

บทที่ 3

วิธีการดำเนินงาน

ในบทนี้จะกล่าวถึงขั้นตอนและวิธีการดำเนินงานจัดทำโครงการตลอดจนการออกแบบโครงสร้างและส่วนต่างๆ ของโครงการ ซึ่งได้รับการศึกษาข้อมูลและทฤษฎีที่เกี่ยวข้องในบทที่ 2 จึงได้นำข้อมูลที่ได้มาศึกษาและวิเคราะห์ออกแบบเพื่อเลือกใช้วัสดุอุปกรณ์ที่จะนำมาทำโครงการ ให้มีความถูกต้องเหมาะสมกับลักษณะงานและดำเนินการสร้างโครงการให้มีประสิทธิภาพและเป็นไปตามวัตถุประสงค์และขอบเขตของโครงการ

3.1 แผนการดำเนินงาน

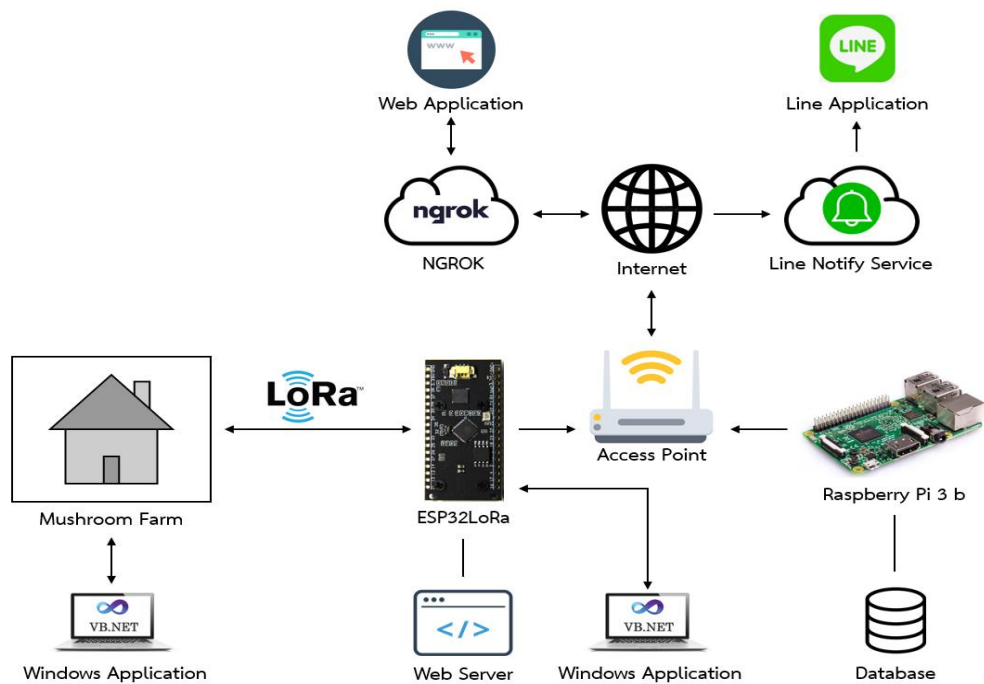
ในการดำเนินงานจัดทำโครงการโรงเพาะเห็ดอัจฉริยะ (Smart Mushroom Farm) ผู้จัดทำได้วางแผนการดำเนินงานในการจัดทำโครงการดังต่อไปนี้

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 แสดงแผนการดำเนินงาน

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 โครงสร้างระบบ



รูปที่ 3.1 แสดงภาพโครงสร้างระบบ

จากรูปที่ 3.1 แสดงภาพโครงสร้างระบบของโครงการโรงเพาะเห็ดอัจฉริยะที่ใช้เทคโนโลยีการสื่อสารไร้สายระยะไกล โดยภาพรวมของระบบประกอบด้วยโรงเรือนเพาะเห็ดที่สามารถควบคุมสภาพแวดล้อมภายในโรงเรือนได้โดยใช้ไมโครคอนโทรลเลอร์ (ESP32LoRa) เป็นตัวควบคุมระบบและทำหน้าที่ในการสื่อสารกับพื้นที่ที่มีสัญญาณอินเทอร์เน็ตผ่านลอรา (LoRa)

พื้นที่ที่มีสัญญาณอินเทอร์เน็ตจะประกอบไปด้วยไมโครคอนโทรลเลอร์ (ESP32LoRa) ทำหน้าที่สื่อสารระหว่างพื้นที่ที่มีสัญญาณอินเทอร์เน็ตและโรงเรือนและเป็น Web Server ส่วน Raspberry Pi ทำหน้าที่เป็นฐานข้อมูลเพื่อให้สามารถเรียกดูค่าสภาพแวดล้อมย้อนหลังได้ และใช้โปรแกรม ngrok ในการทำ Port forwarding เพื่อให้สามารถใช้งานเว็บแอปพลิเคชัน (Web Application) ได้จากทุกที่ที่สามารถใช้งานอินเทอร์เน็ต

เว็บแอปพลิเคชัน (Web Application) สามารถมอนิเตอร์ค่าสภาพแวดล้อมภายในโรงเรือนและควบคุมระบบ โดยผู้จัดทำได้ใช้ภาษา HTML และ JavaScript ในการสร้างเว็บแอปพลิเคชัน (Web Application) จากนั้นนำเว็บแอปพลิเคชัน (Web Application) เก็บไว้ในหน่วยความจำ (Flash Memory) ของไมโครคอนโทรลเลอร์ (ESP32LoRa) และใช้ ESPAsyncWebServer library เพื่อให้ไมโครคอนโทรลเลอร์ (ESP32LoRa) สามารถทำหน้าที่เป็น Web Server ได้

วินโดวส์แอปพลิเคชัน (Windows Application) สามารถมอนิเตอร์ค่าสภาพแวดล้อมภายในโรงเรือนและควบคุมระบบ โดยผู้จัดทำได้ใช้ภาษา Visual Basic ของ .NET Core ในการสร้างวินโดวส์แอปพลิเคชัน (Windows Application) วินโดวส์แอปพลิเคชัน (Windows Application) สามารถใช้งานได้กับไมโครคอนโทรลเลอร์ (ESP32LoRa) ทั้งสองตัวผ่านทาง Serial Port หากใช้งานวินโดวส์แอปพลิเคชัน (Windows Application) กับไมโครคอนโทรลเลอร์ (ESP32LoRa) (Mushroom Node) เมื่อควบคุมการทำงาน เช่น กดปุ่มเปลี่ยนโหมดการทำงาน วินโดวส์แอปพลิเคชัน (Windows Application) จะส่งข้อมูลผ่าน Serial Port มายังไมโครคอนโทรลเลอร์ (ESP32LoRa) จากนั้นไมโครคอนโทรลเลอร์ (ESP32LoRa) จะนำข้อมูลที่ได้มาประมวลผลเพื่อควบคุมการทำงานของระบบ หากใช้งานกับวินโดวส์แอปพลิเคชัน (Windows Application) กับ ไมโครคอนโทรลเลอร์ (ESP32LoRa) (STA Node) เมื่อควบคุมการทำงาน เช่น กดปุ่มเปลี่ยนโหมดการทำงาน วินโดวส์แอปพลิเคชัน (Windows Application) จะส่งข้อมูลผ่าน Serial Port มายังไมโครคอนโทรลเลอร์ (ESP32LoRa) จากนั้น ไมโครคอนโทรลเลอร์ (ESP32LoRa) จะส่งข้อมูลที่ได้จาก Serial Port ผ่าน LoRa ไปยังยังไมโครคอนโทรลเลอร์ (ESP32LoRa) (Mushroom Node) จากนั้นจะนำข้อมูลที่ได้มาประมวลผล

ฐานข้อมูล (Database) ในโครงการนี้ผู้จัดทำได้ใช้ Raspberry Pi เป็นฐานข้อมูล (Database) เพื่อให้สามารถดูค่าสภาพแวดล้อมย้อนหลังได้ ใช้ภาษา PHP ในการบันทึกข้อมูลลงใน

ฐานข้อมูล (Database) การนำข้อมูลจากฐานข้อมูล (Database) มาแสดงในรูปแบบของตารางและกราฟที่สามารถเลือกดูข้อมูลได้ตามช่วงเวลาที่ต้องการ

การแจ้งเตือนไปยัง Line Application (Line Notify Service) ผู้จัดทำได้ใช้ Line Notify Service ในการแจ้งเตือนสถานะต่างๆ ไปยัง Line Application เช่น สถานะ การบันทึกข้อมูล การเปลี่ยนสถานะการทำงานของอุปกรณ์ควบคุมสภาพแวดล้อม หรือการเปลี่ยนโหมดการทำงานของระบบ

ซึ่งจะอธิบายโครงสร้างระบบและการออกแบบส่วนต่างๆ โดยละเอียดตามหัวข้อดังต่อไปนี้

1. โรงเรือนเพาะเห็ด (Mushroom Farm)
2. การสื่อสารระหว่างโรงเรือนและพื้นที่ที่มีสัญญาณอินเทอร์เน็ตโดยใช้ลอรา (LoRa Communication)
3. เว็บแอปพลิเคชัน (Web Application)
4. วินโดว์แอปพลิเคชัน (Windows Application)
5. ฐานข้อมูล (Database)
6. การแจ้งเตือนไปยัง Line Application (Line Notify Service)

3.2.1 โรงเรือนเพาะเห็ด (Mushroom Farm)

3.2.2 การสื่อสารระหว่างโรงเรือนและพื้นที่ที่มีสัญญาณอินเทอร์เน็ตโดยใช้ลอรา (LoRa Communication)

3.2.3 เว็บแอปพลิเคชัน (Web Application)

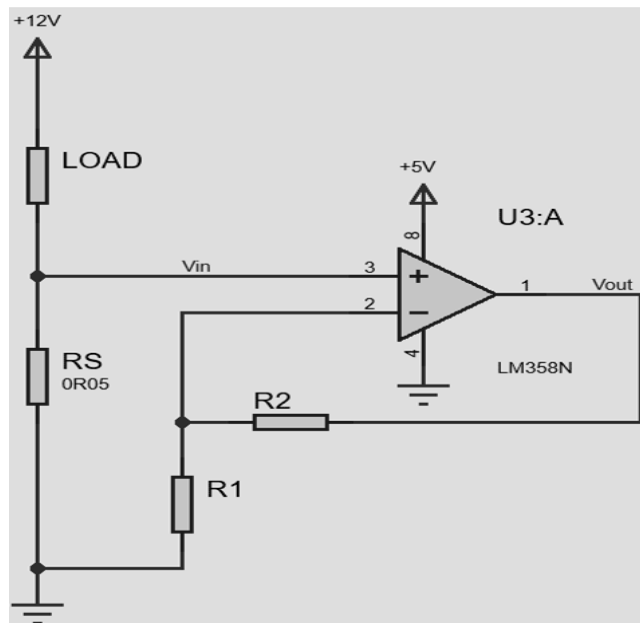
3.2.4 วินโดว์แอปพลิเคชัน (Windows Application)

3.2.5 การแจ้งเตือนไปยัง Line Application (Line Notify Service)

3.3 การออกแบบวงจรตัวต้านทานตรวจสอบกระแส

วงจรตัวต้านทานตรวจสอบกระแส มีวัตถุประสงค์เพื่อใช้ในการตรวจสอบสถานะการทำงานของ พัดลมระบายอากาศ บั้มพ่นหมอก ว่าทำงานจริงตามที่ควบคุมหรือไม่ โดยแสดงการออกแบบวงจรรูปด้านล่าง โดยมีหลักการทำงานคือนำตัวต้านทานที่มีค่าความต้านทาน 0.05 โอห์ม มาต่ออนุกรม

กับ LOAD จากนั้นนำค่าแรงดันตกคร่อมตัวต้านทานมาขยายแรงดันโดยใช้วงจรขยายแบบไม่กลับเฟส (Non-inverting Amplifier) เพื่อให้ไมโครคอนโทรลเลอร์สามารถอ่านค่าแรงดันได้



รูปที่ 3.2 แสดงภาพวงจรตัวต้านทานตรวจสอบกระแส

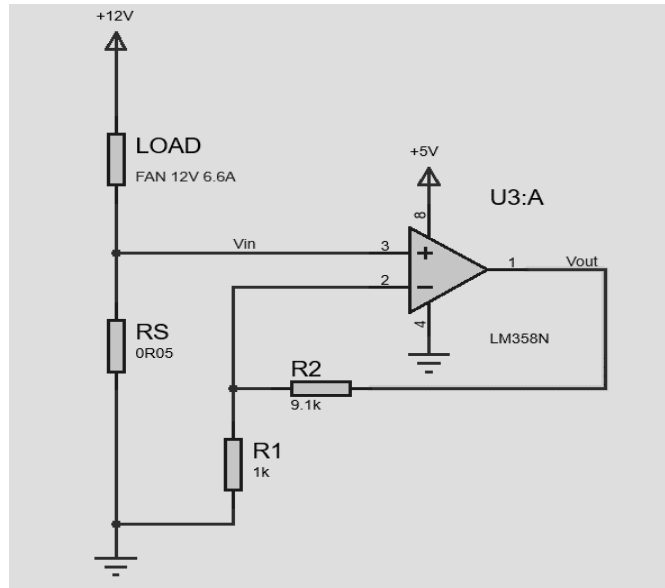
จากรูปที่ 3.2 แสดงภาพของแสดงภาพวงจรตัวต้านทานตรวจสอบกระแส โดยจะประกอบด้วย LOAD คือ พัดลมระบายอากาศ หรือ ปั๊มพ่นหมอก RS คือ ตัวต้านทานตรวจสอบกระแส (Current Sense Resistors) และส่วนของวงจรขยายแบบไม่กลับเฟส (Non-inverting amplifier) ใช้เพื่อขยายแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส

3.3.1 การคำนวณแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับพัดลมระบายอากาศ

ในโครงการนี้ผู้จัดทำใช้พัดลมระบายอากาศขนาด 12 นิ้ว 12V 6.6A ในการควบคุมสภาพแวดล้อมภายในโรงเรือนเพาะเห็ด ดังนั้นสามารถคำนวณแรงดันเอาต์พุตของวงจรตัวต้านทานตรวจสอบกระแส ได้จากสมการ

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

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



รูปที่ 3.3 แสดงภาพวงจรตัวต้านทานตรวจสอบกระแสเมื่อใช้งานกับพัดลมระบายอากาศ

จากรูปที่ 3.3 เพื่อให้ไมโครคอนโทรลเลอร์สามารถแรงดันเอาต์พุตของวงจรได้ โดยสามารถหาอัตราการขยายของวงจรได้จากสมการ

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

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

สามารถหาแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส V_{in} หรือ V_{rs} ได้จากกฎของโอห์มดังสมการ

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

ดังนั้นหาแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแสจากรูปที่ 3.33 กำหนดให้ กระแส $(I) = 6.6A$ ความต้านทานของตัวต้านทานตรวจสอบกระแส $(R_s) = 0.05\Omega$

จากสูตร

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

จะได้

$$V_{in} = IR_s$$

แทนค่า

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

ดังนั้นแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส (V_{in}) มีค่าเท่ากับ $0.33V$ ดังนั้นสามารถคำนวณแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับพัดลมระบายอากาศได้จากสมการ

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

แทนค่า

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

ดังนั้นแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับพัดลมระบายอากาศ (V_{out}) เท่ากับ $0.33V$ สามารถคำนวณกำลังวัตต์ของตัวต้านทานตรวจสอบกระแส (Prs)

จากสูตร

$$P = VI$$

จะได้

$$Prs = V_{in}I$$

แทนค่า

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

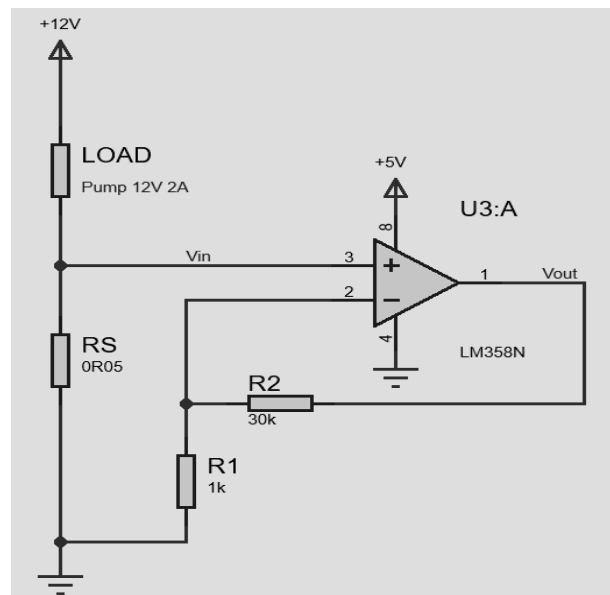
ดังนั้นกำลังวัตต์ของตัวต้านทานตรวจสอบกระแส (Prs) เท่ากับ $2.178W$

3.3.2 การคำนวณแรงดันเอาต์พุตของวงจรเมื่อใช้งานกับปั๊มพ่นหมอก

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

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

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



รูปที่ 3.4 แสดงภาพวงจรตัวต้านทานตรวจสอบกระแสเมื่อใช้งานกับปั๊มพ่นหมอก

จากรูปที่ 3.4 เพื่อให้ไมโครคอนโทรลเลอร์สามารถแรงดันเอาต์พุตของวงจรได้ โดยสามารถหาอัตราการขยายของวงจรได้จากสมการ

จากสูตร

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

จะได้

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

ดังนั้น

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

สามารถแรงดันตกคร่อมต้านทานตรวจสอบกระแส V_{in} หรือ V_{rs} ได้จากกฎของโอห์มดังสมการ

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

ดังนั้นหาแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแสจากรูปที่ 3.4 กำหนดให้ กระแส $(I) = 2A$ ความต้านทานของต้านทานตรวจสอบกระแส $(R_s) = 0.05\Omega$

จากสูตร

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

จะได้

$$V_{in} = IR_s$$

แทนค่า

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

ดังนั้นแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแส (V_{in}) มีค่าเท่ากับ $0.1V$ 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 $3.1V$ สามารถคำนวณกำลังวัตต์ของตัวต้านทานตรวจสอบกระแส (P_{rs})

จากสูตร

$$P = VI$$

จะได้

$$Prs = V_{in}I$$

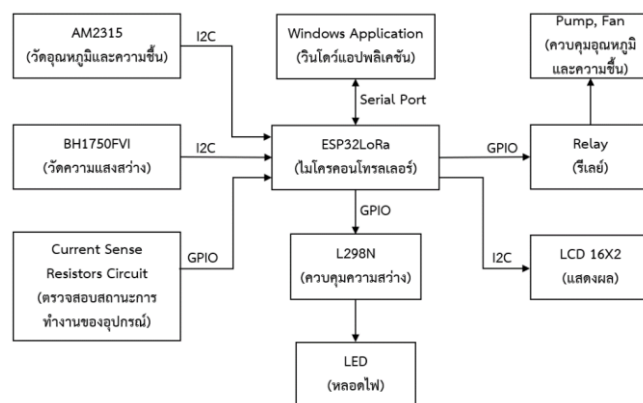
แทนค่า

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

ดังนั้นกำลังวัตต์ของตัวต้านทานตรวจสอบกระแส (Prs) เท่ากับ $0.2W$

3.4 การออกแบบฮาร์ดแวร์ระบบควบคุมสภาพแวดล้อมภายในโรงเรือนเพาะเห็ด

ในการออกแบบฮาร์ดแวร์ระบบควบคุมสภาพแวดล้อมภายในโรงเรือนเพาะเห็ด ผู้จัดทำได้ใช้ AM2315 เป็นเซ็นเซอร์วัดอุณหภูมิและความชื้นภายในโรงเรือน ใช้ BH1750FVI เป็นเซ็นเซอร์วัดความสว่างของแสงภายในโรงเรือน ใช้พัดลมในการระบายอากาศ และปั๊มพ่นหมอกในการควบคุมอุณหภูมิและความชื้นภายในโรงเรือน ใช้หลอด LED ในการให้แสงสว่างภายในโรงเรือน ใช้ตัวต้านทานตรวจสอบกระแส (Current Sense Resistors) สำหรับตรวจสอบสถานะการทำงานของพัดลมและปั๊มพ่นหมอก ใช้จอแอลซีดี (LCD) ในการแสดงผลค่าต่างๆ ภายในระบบ

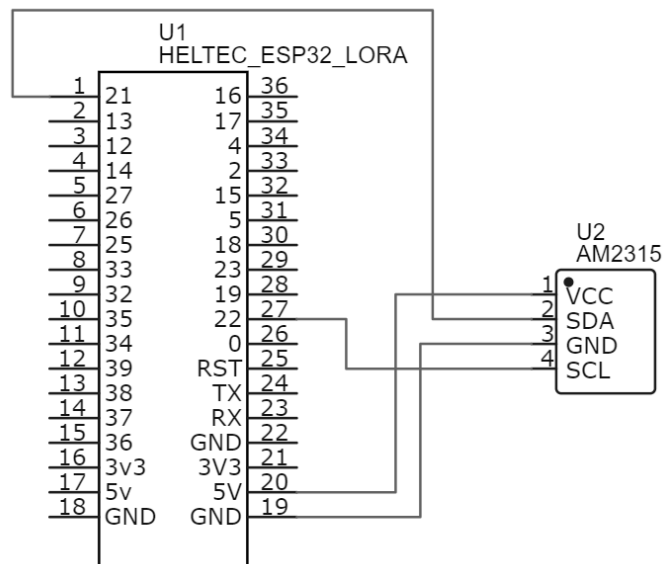


รูปที่ 3.5 แสดงภาพส่วนประกอบของฮาร์ดแวร์ระบบควบคุมสภาพแวดล้อมภายในโรงเรือน

3.4.1 การอ่านค่าอุณหภูมิและความชื้นจากเซ็นเซอร์ AM2315

AM2315 คือเซ็นเซอร์วัดค่าอุณหภูมิและความชื้นคุณภาพสูงที่มีเอาต์พุตเป็นสัญญาณดิจิทัลโดยมีการสอบเทียบสัญญาณดิจิทัลเอาต์พุตแล้ว ใช้โมดูลตรวจจับอุณหภูมิและความชื้น

คุณภาพสูงเพื่อให้แน่ใจว่ามีการวัดค่าที่แม่นยำและความทนทาน เซ็นเซอร์ประกอบด้วยตัววัดความชื้นประเภทความจุไฟฟ้า อุปกรณ์วัดอุณหภูมิที่มีความแม่นยำสูง และเชื่อมต่อด้วยไมโครคอนโทรลเลอร์ 8 บิตประสิทธิภาพสูง AM2315 สื่อสารด้วย I2C ผู้ใช้สามารถเชื่อมต่อ I2C bus ได้โดยตรง



รูปที่ 3.6 แสดงภาพ Schematic การเชื่อมต่อระหว่าง ESP32LoRa และ AM2315

ตารางที่ 3.2 แสดงการต่อขาระหว่าง 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 3.7 แสดงภาพโปรแกรมที่ใช้อ่านค่าจากเซ็นเซอร์ AM2315

จากรูปที่ 3.7 เป็นโปรแกรมภาษา C++ ที่ใช้ในการอ่านค่าจากเซ็นเซอร์ AM2315 โดยมีการเรียกใช้ไลบรารี (Library) Wire.h และ Adafruit_AM2315.h ต่อมาส่วนที่สำคัญที่สุดคือส่วนของการปลุกการทำงานของเซ็นเซอร์ดังรูปที่แสดงด้านบน หากไม่ปลุกการทำงานของเซ็นเซอร์เมื่อไม่มีการใช้งานเซ็นเซอร์เป็นเวลานานจะส่งผลให้ไม่สามารถใช้งานได้ ผลลัพธ์ที่อ่านค่าได้จากเซ็นเซอร์แสดงดังรูปที่ 3.8

```

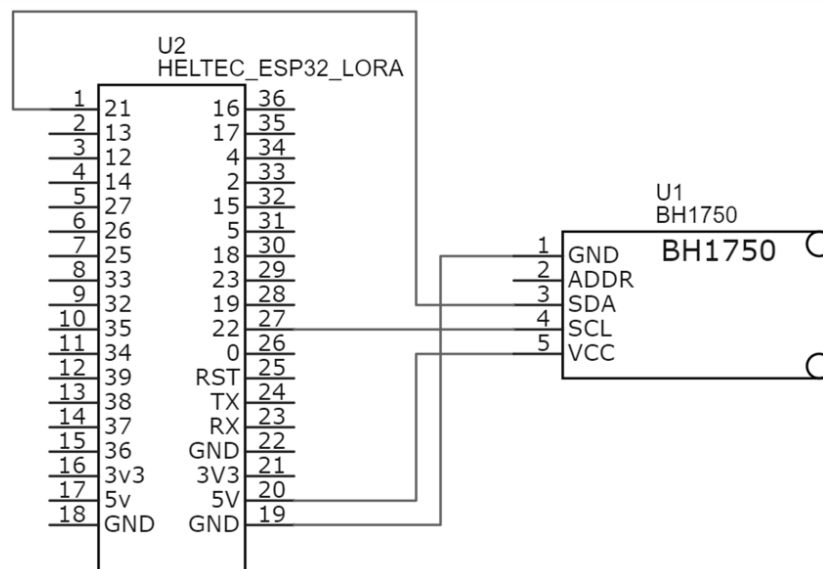
06:30:36.053 -> Temp *C: 31.90
06:30:36.053 -> Hum %: 76.70
06:30:37.049 -> Temp *C: 31.90
06:30:37.049 -> Hum %: 76.70
06:30:38.068 -> Temp *C: 31.90
06:30:38.068 -> Hum %: 76.70
06:30:39.052 -> Temp *C: 31.90
06:30:39.091 -> Hum %: 76.70
06:30:40.190 -> Temp *C: 31.90

```

รูปที่ 3.8 แสดงภาพผลลัพธ์ที่อ่านได้จากเซ็นเซอร์ AM2315

3.4.2 การอ่านค่าความสว่างจากเซ็นเซอร์ BH1750FVI

เซ็นเซอร์ BH1750FVI เป็นเซ็นเซอร์ที่มีค่าความละเอียดในการอ่านสูงถึง 16 บิตเชื่อมต่อแบบ I2C ทำให้ประหยัดขอ GPIO ของไมโครคอนโทรลเลอร์ที่เชื่อมต่อ ให้ค่าในการวัดเป็นหน่วย lux (ลักซ์) ซึ่งเป็นหน่วยการวัดแบบ SI (The International System of Unit) สำหรับความสว่าง (illuminance) = 1 lumen per square meter



รูปที่ 3.9 แสดงภาพ Schematic การเชื่อมต่อระหว่าง ESP32LoRa และ BH1750FVI

ตารางที่ 3.3 แสดงการต่อขาระหว่าง ESP32LoRa และ BH1750FVI

ESP32LoRa	BH1750FVI
5V	VCC
GND	GND
SDA(GPIO21)	SDA
SCL(GPIO22)	SCL

```
#include <Wire.h>
#include <BH1750.h>

BH1750 lightMeter;

void setup(){

    Serial.begin(9600);

    // Initialize the I2C bus (BH1750 library doesn't do this automatically)
    Wire.begin();
    // On esp8266 you can select SCL and SDA pins using Wire.begin
    // For Wemos / Lolin D1 Mini Pro and the Ambient Light shield

    lightMeter.begin();

    Serial.println(F("BH1750 Test begin"));

}

void loop() {

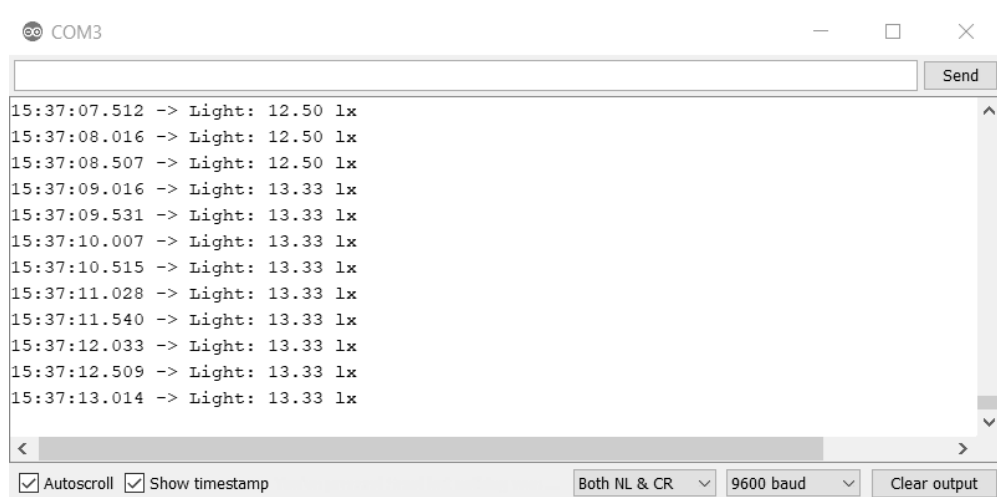
    float lux = lightMeter.readLightLevel();
    Serial.print("Light: ");
    Serial.print(lux);
    Serial.println(" lx");
    delay(1000);

}
```

รูปที่ 3.10 แสดงภาพโปรแกรมที่ใช้อ่านค่าจากเซ็นเซอร์ BH1750FVI

จากรูปที่ 3.10 เป็นโปรแกรมภาษา C++ ที่ใช้ในการอ่านค่าจากเซ็นเซอร์ BH1750FVI โดยมีการเรียกใช้ไลบรารี (Library) Wire.h และ BH1750.h ก่อนอ่านค่าจากเซ็นเซอร์ เริ่มต้นการ

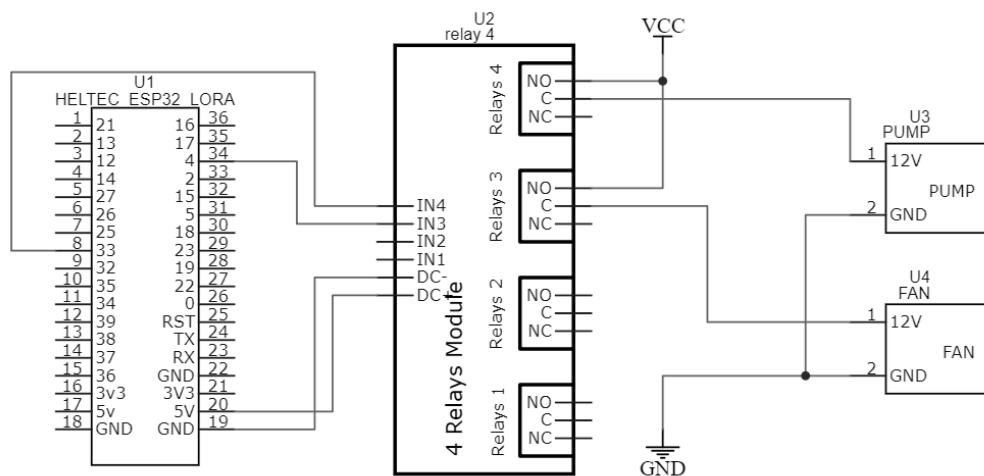
ทำงานของเซ็นเซอร์โดยการเรียกใช้ฟังก์ชัน `Wire.begin();` และ `lightMeter.begin();` ภายในฟังก์ชัน `setup()` จากนั้นทำการอ่านค่าโดยใช้ฟังก์ชัน `lightMeter.readLightLevel();` มาเก็บในตัวแปร `lux` ที่มีประเภทเป็น `float` และแสดงค่าที่ได้ทาง Serial Monitor ในหน่วย `lux` (ลักซ์) ดังรูปที่แสดงด้านล่าง



รูปที่ 3.11 แสดงภาพผลลัพธ์ที่อ่านได้จากเซ็นเซอร์ BH1750FVI

3.4.3 การควบคุมพัดลมและปั๊มพ่นหมอก

การควบคุมพัดลมและปั๊มพ่นหมอกผู้จัดทำได้ใช้ 4-Channel Relay Module ในการควบคุมการทำงานเพราะพัดลมและปั๊มพ่นหมอกใช้แรงดันไฟเลี้ยง 12V 4-Channel Relay Module เป็น Relay แบบ Active LOW จะทำงานเมื่อส่งขา GPIO ของ ไมโครคอนโทรลเลอร์ ESP32LoRa เป็น LOW



รูปที่ 3.12 แสดงภาพ Schematic ของการควบคุมพัดลมและปั้มน้ำหมอก

ตารางที่ 3.4 แสดงการต่อขาระหว่าง 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 3.13 แสดงภาพโปรแกรมส่วนที่ใช้ควบคุมการทำงานของ Relay

จากรูปที่ 3.13 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

3.4.3.1 การทำงานของปั๊มพ่นหมอกและพัดลมในโหมดการทำงานแบบอัตโนมัติ (AUTO)

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

ตารางที่ 3.5 แสดงการทำงานของปั๊มพ่นหมอกและพัดลมในโหมดการทำงานแบบอัตโนมัติ (AUTO)

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

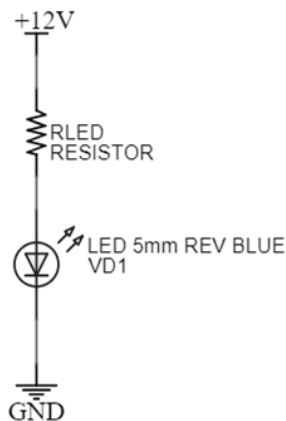
```

รูปที่ 3.14 แสดงภาพโปรแกรมที่ใช้ควบคุมการทำงานของปั๊มพ่นหมอกและพัดลม

จากรูปที่ 3.14 แสดงภาพโปรแกรมภาษา C++ ที่ใช้ควบคุมการทำงานของปั๊มพ่นหมอกและพัดลมในโหมดการทำงานแบบอัตโนมัติ (AUTO) ตัวแปร ctrlMode คือตัวแปรโหมดการทำงานของระบบ ตัวแปร temp คือตัวแปรค่าอุณหภูมิ และตัวแปร humi คือตัวแปรค่าความชื้นที่อ่านได้จากเซ็นเซอร์ AM2315 ตัวแปร set_temp_min และ set_temp_max คือค่าอุณหภูมิต่ำสุดและสูงสุด ตัวแปร set_humi_min และ set_humi_max คือค่าความชื้นต่ำสุดและสูงสุด ตัวแปร pumpState และ fanState ค่าตัวแปรที่ใช้ในการควบคุมการทำงานของปั๊มพ่นหมอกและพัดลม

3.4.4 การควบคุมความสว่างภายในโรงเรือนเพาะเห็ด

ในการควบคุมความสว่างภายในโรงเรือนผู้จัดทำได้ใช้ LED ขนาด 5mm สีน้ำเงินจำนวน 112 หลอดเพื่อให้ความสว่าง จากนั้นใช้ L298N ในการควบคุมความสว่างโดยสัญญาณ PWM จากไมโครคอนโทรลเลอร์



รูปที่ 3.15 แสดงภาพวงจร LED Driver

จากรูปที่ 3.15 จะต้องคำนวณหาค่าของตัวต้านทานที่จะนำมาต่ออนุกรมกับหลอด LED เพื่อจำกัดกระแสตามความต้องการของหลอด LED โดยสามารถคำนวณได้จากสมการ โดยกำหนดให้ $V_{LED} = 3.2V$, $I_{LED} = 20mA$

$$V = V_{RLED} + V_{LED}$$

จะได้

$$V = I_{LED}R_{LED} + V_{LED}$$

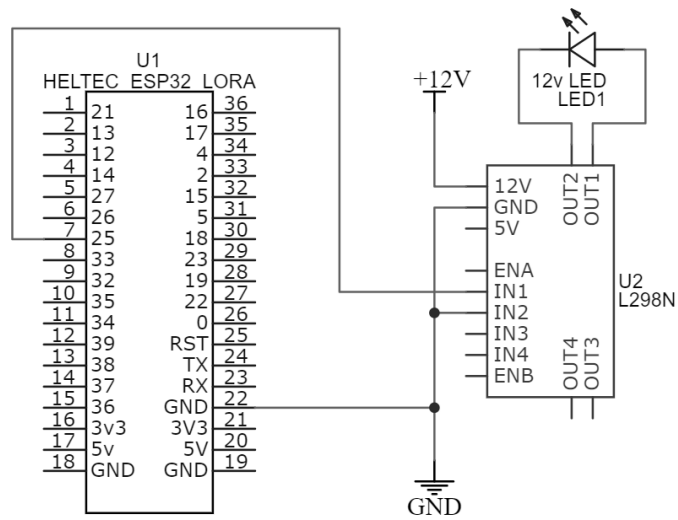
$$R_{LED} = (V - V_{LED}) / I_{LED}$$

แทนค่า

$$R_{LED} = (12V - 3.2V) / 20mA$$

$$R_{LED} = 440 \Omega$$

ดังนั้นค่าของตัวต้านทานที่จะนำมาต่ออนุกรมกับหลอด LED มีค่าเท่ากับ 440 โอห์ม



รูปที่ 3.16 แสดงภาพ Schematic ของการควบคุมความสว่างภายในโรงเรือนเพาะเห็ด

ตารางที่ 3.6 แสดงการต่อขาระหว่าง ESP32LoRa และ L298N

ESP32LoRa	4-Channel Relay Module
GPIO25	IN1
GND	GND
GND	IN2

```

if (lux < set_lux) {
    if (pwmvalue < 100) {
        pwmvalue++;
    }
    if (pwmvalue == 100) {
        pwmvalue = 100;
    }
}
else if (lux > set_lux) {
    if (pwmvalue > 0) {
        pwmvalue--;
    }
    if (pwmvalue == 0) {
        pwmvalue = 0;
    }
}
ledcWrite(ledChannel, pwm(pwmvalue));

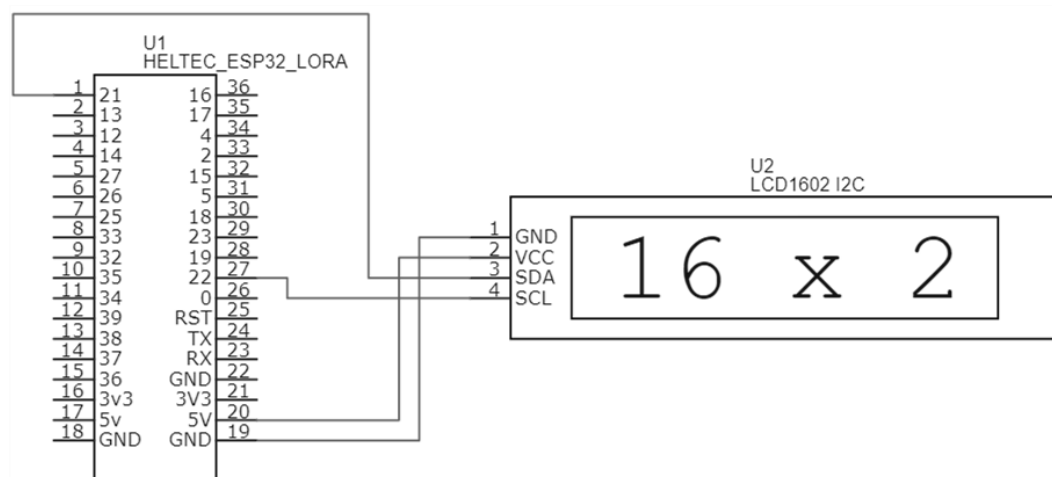
```

รูปที่ 3.17 แสดงภาพโปรแกรมที่ใช้ในการควบคุมความสว่างของหลอดไฟ

จากรูปที่ 3.17 โปรแกรมจะทำการตรวจสอบว่าความสว่างต่ำกว่าหรือสูงกว่าค่าที่กำหนดหรือไม่ หากต่ำกว่าค่าที่กำหนดจะทำการเพิ่มค่า pwmvalue เพื่อเพิ่มความสว่างของหลอด LED หากสูงกว่าค่าที่กำหนดจะทำการลดค่า pwmvalue เพื่อลดความสว่างของหลอด LED

3.4.5 การใช้จอแอลซีดี (LCD) แสดงผล

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



รูปที่ 3.18 แสดงภาพ Schematic การเชื่อมต่อระหว่าง ESP32LoRa และ จอแอลซีดี (LCD)

ตารางที่ 3.7 แสดงการต่อขาระหว่าง 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 3.19 แสดงภาพโปรแกรมทดสอบการทำงานของจอแอลซีดี (LCD)

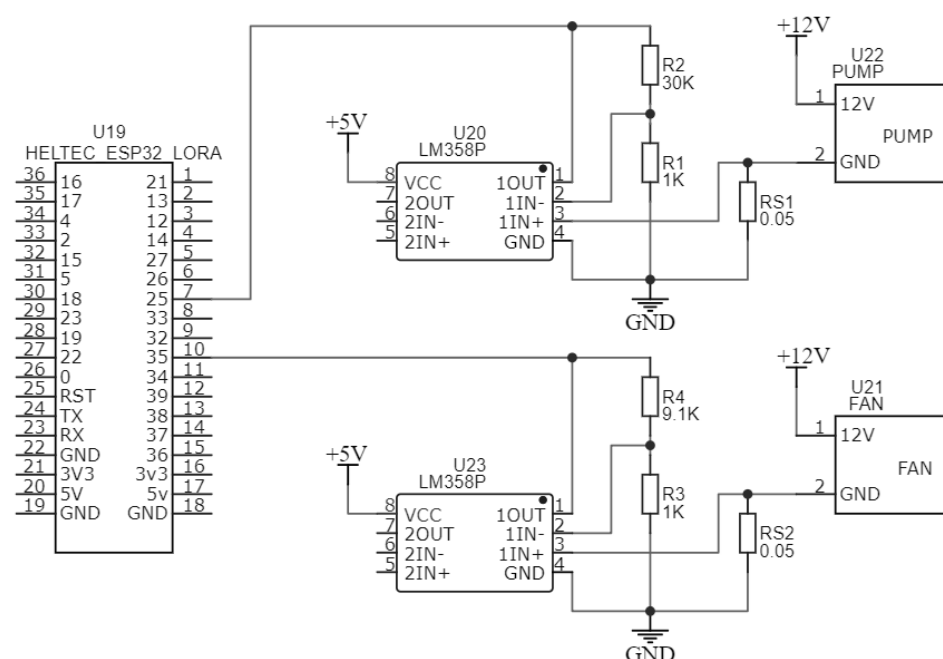
จากรูปที่ 3.19 เป็นโปรแกรมภาษา C++ ที่ใช้ในการทดสอบการแสดงผลของจอแอลซีดี ขนาด 16 x 2 การเชื่อมต่อแบบ I2C โดยกำหนด I2C Address ของจอแอลซีดี (LCD) เป็น Address 0x27 จากนั้นใช้ฟังก์ชัน lcd.begin() เพื่อเริ่มต้นการทำงานของ LCD ใช้ฟังก์ชัน lcd.setCursor() ในการกำหนดตำแหน่งเริ่มต้นของตัวอักษร และใช้ฟังก์ชัน lcd.print() ในการแสดงตัวอักษร รูปที่แสดงด้านล่างแสดงผลลัพธ์ที่ได้จากโปรแกรมทดสอบการทำงานของจอแอลซีดี (LCD) แบบ I2C



รูปที่ 3.20 แสดงภาพผลลัพธ์ที่ได้จากโปรแกรมทดสอบการทำงานของจอแอลซีดี (LCD)

3.4.6 การอ่านค่าแรงดันเอาต์พุตจากวงจรตัวต้านทานตรวจสอบกระแส

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



รูปที่ 3.21 แสดงภาพ Schematic การเชื่อมต่อระหว่าง ESP32LoRa และวงจรตัวต้านทานตรวจสอบกระแส

ตารางที่ 3.8 แสดงการต่อขาระหว่าง ESP32LoRa และวงจรตัวต้านทานตรวจสอบกระแส

ESP32LoRa	วงจรตัวต้านทานตรวจสอบกระแส
GPIO25	OUT (LM358P) PUMP
GPIO35	OUT (LM358P) 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);
}

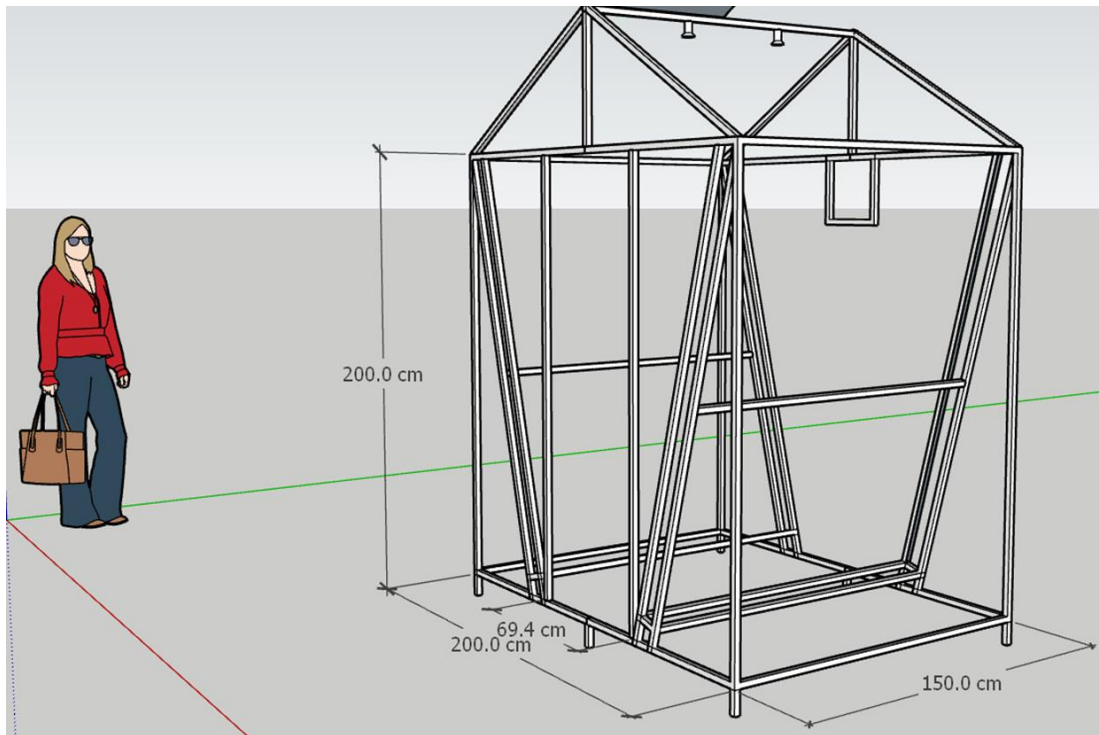
```

รูปที่ 3.22 แสดงภาพโปรแกรมทดสอบการอ่านค่าแรงดันเอาต์พุตจากวงจรตัวต้านทานตรวจสอบกระแส

จากรูปที่ 3.22 แสดงภาพโปรแกรมที่ใช้ในการอ่านค่าแรงดันเอาต์พุตจากวงจรตัวต้านทานตรวจสอบกระแส โดยใช้ GPIO25 และ GPIO35 ของ ESP32LoRa ในการอ่านค่าแรงดันที่ได้จากเอาต์พุตของวงจร ใช้ ADC ความละเอียดในการอ่าน 12-bit จากนั้นนำค่าที่ได้มาคำนวณเป็นแรงดันที่สามารถอ่านได้จากเอาต์พุตของวงจร

3.5 การออกแบบและสร้างโรงเรือนสำหรับเพาะเห็ด

ในโครงการนี้ผู้จัดทำได้ใช้โปรแกรม Google Sketchup ในการออกแบบโรงเรือนสำหรับเพาะเห็ดขนาด 200x150x200 เซนติเมตร ตามรูปที่ 3.23 โดยได้ออกแบบให้เป็นโรงเรือนแบบปิดสามารถควบคุมสภาพแวดล้อมต่างๆภายในโรงเรือนได้เช่น อุณหภูมิ ความชื้น และแสง โครงสร้างของโรงเรือนเพาะเห็ดเป็นเหล็กกล่องขนาด 1x1 นิ้ว ดังรูปที่ 3.24 เพื่อโรงเรือนมีความแข็งแรงและสามารถใช้งานได้ในระยะยาว เมื่อทำการเชื่อมโครงสร้างโรงเรือนเสร็จแล้ว ขั้นตอนต่อไปคือการพ่นสีเพื่อป้องกันการเกิดสนิม ดังรูปที่ 3.25



รูปที่ 3.23 แสดงภาพการออกแบบโรงเรือนที่ใช้เพาะเห็ด



รูปที่ 3.24 แสดงภาพโครงสร้างของโรงเรือนเพาะเห็ด



รูปที่ 3.25 แสดงภาพขั้นตอนการพับสี่โรงเรือนเพาะเห็ด



รูปที่ 3.26 แสดงภาพขั้นตอนการติดลวดตาข่ายเพื่อป้องกันก้อนเห็ดหล่น



รูปที่ 3.27 แสดงภาพขั้นตอนการคลุมโรงเรือนเพาะเห็ดเพื่อเก็บรักษาความชื้น



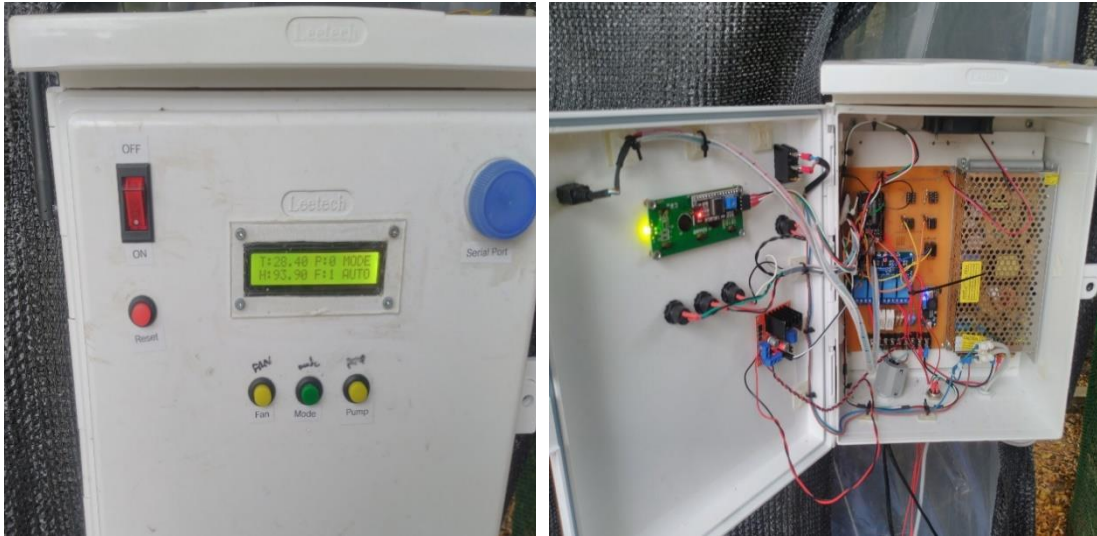
รูปที่ 3.28 แสดงภาพขั้นตอนการติดตั้งพัดลมระบายอากาศ



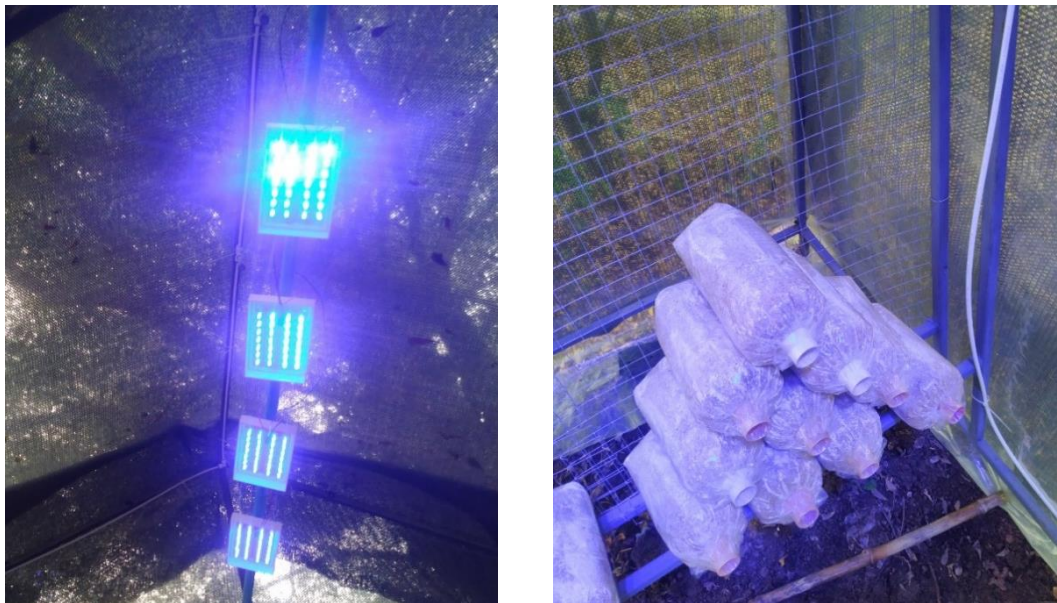
รูปที่ 3.29 แสดงภาพขั้นตอนการติดตั้งหัวพ่นหมอก



รูปที่ 3.30 แสดงภาพขั้นตอนการคลุมโรงเรือนโดยใช้สแลนเพื่อป้องกันแสงแดดและลดความร้อนจากแสงแดด



รูปที่ 3.31 แสดงภาพขั้นตอนการติดตั้งกล่องควบคุมระบบ



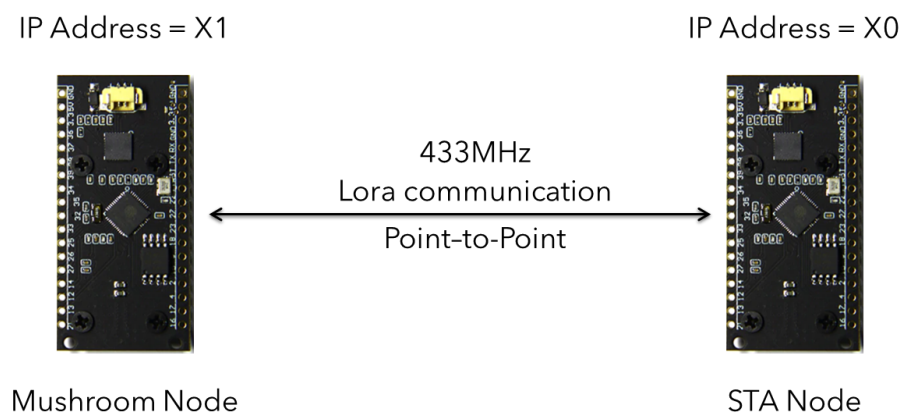
รูปที่ 3.32 แสดงภาพขั้นตอนการติดตั้งหลอด LED สีน้ำเงินเพื่อให้แสงสว่าง



รูปที่ 3.33 แสดงภาพขั้นตอนการติดตั้งเซ็นเซอร์ BH1750FVI

3.6 การออกแบบการสื่อสารผ่าน Lora

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

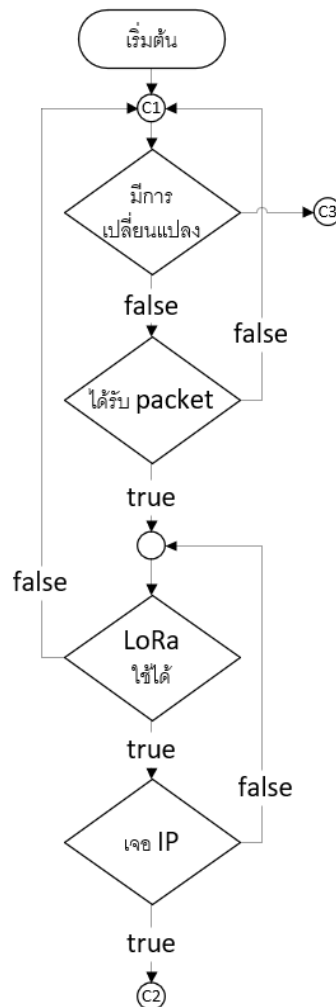


รูปที่ 3.34 แสดงภาพการสื่อสารแบบ Node-to-Node

จากรูปที่ 3.34 Node แต่ละ Node จะมี IP Address เป็นของตัวเองเพื่อใช้ในการระบุตัวตน เช่น Mushroom Node มี IP Address คือ X1 และ STA Node มี IP Address คือ X0

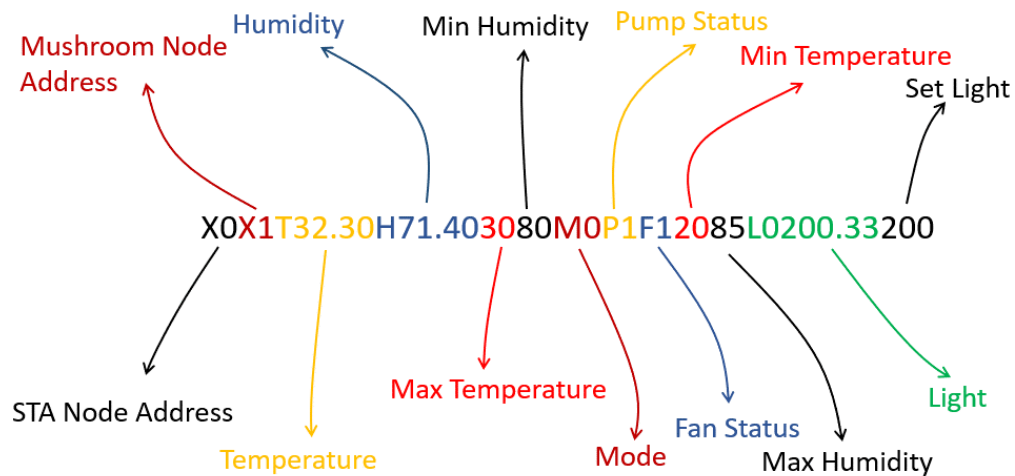
3.6.1 การออกแบบการสื่อสาร Lora ที่ Mushroom Node

ในการออกแบบการสื่อสารผ่าน Lora ที่ Mushroom Node ผู้จัดทำจะขออธิบายรายละเอียดต่างๆ โดยใช้ Flowchart ประกอบการอธิบาย



รูปที่ 3.35 แสดงภาพ Flowchart การทำงานของ Mushroom Node (1)

จากรูปที่ 3.35 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 3.36



รูปที่ 3.36 แสดงภาพรูปแบบของข้อมูล (Data Format)

จากรูปที่ 3.36 แสดงรูปแบบของข้อมูล (Data Format) ที่ส่งไปยัง STA Node เมื่อข้อมูลมีการเปลี่ยนแปลงโดยมีรายละเอียดดังต่อไปนี้

- X0 คือ IP Address ของ STA Node หรือ Node ปลายทางที่ต้องการส่งไป เพื่อให้ปลายทางหรือ 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 คือค่าอุณหภูมิที่วัดได้จากเซ็นเซอร์ AM2315 โดย T คือ Keyword ที่ใช้บอกว่าค่าตัวเลขที่ต่อท้ายมาด้วยจำนวน 4 หลักเป็นค่าอุณหภูมิ
- H71.40 คือค่าความชื้นที่วัดได้จากเซ็นเซอร์ AM2315 โดย H คือ Keyword ที่ใช้บอกว่าค่าตัวเลขที่ต่อท้ายมาด้วยจำนวน 4 หลักเป็นค่าความชื้น
- 30 คือค่าอุณหภูมิสูงสุดที่ใช้ในการควบคุมอุณหภูมิ
- 80 คือค่าอุณหภูมิต่ำสุดที่ใช้ในการควบคุมความชื้น
- M0 คือโหมด (Mode) การทำงานของระบบควบคุมอุณหภูมิและความชื้นภายในโรงเรือนโดย M คือ Keyword ที่ใช้บอกว่าค่าตัวเลขที่ต่อท้ายมาด้วยจำนวน 1 หลักเป็นค่าโหมดการทำงานของระบบ 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
- L0200.33 คือค่าความสว่างที่วัดได้จากเซ็นเซอร์ BH1750FVI โดย L คือ Keyword ที่ใช้บอกว่าค่าตัวเลขที่ต่อท้ายมาด้วยจำนวน 8 หลักเป็นค่าความสว่าง
- 200 คือค่าความสว่างที่กำหนดไว้

การส่งข้อมูลการเปลี่ยนแปลงไปยัง Node ปลายทางหรือ STA Node จะเรียกใช้ฟังก์ชัน sendUpdateData() ในการส่งข้อมูล ฟังก์ชัน sendUpdateData() เป็นฟังก์ชันที่เขียนด้วยภาษา C++ ดังรูปที่ 3.37

```

void sendUpdateData() { //this function will
  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(pumpState);

  LoRa.print("F");
  LoRa.print(fan_check);
  LoRa.print(set_temp_min);
  LoRa.print(set_humi_max);
  LoRa.print("L");
  if (lux < 10.00) {
    LoRa.print("000" + String(lux));
  }
  else if (lux < 100.00) {
    LoRa.print("00" + String(lux));
  }
  else if (lux < 1000.00) {
    LoRa.print("0" + String(lux));
  }
  else if (lux >= 10000.00) {
    Serial.print("Rovflux");
  } else {
    LoRa.print(lux);
  }
  if (set_lux < 100) {
    LoRa.print("0" + String(set_lux));
  } else {
    LoRa.print(set_lux);
  }
  LoRa.endPacket();
}

```

รูปที่ 3.37 แสดงภาพฟังก์ชัน sendUpdateData()

จากรูปที่ 3.37 แสดงฟังก์ชัน sendUpdateData() ที่ใช้ส่งข้อมูลเมื่อมีการเปลี่ยนแปลงโดยรายละเอียดมีการทำงานดังต่อไปนี้ ก่อนส่งข้อมูลจะตรวจสอบค่าแรงดันตกคร่อมตัวต้านทานตรวจสอบกระแสก่อนเพื่อให้ได้ค่าสถานะการทำงานของอุปกรณ์ที่ตรงกับค่าจริง จากนั้นจะส่งข้อมูลที่มีรูปแบบ (Data Format) เหมือนรูปที่ 3.36 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 ค่าอุณหภูมิที่วัดได้จากเซ็นเซอร์ AM2315

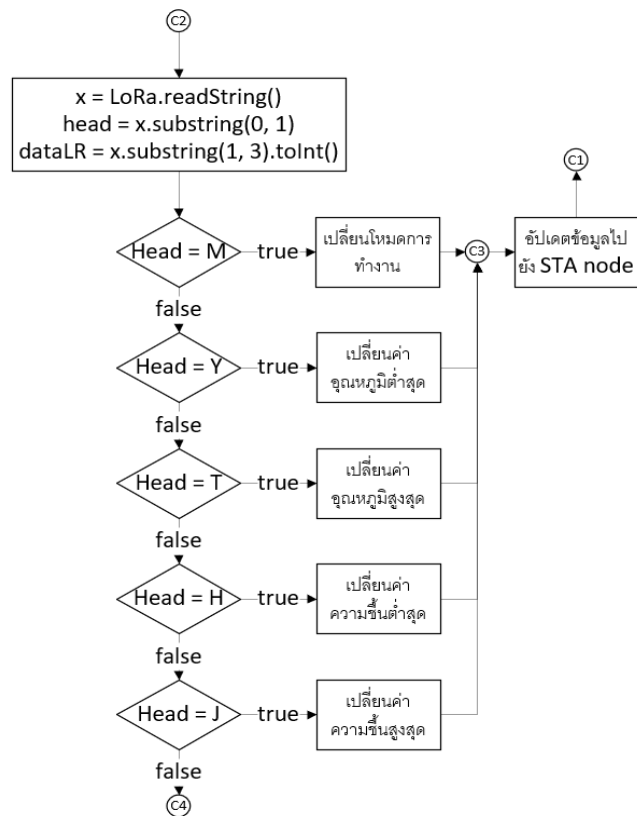
- tempStr ค่าที่ใช้ส่งแทนค่า temp ในกรณีที่อุณหภูมิต่ำกว่า 10.00 เพื่อป้องกันความยาวของข้อมูลคลาดเคลื่อน
- humi ค่าความชื้นที่วัดได้จากเซ็นเซอร์ AM2315
- humiStr ค่าที่ใช้ส่งแทนค่า humi ในกรณีที่ความชื้นต่ำกว่า 10.00 เพื่อป้องกันความยาวของข้อมูลคลาดเคลื่อน
- set_temp_max คือค่าอุณหภูมิสูงสุดที่ใช้ในการควบคุมอุณหภูมิ
- set_humi_min คือค่าอุณหภูมิต่ำสุดที่ใช้ในการควบคุมความชื้น
- ctrlMode คือโหมด (Mode) การทำงานของระบบควบคุมอุณหภูมิและความชื้นภายในโรงเรือน
- set_temp_min คือค่าอุณหภูมิต่ำสุดที่ใช้ในการควบคุมอุณหภูมิ
- set_humi_max คือค่าอุณหภูมิสูงสุดที่ใช้ในการควบคุมความชื้น
- lux คือค่าความสว่างที่วัดได้จากเซ็นเซอร์ BH1750FVI
- set_lux คือค่าความสว่างที่กำหนด

จากรายละเอียดด้านบนเป็นการส่งข้อมูลการเปลี่ยนแปลงไปยัง STA Node ในกรณีที่ข้อมูลมีการเปลี่ยนแปลง แต่ถ้าหากข้อมูลไม่มีการเปลี่ยนแปลงก็จะทำการตรวจสอบเงื่อนไขข้อต่อไป คือการตรวจสอบว่าข้อมูลส่งมาจาก STA Node หรือไม่ ดังรูปที่ 3.35 ถ้าไม่มีข้อมูลที่ส่งมาก็จะกลับไปตรวจสอบการเปลี่ยนแปลงของข้อมูล

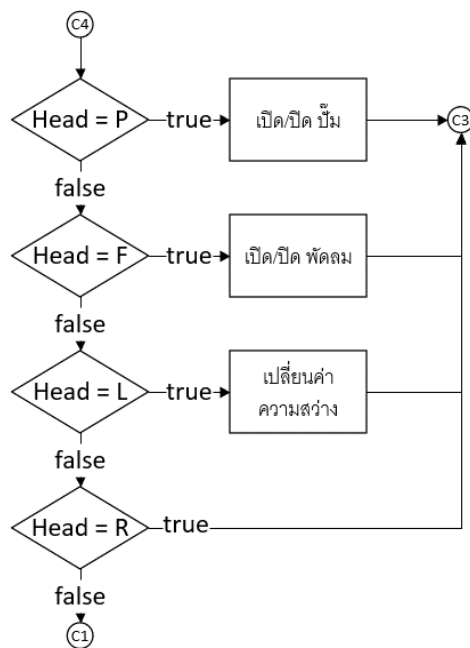
หากมีข้อมูลส่งมาจาก STA Node ก็จะมีการตรวจสอบข้อมูลโดยหา IP Address -ของ ตัวเอง (Mushroom Node IP Address) ว่าข้อมูลที่ส่งมาเป็นข้อมูลของตัวเองหรือไม่ ถ้าใช่ก็จะนำข้อมูลที่ได้มาประมวลผลต่อไปดังรูปที่ 3.38 เป็นโปรแกรมส่วนตรวจสอบข้อมูล

```
//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 3.38 แสดงภาพโปรแกรมส่วนที่ตรวจสอบข้อมูลที่ส่งมายัง Mushroom Node



รูปที่ 3.39 แสดงภาพ Flowchart การทำงานของ Mushroom Node (2)



รูปที่ 3.40 แสดงภาพ Flowchart การทำงานของ Mushroom Node (3)

จากรูปที่ 3.39 และ รูปที่ 3.40 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 3.41 แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (1)

จากรูปที่ 3.41 ถ้าข้อมูลที่รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “M” หมายความว่าต้องการเปลี่ยนโหมดการทำงานของระบบควบคุมอุณหภูมิและความชื้นโดยการทำงานคือจะตรวจสอบว่าถ้าโหมดการทำงานเป็นโหมด AUTO (ctrlMode = false) จะเปลี่ยนเป็นโหมด MANUAL (ctrlMode = true) และถ้าโหมดการทำงานเป็นโหมด MANUAL (ctrlMode = true) ก็ จะเปลี่ยนโหมดการทำงานเป็นโหมด AUTO (ctrlMode = false) กล่าวคือเป็นการทำงานแบบสลับ (toggle) โหมดการทำงาน เมื่อเปลี่ยนโหมดการทำงานสำเร็จแล้วก็จะส่งข้อมูลการเปลี่ยนแปลง กลับไปยัง STA Node โดยใช้ฟังก์ชัน sendUpdateData()

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

รูปที่ 3.42 แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (2)

จากรูปที่ 3.42 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 == "Y") {  
    set_temp_max = dataLR;  
    EEPROM.write(0, set_temp_max);  
    EEPROM.commit();  
    sendUpdateData();  
}
```

รูปที่ 3.43 แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (3)

จากรูปที่ 3.43 ถ้าข้อมูลที่ได้รับจาก 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 == "T") {  
    set_humi_min = dataLR;  
    EEPROM.write(10, set_humi_min);  
    EEPROM.commit();  
    sendUpdateData();  
}
```

รูปที่ 3.44 แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (4)

จากรูปที่ 3.44 ถ้าข้อมูลที่ได้รับจาก 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 3.45 แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (5)

จากรูปที่ 3.45 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 3.46 แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (6)

จากรูปที่ 3.46 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 เช่นเดียวกันถ้า ปัจจุบันค่าควบคุมการทำงานของปั๊ม (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 3.47 แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (7)

จากรูปที่ 3.47 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

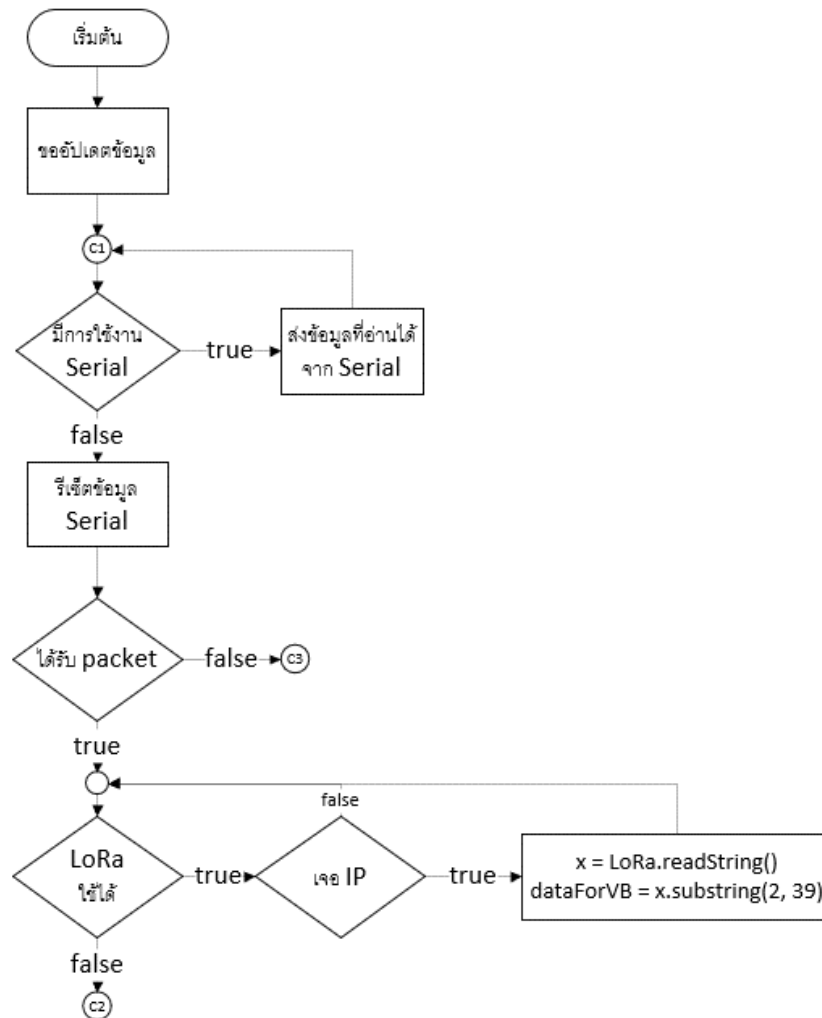
```
else if (head == "L") {
    int lora_lux = x.substring(1, 4).toInt();
    set_lux = lora_lux;
    if (set_lux < 100) {
        String set_luxstr = String(set_lux);
        int I, II, III;
        I = 0;
        II = set_luxstr.substring(0, 1).toInt();
        III = set_luxstr.substring(1, 2).toInt();
        EEPROM.write(2, I);
        EEPROM.write(3, II);
        EEPROM.write(4, III);
        EEPROM.commit();
    } else {
        String set_luxstr = String(set_lux);
        int I, II, III;
        I = set_luxstr.substring(0, 1).toInt();
        II = set_luxstr.substring(1, 2).toInt();
        III = set_luxstr.substring(2, 3).toInt();
        EEPROM.write(2, I);
        EEPROM.write(3, II);
        EEPROM.write(4, III);
        EEPROM.commit();
    }
    sendUpdateData();
}
```

รูปที่ 3.48 แสดงภาพโปรแกรมประมวลผลข้อมูลที่ได้รับจาก STA Node (8)

จากรูปที่ 3.48 ถ้าข้อมูลที่ได้รับจาก STA Node มี Keyword ตัวแรกเท่ากับ “L” หมายความว่าต้องการเปลี่ยนแปลงค่าความสว่างโดยมีรายละเอียดการทำงานดังต่อไปนี้ เมื่อเจอ Keyword “L” จะนำข้อมูลต่อท้ายจำนวน 3 หลักมาเปลี่ยนค่าความสว่างที่กำหนดหรือกำหนดระดับความเข้มของแสง โดยจะทำการตรวจสอบค่าความสว่างที่กำหนดหากมีค่าต่ำกว่า 100 ก็ให้นำเลข 0 มีแทรกด้านหน้าของข้อมูลเพื่อไม่ให้ความยาวของข้อมูลเปลี่ยนแปลง จากนั้นทำการบันทึกค่าความสว่างที่ถูกส่งมาจาก STA Node ลงใน EEPROM และส่งข้อมูลการเปลี่ยนแปลงกลับไปยัง STA Node

3.6.2 การออกแบบการสื่อสาร Lora ที่ STA Node

ในการออกแบบการสื่อสารผ่าน Lora ที่ STA Node ผู้จัดทำจะขออธิบายรายละเอียดต่างๆ โดยใช้ Flowchart ประกอบการอธิบาย



รูปที่ 3.49 แสดงภาพ Flowchart การทำงานของ STA Node (1)

จากรูปที่ 3.49 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
- L คือ Keyword ที่ส่งเมื่อต้องการเปลี่ยนค่าความสว่างหรือระดับความสว่างตามด้วยตัวเลขจำนวน 2 หลักหรือ 3 หลัก เช่น L90, L200

จากรายละเอียดด้านบนเป็นข้อมูลจาก Serial Port เมื่อได้รับข้อมูลจาก Serial Port แล้วก็จะส่งข้อมูลต่อไปยัง Mushroom Node ทันทีดังรูปที่แสดงด้านล่าง

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

รูปที่ 3.50 แสดงภาพโปรแกรมส่วนที่รับข้อมูลจาก Serial Port

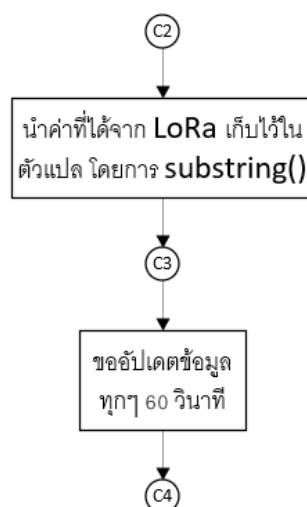
จากรูปที่ 3.50 จะรับข้อมูลจาก Serial Port แล้วส่งข้อมูลไปยัง Mushroom Node ด้วยฟังก์ชัน loraSend() โดยข้อมูลที่ส่งจะมีรูปแบบ (Data Format) ดังนี้ X1M เมื่อ X1 คือ IP Address

ของ Mushroom Node และ M คือ Keyword ที่ส่งไปเพื่อควบคุมการทำงานของระบบดังรายละเอียดด้านบน จากนั้นจะล้างข้อมูลที่ได้รับจาก Serial Port

เมื่อระบบทำงานผ่านขั้นตอนต่างๆ ด้านบนมาแล้วก็ขั้นตอนต่อไปจะทำการตรวจสอบเงื่อนไขข้อต่อไปคือการตรวจสอบว่าข้อมูลส่งมาจาก Mushroom Node หรือไม่ ดังรูปที่ 3.49 ถ้าไม่มีข้อมูลที่ส่งมาก็จะส่งข้อมูลเพื่อขออัปเดตข้อมูล (C3) ทุกๆ 60 วินาที ถ้ามีข้อมูลส่งมาจาก Mushroom Node ก็จะทำให้การหา IP Address - ของตัวเอง (STA Node IP Address) ว่าข้อมูลที่ส่งมาเป็นข้อมูลของตัวเองหรือไม่ ถ้าใช่ก็จะนำข้อมูลที่ได้มาประมวลผลต่อไป

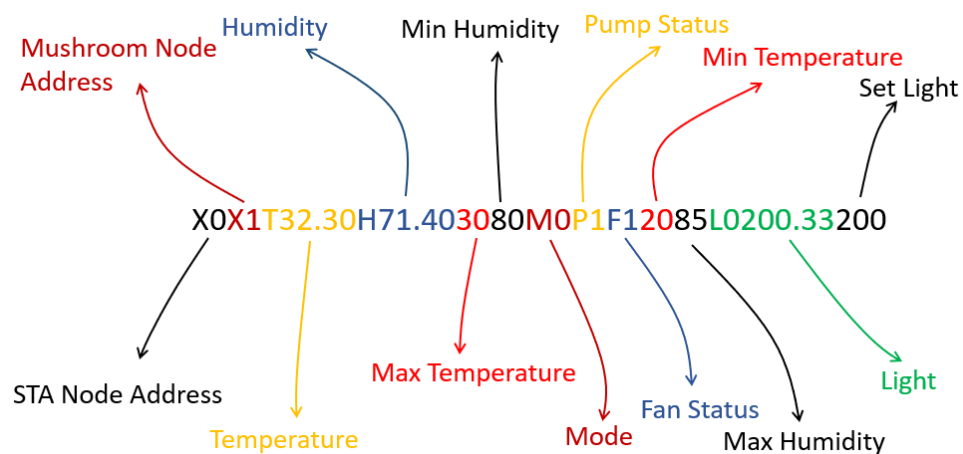
```
// try to parse packet
if (LoRa.parsePacket())
{
    // received a packet
    //Serial.print("Received packet ");
    // read packet
    while (LoRa.available())
    {
        if (LoRa.find(ipAddr))
        { // ip address this device "
            x = LoRa.readString();
            dataForVB = x.substring(2,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

รูปที่ 3.51 แสดงภาพโปรแกรมส่วนที่ตรวจสอบข้อมูลที่ส่งมายัง STA Node



รูปที่ 3.52 แสดงภาพ Flowchart การทำงานของ STA Node (2)

เมื่อได้รับข้อมูลที่ส่งมาจาก Mushroom Node แล้วก็นำข้อมูลที่ได้มาเก็บไว้ในตัวแปลต่างๆ โดยใช้ฟังก์ชัน substring() เพื่อนำข้อมูลที่ได้ไปใช้งานบน Web Application และ Windows Application ตัวอย่างรูปแบบของข้อมูล (Data Format) ที่ส่งมาจาก Mushroom Node ดังรูปที่ 3.53



รูปที่ 3.53 แสดงภาพรูปแบบของข้อมูล (Data Format) ที่ส่งมาจาก Mushroom Node

จากรูปที่ 3.53 แสดงรูปแบบของข้อมูลที่ส่งมาจาก Mushroom Node เมื่อได้รับข้อมูลก็นำข้อมูลที่ได้มา substring() ดังรูปที่ 3.54

```

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();
    String lux_raw = x.substring(29, 36);
    String I = lux_raw.substring(0, 1);
    String II = lux_raw.substring(0, 2);
    String III = lux_raw.substring(0, 3);
    //Serial.println(lux_raw);Serial.println(II

    if(III == "000"){
        lux = lux_raw.substring(3, 7).toFloat();
    }
    else if(II == "00"){
        lux = lux_raw.substring(2, 7).toFloat();
    }
    else if(I == "0"){
        lux = lux_raw.substring(1, 7).toFloat();
    }
    else{
        lux = lux_raw.toFloat();
    }
    //Serial.println(lux);
    String set_lux_raw = x.substring(36, 39);
    //Serial.println(set_lux_raw);
    String o = set_lux_raw.substring(0, 1);
    if(o == "0"){
        set_lux = set_lux_raw.substring(1, 3).toInt()
    }
    else{
        set_lux = set_lux_raw.toInt();
    }
}

```

รูปที่ 3.54 แสดงภาพการใช้ฟังก์ชัน substring() กับข้อมูลที่ได้รับจาก Mushroom Node

จากรูปที่ 3.54 แสดงการใช้ฟังก์ชัน substring() เพื่อนำข้อมูลที่ได้รับมาเก็บไว้ในตัวแปร ที่ถูกต้องโดยมีรายละเอียดดังต่อไปนี้

- 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

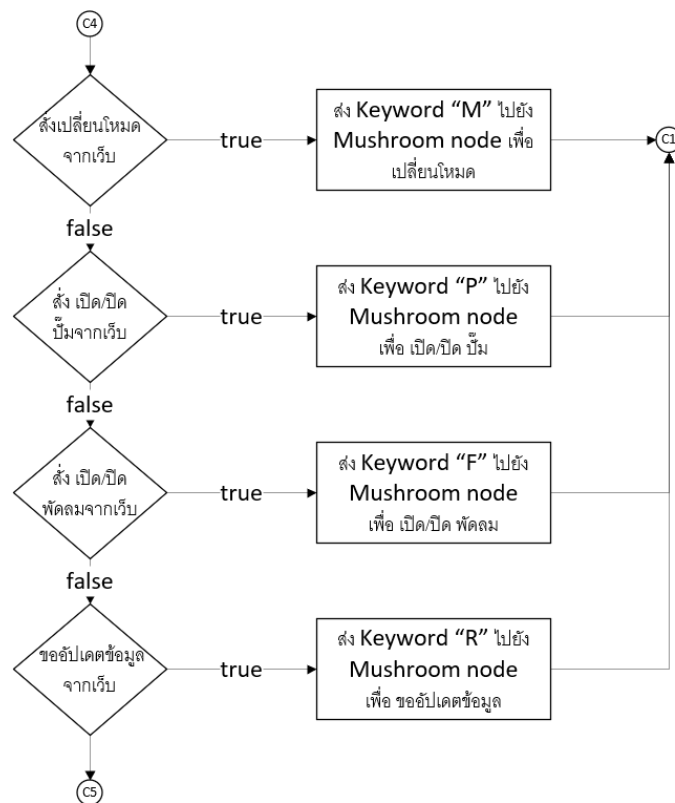
- ตัวแปร ctrlMode เป็นโหมดการทำงานของระบบควบคุมอุณหภูมิและความชื้นภายในโรงเรือน โดยการ substring(19, 20) จากตำแหน่งที่ 19 ถึงตำแหน่งที่ 20 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Int
- ตัวแปร pumpState เป็นค่าสถานะการทำงานของปั๊มที่ได้จากวงจรตัวต้านทานตรวจสอบกระแสโดยการ substring(21, 22) จากตำแหน่งที่ 21 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
- ตัวแปร lux เป็นค่าความสว่างที่ได้จากเซ็นเซอร์ BH1750FVI โดยการ substring(29, 36) จากตำแหน่งที่ 29 ถึงตำแหน่งที่ 36 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Float
- ตัวแปร set_lux เป็นค่าความสว่างที่กำหนดไว้ โดยการ substring(36, 39) จากตำแหน่งที่ 36 ถึงตำแหน่งที่ 39 แล้วเปลี่ยนประเภทของข้อมูลเป็นประเภท Int

เมื่อจัดการกลับข้อมูลที่ส่งมาเรียบร้อยแล้วขั้นตอนต่อไปคือการส่ง Keyword เพื่อขออัปเดตข้อมูลทุกๆ 60 วินาที ดังรูปที่ 3.55

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

รูปที่ 3.55 แสดงภาพโปรแกรมที่ใช้ในการส่ง Keyword ทุก 60 วินาที

จากรูปที่ 3.55 มีหลักการทำงานคือ STA Node จะส่ง Keyword “R” ทุกๆ 60 วินาที โดยใช้ฟังก์ชัน millis() ในการนับเวลา ซึ่งตัวแปร node1 มีค่าเท่ากับ X1 คือ IP Address ของ Mushroom Node รูปแบบข้อมูล (Data Format) คือ X1R

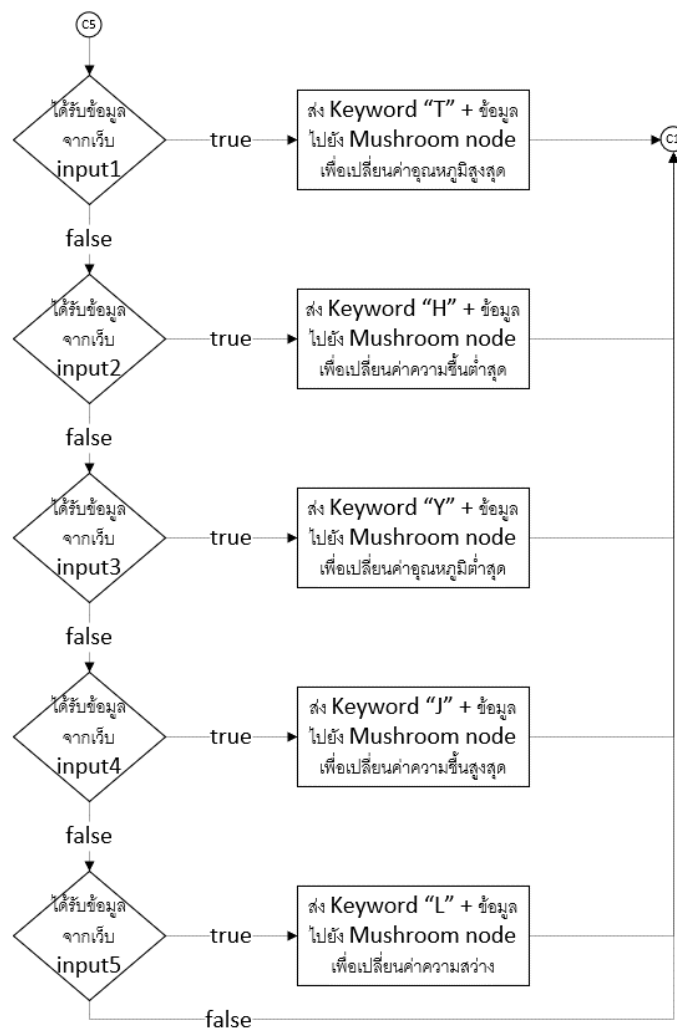


รูปที่ 3.56 แสดงภาพ Flowchart การทำงานของ STA Node (3)

จากรูปที่ 3.55 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 บั้มจากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “M” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการ เปิด/ปิด บั้ม 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



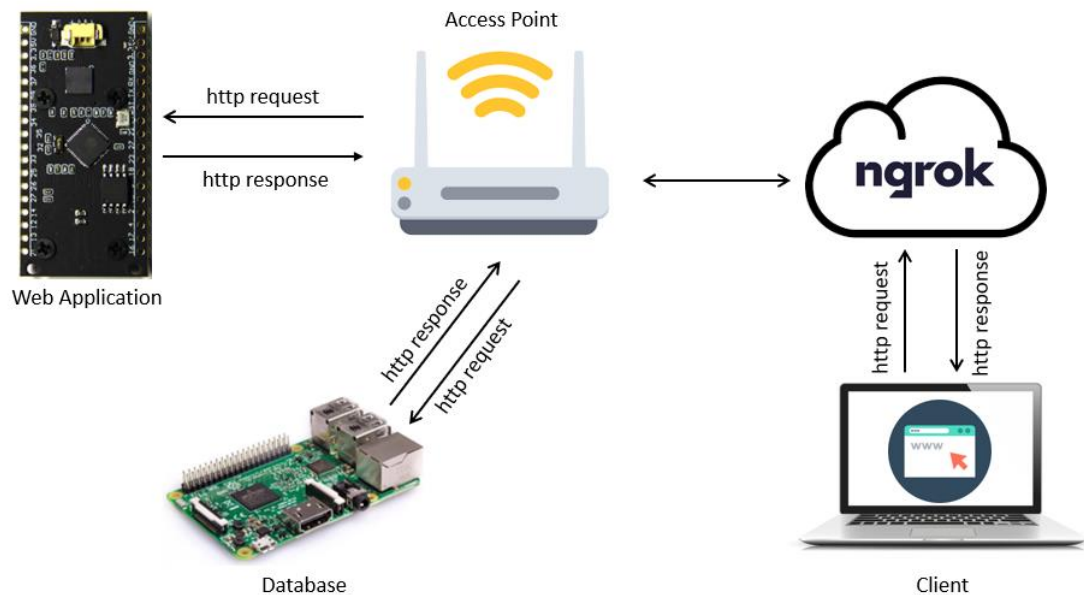
รูปที่ 3.57 แสดงภาพ Flowchart การทำงานของ STA Node (4)

จากรูปที่ 3.57 หากมีการกรอกข้อมูลจากเว็บแอปพลิเคชัน (Web Application) STA Node จะส่งข้อมูลต่อไปยัง Mushroom Node เพื่อเปลี่ยนแปลงข้อมูลตามที่ใช้กรอกในแต่ละ input โดยมีรายละเอียดดังนี้

- 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
- ถ้ามีการกรอกข้อมูลที่ input5 จากเว็บแอปพลิเคชัน (Web Application) จะส่ง Keyword “L” + “ข้อมูลที่กรอก” ไปยัง Mushroom Node เพื่อแจ้งว่าผู้ใช้ต้องการเปลี่ยนค่าความความสว่าง โดยมีรูปแบบของข้อมูลที่ส่งดังนี้ X1L90, X1L200

3.7 การออกแบบและสร้างเว็บแอปพลิเคชัน

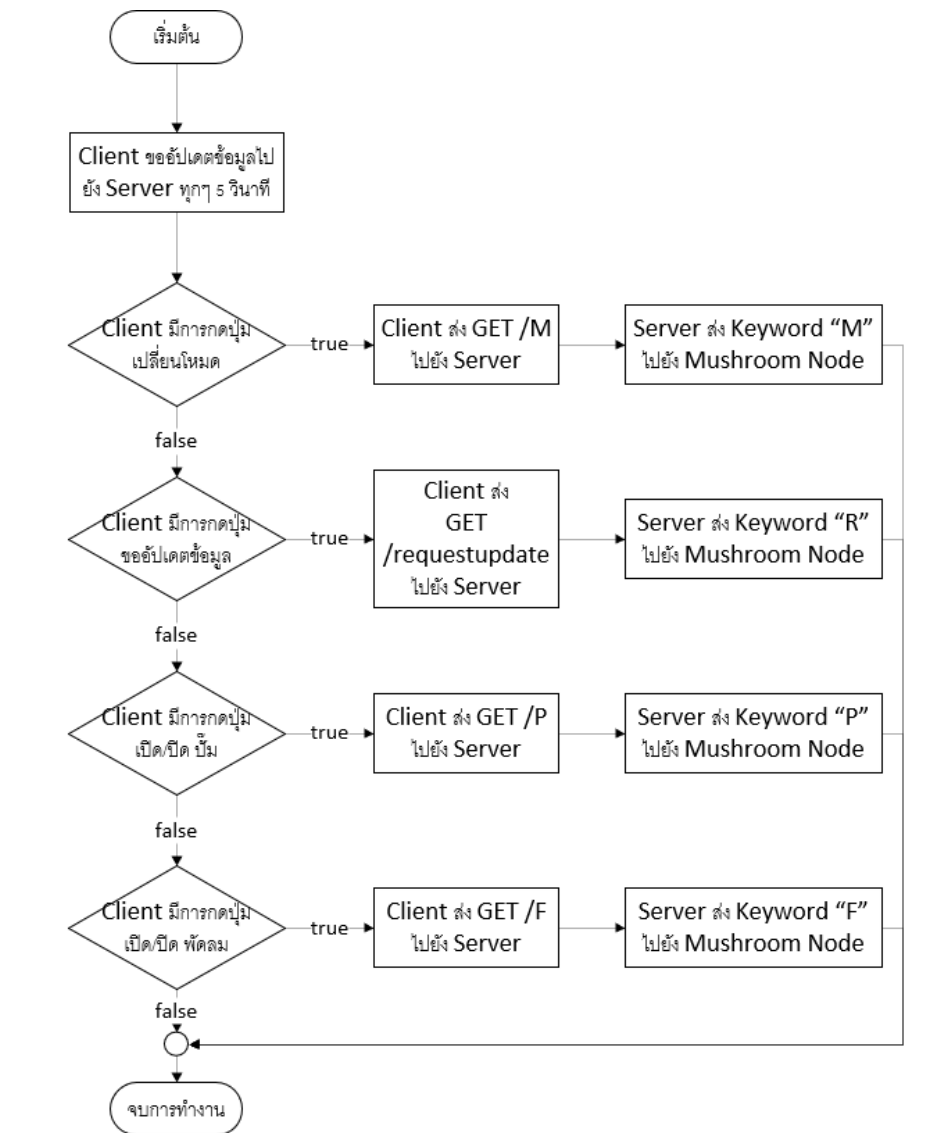
ในโครงงานนี้ผู้จัดทำได้ใช้ภาษา HTML CSS Javascript และ PHP ในการสร้าง Web Application และใช้แอปพลิเคชัน ngrok ที่ติดตั้งบน Raspberry Pi ทำ Port forwarding ของ Web Application เพื่อให้สามารถใช้งานเว็บแอปพลิเคชันได้จากทุกที่ที่สามารถใช้งานอินเทอร์เน็ตได้ Web Application ส่วนที่แสดงค่าสถานะต่างๆและควบคุมการทำงานของระบบจะถูกเก็บอยู่ที่ ESP32LoRa และ Web Application ส่วนที่เกี่ยวกับฐานข้อมูล เช่น บันทึกข้อมูลลงในฐานข้อมูล นำข้อมูลในฐานข้อมูลมาแสดงบน Web Application จะถูกเก็บอยู่ที่ Raspberry Pi 3 Model B เพราะใช้ Raspberry Pi เป็นฐานข้อมูลดังรูปที่ 3.58



รูปที่ 3.58 แสดงภาพโครงสร้างการทำงานของ Web Application

3.7.1 การสื่อสารระหว่าง Server และ Client

การสื่อสารระหว่าง Server และ Client เป็นการสื่อสารผ่านทาง Hypertext Transfer Protocol (HTTP) โดยมีรายละเอียดการทำงานดังรูปที่ 3.59



รูปที่ 3.59 แสดงภาพ Flowchart การสื่อสารระหว่าง Server และ Client

จากรูปที่ 3.59 แสดงภาพ Flowchart การสื่อสารระหว่าง Server และ Client โดยมีรายละเอียดการทำงานดังต่อไปนี้

เริ่มต้นการทำงาน Client จะส่ง HTTP GET Method เพื่อขออัปเดตข้อมูลทุกๆ 5 วินาที เพื่อให้หน้าเว็บมีการอัปเดตข้อมูลโดยใช้ฟังก์ชัน setInterval() ในภาษา JavaScript จากนั้นใช้ Ajax (Asynchronous JavaScript And XML) ในการส่ง HTTP GET Method ไปยัง Server เมื่อ Server ตอบกลับมานำข้อมูลที่ได้ไปแสดงบนหน้าเว็บ 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 3.60 แสดงภาพตัวอย่างโปรแกรมส่วนที่ใช้ในการขออัปเดตข้อมูลทุกๆ 5 วินาที

จากรูปที่ 3.60 เป็นตัวอย่างการขออัปเดตค่าอุณหภูมิจาก Server ทุกๆ 5 วินาทีเพื่อให้ค่าอุณหภูมิที่แสดงบนหน้าเว็บมีการอัปเดต นอกจากนี้ยังมีค่าต่างๆที่ขออัปเดตทุกๆ 5 วินาทีเหมือนค่าอุณหภูมิ เช่น อุณหภูมิต่ำสุด อุณหภูมิสูงสุด ความชื้น ความชื้นต่ำสุด ความชื้นสูงสุด ความสว่าง ความสว่างที่กำหนด สถานะการทำงานของปั๊มและพัดลม โหมดการทำงาน ดังรูปที่ 3.61





Mushroom Web Server


 Temperature **28.50 °C**
Min 26.00 °C Max 30.00 °C


 Humidity **93.50 %**
Min 80.00 % Max 90.00 %


 Light **245.24 lx**
Set Light 200.00 lx

 Pump **OFF**

 Fan **ON**

 Mode **AUTO**

 change mode

 request update

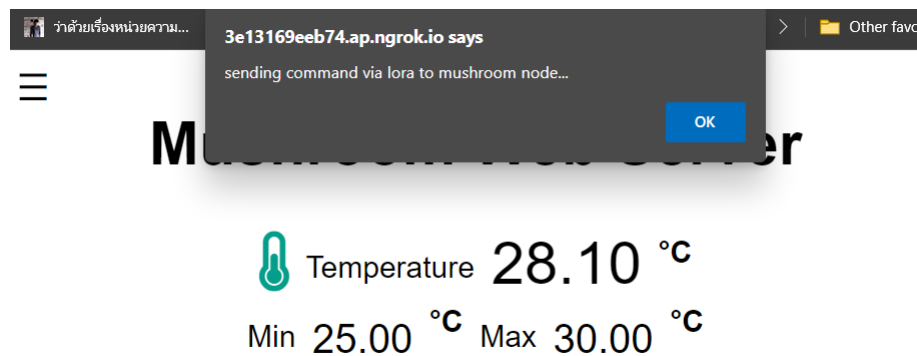
รูปที่ 3.61 แสดงภาพข้อมูลต่างๆบนหน้าเว็บที่อัปเดตทุกๆ 5 วินาที

จากรูปด้านบนแสดงค่าตัวเลขและสถานะต่างๆ Client ขออัปเดตไปยัง Server ทุกๆ 5 วินาที ขั้นตอนต่อไปจะทำการตรวจสอบว่ามีการความคุมการทำงานของระบบหรือไม่โดยมีรายละเอียดดังต่อไปนี้

```
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 3.62 แสดงภาพฟังก์ชันที่ถูกเรียกใช้เมื่อกดปุ่มเปลี่ยนโหมดการทำงาน

จากรูปที่ 3.62 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 ว่าส่งข้อมูลสำเร็จจะแจ้งเตือนดังรูปที่ 3.63



รูปที่ 3.63 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 3.64 แสดงภาพฟังก์ท์ที่ถูกเรียกใช้เมื่อกดปุ่มขออัปเดตข้อมูล

จากรูปที่ 3.64 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 3.65 แสดงภาพฟังก์ชันที่ถูกเรียกใช้เมื่อกดปุ่ม เปิด/ปิด ปั้ม

จากรูปที่ 3.65 ถ้า Client กดปุ่มควบคุมการทำงานของปั้ม (control PUMP) จะเรียกใช้ฟังก์ชัน toggleCheckbox_pump(element) ที่เขียนด้วยภาษา JavaScript โดยฟังก์ชันมีการทำงานดังนี้ เมื่อมีการเรียกใช้จะส่ง HTTP GET Method “/P” ไปยัง Server โดยใช้ Ajax จากนั้น Server จะส่ง Keyword “P” ผ่าน Lora ไปยัง Mushroom Node เพื่อควบคุมการทำงานของปั้ม

```

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();
}

```

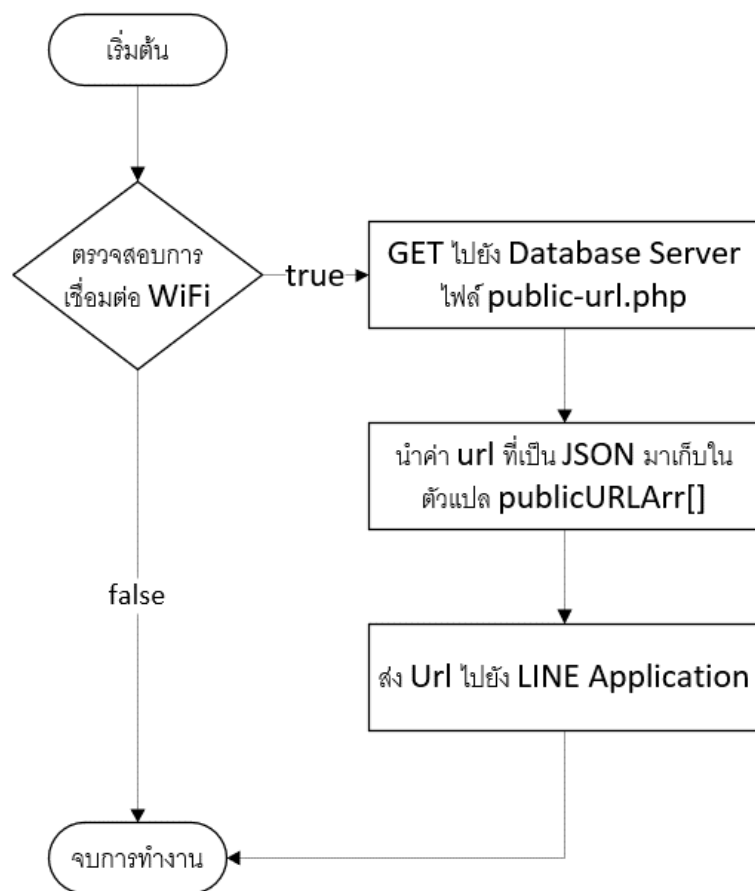
รูปที่ 3.66 แสดงภาพฟังก์ชันที่ถูกเรียกใช้เมื่อกดปุ่ม เปิด/ปิด พัดลม

จากรูปที่ 3.66 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 Application และ Database Server ไปยัง LINE Application

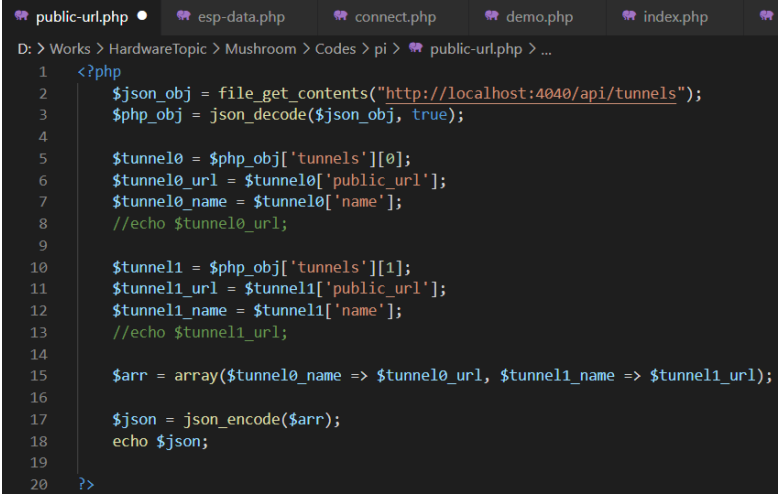
เพื่อความสะดวกในการเข้าถึงเว็บแอปพลิเคชัน (Web Application) เพราะเมื่อใช้โปรแกรม Ngrok ทำ Port forwarding แล้วจะได้ random url มา url จะทำการเปลี่ยนไปทุกครั้ง เมื่อมีการปิดหรือเปิดใช้งาน Ngrok ตัวอย่าง url ที่ได้จากการ random คือ <https://e9d707f7cc96.ap.ngrok.io/> ช่องทางที่ผู้ใช้สามารถดู url ที่ได้ 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 เพื่อให้สะดวกต่อการเข้าถึงโดยมีขั้นตอนดังรูปที่ 3.67



รูปที่ 3.67 แสดงภาพ Flowchart ขั้นตอนการส่ง url ไปยัง LINE Application

จากรูป 3.67 แสดงภาพ Flowchart ขั้นตอนการส่ง url ไปยัง LINE Application โดยมีรายละเอียดดังต่อไปนี้ ขั้นแรกตรวจสอบการเชื่อมต่อ WiFi ถ้าเชื่อมต่อแล้วจะส่ง HTTP GET Method ไปยังไฟล์ public-url.php ที่อยู่ใน Database Server ตัวอย่างที่อยู่ไฟล์ 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



```
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 3.68 แสดงภาพไฟล์ public-url.php

จากรูปที่ 3.68 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 3.69


```

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;
    }

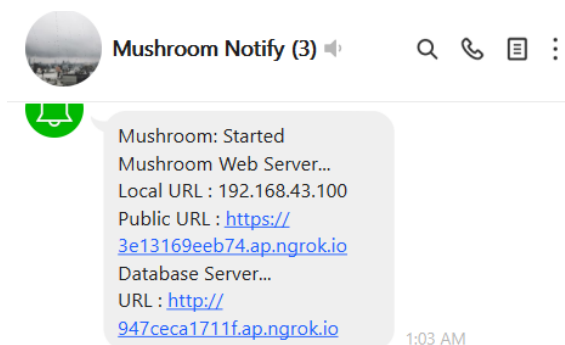
    Serial.print("JSON object = ");
    Serial.println(myObject);

    // 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 3.71 แสดงภาพโปรแกรมส่วนที่จัดการจัด url ที่ได้จากโปรแกรม Ngrok
เมื่อได้ url ของ Web Application และ Database Server แล้วจะส่ง url ที่ได้ไปยัง
LINE Application โดยใช้ฟังก์ชัน NotifyLine()



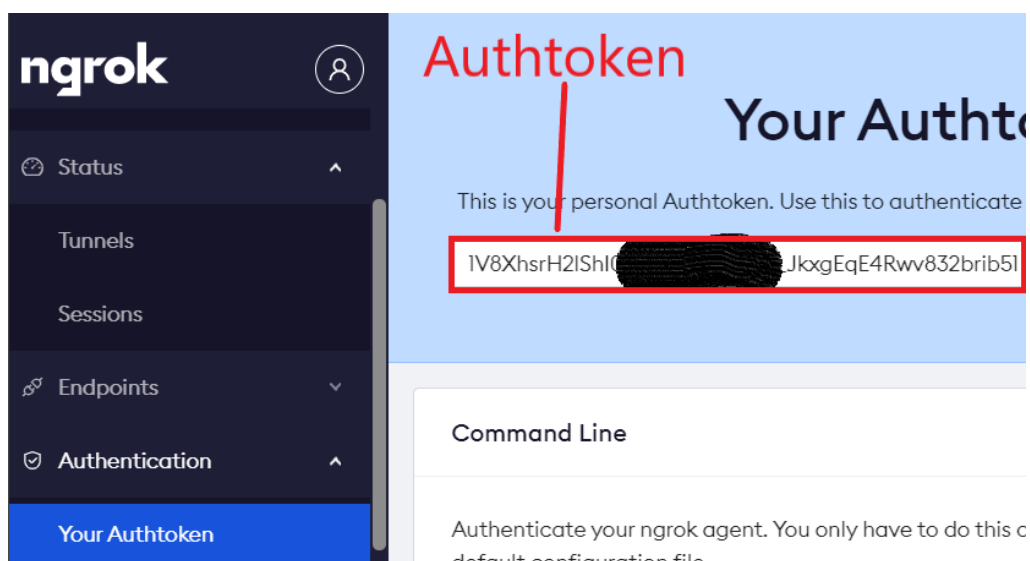
รูปที่ 3.72 แสดงภาพตัวอย่าง url ของ Web Server และ Database Server ที่ส่งไปยัง
LINE Application

3.7.3 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

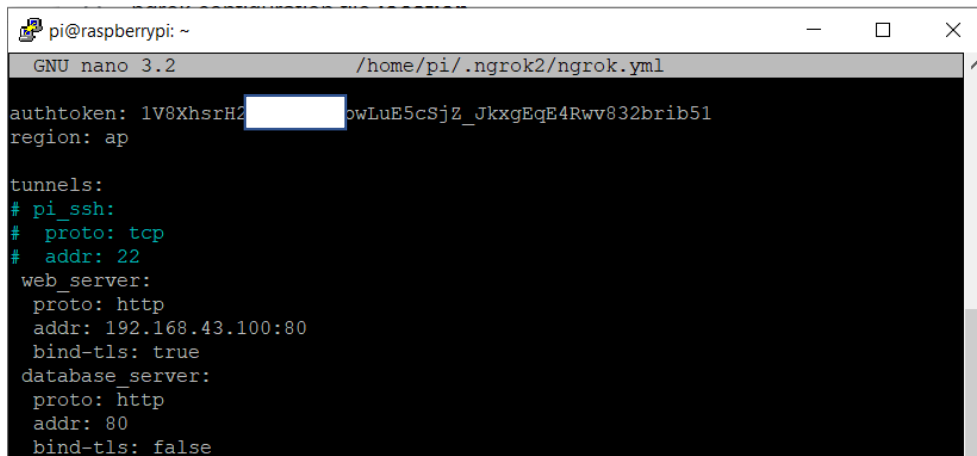
ในโครงการนี้ผู้จัดทำได้ใช้โปรแกรม Ngrok ที่ติดตั้งบน Raspberry Pi ในการทำ Port forwarding โดยมีรายละเอียดดังต่อไปนี้

ขั้นตอนที่ 1 สมัครสมาชิกกับ Ngrok เพื่อนำ authtoken ที่ได้มากรอกใน ngrok configuration file ให้สามารถใช้งานโปรแกรมได้ตลอดเวลา



รูปที่ 3.73 แสดงภาพ Authtoken ที่ได้จากการสมัครสมาชิก

ขั้นตอนที่ 2 แก้ไขไฟล์ ngrok configuration file

A screenshot of a terminal window on a Raspberry Pi. The window title is 'pi@raspberrypi: ~'. The terminal shows the GNU nano 3.2 editor editing the file '/home/pi/.ngrok2/ngrok.yml'. The configuration file content is as follows:

```
authtoken: 1V8XhsrH2[REDACTED]pwLuE5cSjZ_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 3.74 แสดงภาพ ไฟล์ ngrok configuration file

จากรูปที่ 3.74 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) กำหนด เฉพาะ Port เช่น addr: 80 หมายความว่าต้องการทำ Port forwarding IP Address ของเครื่องที่ติดตั้งโปรแกรม Ngrok ซึ่งก็คือ Raspberry Pi

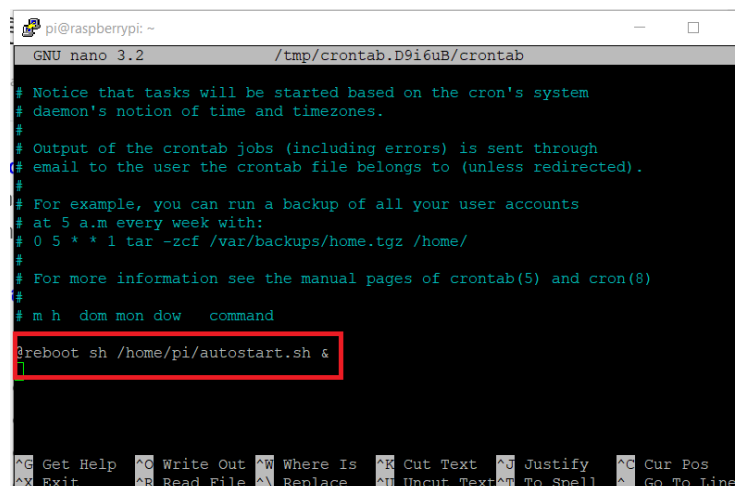
ขั้นตอนที่ 3 สร้างไฟล์ autostart.sh เพื่อใช้ในการ run command `./ngrok start -all` `./ngrok start -all` เป็นคำสั่งให้โปรแกรม Ngrok เริ่มการทำงานจาก ngrok configuration file



```
pi@raspberrypi: ~
GNU nano 3.2 autostart.sh
./ngrok start --all
```

รูปที่ 3.75 แสดงภาพไฟล์ autostart.sh

ขั้นตอนที่ 4 ใช้ Crontab ในการ run คำสั่งที่อยู่ในไฟล์ autostart.sh Crontab คือคำสั่งที่จะทำงานตามเวลาที่กำหนด



```
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 &
```

รูปที่ 3.76 แสดงภาพการใช้ Crontab ในการ run คำสั่ง

จากรูปที่ 3.76 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 Ngrok 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

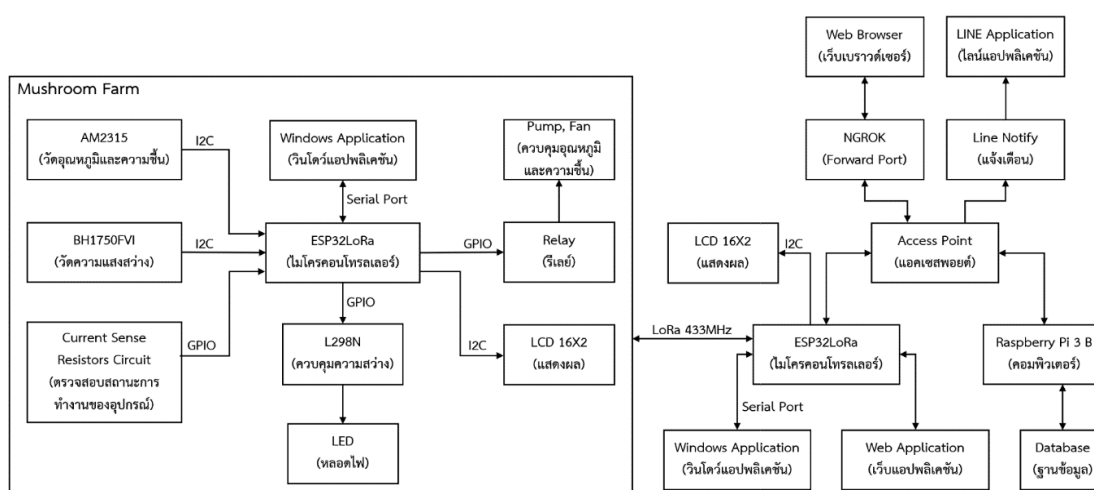
รูปที่ 3.77 แสดงภาพผลลัพธ์ที่ได้จากการทำ Port forwarding

รูปที่ 3.77 คือผลลัพธ์ที่ได้จากการทำ Port forwarding โดย url แรกคือ Public url ของ Web Application และ url ที่สองคือ Public url ของ Database Server

3.8 การออกแบบและสร้างวินโดวแอปพลิเคชัน

ในโครงงานนี้ผู้จัดทำได้มีวัตถุประสงค์ในการนำวินโดวแอปพลิเคชัน (Windows Application) มาประยุกต์ใช้ภายในโครงงาน ใช้ภาษา Visual Basic ของ .NET Core ในการสร้างวินโดวแอปพลิเคชัน (Windows Application)

วินโดวแอปพลิเคชัน (Windows Application) สามารถใช้งานได้กับ ESP32LoRa ได้ทั้งสองตัว ดังรูปที่ 3.78 โดยสื่อสารกันผ่าน Serial Port



รูปที่ 3.78 แสดงภาพโครงสร้างของระบบ

จากรูปที่ 3.78 จะเห็นว่าวินโดวแอปพลิเคชัน (Windows Application) สามารถใช้งานได้กับ ESP32LoRa (Mushroom Node) และ ESP32LoRa (STA Node) โดยใช้ทำงานผ่านทาง Serial Port

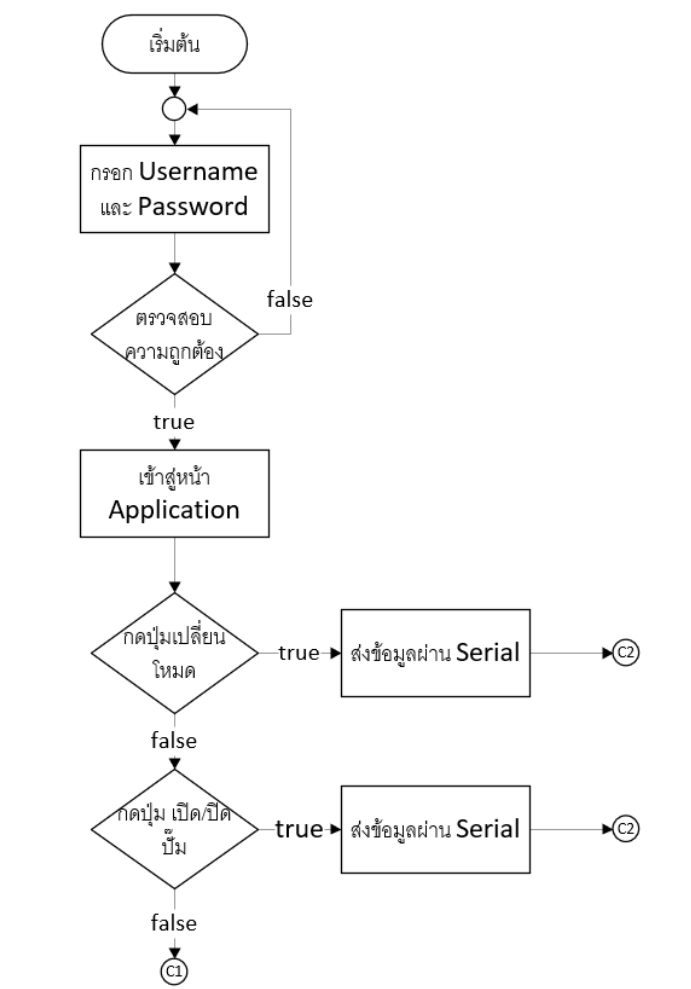
หากใช้งานกับวินโดวแอปพลิเคชัน (Windows Application) กับ ESP32LoRa (Mushroom Node) เมื่อควบคุมการทำงาน เช่น กดปุ่มเปลี่ยนโหมดการทำงาน วินโดวแอปพลิเคชัน (Windows Application) จะส่งข้อมูลผ่าน Serial Port มายัง ESP32LoRa จากนั้น ESP32LoRa จะนำข้อมูลที่ได้นำมาประมวลผลตามข้อมูลที่ได้

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

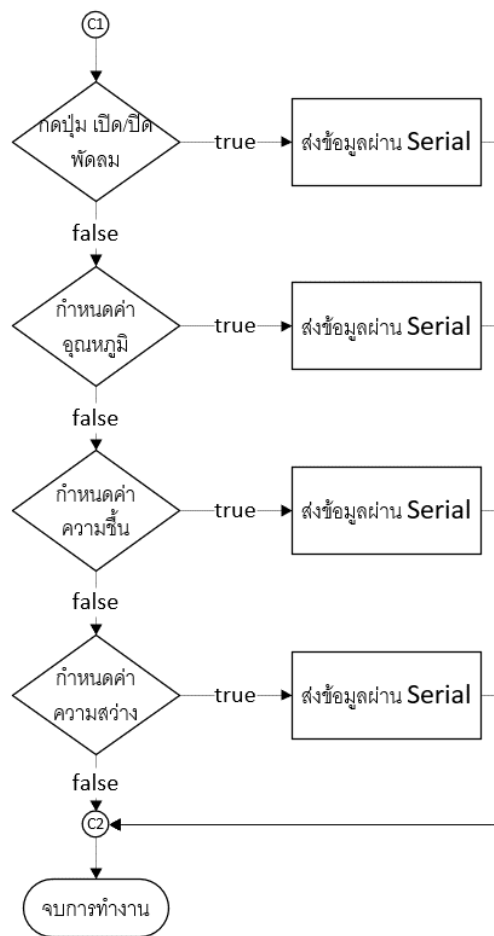
Application) จะส่งข้อมูลผ่าน Serial Port มายัง ESP32LoRa จากนั้น ESP32LoRa จะส่งข้อมูลที่ได้จาก Serial Port ผ่าน LoRa ไปยัง ESP32LoRa (Mushroom Node) จากนั้นจะนำข้อมูลที่ได้มาประมวลผล

3.8.1 การทำงานของวินโดว์แอปพลิเคชัน (Windows Application)

ขั้นตอนการทำงานของวินโดว์แอปพลิเคชัน (Windows Application) มีรายละเอียดการทำงานดังรูปที่แสดงด้านล่าง

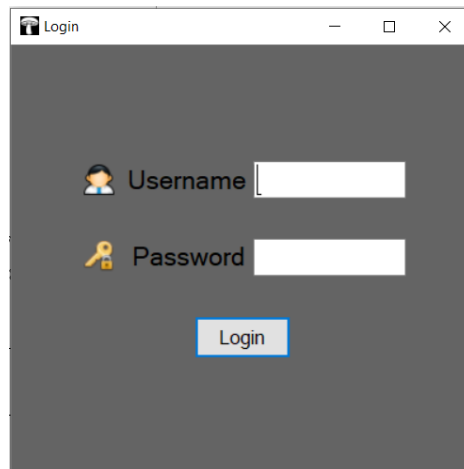


รูปที่ 3.79 แสดงภาพ Flowchart การทำงานของวินโดว์แอปพลิเคชัน (1)



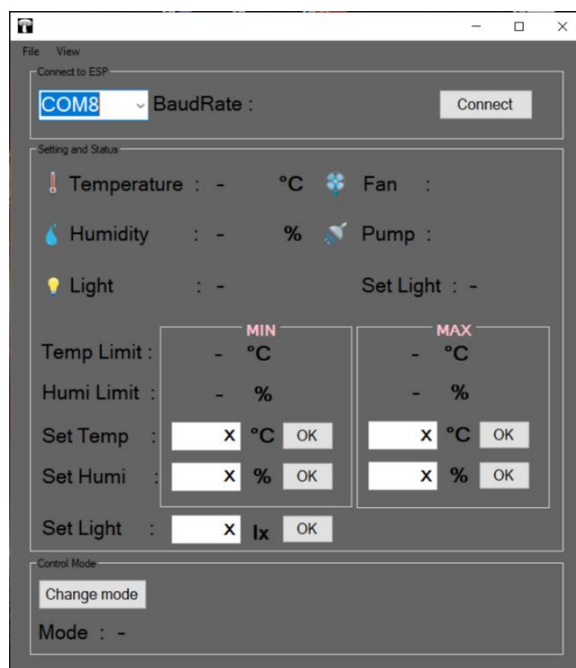
รูปที่ 3.80 แสดงภาพ Flowchart การทำงานของวินโดว์แอปพลิเคชัน (2)

จากรูปที่ 3.79 และ 3.80 แสดงภาพ Flowchart การทำงานของวินโดว์แอปพลิเคชันโดยมีรายละเอียดดังต่อไปนี้ เมื่อแอปพลิเคชันเริ่มต้นทำงานจะให้กรอก Username และ Password ก่อนเพื่อป้องกันความปลอดภัย ดังรูปที่ 3.81



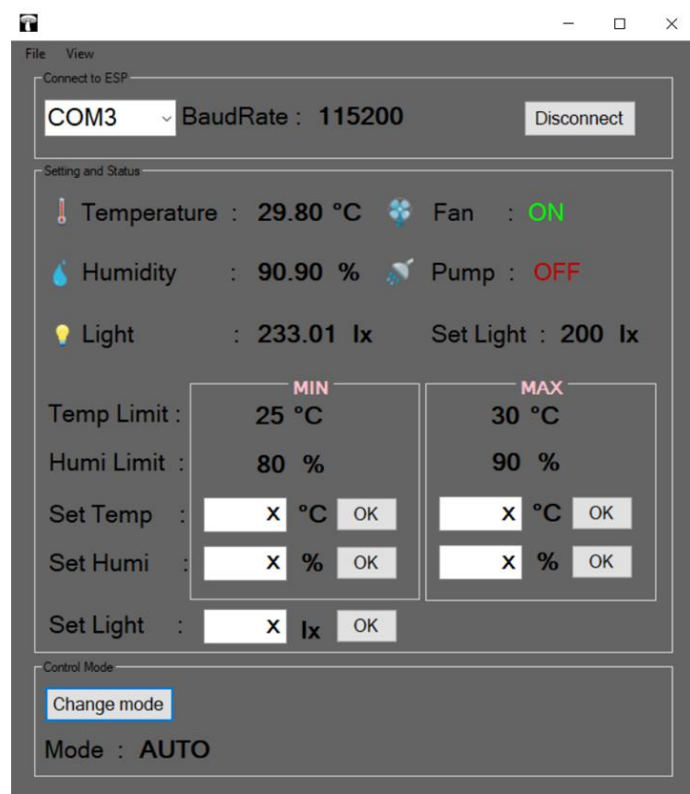
รูปที่ 3.81 แสดงภาพหน้า Login ของวินโดว์แอปพลิเคชัน

เมื่อทำการ Login สำเร็จจะแสดงหน้าแรกของวินโดว์แอปพลิเคชันดังรูปที่แสดงด้านล่าง



รูปที่ 3.82 แสดงภาพวินโดว์แอปพลิเคชันเมื่อยังไม่ได้เชื่อมต่อกับ Serial Port

จากรูปที่ 3.82 แสดงภาพของวินโดว์แอปพลิเคชันเมื่อยังไม่เชื่อมต่อกับ Serial Port เมื่อทำการเชื่อมต่อแล้วจะแสดงค่าสถานะต่างๆ ของโรงเรือนดังรูปที่แสดงด้านล่าง



รูปที่ 3.83 แสดงภาพวินโดว์แอปพลิเคชันเมื่อเชื่อมต่อกับ Serial Port แล้ว

จากรูปที่ 3.83 แสดงภาพวินโดว์แอปพลิเคชันเมื่อเชื่อมต่อกับ Serial Port แล้ว ESP32LoRa จะส่งข้อมูลผ่าน Serial Port มายังวินโดว์แอปพลิเคชันโดยส่งฟังก์ชัน Serial.print() ตัวอย่างข้อมูลคือ T30.50H78.703075M0P1F02685L200.33200 จากนั้นวินโดว์แอปพลิเคชันจะนำข้อมูลที่ได้ประมวลผลดังรูปที่ 3.84 แล้วนำมาแสดงบนหน้าวินโดว์แอปพลิเคชัน

```

temp = line.Substring(1, 5)
Label17.Text = temp

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

set_temp_max = line.Substring(12, 2)
Label37.Text = set_temp_max

set_humi_min = line.Substring(14, 2)
Label22.Text = set_humi_min

ctrlMode = line.Substring(17, 1)
pumpState = line.Substring(19, 1)
fanState = line.Substring(21, 1)

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

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

Dim lux_raw = line.Substring(27, 7)
Dim I = lux_raw.Substring(0, 1)
Dim II = lux_raw.Substring(0, 2)
Dim III = lux_raw.Substring(0, 3)
If III = "000" Then
    lux = lux_raw.Substring(3, 4) '1 3 2
ElseIf II = "00" Then
    lux = lux_raw.Substring(2, 5)
ElseIf I = "0" Then
    lux = lux_raw.Substring(1, 6)
Else
    lux = lux_raw
End If
LabelLux.Text = lux + " lx"

Dim set_lux_raw = line.Substring(34, 3)
Dim O = set_lux_raw.Substring(0, 1)
If O = "0" Then
    set_lux = set_lux_raw.Substring(1, 2)
Else
    set_lux = set_lux_raw
End If
LabelSetLux.Text = set_lux + " lx"

```

รูปที่ 3.84 แสดงภาพส่วนของโปรแกรม Visual Basic ที่จัดการกับข้อมูลที่ส่งมาจาก Serial Port

ขั้นตอนต่อมาหลังจากที่จัดการกับข้อมูลที่ส่งมาจาก Serial Port แล้วจะตรวจสอบว่ามี การควบคุมการทำงานของระบบหรือไม่โดยมีรายละเอียดดังต่อไปนี้

เมื่อกดปุ่มเปลี่ยนโหมดการทำงาน (Change mode) วินโดว์แอปพลิเคชันจะส่ง Keyword “M” ผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("M") เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.85

```

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 3.85 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 3.86

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

รูปที่ 3.86 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 3.87

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

รูปที่ 3.87 แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกดปุ่ม เปิด/ปิด พัดลม

เมื่อกำหนดค่าอุณหภูมิต่ำสุดวินโดว์แอปพลิเคชันจะส่ง Keyword “Y” + ข้อมูลที่กรอกผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("Y" + TextBox1.Text) เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.88

```
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 3.88 แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกำหนดค่าอุณหภูมิต่ำสุด

เมื่อกำหนดค่าอุณหภูมิสูงสุดวินโดว์แอปพลิเคชันจะส่ง Keyword “T” + ข้อมูลที่กรอกผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("T" + TextBox4.Text) เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.89

```
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 3.89 แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกำหนดค่าอุณหภูมิสูงสุด

เมื่อกำหนดค่าความชื้นต่ำสุดวินโดว์แอปพลิเคชันจะส่ง Keyword “H” + ข้อมูลที่กรอกผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("H" + TextBox2.Text) เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.90

```
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 3.90 แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกำหนดค่าความชื้นต่ำสุด

เมื่อกำหนดค่าความชื้นสูงสุดวินโดว์แอปพลิเคชันจะส่ง Keyword “J” + ข้อมูลที่กรอกผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("J" + TextBox3.Text) เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.91


```

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 3.91 แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกำหนดค่าความชื้นสูงสุด

เมื่อกำหนดค่าความสว่างวินโดว์แอปพลิเคชันจะส่ง Keyword “L” + ข้อมูลที่กรอก ผ่าน Serial Port ไปยัง ESP32LoRa โดยใช้ฟังก์ชัน SerialPort1.Write("L" + TextBoxSetLux.Text) เพื่อให้ ESP32LoRa นำ Keyword ที่ได้ไปประมวลผลต่อไปดังรูปที่ 3.92

```

If TextBoxSetLux.TextLength >= 2 Then
    SerialPort1.Write("L" + TextBoxSetLux.Text)
    TextBoxSetLux.Clear()
Else
    MessageBox.Show("The value should be between 10-999 lx.", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning)
End If

```

รูปที่ 3.92 แสดงภาพการส่งข้อมูลผ่าน Serial Port ไปยัง ESP32LoRa เมื่อกำหนดค่าความสว่าง

3.9 การออกแบบฐานข้อมูลและการบันทึกข้อมูล

3.9.1 การออกแบบฐานข้อมูล

ในการออกแบบฐานข้อมูลในโครงการนี้จะใช้แค่ 1 ตาราง (entity) เท่านั้นเพื่อใช้เก็บข้อมูลต่างๆที่เกี่ยวข้องเช่น ค่าอุณหภูมิ (temp) ค่าความชื้น (humi) ค่าความสว่าง (lux) ค่าอุณหภูมิต่ำสุด (temp_limit_min) ค่าอุณหภูมิสูงสุด (temp_limit_max) ค่าความชื้นต่ำสุด (humi_limit_min) ค่าความชื้นสูงสุด (humi_limit_max) ค่าความสว่างที่กำหนด (set_lux) โหมดการทำงาน (ctrl_mode) สถานะการทำงานของปั๊ม (pump_state) สถานการณ์ทำงานของพัดลม (fan_state) และเวลาที่บันทึกข้อมูล (save_time) ดังรูปที่ 3.93 โดยตารางมีชื่อว่า SensorData

#	Name	Type	Collation	Attributes	Null	Default
1	id 🗝️	int(11)			No	None
2	temp	varchar(5)	utf8mb4_general_ci		Yes	NULL
3	humi	varchar(5)	utf8mb4_general_ci		Yes	NULL
4	lux	varchar(5)	utf8mb4_general_ci		Yes	NULL
5	temp_limit_min	varchar(2)	utf8mb4_general_ci		Yes	NULL
6	temp_limit_max	varchar(2)	utf8mb4_general_ci		Yes	NULL
7	humi_limit_min	varchar(2)	utf8mb4_general_ci		Yes	NULL
8	humi_limit_max	varchar(2)	utf8mb4_general_ci		Yes	NULL
9	set_lux	varchar(5)	utf8mb4_general_ci		Yes	NULL
10	ctrl_mode	varchar(6)	utf8mb4_general_ci		Yes	NULL
11	pump_state	varchar(3)	utf8mb4_general_ci		Yes	NULL
12	fan_state	varchar(3)	utf8mb4_general_ci		Yes	NULL
13	save_time	timestamp			No	current_timestamp()

รูปที่ 3.93 แสดงภาพโครงสร้างของฐานข้อมูล

จากรูปที่ 3.93 โครงสร้างของฐานข้อมูลโดยมี Primary key (PK) คือ id สามารถบันทึกข้อมูลเป็นตัวเลขจำนวนเต็มแบบไม่มีเครื่องหมายไม่เกิน 6 หลัก เป็นลำดับของข้อมูลแบบเพิ่มขึ้นอัตโนมัติ (auto increment) มี save_time attribute ใช้ในการเก็บค่าวันที่บันทึกข้อมูล และ #temp #humi #lux #temp_limit_min #temp_limit_max #humi_limit_min #humi_limit_max #set_lux #ctrl_mode #pump_state #fan_state attribute เป็น attribute ที่ใช้ในการเก็บข้อมูลต่างๆภายในระบบ

ตารางที่ 3.9 แสดง Data Dictionary SensorData

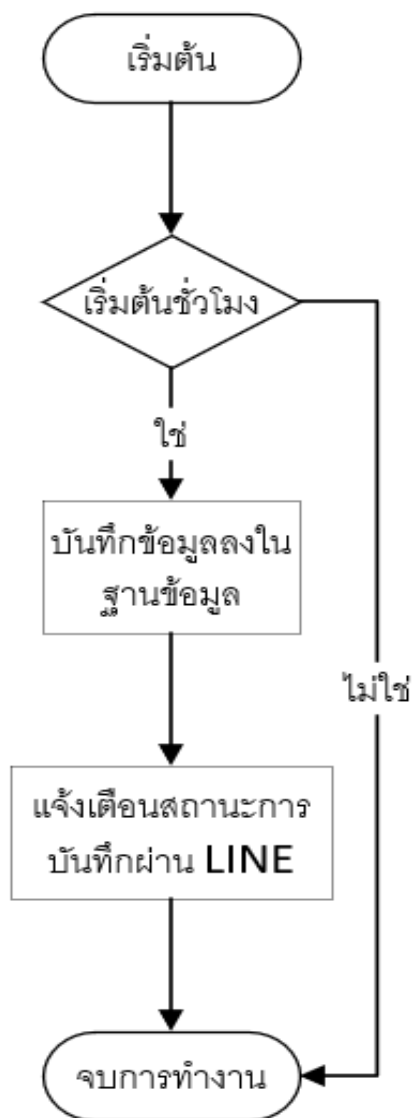
Attribute	Description	Type	Example	Key
id	ลำดับของข้อมูล	int(6)	1	PK
temp	อุณหภูมิ	varchar(5)	32.10	
humi	ความชื้น	varchar(5)	80.70	
lux	ความสว่าง	varchar(6)	213.22	
temp_limit_min	อุณหภูมิต่ำสุด	varchar(2)	20	
temp_limit_max	อุณหภูมิสูงสุด	varchar(2)	29	
humi_limit_min	ความชื้นต่ำสุด	varchar(2)	80	
humi_limit_max	ความชื้นสูงสุด	varchar(2)	85	
Set_lux	ความสว่างที่กำหนด	varchar(5)	200	
ctrl_mode	โหมดการทำงาน	varchar(6)	AUTO	
pump_state	สถานะการทำงานของปั๊ม	varchar(3)	ON	
fan_state	สถานะการทำงานของพัดลม	varchar(3)	ON	
save_time	วันเวลาที่บันทึกข้อมูล	timestamp	2020-09-08 18:00:00	

id	temp	humi	lux	temp_limit_min	temp_limit_max	humi_limit_min	humi_limit_max	set_lux	ctrl_mode	pump_state	fan_state	save_time
141	24.40	99.20	0.00	25	30	85	90	200	AUTO	OFF	OFF	2020-10-18 15:01:00
142	24.00	99.90	0.00	25	30	85	90	200	AUTO	OFF	OFF	2020-10-18 15:01:00
143	23.90	99.90	0.00	25	30	85	90	200	AUTO	OFF	OFF	2020-10-18 15:01:00
144	23.70	99.90	0.00	25	30	85	90	200	AUTO	OFF	OFF	2020-10-18 15:01:00
145	23.70	99.90	0.00	25	30	85	90	200	AUTO	OFF	OFF	2020-10-18 15:01:00
146	23.60	99.90	0.00	25	30	85	90	200	AUTO	OFF	OFF	2020-10-18 15:01:00
147	23.50	99.90	0.00	25	30	85	90	200	AUTO	OFF	OFF	2020-10-18 15:01:00
148	23.50	99.90	0.00	25	30	85	90	200	AUTO	OFF	OFF	2020-10-18 15:01:00
149	23.50	99.90	0.00	25	30	85	90	200	AUTO	OFF	OFF	2020-10-18 15:01:00

รูปที่ 3.94 แสดงภาพตัวอย่างข้อมูลเก็บอยู่ภายในฐานข้อมูล

3.9.2 การบันทึกข้อมูลลงในฐานข้อมูล

ในการจัดทำโครงการนี้ผู้จัดทำได้ใช้ Raspberry pi 3 b เป็นฐานข้อมูลเพื่อเก็บค่าต่างๆ และใช้ไมโครคอนโทรลเลอร์ ESP32 LoRa เป็นเว็บเซิร์ฟเวอร์ (Web Server) โดยจะบันทึกข้อมูลลงในฐานข้อมูลทุกๆ ชั่วโมง เช่น 00.00, 01.00, 02.00, 12.00, 13.00 เป็นต้น ดัง Flowchart ด้านล่าง



รูปที่ 3.95 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 3.96 แสดงภาพการบันทึกข้อมูลลงในฐานข้อมูลทุกชั่วโมง

จากรูปที่ 3.96 จะเป็นการดึงค่าเวลาจริงมาจากเซิร์ฟเวอร์ NTP (Network Time Protocol) มาใช้ในการตรวจสอบเพื่อบันทึกข้อมูล โดยจะตรวจสอบเวลาในหลักนาทิจและวินาที เช่น ถ้าเวลาในหน่วยนาทิจมีค่าเท่ากับ 00 และเวลาในหน่วยวินาทีมีค่าเท่ากับ 00 ก็จะทำให้การบันทึกข้อมูลโดยเรียกใช้ฟังก์ชัน insertDB() ก็จะสามารถบันทึกข้อมูลทุกๆชั่วโมงได้แล้ว

การบันทึกข้อมูลจากเว็บเซิร์ฟเวอร์ (Web Server) ไปยังฐานข้อมูลที่อยู่บน Raspberry pi 3 b ทุกๆชั่วโมงโดยใช้ฟังก์ชัน insertDB() ที่เขียนด้วยภาษา C++ ดังภาพที่ 3.97

```

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 3.97 แสดงภาพฟังก์ชันที่ใช้บันทึกข้อมูลลงในฐานข้อมูล

จากรูปที่ 3.97 เป็นฟังก์ชันที่ทำหน้าบันทึกข้อมูลลงในฐานข้อมูลที่อยู่บน Raspberry pi 3 b โดยตัวแปรที่เป็นประเภท String ที่มีชื่อว่า serverName เป็นตัวแปรที่เก็บที่อยู่ของไฟล์ PHP ที่มีชื่อว่า post-esp-data.php ทำหน้าที่บันทึกข้อมูลลงในฐานข้อมูลที่อยู่บน Raspberry pi โดยที่อยู่นี้ได้จากการทำ port forwarding โดยใช้โปรแกรม Ngrok ตัวอย่างที่อยู่อยู่ <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 แสดงว่าบันทึกข้อมูลสำเร็จ จากนั้นทำการแจ้งเตือนสถานะการบันทึกข้อมูลทาง LINE แอปพลิเคชัน โดยเรียกใช้ฟังก์ชัน NotifyLine()

```

$api_key_value = "tPmAT5Ab3j7F9";

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

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"]);
        $lux = test_input($_POST["lux"]);
        $temp_limit_min = test_input($_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"]);
        $set_lux = test_input($_POST["set_lux"]);
        $ctrl_mode = test_input($_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, lux, temp_limit_min, temp_limit_max, humi_limit_min, humi_limit_max)
VALUES ('" . $temp . "', '" . $humi . "', '" . $lux . "', '" . $temp_limit_min . "', '" . $temp_limit_max . "', '" . $humi_limit_min . "', '" . $humi_limit_max . "')";

        if ($conn->query($sql) === TRUE) {
            echo "New record created successfully";
        }
    }
}

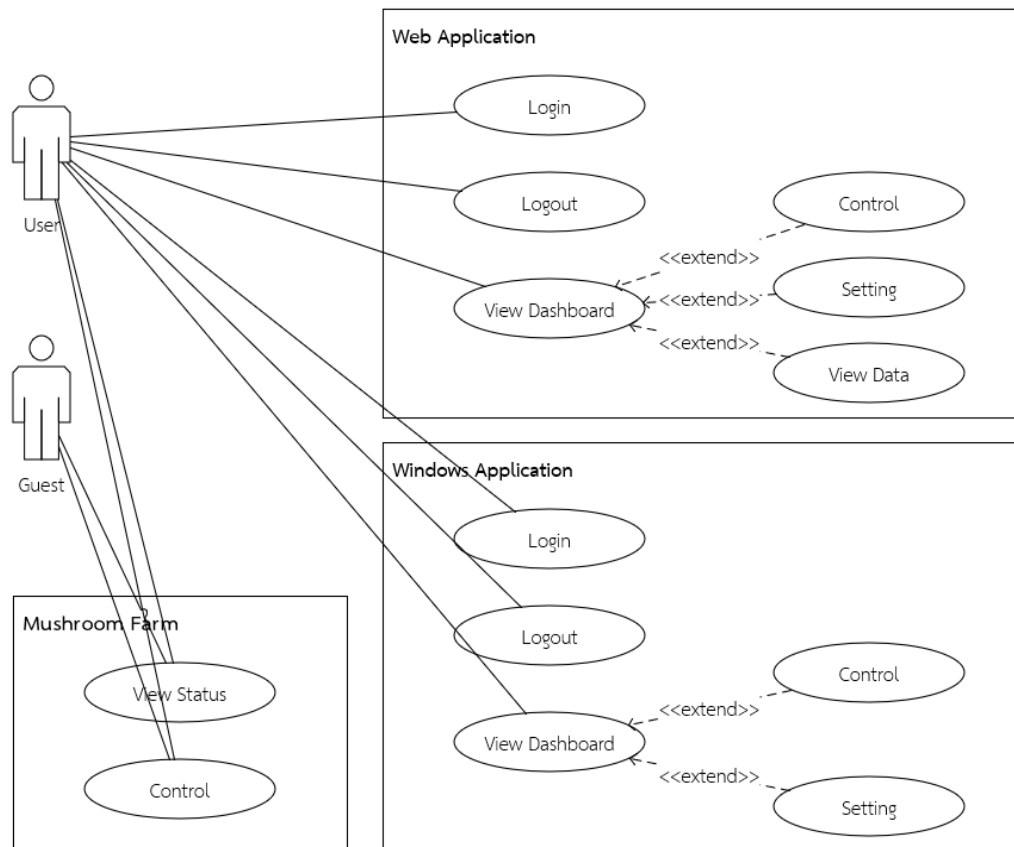
```

รูปที่ 3.98 แสดงภาพไฟล์ post-esp-data.php บางส่วน

จากรูปที่ 3.98 แสดง code บางส่วนของไฟล์ post-esp-data.php โดยมีตัวแปรที่มีชื่อว่า \$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 ถ้าตรงกันจะทำการเชื่อมต่อฐานข้อมูลแล้วบันทึกข้อมูล ถ้าไม่ตรงกันจะแสดงข้อความ "Wrong API Key provided." เพื่อเป็นการป้องกันฐานข้อมูลเบื้องต้น

3.10 การออกแบบ Use Case Diagram



รูปที่ 3.99 แสดงภาพ Use Case Diagram

รายละเอียด Use Case Diagram จากตารางที่ 3.10 ถึงตารางที่ 3.16 ด้านล่างต่อไปนี้
แสดงถึงรายละเอียดการใช้งานของแต่ละ Use Case ตามรูปแบบของ UML Use Case Description

ตารางที่ 3.10 แสดง Login Use Case

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

ตารางที่ 3.11 แสดง 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 3.12 แสดง View Dashboard Use Case

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

ตารางที่ 3.13 แสดง Control Use Case

Use Case Name	Control
Actors	User
Use Case Purpose	เพื่อใช้ในการควบคุมการทำงานของระบบ เช่น เปลี่ยนโหมดการทำงาน ควบคุมการเปิดปิดของอุปกรณ์
Pre-conditions	ใช้ร่วมกับ View Dashboard
Post-conditions	-
Main Course	1.กดปุ่ม change mode 2.ถ้าโหมดการทำงานแบบอัตโนมัติระบบจะควบคุมการทำงานของอุปกรณ์โดยอัตโนมัติเมื่อค่าอุณหภูมิและความชื้นต่ำกว่าหรือสูงกว่าที่กำหนด ถ้าโหมดการทำงานแบบแมนนวลผู้ใช้จะสามารถสั่งเปิดปิดอุปกรณ์ได้ตามต้องการ
Exceptions	เกิดความผิดพลาดภายในระบบ

ตารางที่ 3.14 แสดง Setting Use Case

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

ตารางที่ 3.15 แสดง 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 3.16 แสดง View Status Use Case

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