

Final project Report

(Subject: 2110366 Embedded Systems Lab)

Group คัดไม่ออกครับ

6231306621 Kunanont Kaewkhampa

6430315221 Phuvanach Phoemphol

6431340021 Worameth Suwannapruk

6430274121 Peeraphat Wongsri

Section 3

Role and Responsibility

หน้าที่ได้ถูกแบ่งออกเป็นส่วนย่อยได้ดังนี้

1. system architecture
2. front-end development
3. back-end development
4. embedded system development
5. UI designer and development

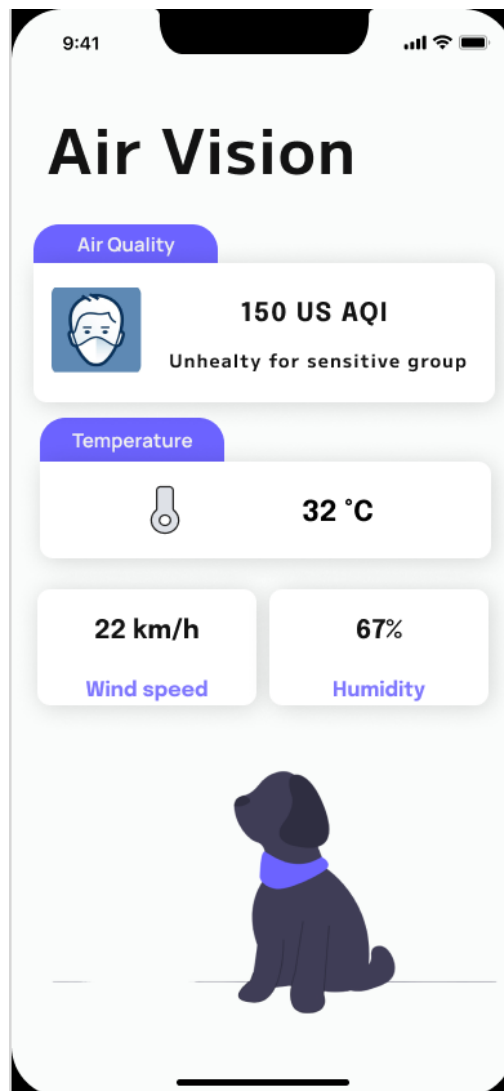
Role	Responsibility by
system architecture	Everyone
front-end development	Peeraphat Wongsri
back-end development	Phuvanach Phoemphol
embedded system development(stm32)	Worameth Suwannapruk
embedded system development(nodemcu)	Kunanont Kaewkhampa
UI designer and development	Everyone

UX/UI design

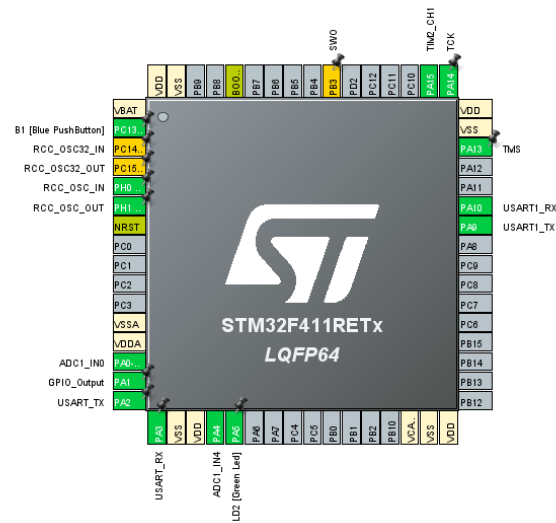
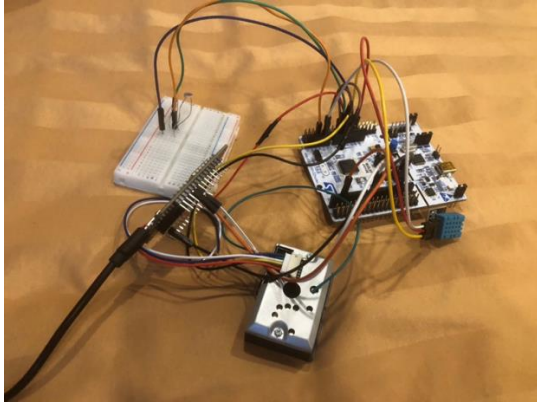
ทางกลุ่มเราได้ตกลงกันว่าจะทำเว็บที่ใช้สำหรับวัดคุณภาพอากาศ ที่สามารถดูค่าต่างๆ ได้ดังนี้

1. คุณภาพอากาศ(ปริมาณ PM 2.5)
2. อุณหภูมิ
3. ความชื้น
4. ความเข้มแสง

เมื่อตกลงกันได้ทางกลุ่มเราจึงได้ช่วยกันออกแบบหน้า UX/UI ซึ่งได้ออกมาดังรูป



system architecture



ในการต่อวงจรทางเราได้ใช้ sensor 3 ชนิด ได้แก่ LDR, DHT11 และ Dust sensor โดยสายส่งข้อมูลที่ส่งเข้า stm32 มีดังนี้

1. Dust sensor ใช้ PA0
2. DHT11 ใช้ PA1
3. LDR ใช้ PA4

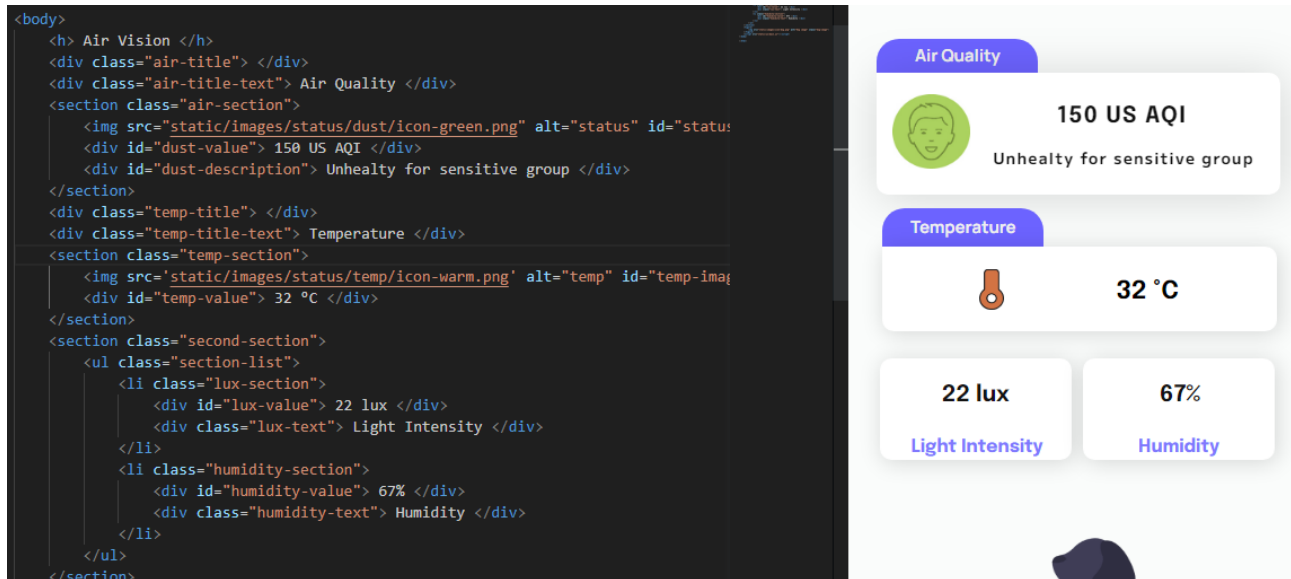
โดย sensor ทุกตัวต่อกับ ground และ power

ในส่วนการส่งข้อมูลจาก stm32 ไปยัง nodemcu นั้นใช้ UART1 ในการส่ง ซึ่งใช้ RX เป็น PA10 และ TX เป็น PA9 ในส่วนของ nodemcu นั้น RX ใช้ PIN 13 และ TX PIN 15 ซึ่งทางกลุ่มของเราได้ทำการต่อ RX ของ stm32 เข้ากับ TX ของ nodemcu และ ต่อ RX ของ nodemcu เข้ากับ TX ของ stm32 และ ต่อ ground ของทั้ง 2 บอร์ดเข้าด้วยกัน

Front-end

ในส่วนของ Front-end นั้นจะถูกแบ่งออกเป็น 3 ส่วน ได้แก่

1. html ที่ใช้ในการแสดง UX/UI ของกลุ่มเรา



โดยในส่วนของ html นั้นจะมี class สำคัญได้แก่

1. "air-section" แสดงค่าคุณภาพอากาศที่วัดได้จาก dust sensor
2. "temp-title" แสดงค่าอุณหภูมิที่วัดได้จาก DHT11
3. "lux-section" แสดงค่าความเข้มแสงที่วัดได้จาก LDR
4. "humidity-section" แสดงค่าความชื้นที่วัดได้จาก DHT11

2. css ใช้ในการตกแต่งhtml

3. JavaScript ที่มี function ที่ใช้ในการดึงข้อมูลจาก Back-end ขึ้นมาแสดงบนหน้า html ซึ่งมี 2 function เป็น function 1 อัน และ function ย่อยด้านใน function หลักอีก 1 อัน โดยฟังก์ชันหลักมีดังนี้

```
const fetchDeviceStatusShadow = async () => {  
  try {  
    const options = {  
      method: 'GET',  
      //headers: {  
        //'Authorization': 'Device 4513aa93-b88c-4e9b-92d3-11ba2920a2ee:Z7XT8mdToAGAYFU8898Xym7sgyBiC64K',  
        //},  
      credentials: 'include',  
    };  
    const response = await fetch(`http://${backendIPAddress}/api/device/shadow/data`, options);  
    const data = await response.json();  
    console.log(data);  
    const dataTable = const dataTable: any  
    updateUIWithData(dataTable.dust, dataTable.temp, dataTable.humidity, dataTable.lux);  
  } catch (error) {  
    console.error('Error:', error);  
  }  
};  
  
fetchDeviceStatusShadow();  
  
setInterval(fetchDeviceStatusShadow, 500);
```

function fetchDeviceStatusShadow

```
const updateUIWithData = (dust, temp, humidity, lux) => {
  dust = Math.round(dust);
  document.getElementById('dust-value').innerHTML = dust + ' AQI';
  document.getElementById('temp-value').innerHTML = temp + ' °C';
  document.getElementById('humidity-value').innerHTML = humidity + '%';
  document.getElementById('lux-value').innerHTML = lux + ' lux';

  const statusImage = document.getElementById('status-image');
  const dustDescription = document.getElementById('dust-description');
  const tempImage = document.getElementById("temp-image");
  if (dust >= 301) {
    statusImage.src = 'static/images/status/dust/icon-maroon.png';
    dustDescription.innerHTML = 'Hazardous';
  } else if (dust >= 201) {
    statusImage.src = 'static/images/status/dust/icon-purple.png';
    dustDescription.innerHTML = 'Very Unhealthy';
  } else if (dust >= 151) {
    statusImage.src = 'static/images/status/dust/icon-red.png';
    dustDescription.innerHTML = 'Unhealthy';
  } else if (dust >= 101) {
    statusImage.src = 'static/images/status/dust/icon-orange.png';
    dustDescription.innerHTML = 'Unhealthy for Sensitive Groups';
  } else if (dust >= 51) {
    statusImage.src = 'static/images/status/dust/icon-yellow.png';
    dustDescription.innerHTML = 'Moderate';
  } else {
    statusImage.src = 'static/images/status/dust/icon-green.png';
    dustDescription.innerHTML = 'Good';
  }
  if (temp >= 40) {
    tempImage.src = 'static/images/status/temp/icon-hot.png'
  }
  else if (temp >= 25) {
    tempImage.src = 'static/images/status/temp/icon-warm.png'
  }
  else {
    tempImage.src = 'static/images/status/temp/icon-cool.png'
  }
};
```

Function updateUIWithData









fetchDeviceStatusShadow จะเป็น function ที่ทำการดึงข้อมูลจาก Back-end มาและเรียก updateUIWithData ที่เป็น function ย่อยของ fetchDeviceStatusShadow ซึ่งจะทำหน้าที่ดังนี้

1. แก้ไขตัวเลขทุกค่าบน html ให้เป็นค่าเดียวกับที่ได้รับมาจาก Back-end
2. ทำการเปลี่ยนคำอธิบายของ Air Quality ให้เป็นไปตามเกณฑ์ที่กำหนดไว้
3. ทำการเปลี่ยนรูปของ Air Quality และ Temperature ให้เป็นไปตามเกณฑ์ที่กำหนดไว้

Back-end

ทำหน้าที่ fetch ข้อมูลจาก NETPIE ผ่าน RESTful API และส่งให้กับ frontend

ในส่วนของโฟลเดอร์ จะประกอบด้วยไฟล์ต่าง ๆ ตามแสดงดังภาพ

	Armph Access-Control-Allow-Origin	87280a2 last week	🕒 5 commits
	.gitignore	add device/status endpoint	last week
	README.md	Create README.md	last week
	app.js	add device/status endpoint	last week
	controller.js	Add device/shadow/data API	last week
	package.json	Access-Control-Allow-Origin	last week
	routes.js	Add device/shadow/data API	last week
	server.js	Access-Control-Allow-Origin	last week

- server.js

```
1 const express = require('express');
2 const routes = require('./routes');
3 const cors = require('cors');
4 require('dotenv').config();
5
6 const app = express();
7
8 app.use((req, res, next) => {
9   res.header('Access-Control-Allow-Origin', 'http://127.0.0.1:5500');
10  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept, Authorization');
11  res.header('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE');
12  res.header('Access-Control-Allow-Credentials', 'true');
13  next();
14 });
15
16 app.use(express.json());
17 app.use('/api', routes);
18
19 const port = process.env.PORT || 3000;
20
21 app.listen(port, () => {
22   console.log('Server is running on port ${port}');
23 });
```

เปิดเซิร์ฟเวอร์ที่ local port 3000, พร้อมให้
Access-Control กับ Frontend
(ณ ที่นี้คือ Port 5000),

- routes.js

กำหนด EndPoint สำหรับรับข้อมูลต่าง ๆ
ตามจำเป็น เพื่อให้ Controller ทำงาน
ต่อไป

```
1 require('dotenv').config();
2 const express = require('express');
3 const router = express.Router();
4 const controller = require('./controller');
5
6 router.get('/device/status', controller.getDeviceStatus);
7 router.get('/device/shadow/data', controller.getDeviceStatusShadow);
8
9 module.exports = router;
```


- controller.js

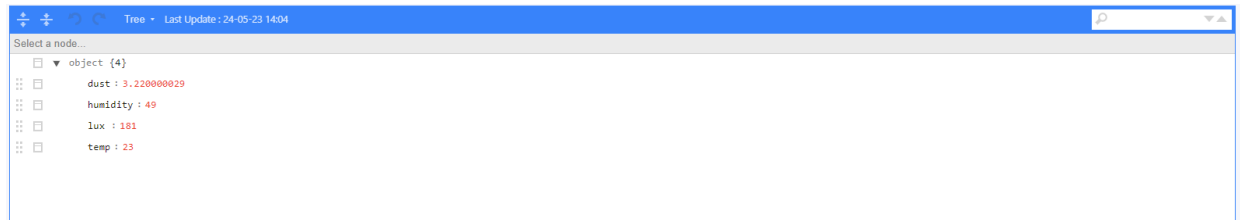
```
1  const axios = require('axios');
2  const { Buffer } = require('buffer');
3
4  exports.getDeviceStatus = async (req, res) => {
5    const apiUrl = 'https://api.netpie.io/v2/device/status';
6    const clientId = process.env.CLIENT_ID;
7    const token = process.env.TOKEN;
8    try {
9      const authHeader = `Basic ${Buffer.from(`${clientId}:${token}`).toString('base64')}`;
10
11      const response = await axios.get(apiUrl, {
12        headers: {
13          Authorization: authHeader,
14        },
15      });
16
17      res.json(response.data);
18    } catch (error) {
19      console.error('Error:', error);
20      res.status(500).json({ error: 'fetch error krub, so sad' });
21    }
22  };
23
24  exports.getDeviceStatusShadow = async (req, res) => {
25    const apiUrl = 'https://api.netpie.io/v2/device/shadow/data';
26    const clientId = process.env.CLIENT_ID;
27    const token = process.env.TOKEN;
28    try {
29      const authHeader = `Basic ${Buffer.from(`${clientId}:${token}`).toString('base64')}`;
30
31      const response = await axios.get(apiUrl, {
32        headers: {
33          Authorization: authHeader,
34        },
35      });
36
37      res.json(response.data);
38    } catch (error) {
39      console.error('Error:', error);
40      res.status(500).json({ error: 'fetch error krub, so sad' });
41    }
42  };
```

สำหรับ EndPoint getDeviceStatus จะเป็นการคืนสถานะของอุปกรณ์ที่เรากำลังสนใจ ซึ่งจำเป็นต้องมีในส่วน
ของ clientId กับ token ซึ่งเก็บไว้ในไฟล์ .env ซึ่งค่าเหล่านี้ ต้องเอามาจาก device ที่สร้างไว้ใน NETPIE

Key

Client ID	<input type="text" value="9142aafd-879f-486f-968b-680293252f98"/>	Copy
Token	<input type="text" value="bG15dSwp9SiTF1NBKfNYyTDF6QjsHWkw"/>	Copy
Secret	<input type="text" value="4RzrT2lmQ8OPSoyLl49bb~V6WZPQHZuB"/>	Copy
Status	Offline	Refresh
Enable	<input checked="" type="checkbox"/>	

สำหรับ `getDeviceStatusShadow` จะทำการดึงข้อมูล ที่มีใน device ที่เราสนใจ (ตามไฟล์ `.env`) ซึ่งในโปรเจกต์นี้ได้ เก็บข้อมูลไว้ทั้งหมด 4 ค่า ได้แก่ 1.dust 2.humidity 3.lux 4.temp



Stm32 Code

ในส่วนนี้จะเป็นการเขียนโค้ดเพื่ออ่านค่าต่างๆจาก sensor ทั้ง 3 ตัว แล้วส่งค่าผ่าน UART ไปให้ ESP8266 หลักๆการรับค่าจาก sensor จะอยู่ในส่วน while แต่สำหรับ sensor แต่ละตัวก็จะต้องมีการกำหนดค่าต่างๆก่อนเข้า while ด้วย

1. เตรียม ADC สำหรับ Dust sensor และ LDR เนื่องจากค่าที่ได้จาก sensor 2 ตัวนี้เป็นแบบ analog แต่เราต้องการข้อมูลแบบ digital ซึ่งสำหรับ Dust sensor จะใช้ ADC channel 0 และ LDR จะใช้ channel 4

```
void ADC_Select_CH0 (void)
{
    ADC_ChannelConfTypeDef sConfig = {0};
    /** Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
    sConfig.Channel = ADC_CHANNEL_0;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
    {
        Error_Handler();
    }
}

void ADC_Select_CH4 (void)
{
    ADC_ChannelConfTypeDef sConfig = {0};
    /** Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
    sConfig.Channel = ADC_CHANNEL_4;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
    {
        Error_Handler();
    }
}
```

2. เตรียมฟังก์ชันสำหรับการอ่านค่าของ sensor DHT11

2.1 DHT11_Start(void) สำหรับการเตรียม DHT11 ของเราให้พร้อมอ่านค่าและเริ่มอ่านค่า

```
uint8_t DHT11_Start (void)
{
    uint8_t Response = 0;
    GPIO_InitTypeDef GPIO_InitStructPrivate = {0};
    GPIO_InitStructPrivate.Pin = DHT11_PIN;
    GPIO_InitStructPrivate.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructPrivate.Speed = GPIO_SPEED_FREQ_LOW;
    GPIO_InitStructPrivate.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(DHT11_PORT, &GPIO_InitStructPrivate); // set the pin as output
    HAL_GPIO_WritePin (DHT11_PORT, DHT11_PIN, 0); // pull the pin low
    HAL_Delay(20); // wait for 20ms
    HAL_GPIO_WritePin (DHT11_PORT, DHT11_PIN, 1); // pull the pin high
    microDelay (30); // wait for 30us
    GPIO_InitStructPrivate.Mode = GPIO_MODE_INPUT;
    GPIO_InitStructPrivate.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(DHT11_PORT, &GPIO_InitStructPrivate); // set the pin as input
    microDelay (40);
    if (!(HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)))
    {
        microDelay (80);
        if ((HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN))) Response = 1;
    }
    pMillis = HAL_GetTick();
    cMillis = HAL_GetTick();
    while ((HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)) && pMillis + 2 > cMillis)
    {
        cMillis = HAL_GetTick();
    }
    return Response;
}
```

2.2 DHT11_Read(void) สำหรับการอ่านข้อมูลโดยจะค่อยๆส่งมาทีละอย่าง โดยจะส่งจำนวนเต็มมา

ก่อนแล้วค่อยส่งทศนิยมมา เราจะได้ค่าของ ความชื้นสัมพัทธ์ และ อุณหภูมิมา

```
uint8_t DHT11_Read (void)
{
    uint8_t a,b;
    for (a=0;a<8;a++)
    {
        pMillis = HAL_GetTick();
        cMillis = HAL_GetTick();
        while (!(HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)) && pMillis + 2 > cMillis)
        { // wait for the pin to go high
            cMillis = HAL_GetTick();
        }
        microDelay (40); // wait for 40 us
        if (!(HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN))) // if the pin is low
            b|= ~(1<<(7-a));
        else
            b|= (1<<(7-a));
        pMillis = HAL_GetTick();
        cMillis = HAL_GetTick();
        while ((HAL_GPIO_ReadPin (DHT11_PORT, DHT11_PIN)) && pMillis + 2 > cMillis)
        { // wait for the pin to go low
            cMillis = HAL_GetTick();
        }
    }
    return b;
}
```

3. Display(float x, int path) สำหรับการป้อนค่าออกมาใน terminal เพื่อเช็คผลลัพธ์ที่ได้จาก sensor

```
void Display (float x, int path)
{
    char str[20] = {0};
    switch (path) {
        case 0:
            sprintf (str, "dust:- %.2f\r\n ", x);
            break;
        case 1:
            sprintf (str, "temp:- %.2f\r\n ", x);
            break;
        case 2:
            sprintf (str, "humid:- %.2f\r\n ", x);
            break;
        default:
            sprintf (str, "lux:- %.2f\r\n ", x);
            break;
    }
    HAL_UART_Transmit(&huart2, &str, 20, 100);
}
```

4. ใน while เป็นการอ่านค่าโดยรวมและส่งค่าไปให้ Esp โดยแต่ละรอบของ while จะต้องมีการ delay ด้วยเพื่อให้ sensor ได้มีเวลาในการวัดและเพื่อให้ esp มีเวลาสำหรับการรับข้อมูล

4.1 การอ่านค่าจาก sensor ทั้งสามตัว

```
while (1)
{
    if(DHT11_Start())
    {
        RHI = DHT11_Read(); // Relative humidity integral
        RHD = DHT11_Read(); // Relative humidity decimal
        TCI = DHT11_Read(); // Celsius integral
        TCD = DHT11_Read(); // Celsius decimal
        SUM = DHT11_Read(); // Check sum
    }

    //Dust
    ADC_Select_CH0();
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    voMeasured = HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);

    calcVoltage = voMeasured * (5.0f / 4096.0f);
    dustDensity = 170 * calcVoltage - 0.1;
    //End Dust

    //Lux
    ADC_Select_CH4();
    HAL_ADC_ConfigChannel(&hadc1, ADC_CHANNEL_4);
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    lux = HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
    //End LUX
}
```

4.2 ปรี้นค่าเพื่อเช็ค และเปลี่ยน format ของข้อมูล พร้อมกับส่ง text ไปให้ esp โดย UART

```
Display(dustDensity, 0);  
Display(TCI, 1);  
Display(RHI, 2);  
Display(lux, 3);  
int len = sprintf(text, "%.2f %d %d %.2f\n", dustDensity, TCI, RHI, lux);  
HAL_UART_Transmit(&huart1, text, len, HAL_MAX_DELAY);  
HAL_Delay(1000);
```

NodeMCU ESP8266

ส่วนของบอร์ด ESP8266 นั้นทำหน้าที่รับข้อมูลจากบอร์ด STM32 แล้วจึงส่งข้อมูลไปยังฐานข้อมูล ซึ่งกลุ่มของพวกเราใช้ NETPIE ในการจัดเก็บข้อมูลอธิบายโค้ดคร่าวๆได้ดังนี้

1. กำหนด PIN เพื่อรับข้อมูลจากบอร์ด STM32 โดยใช้ SoftwareSerial และใส่ Token ต่างๆให้ตรงกับอุปกรณ์ที่เราตั้งค่าไว้ใน NETPIE

```
#define rxPin 13
#define txPin 15

SoftwareSerial mySerial = SoftwareSerial(rxPin, txPin);

//กำหนดข้อมูลเชื่อมต่อ WiFi
const char* ssid = "Peemknn"; //ต้องแก้ไข
const char* password = "kunanontk"; //ต้องแก้ไข
//กำหนดข้อมูลเชื่อมต่อ NETPIE2020 นำ KEY ที่ได้จาก NETPIE มาใส่
const char* mqtt_server = "broker.netpie.io";
const char* client_id = "9142aafd-879f-486f-968b-680293252f98"; //ต้องแก้ไข
const char* token = "bG15dSwp9SiTF1NBKfNYyTDF6QJsHWkw"; //ต้องแก้ไข
const char* secret = "4RZrT2!mQ8OPS0yLl49bb~V6WZPQH ZuBM"; //ต้องแก้ไข
Adafruit_SHT31 sht31 = Adafruit_SHT31();
WiFiClient espClient;
PubSubClient client(espClient);
```

2. เขียนโค้ดเพื่อเชื่อมต่อบอร์ดกับ WIFI

```
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print(F("Connecting to "));
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(F("."));
  }
  Serial.println(F(""));
  Serial.println(F("WiFi connected"));
  Serial.println(F("IP address: "));
  Serial.println(WiFi.localIP());
}
```

3. Setup ทุกอย่างรวมถึง Serial monitor and NETPIE server

```
void setup() {  
  
    //ตั้งค่า serial monitor  
    Serial.begin(9600);  
    mySerial.begin(9600);  
  
    // เชื่อมต่อ WiFi  
    setup_wifi();  
    // ตั้งค่า Broker Netpie และพอร์ต  
    client.setServer(mqtt_server, 1883);  
    client.setCallback(callback);  
}
```

4. เขียน Loop เพื่อรับและส่งข้อมูลขึ้นบน NETPIE โดยจะมีฟังก์ชันแยกเพื่อ Split ค่าต่างๆใน String ของข้อมูลที่ได้รับมาจาก STM32

```
void loop() {  
    if (!client.connected()) {  
        reconnect();  
    }  
    client.loop();  
  
    // กำหนดให้ส่งข้อมูลทุกๆ 30 วินาที  
    static unsigned long lastTime = 0;  
    if (millis() - lastTime >= 1500 || lastTime > millis()) {  
        lastTime = millis();  
        // อ่านค่าอุณหภูมิ และความชื้น  
        if(mySerial.available() > 1){  
            receivedData = mySerial.readString();  
            //receivedData += ' ';  
            Serial.println(receivedData);  
            if(receivedData.length() >= 5){  
                dust = (getValue(receivedData, ' ',0).toFloat());  
                temp = (getValue(receivedData, ' ',1).toInt());  
                humid = (getValue(receivedData, ' ',2).toInt());  
                lux = (getValue(receivedData, ' ',3).toInt());  
            }  
            Serial.print(" dust => ");  
            Serial.println(dust);  
            Serial.print(" temp => ");  
            Serial.println(temp);  
            Serial.print(" humid => ");  
            Serial.println(humid);  
            Serial.print(" lux => ");  
            Serial.println(lux);  
        }  
    }  
}
```


5. ส่งข้อมูลไปที่ Shadow device ของ NETPIE ในรูปแบบของ JSON โดยจะแบ่งข้อมูลเป็นค่าฝุ่น อุณหภูมิ ความชื้น และค่าความเข้มของแสง

```
// กำหนดข้อมูลให้อยู่ในรูปแบบ Json NETPIE
const size_t capacity = JSON_OBJECT_SIZE(1) + JSON_OBJECT_SIZE(4);
StaticJsonDocument<capacity> doc;
JsonObject data = doc.createNestedObject("data");
data["dust"] = dust;
data["temp"] = temp;
data["humidity"] = humid;
data["lux"] = lux;
Serial.print(F("Message send: "));
Serial.println((doc.as<String>()).c_str());
// ส่งข้อมูลไปที่ Shadow Device
client.publish(UPDATEDATA, (doc.as<String>()).c_str());
}
```

6. เขียนโค้ดที่ไว้กำหนดว่าส่งข้อมูลไปที่ NETPIE ทุกๆกี่วินาทีต่อการส่งหนึ่งครั้ง ในที่นี้เรากำหนดเอาไว้ว่าส่งข้อมูลทุกๆ 1.5 วินาที

```
static unsigned long lastTime = 0;
if (millis() - lastTime >= 1500 || lastTime > millis()) {
    lastTime = millis();
}
```