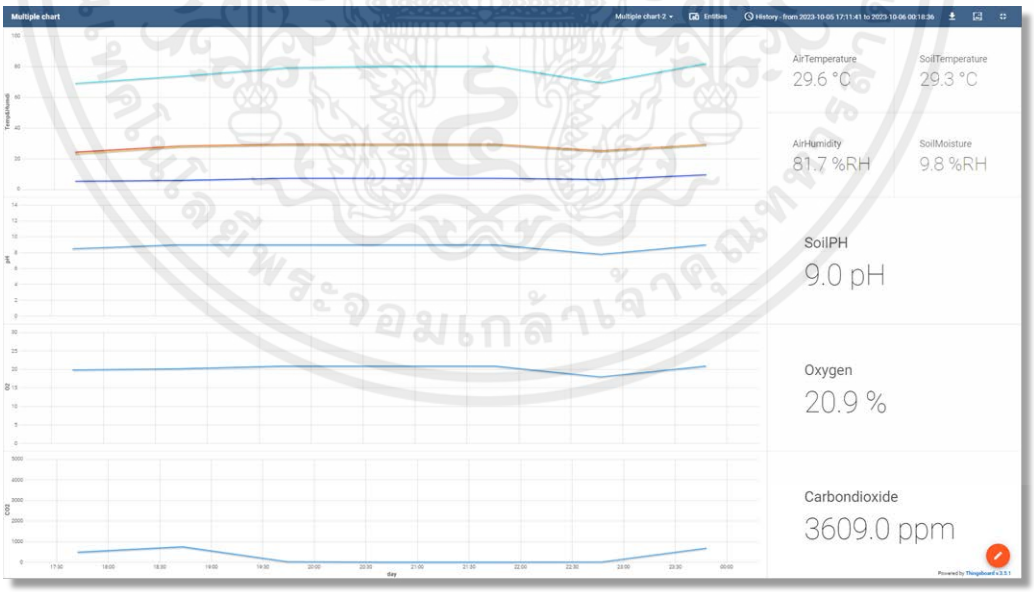
การส่งข้อมูลจากเซนเซอร์บนโปรโตคอล RS485 จะถูกต่อแบบอนุกรมเข้าตัวแปลงสัญญาณ (MAX485) เข้าขาดิจิตอลที่ ESP-WROOM-32 โดยรับค่าผ่านโปรแกรม Arduino IDE ค่าที่ได้รับจะแสดงบนจอ serial monitor



รูปที่ 4.20 หน้าจอ serial monitor แสดงค่าข้อมูลที่วัดได้

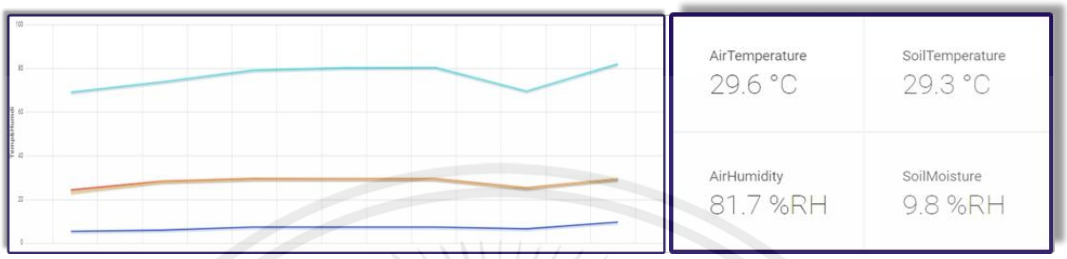
4.2.2 การสร้างระบบรับข้อมูลรายงานสภาพอากาศ

หน้าจอแสดงผลแบบเรียลไทม์หรือแดชบอร์ด เมื่อค่าที่ได้รับจากESP-WROOM-32 ถูกส่งมาแพลตฟอร์ม ThingsBoard เพื่อจัดทำการรวบรวมข้อมูลในแต่ละช่วงเวลาแสดงค่าการทำงาน ณ ขณะนั้น โดยให้ค่าแกน Y เป็นค่าข้อมูล และค่าแกน X เป็นค่าเวลาตามรูปที่ 4.21

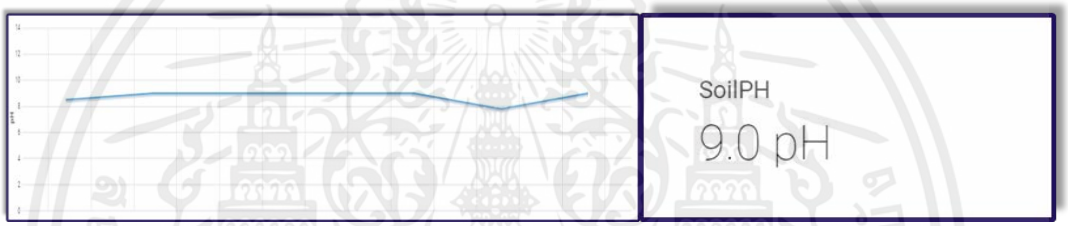


รูปที่ 4.21 หน้าจอแสดงผลแบบเรียลไทม์

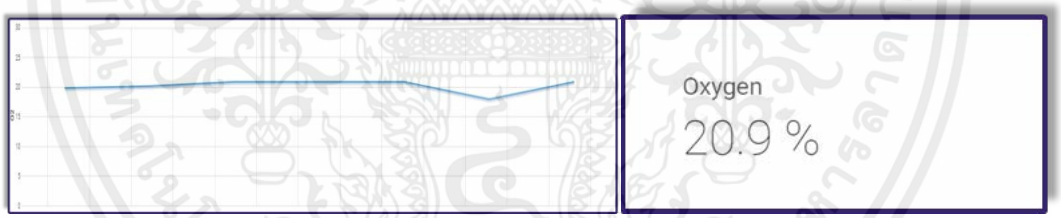
หน้าจอแสดงผลแบบเรียลไทม์จะทำการแสดงข้อมูลภายในอากาศโดยประกอบด้วย อุณหภูมิ ความชื้น แก๊สไฮโดรเจนซัลไฟด์ แก๊สออกซิเจน แก๊สคาร์บอนไดออกไซด์ และภายในดินจะประกอบด้วยค่าความเป็นกรดด่าง อุณหภูมิ และความชื้น ตามรูปที่ 4.22



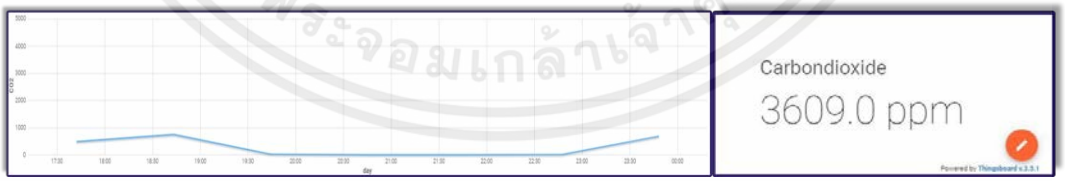
รูปที่ 4.22 อุณหภูมิ ความชื้นภายในอากาศ และดิน



รูปที่ 4.23 ค่าความเป็นกรดด่างภายในดิน



รูปที่ 4.24 ค่าแก๊สออกซิเจนภายในอากาศ

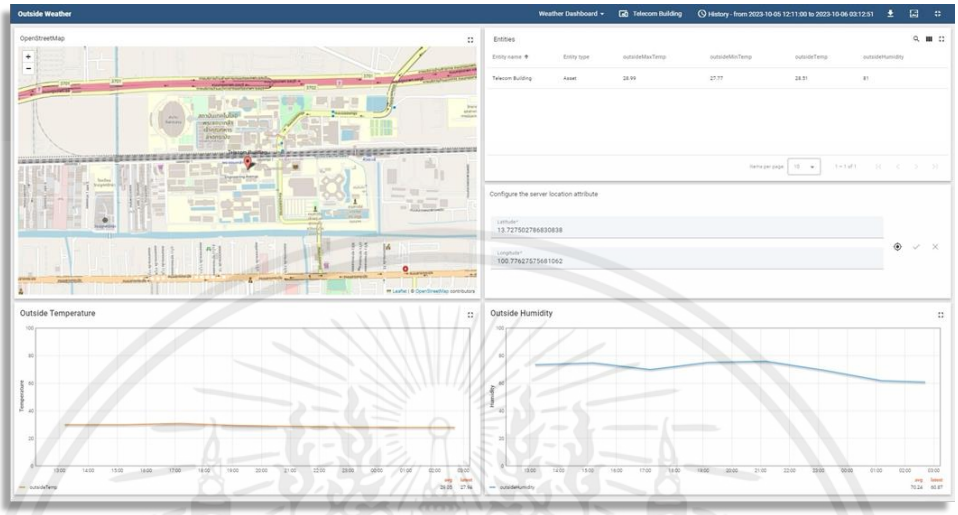


รูปที่ 4.25 ค่าแก๊สไฮโดรเจนซัลไฟด์ในอากาศ

4.2.3 การแสดงค่าพิกัด และการรายงานสภาพอากาศจากกล่องจำลอง

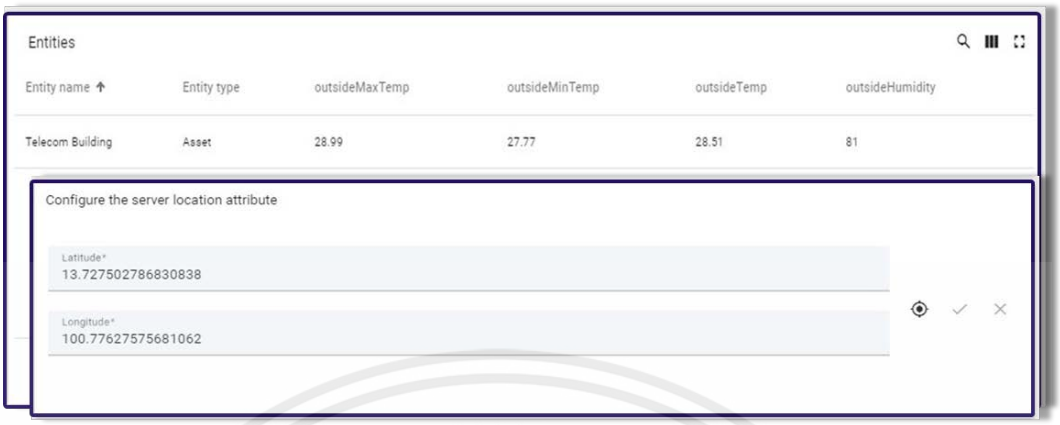
เนื่องจากตัวกล่องจำลองการเลี้ยงแมลงไม่สามารถระบุตำแหน่งที่อยู่ได้ ทางผู้จัดทำจึงทำการระบุพิกัดกล่องเพื่อแสดงบนหน้าเว็บไซต์ และทำการใช้ REST API เพื่อนำค่าข้อมูลจากผู้

ให้บริการเว็บไซต์ที่รวบรวมค่าทางสภาพอากาศมาแสดงบนเว็บ ThingsBoard เพื่อแสดงค่าอากาศตามพิกัด ณ ปัจจุบัน ตามรูปที่ 4.26

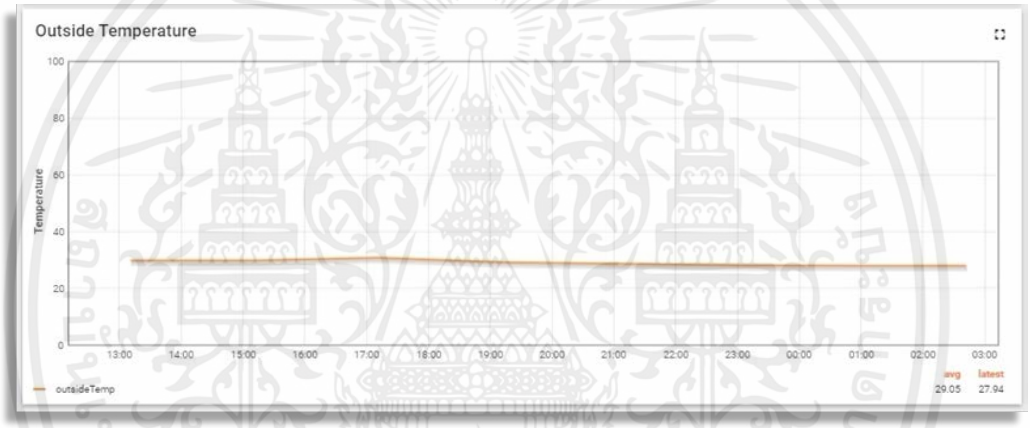


รูปที่ 4.26 หน้าจอแสดงพิกัดของกล่องจำลองการเลี้ยงแมลง และรายงานสภาพอากาศจาก REST API

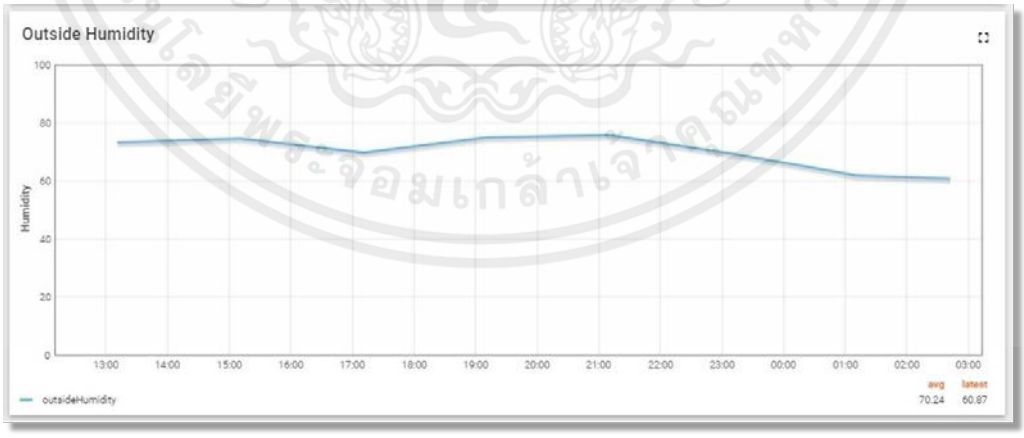
รูปที่ 4.27 หน้าจอแสดงพิกัดบนแมพ



รูปที่ 4.28 การระบุพิกัดกล่องจำลองทางการเกษตร



รูปที่ 4.29 หน้าจอแสดงค่าอุณหภูมิโดยรอบจาก REST API

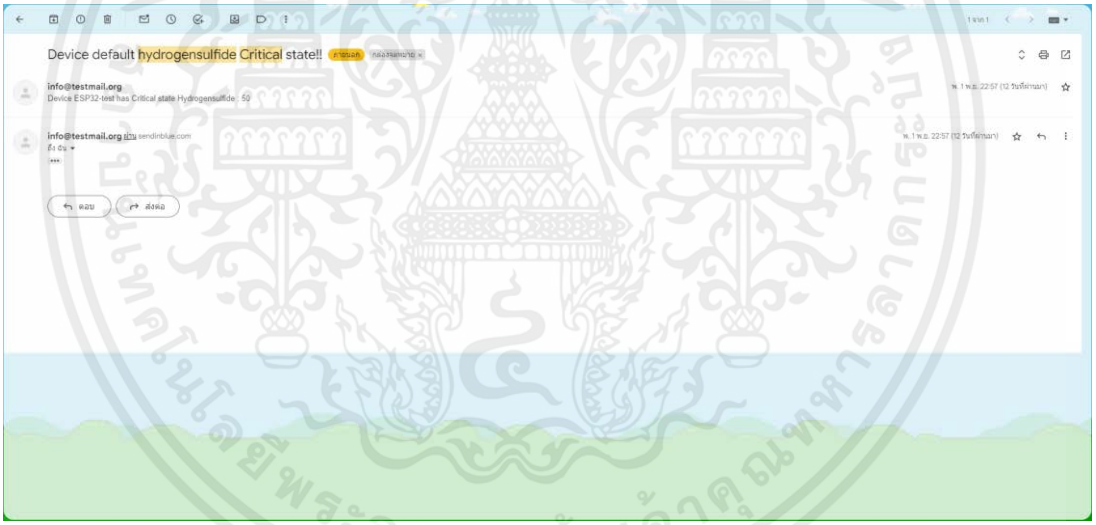


รูปที่ 4.30 หน้าจอแสดงความชื้นโดยรอบจาก REST API

4.2.4 การทำงานของระบบแจ้งเตือน

เมื่อค่าข้อมูลที่ได้รับจากกล่องจำลองทางการเกษตรมีค่ามากกว่าที่กำหนด แพลตฟอร์ม ThingsBoard จะทำการใช้โปรโตคอล SMTP เพื่อส่งค่าไปยังอีเมลที่ใส่ไว้

รูปที่ 4.31 หน้าจอแสดงผลเมื่อค่าที่ได้รับน้อยกว่าที่กำหนด



รูปที่ 4.32 ระบบแจ้งเตือนผ่านอีเมลที่กำหนดไว้

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์เรื่อง ออกแบบและพัฒนาการตรวจสอบภาพแวดล้อมที่เหมาะสำหรับการเลี้ยงแมลงในระบบโรงเรือน โดยผู้จัดทำออกแบบระบบเลี้ยงแมลงในระบบโรงเรือนหรือกล่องจำลองการเลี้ยงแมลงโดยคำนึงถึง 2 ด้าน คือ 1. ด้านการเพาะเลี้ยงแมลง โดยจะออกแบบ และพัฒนาระบบตรวจจับสภาพแวดล้อมภายในกล่องจำลองการเลี้ยงแมลงที่ส่งข้อมูลแบบไร้สาย 2. ด้านการกำจัดของเสียทางอากาศ โดยส่วนนี้จะทำงานแบบอัตโนมัติ และมีระบบแจ้งเตือนเมื่อเกิดความผิดปกติในระบบ ซึ่งในโครงงานนี้เป็นเพียงการจำลองสถานการณ์เพื่อตรวจจับสภาพแวดล้อมภายในกล่องและได้ทำการศึกษาทฤษฎีเกี่ยวกับวงจรชีวิตของแมลงเพื่อออกแบบกล่องจำลอง แต่ไม่ได้มีการเลี้ยงจริงเนื่องเป็นการรบกวนสิ่งมีชีวิต โดยได้ศึกษาทฤษฎี และหลักการใช้งานของอุปกรณ์ เพื่อนำมาประยุกต์ใช้ในการทำปริญญานิพนธ์ และสามารถสรุปผลการดำเนินงานได้ ดังนี้

1) สามารถออกแบบ และสร้างกล่องจำลองการเลี้ยงแมลงที่ส่งข้อมูลแบบไร้สายจาก EPS32 ไปยังแพลตฟอร์ม ThingsBoard บนคอมพิวเตอร์ได้
2) สามารถจัดทำเว็บแสดงผลรายงานสภาพแวดล้อมแบบเรียลไทม์ของระบบผ่านแพลตฟอร์ม Thingsboard และจัดเก็บข้อมูลในฐานข้อมูลของ PosgetSQL
3) สามารถออกแบบกำจัดของเสียด้วยระบบพัดลมระบายอากาศแก่กล่องจำลองการเลี้ยงแมลงเมื่อได้รับแจ้งเตือนว่าปริมาณข้อมูลภายในอากาศสูงเกินที่กำหนด
4) เมื่อออกแบบระบบแล้ว สามารถนำค่าข้อมูลมาเพิ่มประสิทธิภาพในการทำงานของกล่องจำลองเลี้ยงแมลง รวมถึงช่วยประเมินและวิเคราะห์ของข้อมูล

5.2 ข้อเสนอแนะ

ปริญญานิพนธ์เรื่อง ออกแบบและพัฒนาการตรวจสอบภาพแวดล้อมที่เหมาะสำหรับการเลี้ยงแมลงในระบบโรงเรือนสามารถทำการวัดสภาพแวดล้อมได้จริง แต่ยังมีข้อจำกัดอยู่คือ

1) กล่องจำลองการเลี้ยงแมลงไม่มีระบบจีพีเอส ทำให้ไม่สามารถติดตามพิกัดได้แบบเรียลไทม์ และในปัจจุบันต้องทำการกำหนดพิกัดเอง

2) แบตเตอรี่ภายในระบบจ่ายไฟไม่สามารถดูระดับคงเหลือได้ จึงควรติดตั้งพาวเวอร์ซัพพลายเพื่อต่อจากไฟบ้าน 220 โวลต์เข้ากับระบบจ่ายไฟ
3) เมื่อระบบจ่ายไฟของกล่องจำลองการเลี้ยงแมลงหมด ระบบฐานข้อมูลจะไม่สามารถทำงานได้

บรรณานุกรม

[1] นงลักษณ์ อัจนปัญญา. “ตุนเสบียง เตรียม อาหารอวกาศ ที่พัฒนาถึงขั้นเตรียมเดลิเวอรีเมนโปรดตามใจนักบินสั่ง.” https://www.sarakadeelite.com/lite/food-in-space/.

[2] NASA. “FY 2024 Budget Estimates.” https://www.nasa.gov/wp-content/uploads/2023/03/nasa-fy-2024-cj-v3.pdf.

[3] นพพร จูจันทร์. “Arduino.” http://elec2web.blogspot.com/2016/04/arduino-uno-atmega328.html.

[4] Artronshop. “ESP32 เบื้องต้น.” artronshop.co.th/article/51/esp32-เบื้องต้น-บทที่-1-แนะนำ-esp32

[5] Arduino4. “MAX485 Module.” https://www.arduino4.com/product/301/max485-module-rs485-module-ttl-to-rs-485-module.

[6] Arduitronics. “Industrial Soil Moisture & Temperature Sensor MODBUS-RTU RS485.” https://www.arduitronics.com/product/5313/industrial-soil-moisture-temperature-sensor-modbus-rtu-rs485-0-2v-analog-voltage-s-soil-mt-02a-%E0%B9%81%E0%B8%97%E0%B9%89%E0%B8%88%E0%B8%B2.

[7] Sumtech. “Water Proof Temperature Humidity Sensor RS485.” https://www.sumtechstores.com/product/521/water-proof-temperature-humidity-sensor-rs485-3.

[8] Miniature-solution. “เซ็นเซอร์ก๊าซไฮโดรเจนซัลไฟด์ H2S Hydrogen sulfide gas sensor RS485 output.” https://www.miniature-solution.com/product/-h2s-hydrogen-sulfide-gas-sensor-rs485-output.

[9] Miniature-solution. “Soil Ph Sensor สำหรับวัดค่า พีเอช ในดินเชิงการเกษตร OUTPUT RS485.” https://www.miniature-solution.com/product/27/soil-ph-sensor-output-rs485.

[10] บริษัท โอเมก้า เมชเชอริ่ง อินสทรูเม้นท์ จำกัด. “มาตรฐาน IP.”
https://www.omi.co.th/th/article/%E0%B8%A1%E0%B8%B2%E0%B8%95%E0%B8%A3%E0%B8%90%E0%B8%B2%E0%B8%99-ip.

[11] Nectec. “การสื่อสารในงานอุตสาหกรรมด้วยโพรโทคอล Modbus.” https://www.nectec.or.th/news/news-public-document/modbusprotocol.html.

[12] Hmong. “ทรานซิสเตอร์–ทรานซิสเตอร์ลอจิก.” https://hmong.in.th/wiki/Transistor-transistor_logic.

[13] Thaiconfig. “TCP.” https://thaiconfig.com/network/tcp-.

[14] บริษัท โอเมก้า เมชเชอริ่ง อินสทรูเม้นท์ จำกัด. “RS485.” https://www.omi.co.th/th/article/rs485.

[15] sumtech technology international co. ltd. “Carbon dioxide Sensor RS485.” https://www.sumtechstores.com/product/517/co2-sensor-transmitter-rs485-output-2

[16] sumtech technology international co. ltd. “Soil Temperature Moisture PH and EC Sensor RS485.” https://www.sumtechstores.com/product/116/soil-npkphectemperaturemoisture-rs485

[17] บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด. “Raspberry Pi 4 – 8GB RAM.” https://inex.co.th/home/product/raspberry-pi-4-8gb-ram/.

[18] Arduino4. “L298N Motor Driver Module โมดูลขับมอเตอร์ ควบคุมมอเตอร์.” https://www.arduino4.com/product/40/l298n-motor-driver- module.

[19] Zoey Zhang. “วงจรควบคุมความเร็ว DC FAN.” https://th.fmuser.net/content/?12048.html

[20] บุลวัชร์ เจริญยืนนาน. “รีเลย์ คืออะไร มีหลักการทำงานอย่างไรบ้าง.” https://misumitechnical.com/technical/electrical/relay-working-principles/

[21] Sumipol. “ลิมิตสวิตซ์.” https://www.sumipol.com/knowledge/what-is-limit-switch-omron/.

[22] AllNewStep. “โมดูลเรกูเลเตอร์ LM2596 DC.” https://www.allnewstep.com/product/145/-lm2596-dc-lm2596s-lm2596-4-35v-input-voltage-dc-dc-step-down-adjustable-power-suppl.

[23] Amazon Web Services (AWS). “MQTT คืออะไร.” https://aws.amazon.com/th/what-is/mqtt/.

[24] Amazon Web Services (AWS). “API (Application Programming Interfaces) คืออะไร.” https://aws.amazon.com/th/what-is/api/.

[25] SMTP. “SMTP คืออะไร.” https://www.mindphp.com/.html#google_vignette

[26] มาโนชญ์ แสงศิริ. “Arduino ผู้นำด้านฮาร์ดแวร์และระบบนิเวศซอฟต์แวร์แบบเปิดระดับโลก.” https://www.scimath.org/article-technology/item/9815-arduino.

[27] Amazon Web Services (AWS). “MySQL และ PostgreSQL แตกต่างกันอย่างไร.” https://aws.amazon.com/th/compare/the-difference-between-mysql-vs-postgresql/

[28] Appomax. “Thingsboard คืออะไร.” https://www.appomax.co/thingsboard.

[29] บริษัทบิมอ็อบเจ็คท์ (ไทยแลนด์) จำกัด. “รู้จัก SketchUp โปรแกรมยอดฮิต.” https://bimspaces.com/blog/free-model-sketchup/

[30] Cyber Elite. “Machine Learning เทคโนโลยีประโยชน์ครอบจักรวาล.” https://www.cyberelite.co.th/blog/machine-learning/

[31] EDA. “ประสบการณ์ใช้งาน EDA ที่ง่ายกว่า.” https://easyeda.com/th

[32] นรินาม. “Visual Programming คืออะไรและทำงานอย่างไร.” https://appmaster.io/th/blog/kaarekhiiynopraekrmdwyphaaphkhuue-aairaelathamngaan-yaangair

ภาคผนวก

ESP-WROOM-32 Coding.

```
#include <ModbusMaster.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <ArduinoJson.h>
#include <WiFi.h>
#include <TridentTD_LineNotify.h>
#include <PubSubClient.h>
#include <INA226_WE.h>
#include "FS.h"
#include "SD.h"
#include <SPI.h>
#include "time.h"

#define WIFI_AP "Gege"
#define WIFI_PASSWORD "0914304309"
#define TOKEN "0RswMVHFaZzBjWpKX3rP"
#define LINE_TOKEN  "DUFwtQvbZU76WXrtgI0dVBiua7EylBaYgmLGgESBgR1"
#define RELAY_PIN 32
#define RE 4
#define DE 2
#define I2C_ADDRESS 0x40

int ENA = 14;
int IN1 = 27;
int IN2 = 26;
int fan_pwm_pin = 3;
int rpm;
int motorActivated = 0;
int motorActivated2 = 0;
int status = WL_IDLE_STATUS;
String backupdata;
```

```
float val2 = 0;
float val10 = 0;
float val11 = 0;
float val12 = 0;
float val13 = 0;
float val14 = 0;
float val15 = 0;
char thingsboardServer[] = "172.20.10.8";
const int limitSwitch1Pin = 25;
const int limitSwitch2Pin = 33;
const char* ntpServer = "pool.ntp.org";
const unsigned long timeoutPeriod = 5000;
unsigned long lastRequestTime = 0;
unsigned long epochTime;
unsigned long getTime() {
  time_t now;
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
    return(0);
  }
  time(&now);
  return now;
}

INA226_WE ina226 = INA226_WE(I2C_ADDRESS);
ModbusMaster node1;
ModbusMaster node2;
ModbusMaster node3;
ModbusMaster node4;
ModbusMaster node5;
```

```
SoftwareSerial mod(16, 17);
WiFiClient wifiClient;
PubSubClient client(wifiClient);
File dataFile;

void setup() {
  Serial.begin(115200);
  Serial2.begin(4800);
  delay(10);
  InitWiFi();
  Wire.begin();
  ina226.init();
  ina226.setAverage(AVERAGE_16);
  ina226.setConversionTime(CONV_TIME_1100);
  ina226.setMeasureMode(CONTINUOUS);
  ina226.setResistorRange(0.05, 10.0);
  ina226.setCorrectionFactor(0.93);
  ina226.waitUntilConversionCompleted();
  initSDCard();
  configTime(0, 0, ntpServer);
  LINE.setToken(LINE_TOKEN);
  pinMode(RE, OUTPUT);
  pinMode(DE, OUTPUT);
  digitalWrite(RE, 0);
  digitalWrite(DE, 0);
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  pinMode(limitSwitch1Pin, INPUT_PULLUP);
```

```
  pinMode(limitSwitch2Pin, INPUT_PULLUP);
  pinMode(fan_pwm_pin, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);
  node1.begin(1, Serial2);
  node2.begin(2, Serial2);
  node3.begin(3, Serial2);
  node4.begin(4, Serial2);
  node5.begin(5, Serial2);
  node1.preTransmission(preTransmission);
  node1.postTransmission(postTransmission);
  node2.preTransmission(preTransmission);
  node2.postTransmission(postTransmission);
  node3.preTransmission(preTransmission);
  node3.postTransmission(postTransmission);
  node4.preTransmission(preTransmission);
  node4.postTransmission(postTransmission);
  node5.preTransmission(preTransmission);
  node5.postTransmission(postTransmission);
  client.setServer(thingsboardServer, 1883);
  client.setCallback(on_message);
}

void loop() {
  Serial.println("Inside loop");
   if (!client.connected()) {
    reconnect();
  }
  client.loop();
  unsigned long currentTime = millis();
  static unsigned long lastUpdateTime = 0;
  if (currentTime - lastUpdateTime >= 30000) {
```

```
float val1 = Cal_oxygen();
float val2 = Cal_hydrogensulfide();
delay(2000);
float val3 = Cal_carbondioxide();
delay(2000);
float val4 = Cal_Airhumidity();
float val5 = Cal_Airtemperature();
float val6 = Cal_Soilmoisture();
float val7 = Cal_Soiltemperature();
float val8 = Cal_Soilconductivity();
float val9 = Cal_SoilPH();
readINA226Data(val10, val11, val12, val13, val14);
Serial.print("Oxygen : ");
Serial.print(val1);
Serial.println(" % ");
Serial.print("Hydrogen sulfide : ");
Serial.print(val2);
Serial.println(" ppm ");
Serial.print("Carbon Dioxide : ");
Serial.print(val3);
Serial.println(" ppm ");
Serial.print("AirHumidity : ");
Serial.print(val4);
Serial.println(" %RH ");
Serial.print("AirTemperature : ");
Serial.print(val5);
Serial.println(" °C ");
Serial.print("SoilMoisture : ");
Serial.print(val6);
Serial.println(" %RH ");
Serial.print("SoilTemperature : ");
```

```
Serial.print(val7);
Serial.println(" ℃ ");
Serial.print("SoilConductivity : ");
Serial.print(val8);
Serial.println(" S/m ");
Serial.print("SoilPH : ");
Serial.print(val9);
Serial.println(" PH ");
Serial.print("shuntVoltage : ");
Serial.print(val10);
Serial.println(" mV ");
Serial.print("busVoltage : ");
Serial.print(val11);
Serial.println(" V ");
Serial.print("current : ");
Serial.print(val12);
Serial.println(" mA ");
Serial.print("power : ");
Serial.print(val13);
Serial.println(" mW ");
Serial.print("loadVoltage : ");
Serial.print(val14);
Serial.println(" V ");
epochTime = getTime();
String dateTime = epochToDateTime(epochTime);
Serial.print("Date and time: ");
Serial.println(dateTime);
backupdata = String(epochTime) + ",sensor1:" + String(val1) + ",sensor2:" + String(val2)
                    + ",sensor3:" + String(val3) + ",sensor4:" + String(val4)
                    + "," + String(val5) + ",sensor5:" + String(val6) + ","
                    + String(val7) + "," + String(val8) + "," + String(val9)
```

```
                                + "\r\n";
    Serial.print("Saving data: ");
    Serial.println(backupdata);
    dataFile.println(backupdata.c_str());
    LINE.notify("Message appended");
    LINE.notify(backupdata);
    dataFile.close();
    delay(1000);
    dataFile = SD.open("/data/data.txt",FILE_APPEND);
    String telemetryData = "{\"Oxygen\":" + String(val1, 1) + ", \"Hydrogensulfide\":"
                                + String(val2, 1) + ", \"Carbondioxide\":" + String(val3, 1)
                                + ", \"AirHumidity\":" + String(val4, 1) + ", \"AirTemperature\":"
                                + String(val5, 1) + ", \"SoilMoisture\":" + String(val6, 1)
                                + ", \"SoilTemperature\":" + String(val7, 1)
                                + ", \"SoilConductivity\":" + String(val8, 1)
                                + ", \"SoilPH\":" + String(val9, 1) +"}";
    String telemetryData2 = "{\"ShuntVoltage\":" + String(val10, 1) + ", \"BusVoltage\":"
                                 + String(val11, 1) + ", \"Current_mA\":" + String(val12, 1)
                                 + ", \"BusPower\":" + String(val13, 1)   + ", \"TotalVoltage\":"
                                 + String(val14, 1) + "}";
    client.publish("v1/devices/me/telemetry", telemetryData.c_str());
    client.publish("v1/devices/me/telemetry", telemetryData2.c_str());
    Serial.println("Data sent to ThingsBoard");
    lastUpdateTime = currentTime;
  }
  delay(100);
}

void on_message(const char* topic, byte* payload, unsigned int length) {
  Serial.println("On message");
  char json[length + 1];
```

```
strncpy(json, (char*)payload, length);
json[length] = '\0';
Serial.print("Topic: ");
Serial.println(topic);
Serial.print("Message: ");
Serial.println(json);
StaticJsonDocument<256> doc;
DeserializationError error = deserializeJson(doc, json);
if (error) {
  Serial.print("Error parsing JSON: ");
  Serial.println(error.c_str());
  return;
}
const char* method = doc["method"];
int speed_fan = doc["params"]["speed_fan"];
int status_fan = doc["params"]["status_fan"];
int status_motor = doc["params"]["status_motor"];
Serial.print("Method: ");
Serial.println(method);
Serial.print("Speed Fan: ");
Serial.println(speed_fan);
Serial.print("Status Fan: ");
Serial.println(status_fan);
Serial.print("Status Motor: ");
Serial.println(status_motor);
  if (status_fan  == 0 && status_motor  ==  0 && motorActivated == 0) {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, 255);
  int pwm2 = 255;
  analogWrite(fan_pwm_pin, pwm2);
```

```
    digitalWrite(RELAY_PIN, HIGH);
    Serial.println("PWM Fan high");
    while (digitalRead(limitSwitch2Pin) == HIGH) {
    Serial.println("Rotating in CCW");
    delay(1000);
    }
    Stopmotor();
    motorActivated = 1;
    }
    if (status_fan == 1 && status_moter == 1 && motorActivated == 1 &&
        motorActivated2 == 0) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, 255); // Set maximum PWM speed
    digitalWrite(RELAY_PIN, LOW);
    Serial.println("PWM Fan Close");
    while (digitalRead(limitSwitch1Pin) == HIGH) {
    Serial.println("Rotating in CW");
    delay(1000);
    }
    Stopmotor();
    motorActivated2 = 1;
    }
}

void readINA226Data(float &val10, float &val11, float &val12, float &val13, float &val14)
{
  ina226.readAndClearFlags();
  val10 = ina226.getShuntVoltage_mV();
  val11 = ina226.getBusVoltage_V();
  val12 = ina226.getCurrent_mA();
```

```
  val13 = ina226.getBusPower();
  delay(1000);
  val14 = val11 + (val10 / 1000);
  val15 = (val10*50)/1000;
  Serial.print("current 2 : ");
  Serial.print(val15);
  Serial.println(" mA ");
}

void InitWiFi() {
  Serial.println("Connecting to AP ...");
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to AP");
}

void reconnect() {
  while (!client.connected()) {
    status = WiFi.status();
    if (status != WL_CONNECTED) {
      WiFi.begin(WIFI_AP, WIFI_PASSWORD);
      while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
      }
      Serial.println("Connected to AP");
    }
    Serial.print("Connecting to ThingsBoard node ...");
```

```
    if (client.connect("ESP8266 Device", TOKEN, NULL)) {
      Serial.println("[DONE]");
      client.subscribe("v1/devices/me/rpc/request/+");
    } else {
      Serial.print("[FAILED] [ rc = ");
      Serial.print(client.state());
      Serial.println(" : retrying in 5 seconds]");
      delay(5000);
    }
  }
}

void initSDCard(){
   if (!SD.begin()) {
    Serial.println("Card Mount Failed");
    return;
  }
  uint8_t cardType = SD.cardType();
  if(cardType == CARD_NONE){
    Serial.println("No SD card attached");
    return;
  }
  Serial.print("SD Card Type: ");
  if(cardType == CARD_MMC){
    Serial.println("MMC");
  } else if(cardType == CARD_SD){
    Serial.println("SDSC");
  } else if(cardType == CARD_SDHC){
    Serial.println("SDHC");
  } else {
    Serial.println("UNKNOWN");
```

```
  }
  uint64_t cardSize = SD.cardSize() / (1024 * 1024);
  Serial.printf("SD Card Size: %lluMB\n", cardSize);
  if (!SD.exists("/data")) {
    SD.mkdir("/data");
  }
  dataFile = SD.open("/data/data.txt", FILE_WRITE);
  if (!dataFile) {
    Serial.println("Couldn't open file");
  }
}
String epochToDateTime(unsigned long epoch) {
  struct tm timeinfo;
  char timeString[30];
  gmtime_r((const time_t*)&epoch, &timeinfo);
  timeinfo.tm_hour += 7;
  mktime(&timeinfo);
  strftime(timeString, sizeof(timeString), "%Y-%m-%d %H:%M:%S", &timeinfo);
  return String(timeString);
}

void preTransmission()
{
  digitalWrite(RE, 1);
  digitalWrite(DE, 1);
}

void postTransmission()
{
  digitalWrite(RE, 0);
  digitalWrite(DE, 0);
```

```
}

float Cal_carbondioxide() {
  uint16_t readRegisterAddress = 5;
  uint16_t readRegisterValue;
  if (millis() - lastRequestTime >= timeoutPeriod) {
    int8_t resultsensor3 = node3.readHoldingRegisters(readRegisterAddress, 1);
    if (resultsensor3 == node3.ku8MBSuccess) {
      readRegisterValue = node3.getResponseBuffer(0);
    }
    else {}
    lastRequestTime = millis();
  }
  delay(100);
  return float(readRegisterValue);
}

float Cal_oxygen() {
  uint8_t resultsensor1;
  resultsensor1 = node1.readInputRegisters(0, 1);
  return float(node1.getResponseBuffer(0)*0.1);
}

float Cal_hydrogensulfide() {
  uint8_t resultsensor2;
  resultsensor2 = node2.readInputRegisters(0, 1);
  return float(node2.getResponseBuffer(0));
}

float Cal_Airhumidity() {
  uint8_t resultsensor4H;
```

```
  resultsensor4H = node4.readInputRegisters(0, 2);
  return float(node4.getResponseBuffer(0)*0.1);
}

float Cal_Airtemperature() {
  uint8_t resultsensor4T;
  resultsensor4T = node4.readInputRegisters(0, 2);
  return float(node4.getResponseBuffer(1)*0.1);
}

float Cal_Soilmoisture() {
  uint8_t resultsensor5M;
  resultsensor5M = node5.readInputRegisters(0, 4);
  return float(node5.getResponseBuffer(0)*0.1);
}

float Cal_Soiltemperature() {
  uint8_t resultsensor5T;
  resultsensor5T = node5.readInputRegisters(0, 4);
  return float(node5.getResponseBuffer(1)*0.1);
}

float Cal_Soilconductivity() {
  uint8_t resultsensor5C;
  resultsensor5C = node5.readInputRegisters(0, 4);
  return float(node5.getResponseBuffer(2)*0.1);
}

float Cal_SoilPH() {
  uint8_t resultsensor5P;
  resultsensor5P = node5.readInputRegisters(0, 4);
```

```
  return float(node5.getResponseBuffer(3)*0.1);
}

void Stopmotor() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 0);
  Serial.println("Motor stopped");
}
```