

บทที่ 3

วิธีการดำเนินงาน

ในบทนี้จะกล่าวถึงขั้นตอนและวิธีการดำเนินงานจัดทำโครงการตลอดจนการออกแบบโครงสร้างและส่วนต่างๆ ของโครงการ ซึ่งได้รับการศึกษาข้อมูลและทฤษฎีที่เกี่ยวข้องในบทที่ 2 จึงได้นำข้อมูลที่ได้มาศึกษาและวิเคราะห์ออกแบบเพื่อเลือกใช้วัสดุอุปกรณ์ที่จะนำมาทำโครงการ ให้มีความถูกต้องเหมาะสมกับลักษณะงานและดำเนินการสร้างโครงการให้มีประสิทธิภาพและเป็นไปตามวัตถุประสงค์ของโครงการ

3.1 แผนการดำเนินงาน

ในการดำเนินงานจัดทำโครงการโรงเพาะเห็ดอัจฉริยะ (Smart Mushroom Farm) ผู้จัดทำได้วางแผนการดำเนินงานในการจัดทำโครงการดังต่อไปนี้

1. ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง
2. ศึกษา ออกแบบ โครงสร้างของระบบ
3. ศึกษาและออกแบบ เว็บแอปพลิเคชัน วินโดว์แอปพลิเคชัน และ ฐานข้อมูล
4. ศึกษาและออกแบบโรงเรือนสำหรับเพาะเห็ด
5. ทำการทดลองใช้งาน ไมโครคอนโทรลเลอร์ ร่วมกับอุปกรณ์ต่างๆ
6. ทำการสร้าง เว็บแอปพลิเคชัน และ วินโดว์แอปพลิเคชัน
7. ทำการสร้าง โรงเรือนสำหรับเพาะเห็ด และ ติดตั้งอุปกรณ์อุปกรณ์ต่างๆ
8. ทดสอบการทำงาน และ แก้ไขข้อบกพร่อง
9. จัดทำรูปเล่มรายงาน
10. นำเสนอผลงาน

ตารางที่ 3.1 แผนการดำเนินงาน

กิจกรรม	ก.ค. 2563				ส.ค. 2563				ก.ย. 2563				ต.ค. 2563			
1.																
2.																
3.																
4.																
5.																
6.																
7.																
8.																
9.																
10.																

3.2 โครงสร้างของระบบ

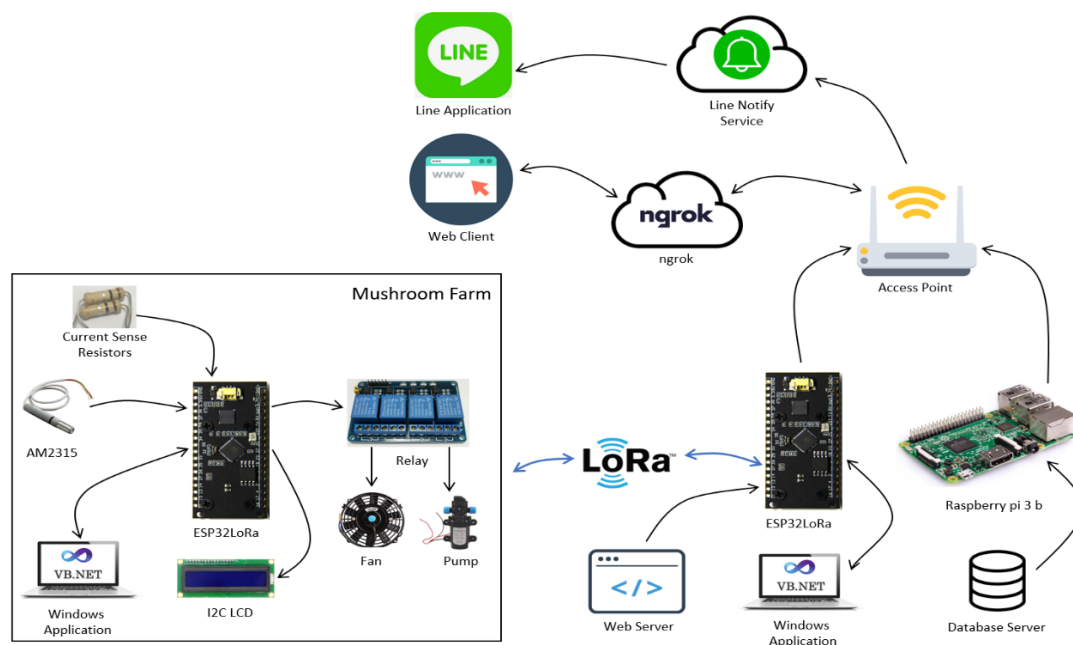
โครงสร้างระบบของโครงการนี้ประกอบด้วยส่วนต่างๆดังรูปที่ 3.1 และ รูปที่ 3.2 ซึ่งจะอธิบายโครงสร้างของระบบตามหัวข้อดังต่อไปนี้ 1. โรงเรือนเพาะเห็ด 2. Lora Communication 3.Windows Application 4. Web Application 5. Line Notify

ส่วนที่ 1 โรงเรือนเพาะเห็ดส่วนนี้จะประกอบไปด้วยอุปกรณ์ฮาร์ดแวร์ที่ใช้ในระบบโดยแต่ละอุปกรณ์มีวัตถุประสงค์การนำมาใช้งานดังต่อไปนี้

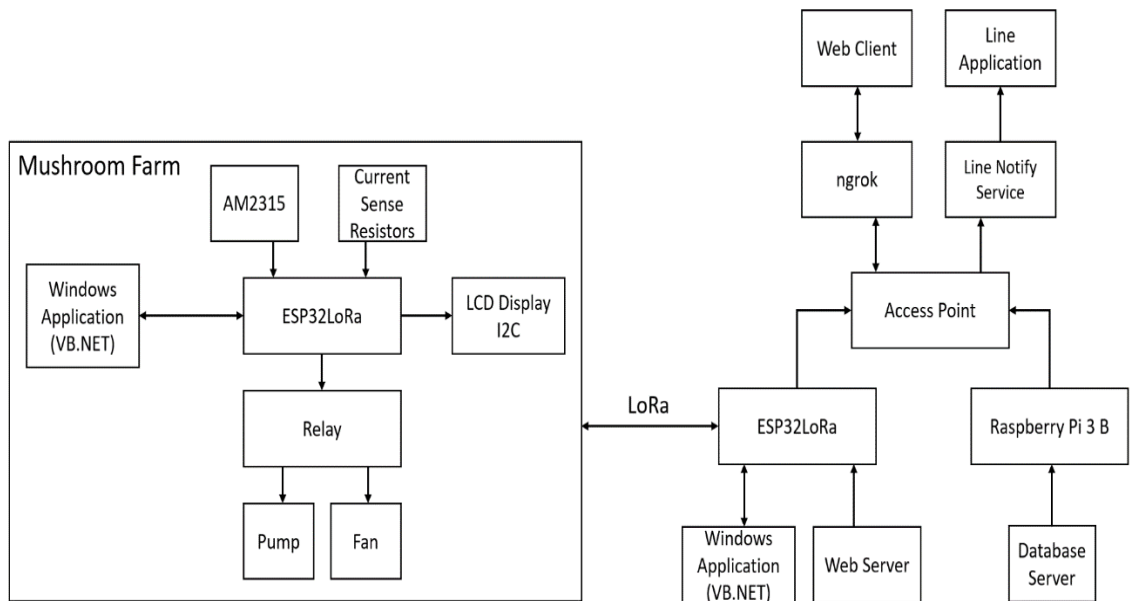
- AM2315 คือเซนเซอร์ที่ใช้วัดอุณหภูมิและความชื้นภายในโรงเรือน สามารถใช้งานกับไมโครคอนโทรลเลอร์ ESP32LoRa ผ่านทาง I²C bus
- ESP32LoRa คือไมโครคอนโทรลเลอร์ที่ใช้ควบคุมภายในระบบ เช่น การอ่านค่าจากเซนเซอร์ การควบคุมการทำงานของอุปกรณ์ เป็นต้น
- Current Sense Resistors คือตัวต้านทานที่มีค่าความต้านทานต่ำมากใช้สำหรับการสอบการไหลของกระแส เพื่อตรวจสอบสถานะการทำงานของอุปกรณ์

- I2C LCD คือจอที่ใช้แสดงผลข้อมูลต่างๆที่หน้าโรงเรือนเพาะเห็ด เช่น ค่าอุณหภูมิ ค่าความชื้น โหมดการทำงานของระบบ สถานะการทำงานของอุปกรณ์ เป็นต้น สามารถใช้งานกับไมโครคอนโทรลเลอร์ ESP32LoRa ผ่านทาง I2C bus
- Relay คืออุปกรณ์ที่ใช้ในการควบคุมการทำงานของพัดลมและปั้ม
- Fan พัดลมระบายอากาศใช้ในการระบายอากาศภายในโรงเรือน
- Pump ปั้มใช้ในการพ่นหมอกเพื่อเพิ่มความชื้นภายในโรงเรือน
-

ส่วนที่ 2 Lora Communication ในโครงงานนี้ผู้จัดทำได้นำเทคโนโลยีการสื่อสาร Lora มาประยุกต์ใช้เพื่อเป็นตัวกลางการสื่อสารระหว่างโรงเรือนเพาะเห็ด (Mushroom Node) และพื้นที่ที่มีสัญญาณอินเทอร์เน็ต (STA Node) เพื่อแก้ปัญหาที่ตั้งของโรงเรือนไม่มีสัญญาณอินเทอร์เน็ต โดยเป็นการสื่อสารแบบ Point to Point หรือ Node-to-Node Communication



รูปที่ 3.1 แสดงภาพ System Diagram



รูปที่ 3.2 แสดงภาพ Block Diagram

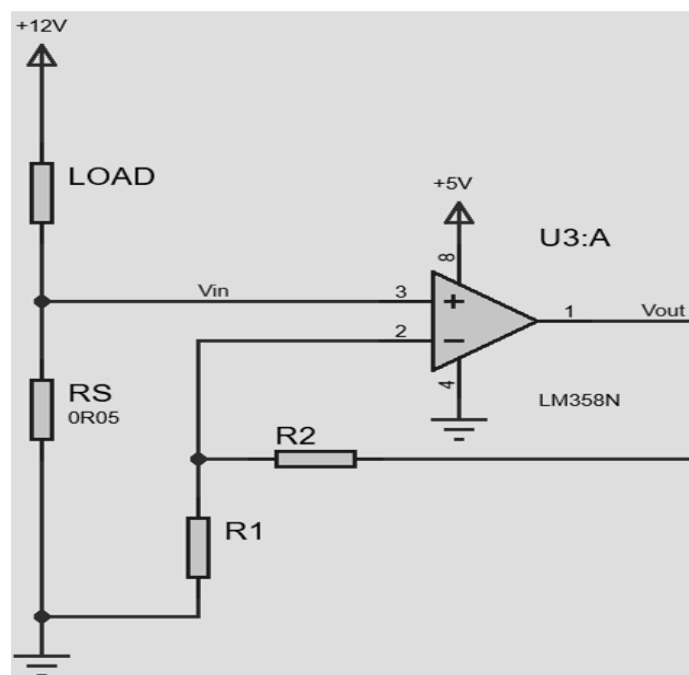
ส่วนที่ 3. Windows Application สามารถดูค่าสถานะและควบคุมระบบภายในโรงเรือนได้ผ่านทาง Serial Port ระหว่างคอมพิวเตอร์และไมโครคอนโทรลเลอร์ ใช้ภาษา Visual Basic ของ .NET Core ในการสร้าง Windows Application สามารถใช้งานได้กับไมโครคอนโทรลเลอร์ที่อยู่โรงเรือนเพาะเห็ด (Mushroom Node) และไมโครคอนโทรลเลอร์ที่ทำหน้าที่เป็น Web Server (STA Node) สามารถนำวินโดวแอปพลิเคชันไปติดตั้งที่คอมพิวเตอร์เครื่องอื่นที่เป็น ระบบปฏิบัติการวินโดวแล้วนำแอปพลิเคชันมาใช้งานได้ทันที

ส่วนที่ 4 Web Application ใช้ภาษา HTML CSS Javascript และ PHP ในการสร้าง Web Application และใช้แอปพลิเคชัน ngrok ที่ติดตั้งบน Raspberry Pi ทำ Port forwarding ของ Web Server เพื่อให้สามารถใช้งานเว็บแอปพลิเคชันได้จากทุกที่ที่สามารถใช้งานอินเทอร์เน็ตได้ Web Application ส่วนที่แสดงค่าสถานะต่างๆและควบคุมการทำงานของระบบจะถูกเก็บอยู่ที่ ESP32LoRa และ Web Application ส่วนที่เกี่ยวกับฐานข้อมูล เช่น บันทึกข้อมูลลงในฐานข้อมูล นำข้อมูลในฐานข้อมูลมาแสดงบน Web Application เป็นต้น จะถูกเก็บอยู่ที่ Raspberry Pi 3 Model B เพราะใช้ Raspberry Pi เป็นฐานข้อมูล

ส่วนที่ 5 Line Notify เป็นบริการของ LINE ที่ให้สามารถส่งข้อความ การแจ้งเตือนต่าง ๆ ไปยังบัญชีหรือกลุ่มต่างๆ ผ่านทาง API ในโครงการนี้ผู้จัดทำได้นำ Line Notify มาใช้ในการแจ้งเตือนสถานะต่างๆไปยัง Line Application เช่น การแจ้งเตือนเมื่อ Web Server เริ่มทำงานระบบจะส่ง URL ที่ได้จากโปรแกรม Ngrok เพื่อให้สะดวกต่อการเข้าใช้งาน Web Application การแจ้งเตือนเมื่ออุปกรณ์เปลี่ยนสถานะการทำงาน การแจ้งเมื่อมีการบันทึกข้อมูลลงในฐานข้อมูล เป็นต้น

3.3 การออกแบบวงจรตัวต้านทานตรวจสอบกระแส

วงจรตัวต้านทานตรวจสอบกระแส มีวัตถุประสงค์เพื่อใช้ในการตรวจสอบสถานะการทำงานของ พัดลมระบายอากาศ บั้มพ่นหมอก ว่าทำงานจริงตามที่ควบคุมหรือไม่ โดยแสดงการออกแบบวงจรรูปด้านล่าง โดยมีหลักการทำงานคือนำตัวต้านทานที่มีค่าความต้านทาน 0.05 โอห์ม มาต่ออนุกรมกับ LOAD จากนั้นนำค่าแรงดันตกคร่อมตัวต้านทานมาขยายแรงดันโดยใช้วงจรขยายวงจรขยายแบบไม่กลับเฟส (Non-inverting Amplifier) เพื่อให้ไมโครคอนโทรลเลอร์สามารถอ่านค่าแรงดันได้



รูปที่ 3.ป แสดงภาพวงจรตัวต้านทานตรวจสอบกระแส

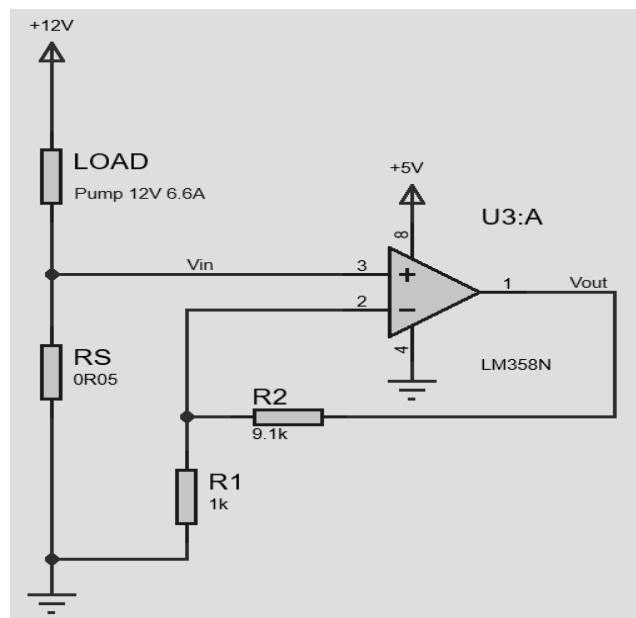
จากรูปที่ 3.ป แสดงภาพของแสดงภาพวงจรตัวต้านทานตรวจสอบกระแส โดยจะประกอบด้วย LOAD คือ พัดลมระบายอากาศ หรือ ปั๊มพ่นหมอก RS คือ ตัวต้านทานตรวจสอบกระแส (Current Sense Resistors) และส่วนของวงจรขยายแบบไม่กลับเฟส (Non-inverting amplifier) ใช้เพื่อขยายแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส

3.3.1 การคำนวณแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับพัดลมระบายอากาศ

ในโครงการนี้ผู้จัดทำใช้พัดลมระบายอากาศขนาด 12 นิ้ว 12V 6.6A ในการควบคุมอุณหภูมิภายในโรงเรือนเพาะเห็ด ดังนั้นสามารถคำนวณแรงดันเอาต์พุตของวงจรตัวต้านทานตรวจสอบกระแส ได้จากสมการ

$$V_{out} = (1 + \frac{R_2}{R_1})V_{in}$$

จากสมการด้านบน V_{in} คือแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส (Current Sense Resistors) ดังนั้นจึงต้องทราบค่า V_{in} ก่อน



รูปที่ 3.ป แสดงภาพวงจรตัวต้านทานตรวจสอบกระแสเมื่อใช้งานกับพัดลมระบายอากาศ

จากรูปที่ 3.ป เพื่อให้ไมโครคอนโทรลเลอร์สามารถแรงดันเอาต์พุตของวงจรได้ โดยสามารถหา อัตราการขยายของวงจรได้จากสมการ

จากสูตร

$$\frac{V_{out}}{V_{in}} = A_{(v)} = 1 + \frac{R_2}{R_1}$$

จะได้

$$\text{อัตราการขยาย } (A_{(v)}) = 1 + \frac{9.1}{1}$$

ดังนั้น

$$\text{อัตราการขยาย } (A_{(v)}) = 10.1 \text{ เท่า}$$

สามารถหาแรงดันตกคร่อมต้านทานตรวจสอบกระแส V_{in} หรือ V_{rs} ได้จากกฎของโอห์มตั้ง
สมการ

$$I = \frac{V}{R}$$

ดังนั้นหาแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแสจากรูปที่ 3.ป กำหนดให้ กระแส (I) =
6.6A ความต้านทานของต้านทานตรวจสอบกระแส (Rs) = 0.05 Ω

จากสูตร

$$I = \frac{V}{R}$$

จะได้

$$V_{in} = I R_s$$

แทนค่า

$$V_{in} = (6.6)(0.05)$$

ดังนั้นแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส (Vin) มีค่าเท่ากับ 0.33V เท่ากับ V_{in}
ดังนั้นสามารถคำนวณแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับพัดลมระบายอากาศได้จากสมการ

$$V_{out} = (1 + \frac{R_2}{R_1})V_{in}$$

แทนค่า

$$V_{out} = (1 + \frac{9.1}{1}) 0.33$$

ดังนั้นแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับพัดลมระบายอากาศ (V_{out}) เท่ากับ 3.33V สามารถคำนวณกำลังวัตต์ของตัวต้านทานตรวจสอบกระแส (Prs)

จากสูตร

$$P = VI$$

จะได้

$$Prs = V_{in} I$$

แทนค่า

$$Prs = (0.33)(6.6)$$

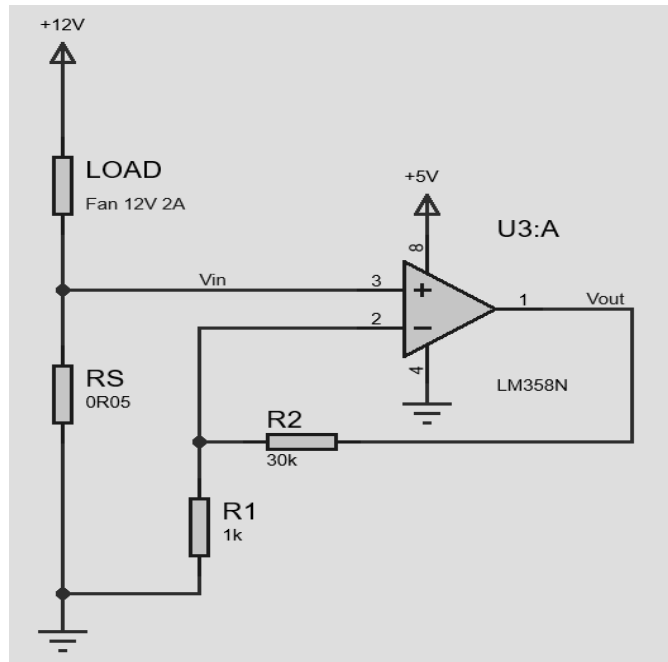
ดังนั้นกำลังวัตต์ของตัวต้านทานตรวจสอบกระแส (Prs) เท่ากับ 2.178W

3.3.2 การคำนวณแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับปั๊มพ่นหมอก

ในโครงการนี้ผู้จัดทำใช้ปั๊มพ่นหมอกขนาดแรงดัน 4.8 bar / 70 PSI 12V 2A ในการควบคุมความชื้นภายในโรงเรือนเพาะเห็ด ดังนั้นสามารถคำนวณแรงดันเอาต์พุตของวงจรตัวต้านทานตรวจสอบกระแส ได้จากสมการ

$$V_{out} = (1 + \frac{R_2}{R_1})V_{in}$$

จากสมการด้านบน V_{in} คือแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส (Current Sense Resistors) ดังนั้นจึงต้องทราบค่า V_{in} ก่อน



รูปที่ 3.ป แสดงภาพวงจรตัวต้านทานตรวจสอบกระแสเมื่อใช้งานกับปั้มพ่นหมอก

จากรูปที่ 3.ป เพื่อให้ไมโครคอนโทรลเลอร์สามารถแรงดันเอาต์พุตของวงจรได้ โดยสามารถหาอัตราขยายของวงจรได้จากสมการ

จากสูตร

$$\frac{V_{out}}{V_{in}} = A_{(v)} = 1 + \frac{R_2}{R_1}$$

จะได้

$$\text{อัตราขยาย } (A_{(v)}) = 1 + \frac{30}{1}$$

ดังนั้น

$$\text{อัตราขยาย } (A_{(v)}) = 31 \text{ เท่า}$$

สามารถหาแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส V_{in} หรือ V_{rs} ได้จากกฎของโอห์มดั่ง
สมการ

$$I = \frac{V}{R}$$

ดังนั้นหาแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแสจากรูปที่ 3.ป กำหนดให้ กระแส (I) =
 $2A$ ความต้านทานของตัวต้านทานตรวจสอบกระแส (R_s) = 0.05Ω

จากสูตร

$$I = \frac{V}{R}$$

จะได้

$$V_{in} = IR_s$$

แทนค่า

$$V_{in} = (2)(0.05)$$

ดังนั้นแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส (V_{in}) มีค่าเท่ากับ $0.1V$ เท่ากับ V_{in}
ดังนั้นสามารถคำนวณแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับปั๊มพ่นหมอกได้จากสมการ

$$V_{out} = \left(1 + \frac{R_2}{R_1}\right)V_{in}$$

แทนค่า

$$V_{out} = \left(1 + \frac{30}{1}\right)0.1$$

ดังนั้นแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับปั๊มพ่นหมอก (V_{out}) เท่ากับ $3.1V$ สามารถ
คำนวณกำลังวัตต์ของตัวต้านทานตรวจสอบกระแส (P_{rs})

จากสูตร

$$P = VI$$

จะได้

$$Prs = Vrs I$$

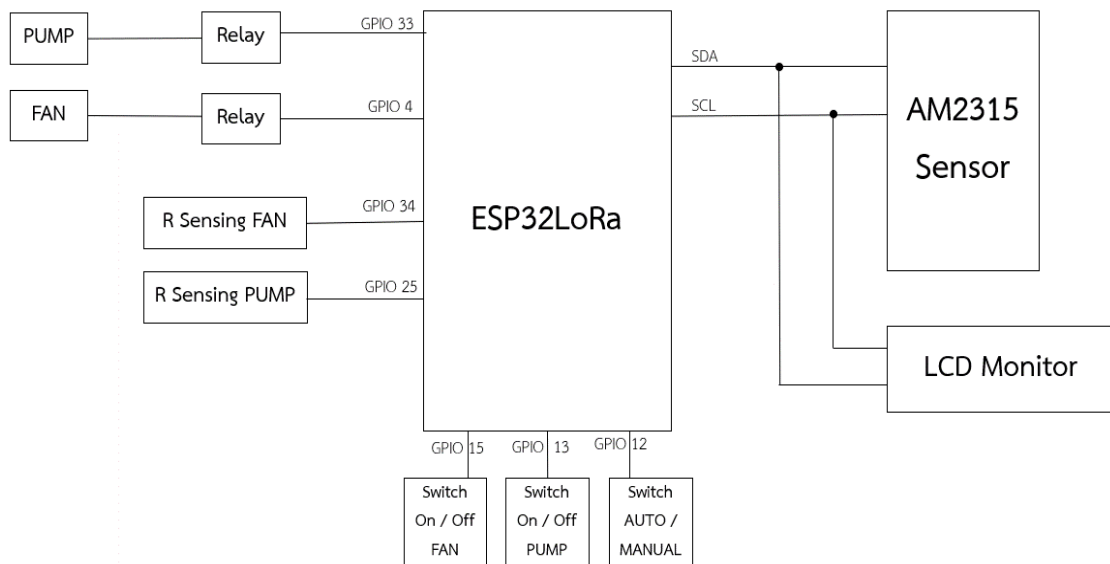
แทนค่า

$$Prs = (0.1)(2)$$

ดังนั้นกำลังวัตต์ของตัวต้านทานตรวจสอบกระแส (Prs) เท่ากับ 0.2W

3.4 การออกแบบฮาร์ดแวร์ระบบควบคุมอุณหภูมิและความชื้นภายในโรงเรือนเพาะเห็ด

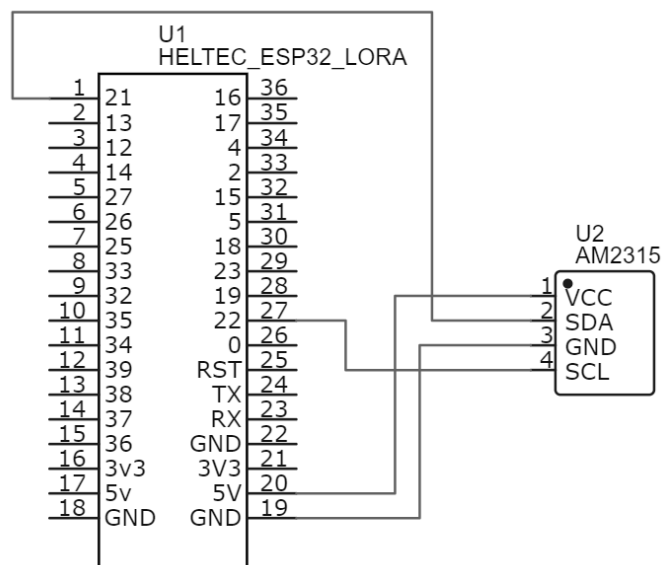
ในการออกแบบฮาร์ดแวร์ระบบควบคุมอุณหภูมิและความชื้นภายในโรงเรือนเพาะเห็ด ผู้จัดทำได้ใช้ AM2315 เป็นเซนเซอร์วัดอุณหภูมิและความชื้นภายในโรงเรือน และใช้พัดลมในการระบายอากาศ ใช้ปั๊มพ่นหมอกในการให้ความชื้นภายในโรงเรือน ใช้ตัวต้านทานตรวจสอบกระแส (Current Sense Resistors) สำหรับตรวจสอบสถานะการทำงานของพัดลมและปั๊มพ่นหมอก ใช้จอแอลซีดี (LCD) ในการแสดงผลค่าต่างๆ ภายในระบบ ดังรูปที่แสดงด้านล่าง



รูปที่ 3.ป แสดงภาพ Box Diagram ของฮาร์ดแวร์ระบบควบคุมอุณหภูมิและความชื้น

3.4.1 การอ่านค่าอุณหภูมิและความชื้นจากเซนเซอร์ AM2315

AM2315 คือเซนเซอร์วัดค่าอุณหภูมิและความชื้นคุณภาพสูงที่มีเอาต์พุตเป็นสัญญาณดิจิทัล โดยมีการสอบเทียบสัญญาณดิจิทัลเอาต์พุตแล้ว ใช้โมดูลตรวจจับอุณหภูมิและความชื้นคุณภาพสูงเพื่อให้แน่ใจว่ามีการวัดค่าที่แม่นยำและมีความทนทาน เซนเซอร์ประกอบด้วยตัววัดความชื้นประเภทความจุไฟฟ้า อุปกรณ์วัดอุณหภูมิที่มีความแม่นยำสูง และเชื่อมต่อกับไมโครคอนโทรลเลอร์ 8 บิตประสิทธิภาพสูง AM2315 สื่อสารด้วย I2C ผู้ใช้สามารถเชื่อมต่อ I2C bus ได้โดยตรง



รูปที่ 3.ป แสดงภาพ Schematic การเชื่อมต่อระหว่าง ESP32LoRa และ AM2315

ตารางที่ 3.ป การต่อขาระหว่าง ESP32LoRa และ AM2315

ESP32LoRa	AM2315
5V	VCC
GND	GND
SDA(GPIO21)	SDA
SCL(GPIO22)	SCL

```

#include <Wire.h>
#include <Adafruit_AM2315.h>

// Connect RED of the AM2315 sensor to 5.0V
// Connect BLACK to Ground
// Connect WHITE to i2c clock
// Connect YELLOW to i2c data

Adafruit_AM2315 am2315;

void setup() {
  Serial.begin(115200);
  Wire.begin (21, 22); // sda= GPIO_21 /scl= GPIO_22

  // Wake up the sensor
  Wire.beginTransmission(AM2315_I2CADDR);
  delay(2);
  Wire.endTransmission();

  while (!Serial) {
    delay(10);
  }
  Serial.println("AM2315 Test!");

  if (! am2315.begin()) {
    Serial.println("Sensor not found, check wiring & pullups!");
    while (1);
  }
}

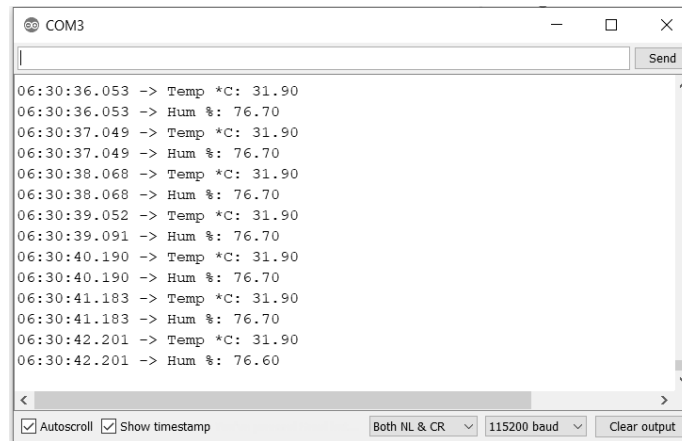
void loop() {
  float temperature, humidity;
  am2315.readTemperatureAndHumidity(&temperature, &humidity);
  // if (! am2315.readTemperatureAndHumidity(&temperature, &humidity)) {
  //   Serial.println("Failed to read data from AM2315");
  //   return;
  // }
  Serial.print("Temp *C: "); Serial.println(temperature);
  Serial.print("Hum %: "); Serial.println(humidity);

  delay(1000);
}

```

รูปที่ 3.ป แสดงภาพโปรแกรมที่ใช้อ่านค่าจากเซนเซอร์ AM2315

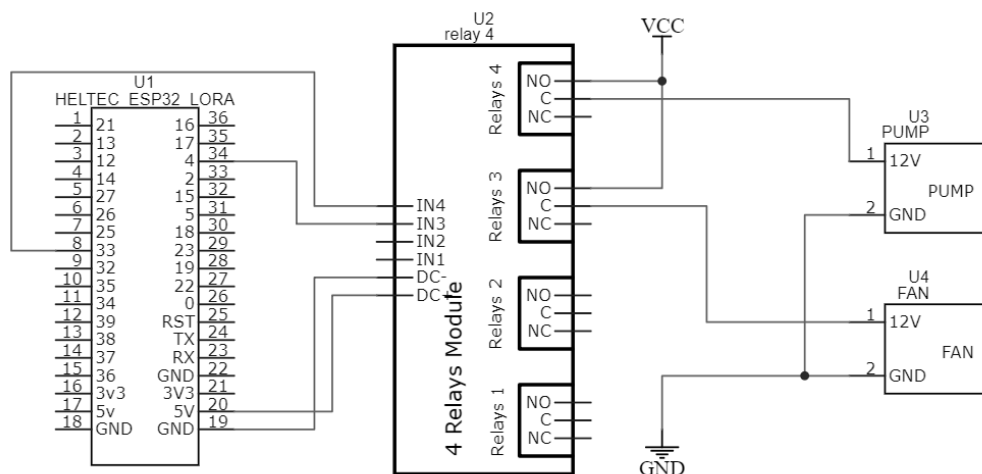
จากรูปที่ 3.ป เป็นโปรแกรมภาษา C++ ที่ใช้ในการอ่านค่าจากเซนเซอร์ AM2315 โดยมีการเรียกใช้ไลบรารี (Library) Wire.h และ Adafruit_AM2315.h ต่อมาส่วนที่สำคัญที่สุดคือส่วนของการปลุกการทำงานของเซนเซอร์ดังรูปที่แสดงด้านบน หากไม่ปลุกการทำงานของเซนเซอร์เมื่อไม่มีการใช้งานเซนเซอร์เป็นเวลานานจะส่งผลให้ไม่สามารถใช้งานได้ ผลลัพธ์ที่อ่านค่าได้จากเซนเซอร์แสดงดังรูปที่ 3.ป



รูปที่ 3.ป แสดงภาพผลลัพธ์ที่อ่านได้จากเซนเซอร์ AM2315

3.4.2 การควบคุมพัดลมและปั้มน้ำ

การควบคุมพัดลมและปั้มน้ำผู้จัดทำได้ใช้ 4-Channel Relay Module ในการควบคุมการทำงานเพราะพัดลมและปั้มน้ำใช้แรงดันไฟเลี้ยง 12V 4-Channel Relay Module เป็น Relay แบบ Active LOW จะทำงานเมื่อสั่งขา GPIO ของ ไมโครคอนโทรลเลอร์ ESP32LoRa เป็น LOW



รูปที่ 3.ป แสดงภาพ Schematic ของการควบคุมพัดลมและปั้มน้ำ

ตารางที่ 3.ป การต่อขาระหว่าง ESP32LoRa และ 4-Channel Relay Module

ESP32LoRa	4-Channel Relay Module
5V	DC +
GND	DC -
GPIO33	IN4
GPIO4	IN3

```
//-----Active low relays
if (pumpState == true) {
    digitalWrite(pump, false);
} else {
    digitalWrite(pump, true);
}
if (fanState == true) {
    digitalWrite(fan, false);
} else {
    digitalWrite(fan, true);
}
//-----
```

รูปที่ 3.ป แสดงภาพโปรแกรมส่วนที่ใช้ควบคุมการทำงานของ Relay

จากรูปที่ 3.ป แสดงภาพโปรแกรมส่วนที่ใช้ควบคุมการทำงานของ Relay มีรายละเอียดคือ ตัวแปร pumpState และ ตัวแปร fanState เป็นตัวแปรประเภท Boolean ค่าสามารถมีค่าที่เป็นไปได้คือ 0 และ 1 หรือ true และ false โดยผู้จัดทำได้ออกแบบระบบให้พัดลมและปั๊มพ่นหมอกทำงานเมื่อตัวแปร pumpState และ ตัวแปร fanState มีค่าเท่ากับ true แต่ Relay ที่นำมาใช้จะทำงานเมื่อตัวแปร pumpState และ ตัวแปร fanState มีค่าเท่ากับ false ผู้จัดทำจึงได้เขียนโปรแกรมเพื่อให้ Relay และระบบสามารถทำงานร่วมกันได้โดยไม่ต้องแก้ไขทั้งระบบ

3.4.2.1 การทำงานของปั๊มพ่นหมอกและพัดลมในโหมดการทำงานแบบอัตโนมัติ (AUTO)

การทำงานของปั๊มพ่นหมอกและพัดลมในโหมดการทำงานแบบอัตโนมัติ (AUTO) จะพิจารณาตัวแปร 4 ตัวแปรคือ ค่าอุณหภูมิต่ำสุด ค่าอุณหภูมิสูงสุด ค่าความชื้นต่ำสุด ค่าความชื้นสูงสุด เพื่อรักษาระดับอุณหภูมิและความชื้นให้อยู่ในช่วงที่กำหนด สมมติกำหนดให้ ค่าอุณหภูมิต่ำสุดเท่ากับ 25°C ค่าอุณหภูมิสูงสุดเท่ากับ 32°C ค่าความชื้นต่ำสุดเท่ากับ 80% ค่าความชื้นสูงสุดเท่ากับ 85%

ตารางที่ 3.ป แสดงการทำงานของปั๊มพ่นหมอกและพัดลมในโหมดการทำงานแบบอัตโนมัติ (AUTO)

อุณหภูมิ(°C)	ความชื้น(%)	ปั๊มพ่นหมอก	พัดลม
< 25	< 80	ON	OFF
< 25	80-85	OFF	OFF
< 25	> 85	OFF	ON
25-32	< 80	ON	OFF
25-32	80-85	OFF	OFF
25-32	> 85	OFF	ON
> 32	< 80	ON	ON
> 32	80-85	OFF	ON
> 32	> 85	OFF	ON


```

//check mode-----
if (ctrlMode == false) { // auto mode
    //temperature and humidity control
    if (temp < set_temp_min && humi < set_humi_min) {
        pumpState = true; fanState = false;
    }
    if (temp < set_temp_min && (humi >= set_humi_min && humi <= set_humi_max)) {
        pumpState = false; fanState = false;
    }
    if (temp < set_temp_min && humi > set_humi_max) {
        pumpState = false; fanState = true;
    }
    if ((temp >= set_temp_min && temp <= set_temp_max) && humi < set_humi_min) {
        pumpState = true; fanState = false;
    }
    if ((temp >= set_temp_min && temp <= set_temp_max) && (humi >= set_humi_min && humi <= set_humi_max)) {
        pumpState = false; fanState = false;
    }
    if ((temp >= set_temp_min && temp <= set_temp_max) && humi > set_humi_max) {
        pumpState = false; fanState = true;
    }
    if (temp > set_temp_max && humi < set_humi_min) {
        pumpState = true; fanState = true;
    }
    if (temp > set_temp_max && (humi >= set_humi_min && humi <= set_humi_max)) {
        pumpState = false; fanState = true;
    }
    if (temp > set_temp_max && humi > set_humi_max) {
        pumpState = false; fanState = true;
    }
}
}

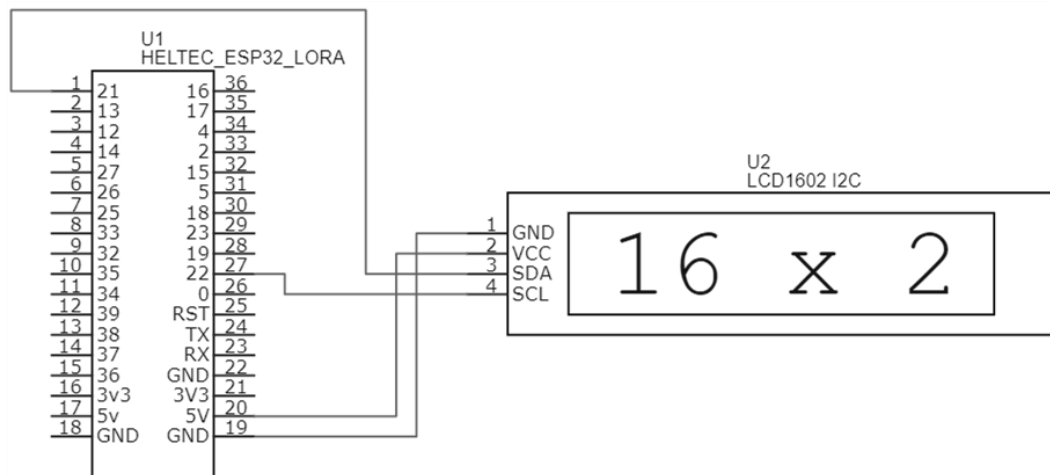
```

รูปที่ 3.ป แสดงภาพโปรแกรมที่ใช้ควบคุมการทำงานของปั้มน้ำหมอกและพัดลม

จากรูปที่ 3.ป แสดงภาพโปรแกรมภาษา C++ ที่ใช้ควบคุมการทำงานของปั้มน้ำหมอกและพัดลมในโหมดการทำงานแบบอัตโนมัติ (AUTO) ตัวแปร ctrlMode คือตัวแปรโหมดการทำงานของระบบ ตัวแปร temp คือตัวแปรค่าอุณหภูมิ และตัวแปร humi คือตัวแปรค่าความชื้นที่อ่านได้จากเซนเซอร์ AM2315 ตัวแปร set_temp_min และ set_temp_max คือค่าอุณหภูมิต่ำสุดและสูงสุด ตัวแปร set_humi_min และ set_humi_max คือค่าความชื้นต่ำสุดและสูงสุด ตัวแปร pumpState และ fanState ค่าตัวแปรที่ใช้ในการควบคุมการทำงานของปั้มน้ำหมอกและพัดลม

3.4.3 การใช้จอแอลซี (LCD) แสดงผล

ในโครงการนี้ผู้จัดทำได้ใช้จอแอลซีดี (I2C LCD) ขนาด 16 x 2 ในการแสดงผลข้อมูลต่างๆในระบบ เช่น ค่าอุณหภูมิ ค่าความชื้น ค่าสถานะการทำงานของอุปกรณ์ โหมดการทำงานของระบบ เป็นต้น



รูปที่ 3.ป แสดงภาพ Schematic การเชื่อมต่อระหว่าง ESP32LoRa และ จอแอลซีดี (LCD)

ตารางที่ 3.ป การต่อขาระหว่าง ESP32LoRa และ จอแอลซีดี (LCD)

ESP32LoRa	LCD
5V	VCC
GND	GND
SDA (GPIO21)	SDA
SCL (GPIO22)	SCL

```

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
    // initialize the LCD
    lcd.begin();
    lcd.setCursor(0, 0);
    lcd.print("I2C LCD TEST..");
    lcd.setCursor(0, 1);
    lcd.print("Hello World");
}

void loop()
{
    // Do nothing here...
}

```

รูปที่ 3.ป แสดงภาพโปรแกรมทดสอบการทำงานของจอแอลซีดี (LCD)

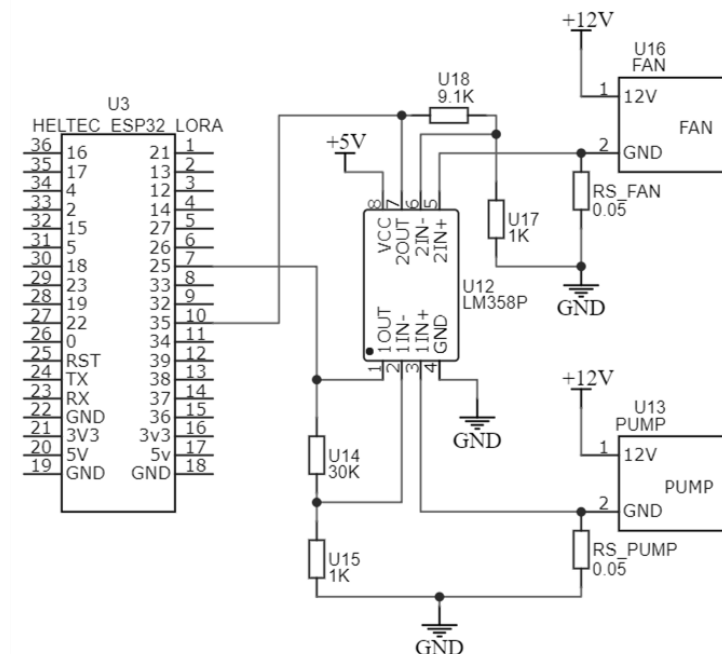
จากรูปที่ 3.ป เป็นโปรแกรมภาษา C++ ที่ใช้ในการทดสอบการแสดงผลของจอแอลซีดีขนาด 16 x 2 การเชื่อมต่อแบบ I2C โดยกำหนด I2C Address ของจอแอลซีดี (LCD) เป็น Address 0x27 จากนั้นใช้ฟังก์ชัน lcd.begin() เพื่อเริ่มต้นการทำงานของ LCD ใช้ฟังก์ชัน lcd.setCursor() ในการกำหนดตำแหน่งเริ่มต้นของตัวอักษร และใช้ฟังก์ชัน lcd.print() ในการแสดงตัวอักษร รูปที่แสดงด้านล่างแสดงผลลัพธ์ที่ได้จากโปรแกรมทดสอบการทำงานของจอแอลซีดี (LCD) แบบ I2C



รูปที่ 3.ป แสดงภาพผลลัพธ์ที่ได้จากโปรแกรมทดสอบการทำงานของจอแอลซีดี (LCD)

3.4.4 การอ่านค่าแรงดันเอาต์พุตจากวงจรตัวต้านทานตรวจสอบกระแส

ในโครงการนี้ผู้จัดทำได้นำตัวต้านทานตรวจสอบกระแสมาประยุกต์ใช้ เพื่อใช้ในการตรวจสอบสถานะการทำงานของอุปกรณ์ เพื่อให้แน่ใจอุปกรณ์ทำงานตามที่ควบคุมหรือไม่ จากนั้นผู้จัดทำได้ใช้ ADC (Analog to Digital Converter) ของ ESP32LoRa ในการอ่านค่าแรงดันเอาต์พุตจากวงจรตัวต้านทานตรวจสอบกระแสดังรูปที่แสดงด้านล่าง



รูปที่ 3.ป แสดงภาพ Schematic การเชื่อมต่อระหว่าง ESP32LoRa และวงจรตัวต้านทานตรวจสอบกระแส

ตารางที่ 3.ป การต่อขาระหว่าง ESP32LoRa และวงจรตัวต้านทานตรวจสอบกระแส

ESP32LoRa	วงจรตัวต้านทานตรวจสอบกระแส
GPIO25	OUT (LM741) PUMP
GPIO35	OUT (LM741) FAN

```

#define R_sensing_pump 25
#define R_sensing_fan 35

int adc_pump_value, adc_fan_value = 0;
float Rs_pump_voltage, Rs_fan_voltage = 0;

void setup() {
  Serial.begin(115200);
  analogReadResolution(12);
}

void loop() {
  // Reading adc values
  adc_pump_value = analogRead(R_sensing_pump);
  adc_fan_value = analogRead(R_sensing_fan);
  Serial.print(adc_pump_value); Serial.print(" "); Serial.println(adc_fan_value);

  //calculate to voltage
  Rs_pump_voltage = (adc_pump_value * 3.3 / 4095);
  Rs_fan_voltage = (adc_fan_value * 3.3 / 4095);
  Serial.print(Rs_pump_voltage); Serial.print(" "); Serial.println(Rs_fan_voltage);

  delay(300);
}

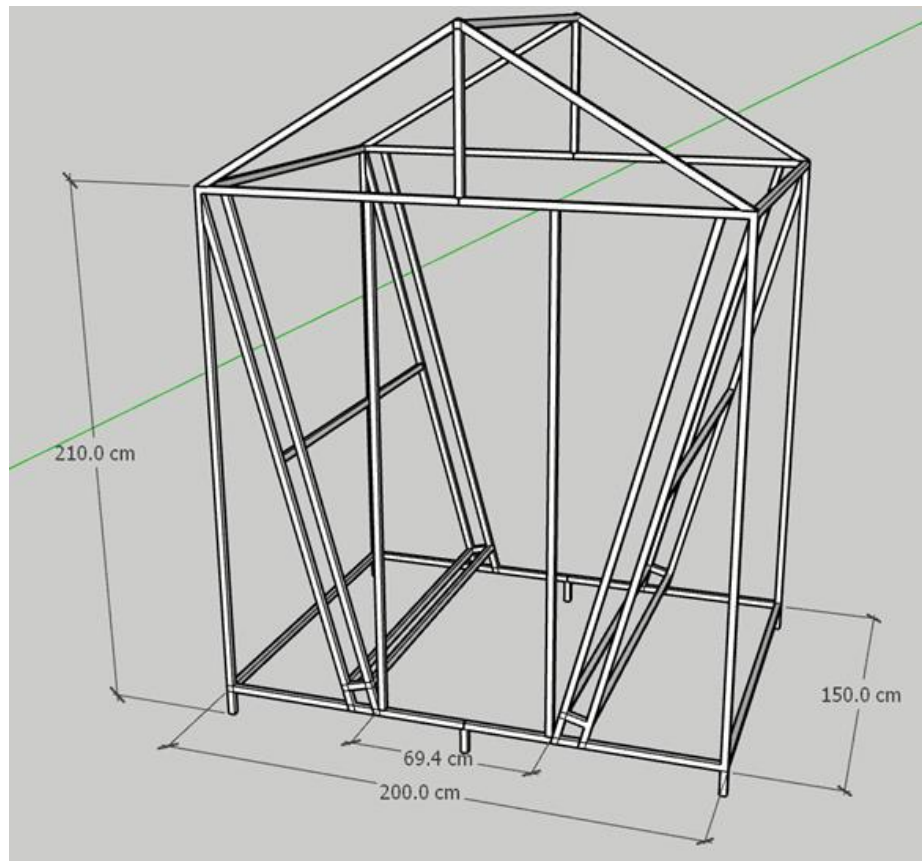
```

รูปที่ 3.ป แสดงภาพโปรแกรมทดสอบการอ่านค่าแรงดันเอาต์พุตจากวงจรตัวต้านทานตรวจสอบ
กระแส

จากรูปที่ 3.ป แสดงภาพโปรแกรมที่ใช้ในการอ่านค่าแรงดันเอาต์พุตจากวงจรตัวต้านทาน
ตรวจสอบกระแส โดยใช้ GPIO25 และ GPIO35 ของ ESP32LoRa ในการอ่านค่าแรงดันที่ได้จากเอาต์พุต
ของวงจร ใช้ ADC ความละเอียดในการอ่าน 12-bit จากนั้นนำค่าที่ได้มาคำนวณเป็นแรงดันที่สามารถอ่านได้
จากเอาต์พุตของวงจร

3.5 การออกแบบและสร้างโรงเรือนสำหรับเพาะเห็ด

ในโครงการนี้ผู้จัดทำได้ใช้โปรแกรม Google Sketchup ในการออกแบบโรงเรือนสำหรับเพาะเห็ดขนาด 200x150x210 เซนติเมตร ตามรูปที่ 3.3 โดยได้ออกแบบให้เป็นโรงเรือนแบบปิด ควบคุมอุณหภูมิภายในโรงเรือนด้วยพัดลมระบายอากาศ และควบคุมความชื้นโดยใช้ปั้มพ่นหมอก โครงสร้างของโรงเรือนเพาะเห็ดเป็นเหล็กกล่องขนาด 1x1 นิ้ว ดังรูปที่ 3.4 เพื่อโรงเรือนมีความแข็งแรงและสามารถใช้งานได้ในระยะยาว เมื่อทำการเชื่อมโครงสร้างโรงเรือนเสร็จแล้ว ขั้นตอนต่อไปคือการพ่นสีเพื่อป้องกันการเกิดสนิม ดังรูปที่ 3.5



รูปที่ 3.3 แสดงภาพการออกแบบโรงเรือนที่ใช้เพาะเห็ด



รูปที่ 3.4 แสดงภาพโครงสร้างของโรงเรือนเพาะเห็ด



รูปที่ 3.5 แสดงภาพขั้นตอนการพ่นสีโรงเรือนเพาะเห็ด



รูปที่ 3.ป แสดงภาพขั้นตอนการติดลวดตาข่ายเพื่อป้องกันก้อนเห็ดหล่น



รูปที่ 3.ป แสดงภาพขั้นตอนการคลุมโรงเรือนเพาะเห็ดเพื่อเก็บรักษาความชื้น



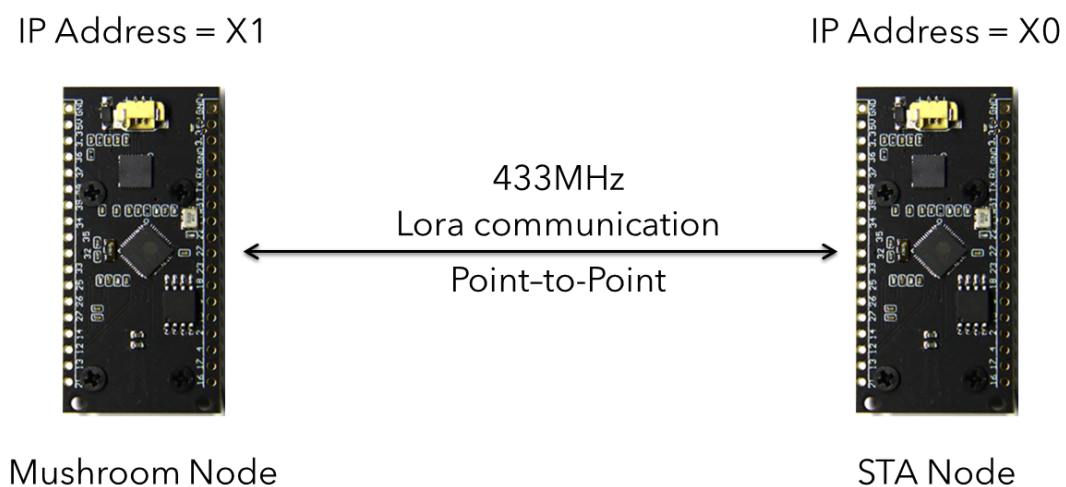
รูปที่ 3.ป แสดงภาพขั้นตอนการติดตั้งพัดลมระบายอากาศ



รูปที่ 3.ป แสดงภาพขั้นตอนการติดตั้งหัวพ่นหมอก

3.6 การออกแบบการสื่อสารผ่าน Lora

ในโครงงานนี้ผู้จัดทำได้นำเทคโนโลยีการสื่อสาร Lora มาประยุกต์ใช้เพื่อเป็นตัวกลางการสื่อสารระหว่างโรงเรือนเพาะเห็ด (Mushroom Node) และพื้นที่ที่มีสัญญาณอินเทอร์เน็ต (STA Node) เพื่อแก้ปัญหาที่ตั้งของโรงเรือนไม่มีสัญญาณอินเทอร์เน็ต โดยเป็นการสื่อสารแบบ Point to Point หรือ Node-to-Node Communication ดังรูปด้านล่าง

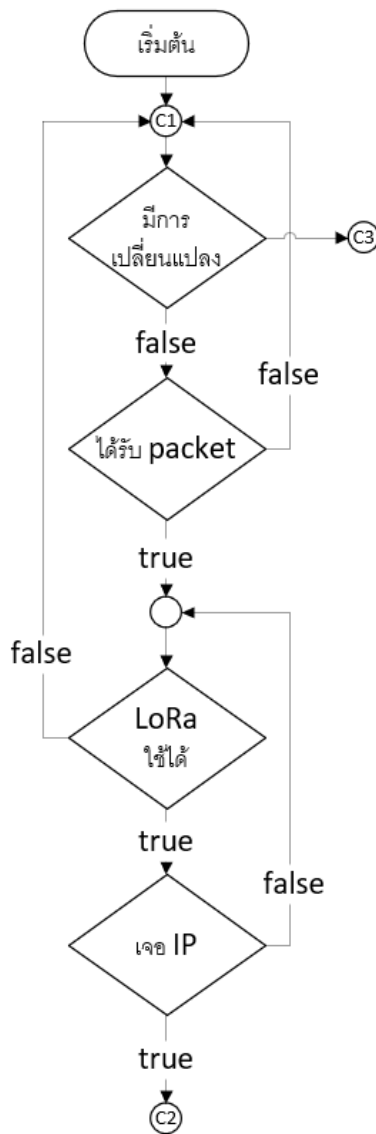


รูปที่ 3.ป แสดงภาพการสื่อสารแบบ Node-to-Node

จากรูปที่ 3.ป Node แต่ละ Node จะมี IP Address เป็นของตัวเองเพื่อใช้ในการระบุตัวตน เช่น Mushroom Node มี IP Address คือ X1 และ STA Node มี IP Address คือ X0

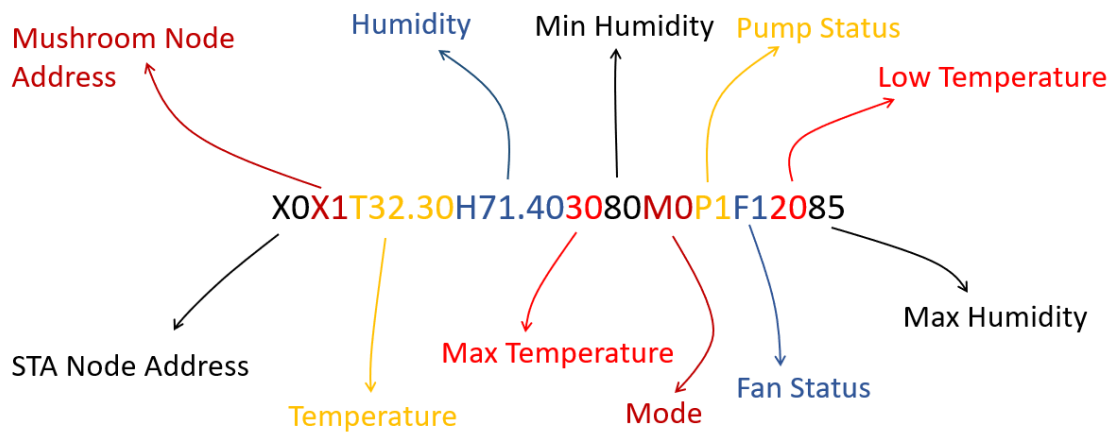
3.6.1 การออกแบบการสื่อสาร Lora ที่ Mushroom Node

ในการออกแบบการสื่อสารผ่าน Lora ที่ Mushroom Node ผู้จัดทำจะขออธิบายรายละเอียดต่างๆ โดยใช้ Flowchart ประกอบการอธิบาย



รูปที่ 3.ป แสดงภาพ Flowchart การทำงานของ Mushroom Node (1)

จากรูปที่ 3.ป แสดงการทำงานของ Mushroom Node เริ่มต้นระบบจะตรวจสอบว่าถ้ามีการเปลี่ยนแปลงของข้อมูลก็จะส่งข้อมูลการเปลี่ยนแปลง (C3) ไปยัง STA Node เช่น เมื่อผู้ใช้กดปุ่มเปลี่ยนโหมดการทำงานที่ Mushroom Node จากนั้น Mushroom Node ก็จะส่งข้อมูลการเปลี่ยนแปลงโหมดการทำงานไปยัง STA Node เพื่อให้ STA Node ทราบถึงการเปลี่ยนแปลงของข้อมูลโดยมีรูปแบบของข้อมูล (Data Format) ดังรูปที่ 3.ป



รูปที่ 3.ป แสดงภาพรูปแบบของข้อมูล (Data Format)

จากรูปที่ 3.ป แสดงรูปแบบของข้อมูล (Data Format) ที่ส่งไปยัง STA Node เมื่อข้อมูลมีการเปลี่ยนแปลงโดยมีรายละเอียดดังต่อไปนี้

- X0 คือ IP Address ของ STA Node หรือ Node ปลายทางที่ต้องการส่งไปเพื่อให้ปลายทางหรือ STA Node ทราบว่าข้อมูลที่ส่งไปนั้นเป็นข้อมูลที่ต้องการส่งไปยังปลายทางหรือ STA Node
- X1 คือ IP Address ของ Mushroom Node หรือ Node ต้นทางโดยมีวัตถุประสงค์คือเพื่อให้ Node ปลายทางหรือ STA Node ทราบว่าข้อมูลที่ส่งมาเป็นของ Node ไหน ผู้จัดทำได้ออกแบบในลักษณะนี้เพื่อกรณีมี Node ต้นทางมากกว่า 1 Node
- T32.30 คือค่าอุณหภูมิที่วัดได้จากเซนเซอร์ AM2315 โดย T คือ Keyword ที่ใช้บอกว่าค่าตัวเลขที่ต่อท้ายมาด้วยจำนวน 4 หลักเป็นค่าอุณหภูมิ
- H71.40 คือค่าความชื้นที่วัดได้จากเซนเซอร์ AM2315 โดย H คือ Keyword ที่ใช้บอกว่าค่าตัวเลขที่ต่อท้ายมาด้วยจำนวน 4 หลักเป็นค่าความชื้น
- 30 คือค่าอุณหภูมิสูงสุดที่ใช้ในการควบคุมอุณหภูมิ
- 80 คือค่าอุณหภูมิต่ำสุดที่ใช้ในการควบคุมความชื้น
- M0 คือโหมด (Mode) การทำงานของระบบควบคุมอุณหภูมิและความชื้นภายในโรงเรือนโดย M คือ Keyword ที่ใช้บอกว่าค่าตัวเลขที่ต่อท้ายมาด้วยจำนวน 1 หลัก

เป็นค่าโหมดการทำงานของระบบ 0 คือเป็นโหมดการทำงานแบบอัตโนมัติ (AUTO Mode) และถ้าเป็น 1 คือเป็นโหมดการทำงานแบบควบคุมเอง (MANUAL Mode)

- P1 คือค่าสถานะการทำงานของปั๊ม (Pump) ที่อ่านได้จากวงจรตัวต้านทาน ตรวจสอบกระแสโดย P คือ Keyword ที่ใช้บอกว่าค่าตัวเลขที่ต่อท้ายมาด้วยจำนวน 1 หลักเป็นค่าสถานะการทำงานของปั๊ม (Pump) 1 คือค่าที่บอกว่าปั๊ม (Pump) กำลังทำงานถ้าเป็น 0 คือค่าที่บอกว่าปั๊ม (Pump) ไม่มีการทำงาน
- F1 คือค่าสถานะการทำงานของพัดลม (Fan) ที่อ่านได้จากวงจรตัวต้านทาน ตรวจสอบกระแสโดย F คือ Keyword ที่ใช้บอกว่าค่าตัวเลขที่ต่อท้ายมาด้วยจำนวน 1 หลักเป็นค่าสถานะการทำงานของพัดลม (Fan) 1 คือค่าที่บอกว่าพัดลม (Fan) กำลังทำงานถ้าเป็น 0 คือค่าที่บอกว่าพัดลม (Fan) ไม่มีการทำงาน
- 20 คือค่าอุณหภูมิต่ำสุดที่ใช้ในการควบคุมอุณหภูมิ
- 85 คือค่าอุณหภูมิสูงสุดที่ใช้ในการควบคุมความชื้น

การส่งข้อมูลการเปลี่ยนแปลงไปยัง Node ปลายทางหรือ STA Node จะเรียกใช้ฟังก์ชัน `sendUpdateData()` ในการส่งข้อมูล ฟังก์ชัน `sendUpdateData()` เป็นฟังก์ชันที่เขียนด้วยภาษา C++ ดังรูปที่ 3.ป

```

void sendUpdateData() { //this function will cal
//-----
//change pumpState and fanState before sendUpd
if (Rs_pump_voltage > 0.10) {
    pump_check = true;
} else {
    pump_check = false;
}
if (Rs_fan_voltage > 0.10) {
    fan_check = true;
} else {
    fan_check = false;
}
//-----
LoRa.beginPacket();
LoRa.print(String(des) + String(ipAddr));
LoRa.print("T");
if (temp < 10.00) {
    String tempStr = "0" + String(temp);
    LoRa.print(tempStr);
} else {
    LoRa.print(temp);
}
LoRa.print("H");
if (humi < 10.00) {
    String humiStr = "0" + String(humi);
    LoRa.print(humiStr);
} else {
    LoRa.print(humi);
}
LoRa.print(set_temp_max);
LoRa.print(set_humi_min);
LoRa.print("M");
LoRa.print(ctrlMode);
LoRa.print("P");
LoRa.print(pump_check);
LoRa.print("F");
LoRa.print(fan_check);
LoRa.print(set_temp_min);
LoRa.print(set_humi_max);
LoRa.endPacket();
}

```

รูปที่ 3.ป แสดงภาพฟังก์ชัน sendUpdateData()

จากรูปที่ 3.ป แสดงฟังก์ชัน sendUpdateData() ที่ใช้ส่งข้อมูลเมื่อมีการเปลี่ยนแปลงโดยรายละเอียดมีการทำงานดังต่อไปนี้ ก่อนส่งข้อมูลจะตรวจสอบค่าแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแสก่อนเพื่อให้ได้ค่าสถานะการทำงานของอุปกรณ์ที่ตรงกับค่าจริง จากนั้นจะส่งข้อมูลที่มีรูปแบบ (Data Format) เหมือนรูปที่ 3.ป ตัวแปรต่างๆที่ถูกใช้ในฟังก์ชันมีรายละเอียดดังต่อไปนี้

- Rs_pump_voltage แรงดันที่อ่านได้จากวงจรตัวต้านทานตรวจสอบกระแสที่ใช้กับปั๊ม (Pump) ผ่าน ADC Pin
- Rs_fan_voltage แรงดันที่อ่านได้จากวงจรตัวต้านทานตรวจสอบกระแสที่ใช้กับพัดลม (Fan) ผ่าน ADC Pin
- pump_check ค่าที่บอกสถานะการทำงานของปั๊ม (Pump)
- fan_check ค่าที่บอกสถานะการทำงานของพัดลม (Fan)
- des IP Address ของ STA Node หรือ Node ปลายทาง
- ipAddr IP Address ของ Mushroom Node หรือ Node ต้นทาง
- temp ค่าอุณหภูมิที่วัดได้จากเซนเซอร์ AM2315
- tempStr ค่าที่ใช้ส่งแทนค่า temp ในกรณีที่อุณหภูมิต่ำกว่า 10.00 เพื่อป้องกันความยาวของข้อมูลคลาดเคลื่อน
- humi ค่าความชื้นที่วัดได้จากเซนเซอร์ AM2315
- humiStr ค่าที่ใช้ส่งแทนค่า humi ในกรณีที่ความชื้นต่ำกว่า 10.00 เพื่อป้องกันความยาวของข้อมูลคลาดเคลื่อน
- set_temp_max คือค่าอุณหภูมิสูงสุดที่ใช้ในการควบคุมอุณหภูมิ
- set_humi_min คือค่าอุณหภูมิต่ำสุดที่ใช้ในการควบคุมความชื้น
- ctrlMode คือโหมด (Mode) การทำงานของระบบควบคุมอุณหภูมิและความชื้นภายในโรงเรือน
- set_temp_min คือค่าอุณหภูมิต่ำสุดที่ใช้ในการควบคุมอุณหภูมิ
- set_humi_max คือค่าอุณหภูมิสูงสุดที่ใช้ในการควบคุมความชื้น

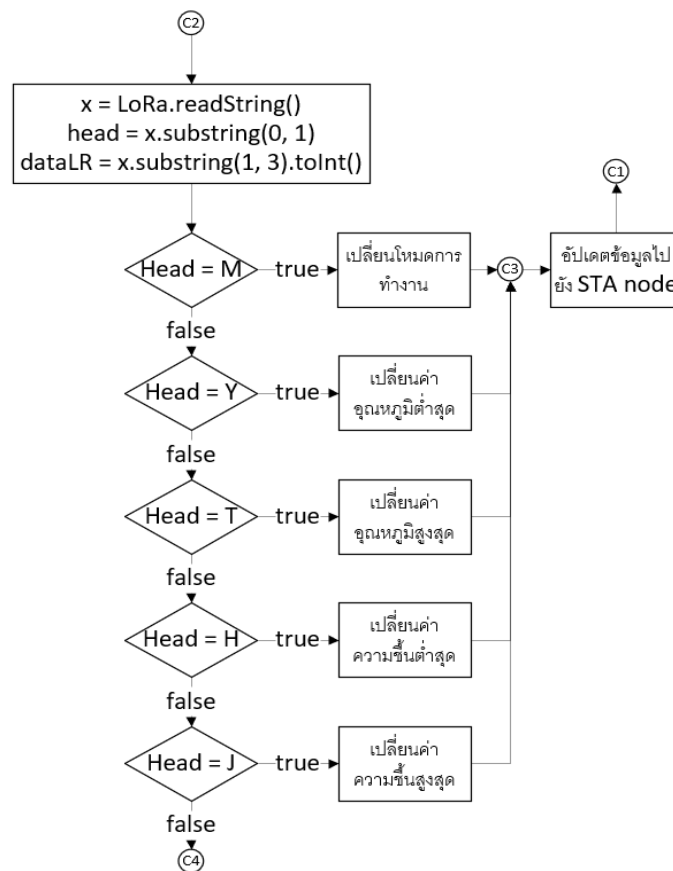
จากรายละเอียดด้านบนเป็นการส่งข้อมูลการเปลี่ยนแปลงไปยัง STA Node ในกรณีที่ข้อมูลมีการเปลี่ยนแปลง แต่ถ้าหากข้อมูลไม่มีการเปลี่ยนแปลงก็จะทำการตรวจสอบเงื่อนไขข้อต่อไปคือการตรวจสอบว่าข้อมูลส่งมาจาก STA Node หรือไม่ ดังรูปที่ 3.ป ถ้าไม่มีข้อมูลที่ส่งมาก็จะกลับไปตรวจสอบการเปลี่ยนแปลงของข้อมูล ถ้ามีข้อมูลส่งมาจาก STA Node ก็จะทำให้การหา IP Address - ของตัวเอง (Mushroom Node IP Address) ว่าข้อมูลที่ส่งมาเป็นข้อมูลของตัวเองหรือไม่ ถ้าใช่ก็จะนำข้อมูลที่ได้มาประมวลผลต่อไป

```

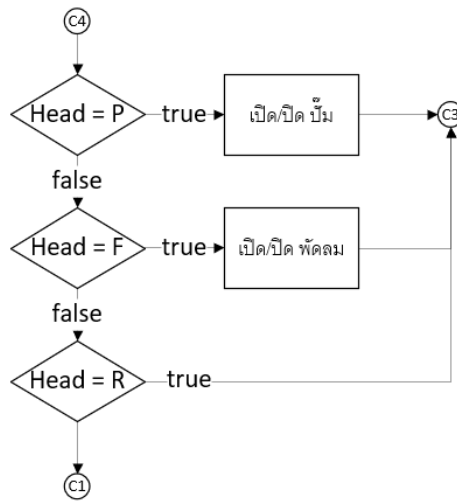
//LoRa-----
if (LoRa.parsePacket()) {
  // received a packet
  // Serial.print("Received packet ");
  // read packet
  while (LoRa.available()) {
    if (LoRa.find(ipAddr)) {
      x = LoRa.readString();
      //Serial.println(x);
      String head = x.substring(0, 1);
      int dataLR = x.substring(1, 3).toInt();

```

รูปที่ 3.ป แสดงภาพโปรแกรมส่วนที่ตรวจสอบข้อมูลที่ส่งมายัง Mushroom Node



รูปที่ 3.ป แสดงภาพ Flowchart การทำงานของ Mushroom Node (2)



รูปที่ 3.ป แสดงภาพ Flowchart การทำงานของ Mushroom Node (3)

จากรูปที่ 3.ป และ รูปที่ 3.ป เมื่อได้รับข้อมูลที่ส่งมาจาก STA Node แล้วขั้นตอนต่อไปคือการนำข้อมูลมาประมวลผลโดยตัวอย่างของข้อมูลที่ถูกส่งมาจาก STA Node มีดังนี้ X1M, X1Y20, X1T30, X1H80, X1J85, X1P, X1F, หรือ X1R เป็นต้น มีรายละเอียดการทำงานในแต่ละข้อมูลดังนี้

```

if (head == "M") {
    if (ctrlMode == false) {
        ctrlMode = true;
    }
    else {
        ctrlMode = false;
    }
    sendUpdateData();
}

```

รูปที่ 3.ป แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (1)

จากรูปที่ 3.ป ถ้าข้อมูลที่ได้รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “M” หมายความว่าต้องการเปลี่ยนโหมดการทำงานของระบบควบคุมอุณหภูมิและความชื้นโดยการทำงานคือจะตรวจสอบ

ว่าถ้าโหมดการทำงานเป็นโหมด AUTO (ctrlMode = false) จะเปลี่ยนเป็นโหมด MANUAL (ctrlMode = true) และถ้าโหมดการทำงานเป็นโหมด MANUAL (ctrlMode = true) ก็จะเปลี่ยนโหมดการทำงานเป็นโหมด AUTO (ctrlMode = false) กล่าวคือเป็นการทำงานแบบสลับ (toggle) โหมดการทำงาน เมื่อเปลี่ยนโหมดการทำงานสำเร็จแล้วก็จะส่งข้อมูลการเปลี่ยนแปลงกลับไปยัง STA Node โดยใช้ฟังก์ชัน sendUpdateData()

```
else if (head == "Y") {  
    set_temp_min = dataLR;  
    EEPROM.write(1, set_temp_min);  
    EEPROM.commit();  
    sendUpdateData();  
}
```

รูปที่ 3.ป แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (2)

จากรูปที่ 3.ป ถ้าข้อมูลที่ได้รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “Y” หมายความว่าต้องการเปลี่ยนแปลงค่าอุณหภูมิต่ำสุดที่ใช้ในการควบคุมอุณหภูมิ เช่น ปัจจุบันค่าอุณหภูมิต่ำสุดคือ 20°C ถ้าได้รับ Keyword “Y” ส่งมาจาก STA Node แสดงว่าต้องการเปลี่ยนค่าอุณหภูมิต่ำสุด โดยข้อมูลที่ส่งมาอาจจะเป็น Y21 ค่าตัวเลข 2 หลักต่อจาก Keyword เป็นค่าอุณหภูมิต่ำสุดที่ต้องการเปลี่ยน เมื่อได้รับค่าที่ต้องการเปลี่ยนแล้วก็จะเปลี่ยนค่าแล้วบันทึกค่าที่เปลี่ยนลงใน EEPROM ตำแหน่งที่ 1 จากนั้นก็จะส่งข้อมูลการเปลี่ยนแปลงกลับไปยัง STA Node โดยใช้ฟังก์ชัน sendUpdateData()

```
else if (head == "T") {  
    set_temp_max = dataLR;  
    EEPROM.write(0, set_temp_max);  
    EEPROM.commit();  
    sendUpdateData();  
}
```

รูปที่ 3.ป แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (3)

จากรูปที่ 3.ป ถ้าข้อมูลที่รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “T” หมายความว่าต้องการเปลี่ยนแปลงค่าอุณหภูมิสูงสุดที่ใช้ในการควบคุมอุณหภูมิ เช่น ปัจจุบันค่าอุณหภูมิสูงสุดคือ 32°C ถ้าได้รับ Keyword “T” ส่งมาจาก STA Node แสดงว่าต้องการเปลี่ยนค่าอุณหภูมิสูงสุด โดยข้อมูลที่ส่งมาอาจจะเป็น T30 ค่าตัวเลข 2 หลักต่อจาก Keyword เป็นค่าอุณหภูมิสูงสุดที่ต้องการเปลี่ยน เมื่อได้รับค่าที่ต้องการเปลี่ยนแล้วก็จะเปลี่ยนค่าแล้วบันทึกค่าที่เปลี่ยนลงใน EEPROM ตำแหน่งที่ 0 จากนั้นก็จะส่งข้อมูลการเปลี่ยนแปลงกลับไปยัง STA Node โดยใช้ฟังก์ชัน sendUpdateData()

```
else if (head == "H") {  
    set_humi_min = dataLR;  
    EEPROM.write(10, set_humi_min);  
    EEPROM.commit();  
    sendUpdateData();  
}
```

รูปที่ 3.ป แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (4)

จากรูปที่ 3.ป ถ้าข้อมูลที่รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “H” หมายความว่าต้องการเปลี่ยนแปลงค่าความชื้นต่ำสุดที่ใช้ในการควบคุมความชื้น เช่น ปัจจุบันค่าความชื้นต่ำสุดคือ 80% ถ้าได้รับ Keyword “H” ส่งมาจาก STA Node แสดงว่าต้องการเปลี่ยนค่าความชื้นต่ำสุด โดยข้อมูลที่ส่งมาอาจจะเป็น H79 ค่าตัวเลข 2 หลักต่อจาก Keyword เป็นค่าความชื้นต่ำสุดที่ต้องการเปลี่ยน เมื่อได้รับค่าที่ต้องการเปลี่ยนแล้วก็จะเปลี่ยนค่าแล้วบันทึกค่าที่เปลี่ยนลงใน EEPROM ตำแหน่งที่ 10 จากนั้นก็จะส่งข้อมูลการเปลี่ยนแปลงกลับไปยัง STA Node โดยใช้ฟังก์ชัน sendUpdateData()

```
else if (head == "J") {  
    set_humi_max = dataLR;  
    EEPROM.write(9, set_humi_max);  
    EEPROM.commit();  
    sendUpdateData();  
}
```

รูปที่ 3.ป แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (5)

จากรูปที่ 3.ป ถ้าข้อมูลที่ได้รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “J” หมายความว่าต้องการเปลี่ยนแปลงค่าความชื้นสูงสุดที่ใช้ในการควบคุมความชื้น เช่น ปัจจุบันค่าความชื้นสูงสุดคือ 90% ถ้าได้รับ Keyword “H” ส่งมาจาก STA Node แสดงว่าต้องการเปลี่ยนค่าความชื้นสูงสุด โดยข้อมูลที่ส่งมาอาจจะเป็น H85 ค่าตัวเลข 2 หลักต่อจาก Keyword เป็นค่าความชื้นสูงสุดที่ต้องการเปลี่ยน เมื่อได้รับค่าที่ต้องการเปลี่ยนแล้วก็จะเปลี่ยนค่าแล้วบันทึกค่าที่เปลี่ยนลงใน EEPROM ตำแหน่งที่ 9 จากนั้นก็จะส่งข้อมูลการเปลี่ยนแปลงกลับไปยัง STA Node โดยใช้ฟังก์ชัน sendUpdateData()

```
else if (head == "P") {  
    if (pumpState == false) {  
        pumpState = true;  
    } else {  
        pumpState = false;  
    }  
    sendUpdateData();  
}
```

รูปที่ 3.ป แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (6)

จากรูปที่ 3.ป ถ้าข้อมูลที่ได้รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “P” หมายความว่าต้องการ เปิด/ปิด การทำงานของปั๊ม (Pump) โดยมีรายละเอียดการทำงานดังต่อไปนี้ ขั้นแรกจะตรวจสอบว่าถ้าค่าควบคุมการทำงานของปั๊ม (pumpState) มีค่าเท่ากับ false (pumpState == false) หรือสั่งให้ปั๊มหยุดทำงาน ก็จะเปลี่ยนค่าควบคุมการทำงานของปั๊ม (pumpState) ให้มีค่าเท่ากับ true (pumpState = true) เพื่อให้ปั๊มกลับมาทำงาน เช่นเดียวกันถ้าปัจจุบันค่าควบคุมการทำงานของปั๊ม (pumpState) มีค่าเท่ากับ true ก็จะเปลี่ยนค่าควบคุมการทำงานของปั๊ม (pumpState) ให้มีค่าเท่ากับ false กล่าวคือเป็นการทำงานแบบสลับ (toggle) ค่าควบคุมการทำงานของปั๊ม (pumpState) เมื่อเปลี่ยนสำเร็จแล้วก็จะส่งข้อมูลการเปลี่ยนแปลงกลับไปยัง STA Node โดยใช้ฟังก์ชัน sendUpdateData()

```

else if (head == "F") {
    if (fanState == false) {
        fanState = true;
    } else {
        fanState = false;
    }
    sendUpdateData();
}
else if (head == "R") {

    sendUpdateData();
}

```

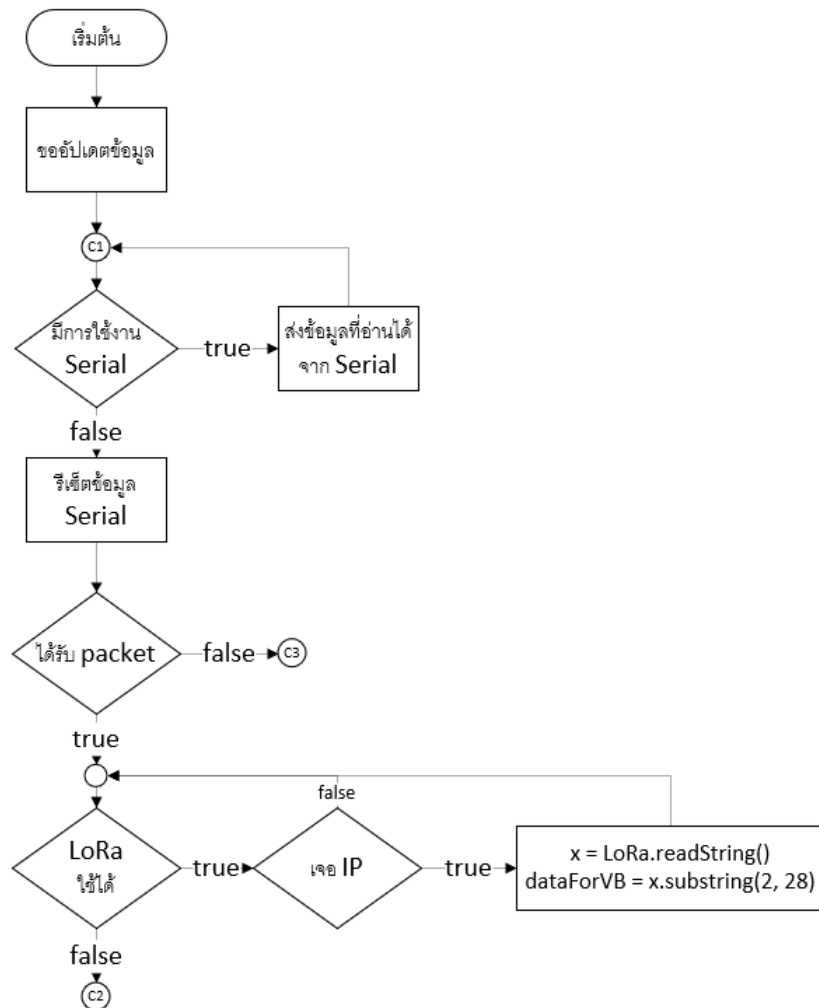
รูปที่ 3.ป แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (7)

จากรูปที่ 3.ป ถ้าข้อมูลที่ได้รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “F” หมายความว่าต้องการ เปิด/ปิด การทำงานของพัดลม (Fan) โดยมีรายละเอียดการทำงานดังต่อไปนี้ ขั้นแรกจะตรวจสอบว่าถ้าค่าควบคุมการทำงานของพัดลม (fanState) มีค่าเท่ากับ false (fanState == false) หรือสั่งให้พัดลมหยุดทำงาน ก็จะเปลี่ยนค่าควบคุมการทำงานของพัดลม (fanState) ให้มีค่าเท่ากับ true (fanState = true) เพื่อให้พัดลมกลับมาทำงาน เช่นเดียวกันถ้าปัจจุบันค่าควบคุมการทำงานของพัดลม (fanState) มีค่าเท่ากับ true ก็จะเปลี่ยนค่าควบคุมการทำงานของพัดลม (fanState) ให้มีค่าเท่ากับ false กล่าวคือเป็นการทำงานแบบสลับ (toggle) ค่าควบคุมการทำงานของพัดลม (fanState) เมื่อเปลี่ยนสำเร็จแล้วก็จะส่งข้อมูลการเปลี่ยนแปลงกลับไปยัง STA Node โดยใช้ฟังก์ชัน sendUpdateData()

ในส่วนต่อมาถ้าข้อมูลที่ได้รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “R” หมายความว่า STA Node ต้องการอัปเดตข้อมูลการเปลี่ยนแปลงโดยให้ Mushroom Node ส่งข้อมูลการเปลี่ยนแปลงไปให้

3.6.2 การออกแบบการสื่อสาร Lora ที่ STA Node

ในการออกแบบการสื่อสารผ่าน Lora ที่ STA Node ผู้จัดทำจะขออธิบายรายละเอียดต่างๆ โดยใช้ Flowchart ประกอบการอธิบาย



รูปที่ 3.ป แสดงภาพ Flowchart การทำงานของ STA Node (1)

จากรูปที่ 3.ป แสดงการทำงานของ STA Node เริ่มต้นจะส่ง Keyword “R” ไปยัง Mushroom Node เพื่อขออัปเดตข้อมูลก่อนเริ่มระบบ โดยข้อมูลที่ส่งไปมีรูปแบบดังนี้ X1R เมื่อ X1 คือ IP Address ของ Mushroom Node เพื่อให้ทราบว่าต้องการส่งข้อมูลไปยัง Mushroom Node และ Keyword “R” คือ Keyword ที่ส่งไปเพื่อให้ Mushroom Node ทราบว่าต้องการอัปเดตข้อมูล

ขั้นตอนต่อไปจะตรวจสอบว่ามีข้อมูลส่งมาจาก Serial Port หรือ Windows Application หรือไม่ถ้ามีก็จะส่งผ่านข้อมูลไปยัง Mushroom Node เลยโดยไม่มีการประมวลผลใดๆ ตัวอย่างข้อมูลที่ส่งมาจาก Serial Port มีดังต่อไปนี้

- M คือ Keyword ที่ส่งเมื่อกดปุ่มเปลี่ยนโหมด
- P คือ Keyword ที่ส่งเมื่อกดปุ่มเปิดหรือปิดปั๊ม
- F คือ Keyword ที่ส่งเมื่อกดปุ่มเปิดหรือปิดพัดลม
- Y คือ Keyword ที่ส่งเมื่อเปลี่ยนค่าอุณหภูมิต่ำสุดตามด้วยตัวเลขจำนวน 2 หลัก เช่น Y25 เมื่อ 25 คือค่าที่ต้องการเปลี่ยน
- T คือ Keyword ที่ส่งเมื่อเปลี่ยนค่าอุณหภูมิสูงสุดตามด้วยตัวเลขจำนวน 2 หลัก เช่น T32 เมื่อ 32 คือค่าที่ต้องการเปลี่ยน
- H คือ Keyword ที่ส่งเมื่อเปลี่ยนค่าความชื้นต่ำสุดตามด้วยตัวเลขจำนวน 2 หลัก เช่น H80 เมื่อ 80 คือค่าที่ต้องการเปลี่ยน
- J คือ Keyword ที่ส่งเมื่อเปลี่ยนค่าความชื้นสูงสุดตามด้วยตัวเลขจำนวน 2 หลัก เช่น J90 เมื่อ 90 คือค่าที่ต้องการเปลี่ยน

จากรายละเอียดด้านบนเป็นข้อมูลจาก Serial Port เมื่อได้รับข้อมูลจาก Serial Port แล้วก็ส่งข้อมูลต่อไปยัง Mushroom Node ทันทีดังรูปที่แสดงด้านล่าง

```
while (Serial.available())
{ //Serial from VB
  line = Serial.readString();
  loraSend(node1 + line);
}
line = "";
```

รูปที่ 3.ป แสดงภาพโปรแกรมส่วนที่รับข้อมูลจาก Serial Port

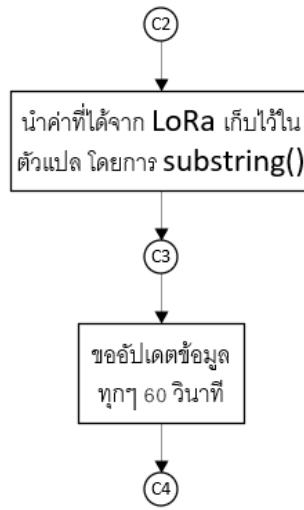
จากรูปที่ 3.ป จะรับข้อมูลจาก Serial Port แล้วส่งข้อมูลไปยัง Mushroom Node ด้วยฟังก์ชัน loraSend() โดยข้อมูลที่ส่งจะมีรูปแบบ (Data Format) ดังนี้ X1M เมื่อ X1 คือ IP Address

ของ Mushroom Node และ M คือ Keyword ที่ส่งไปเพื่อควบคุมการทำงานของระบบดังรายละเอียดด้านบน จากนั้นจะล้างข้อมูลที่ได้รับจาก Serial Port

เมื่อระบบทำงานผ่านขั้นตอนต่างๆ ด้านบนมาแล้วก็ขั้นตอนต่อไปจะทำการตรวจสอบเงื่อนไขข้อต่อไปคือการตรวจสอบว่าข้อมูลส่งมาจาก Mushroom Node หรือไม่ ดังรูปที่ 3.ป ถ้าไม่มีข้อมูลที่ส่งมาก็จะส่งข้อมูลเพื่อขออัปเดตข้อมูล (C3) ทุกๆ 60 วินาที ถ้ามีข้อมูลส่งมาจาก Mushroom Node ก็จะทำกรหา IP Address -ของตัวเอง (STA Node IP Address) ว่าข้อมูลที่ส่งมาเป็นข้อมูลของตัวเองหรือไม่ ถ้าใช่ก็นำข้อมูลที่ได้มาประมวลผลต่อไป

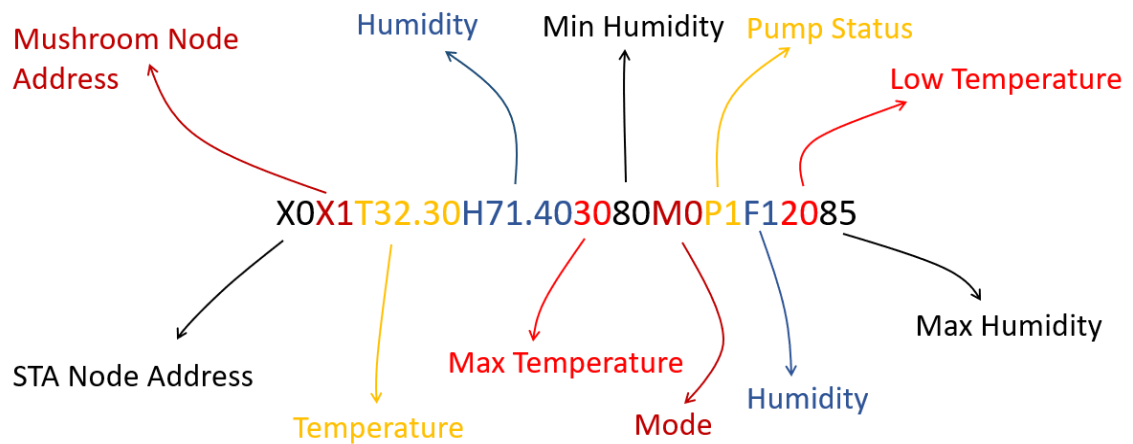
```
// try to parse packet
if (LoRa.parsePacket())
{
    // received a packet
    //Serial.print("Received packet ");
    // read packet
    while (LoRa.available())
    {
        if (LoRa.find(ipAddr))
        { // ip address this device "
            x = LoRa.readString();
            dataForVB = x.substring(2, 28);
            Serial.println(dataForVB);
        }
    }
}
```

รูปที่ 3.ป แสดงภาพโปรแกรมส่วนที่ตรวจสอบข้อมูลที่ส่งมายัง STA Node



รูปที่ 3.ป แสดงภาพ Flowchart การทำงานของ STA Node (2)

เมื่อได้รับข้อมูลที่ส่งมาจาก Mushroom Node แล้วก็จะนำข้อมูลที่ได้อีกมาเก็บไว้ในตัวแปรต่างๆ โดยใช้ฟังก์ชัน `substring()` เพื่อนำข้อมูลที่ได้ไปใช้งานบน Web Application และ Windows Application ตัวอย่างรูปแบบของข้อมูล (Data Format) ที่ส่งมาจาก Mushroom Node ดังรูปที่ 3.ป



รูปที่ 3.ป แสดงภาพรูปแบบของข้อมูล (Data Format) ที่ส่งมาจาก Mushroom Node

จากรูปที่ 3.ป แสดงรูปแบบของข้อมูลที่ส่งมาจาก Mushroom Node เมื่อได้รับข้อมูลก็จะนำข้อมูลที่ได้อามา substring() ดังรูปที่ 3.ป

```
node = x.substring(0, 2);
if (node == "X1")
{
    temp = x.substring(3, 8).toFloat();
    humi = x.substring(9, 14).toFloat();
    set_temp_max = x.substring(14, 16).toInt();
    set_humi_min = x.substring(16, 18).toInt();
    ctrlMode = x.substring(19, 20).toInt();
    pumpState = x.substring(21, 22).toInt();
    fanState = x.substring(23, 24).toInt();
    set_temp_min = x.substring(24, 26).toInt();
    set_humi_max = x.substring(26, 28).toInt();
}
```

รูปที่ 3.ป แสดงภาพการใช้ฟังก์ชัน substring() กับข้อมูลที่ได้รับจาก Mushroom Node

จากรูปที่ 3.ป แสดงการใช้ฟังก์ชัน substring() เพื่อนำข้อมูลที่ได้รับมาเก็บไว้ในตัวแปรที่ถูกต้องโดยมีรายละเอียดดังต่อไปนี้

- ตัวแปร node เป็นตัวแปรที่ใช้เก็บข้อมูลว่าข้อมูลที่ได้รับถูกส่งมาจาก Node ไหน โดยการ substring(0, 2) จากตำแหน่งที่ 0 ถึงตำแหน่งที่ 2
- ตัวแปร temp เป็นค่าอุณหภูมิ โดยการ substring(3, 8) จากตำแหน่งที่ 3 ถึงตำแหน่งที่ 8 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Float
- ตัวแปร humi เป็นค่าความชื้น โดยการ substring(9, 14) จากตำแหน่งที่ 9 ถึงตำแหน่งที่ 14 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Float
- ตัวแปร set_temp_max เป็นค่าอุณหภูมิสูงสุด โดยการ substring(14, 16) จากตำแหน่งที่ 14 ถึงตำแหน่งที่ 16 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Int
- ตัวแปร set_humi_min เป็นค่าความชื้นต่ำสุด โดยการ substring(16, 18) จากตำแหน่งที่ 16 ถึงตำแหน่งที่ 18 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Int

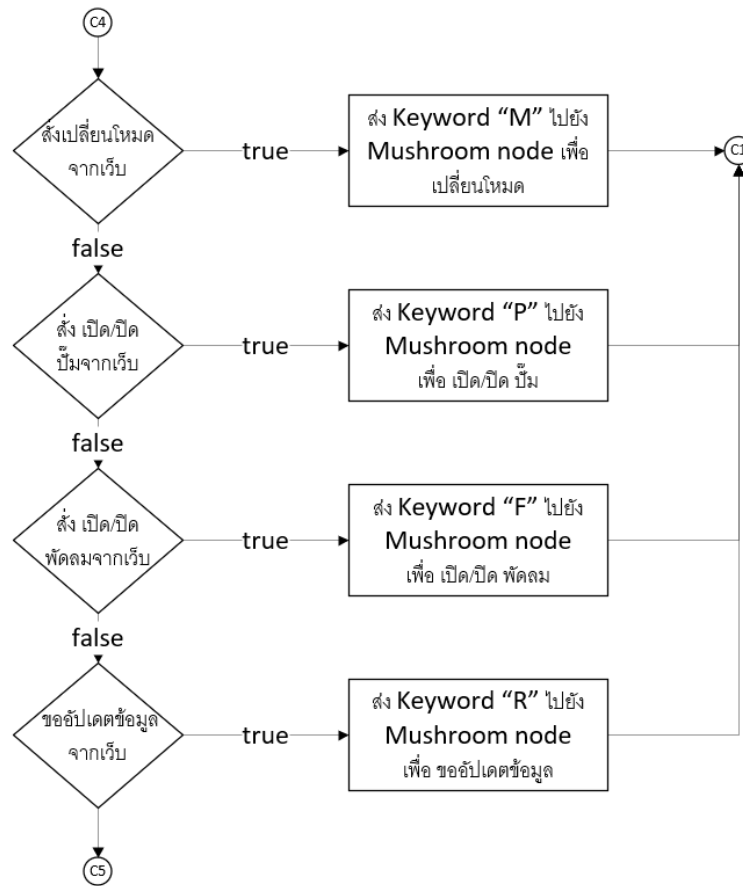
- ตัวแปร ctrlMode เป็นโหมดการทำงานของระบบควบคุมอุณหภูมิและความชื้นภายในโรงเรือน โดยการ substring(19, 20) จากตำแหน่งที่ 19 ถึงตำแหน่งที่ 20 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Int
- ตัวแปร pumpState เป็นค่าสถานะการทำงานของปั๊มที่ได้จากวงจรตัวต้านทานตรวจสอบกระแสโดยการ substring(21, 22) จากตำแหน่งที่ 21 ถึงตำแหน่งที่ 22 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Int
- ตัวแปร fanState เป็นค่าสถานะการทำงานของพัดลมที่ได้จากวงจรตัวต้านทานตรวจสอบกระแสโดยการ substring(23, 24) จากตำแหน่งที่ 23 ถึงตำแหน่งที่ 24 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Int
- ตัวแปร set_temp_min เป็นค่าอุณหภูมิต่ำสุด โดยการ substring(24, 26) จากตำแหน่งที่ 24 ถึงตำแหน่งที่ 26 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Int
- ตัวแปร set_humi_max เป็นค่าความชื้นต่ำสุด โดยการ substring(26, 28) จากตำแหน่งที่ 26 ถึงตำแหน่งที่ 28 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Int

เมื่อจัดการกลับข้อมูลที่ส่งมาเรียบร้อยแล้วขั้นตอนต่อไปคือการส่ง Keyword เพื่อขออัปเดตข้อมูลทุกๆ 60 วินาที ดังรูปที่ 3.ป

```
//for request data via lora every 60s
if (currentMillis - previousMillis_request >= interval_request)
{
    previousMillis_request = currentMillis;
    loraSend(node1 + "R");
}
```

รูปที่ 3.ป แสดงภาพโปรแกรมที่ใช้ในการส่ง Keyword ทุก 60 วินาที

จากรูปที่ 3.ป มีหลักการทำงานคือ STA Node จะส่ง Keyword “R” ทุกๆ 60 วินาทีโดยใช้ฟังก์ชัน millis() ในการนับเวลา ซึ่งตัวแปร node1 มีค่าเท่ากับ X1 คือ IP Address ของ Mushroom Node รูปแบบข้อมูล (Data Format) คือ X1R

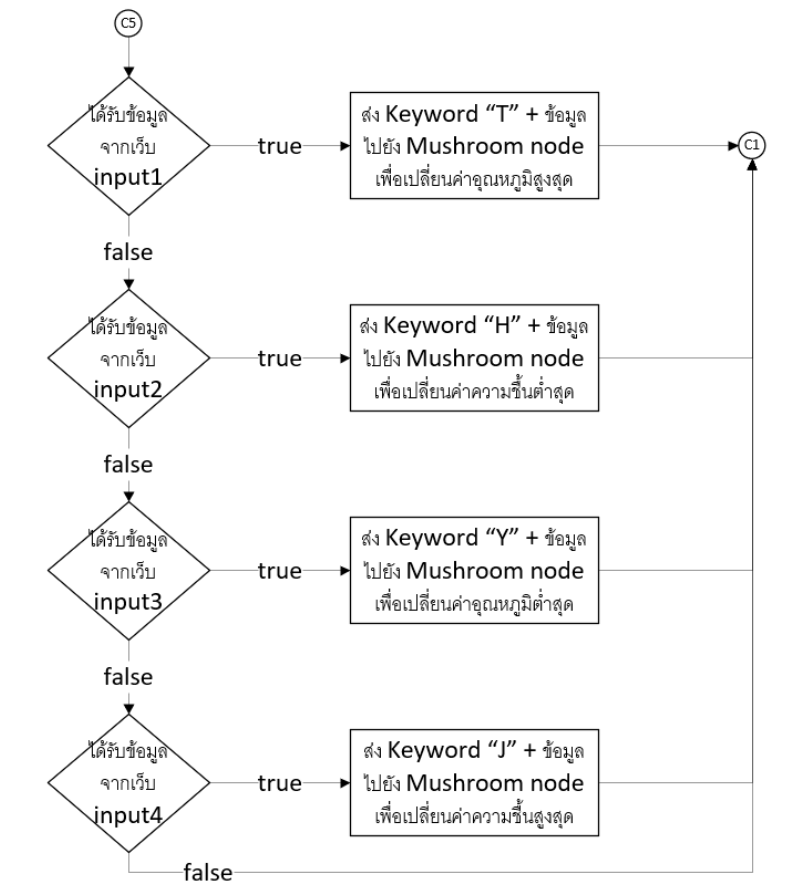


รูปที่ 3.ป แสดงภาพ Flowchart การทำงานของ STA Node (3)

จากรูปที่ 3.ป เมื่อ Keyword เพื่อขออัปเดตข้อมูลทุกๆ 60 วินาทีแล้ว ขั้นตอนต่อไปคือการตรวจสอบว่ามีการควบคุมการทำงานของระบบจากเว็บแอปพลิเคชัน (Web Application) หรือไม่ โดยมีรายละเอียดการทำงานดังต่อไปนี้

- ถ้ามีการกดปุ่มเปลี่ยนโหมดการทำงานจากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “M” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการเปลี่ยนโหมดการทำงาน โดยมีรูปแบบของข้อมูลที่ส่งดังนี้ X1M
- ถ้ามีการกดปุ่ม เปิด/ปิด ปั๊มจากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “M” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการ เปิด/ปิด ปั๊ม โดยมีรูปแบบของข้อมูลที่ส่งดังนี้ X1P

- ถ้ามีการกดปุ่ม เปิด/ปิด พัดลมจากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “F” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการ เปิด/ปิด พัดลม โดยมีรูปแบบของข้อมูลที่ส่งดังนี้ X1F
- ถ้ามีการกดปุ่มขออัปเดตข้อมูลจากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “R” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการอัปเดตข้อมูล โดยมีรูปแบบของข้อมูลที่ส่งดังนี้ X1R

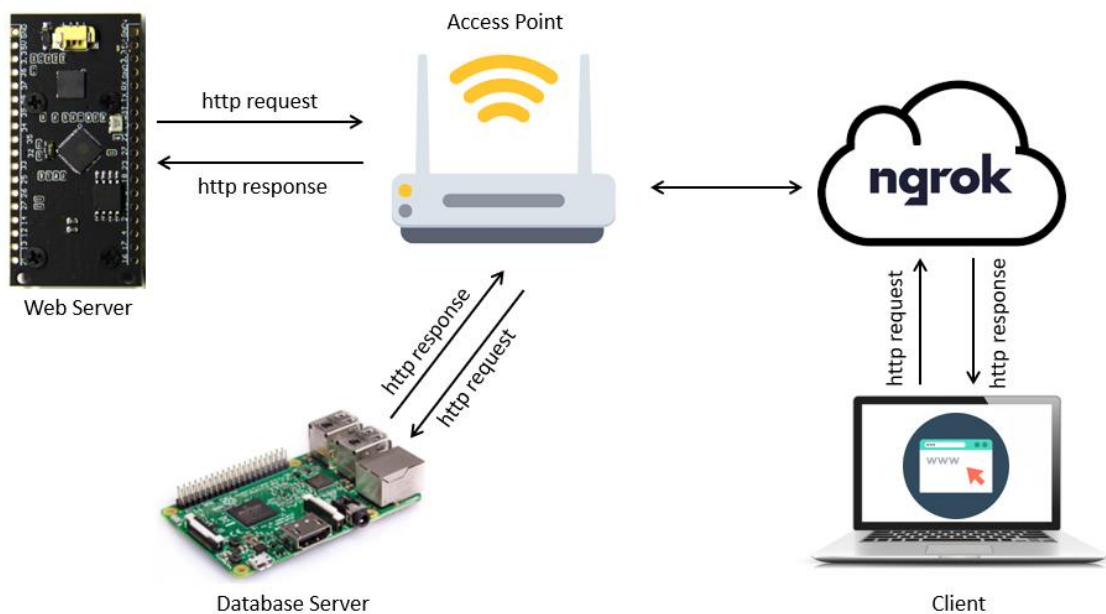


รูปที่ 3.ป แสดงภาพ Flowchart การทำงานของ STA Node (4)

- ถ้ามีการกรอกข้อมูลที่ input1 จากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “T” + “ข้อมูลที่กรอก” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการเปลี่ยนค่าอุณหภูมิสูงสุด โดยมีรูปแบบของข้อมูลที่ส่งดังนี้ X1T32
- ถ้ามีการกรอกข้อมูลที่ input2 จากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “H” + “ข้อมูลที่กรอก” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการเปลี่ยนค่าความชื้นต่ำสุด โดยมีรูปแบบของข้อมูลที่ส่งดังนี้ X1H80
- ถ้ามีการกรอกข้อมูลที่ input3 จากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “Y” + “ข้อมูลที่กรอก” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการเปลี่ยนค่าอุณหภูมิต่ำสุด โดยมีรูปแบบของข้อมูลที่ส่งดังนี้ X1Y25
- ถ้ามีการกรอกข้อมูลที่ input4 จากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “J” + “ข้อมูลที่กรอก” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการเปลี่ยนค่าความชื้นสูงสุด โดยมีรูปแบบของข้อมูลที่ส่งดังนี้ X1J90

3.7 การออกแบบและสร้างเว็บแอปพลิเคชัน

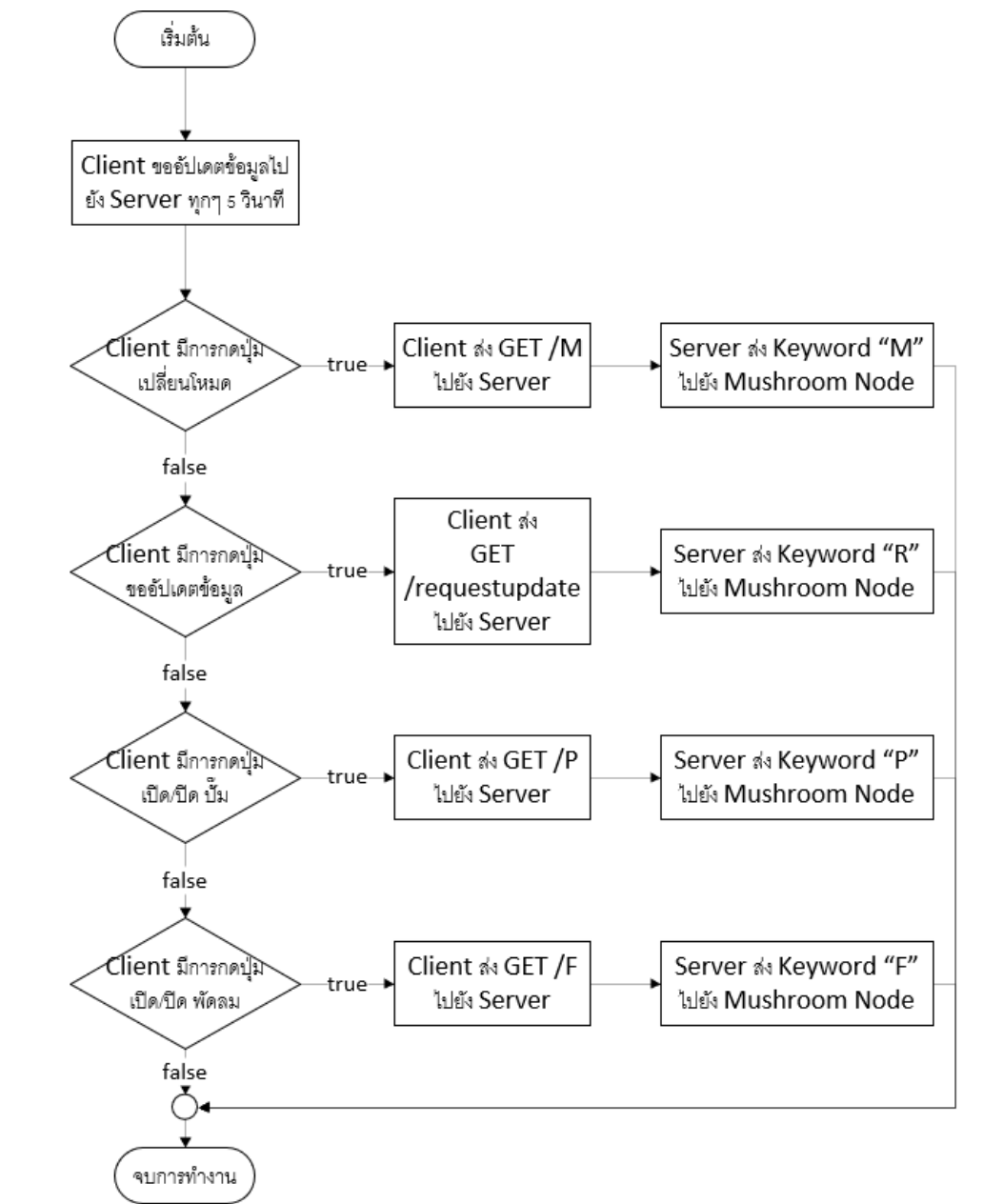
ในโครงงานนี้ผู้จัดทำได้ใช้ภาษา HTML CSS Javascript และ PHP ในการสร้าง Web Application และใช้แอปพลิเคชัน ngrok ที่ติดตั้งบน Raspberry Pi ทำ Port forwarding ของ Web Server เพื่อให้สามารถใช้งานเว็บแอปพลิเคชันได้จากทุกที่ที่สามารถใช้งานอินเทอร์เน็ตได้ Web Application ส่วนที่แสดงค่าสถานะต่างๆและควบคุมการทำงานของระบบจะถูกเก็บอยู่ที่ ESP32LoRa และ Web Application ส่วนที่เกี่ยวกับฐานข้อมูล เช่น บันทึกข้อมูลลงในฐานข้อมูล นำข้อมูลในฐานข้อมูลมาแสดงบน Web Application จะถูกเก็บอยู่ที่ Raspberry Pi 3 Model B เพราะใช้ Raspberry Pi เป็นฐานข้อมูลดังรูปที่ 3.ป



รูปที่ 3.ป แสดงภาพโครงสร้างการทำงานของ Web Application

3.7.1 การสื่อสารระหว่าง Server และ Client

การสื่อสารระหว่าง Server และ Client เป็นการสื่อสารผ่านทาง Hypertext Transfer Protocol (HTTP) โดยมีรายละเอียดการทำงานดังรูปที่ 3.ป



รูปที่ 3.ป แสดงภาพ Flowchart การสื่อสารระหว่าง Server และ Client

จากรูปที่ 3.ป แสดงภาพ Flowchart การสื่อสารระหว่าง Server และ Client โดยมีรายละเอียดการทำงานดังต่อไปนี้

เริ่มต้นการทำงาน Client จะส่ง HTTP GET Method เพื่อขออัปเดตข้อมูลทุกๆ 5 วินาที เพื่อให้หน้าเว็บมีการอัปเดตข้อมูลโดยใช้ฟังก์ต์ setInterval() ในภาษา JavaScript จากนั้นใช้ Ajax (Asynchronous JavaScript And XML) ในการส่ง HTTP GET Method ไปยัง Server เมื่อ Server ตอบกลับมานำข้อมูลที่ได้นำไปแสดงบนหน้าเว็บ ดังภาพด้านล่าง

```
setInterval(function () {  
    var xhttp = new XMLHttpRequest();  
    xhttp.onreadystatechange = function () {  
        if (this.readyState == 4 && this.status == 200) {  
            document.getElementById("temperature").innerHTML = this.responseText;  
        }  
    };  
    xhttp.open("GET", "/temperature", true);  
    xhttp.send();  
}, 5000);
```

รูปที่ 3.ป แสดงภาพตัวอย่างโปรแกรมส่วนที่ใช้ในการขออัปเดตข้อมูลทุกๆ 5 วินาที

จากรูปที่ 3.ป เป็นการขออัปเดตค่าอุณหภูมิจาก Server ทุกๆ 5 วินาทีเพื่อให้ค่าอุณหภูมิที่แสดงบนหน้าเว็บมีการอัปเดต นอกจากนี้ยังมีค่าต่างๆที่ขออัปเดตทุกๆ 5 วินาทีเหมือนค่าอุณหภูมิ เช่น อุณหภูมิต่ำสุด อุณหภูมิสูงสุด ความชื้น ความชื้นต่ำสุด ความชื้นสูงสุด สถานะการทำงานของปั้มและพัดลม โหมดการทำงาน ดังรูปที่ 3.ป



Mushroom Web Server



Temperature **28.10 °C**
Min 25.00 °C Max 30.00 °C



Humidity **90.90 %**
Min 80.00 % Max 99.00 %




Pump **ON**




Fan **OFF**



Mode **AUTO**

 change mode

 request update

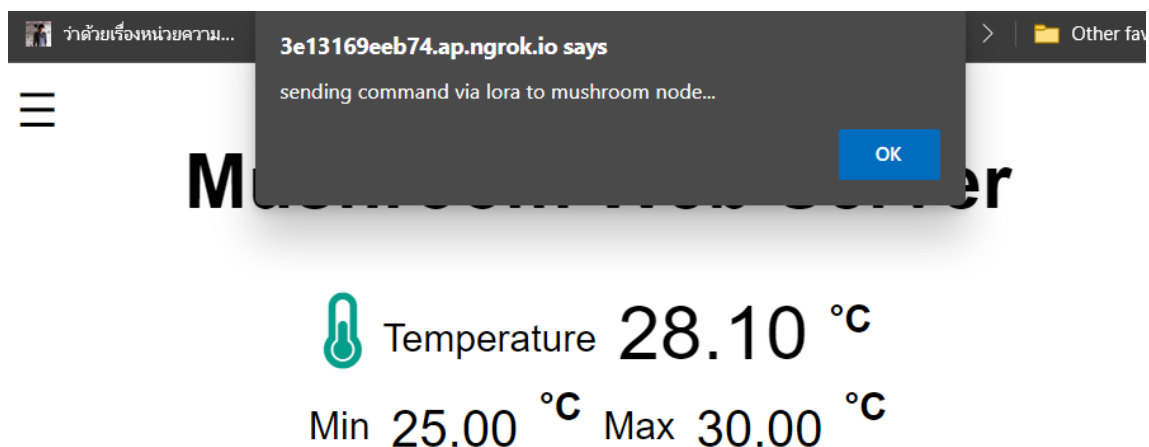
รูปที่ 3.ป แสดงภาพข้อมูลต่างๆบนหน้าเว็บที่อัปเดตทุกๆ 5 วินาที

จากรูปด้านบนแสดงค่าตัวเลขและสถานะต่างๆ Client ขออัปเดตไปยัง Server ทุกๆ 5 วินาที ขั้นตอนต่อไปจะทำตรวจสอบว่ามีการความคุมการทำงานของระบบหรือไม่โดยมีรายละเอียดดังต่อไปนี้

```
function getmode() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            alert(this.responseText);
        }
    };
    xhttp.open("GET", "/M", true);
    xhttp.send();
}
```

รูปที่ 3.ป แสดงภาพฟังก์ท์ที่ถูกเรียกใช้เมื่อกดปุ่มเปลี่ยนโหมดการทำงาน

จากรูปที่ 3.ป ถ้า Client กดปุ่มเปลี่ยนโหมดการทำงาน (change mode) จะเรียกใช้ฟังก์ชัน getmode() ที่เขียนด้วยภาษา JavaScript โดยฟังก์ชันมีการทำงานดังนี้ เมื่อมีการเรียกใช้จะส่ง HTTP GET Method “/M” ไปยัง Server โดยใช้ Ajax จากนั้น Server จะส่ง Keyword “M” ผ่าน Lora ไปยัง Mushroom Node เพื่อขอเปลี่ยนโหมดการทำงาน เมื่อ Server ตอบกลับมายัง Client ว่าส่งข้อมูลสำเร็จ จะแจ้งเตือนดังรูปที่ 3.ป



รูปที่ 3.ป แสดงภาพการแจ้งเตือนเมื่อกดปุ่มเปลี่ยนโหมดการทำงาน

```

function requestupdate() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            alert(this.responseText);
        }
    };
    xhttp.open("GET", "/requestupdate", true);
    xhttp.send();
}

```

รูปที่ 3.ป แสดงภาพฟังก์ชันที่ถูกเรียกใช้เมื่อกดปุ่มขออัปเดตข้อมูล

จากรูปที่ 3.ป ถ้า Client กดปุ่มขออัปเดตข้อมูล (request update) จะเรียกใช้ฟังก์ชัน requestupdate() ที่เขียนด้วยภาษา JavaScript โดยฟังก์ชันมีการทำงานดังนี้ เมื่อมีการเรียกใช้จะส่ง HTTP GET Method “/requestupdate” ไปยัง Server โดยใช้ Ajax จากนั้น Server จะส่ง Keyword “R” ผ่าน Lora ไปยัง Mushroom Node เพื่อขออัปเดตข้อมูล

```

function toggleCheckbox_pump(element) {
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            alert(this.responseText);
        }
    };
    if(element.checked){ xhr.open("GET", "/P", true); }
    else { xhr.open("GET", "/P", true); }
    xhr.send();
}

```

รูปที่ 3.ป แสดงภาพฟังก์ชันที่ถูกเรียกใช้เมื่อกดปุ่ม เปิด/ปิด ปั้ม

จากรูปที่ 3.ป ถ้า Client กดปุ่มควบคุมการทำงานของปั้ม (control PUMP) จะเรียกใช้ฟังก์ชัน toggleCheckbox_pump(element) ที่เขียนด้วยภาษา JavaScript โดยฟังก์ชันมีการทำงาน

ดังนี้ เมื่อมีการเรียกใช้จะส่ง HTTP GET Method “/P” ไปยัง Server โดยใช้ Ajax จากนั้น Server จะส่ง Keyword “P” ผ่าน Lora ไปยัง Mushroom Node เพื่อควบคุมการทำงานของปั๊ม

```
function toggleCheckbox_fan(element) {  
    var xhr = new XMLHttpRequest();  
    xhr.onreadystatechange = function () {  
        if (this.readyState == 4 && this.status == 200) {  
            alert(this.responseText);  
        }  
    };  
    if(element.checked){ xhr.open("GET", "/F", true); }  
    else { xhr.open("GET", "/F", true); }  
    xhr.send();  
}
```

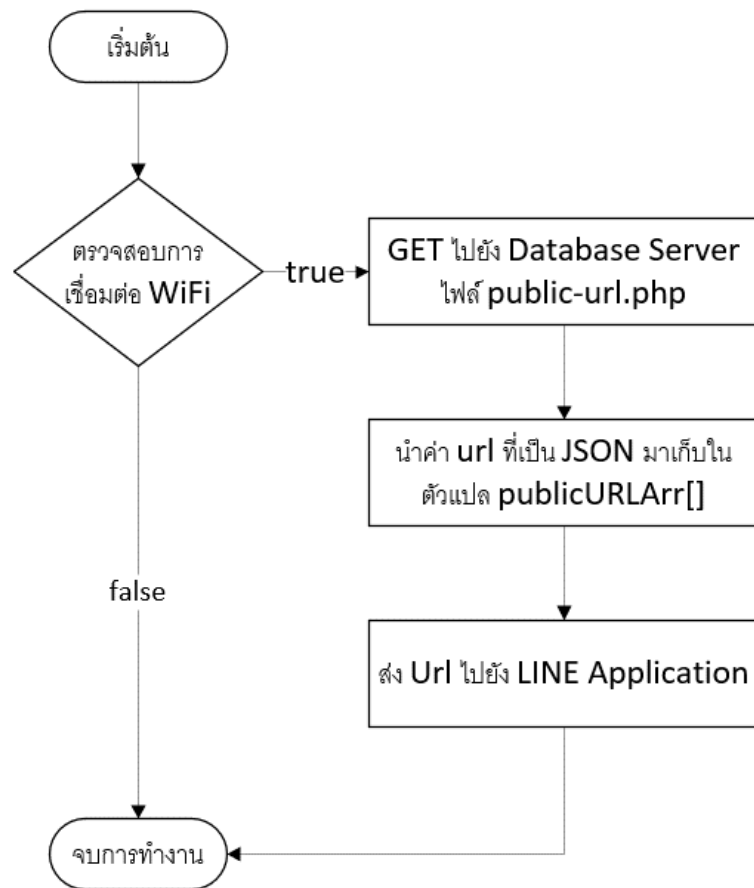
รูปที่ 3.ป แสดงภาพฟังก์ชันที่ถูกเรียกใช้เมื่อกดปุ่ม เปิด/ปิด พัดลม

จากรูปที่ 3.ป ถ้า Client กดปุ่มควบคุมการทำงานของพัดลม (control FAN) จะเรียกใช้ฟังก์ชัน toggleCheckbox_fan(element) ที่เขียนด้วยภาษา JavaScript โดยฟังก์ชันมีการทำงานดังนี้ เมื่อมีการเรียกใช้จะส่ง HTTP GET Method “/F” ไปยัง Server โดยใช้ Ajax จากนั้น Server จะส่ง Keyword “F” ผ่าน Lora ไปยัง Mushroom Node เพื่อควบคุมการทำงานของพัดลม

3.7.2 การส่ง url ของ Web Server และ Database Server ไปยัง LINE Application

เพื่อความสะดวกในการเข้าถึงเว็บแอปพลิเคชัน (Web Application) เพราะเมื่อใช้โปรแกรม Ngrok ทำ Port forwarding แล้วจะได้ random url มา url จะทำการเปลี่ยนไปทุกครั้งเมื่อมีการปิดหรือเปิดใช้งาน Ngrok ตัวอย่าง url ที่ได้จากการ random คือ https://e9d707f7cc96.ap.ngrok.io/ ช่องทางที่ผู้ใช้สามารถดู url ที่ได้ 3 อยู่สามช่องทาง 1. ดูที่หน้าโปรแกรม Ngrok 2. ดูที่เว็บของ Ngrok 3. ดูที่ localhost ของเครื่องที่ติดตั้งโปรแกรม Ngrok โดยเข้าไปที่ url ดังนี้ http://localhost:4040/api/tunnels

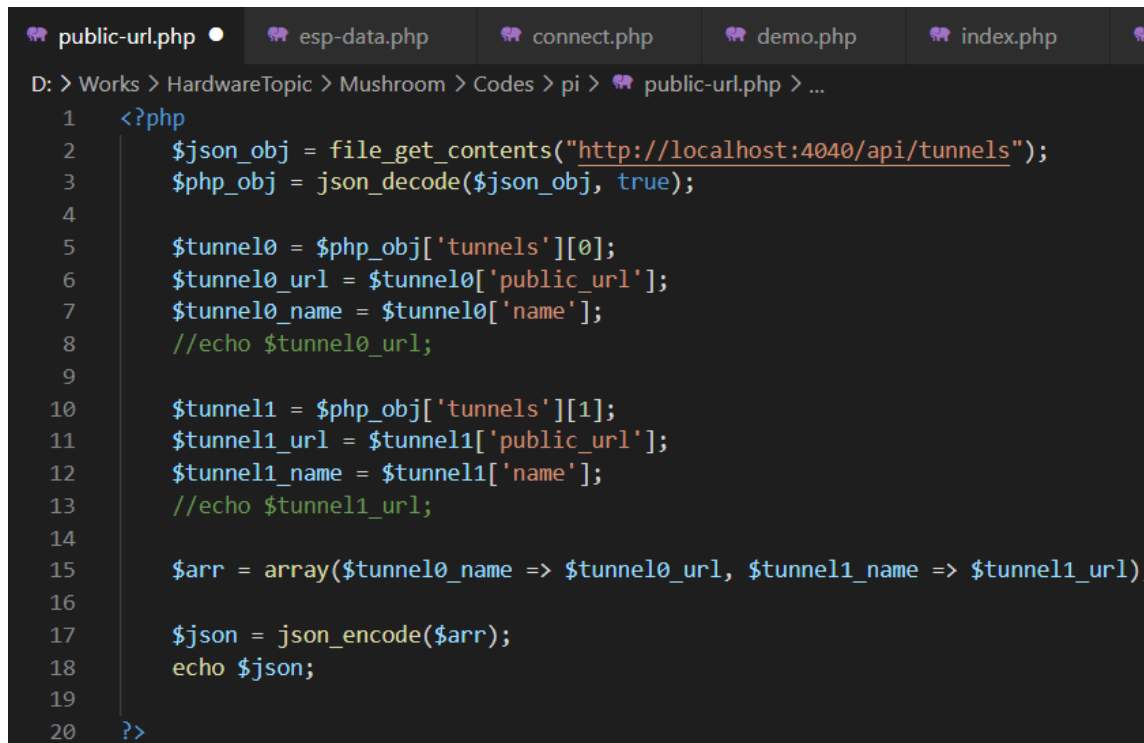
จากทั้ง 3 วิธีที่กล่าวมาผู้จัดทำจึงได้นำวิธีที่ 3 มาประยุกต์ใช้งานเพื่อส่ง url ของเว็บแอปพลิเคชัน (Web Application) ไปยัง LINE Application เพื่อให้สะดวกต่อการเข้าถึงโดยมีขั้นตอนดังรูปที่แสดงด้านล่าง



รูปที่ 3.ป แสดงภาพ Flowchart ขั้นตอนการส่ง url ไปยัง LINE Application

จากรูป 3.x แสดงภาพ Flowchart ขั้นตอนการส่ง url ไปยัง LINE Application โดยมีรายละเอียดดังต่อไปนี้ ขั้นแรกตรวจสอบการเชื่อมต่อ WiFi ถ้าเชื่อมต่อแล้วจะส่ง HTTP GET Method ไปยังไฟล์ public-url.php ที่อยู่ใน Database Server ตัวอย่างที่อยู่ในไฟล์ public-url.php คือ <http://1d7113101b75.ap.ngrok.io/public-url.php> หรือผ่าน Local IP Address เพราะ ESP32LoRa

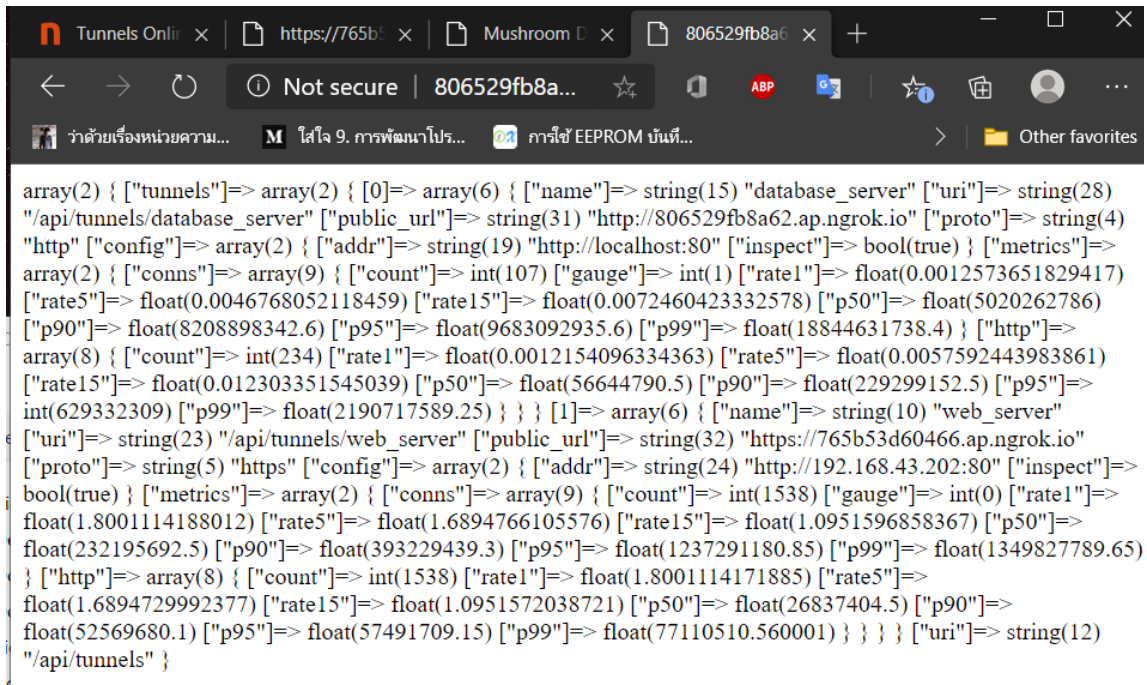
และ Raspberry Pi 3 b เชื่อมต่อกับ Access Point เดียวกัน ตัวอย่าง Local IP Address <http://192.168.43.181/public-url.php> โดยไฟล์ public-url.php มีรายละเอียดดังรูปที่แสดงด้านล่าง



```
public-url.php ● esp-data.php connect.php demo.php index.php
D: > Works > HardwareTopic > Mushroom > Codes > pi > public-url.php > ...
1  <?php
2      $json_obj = file_get_contents("http://localhost:4040/api/tunnels");
3      $php_obj = json_decode($json_obj, true);
4
5      $tunnel0 = $php_obj['tunnels'][0];
6      $tunnel0_url = $tunnel0['public_url'];
7      $tunnel0_name = $tunnel0['name'];
8      //echo $tunnel0_url;
9
10     $tunnel1 = $php_obj['tunnels'][1];
11     $tunnel1_url = $tunnel1['public_url'];
12     $tunnel1_name = $tunnel1['name'];
13     //echo $tunnel1_url;
14
15     $arr = array($tunnel0_name => $tunnel0_url, $tunnel1_name => $tunnel1_url);
16
17     $json = json_encode($arr);
18     echo $json;
19
20  ?>
```

รูปที่ 3.ป แสดงภาพไฟล์ public-url.php

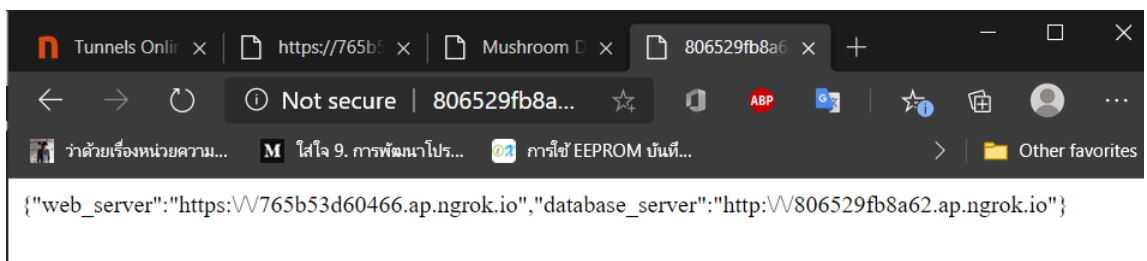
จากรูปที่ 3.ป โปรแกรมที่เขียนด้วยภาษา PHP โดยมีหลักการทำงานคือเรียกไปที่ <http://localhost:4040/api/tunnels> จะได้ข้อมูลรายละเอียดต่างๆ เกี่ยวกับ Web Application เช่น public_url คือตัวแปรที่เก็บค่า url ที่ได้จากโปรแกรม Ngrok ตัวแปร public_url จะถูกเก็บอยู่ในรูปแบบ JSON ดังรูปที่ 3.ป

A screenshot of a web browser window. The address bar shows a URL starting with 'https://765b...'. The page content displays a large JSON object. The JSON has a 'tunnels' key containing an array of two tunnel objects. The first tunnel is for 'database_server' and the second is for 'web_server'. Each tunnel object contains 'uri', 'public_url', 'config', and 'metrics' sub-objects. The 'config' sub-objects include 'addr', 'inspect', and 'metrics' (which contains 'conns' with various statistics like count, gauge, rate1, rate5, p50, p90, p95, p99).

```
array(2) { ["tunnels"]=> array(2) { [0]=> array(6) { ["name"]=> string(15) "database_server" ["uri"]=> string(28) "/api/tunnels/database_server" ["public_url"]=> string(31) "http://806529fb8a62.ap.ngrok.io" ["proto"]=> string(4) "http" ["config"]=> array(2) { ["addr"]=> string(19) "http://localhost:80" ["inspect"]=> bool(true) } ["metrics"]=> array(2) { ["conns"]=> array(9) { ["count"]=> int(107) ["gauge"]=> int(1) ["rate1"]=> float(0.0012573651829417) ["rate5"]=> float(0.0046768052118459) ["rate15"]=> float(0.007246042332578) ["p50"]=> float(5020262786) ["p90"]=> float(8208898342.6) ["p95"]=> float(9683092935.6) ["p99"]=> float(18844631738.4) } ["http"]=> array(8) { ["count"]=> int(234) ["rate1"]=> float(0.0012154096334363) ["rate5"]=> float(0.0057592443983861) ["rate15"]=> float(0.012303351545039) ["p50"]=> float(56644790.5) ["p90"]=> float(229299152.5) ["p95"]=> int(629332309) ["p99"]=> float(2190717589.25) } } [1]=> array(6) { ["name"]=> string(10) "web_server" ["uri"]=> string(23) "/api/tunnels/web_server" ["public_url"]=> string(32) "https://765b53d60466.ap.ngrok.io" ["proto"]=> string(5) "https" ["config"]=> array(2) { ["addr"]=> string(24) "http://192.168.43.202:80" ["inspect"]=> bool(true) } ["metrics"]=> array(2) { ["conns"]=> array(9) { ["count"]=> int(1538) ["gauge"]=> int(0) ["rate1"]=> float(1.8001114188012) ["rate5"]=> float(1.6894766105576) ["rate15"]=> float(1.0951596858367) ["p50"]=> float(232195692.5) ["p90"]=> float(393229439.3) ["p95"]=> float(1237291180.85) ["p99"]=> float(1349827789.65) } ["http"]=> array(8) { ["count"]=> int(1538) ["rate1"]=> float(1.8001114171885) ["rate5"]=> float(1.6894729992377) ["rate15"]=> float(1.0951572038721) ["p50"]=> float(26837404.5) ["p90"]=> float(52569680.1) ["p95"]=> float(57491709.15) ["p99"]=> float(77110510.560001) } } } ["uri"]=> string(12) "/api/tunnels" }
```

รูปที่ 3.ป แสดงภาพข้อมูล JSON ที่ได้จาก http://localhost:4040/api/tunnels

จึงต้องทำการ decode ก่อนที่จะนำมาใช้งานโดยใช้ฟังก์ชัน json_decode ข้อมูลที่ได้จาก http://localhost:4040/api/tunnels ก่อนจากนั้นนำค่าที่ได้มาเก็บไว้ในตัวแปรต่างๆ แล้วสร้าง array เพื่อใช้สำหรับเก็บตัวแปรต่างๆเพื่อให้ง่ายต่อการแปลงกลับไปเป็น JSON อีกครั้งโดยใช้ฟังก์ชัน json_encode ผลลัพธ์ที่ได้จากไฟล์ public-url.php ดังรูปที่ 3.ป

A screenshot of a web browser window. The address bar shows a URL starting with 'https://765b...'. The page content displays a JSON object with two keys: 'web_server' and 'database_server'. The 'web_server' value is 'https://765b53d60466.ap.ngrok.io' and the 'database_server' value is 'http://806529fb8a62.ap.ngrok.io'.

```
{"web_server": "https://765b53d60466.ap.ngrok.io", "database_server": "http://806529fb8a62.ap.ngrok.io"}
```

รูปที่ 3.ป แสดงภาพผลลัพธ์ที่ได้จากไฟล์ public-url.php

เมื่อ Web Server (ESP32LoRa) ใช้ HTTP GET Method ไปยังไฟล์ public-url.php ที่อยู่บน Database Server (Raspberry Pi) แล้วจะนำผลลัพธ์ที่ได้มา decode แล้วเก็บค่าที่ได้ไว้ในตัวแปร tunnelsNameArr[] publicURLArr[] และ ดังรูปที่แสดงด้านล่าง

```
if (WiFi.status() == WL_CONNECTED)
{

    json = httpGETRequest(serverName);
    Serial.println(json);
    JSONVar myObject = JSON.parse(json);

    // JSON.typeof(jsonVar) can be used to get the type of the var
    if (JSON.typeof(myObject) == "undefined")
    {
        Serial.println("Parsing input failed!");
        return;
    }

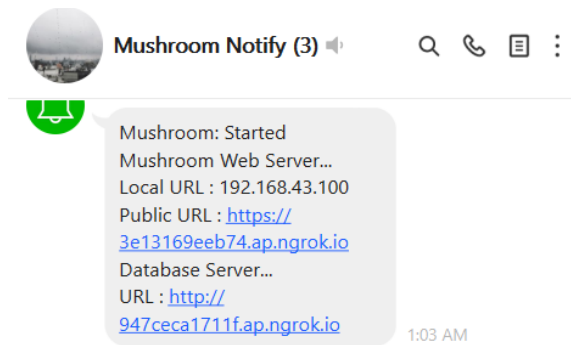
    Serial.print("JSON object = ");
    Serial.println(myObject);

    // myObject.keys() can be used to get an array of all the keys in the object
    JSONVar keys = myObject.keys();

    for (int i = 0; i < keys.length(); i++)
    {
        JSONVar value = myObject[keys[i]];
        //Serial.print(keys[i]);
        //Serial.print(" = ");
        //Serial.println(value);
        tunnelsNameArr[i] = keys[i];
        publicURLArr[i] = value;
    }
    Serial.print(tunnelsNameArr[0] + " = ");
    Serial.println(publicURLArr[0]);
    Serial.print(tunnelsNameArr[1] + " = ");
    Serial.println(publicURLArr[1]);
}
else
{
    Serial.println("WiFi Disconnected -Get ngrok public url");
}
```

รูปที่ 3.ป แสดงภาพโปรแกรมส่วนที่จัดการจัด url ที่ได้จากโปรแกรม Ngrok

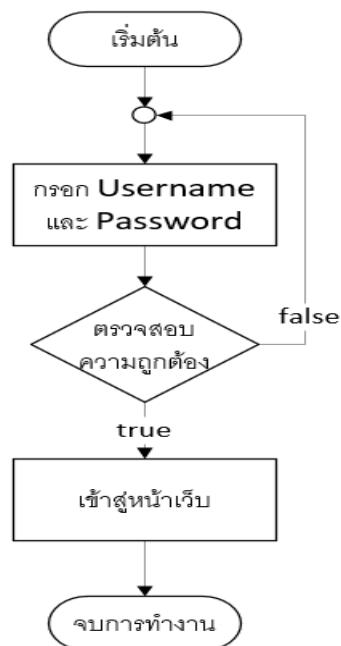
เมื่อได้ url ของ Web Server และ Database Server แล้วจะส่ง url ที่ได้ไปยัง LINE Application โดยใช้ฟังก์ชัน NotifyLine()



รูปที่ 3.ป แสดงภาพตัวอย่าง url ของ Web Server และ Database Server ที่ส่งไปยัง LINE Application

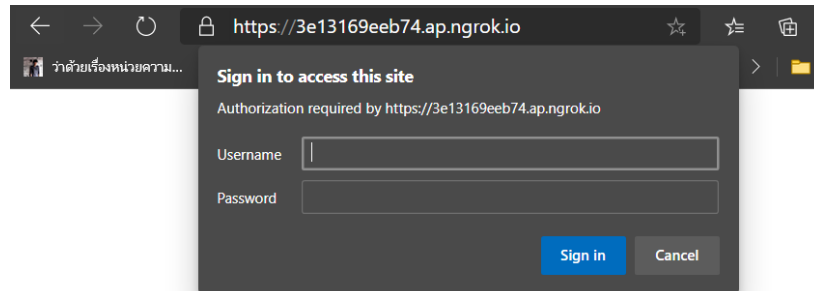
3.7.3 การใช้งานเว็บแอปพลิเคชัน (Web Application)

การใช้งานเว็บแอปพลิเคชัน (Web Application) จะต้องทำการ login ก่อนที่จะใช้งานเพื่อเป็นการป้องกันความปลอดภัยของระบบในเบื้องต้น ดังรูปที่แสดงด้านล่าง



รูปที่ 3.ป แสดงภาพ Flowchart การ login ใช้งาน Web Application


จากรูปที่ 3.ป แสดงภาพ Flowchart การ login เข้าใช้งาน Web Application โดยจะตรวจสอบความถูกต้องของ Username และ Password ถ้าข้อมูลที่กรอกถูกต้องก็จะแสดงหน้าเว็บดังรูปที่แสดงด้านล่าง





รูปที่ 3.ป แสดงภาพหน้า login ก่อนเข้าใช้งานเว็บแอปพลิเคชัน




Mushroom Web Server


 Temperature **28.10 °C**
Min 25.00 °C Max 30.00 °C


 Humidity **90.90 %**
Min 80.00 % Max 99.00 %

 Pump **ON**

 Fan **OFF**

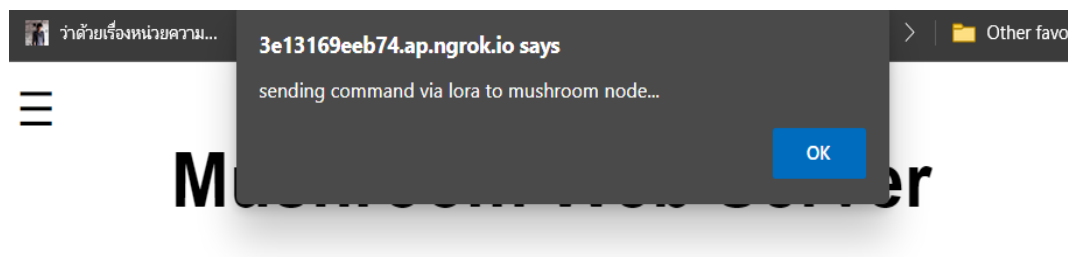
 Mode **AUTO**

 change mode

 request update

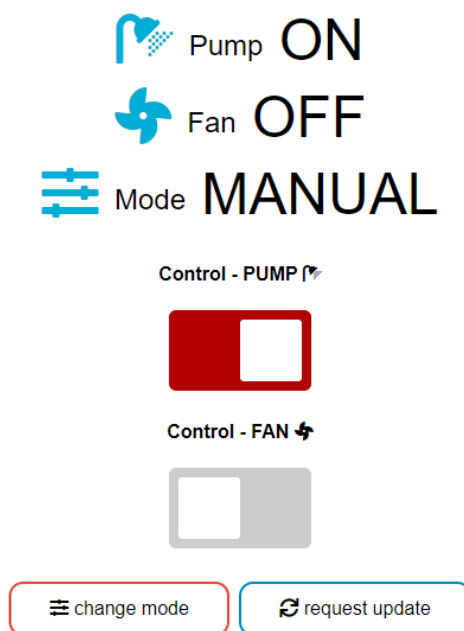
รูปที่ 3.ป แสดงภาพหน้าแรกของเว็บแอปพลิเคชัน

จากรูปที่ 3.ป แสดงภาพหน้าแรกของเว็บแอปพลิเคชันจะประกอบด้วยส่วนของการแสดงค่าสถานะต่างๆ และส่วนของการควบคุมการทำงานของระบบ จะเห็นว่าปัจจุบันหน้าเว็บแอปพลิเคชันจะแสดงโหมดการทำงานเป็นโหมด AUTO หากต้องการควบคุมการทำงานของปั๊มและพัดลมจะต้องกดปุ่มเปลี่ยนโหมดเพื่อเปลี่ยนโหมดการทำงานเป็นโหมด MANUAL ก่อน รูปด้านล่างแสดงการแจ้งเตือนเมื่อกดปุ่มเปลี่ยนโหมด (change mode)



รูปที่ 3.ป แสดงภาพการแจ้งเตือนเมื่อกดปุ่มการควบคุมต่างๆ

จากรูปที่ 3.ป เมื่อกดปุ่มควบคุมการทำงานของระบบจะแจ้งเตือนว่ากำลังส่งคำสั่งผ่าน Lora ไปยัง Mushroom Node เช่น กดปุ่มเปลี่ยนโหมดการทำงาน (change mode) กดปุ่มขออัปเดตข้อมูล (request update) กดปุ่ม เปิด/ปิด ปั๊มและพัดลม



รูปที่ 3.ป แสดงภาพหน้าเว็บเมื่อโหมดการทำงานเป็นโหมด MANUAL

จากรูปที่ 3.ป เมื่อเปลี่ยนโหมดการทำงานจาก AUTO เป็น MANUAL สำเร็จแล้วหน้าเว็บจะแสดงปั๊ม เปิด/ปิด ป้อนและพัลส์ จากรูปจะเห็นว่าสถานะการทำงานของปั๊มกำลังทำงาน (ON) และปั๊มควบคุมจะเปลี่ยนแปลงตามสถานะการทำงานของปั๊ม

เมื่อกดที่แถบด้านซ้ายบนของหน้าเว็บจะแสดงเมนูนำทาง (side navigation menu) เลื่อนออกมาจากด้านซ้ายบนของหน้าเว็บ เมนูนำทาง (side navigation menu) จะประกอบด้วยส่วนต่างๆดังนี้

- Data Logger คือส่วนที่แสดงข้อมูลที่อยู่ในฐานข้อมูลในรูปแบบตาราง
- Chart คือส่วนที่แสดงอุณหภูมิและความชื้นในรูปแบบของ chart
- phpMyAdmin คือส่วนที่ใช้ในการจัดการฐานข้อมูล
- Setting คือส่วนที่ใช้กำหนดค่า อุณหภูมิต่ำสุด อุณหภูมิสูงสุด ความชื้นต่ำสุด ความชื้นสูงสุด
- Logout คือส่วนที่ใช้ในการออกจากระบบ

รูปด้านล่างแสดงภาพเมนูนำทาง (side navigation menu) เมื่อกดที่แถบด้านซ้ายบนของหน้าเว็บ



รูปที่ 3.ป แสดงภาพเมนูนำทาง (side navigation menu)

243	30.20	82.20	20	30	80	85	AUTO	ON	ON	2020-08-22 01:00:00
244	29.90	83.30	20	30	80	85	AUTO	ON	OFF	2020-08-22 02:00:00
245	30.60	82.20	20	30	80	85	AUTO	ON	ON	2020-08-22 03:00:00
246	30.60	82.30	20	30	80	85	AUTO	ON	ON	2020-08-22 04:00:00
247	29.40	86.90	20	30	80	85	AUTO	OFF	OFF	2020-08-22 05:00:00
248	29.70	83.80	20	30	80	85	AUTO	ON	OFF	2020-09-07 19:00:00
249	29.80	84.40	20	30	80	85	AUTO	ON	OFF	2020-09-07 20:00:00
250	30.20	83.60	20	30	80	85	AUTO	ON	ON	2020-09-07 21:00:00
251	30.10	83.80	20	30	80	85	AUTO	ON	ON	2020-09-07 22:00:00
252	30.20	84.00	20	30	80	85	AUTO	ON	ON	2020-09-07 23:00:00
253	30.00	83.90	20	30	80	85	AUTO	ON	OFF	2020-09-08 00:00:00
254	29.70	84.00	20	29	80	85	AUTO	ON	ON	2020-09-08 01:00:00
255	29.80	83.70	20	29	80	85	AUTO	ON	ON	2020-09-08 02:00:01
256	31.70	79.20	20	29	80	85	AUTO	ON	ON	2020-09-08 14:00:00
257	32.10	80.00	20	29	80	85	AUTO	ON	ON	2020-09-08 18:00:00
258	28.20	90.50	25	30	80	99	MANUAL	OFF	OFF	2020-09-15 02:00:00
259	28.00	90.90	25	30	80	99	MANUAL	OFF	OFF	2020-09-15 03:00:00

รูปที่ 3.ป แสดงภาพตัวอย่างข้อมูลที่แสดงบนหน้า Data Logger

จากรูปที่ 3.ป หน้า Data Logger ค่าเริ่มต้นจะแสดงข้อมูลทั้งหมดที่มีอยู่ในฐานข้อมูล หรือสามารถเลือกแสดงข้อมูลได้เช่น เลือกแสดงข้อมูลในวันที่ที่กำหนด เลือกแสดงข้อมูลระหว่างวันที่ถึงวันที่ ดังรูปที่ 3.ป และ 3.ป

Search Date

2020-09-15

Search

Start Date

End Date

Search

ID	Temperature	Humidity	Temp Limit Min	Temp Limit Max	Humi Limit Min	Humi Limit Max	Ctrl Mode	Pump State	Fan State	Timestamp
258	28.20	90.50	25	30	80	99	MANUAL	OFF	OFF	2020-09-15 02:00:00
259	28.00	90.90	25	30	80	99	MANUAL	OFF	OFF	2020-09-15 03:00:00

รูปที่ 3.ป แสดงภาพเลือกแสดงข้อมูลในวันที่ที่กำหนด

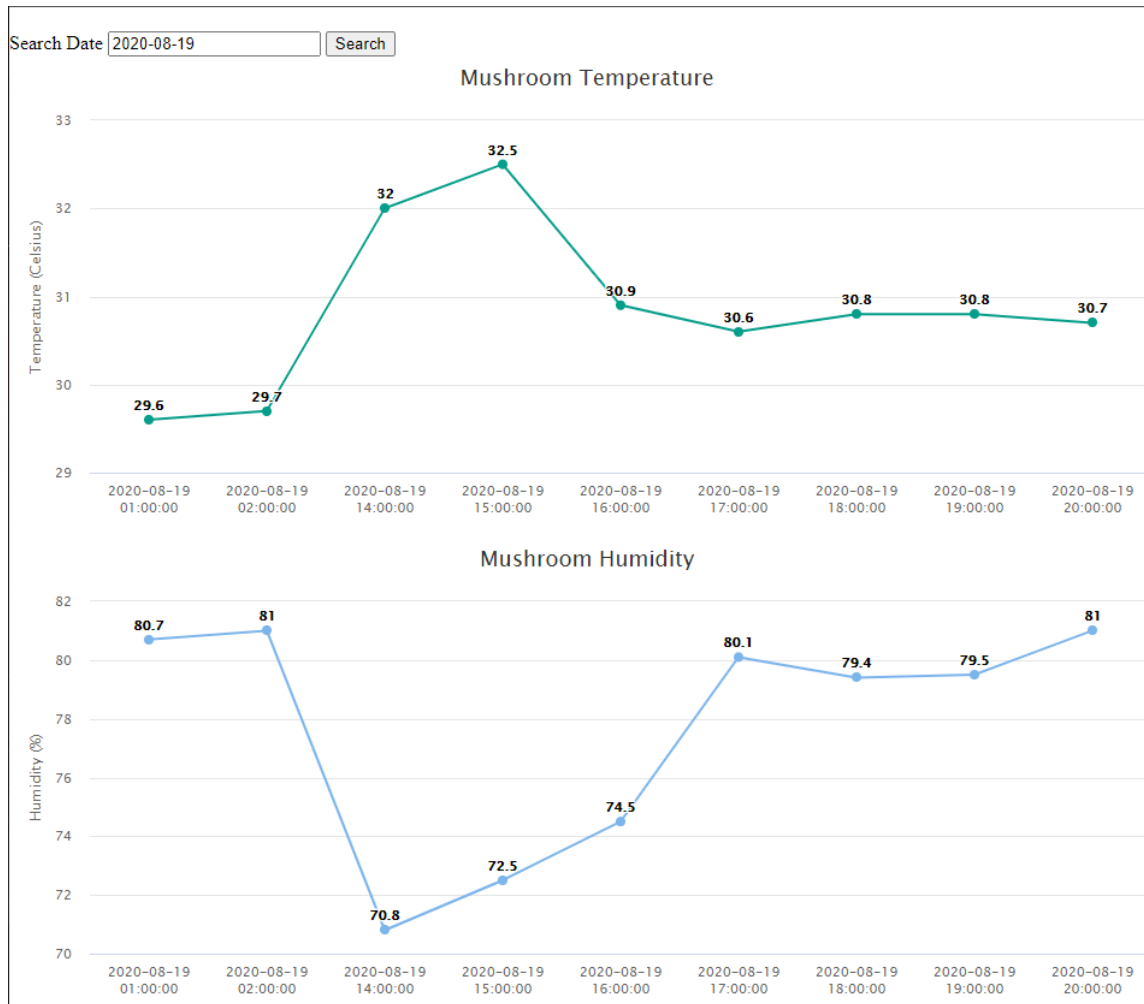
Search Date <input type="text"/>			<input type="button" value="Search"/>		Start Date <input type="text" value="2020-09-07"/>			End Date <input type="text" value="2020-09-15"/>		
ID	Temperature	Humidity	Temp Limit Min	Temp Limit Max	Humi Limit Min	Humi Limit Max	Ctrl Mode	Pump State	Fan State	Timestamp
248	29.70	83.80	20	30	80	85	AUTO	ON	OFF	2020-09-07 19:00:00
249	29.80	84.40	20	30	80	85	AUTO	ON	OFF	2020-09-07 20:00:00
250	30.20	83.60	20	30	80	85	AUTO	ON	ON	2020-09-07 21:00:00
251	30.10	83.80	20	30	80	85	AUTO	ON	ON	2020-09-07 22:00:00
252	30.20	84.00	20	30	80	85	AUTO	ON	ON	2020-09-07 23:00:00
253	30.00	83.90	20	30	80	85	AUTO	ON	OFF	2020-09-08 00:00:00
254	29.70	84.00	20	29	80	85	AUTO	ON	ON	2020-09-08 01:00:00
255	29.80	83.70	20	29	80	85	AUTO	ON	ON	2020-09-08 02:00:01
256	31.70	79.20	20	29	80	85	AUTO	ON	ON	2020-09-08 14:00:00
257	32.10	80.00	20	29	80	85	AUTO	ON	ON	2020-09-08 18:00:00
258	28.20	90.50	25	30	80	99	MANUAL	OFF	OFF	2020-09-15 02:00:00
259	28.00	90.90	25	30	80	99	MANUAL	OFF	OFF	2020-09-15 03:00:00

รูปที่ 3.ป แสดงภาพเลือกแสดงข้อมูลระหว่างวันที่ 2020-09-07 ถึงวันที่ 2020-09-15

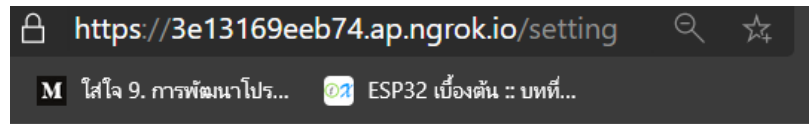


รูปที่ 3.ป แสดงภาพค่าอุณหภูมิและความชื้นที่แสดงในรูปแบบกราฟ

จากรูปที่ 3.ป หน้า Chart ค่าเริ่มต้นจะเป็นค่าอุณหภูมิและความชื้นในวันที่ปัจจุบันและสามารถดูข้อมูลในวันที่ที่ต้องการได้ดังรูปที่ 3.ป



รูปที่ 3.ป แสดงภาพข้อมูลอุณหภูมิและความชื้นแสดงตามวันที่ 2020-08-19



รูปที่ 3.ป แสดงภาพหน้า setting

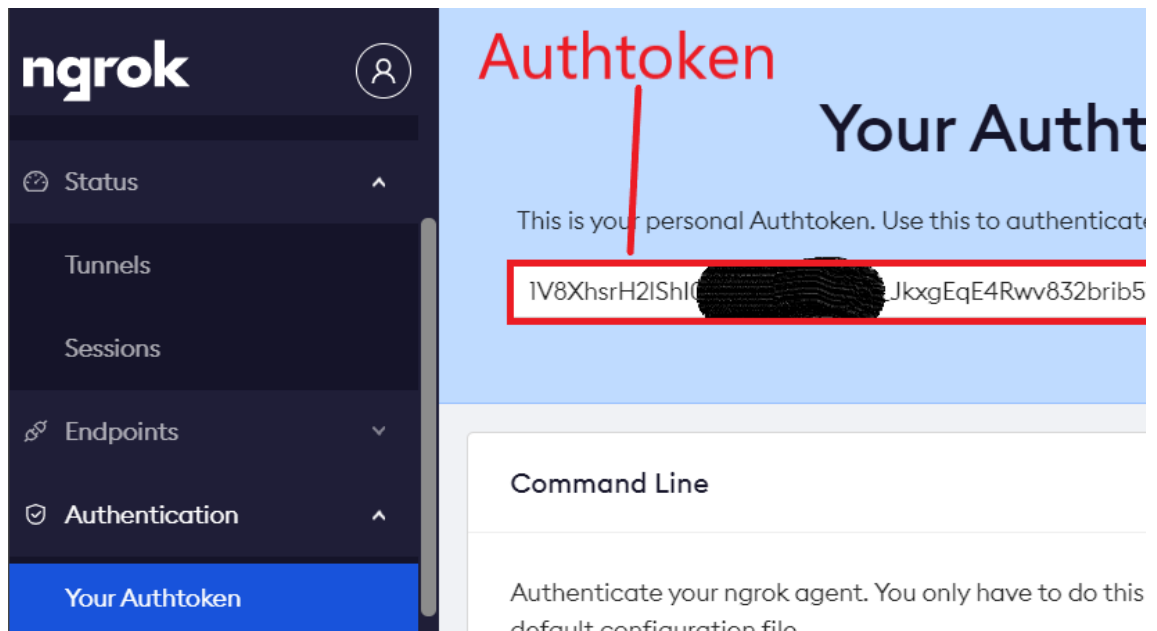
จากรูปที่ 3.ป หน้า setting เป็นหน้าที่ใช้ในการกำหนดค่าที่ใช้ในการควบคุมอุณหภูมิและความชื้น เช่น ค่าอุณหภูมิต่ำสุด ค่าอุณหภูมิสูงสุด ค่าความชื้นต่ำสุด ค่าความชื้นสูงสุด

3.7.4 การใช้โปรแกรม Ngrok ในการทำ Port forwarding

โปรแกรม Ngrok เป็น Tool Open Source พัฒนาโดย GitHub ซึ่งอำนวยความสะดวกให้บุคคลอื่นสามารถเข้าใช้งาน Website หรือ Application ที่กำลังทำงานอยู่บนเครื่อง Localhost โดยบุคคลอื่นสามารถเข้าใช้งาน Website หรือ Application กำลังทำงานอยู่บนเครื่อง Localhost ผ่านทาง URL ของทาง Ngrok โดยที่ทาง Ngrok จะทำการสุ่มสร้าง URL ขึ้นมา และ URL ที่ได้มานั้น จะทำการเปลี่ยนไปทุกครั้งเมื่อมีการปิดหรือเปิดใช้งาน Ngrok

ในโครงงานนี้ผู้จัดทำได้ใช้โปรแกรม Ngrok ที่ติดตั้งบน Raspberry Pi ในการทำ Port forwarding โดยมีรายละเอียดดังต่อไปนี้

ขั้นตอนที่ 1 สมัครสมาชิกกับ Ngrok เพื่อนำ authtoken ที่ได้มารอกใน ngrok configuration file ให้สารณใช้งานโปรแกรมได้ตลอดเวลา



รูปที่ 3.ป แสดงภาพ Authtoken ที่ได้จากการสมัครสมาชิก

ขั้นตอนที่ 2 แก้ไขไฟล์ ngrok configuration file

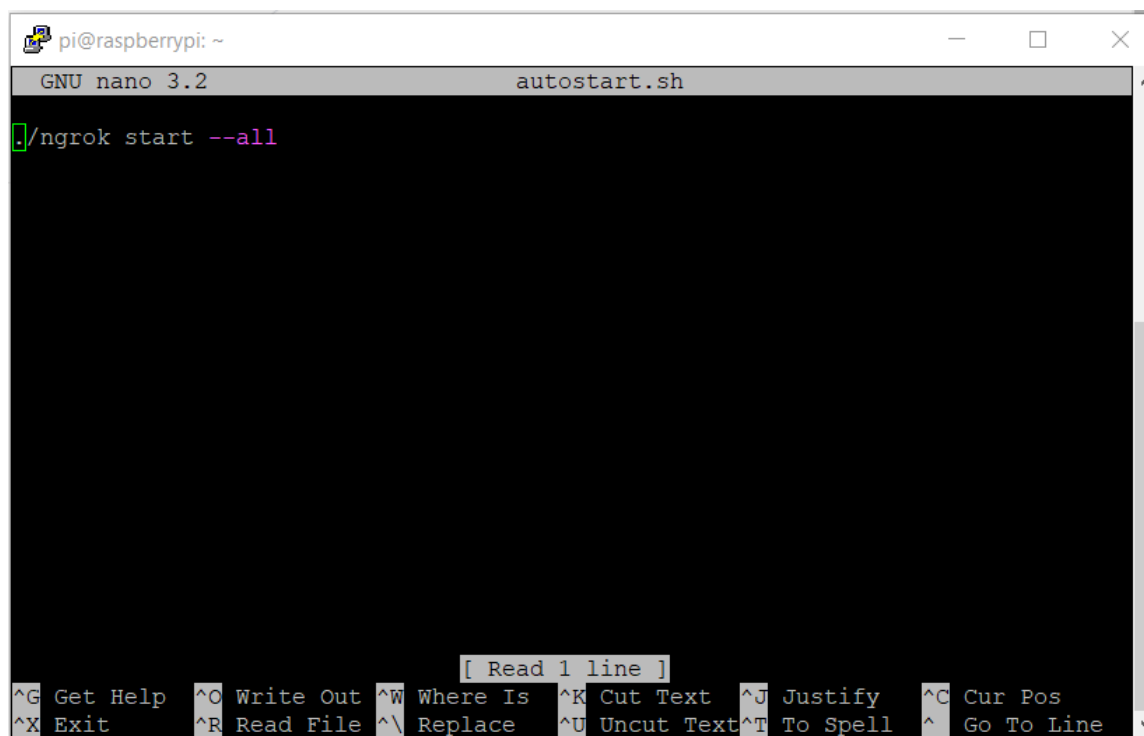
```
pi@raspberrypi: ~  
GNU nano 3.2 /home/pi/.ngrok2/ngrok.yml  
authtoken: 1V8XhsrH2l[REDACTED]uE5cSjZ_JkxgEqE4Rwv832brib51  
region: ap  
  
tunnels:  
# pi_ssh:  
# proto: tcp  
# addr: 22  
web_server:  
  proto: http  
  addr: 192.168.43.100:80  
  bind-tls: true  
database_server:  
  proto: http  
  addr: 80  
  bind-tls: false
```

รูปที่ 3.ป แสดงภาพ ไฟล์ ngrok configuration file

จากรูปที่ 3.ป แก้ไขไฟล์ ngrok configuration file นำ Authtoken ที่ได้จากการสมัครสมาชิก มากรอกใน ngrok configuration file เลือกพื้นที่ของ Ngrok Server โดยกำหนด region: ap หรือ Asia pacific เพื่อการใช้งานอย่างมีประสิทธิภาพ กำหนด IP Address ภายใน Local Area Network ที่ต้องการทำ Port forwarding มีรายละเอียดดังนี้

- การทำ Port forwarding ของ Web Server (ESP32LoRa) ต้องกำหนด IP Address ของ ESP32LoRa ตามด้วย Port เช่น addr: 192.168.43.100:80
- การทำ Port forwarding ของ Database Server (Raspberry Pi) กำหนดเฉพาะ Port เช่น addr: 80 หมายความว่าต้องการทำ Port forwarding IP Address ของเครื่องที่ติดตั้งโปรแกรม Ngrok ซึ่งก็คือ Raspberry Pi

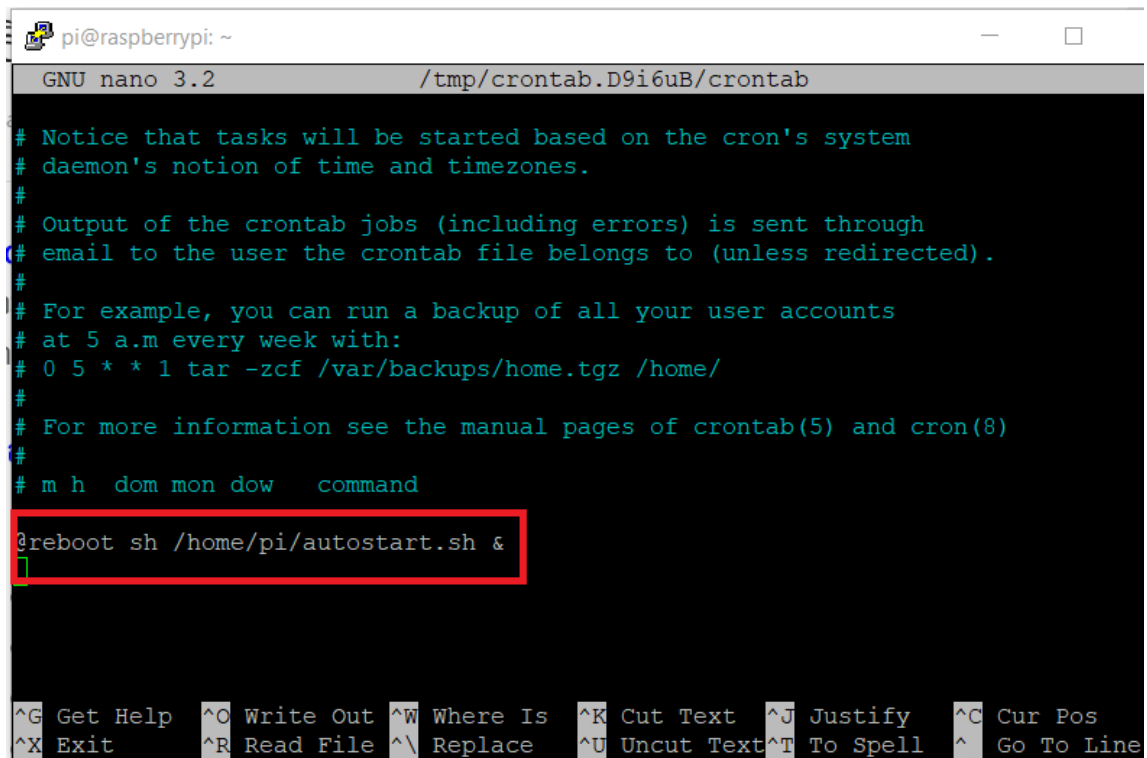
ขั้นตอนที่ 3 สร้างไฟล์ autostart.sh เพื่อใช้ในการ run command `./ngrok start --all` `./ngrok start --all` เป็นคำสั่งให้โปรแกรม Ngrok เริ่มการทำงานจาก ngrok configuration file



```
pi@raspberrypi: ~  
GNU nano 3.2 autostart.sh  
./ngrok start --all  
[ Read 1 line ]  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^ Go To Line
```

รูปที่ 3.ป แสดงภาพไฟล์ autostart.sh

ขั้นตอนที่ 4 ใช้ Crontab ในการ run คำสั่งที่อยู่ในไฟล์ autostart.sh Crontab คือคำสั่งที่
จะทำงานตามเวลาที่กำหนด



```
pi@raspberrypi: ~  
GNU nano 3.2 /tmp/crontab.D9i6uB/crontab  
  
# Notice that tasks will be started based on the cron's system  
# daemon's notion of time and timezones.  
#  
# Output of the crontab jobs (including errors) is sent through  
# email to the user the crontab file belongs to (unless redirected).  
#  
# For example, you can run a backup of all your user accounts  
# at 5 a.m every week with:  
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/  
#  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h dom mon dow   command  
reboot sh /home/pi/autostart.sh &  
  
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos  
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

รูปที่ 3.ป แสดงภาพการใช้ Crontab ในการ run คำสั่ง

จากรูปที่ 3.ป กรอบสีแดงคือส่วนที่เพิ่มเข้าไปใน Crontab โดยมีรายละเอียดคือการสั่งให้ Raspberry Pi run คำสั่งที่อยู่ในไฟล์ autostart.sh เมื่อ reboot หรือ เมื่อ Raspberry Pi เริ่มทำงาน

จากทั้ง 4 ขั้นตอนที่กล่าวมาผลลัพธ์ที่ได้คือโปรแกรม Ngrok ที่บน Raspberry Pi จะทำ Port forwarding โดยอัตโนมัติเมื่อ Raspberry Pi เริ่มทำงาน เพื่อช่วยอำนวยความสะดวกโดยไม่ต้องสั่งให้โปรแกรม Ngork ทำงาน

Region ▾	URL ▾	Client IP ▾	Established ▾
▶ AP	https://9004ba575174.ap.ngrok.io	2001:44c8:42bf:d90:bec1:145b:81ae:a2fb	3h ago
▶ AP	http://de8a7b18c985.ap.ngrok.io	2001:44c8:42bf:d90:bec1:145b:81ae:a2fb	3h ago

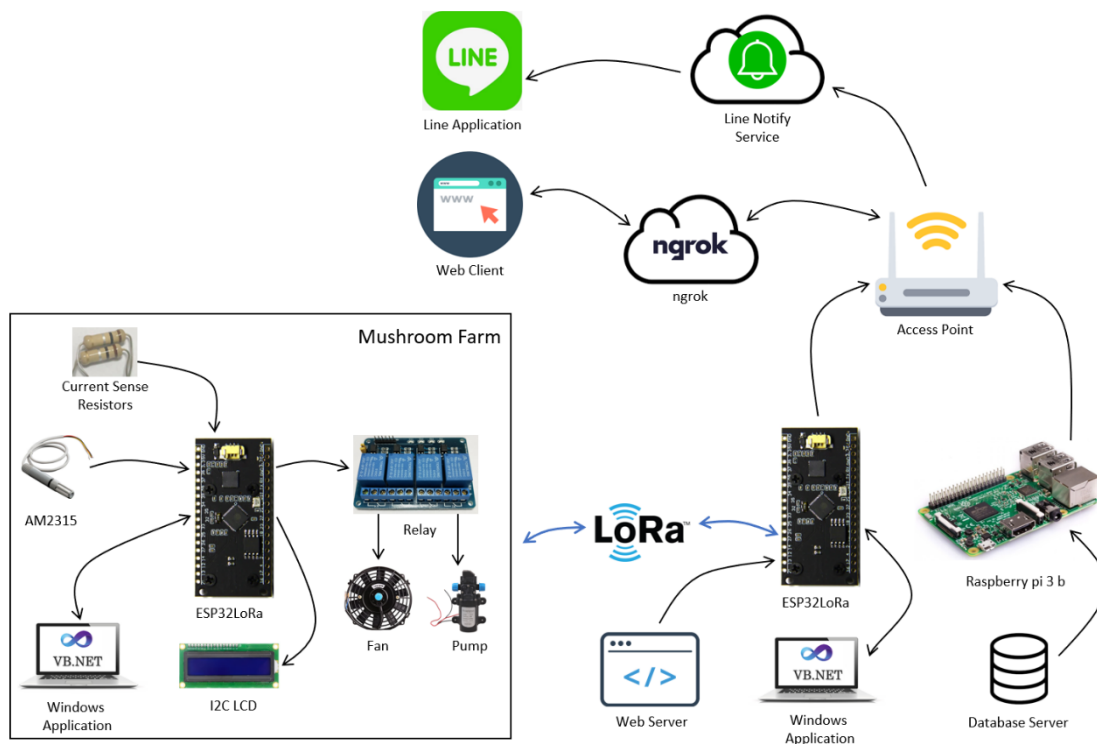
รูปที่ 3.ป แสดงภาพผลลัพธ์ที่ได้จากการทำ Port forwarding

รูปที่ 3.ป คือผลลัพธ์ที่ได้จากการทำ Port forwarding โดย url แรกคือ Public url ของ Web Server และ url ที่สองคือ Public url ของ Database Server

3.8 การออกแบบและสร้างวินโดว์แอปพลิเคชัน

ในโครงการนี้ผู้จัดทำได้มีวัตถุประสงค์ในการนำวินโดว์แอปพลิเคชัน (Windows Application) มาประยุกต์ใช้ภายในโครงการ ใช้ภาษา Visual Basic ของ .NET Core ในการสร้างวินโดว์แอปพลิเคชัน (Windows Application)

วินโดว์แอปพลิเคชัน (Windows Application) สามารถใช้งานได้กับ ESP32LoRa ได้ทั้งสองตัว ดังรูปที่ 3.ป โดยสื่อสารกันผ่าน Serial Port



รูปที่ 3.ป แสดงภาพตัวอย่างการใช้งานวินโดวแอปพลิเคชัน (Windows Application)

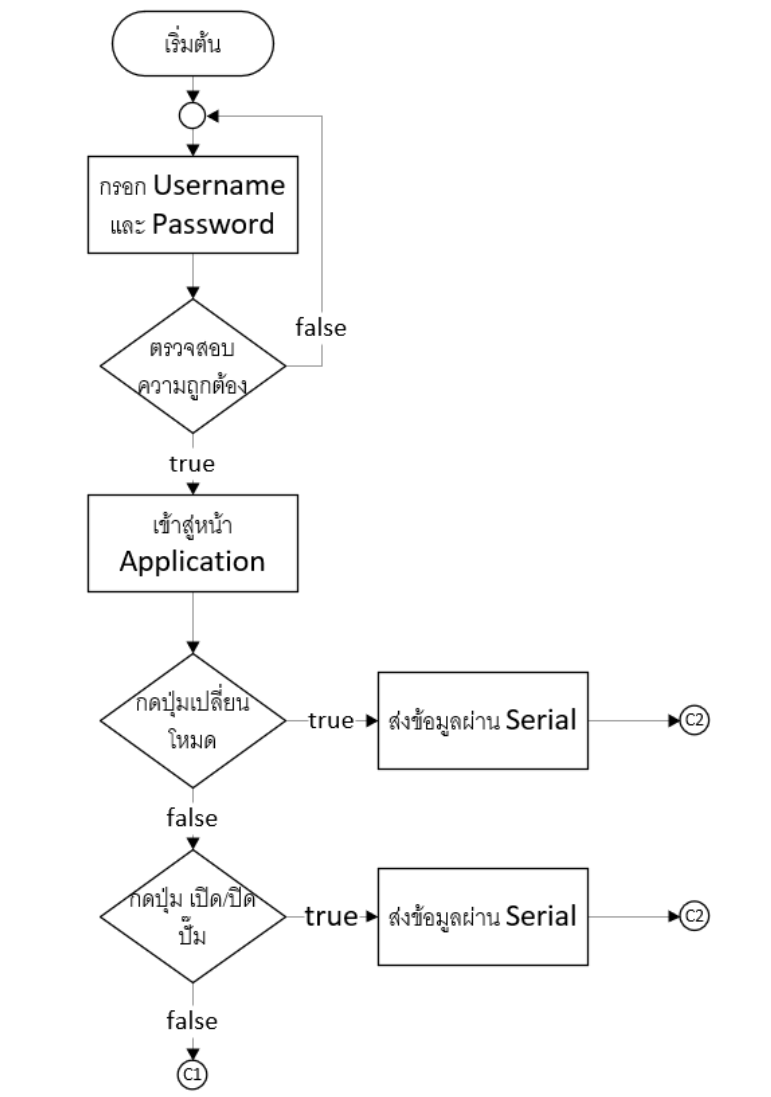
จากรูปที่ 3.ป จะเห็นว่าวินโดวแอปพลิเคชัน (Windows Application) สามารถใช้งานได้กับ ESP32LoRa (Mushroom Node) และ ESP32LoRa (STA Node) โดยใช้งานผ่านทาง Serial Port

หากใช้งานกับวินโดวแอปพลิเคชัน (Windows Application) กับ ESP32LoRa (Mushroom Node) เมื่อควบคุมการทำงาน เช่น กดปุ่มเปลี่ยนโหมดการทำงาน วินโดวแอปพลิเคชัน (Windows Application) จะส่งข้อมูลผ่าน Serial Port มายัง ESP32LoRa จากนั้น ESP32LoRa จะนำข้อมูลที่ได้มาประมวลผลตามข้อมูลที่ได้

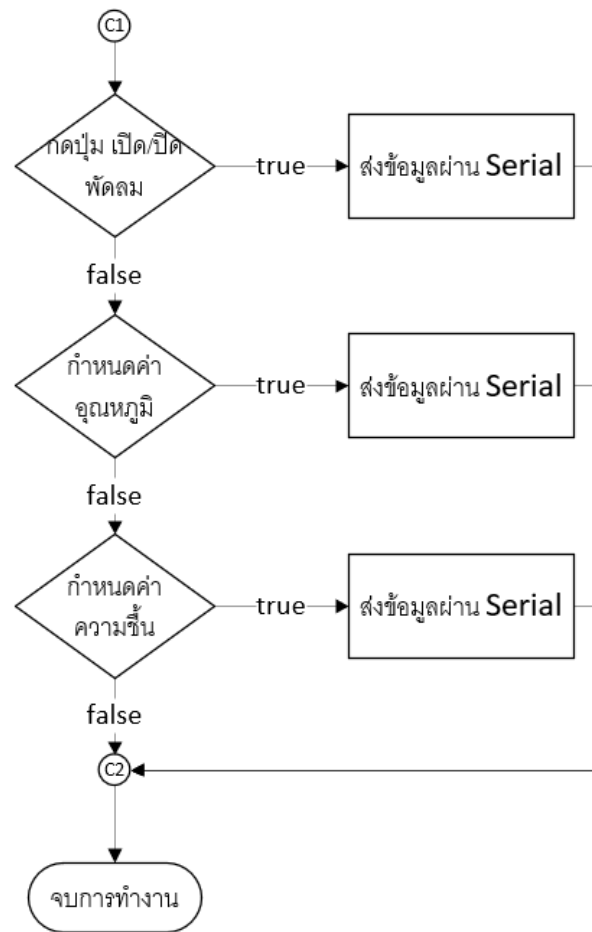
หากใช้งานกับวินโดวแอปพลิเคชัน (Windows Application) กับ ESP32LoRa (STA Node) เมื่อควบคุมการทำงาน เช่น กดปุ่มเปลี่ยนโหมดการทำงาน วินโดวแอปพลิเคชัน (Windows Application) จะส่งข้อมูลผ่าน Serial Port มายัง ESP32LoRa จากนั้น ESP32LoRa จะส่งข้อมูลที่ได้จาก Serial Port ผ่าน LoRa ไปยัง ESP32LoRa (Mushroom Node) จากนั้นจะนำข้อมูลที่ได้มาประมวลผล

3.8.1 การทำงานของวินโดว์แอปพลิเคชัน (Windows Application)

ขั้นตอนการทำงานของวินโดว์แอปพลิเคชัน (Windows Application) มีรายละเอียดการทำงานดังรูปที่แสดงด้านล่าง

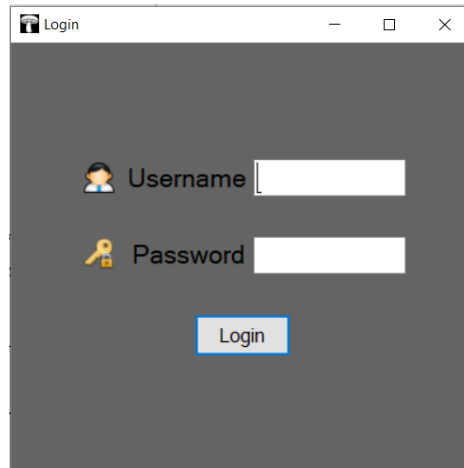


รูปที่ 3.ป แสดงภาพ Flowchart การทำงานของวินโดว์แอปพลิเคชัน (1)



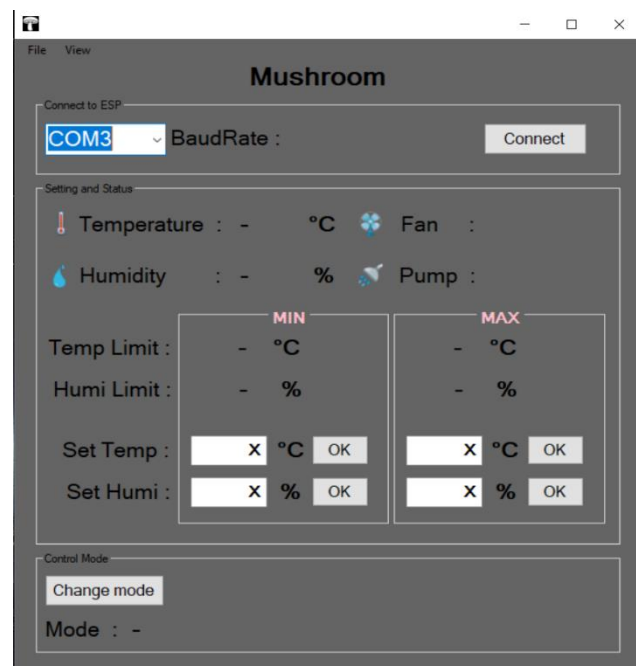
รูปที่ 3.ป แสดงภาพ Flowchart การทำงานของวินโดว์แอปพลิเคชัน (2)

จากรูปที่ 3.ป และ 3.ป แสดงภาพ Flowchart การทำงานของวินโดว์แอปพลิเคชันโดยมีรายละเอียดดังต่อไปนี้ เมื่อแอปพลิเคชันเริ่มต้นทำงานจะให้กรอก Username และ Password ก่อนเพื่อป้องกันความปลอดภัย ดังรูปที่ 3.ป



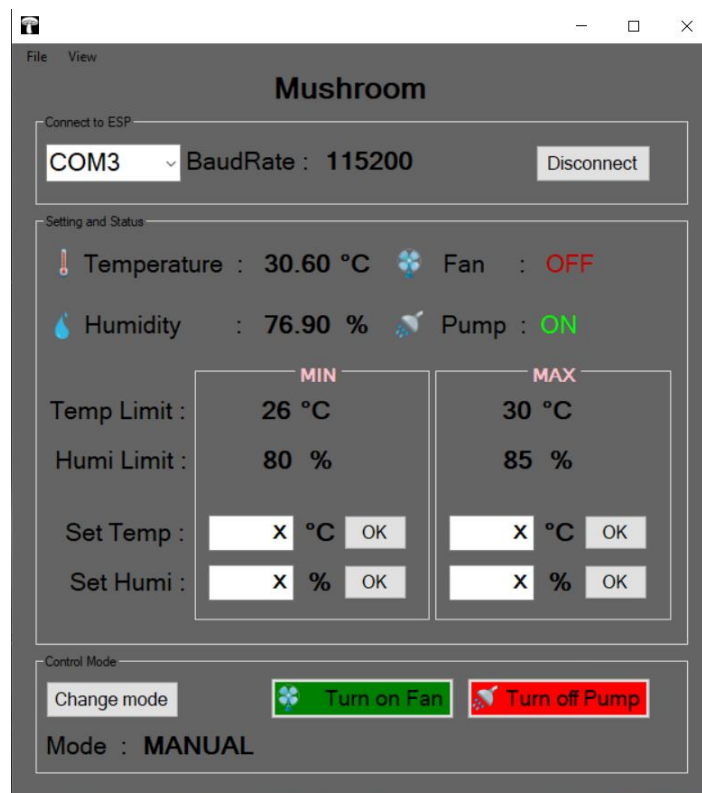
รูปที่ 3.ป แสดงภาพหน้า Login ของวินโดว์แอปพลิเคชัน

เมื่อทำการ Login สำเร็จจะแสดงหน้าแรกของวินโดว์แอปพลิเคชันดังรูปที่แสดงด้านล่าง



รูปที่ 3.ป แสดงภาพวินโดว์แอปพลิเคชันเมื่อยังไม่ได้เชื่อมต่อกับ Serial Port

จากรูปที่ 3.ป แสดงภาพของวินโดว์แอปพลิเคชันเมื่อยังไม่เชื่อมต่อกับ Serial Port เมื่อทำการเชื่อมต่อแล้วจะแสดงค่าสถานะต่างๆ ของโรงเรือนดังรูปที่แสดงด้านล่าง



รูปที่ 3.ป แสดงภาพวินโดว์แอปพลิเคชันเมื่อเชื่อมต่อกับ Serial Port แล้ว

จากรูปที่ 3.ป แสดงภาพวินโดว์แอปพลิเคชันเมื่อเชื่อมต่อกับ Serial Port แล้ว ESP32LoRa จะส่งข้อมูลผ่าน Serial Port มายังวินโดว์แอปพลิเคชันโดยส่งฟังก์ชัน Serial.print() ตัวอย่างข้อมูลคือ T30.50H78.703075M0P1F02685 จากนั้นวินโดว์แอปพลิเคชันจะนำข้อมูลที่ได้ประมวลผลดังรูปที่ 3.ป แล้วนำมาแสดงบนหน้าวินโดว์แอปพลิเคชัน

```

Private Sub SerialPort1_DataReceived(sender As Object, e As SerialDataReceivedEventArgs)
    Dim line As String = SerialPort1.ReadLine() 'serial value

    ' Label16.Text = line 'debug

    Try
        temp = line.Substring(1, 5) 'temp ค่าอุณหภูมิ
        Label17.Text = temp 'show temp

        humi = line.Substring(7, 5) 'humi ค่าความชื้น
        Label18.Text = humi 'show humi

        set_temp_max = line.Substring(12, 2) 'setTemp ค่าอุณหภูมิที่กำหนดส่งมาจาก ESP
        Label37.Text = set_temp_max

        set_humi_min = line.Substring(14, 2) 'setHumi ค่าความชื้นที่ส่งมาจาก ESP
        Label22.Text = set_humi_min 'show setHumi

        ctrlMode = line.Substring(17, 1) 'โหมดการทำงานของระบบ
        pumpState = line.Substring(19, 1) 'สถานะการทำงานของ pump
        fanState = line.Substring(21, 1) 'สถานะการทำงานของ fan

        set_temp_min = line.Substring(22, 2)
        Label21.Text = set_temp_min

        set_humi_max = line.Substring(24, 2)
        Label36.Text = set_humi_max

    Catch ex As Exception
        'MessageBox.Show("Error, Serial data.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub

```

รูปที่ 3.ป แสดงภาพส่วนของโปรแกรม Visual Basic ที่จัดการกับข้อมูลที่ส่งมาจาก Serial Port

ขั้นตอนต่อมาหลังจากที่จัดการกับข้อมูลที่ส่งมาจาก Serial Port แล้วจะตรวจสอบว่ามีการควบคุมการทำงานของระบบหรือไม่โดยมีรายละเอียดดังต่อไปนี้

เมื่อกดปุ่มเปลี่ยนโหมดการทำงาน (Change mode) วินโดว์แอปพลิเคชันจะส่ง Keyword “M” ผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("M") เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.ป

```

Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click

    Try
        SerialPort1.Write("M")
    Catch ex As Exception
        MessageBox.Show("The serial port is closed.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try

End Sub

```

รูปที่ 3.ป แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกดปุ่มเปลี่ยนโหมด

เมื่อกดปุ่มเปิดปั๊ม (Turn on Pump) หรือ ปุ่มปิดปั๊ม (Turn off Pump) วินโดว์แอปพลิเคชันจะส่ง Keyword “P” ผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("P") เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.ป

```
Private Sub Button7_Click(sender As Object, e As EventArgs) Handles Button7.Click
    SerialPort1.Write("P")
End Sub
```

รูปที่ 3.ป แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกดปุ่ม เปิด/ปิด ปั๊ม

เมื่อกดปุ่มเปิดพัดลม (Turn on Fan) หรือ ปุ่มปิดพัดลม (Turn off Fan) วินโดว์แอปพลิเคชันจะส่ง Keyword “F” ผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("F") เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.ป

```
Private Sub Button6_Click(sender As Object, e As EventArgs) Handles Button6.Click
    SerialPort1.Write("F")
End Sub
```

รูปที่ 3.ป แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกดปุ่ม เปิด/ปิด พัดลม

เมื่อกำหนดค่าอุณหภูมิต่ำสุดวินโดว์แอปพลิเคชันจะส่ง Keyword “Y” + ข้อมูลที่กรอก ผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("Y" + TextBox1.Text) เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.ป

```
If TextBox1.TextLength = 2 Then
    SerialPort1.Write("Y" + TextBox1.Text)
    TextBox1.Clear()
Else
    MessageBox.Show("The value should be between 10-99°C.", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning)
End If
```

รูปที่ 3.ป แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกำหนดค่าอุณหภูมิต่ำสุด

เมื่อกำหนดค่าอุณหภูมิสูงสุดวินโดว์แอปพลิเคชันจะส่ง Keyword “T” + ข้อมูลที่กรอก ผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("T" + TextBox4.Text) เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.ป

```
If TextBox4.TextLength = 2 Then
    SerialPort1.Write("T" + TextBox4.Text)
    TextBox4.Clear()
Else
    MessageBox.Show("The value should be between 10-99°C.", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning)
End If
```

รูปที่ 3.ป แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกำหนดค่าอุณหภูมิสูงสุด

เมื่อกำหนดค่าความชื้นต่ำสุดวินโดว์แอปพลิเคชันจะส่ง Keyword “H” + ข้อมูลที่กรอก ผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("H" + TextBox2.Text) เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.ป

```
If TextBox2.TextLength = 2 Then
    SerialPort1.Write("H" + TextBox2.Text)
    TextBox2.Clear()
Else
    MessageBox.Show("The value should be between 10-99%", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning)
End If
```

รูปที่ 3.ป แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกำหนดค่าความชื้นต่ำสุด

เมื่อกำหนดค่าความชื้นสูงสุดวินโดว์แอปพลิเคชันจะส่ง Keyword “J” + ข้อมูลที่กรอก ผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("J" + TextBox3.Text) เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.ป

```

If TextBox3.TextLength = 2 Then
    SerialPort1.Write("J" + TextBox3.Text)
    TextBox3.Clear()
Else
    MessageBox.Show("The value should be between 10-99°C.", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning)
End If

```

รูปที่ 3.ป แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกำหนดค่าความชื้นสูงสุด

3.9 การออกแบบฐานข้อมูลและการบันทึกข้อมูล

3.9.1 การออกแบบฐานข้อมูล

ในการออกแบบฐานข้อมูลในโครงการนี้จะใช้แค่ 1 ตาราง (entity) เท่านั้นเพื่อใช้เก็บข้อมูลต่างๆที่เกี่ยวข้องเช่น ค่าอุณหภูมิ (temp) ค่าความชื้น (humi) ค่าอุณหภูมิต่ำสุด (temp_limit_min) ค่าอุณหภูมิสูงสุด (temp_limit_max) ค่าความชื้นต่ำสุด (humi_limit_min) ค่าความชื้นสูงสุด (humi_imit_max) โหมดการทำงาน (ctrl_mode) สถานะการทำงานของปั๊ม (pump_state) สถานการณ์ทำงานของพัดลม (fan_state) และเวลาที่บันทึกข้อมูล (save_time) ดังรูปที่ 3.ป โดยตารางมีชื่อว่า SensorData

#	Name	Type	Collation	Attributes	Null	Default
<input type="checkbox"/>	1 id 🗝️	int(6)		UNSIGNED	No	None
<input type="checkbox"/>	2 temp	varchar(10)	utf8mb4_general_ci		Yes	NULL
<input type="checkbox"/>	3 humi	varchar(10)	utf8mb4_general_ci		Yes	NULL
<input type="checkbox"/>	4 temp_limit_min	varchar(10)	utf8mb4_general_ci		Yes	NULL
<input type="checkbox"/>	5 temp_limit_max	varchar(10)	utf8mb4_general_ci		Yes	NULL
<input type="checkbox"/>	6 humi_limit_min	varchar(10)	utf8mb4_general_ci		Yes	NULL
<input type="checkbox"/>	7 humi_limit_max	varchar(10)	utf8mb4_general_ci		Yes	NULL
<input type="checkbox"/>	8 ctrl_mode	varchar(10)	utf8mb4_general_ci		Yes	NULL
<input type="checkbox"/>	9 pump_state	varchar(10)	utf8mb4_general_ci		Yes	NULL
<input type="checkbox"/>	10 fan_state	varchar(10)	utf8mb4_general_ci		Yes	NULL
<input type="checkbox"/>	11 save_time	timestamp			No	current_timestamp()

รูปที่ 3.ป แสดงภาพโครงสร้างของฐานข้อมูล

จากรูปที่ 3.ป โครงสร้างของฐานข้อมูลโดยมี Primary key (PK) คือ id สามารถบันทึกข้อมูลเป็นตัวเลขจำนวนเต็มแบบไม่มีเครื่องหมายไม่เกิน 6 หลัก เป็นลำดับของข้อมูลแบบเพิ่มขึ้นอัตโนมัติ (auto increment) มี save_time attribute ใช้ในการเก็บค่าวันที่บันทึกข้อมูล และ #temp #humi #temp_limit_min #temp_limit_max #humi_limit_min #humi_limit_max #ctrl_mode #pump_state #fan_state attribute เป็น attribute ที่ใช้ในการเก็บข้อมูลต่างๆภายในระบบ สามารถบันทึกข้อมูลประเภทตัวอักษรรวมกันไม่เกิน 10 ตัวอักษร

ตารางที่ 3.ป Data Dictionary SensorData

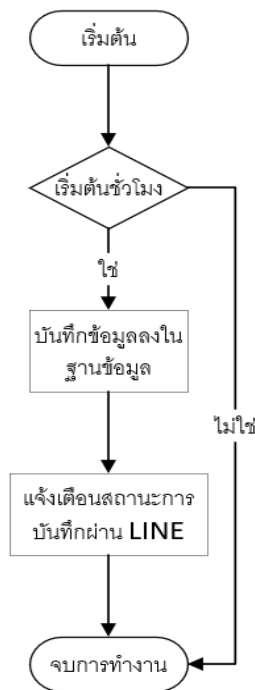
Attribute	Description	Type	Example	Key
id	ลำดับของข้อมูล	int(6)	1	PK
temp	อุณหภูมิ	varchar(10)	32.10	
humi	ความชื้น	varchar(10)	80.70	
temp_limit_min	อุณหภูมิต่ำสุด	varchar(10)	20	
temp_limit_max	อุณหภูมิสูงสุด	varchar(10)	29	
humi_limit_min	ความชื้นต่ำสุด	varchar(10)	80	
humi_limit_max	ความชื้นสูงสุด	varchar(10)	85	
ctrl_mode	โหมดการทำงาน	varchar(10)	AUTO	
pump_state	สถานะการทำงานของปั๊ม	varchar(10)	ON	
fan_state	สถานะการทำงานของพัดลม	varchar(10)	ON	
save_time	วันที่บันทึกข้อมูล	timestamp	2020-09-08 18:00:00	

id	temp	humi	temp_limit_min	temp_limit_max	humi_limit_min	humi_limit_max	ctrl_mode	pump_state	fan_state	save_time
257	32.10	80.00	20	29	80	85	AUTO	ON	ON	2020-09-08 18:00:00
256	31.70	79.20	20	29	80	85	AUTO	ON	ON	2020-09-08 14:00:00
255	29.80	83.70	20	29	80	85	AUTO	ON	ON	2020-09-08 02:00:01
254	29.70	84.00	20	29	80	85	AUTO	ON	ON	2020-09-08 01:00:00
253	30.00	83.90	20	30	80	85	AUTO	ON	OFF	2020-09-08 00:00:00
252	30.20	84.00	20	30	80	85	AUTO	ON	ON	2020-09-07 23:00:00
251	30.10	83.80	20	30	80	85	AUTO	ON	ON	2020-09-07 22:00:00
250	30.20	83.60	20	30	80	85	AUTO	ON	ON	2020-09-07 21:00:00
249	29.80	84.40	20	30	80	85	AUTO	ON	OFF	2020-09-07 20:00:00

รูปที่ 3.ป แสดงภาพตัวอย่างข้อมูลเก็บอยู่ภายในฐานข้อมูล

3.9.2 การบันทึกข้อมูลลงในฐานข้อมูล

ในการจัดทำโครงการนี้ผู้จัดทำได้ใช้ Raspberry pi 3 b เป็นฐานข้อมูลเพื่อเก็บค่าต่างๆและใช้ไมโครคอนโทรลเลอร์ ESP32 LoRa เป็นเว็บเซิร์ฟเวอร์ (Web Server) โดยจะบันทึกข้อมูลลงในฐานข้อมูลทุกๆชั่วโมง เช่น 00.00, 01.00, 02.00, 12.00, 13.00 เป็นต้น ดัง Flowchart ด้านล่าง



รูปที่ 3.ป แสดงภาพ Flowchart ขั้นตอนการบันทึกข้อมูลลงในฐานข้อมูล

```

struct tm timeinfo;
if (!getLocalTime(&timeinfo))
{
    Serial.println("Failed to obtain time");
    return;
}
char timeSec[3];
strftime(timeSec, 3, "%S", &timeinfo); //Second
char timeMin[3];
strftime(timeMin, 3, "%M", &timeinfo); //Minute

//Insert data into database every 1hr etc.(12:00:00) HH:MM:SS
const char *second_db = "00";
const char *minute_db = "00";
if (strcmp(timeSec, second_db) == 0 && strcmp(timeMin, minute_db) == 0)
{
    insertDB();
    delay(1000);
    //Serial.println("Insert into DB...");
}

```

รูปที่ 3.ป แสดงภาพการบันทึกข้อมูลลงในฐานข้อมูลทุกชั่วโมง

จากรูปที่ 3.ป จะเป็นการดึงค่าเวลาจริงมาจากเซิร์ฟเวอร์ NTP (Network Time Protocol) มาใช้ในการตรวจสอบเพื่อบันทึกข้อมูล โดยจะตรวจสอบเวลาในหลักนาที่และวินาที เช่น ถ้าเวลาในหน่วยนาที่มีค่าเท่ากับ 00 และเวลาในหน่วยวินาทีที่มีค่าเท่ากับ 00 ก็จะทำให้การบันทึกข้อมูลโดยเรียกใช้ฟังก์ชัน insertDB() ก็จะสามารถบันทึกข้อมูลทุกๆชั่วโมงได้แล้ว

การบันทึกข้อมูลจากเว็บเซิร์ฟเวอร์ (Web Server) ไปยังฐานข้อมูลที่อยู่บน Raspberry pi 3 b ทุกๆชั่วโมงโดยใช้ฟังก์ชัน insertDB() ที่เขียนด้วยภาษา C++ ดังภาพที่ 3.ป

```

void insertDB()
{
    // Your Domain name with URL path or IP address with path
    const String serverName = database_server_url + "/post-esp-data.php";
    http.begin(serverName);

    // Specify content-type header
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    String ctrlModeStr, pumpStateStr, fanStateStr;
    if (ctrlMode == 1){
        ctrlModeStr = "MANUAL";
    }else{
        ctrlModeStr = "AUTO";
    }
    if (pumpState == 1){
        pumpStateStr = "ON";
    }else{
        pumpStateStr = "OFF";
    }
    if (fanState == 1){
        fanStateStr = "ON";
    }else{
        fanStateStr = "OFF";
    }
    // Prepare your HTTP POST request data
    String httpRequestData = "api_key=" + apiKeyValue + "&temp=" + String(temp) + "&humid=" + St
    // Send HTTP POST request
    int httpResponseCode = http.POST(httpRequestData);

    NotifyLine("Saving Data into Database.\nHTTP Response code: " + String(httpResponseCode));
    // Free resources
    http.end();
}

```

รูปที่ 3.ป แสดงภาพฟังก์ชันที่ใช้บันทึกข้อมูลลงในฐานข้อมูล

จากรูปที่ 3.ป เป็นฟังก์ชันที่ทำหน้าบันทึกข้อมูลลงในฐานข้อมูลที่อยู่บน Raspberry pi 3 b โดยตัวแปรที่เป็นประเภท String ที่มีชื่อว่า serverName เป็นตัวแปรที่เก็บที่อยู่ของไฟล์ PHP ที่มีชื่อว่า post-esp-data.php ทำหน้าที่บันทึกข้อมูลลงในฐานข้อมูลที่อยู่บน Raspberry pi โดยที่อยู่นี้ได้จากการทำ port forwarding โดยใช้โปรแกรม Ngrok ตัวอย่างที่อยู่ <http://bc14e80e8393.ap.ngrok.io/post-esp-data.php>

ขั้นตอนต่อมาเป็นการจัดรูปแบบเพื่อเตรียมการก่อนการบันทึกข้อมูลผ่าน HTTP POST REQUEST ตัวแปรที่เป็นประเภท String ที่มีชื่อว่า httpRequestData เป็นตัวแปรที่เก็บข้อมูลต่างๆ ที่จะบันทึกลงในฐานข้อมูล ตัวแปรที่มีชื่อว่า httpResponseCode จะทำหน้าที่ POST ข้อมูลไปยังไฟล์ post-esp-data.php แล้วเก็บค่า http response เช่นเซิร์ฟเวอร์ตอบกลับว่า 200 OK แสดงว่าบันทึก

ข้อมูลสำเร็จ จากนั้นทำการแจ้งเตือนสถานะการบันทึกข้อมูลทาง LINE แอปพลิเคชัน โดยเรียกใช้ฟังก์ชัน NotifyLine()

```
// If you change this value, the ESP32 sketch needs to match
$api_key_value = "tPmAT5Ab3j7F9";

$api_key = $temp = $humi = $temp_limit_min = $temp_limit_max = $humi_limit_min = $humi_limit_max = $ctr

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $api_key = test_input($_POST["api_key"]);
    if($api_key == $api_key_value) {
        $temp = test_input($_POST["temp"]);
        $humi = test_input($_POST["humi"]);
        $temp_limit_min = test_input($_POST["temp_limit_min"]);
        $temp_limit_max = test_input($_POST["temp_limit_max"]);
        $humi_limit_min = test_input($_POST["humi_limit_min"]);
        $humi_limit_max = test_input($_POST["humi_limit_max"]);
        $ctrl_mode = test_input($_POST["ctrl_mode"]);
        $pump_state = test_input($_POST["pump_state"]);
        $fan_state = test_input($_POST["fan_state"]);

        // Create connection
        $conn = new mysqli($servername, $username, $password, $dbname);
        // Check connection
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

        $sql = "INSERT INTO SensorData (temp, humi, temp_limit_min, temp_limit_max, humi_limit_min, humi_limit_max, ctrl_mode, pump_state, fan_state) VALUES ('" . $temp . "', '" . $humi . "', '" . $temp_limit_min . "', '" . $temp_limit_max . "', '" . $humi_limit_min . "', '" . $humi_limit_max . "', '" . $ctrl_mode . "', '" . $pump_state . "', '" . $fan_state . "')";

        if ($conn->query($sql) === TRUE) {
            echo "New record created successfully";
        }
        else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }

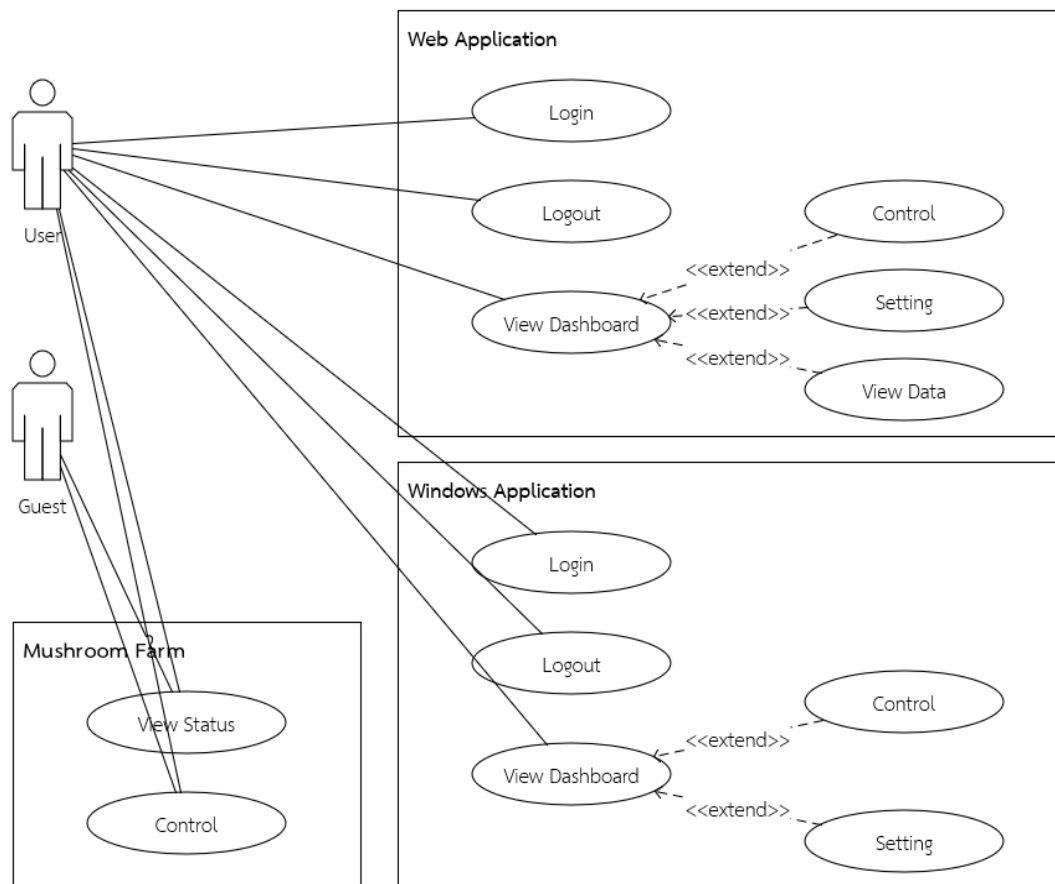
        $conn->close();
    }
    else {
        echo "Wrong API Key provided.";
    }
}
```

รูปที่ 3.ป แสดงภาพไฟล์ post-esp-data.php บางส่วน

จากรูปที่ 3.ป แสดง code บางส่วนของไฟล์ post-esp-data.php โดยมีตัวแปลที่มีชื่อว่า \$api_key_value ที่มีค่าเท่ากับ “tPmAT5Ab3j7F9” เป็นค่าที่ใช้ตรวจสอบว่าข้อมูลที่ส่งมา มาจากแหล่งที่มีที่ถูกต้อง เช่น ถ้า ESP32 LoRa ต้องการที่จะบันทึกข้อมูลลงในฐานข้อมูลจะต้องส่งค่า \$api_key_value มาด้วยผ่าน HTTP POST

ต่อมาไฟล์ post-esp-data.php จะตรวจสอบว่าถ้ามีการ POST มาจะตรวจสอบ \$api_key_value ถ้าตรงกันจะทำการเชื่อมต่อบริษัทข้อมูลแล้วบันทึกข้อมูล ถ้าไม่ตรงกันจะแสดงข้อความ "Wrong API Key provided." เพื่อเป็นการป้องกันฐานข้อมูลเบื้องต้น

3.10 การออกแบบ Use Case Diagram



รูปที่ 3.ป แสดงภาพ Use Case Diagram

รายละเอียด Use Case Diagram จากตารางที่ 3.ป ถึงตารางที่ 3.ป ด้านล่างต่อไปนี้แสดงถึงรายละเอียดการใช้งานของแต่ละ Use Case ตามรูปแบบของ UML Use Case Description

ตารางที่ 3.ป Login Use Case

Use Case Name	Login
Actors	User
Use Case Purpose	เพื่อเข้าสู่ระบบ
Pre-conditions	-
Post-conditions	เข้าสู่ระบบเพื่อดูค่าสถานะต่างๆ ควบคุมการทำงานของอุปกรณ์ กำหนดค่าสูงสุดต่ำสุดของอุณหภูมิและความชื้น
Main Course	1.กรอก Username และ Password 2.กดปุ่ม Login 3.เข้าสู่ระบบเสร็จสิ้น
Exceptions	Username หรือ Password ไม่ถูกต้อง

ตารางที่ 3.ป Logout Use Case

Use Case Name	Logout
Actors	User
Use Case Purpose	เพื่อใช้ในการออกจากระบบ
Pre-conditions	เข้าสู่ระบบมาแล้วก่อนหน้านี้
Post-conditions	ออกจากระบบสำเร็จ
Main Course	1.กดปุ่ม Logout 2.ออกจากระบบสำเร็จ
Exceptions	-

ตารางที่ 3.ป View Dashboard Use Case

Use Case Name	View Dashboard
Actors	User
Use Case Purpose	เพื่อใช้ในการแสดงสถานะต่างๆ และจัดการระบบ
Pre-conditions	เข้าสู่ระบบมาแล้วก่อนหน้า
Post-conditions	-
Main Course	แสดงค่าสถานะต่างๆ เช่น อุณหภูมิ ความชื้น สถานะการทำงานของอุปกรณ์
Exceptions	แสดงค่าผิดพลาด

ตารางที่ 3.ป Control Use Case

Use Case Name	Control
Actors	User
Use Case Purpose	เพื่อใช้ในการควบคุมการทำงานของระบบ เช่น เปลี่ยนโหมดการทำงาน ความคุมการเปิดปิดของ อุปกรณ์
Pre-conditions	ใช้ร่วมกับ View Dashboard
Post-conditions	-
Main Course	1.กดปุ่ม change mode 2.ถ้าโหมดการทำงานแบบอัตโนมัติระบบจะ ควบคุมการทำงานของอุปกรณ์โดยอัตโนมัติ เมื่อ ค่าอุณหภูมิและความชื้นต่ำกว่าหรือสูงกว่าที่ กำหนด ถ้าโหมดการทำงานแบบแมนนวลผู้ใช้จะ สามารถสั่งเปิดปิดอุปกรณ์ได้ตามต้องการ
Exceptions	เกิดความผิดพลาดภายในระบบ

ตารางที่ 3.ป Setting Use Case

Use Case Name	Setting
Actors	User
Use Case Purpose	เพื่อใช้ในการกำหนดค่าสูงสุดต่ำสุดของอุณหภูมิและความชื้น
Pre-conditions	ใช้ร่วมกับ View Dashboard
Post-conditions	-
Main Course	1.กำหนดค่าสูงสุดต่ำสุด 2.กำหนดค่าสูงสุดต่ำสุดสำเร็จ
Exceptions	เกิดความผิดพลาดภายในระบบ

ตารางที่ 3.ป View Data Use Case

Use Case Name	View Data
Actors	User
Use Case Purpose	เพื่อใช้ในการแสดงผลข้อมูลที่อยู่ใน database ในรูปแบบต่างๆ
Pre-conditions	ใช้ร่วมกับ View Dashboard
Post-conditions	-
Main Course	1.กดปุ่ม datalogger หรือ chart 2.แสดงข้อมูลในรูปแบบต่างๆ
Exceptions	ไม่สามารถติดต่อฐานข้อมูลได้

ตารางที่ 3.ป View Status Use Case

Use Case Name	View Status
Actors	User, Guest
Use Case Purpose	เพื่อใช้ในการแสดงสถานะบนจอ LCD Display ที่ โรงเพาะเห็ด
Pre-conditions	-
Post-conditions	-
Main Course	1.ดูค่าสถานะบนจอ LCD Display
Exceptions	เกิดความผิดพลาดภายในระบบ