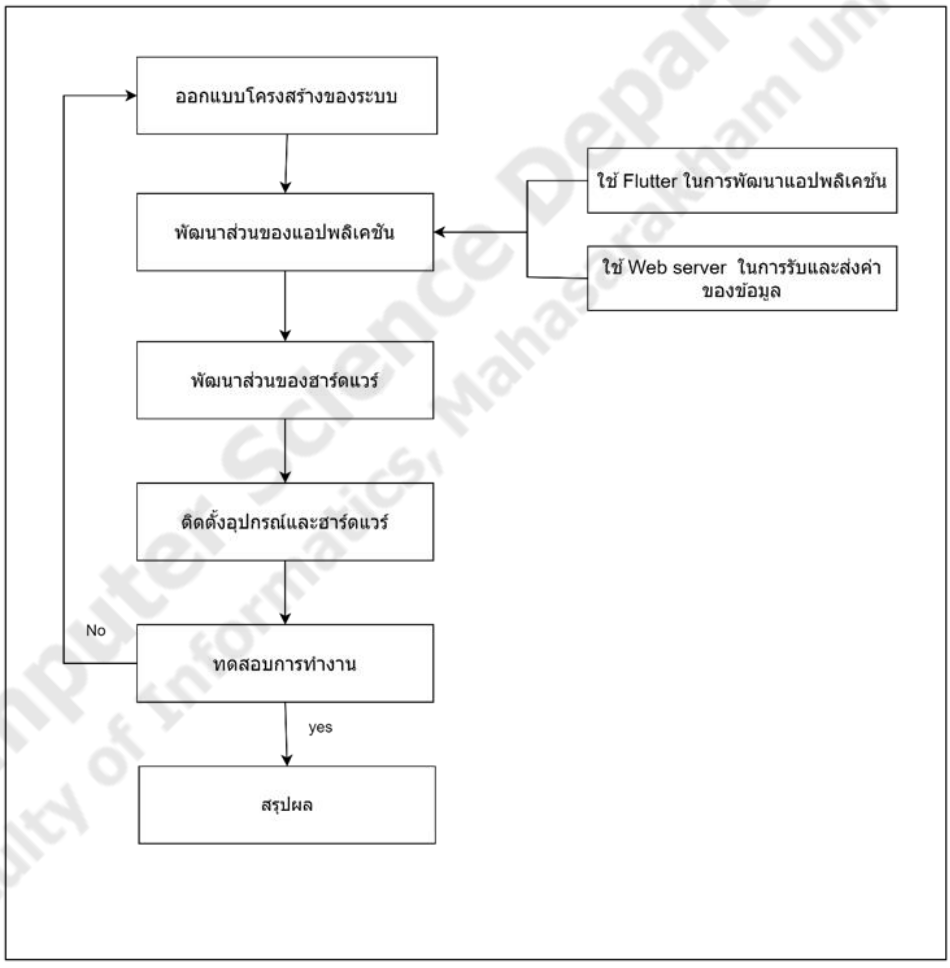
บทที่ 3

ขั้นตอนการดำเนินงาน

สำหรับในบทนี้จะกล่าวถึงขั้นตอนในการดำเนินงานของโครงงานปริญญานิพนธ์ ซึ่งทำให้ทราบถึงการวิเคราะห์และการออกแบบระบบโดยละเอียดว่ามีแนวทางในการทำงานหรือมีขั้นตอนในการทำงานของระบบอย่างไรบ้างโดยขั้นตอนในการดำเนินงานมีรายละเอียดดังนี้

3.1 กรอบการดำเนินงาน



ภาพประกอบที่ 3.1 กรอบการดำเนินงาน

3.1.1 คำอธิบาย

1. ออกแบบโครงสร้างของระบบ จะมีองค์ประกอบทั้งหมด 3 องค์ประกอบ คือ แอปพลิเคชัน ถังขยะIoT และ Server โดยถังขยะ IoT จะมีเซ็นเซอร์อยู่ซึ่งทำหน้าที่คอยรับค่าต่างๆแล้วส่งไปยัง

ฐานข้อมูล แอปพลิเคชันจะมีหน้าที่คอยรับค่าข้อมูลจากผู้ใช้นการเข้าระบบ และ ข้อมูลที่จะนำมาทำรายงานอัตราการเต็มของถังขยะ แล้วจากนั้นจะส่งขึ้นมูลไปยัง server ค้นหาข้อมูล และเอาข้อมูลมาเพื่อยืนยันข้อมูลผู้ใช้

2. พัฒนาส่วนแอปพลิเคชัน จะใช้ Flutter ซึ่งเขียนด้วยโปรแกรม Android studio ในการออกแบบแอปพลิเคชัน และหน้าที่ในส่วนของแอปพลิเคชัน โดยการนำข้อมูลจาก server มาแสดงผลในแอปพลิเคชัน

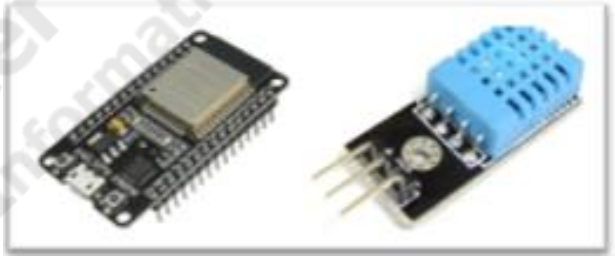
3. พัฒนาส่วนของเซ็นเซอร์ ทำการรับค่าข้อมูลจากเซ็นเซอร์ต่าง ๆ แล้วส่งข้อมูลขึ้นไปเก็บไว้ยังฐานข้อมูล MySQL เซ็นเซอร์ต่างๆ จะตั้งค่าให้ส่งไปยังฐานข้อมูลในทุกๆ 30 นาที

4. การติดตั้งอุปกรณ์และเซนเซอร์ ติดตั้งอุปกรณ์พร้อมกับเซนเซอร์ต่างๆ ลงไปยังถังขยะที่เตรียมไว้ โดยที่นำเซนเซอร์อัลตราโซนิก มาติดฝาไว้วัดระยะจากฝาปิดถึงก้นถัง มีระยะห่างเท่าไหร่เพื่อที่ได้กำหนดlevel ความจุของถัง และ เซนเซอร์อื่นก็ติดตั้งใต้ฝาปิดเพื่อป้องกันการโดนน้ำ เพราะอาจทำให้เซนเซอร์ทำงานผิดผลาดได้ และ ป้องกันความเสียหายของตัวเซนเซอร์ และ การเก็บขยะจะเก็บขยะด้วยวิธีการเปิดฝาถังที่มีเซ็นออกแล้วเทขยะออกจากถังขยะ เพื่อป้องกันการทำให้เซนเซอร์เกิดความเสียหายได้ โดยแหล่งพลังงานที่

3.1.2 เชื่อมต่อSensor 4 อย่างกับบอร์ดESP32

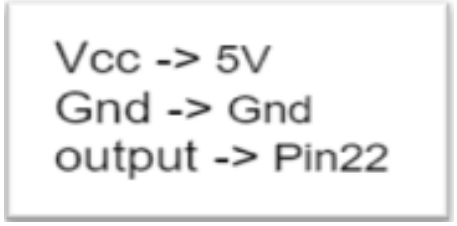
วิธีการต่อเซนเซอร์ทั้ง 4 อย่าง

1. เซนเซอร์วัดอุณหภูมิและความชื้น DHT11, ESP32

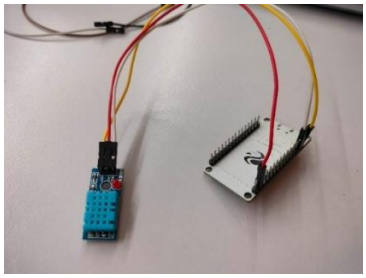


ภาพประกอบที่ 3.2 DHT11, ESP32

- วิธีการต่อ เซนเซอร์วัดอุณหภูมิและความชื้น DHT11, ESP32



ภาพประกอบที่ 3.3 ขาต่อ DHT11



ภาพประกอบที่ 3.4 การต่อ DHT11

- คำสั่งเซนเซอร์วัดอุณหภูมิและความชื้น DHT11

ภาพประกอบที่ 3.5 คำสั่งเซนเซอร์วัดอุณหภูมิและความชื้น

บรรทัดที่ 1 การเรียกใช้ library วัดอุณหภูมิและความชื้น DHT11

บรรทัดที่ 2 กำหนดตัวแปรที่ต่อกับขา Digital ขาที่ 22

บรรทัดที่ 3 ประกาศ type ตัวแปรของ DHT11

บรรทัดที่ 5 ประกาศตัวแปรเพื่อที่จะใช้งาน DHT

บรรทัดที่ 11 เป็นคำสั่งเปิดใช้งาน DHT

บรรทัดที่ 14 เป็นคำสั่งในการ แสดงค่าออกมาแบบวนลูป

บรรทัดที่ 15 เป็นคำสั่งก่อนอ่านค่าและแสดงผลครั้งถัดไป

บรรทัดที่ 16 เป็นคำสั่งในการดึงค่าความชื้นในอากาศเป็นเปอร์เซ็นต์

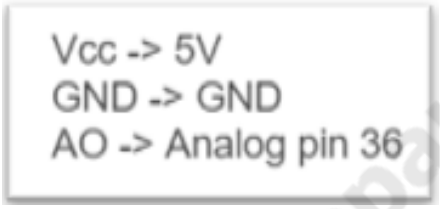
บรรทัดที่ 17 เป็นคำสั่งในการดึงค่าอุณหภูมิในอากาศมีหน่วยเป็นองศาเซลเซียส

บรรทัดที่ 19-23 เป็นการพิมพ์ค่า ความชื้นในอากาศและอุณหภูมิ

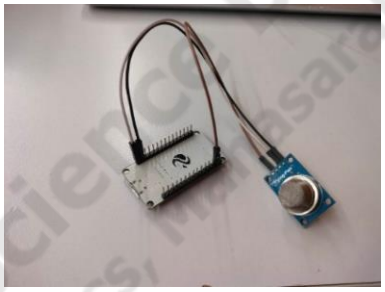
2. เซนเซอร์วัดคุณภาพอากาศโดยรวม MQ-135, ESP32

ภาพประกอบที่ 3.6 MQ-135, ESP32

- วิธีการต่อ เซนเซอร์วัดคุณภาพอากาศโดยรวม MQ-135, ESP32



ภาพประกอบที่ 3.7 ขาต่อ MQ-135



ภาพประกอบที่ 3.8 การต่อ MQ-135

- คำสั่งเซนเซอร์วัดคุณภาพอากาศโดยรวม MQ-135

MQ-135.ino

```
#include <MQUnifiedsensor.h>
#define PIN_MQ135 36

void setup() {
  Serial.begin(115200);
}
void loop() {
  int Air_Quality = analogRead(PIN_MQ135); // read the input on analog pin 0:
  Serial.print("MQ135 Air Quality: ");
  Serial.println(Air_Quality);
  if (Air_Quality < 400){
      Serial.println("\tAir Quality: Good ");
  }else if(Air_Quality > 400 && Air_Quality < 750){
      Serial.println("\tAir Quality: middle ");
  }
  else {
    Serial.println("\tAir Quality: Bad");
  }
  delay(60000);
}
```

ภาพประกอบที่ 3.9 คำสั่งเซนเซอร์วัดคุณภาพอากาศโดยรวม

บรรทัดที่ 1 การเรียกใช้ library ใช้Sensor MQ

บรรทัดที่ 2 กำหนดตัวแปรที่ต่อกับขาที่ 36

บรรทัดที่ 7 เป็นคำสั่งในการ แสดงค่าออกมาแบบวนลูป

บรรทัดที่ 8 ประกาศตัวแปรมาเก็บค่า Air Quality ที่ดึงข้อมูลมาจากตัวแปรที่ต่อกับขาที่ 36 ที่ได้มีการแปลงค่าจาก digital เป็น analog

บรรทัดที่ 9-10 เป็นการพิมพ์ค่า ที่ได้จากค่าข้อมูลมาจากตัวแปรที่ต่อกับขาที่ 36

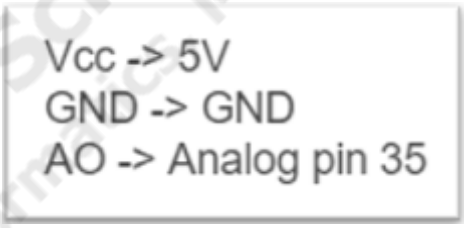
บรรทัดที่ 11-18 เป็นคำสั่งเช็คเงื่อนไขของค่า Air Quality จาก Sensor MQ-135

บรรทัดที่ 19 เป็นคำสั่งก่อนอ่านค่าและแสดงผลครั้งถัดไป

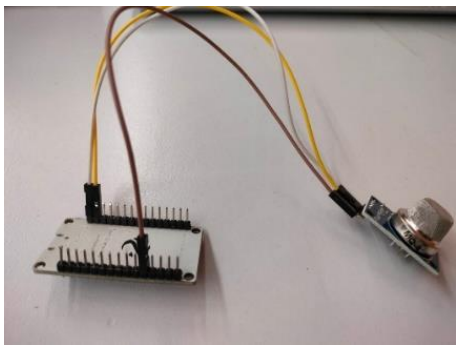
3. เซนเซอร์วัดก๊าซมีเทน MQ-4, ESP32

ภาพประกอบที่ 3.10 MQ-4, ESP326

- วิธีการต่อ เซนเซอร์วัดก๊าซมีเทน MQ-4, ESP326

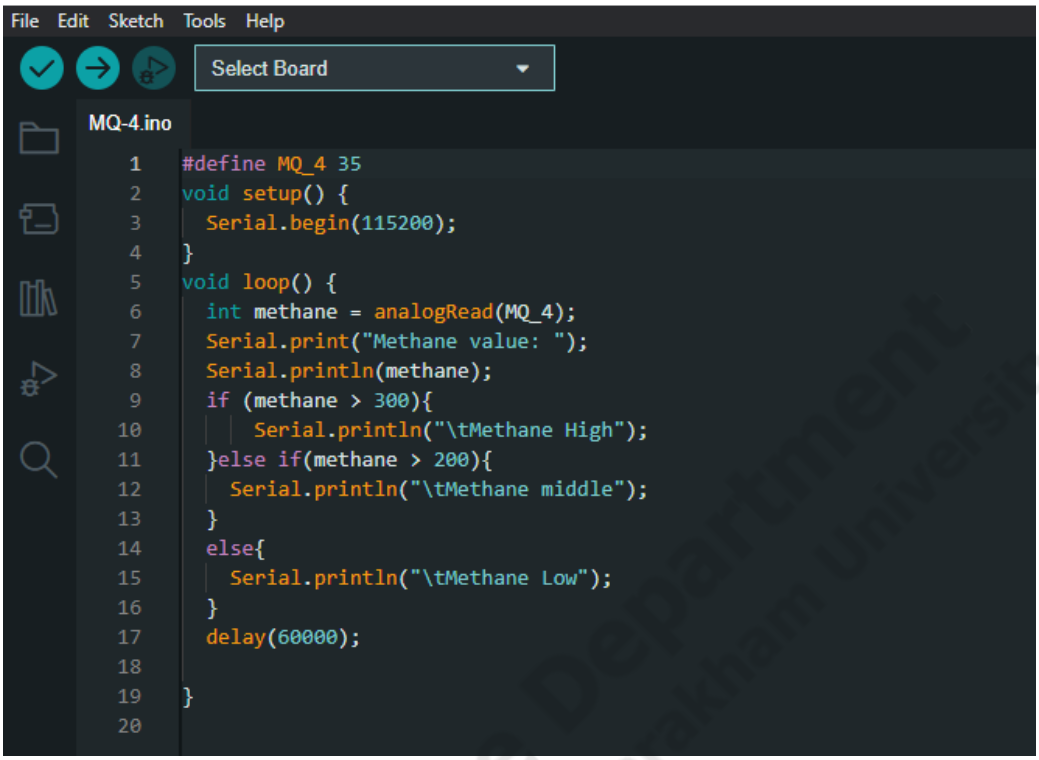


ภาพประกอบที่ 3.11 ขาต่อ MQ-4



ภาพประกอบที่ 3.12 การต่อ MQ-4

- คำสั่งเซนเซอร์วัดวัดก๊าซมีเทน MQ-4



ภาพประกอบที่ 3.13 คำสั่งเซนเซอร์วัดวัดก๊าซมีเทน

บรรทัดที่ 1 กำหนดตัวแปรที่ต่อกับขาที่ 35

บรรทัดที่ 5 เป็นคำสั่งในการ แสดงค่าออกมาแบบวนลูป

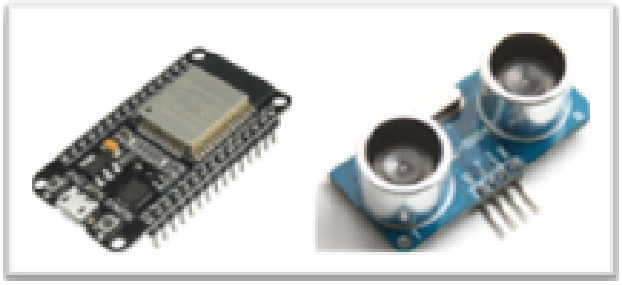
บรรทัดที่ 6 ประกาศตัวแปรมาเก็บค่าก๊าซมีเทน ที่ดึงข้อมูลมาจากตัวแปรที่ต่อกับขาที่ 35 ที่ได้มีการแปลงค่าจาก digital เป็น analog

บรรทัดที่ 7-8 เป็นการพิมพ์ค่า ที่ได้จากค่าก๊าซมีเทน ที่ได้จากตัวแปรที่ต่อกับขาที่ 35

บรรทัดที่ 9-16 เป็นคำสั่งเช็คเงื่อนไขของค่าก๊าซมีเทน จาก Sensor MQ-4

บรรทัดที่ 17 เป็นคำสั่งก่อนอ่านค่าและแสดงผลครั้งถัดไป

4. เซนเซอร์อัลตราโซนิก Ultrasonic Sensor Module (HC-SR04), ESP32



ภาพประกอบที่ 3.14 Ultrasonic Sensor Module (HC-SR04), ESP32

- วิธีการต่อเซนเซอร์ Ultrasonic Sensor Module (HC-SR04), ESP32

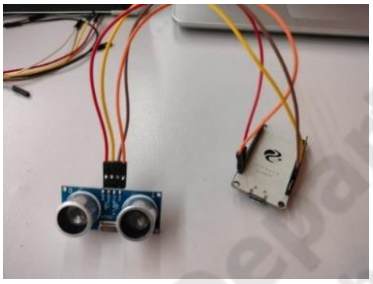
VCC -> 5V

GND -> GND

TRIG -> 18

ECHO -> 5

ภาพประกอบที่ 3.15 ขาต่อ Ultrasonic Sensor Module (HC-SR04)



ภาพประกอบที่ 3.16 การต่อ Ultrasonic Sensor Module (HC-SR04)

- คำสั่งเซนเซอร์อัลตราโซนิก Ultrasonic Sensor Module (HC-SR04)

```
1  #define TRIG_PIN 18 // ESP32 pin GIOP23 connected to Ultrasonic Sensor's TRIG pin
2  #define ECHO_PIN 5 // ESP32 pin GIOP22 connected to Ultrasonic Sensor's ECHO pin
3
4  float duration_us, cm;
5  String Capacity = "";
6  void setup() {
7    Serial.begin (115200);
8    pinMode(TRIG_PIN, OUTPUT);
9    pinMode(ECHO_PIN, INPUT);
10 }
11
12 void loop() {
13   digitalWrite(TRIG_PIN, HIGH);
14   delayMicroseconds(10);
15   digitalWrite(TRIG_PIN, LOW);
16   duration_us = pulseIn(ECHO_PIN, HIGH);
17   cm = 0.017 * duration_us;
18   Serial.print("distance: ");
19   Serial.print(cm);
20   Serial.println(" cm");
21   delay(1000);
22 }
```

ภาพประกอบที่ 3.17 คำสั่งเซนเซอร์อัลตราโซนิก

บรรทัดที่ 1 กำหนดตัวแปรที่ต่อกับขาที่ 18

บรรทัดที่ 2 กำหนดตัวแปรที่ต่อกับขาที่ 5

บรรทัดที่ 4 ประกาศตัวแปร 2 ตัวเป็น type float

บรรทัดที่ 5 ประกาศตัวแปรประเภท String

บรรทัดที่ 8 เป็นการ set ค่า pin ของขา TRIG_PIN เป็น Mode Output

บรรทัดที่ 9 เป็นการ set ค่า pin ของขา ECHO_PIN เป็น Mode Input

บรรทัดที่ 12 เป็นคำสั่งในการ แสดงค่าออกมาแบบวนลูป

บรรทัดที่ 13-15 เป็นการตั้งค่าพิน ให้กับขา trig

บรรทัดที่ 16 อ่าน Echo Pin และ pulseln() จะคืนค่าระยะเวลา เป็นไมโครวินาที

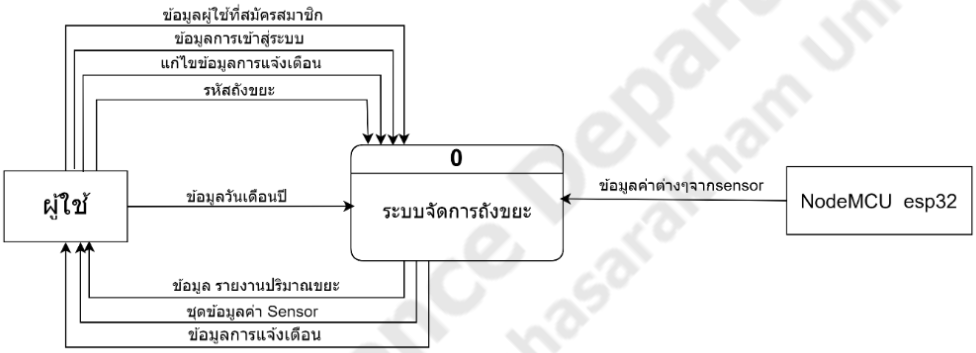
บรรทัดที่ 17 คำนวณค่าระยะทางเป็น cm

บรรทัดที่ 18-20 เป็นการพิมพ์ค่าระยะทางที่คำนวณมาได้

บรรทัดที่ 21 เป็นคำสั่งก่อนอ่านค่าและแสดงผลครั้งถัดไป

3.2 การออกแบบระบบ

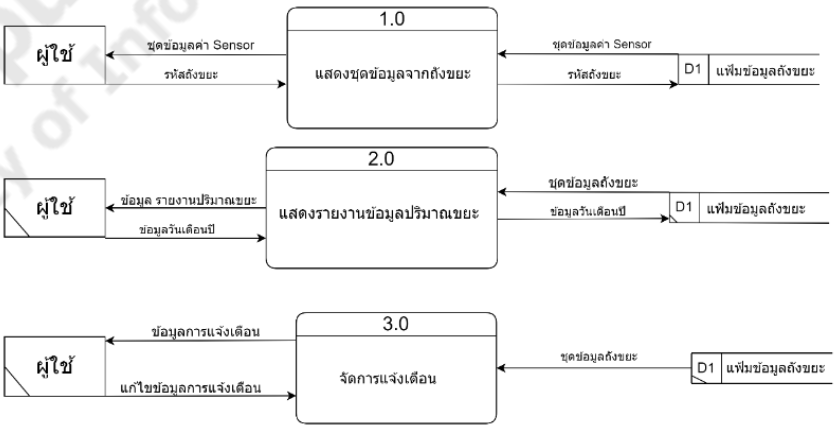
3.2.1 แผนภาพบริบท (Context Diagram)



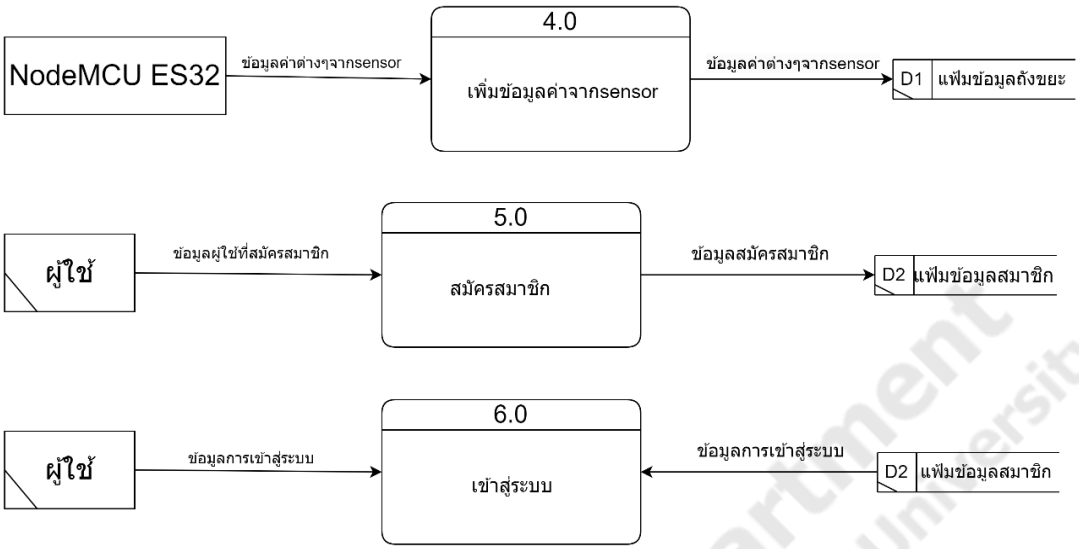
ภาพประกอบที่ 3.18 แผนภาพบริบท (Context Diagram)

3.2.2 แผนภาพการไหลของข้อมูล (Data Flow Diagram)

3.2.2.1 Data Flow Diagram Level 1



ภาพประกอบที่ 3.19 Data Flow Diagram Level 1_1



ภาพประกอบที่ 3.20 Data Flow Diagram Level 1_2

3.3 พจนานุกรมข้อมูล (Data Dictionary)

3.3.1 External Entity Description

ตารางที่ 3.1 External Entity Description

Name	Descriptions	Input Data Flow	Output Data Flow
ผู้ใช้	ผู้ใช้สามารถใช้งานระบบจัดการถังขยะ การดูข้อมูลค่าที่ได้จากถังขยะ IoTได้	- ข้อมูล รายงานปริมาณขยะ - ข้อมูลการแจ้งเตือน - ชุดข้อมูลค่า Sensor	- แก้ไขข้อมูลการแจ้งเตือน - รหัสถังขยะ - username - password
NodeMCU esp32	อุปกรณ์เซนเซอร์ที่ใช้ในการส่งข้อมูลต่างๆ		- ข้อมูลค่าต่างๆจาก sensor

3.3.2 Data Flow (Data Flow Description and Data Structure of Data Flow)

เป็นขั้นตอนการทำงานของแอปพลิเคชันซึ่งทำให้เราทราบถึงการรับ-ส่งข้อมูลแสดงถึงการไหลของข้อมูลทั้งข้อมูลเข้า (Input) และข้อมูลส่งออก (Output) ระหว่างข้อมูลต้นทางถึงข้อมูลปลายทาง โดย อธิบายข้อมูลและขั้นตอนในการดำเนินงานดังต่อไปนี้

ตารางที่ 3.2 Data Flow

Name	Descriptions	Source	Description	Data Structure
รหัสถังขยะ	รหัสถังขยะ	-ผู้ใช้	-process 1.0	IDถังขยะ
		-process 1.0	-D1แฟ้มข้อมูลถังขยะ	
ชุดข้อมูลค่า Sensor	ชุดข้อมูลค่า Sensor ต่างๆ ในฐานข้อมูล	-D1แฟ้มข้อมูลถังขยะ	-process 1.0	IDถังขยะ+ชื่อถังขยะ+ระดับความจุของถังขยะ+ละติจูด+ลองจิจูด+ที่ตั้งของถังขยะ+คุณภาพอากาศ+ค่าคุณภาพอากาศ(PPM)+ค่าก๊าซมีเทน + ค่าความชื้น + วัน/เดือน/ปี/เวลา
		-process 1.0	-ผู้ใช้	
ข้อมูลรายงานปริมาณขยะ	ข้อมูล รายงานปริมาณขยะ	-process 2.0	-ผู้ใช้	ชื่อถังขยะ+ระดับความจุของถังขยะ + วัน/เดือน/ปี/เวลา
ข้อมูลวันเดือนปี	ข้อมูลวันเดือนปี	-ผู้ใช้	-process 2.0	วัน/เดือน/ปี/เวลา
		-process 2.0	-D1แฟ้มข้อมูลถังขยะ	
ชุดข้อมูลถังขยะ	ชุดข้อมูลถังขยะแต่ละวัน เดือน ปี	-D1แฟ้มข้อมูลถังขยะ	-process 2.0	ชื่อถังขยะ+ระดับความจุของถังขยะ+ละติจูด+ลองจิจูด+ที่ตั้งของถังขยะ+คุณภาพอากาศ+ค่าคุณภาพอากาศ(PPM)+ค่าก๊าซมีเทน + ค่าความชื้น + วัน/เดือน/ปี/เวลา
		-D1แฟ้มข้อมูลถังขยะ	-process 3.0	
ข้อมูลการแจ้งเตือน	ข้อมูลการแจ้งเตือน	-process 3.0	-ผู้ใช้	ชื่อถังขยะ+ ที่ตั้งของถัง ขยะ และระดับความจุของถังขยะ หรือคุณภาพอากาศ หรือ ค่าก๊าซมีเทน และ เซนเซอร์Error

ตารางที่ 3.2 Data Flow (ต่อ)

Name	Descriptions	Source	Description	Data Structure
แก้ไขข้อมูลการแจ้งเตือน	ตั้งค่าแก้ไขการแจ้งเตือน	-ผู้ใช้	-process 3.0	ระดับความจุของถังขยะ หรือ คุณภาพอากาศ หรือ ค่าก๊าซมีเทน
ข้อมูลค่าต่างๆจากsensor	ข้อมูลค่าต่างๆจากsensor	-NodeMCU Esp32	-process 4.0 เพิ่มข้อมูลค่าจากsensor	IDถังขยะ+ชื่อถังขยะ+ระดับความจุของถังขยะ+ละติจูด+ลองจิจูด+ที่ตั้งของถังขยะ+คุณภาพอากาศ+ค่าคุณภาพอากาศ (PPM)+ค่าก๊าซมีเทน + ค่าความชื้น
		-process 4.0 เพิ่มข้อมูลค่าจากsensor	-D1แฟ้มข้อมูลถังขยะ	
ข้อมูลสมัครสมาชิก	รายละเอียดของข้อมูลการสมัคร	- ผู้ใช้	- process 5.0 สมัครสมาชิก	[username + password]
ข้อมูลการเข้าสู่ระบบ	การตรวจสอบการเข้าสู่ระบบ	- ผู้ใช้	- process 6.0 เข้าสู่ระบบ	[username + password]
		- แฟ้มข้อมูลสมาชิก	- process 6.0 เข้าสู่ระบบ	
ข้อมูลผู้ใช้ที่สมัครสมาชิก	รายละเอียดข้อมูลของสมาชิก	- process 5.0 สมัครสมาชิก	- แฟ้มข้อมูลสมาชิก	[username + password]

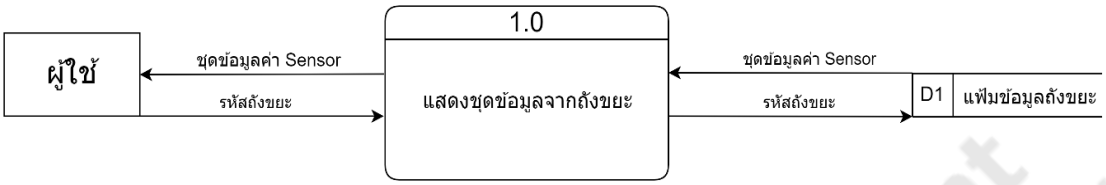
3.3.3 Data Store Description and Data Structure

ตารางที่ 3.3 Data Store

ID	Data Store	Description	Data Structure
D1	แฟ้มข้อมูลถังขยะ	เก็บข้อมูลต่างๆ ที่วัดค่าได้จากถังขยะ	IDถังขยะ+ชื่อถังขยะ+ระดับความจุ+ละติจูด+ลองจิจูด+ที่ตั้ง+คุณภาพอากาศ+ (PPM)+ค่าก๊าซมีเทน+ ความชื้น + วัน/เดือน/ปี/เวลา
D2	แฟ้มข้อมูลสมาชิก	เก็บข้อมูลของสมาชิก	Username + Password

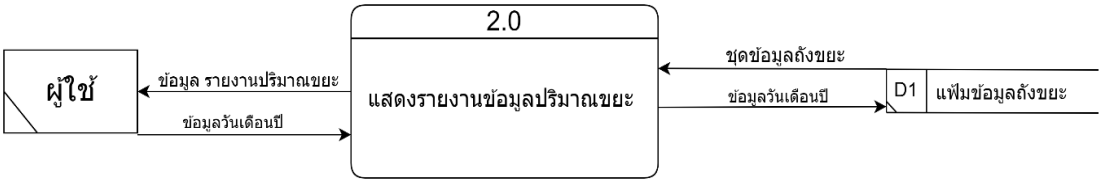
3.3.4 คำอธิบายการประมวลผล (Process Description)

3.3.4.1 Process 1.0 แสดงข้อมูลจากถังขยะ



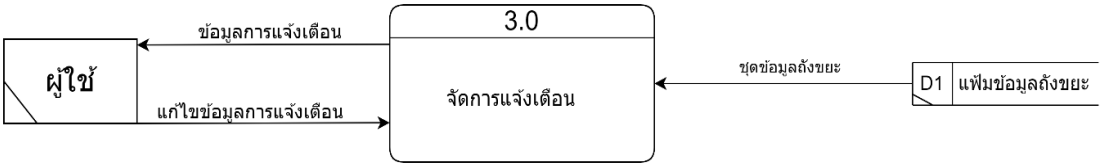
ID	1.0
NAME	แสดงชุดข้อมูลจากถังขยะ
DESCRIPTION	การแสดงชุดข้อมูลจากถังขยะ
INPUT DATA FLOW	-รหัสถังขยะ -ชุดข้อมูลค่า Sensor
OUT DATA FLOW	-ชุดข้อมูลค่า Sensor -รหัสถังขยะ
PROCESS DESCRIPTION	เริ่มต้น 1. รับ รหัสถังขยะ 2. ตรวจสอบข้อมูลทั้งหมดจากแฟ้มข้อมูลถังขยะ ด้วยรหัสถังขยะที่รับเข้ามา 3. แสดงข้อมูลจากแฟ้มข้อมูลถังขยะของข้อมูลรหัสถังขยะนั้นๆ จบการทำงาน

3.3.4.2 Process 2.0 แสดงรายงานข้อมูลปริมาณขยะ



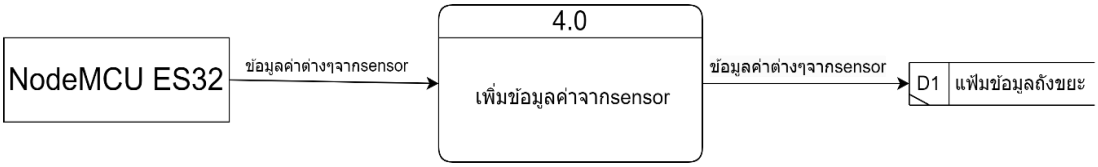
ID	2.0
NAME	แสดงรายงานข้อมูลปริมาณขยะ
DESCRIPTION	การแสดงรายงานข้อมูลปริมาณขยะ
INPUT DATA FLOW	-ข้อมูลวันเดือนปี -ชุดข้อมูลถังขยะ
OUT DATA FLOW	-ข้อมูล รายงานปริมาณขยะ -ข้อมูลวันเดือนปี
PROCESS DESCRIPTION	เริ่มต้น 1. รับ ช่วงเวลา [วันที่,เดือน,ปี] 2. ตรวจสอบข้อมูลความจุของแต่ละถังจากแฟ้มข้อมูลถังขยะตามช่วงเวลาและรหัสผู้ใช้ที่รับเข้ามา 2.1ถ้า (**มีข้อมูล**) แสดงรายงานข้อมูลอัตราความจุของแต่ละถังขยะ ตามวัน, เดือน, ปี 2.2 ถ้า (**ไม่มีข้อมูล**) แสดงข้อความ “ไม่มีข้อมูลอัตราความจุของแต่ละถังขยะ ตามวัน, เดือน, ปี” จบการทำงาน

3.3.4.3 Process 3.0 จัดการแจ้งเตือน



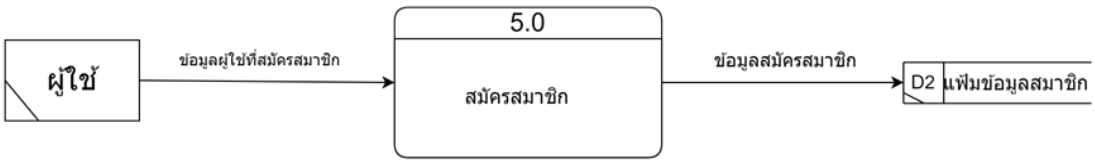
ID	3.0
NAME	จัดการแจ้งเตือน
DESCRIPTION	การจัดการแจ้งเตือน
INPUT DATA FLOW	-ชุดข้อมูลถังขยะ -แก้ไขข้อมูลการแจ้งเตือน
OUT DATA FLOW	-ข้อมูลการแจ้งเตือน
PROCESS DESCRIPTION	เริ่มต้น 1. ตั้งค่าการแจ้งว่าระดับความจุถึงระดับใดถึงแจ้งเตือน (Default 100% ถึงแจ้งเตือน) 2. ตรวจสอบค่าจากชุดข้อมูลถังขยะว่ามีถังขยะไหนมีระดับความจุถึงเกณฑ์ในการแจ้งเตือนแล้ว และ ยังมีค่าคุณภาพอากาศ และค่าก๊าซมีเทนในการแจ้งเตือนด้วย 2.1 ถ้า “ค่าความจุ หรือ ค่าคุณภาพอากาศ หรือ ค่า ก๊าซมีเทน **เกินเกณฑ์**ที่กำหนดไว้” ไปที่ 3 2.2 ถ้า “ค่าความจุ หรือ ค่าคุณภาพอากาศ หรือ ค่าก๊าซมีเทน **ไม่เกินเกณฑ์**ที่กำหนดไว้” วนกลับไปทำงานที่ 2 ใหม่ 3. ทำการแจ้งเตือนในแอปจัดการถังขยะ จบการทำงาน

3.3.4.4 Process 4.0 เพิ่มข้อมูลค่า Sensor



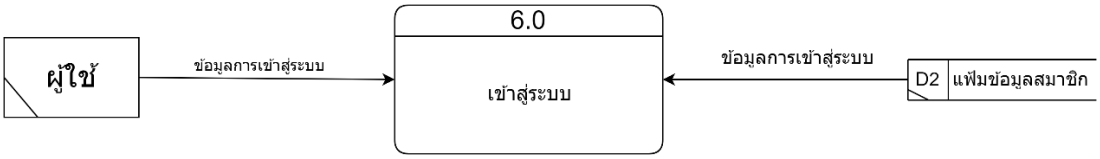
ID	4.0
NAME	เพิ่มข้อมูลค่าจากsensor
DESCRIPTION	การเพิ่มข้อมูลค่าจากsensor ต่างๆ ลงในฐานข้อมูล
INPUT DATA FLOW	-ข้อมูลค่าต่างๆจากsensor
OUT DATA FLOW	-ข้อมูลค่าต่างๆจากsensor
PROCESS DESCRIPTION	เริ่มต้น 1. รับ รหัสถังขยะ, ชื่อถังขยะ ,ค่าจากsensor ต่างๆ, สถานที่ตั้ง, วันเวลา 2. ตรวจสอบว่าข้อมูลครบถ้วนถูกต้องหรือไม่ 2.1 ถ้า “**ข้อมูลครบถ้วนถูกต้อง**” ไปที่ 3 2.2 ถ้า “**ข้อมูลไม่ครบถ้วนถูกต้อง**” ไปที่ 1 3. เพิ่มข้อมูลลงในฐานข้อมูล จบการทำงาน

3.3.4.5 Process 5.0 สมัครสมาชิก



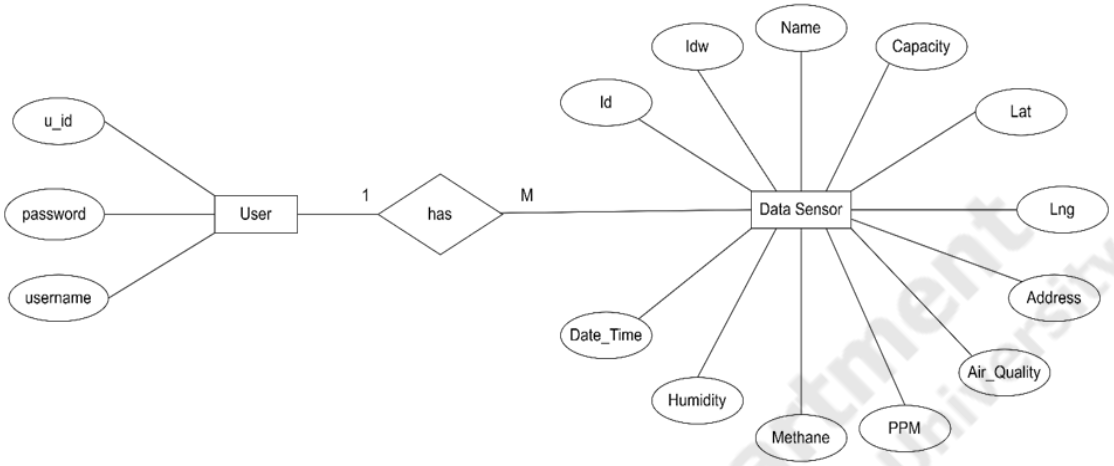
ID	5.0
NAME	สมัครสมาชิก
DESCRIPTION	ผู้ใช้สมัครสมาชิกเพื่อเป็นสมาชิก
INPUT DATA FLOW	- ข้อมูลผู้ใช้ที่สมัครสมาชิก
OUT DATA FLOW	- ข้อมูลสมัครสมาชิก
PROCESS DESCRIPTION	เริ่มต้น 1. รับข้อมูลที่ผู้ใช้สมัครสมาชิก 2. ตรวจสอบข้อมูลสมัครสมาชิกว่าครบถ้วนและถูกต้องหรือไม่ 2.1 ถ้าข้อมูลครบถ้วน ไปที่ข้อ 3. 2.2 ถ้าข้อมูลสมัครสมาชิกครบถ้วนแต่ไม่ถูกต้องให้แสดงข้อความ“**กรุณาป้อนข้อมูลให้ถูกต้อง**” กลับไปข้อ 1. 3. ตรวจสอบในแฟ้มข้อมูลสมาชิกว่ามีข้อมูลusernameนี้แล้วหรือไม่ 3.1 ถ้ามีข้อมูลusernameแล้วให้แสดงข้อความ “**พบข้อมูลอีเมลซ้ำในระบบ**” กลับไปที่ข้อ 1. 3.2 ถ้าไม่มีข้อมูลusernameซ้ำในระบบให้เพิ่มข้อมูลสมัครสมาชิกลงในแฟ้มข้อมูลสมาชิกและแสดงข้อความ “**สมัครสมาชิกสำเร็จ**” จบการทำงาน

3.3.4.6 Process 6.0 เข้าสู่ระบบ



ID	6.0
NAME	เข้าสู่ระบบ
DESCRIPTION	เข้าสู่ระบบเพื่อใช้งานแอปพลิเคชัน
INPUT DATA FLOW	- ข้อมูลการเข้าสู่ระบบ
OUT DATA FLOW	- ข้อมูลการเข้าสู่ระบบ
PROCESS DESCRIPTION	เริ่มต้น 1. รับข้อมูลการเข้าสู่ระบบ (username และ รหัสผ่าน) 2. ตรวจสอบข้อมูลการเข้าสู่ระบบว่ามีข้อมูลที่กรอกครบถ้วนหรือไม่ 2.1 ถ้าป้อนข้อมูลการเข้าสู่ระบบครบถ้วน ไปที่ข้อ 3. 2.2 ถ้าป้อนข้อมูลการเข้าสู่ระบบไม่ครบ ให้แสดงข้อความ“**กรุณาป้อนข้อมูลให้ครบ**” กลับไปข้อ 1. 3. ตรวจสอบข้อมูลการเข้าสู่ระบบในแฟ้มข้อมูลสมาชิก 3.1 ถ้ามีข้อมูลการเข้าสู่ระบบในแฟ้มข้อมูลสมาชิก ให้แสดงข้อความ“**เข้าสู่ระบบสำเร็จ**” 3.2 ถ้าไม่มีข้อมูลการเข้าสู่ระบบในแฟ้มข้อมูลสมาชิก ให้แสดงข้อความ“**กรุณาตรวจสอบ username หรือ รหัสผ่านให้ถูกต้อง**” กลับไปที่ข้อ 1. จบการทำงาน

3.4 แผนภาพความสัมพันธ์ (Entity Relationship Diagram)



ภาพประกอบที่ 3.21 Entity Relationship Diagram

3.5 การออกแบบฐานข้อมูล (Database Design)

3.5.1 ข้อมูลค่าจากถังขยะ

ตารางที่ 3.4 ข้อมูลค่าจากถังขยะ (Data Sensor)

Attribute	Type	Description	Example
Idw	Int (255)	รหัสของถังขยะ	1
Name	varchar (255)	ชื่อถังขยะ	Bin01
Capacity	varchar (255)	ความจุของถังขยะ	50
Lat	varchar (255)	ละติจูด	16.246501
Lng	varchar (255)	ลองจิจูด	103.252080
Address	varchar (255)	ชื่อที่ตั้งของถังขยะ	MSU
Air Quality	varchar (255)	คุณภาพอากาศ	Good
PPM	Int (255)	ค่าคุณภาพอากาศ (PPM)	125
Methane	varchar (255)	ค่าก๊าซมีเทน	No Methane
Humidity	varchar (255)	ความชื้น	65
Date_Time	varchar (255)	เดือน วัน ปี เวลา	01/01/2023 09:00:00

3.5.2 ข้อมูลสมาชิก

ตารางที่ 3.5 ข้อมูลสมาชิก (User)

Attribute	Type	Description	Example
u_id	Int	รหัสของผู้ใช้	1
username	varchar (255)	ชื่อผู้ใช้	Pee@gmail.com
password	varchar (255)	รหัสผ่าน	1234@

3.6 การพัฒนาระบบ

3.6.1 โค้ด PHP ที่ใช้เชื่อมต่อกับฐานข้อมูล

3.6.1.1 โค้ดการเชื่อมต่อกับฐานข้อมูล

```php
<?php

$servername = "localhost";
$username = "id20351938_adminesp32";
$password = "[illegible]";
$dbname = "id20351938_wastebinesp32";
date_default_timezone_set('Asia/Bangkok');
$date = date('d/m/Y H:i:s',time());

$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn){
        echo "Connection Success db";
}
else{
        die("Connection Failed" . $conn->connect_error);

}
```

ภาพประกอบที่ 3.22 โค้ดเชื่อมต่อกับฐานข้อมูล

บรรทัดที่ 3 สร้างตัวแปรเก็บ Host ของฐานข้อมูล

บรรทัดที่ 4-5 สร้างตัวแปรเก็บ Username และ Password ในการloginเข้าใช้ฐานข้อมูล

บรรทัดที่ 6 สร้างตัวแปรเก็บ ชื่อของ Database ของฐานข้อมูล

บรรทัดที่ 7 set Time Zone ให้เป็นเวลาประเทศไทย

บรรทัดที่ 8 สร้างตัวแปรเก็บ วัน/เดือน/ปี เวลา

บรรทัดที่ 10 คำสั่งในการเชื่อมต่อกับฐานข้อมูล

บรรทัดที่ 11-17 เงื่อนไขถ้าเชื่อมต่อสำเร็จ แสดงข้อความในบรรทัดที่ 12 แต่ถ้าไม่สำเร็จ สำเร็จแสดงข้อความในบรรทัดที่ 15 และแสดงConnect Error เพราะอะไร

3.6.1.2 โค้ดการเพิ่มข้อมูลลงฐานข้อมูล

```php
if (isset($_POST['Idw'])){
    $Idw = $_POST['Idw'];
    $Name = $_POST['Name'];
    $Capacity = $_POST['Capacity'];
    $lat = $_POST['lat'];
    $lng = $_POST['lng'];
    $Address = $_POST['Address'];
    $Air_Quality = $_POST['Air_Quality'];
    $PPM = $_POST['PPM'];
    $Methane = $_POST['Methane'];
    $Humidity = $_POST['Humidity'];

    $sql = "INSERT INTO `wastebin`(`Id`, `Idw`, `Name`, `Capacity`, `Lat`, `Lng`, `Address`, `Air_Quality`, `PPM`,
        `Methane`,`Humidity`, `Date_Time`) VALUES (NULL,'".$Idw."', '".$Name."','".$Capacity."', '".$lat."', '".$lng."'
        ,'".$Address."', '".$Air_Quality."','".$PPM."','".$Methane."','".$Humidity."','".$date."')";
    if ($conn->query($sql) === TRUE) {
            echo "New record created successfully";
    } else {
            echo "Error: " . $sql . "<br>" . $conn->error;
    }
}

?>
```

ภาพประกอบที่ 3.23 โค้ดการเพิ่มข้อมูลลงฐานข้อมูล

บรรทัดที่ 18 เงื่อนไขเมื่อ POST ส่งค่า Idw มาโปรแกรมก็จะทำงาน

บรรทัดที่ 19-27 ประการตัวแปรในการเก็บค่าข้อมูลที่ส่งมาผ่านAPI แบบ POST

บรรทัดที่ 29 คำสั่ง SQL ในการเพิ่มข้อมูล โดยนำข้อมูลที่รับมาไปเพิ่มลงในฐานข้อมูล

บรรทัดที่ 30-35 คำสั่งเงื่อนไข ถ้าquery ข้อมูลสำเร็จ แสดงข้อความใน บรรทัดที่ 31 แต่ถ้าไม่สำเร็จ แสดงข้อความใน บรรทัดที่ 33

3.6.1.3 โค้ดการแสดงข้อมูลวันเวลาล่าสุด

```php
<?php
require('connection.php');


$sql = " select * from wastebin bin inner join ( select `Name`, max(`Date_Time`) as MaxDate from wastebin group by `Name`
    ) tm on bin.`Name` = tm.`Name` and bin.`Date_Time` = tm.MaxDate";

$results = $connection->query($sql);

$emparray = array();
    while($row = mysqli_fetch_assoc($results)){
        $emparray[] = $row;
    }

echo json_encode($emparray);


?>
```

ภาพประกอบที่ 3.24 โค้ดการแสดงข้อมูลวันเวลาล่าสุด

บรรทัดที่ 2 เรียกใช้file สำหรับการเชื่อมต่อกับฐานข้อมูล

บรรทัดที่ 5 คำสั่งSQLในการค้นหาข้อมูล โดยใช้ Inner join ในการค้นหาว่าข้อมูลแต่ละชื่อ ข้อมูลใหนเป็นข้อมูลวันเวลาล่าสุด

บรรทัดที่ 7 สร้างตัวแปรในการ เก็บข้อมูลที่ query จากฐานข้อมูล

บรรทัดที่ 9 ประกาศตัวแปร Array

บรรทัดที่ 10-12 คำสั่งวนลูปการนำค่าที่ได้จากฐานข้อมูลมาเก็บไว้ใน Array

บรรทัดที่ 14 คำสั่ง นำข้อมูลที่เก็บใน Array มา encode เป็น Json

3.6.2 โค้ด Arduino เชื่อมต่อกับ Web Service (API)

```
#include <WIFI.h>
#define HOST "proesp32.000webhostapp.com"          // Enter HOST URL without "http:// " and "/" at the end of URL
#define WIFI_SSID "NTBSK_01_29_2.4G"               // WIFI SSID here
#define WIFI_PASSWORD "55555555"        // WIFI password here
#include <HTTPClient.h>

String  postData;

String Idw = "4";
String Name = "Bin04";
String Capacity = "50%";
String lat = "16.246501";
String lng = "103.252000";
String Address = "MSU";
String Air_Quality = "GOOD";
String ppm = "135" ;
String Methane = "No Methane";

void setup() {
    Serial.begin(115200);
    Serial.println("Communication Started \n\n");
    delay(1000);
    pinMode(LED_BUILTIN, OUTPUT);     // Initialize built in led on the board
    WIFI.mode(WIFI_STA);
    WIFI.begin(WIFI_SSID, WIFI_PASSWORD);                                    //try to connect with wifi
    Serial.print("Connecting to ");
    Serial.print(WIFI_SSID);
    while (WIFI.status() != WL_CONNECTED)
    { Serial.print(".");
        delay(500); }

    Serial.println();
    Serial.print("Connected to ");
    Serial.println(WIFI_SSID);
    Serial.print("IP Address is : ");
    Serial.println(WIFI.localIP());    //print local IP address
}
```

ภาพประกอบที่ 3.25 โค้ดส่วนเชื่อมต่อกับWiFi

บรรทัดที่ 1-5 เรียกใช้งาน library

บรรทัดที่ 7-17 ประกาศตัวอย่างข้อมูล

บรรทัดที่ 24 เรียกใช้ Wi-Fi

บรรทัดที่ 25 เชื่อมต่อ Wi-Fi โดย WIFI_SSID คือ ชื่อ และ WIFI_PASSWORD คือ รหัสผ่าน

บรรทัดที่ 28-30 Loop Wi-Fi Status

บรรทัดที่ 33-34 โชว์ว่าเชื่อมต่อกับ Wi-Fi อะไร

บรรทัดที่ 35-36 โชว์ Ip-Address

```
void loop() {

    HTTPClient http;    // http object of clas HTTPClient
    WiFiClient wclient; // wclient object of clas HTTPClient

    postData = "&Idw=" + Idw+ "&Name=" + Name + "&Capacity=" + Capacity
             + "&lat=" + lat + "&lng=" + lng + "&Address=" + Address+ "&Air_Quality="
             + Air_Quality + "&PPM=" + ppm + "&Methane="+ Methane + "&Humidity=" + Humidity ;

    http.begin(wclient, "http://proesp32.000webhostapp.com/config.php");              // Connect to host where MySQL databse is hosted
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");            //Specify content-type header

    int httpCode = http.POST(postData);   // Send POST request to php file and store server response code in variable named httpCode
    if (httpCode == 200) {
      Serial.println("Values uploaded successfully.");
      Serial.println(httpCode);
      String webpage = http.getString();    // Get html webpage output and store it in a string
      Serial.println(webpage + "\n");
    }else {
      Serial.println(httpCode);
      Serial.println("Failed to upload values. \n");
      http.end();
      return;
    }
    delay(60000);
}
```

ภาพประกอบที่ 3.26 โค้ดในส่วนการส่งข้อมูลไปฐานข้อมูล

บรรทัดที่ 39 เริ่ม Loop การทำงานส่งค่าข้อมูล

บรรทัดที่ 41-42 ประกาศ Object ของ class HTTP Client

บรรทัดที่ 44 เก็บข้อมูลไว้ใน Post Data

บรรทัดที่ 48 เชื่อมต่อกับ MySQL Database

บรรทัดที่ 49 ระบุ Content-Type

บรรทัดที่ 51 ส่ง POST ไปยังไฟล์ PHP และ Server ตอบกลับเป็น HTTP Code

บรรทัดที่ 52-62 เป็นเงื่อนไขว่าเชื่อมต่อสำเร็จหรือไม่

3.6.3 โค้ดในส่วนของแอปพลิเคชัน (Mobile)

3.6.3.1 การแสดงข้อมูลถังขยะ

```
64          ? ListView.separated(
65              itemCount: snapshot.data!.length,
66              itemBuilder: (context, index) {
67                return ListTile(
68                title: Text(snapshot.data![index].Name),
69                subtitle: Text(snapshot.data![index].Capacity+"%"),
70                  onTap: () {
71                    Navigator.push(
72                        context,
73                        MaterialPageRoute(
74                        builder: (context) => DetailScreen(article: snapshot.data![index]),
75                    ), // MaterialPageRoute
76                    );
77                  }
78              ); // ListTile
```

ภาพประกอบที่ 3.27 โค้ดแสดงlistข้อมูลบนแอป

บรรทัดที่ 64 เริ่มต้นการสร้าง ListView

บรรทัดที่ 65 กำหนดจำนวนรายการที่ต้องแสดงใน ListView โดยใช้ค่าจาก snapshot โดยจำนวนรายการจะเท่ากับความยาวของ List ที่อยู่ใน snapshot

บรรทัดที่ 66 กำหนด function สำหรับสร้างแต่ละรายการของ ListView โดยใน function นี้จะมี parameter 2 ตัวคือ context และ index

บรรทัดที่ 67 สร้าง ListTile widget ที่จะแสดงข้อมูลในแต่ละรายการของ ListView

บรรทัดที่ 68 กำหนดข้อความในส่วนของ Title ของ ListTile โดยใช้ข้อมูลชื่อจาก snapshot ซึ่งเป็น List และอ้างอิงถึง index ที่กำลังสร้าง

บรรทัดที่ 69 กำหนดข้อความในส่วนของ Subtitle ของ ListTile โดยใช้ข้อมูลความจุจาก snapshot ซึ่งเป็น List และอ้างอิงถึง index ที่กำลังสร้าง และเพิ่มเครื่องหมายเปอร์เซ็นต์ (%) ด้วย

บรรทัดที่ 70 กำหนด function ที่จะถูกเรียกเมื่อผู้ใช้แตะที่ ListTile

บรรทัดที่ 71-74 เป็นการเปิดหน้าจอใหม่ โดยใช้ Navigator ไปที่ Detail Screen

3.6.3.2 การดึงข้อมูลจากฐานข้อมูล

```
107  Future<List<Article>> fetchArticle() async {
108    // ทำการดึงข้อมูลจาก server ตาม url ที่กำหนด
109    final response = await http
110        .get(Uri.parse('https://proesp32.000webhostapp.com/getLatestData.php'));
111    // เมื่อมีข้อมูลกลับมา
112    if (response.statusCode == 200) {
113      // ส่งข้อมูลที่เป็น JSON String data ไปทำการแปลง เป็นข้อมูล List<Article
114      // โดยใช้คำสั่ง compute ทำงานเบื้องหลัง เรียกใช้ฟังก์ชั่นชื่อ parseArticles
115      // ส่งข้อมูล JSON String data ผ่านตัวแปร response.body
116      return compute(parseArticles, response.body);
117    } else { // กรณี error
118      throw Exception('Failed to load article');
119    }
120  }
121
122  List<Article> parseArticles(String responseBody) {
123    final parsed = jsonDecode(responseBody).cast<Map<String, dynamic>>();
124    return parsed.map<Article>((json) => Article.fromJson(json)).toList();
125  }
```

ภาพประกอบที่ 3.28 โค้ดในส่วนดึงข้อมูล

บรรทัดที่ 107 เริ่มต้นการเขียนฟังก์ชันชื่อ fetchArticle โดยกำหนดว่าจะ return ข้อมูลในรูปแบบ Future<List<Article>>

บรรทัดที่ 109-110 เป็นการส่ง HTTP GET request ไปยัง URL ที่กำหนด โดยใช้ package http และระบุว่าต้องรอการตอบกลับด้วย await จนกว่าจะได้รับ response

บรรทัดที่ 112 ถ้า response status code เป็น 200 (OK) หมายความว่าสามารถดึงข้อมูลจาก server ได้สำเร็จ

บรรทัดที่ 116 ใช้ฟังก์ชัน compute เพื่อเรียกใช้ฟังก์ชัน parseArticles โดยส่ง JSON string data ผ่านตัวแปร response.body และ return ข้อมูลที่ได้เป็น Future<List<Article>>

บรรทัดที่ 118 ถ้า response status code ไม่ใช่ 200 หมายความว่าไม่สามารถดึงข้อมูลจาก server ได้ จึง throw Exception แจ้งเตือนว่าการดึงข้อมูลล้มเหลว

บรรทัดที่ 122 เริ่มต้นการเขียนฟังก์ชันชื่อ parseArticles โดยรับข้อมูลเป็น JSON string ในรูปแบบของ responseBody

บรรทัดที่ 123-124 แปลง JSON string เป็น Map<String, dynamic> โดยใช้ฟังก์ชัน jsonDecode()

```dart
153   factory Article.fromJson(Map<String, dynamic> json) {
154     return Article(
155       Id: json['Id'] as String,
156       Idw: json['Idw'] as String,
157       Name: json['Name'] as String,
158       Capacity: json['Capacity'] as String,
159       Lat: json['Lat'] as String,
160       Lng: json['Lng'] as String,
161       Address: json['Address'] as String,
162       Air_Quality: json['Air_Quality'] as String,
163       PPM: json['PPM'] as String,
164       Methane: json['Methane'] as String,
165       Date_Time: json['Date_Time'] as String,
166     );
167   }
168 }
```

ภาพประกอบที่ 3.29 โค้ดสร้างตัวเก็บข้อมูลที่ดึงมาใช้

บรรทัดที่ 154-168 สร้าง object ของ class Article จากข้อมูลที่ได้รับมาในรูปแบบ JSON โดยรับข้อมูล JSON

3.6.3.3 หน้า Google Map

```dart
29      body: GoogleMap(
30        mapType: MapType.normal,
31        initialCameraPosition: CameraPosition(
32          target: LatLng(15.6845261, 104.0978119),
33          zoom: 16,
34        ),  // CameraPosition
35        onMapCreated: (GoogleMapController controller) {
36          _controller.complete(controller);
37        },
38        markers: {
39          Marker(
40              markerId: MarkerId("1"),
41              position: LatLng(15.6845261, 104.0978119),
42              infoWindow: InfoWindow(title: "บ้านผม", snippet: "บ้านผม")),  // Marker
43        },
44      ),  // GoogleMap
```

ภาพประกอบที่ 3.30 โค้ดแสดง Google Map

บรรทัดที่ 29 สร้าง GoogleMap widget

บรรทัดที่ 30 กำหนดประเภทของแผนที่เป็น MapType.normal

บรรทัดที่ 31-33 กำหนดตำแหน่งและการขยายแผนที่เริ่มต้นด้วย initialCameraPosition โดยกำหนด target ให้เป็นพิกัด LatLng ของตำแหน่งที่ต้องการแสดงแผนที่ และกำหนด zoom level เริ่มต้นเป็น 16

บรรทัดที่ 35-36 กำหนด onMapCreated callback ซึ่งจะถูกเรียกเมื่อ GoogleMap widget ถูกสร้างเรียบร้อยแล้ว เพื่อให้คุณสามารถควบคุมแผนที่ได้

บรรทัดที่ 38-42 กำหนด Marker บนแผนที่โดยใช้ Marker widget โดยกำหนด markerId และ position ของ Marker และกำหนดข้อความใน InfoWindow ด้วย title และ snippet ตามลำดับ

3.6.3.4 หน้า Charts

```
157  Future<Map<int, List<Article>>> fetchArticle(DateTime selectedDate) async {
158    final formattedDate = DateFormat('MM/dd/yyyy').format(selectedDate);
159    final response = await http.get(Uri.parse('https://proesp32.000webhostapp.com/getDataByDate.php?date=$formattedDate'));
160    if (response.statusCode == 200) {
161      final articles = parseArticles(response.body);
162      final idwMap = <int, List<Article>>{};
163      articles.forEach((article) {
164        final idw = article.Idw;
165        if (!idwMap.containsKey(idw)) {
166          idwMap[idw] = [];
167        }
168        if (article.Date_Time.month == selectedDate.month && article.Date_Time.day == selectedDate.day
169            && article.Date_Time.year == selectedDate.year) {
170          idwMap[idw]!.add(article);
171        }
172      });
173      return idwMap;
174    } else {
175      throw Exception('Failed to load article');
176    }
177  }
```

ภาพประกอบที่ 3.31 โค้ดในส่วนการแสดง Charts

บรรทัดที่ 157 ประกาศฟังก์ชั่น fetchArticle ซึ่งรับ DateTime และจะคืนค่าเป็น Map<int, List<Article>>

บรรทัดที่ 158 แปลงวันที่ที่รับเข้ามาเป็น string ในรูปแบบ "MM/dd/yyyy"

บรรทัดที่ 159 ส่ง request ไปที่ server โดยใช้ http.get และนำ response ที่ได้เก็บไว้ในตัวแปร response

บรรทัดที่ 160-166 ตรวจสอบว่า response มีสถานะเป็น 200 หรือไม่ หากใช่ จะทำการแปลง JSON ของ response ให้เป็น List<Article> โดยใช้ฟังก์ชั่น parseArticles และจะเรียกใช้งานฟังก์ชั่น parseArticles นี้เพื่อแปลง JSON ใน response.body เป็น List<Article>

บรรทัดที่ 168-170 การสร้าง Map<int, List<Article>> โดยใช้ idwMap เพื่อเก็บบทความตาม IDW และวันที่ที่ถูกเลือก โดยเริ่มต้นให้ idwMap เป็น map ว่าง และวนลูปผ่าน articles เพื่อนำบทความไปเก็บใน idwMap โดยเช็คเงื่อนไขว่าบทความนั้นๆ เป็นของวันที่ที่ถูกเลือกหรือไม่ ถ้าใช่ก็จะนำบทความนั้นไปเก็บใน List<Article> ของ IDW นั้นๆ

บรรทัดที่ 174 คืนค่า idwMap ที่ได้รับการสร้างขึ้น

```
179  List<Article> parseArticles(String responseBody) {
180    final parsed = jsonDecode(responseBody).cast<Map<String, dynamic>>();
181    return parsed.map<Article>((json) {
182      if (!json.containsKey('Capacity') || !json.containsKey('Date_Time')) {
183        throw FormatException('Invalid JSON data');
184      }
185
186      try {
187        return Article.fromJson(json);
188      } catch (e) {
189        throw FormatException('Invalid JSON data: $e');
190      }
191    }).toList();
192  }
```

ภาพประกอบที่ 3.32 โค้ดในส่วนเปลี่ยนข้อมูลเป็นJSON

บรรทัดที่ 179 รับพารามิเตอร์ responseBody ที่เป็น JSON String ที่ได้รับมาจาก API

บรรทัดที่ 180 รับพารามิเตอร์ responseBody ที่เป็น JSON String ที่ได้รับมาจาก API

บรรทัดที่ 181 ใช้ method map ของ List เพื่อสร้าง List ใหม่โดยแต่ละ element จะถูกแปลงเป็น Article object ด้วย method fromJson โดยผ่าน parameter แต่ละตัวเข้าไป

บรรทัดที่ 182-183 ตรวจสอบว่า json object นั้นมี key 'Capacity' และ 'Date_Time' หรือไม่ ถ้าไม่มีจะ throw FormatException

บรรทัดที่ 187 ใช้ method fromJson ของ Article class เพื่อแปลง json object เป็น Article object

บรรทัดที่ 188-189 กรณีเกิดการ error ในการแปลง json object เป็น Article object จะทำ throw Format Exception พร้อมกับ message ว่า Invalid JSON data: และผลลัพธ์ของ exception ที่เกิดขึ้น

บรรทัดที่ 190 return ออกมาเป็น List<Article> ด้วย toList()

```
86  Expanded(
87    child: FutureBuilder<Map<int, List<Article>>>(
88      future: articles,
89      builder: (context, snapshot) {
90        if (snapshot.hasData) {
91          final idwMap = snapshot.data!;
92          final data = selectedIdw != null ? idwMap[selectedIdw!] ?? [] : idwMap.values.expand((articles) => articles).toList();
93          data.sort((a, b) => a.Date_Time.compareTo(b.Date_Time));
94          final averageCapacity = data.isNotEmpty
95              ? data.map((article) => article.Capacity).reduce((a, b) => a + b) / data.length
96              : 0.0;
```

ภาพประกอบที่ 3.33 โค้ดในส่วนสร้างExpandedของกราฟ

บรรทัดที่ 86-87 สร้าง Widget ชนิด Expanded ซึ่งจะขยายพื้นที่ของ Widget หลัก ในส่วนของ child จะเป็น FutureBuilder ซึ่งจะรอให้ Future ชื่อ articles เสร็จสิ้นแล้วแสดงผลลัพธ์ที่ได้จาก Future นั้น ในส่วนของ builder จะเป็นฟังก์ชันที่จะถูก

เรียกเมื่อ Future สำเร็จและมีผลลัพธ์ (snapshot) โดยจะรับค่า snapshot แล้วแสดงผลลัพธ์

บรรทัดที่ 90-92 ตรวจสอบว่า snapshot มีข้อมูลหรือไม่ ถ้ามีก็จะดึงข้อมูลมาเก็บไว้ในตัวแปร idwMap โดยจะใช้ดึงค่าของ selectedIdw ออกมาเพื่อใช้ในการกรองข้อมูล โดยถ้า selectedIdw มีค่าก็จะกรองข้อมูลใน idwMap ตาม selectedIdw แต่ถ้าไม่มีก็จะเอาข้อมูลทั้งหมดใน idwMap มาใช้

บรรทัดที่ 93 เรียงลำดับข้อมูลใน data โดยใช้ฟังก์ชัน sort() โดยจะเรียงลำดับโดยใช้เวลา (Date_Time) ของข้อมูล โดยเทียบกันไปเลยว่าข้อมูลไหนมาก่อน โดยจะใช้ method compareTo() ของ DateTime ในการเปรียบเทียบ

บรรทัดที่ 94 -96 คำนวณค่าเฉลี่ย Capacity ของข้อมูลใน data

```
40          Expanded(
41            child: FutureBuilder<List<Article>>(
42              future: articles,
43              builder: (context, snapshot) {
44                if (snapshot.hasData) {
45                  final data = snapshot.data!;
46                  final averageCapacity = data.isNotEmpty
47                      ? data.map((article) => article.Capacity).reduce((a, b) => a + b) / data.length
48                      : 0.0;
49                  final articlesByGroup = groupArticlesByGroup(data);
50
51                  final seriesList = articlesByGroup.entries
52                      .map((entry) => StackedLineSeries<Article, DateTime>(
53                    name: 'IDW ${entry.key}',
54                    dataSource: entry.value,
55                    xValueMapper: (Article article, _) =>
56                    article.Date_Time, // map Date_Time to x-axis
57                    yValueMapper: (Article article, _) =>
58                    article.Capacity, // map Capacity to y-axis
59                    color: idwColors[entry.key]!,
60                  ))  // StackedLineSeries
61                      .toList();
```

ภาพประกอบที่ 3.34 โค้ดในส่วนสร้างWidgetของกราฟ

บรรทัดที่ 41 สร้าง Widget ของ FutureBuilder ที่รอรับข้อมูล List<Article> จาก Future ชื่อ articles โดยใช้ builder

บรรทัดที่ 43 ระบุ function ที่จะเป็นตัวกำหนดการแสดงผลของ widget โดยมี context และ snapshot เป็น input parameter

บรรทัดที่ 44 หาก snapshot มีข้อมูล snapshot.hasData จะสร้าง chart โดยใช้ข้อมูลจาก snapshot.data

บรรทัดที่ 47 คำนวณค่าเฉลี่ยของ Capacity ใน List ของ Article ทั้งหมด โดยใช้ map และ reduce เป็นการบวกผลรวมแต่ละตัวแล้วหารด้วยจำนวนข้อมูลใน List ในกรณีที่ List ว่าง จะใช้ค่าเริ่มต้นเป็น 0.0

บรรทัดที่ 49 จัดกลุ่ม Article ตามกลุ่ม IDW และเก็บลงใน Map โดยใช้ groupArticlesByGroup

บรรทัดที่ 51-61 สร้าง List ของ StackedLineSeries จาก articlesByGroup โดยใช้ map และ toList

```
97  return Column(
98    children: [
99      SfCartesianChart(
100       title: ChartTitle(text: 'Capacity vs Time'),
101       legend: Legend(
102         isVisible: true,
103         title: LegendTitle(text: 'Legend'),
104       ),  // Legend
105       tooltipBehavior: TooltipBehavior(enable: true),
106       series: <ChartSeries>[
107         StackedLineSeries<Article, DateTime>(
108           name: 'Capacity',
109           dataSource: data,
110           color: idwColors[selectedIdw] ?? Colors.black,
111           xValueMapper: (Article article, _) => article.Date_Time,
112           yValueMapper: (Article article, _) => article.Capacity,
113           dataLabelSettings: DataLabelSettings(isVisible: true),
114         ),  // StackedLineSeries
115       ],  // <ChartSeries>[]
116       primaryXAxis: DateTimeAxis(
117         minimum: DateTime(selectedDate.year, selectedDate.month, selectedDate.day, 6), // set minimum time to 6 am
118       ),  // DateTimeAxis
119     ),  // SfCartesianChart
120     Text('Average capacity: ${averageCapacity.toStringAsFixed(2)}'),
```

ภาพประกอบที่ 3.35 โค้ดในส่วนการแสดงกราฟแบบเส้น

บรรทัดที่ 97 สร้าง Widget แบบ Column ซึ่งเป็น Widget ที่เรียงลำดับ Widget ลงมาเป็นแนวตั้ง

บรรทัดที่ 99 สร้าง Widget แบบ SfCartesianChart ซึ่งเป็น Widget ที่แสดงผลกราฟแท่งข้อมูล โดยใช้งานไลบรารี Syncfusion Flutter Charts

บรรทัดที่ 101-102 กำหนดให้แสดงตัวคำอธิบายกราฟ (Legend) โดยกำหนด isVisible: true เพื่อให้แสดงอยู่ในกราฟ

บรรทัดที่ 105 กำหนดให้แสดง tooltip เมื่อเอาเมาส์ไปชี้ไปที่กราฟ

บรรทัดที่ 112-113 กำหนด series ให้กับกราฟ โดยกำหนดให้เป็น StackedLineSeries ซึ่งเป็นกราฟเส้นที่เรียงต่อกันและจะแสดงผลเป็นกราฟแท่งกับแกน Y ซึ่งจะแสดงข้อมูล Capacity และ xValueMapper และ yValueMapper จะกำหนดให้แสดงข้อมูล Date_Time และ Capacity ตามลำดับ โดยจะอ้างอิงจาก DataSource ที่กำหนดเข้ามา

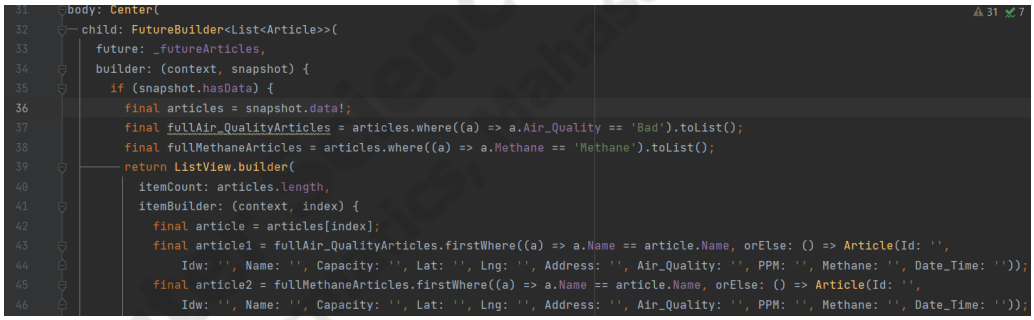
บรรทัดที่ 116-117 กำหนดแกน X ให้เป็น DateTimeAxis ซึ่งจะแสดงวันที่และเวลา และกำหนดให้ minimum ของแกน X เป็นเวลา 6 AM

3.6.3.5 หน้า Detail Screen

```
Text(
  'ID: ${article.Idw}',
  style: TextStyle(fontSize: 20),
), // Text
SizedBox(height: 8),
Text(
  'Capacity: ${article.Capacity+"%"}',
  style: TextStyle(fontSize: 20),
), // Text
SizedBox(height: 8),
Text(
  'Air Quality: ${article.Air_Quality}',
  style: TextStyle(fontSize: 20),
), // Text
SizedBox(height: 8),
Text(
  'PPM: ${article.PPM}',
  style: TextStyle(fontSize: 20),
), // Text
SizedBox(height: 8),
Text(
  'Methane: ${article.Methane}',
  style: TextStyle(fontSize: 20),
), // Text
SizedBox(height: 8),
Text(
  'Date/Time: ${article.Date_Time}',
```

ภาพประกอบที่ 3.36 โค้ดในส่วนแสดงรายละเอียดทั้งหมด

3.6.3.6 หน้า Notification



```
body: Center(
  child: FutureBuilder<List<Article>>(
    future: _futureArticles,
    builder: (context, snapshot) {
      if (snapshot.hasData) {
        final articles = snapshot.data!;
        final fullAir_QualityArticles = articles.where((a) => a.Air_Quality == 'Bad').toList();
        final fullMethaneArticles = articles.where((a) => a.Methane == 'Methane').toList();
        return ListView.builder(
          itemCount: articles.length,
          itemBuilder: (context, index) {
            final article = articles[index];
            final article1 = fullAir_QualityArticles.firstWhere((a) => a.Name == article.Name, orElse: () => Article(Id: '',
                Idw: '', Name: '', Capacity: '', Lat: '', Lng: '', Address: '', Air_Quality: '', PPM: '', Methane: '', Date_Time: ''));
            final article2 = fullMethaneArticles.firstWhere((a) => a.Name == article.Name, orElse: () => Article(Id: '',
                Idw: '', Name: '', Capacity: '', Lat: '', Lng: '', Address: '', Air_Quality: '', PPM: '', Methane: '', Date_Time: ''));
```

ภาพประกอบที่ 3.37 โค้ดในส่วนการทำNotification

บรรทัดที่ 31 widget Center ซึ่งจะแสดง Widget ที่อยู่ตรงกลางจอ

บรรทัดที่ 32-34 ใช้ FutureBuilder ในการดึงข้อมูล List<Article> จากตัวแปร _futureArticles และสร้าง ListView โดยใช้ builder เพื่อสร้าง Widget จาก snapshot

บรรทัดที่ 35 ใน builder จะเริ่มต้นด้วยการตรวจสอบว่า snapshot มีข้อมูลหรือไม่ ถ้ามีข้อมูลจะนำข้อมูล List<Article> ที่ได้จาก snapshot.data มาใช้งาน

บรรทัดที่ 37-38 ใช้ where() method เพื่อกรอง Article ที่มี Air_Quality เท่ากับ 'Bad' และ Methane เท่ากับ 'Methane' จาก articles และนำมาเก็บไว้ใน fullAir_QualityArticles และ fullMethaneArticles ตามลำดับ

บรรทัดที่ 39-40 สร้าง ListView.builder โดยใช้ itemCount เท่ากับจำนวน articles และใช้ itemBuilder เพื่อสร้าง Widget ในแต่ละรายการบทความ

บรรทัดที่ 42 นำข้อมูล Article ใน index มาเก็บไว้ในตัวแปร article

บรรทัดที่ 43-46 ใช้ firstWhere() method เพื่อค้นหา Article ใน fullAir_QualityArticles และ fullMethaneArticles ที่มีชื่อเท่ากับ article.Name และเก็บไว้ใน article1 และ article2

```
47        return Column(
48          children: [
49            if(article.Capacity == '100')
50              ListTile(
51                title: Text(article.Name),
52                subtitle: Text('Capacity: ${article.Capacity}'),
53                trailing: Icon(Icons.notification_important),
54              ),  // ListTile
55            if (article1.Air_Quality == 'Bad') // Check if Air_Quality is bad
56              ListTile(
57                title: Text('Air Quality Notification'),
58                subtitle: Text('Air Quality is bad at ${article1.Name}'),
59                trailing: Icon(Icons.notification_important),
60              ),  // ListTile
61            if (article2.Methane == 'Methane') // Check if Methane is detected
62              ListTile(
63                title: Text('Methane Notification'),
64                subtitle: Text('Methane is detected at ${article2.Name}'),
65                trailing: Icon(Icons.notification_important),
66              ),  // ListTile
67          ],
68        );  // Column
69      },
70    );  // ListView.builder
```

ภาพประกอบที่ 3.38 โค้ดกำหนดเงื่อนการแจ้งเตือน

บรรทัดที่ 49-53 เช็คเงื่อนไขว่าถ้าค่า Capacity ของ article เท่ากับ '100' ให้แสดงผล ListTile Widget ที่มี title เป็นชื่อของ article และ subtitle เป็นข้อความ 'Capacity: ${article.Capacity}' พร้อมกับแสดงไอคอน Icons.notification_important ด้านขวาสุดของ ListTile

บรรทัดที่ 55-59 เช็คเงื่อนไขว่าถ้าค่า Air_Quality ของ article1 เท่ากับ 'Bad' ให้แสดงผล ListTile Widget ที่มี title เป็น 'Air Quality Notification' และ subtitle เป็นข้อความ 'Air Quality is bad at ${article1 .Name}' พร้อมกับแสดงไอคอน Icons.notification_important ด้านขวาสุดของ ListTile

บรรทัดที่ 61-65 เช็คเงื่อนไขว่าถ้าค่า Methane ของ article2 เท่ากับ 'Methane' ให้แสดงผล ListTile Widget ที่มี title เป็น 'Methane Notification' และ subtitle เป็นข้อความ 'Methane is detected at ${article2.Name}' พร้อมกับแสดงไอคอน Icons.notification_important ด้านขวาสุดของ ListTile

3.6.3.7 หน้าการเข้าสู่ระบบ

```
20  Future Login() async{
21    var url = "https://proesp32.000webhostapp.com/getUser.php";
22    var response = await http.post(Uri.parse(url), body: {
23      "username" : _usernameController.text,
24      "password" : _passwordController.text,
25    });
26    var data = json.decode(response.body);
27    if(data == "Success"){
28      Fluttertoast.showToast(
29          msg: "Login Successful",
30          toastLength: Toast.LENGTH_SHORT,
31          gravity: ToastGravity.BOTTOM,
32          timeInSecForIosWeb: 1,
33          backgroundColor: Colors.red,
34          textColor: Colors.white,
35          fontSize: 16.0
36      );
37      // Navigate to the next page.
38      Navigator.push(
39          context,
40          MaterialPageRoute(
41          builder: (context) => MyHomePage(),
42    ),
43        );
44    }else{
45      Fluttertoast.showToast(
46          msg: "Username or Password Incorrect!",
```

ภาพประกอบที่ 3.39 โค้ดในส่วนเข้าสู่ระบบ

บรรทัดที่ 21 – 24 สร้างฟังก์ชั่นเพื่อจัดการคำขอการเข้าสู่ระบบส่งคำขอ HTTP POST ไปยังเซิร์ฟเวอร์โดยใช้ไลบรารี http คำสำคัญ wait ใช้เพื่อรอการตอบกลับของเซิร์ฟเวอร์ก่อนที่จะดำเนินการฟังก์ชั่นต่อไป

บรรทัด 27-46 ตรวจสอบ username และ password มีใน database หรือไม่ ถ้ามีจะเข้าสู่ระบบสำเร็จ ถ้าไม่ก็ไม่สำเร็จ