80th Anniversary
KASETSART UNIVERSITY
Prosperity and Sustainability of the Nation
1943-2023

IOT Report

on

Dust-air monitoring with Netpie2020

By

Mr. Pongsathorn Utsahawattanasuk ID 6210554784

This term report is partial fulfillment of the requirement for 01205431 Internet of thing

December, 2022

Table of Contents

Preface

This report is the outcome of the IOT Project at Kasetsart University. The goal of this project is to apply the knowledge gained in the IOT course. This is an implementation of the concepts of MQTT, i2c communication, NodeMCU, and also verification techniques such as multitasking and EEPROM.

We would like to sincerely thank Dr. Varodom Toochinda for his advice and his GitHub channel, which allowed us to study it.

IoT project summary

1. Course project counts as 30% of total score.
2. You can do it individually, or in group of no more than 3 students. Every person in a group receives the same project score.
3. A report in electronic form (PDF, MS word, or Jupyter notebook), together with a video clip demonstration, are required.
4. Your project must have a hardware platform for your IoT, such as ESP8266, ESP32, Raspberry Pi, etc. Your group must acquire the hardware by your own. You don't need to submit any hardware along with the project.
5. Any IoT cloud can be used, such as NETPIE, Thingspeak, Thingsboard, Adafruit, Blynk, or from Google, Microsoft, Amazon etc., as long as it covers the functions in 6.
6. You IoT application should attempt to fulfill these requirements.
   A. It must display some variables for monitoring (humidity, temperature, pressure, velocity, distance etc.)
   B. It should be able to control some hardware (use a slider, button, toggle to adjust some parameter or change system state, etc.)
   C. It must compute something, ranging from some simple formula to more complex algorithm such as FFT, PID, digital signal processing. Score is expected to depend on how challenging is your development.
   D. The more advanced your IoT uses MCU resources (timer, processor cores) and coding techniques (ex. multitasking) , the more likely you'd score better on the project.
7. All class projects will be compared for novelty, creativity, complexity, practicality etc. and will be graded accordingly.
8. Report quality is also essential. it must be neat and professional, with sections like contents, introduction, materials, methods, conclusion, references. Pictures must be clear and numbered.
9. The deadline for project report is the final exam date from IUP office.
10. Only one member from each group submits the report and video clip to this assignment. In case there are duplicates, I would choose one randomly from a group member

Introduction

Everyone worldwide is affected by the serious environmental health issue of outdoor air pollution. In 2016, it was projected that outdoor air pollution in both urban and rural regions contributed to 4.2 million premature deaths annually worldwide. This mortality is attributable to exposure to fine particulate matter (PM2.5), which causes cancer, cardiovascular disease, and respiratory disease. The burden of outdoor air pollution is disproportionately felt by residents of low- and middle-income countries, accounting for 91% of the 4.2 million premature deaths worldwide. The WHO's South-East Asia and Western Pacific areas bear the brunt of this load. The most recent burden estimates show how seriously air pollution affects cardiovascular disease and mortality.

So, based on this concept, I want to implement an IOT device to monitor the dust and air quality, as well as add more features to this project. The data from this IOT device should be able to publish to the MQTT protocol. In this case, I chose the Netpie2020 cloud IOT platform.

Materials

Hardware

1. NodeMCU esp8266
2. bme280, pms7003 sensors
3. button, wires, resistor
4. OLED(ssd1306)
5. Laptop
6. NodeMCU esp32

Software

1. Arduino IDE link

IOT Cloud service

2. Netpie2020 link

Method

Components:

1. ESP8266 NodeMcu

NodeMCU is an open-source Lua-based firmware and development board specifically targeted for IoT-based applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems and hardware that is based on the ESP-12 module. The NodeMCU ESP8266 development board comes with the ESP-12E module containing the ESP8266 chip, which has a Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and has an adjustable clock frequency of 80 MHz to 160 MHz. NodeMCU has 128 KB of RAM and 4 MB of flash memory to store data and programs. Its high processing power, in-built Wi-Fi and Bluetooth, and deep sleep operating features make it ideal for IoT projects.

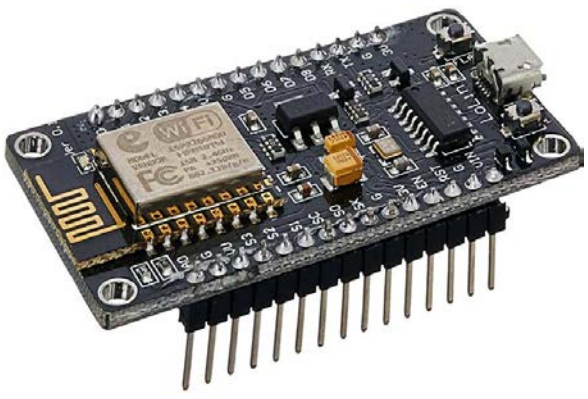


Figure 1 esp8266

2. <u>Bme280</u>

BME280 sensor, an environmental sensor with temperature, barometric pressure and humidity. This sensor is great for all sorts of indoor environmental sensing and can even be used in both I2C and SPI. This precision sensor from Bosch is the best low-cost sensing solution for measuring humidity with ±3% accuracy, barometric pressure with ±1 hPa absolute accuracy, and temperature with ±1.0°C accuracy. Because pressure changes with altitude, and the pressure measurements are so good, you can also use it as an altimeter with ±1 meter or better accuracy. It has the same specifications, but can use either I2C or SPI. For simple easy wiring, go with I2C.

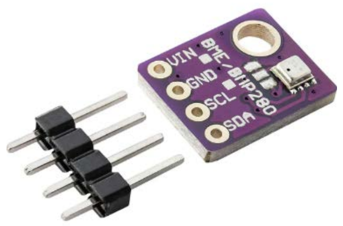


Figure 2 bme280 sensor

3. <u>Pms7003</u>

PMS7003 is a kind of digital and universal particle concentration sensor, which can be used to obtain the number of suspended particles in the air, i.e. the concentration of particles, and output them in the form of digital interface.[3] This sensor can be inserted into variable instruments related to the concentration of suspended particles in the air or other environmental improvement equipment's to provide correct concentration data in time.

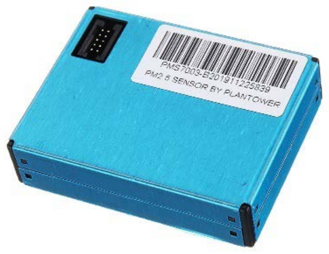
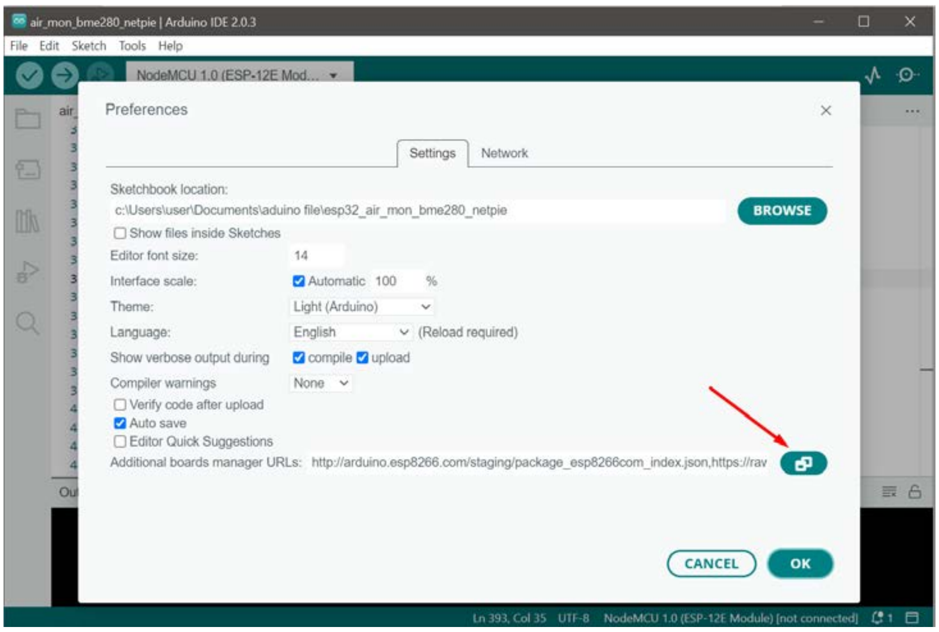


Figure 3 pms7003

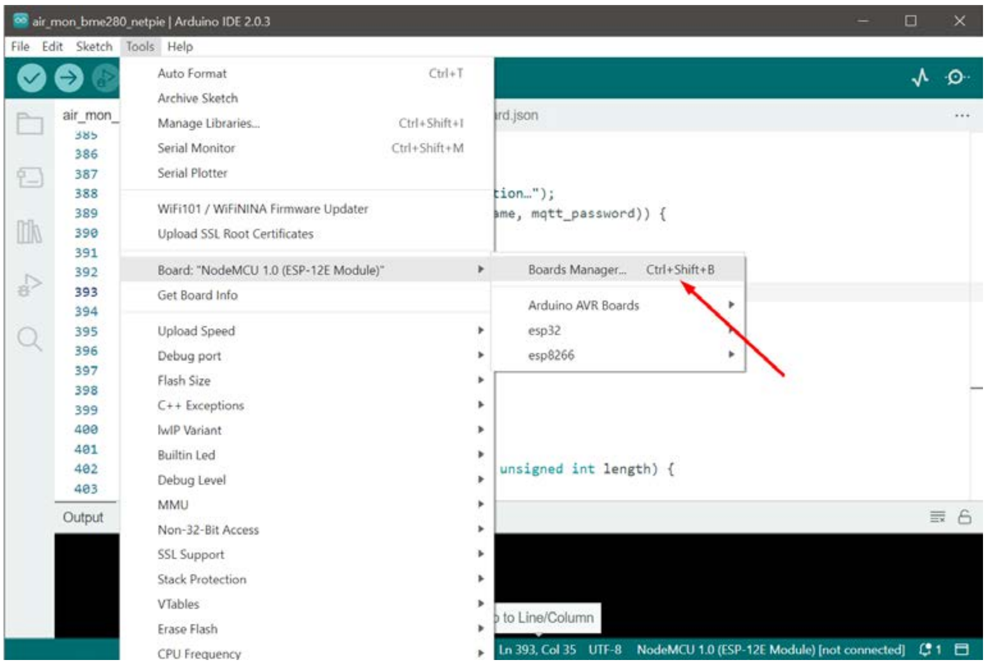
Prepare the Arduino IDE with esp8266

1. Download and install the Arduino IDE on your operating system.
2. Install the ESP8266 add-on for the Arduino IDE. Go to **file** then **Preference**
3. Enter the URL in the Additional Board Manager field: http://arduino.esp8266.com/staging/package_esp8266com_index.json
4. Then click on OK button

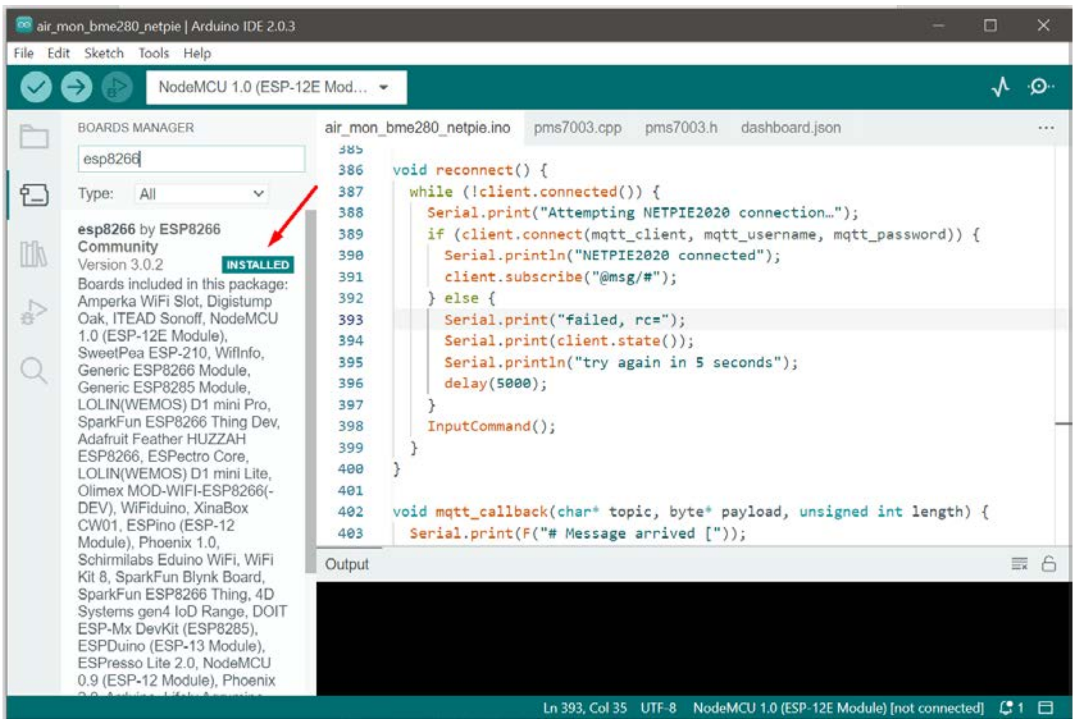

air_mon_bme280_netpie | Arduino IDE 2.0.3
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12E Mod...
Preferences
Settings
Network
BROWSE
CANCEL
OK
Ln 393, Col 35 UTF-8 NodeMCU 1.0 (ESP-12E Module) [not connected]

Sketchbook location: c:\Users\user\Documents\aduino file\esp32_air_mon_bme280_netpie
☐ Show files inside Sketches
Editor font size: 14
Interface scale: ☑ Automatic 100 %
Theme: Light (Arduino)
Language: English (Reload required)
Show verbose output during ☑ compile ☑ upload
Compiler warnings None
☐ Verify code after upload
☑ Auto save
☐ Editor Quick Suggestions
Additional boards manager URLs: http://arduino.esp8266.com/staging/package_esp8266com_index.json,https://rav

5. Go to **Tools** then **Board** then **Board Manager**


air_mon_bme280_netpie | Arduino IDE 2.0.3
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Board: "NodeMCU 1.0 (ESP-12E Module)"
Get Board Info
Upload Speed
Debug port
Flash Size
C++ Exceptions
lwIP Variant
Builtin Led
Debug Level
MMU
Non-32-Bit Access
SSL Support
Stack Protection
VTables
Erase Flash
CPU Frequency
Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
esp32
esp8266
Output
Ln 393, Col 35 UTF-8 NodeMCU 1.0 (ESP-12E Module) [not connected]

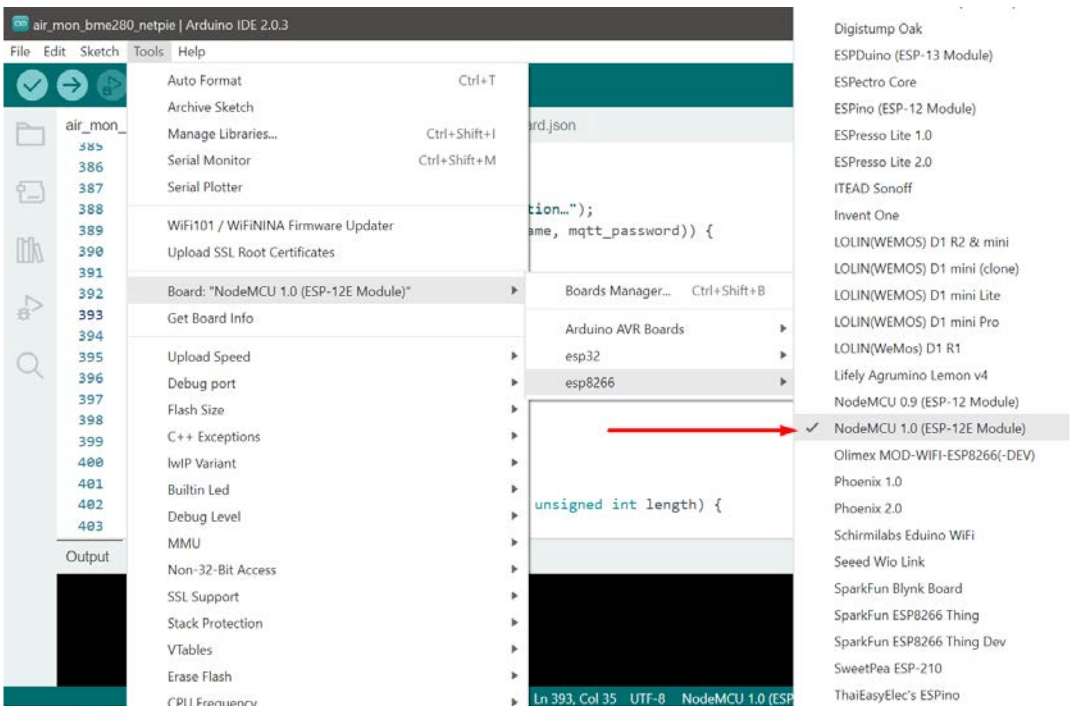
6. Type esp8266 then install the board.


air_mon_bme280_netpie | Arduino IDE 2.0.3
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12E Mod...
BOARDS MANAGER
esp8266
Type: All
air_mon_bme280_netpie.ino
pms7003.cpp
pms7003.h
dashboard.json
INSTALLED
Output
Ln 393, Col 35 UTF-8 NodeMCU 1.0 (ESP-12E Module) [not connected]

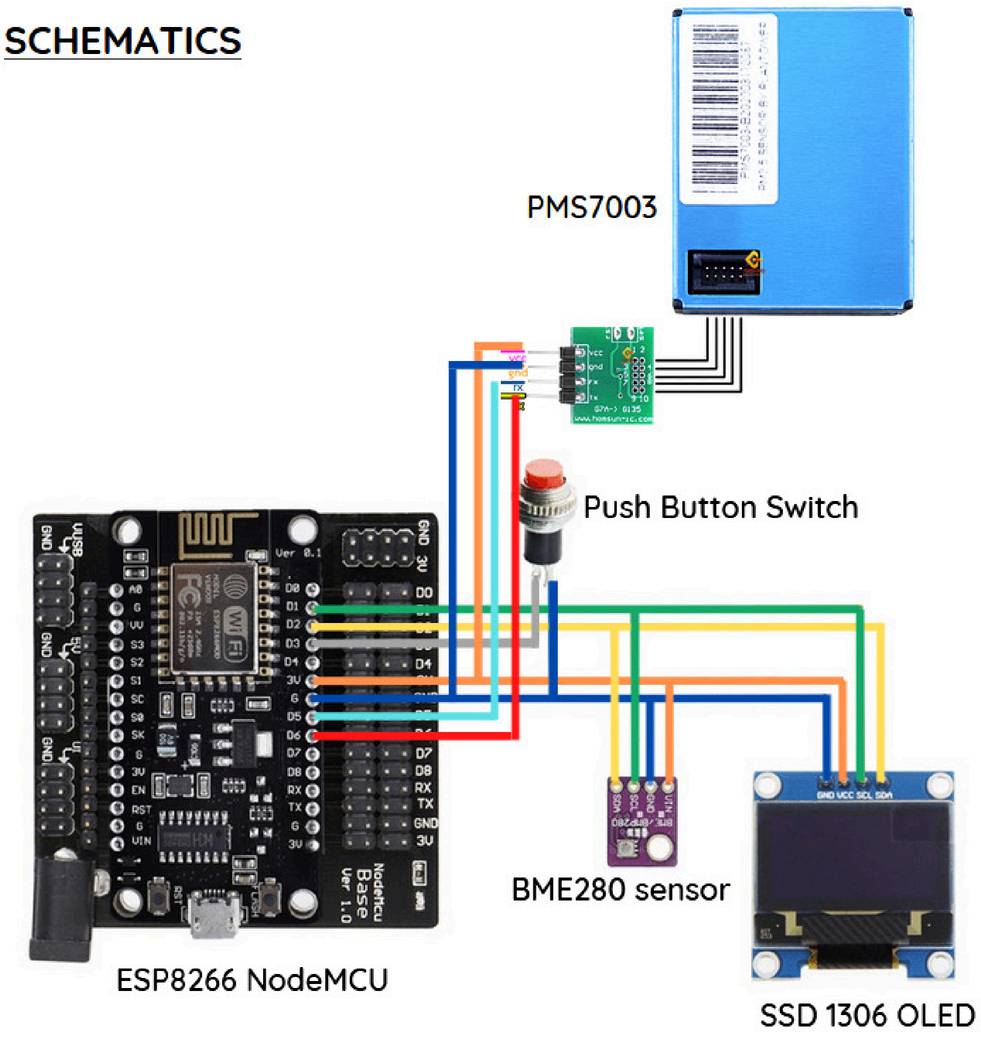
esp8266 by **ESP8266 Community**
Version 3.0.2
Boards included in this package: Amperka WiFi Slot, Digistump Oak, ITEAD Sonoff, NodeMCU 1.0 (ESP-12E Module), SweetPea ESP-210, WifInfo, Generic ESP8266 Module, Generic ESP8285 Module, LOLIN(WEMOS) D1 mini Pro, SparkFun ESP8266 Thing Dev, Adafruit Feather HUZZAH ESP8266, ESPectro Core, LOLIN(WEMOS) D1 mini Lite, Olimex MOD-WIFI-ESP8266(-DEV), WiFiduino, XinaBox CW01, ESPino (ESP-12 Module), Phoenix 1.0, Schirmilabs Eduino WiFi, WiFi Kit 8, SparkFun Blynk Board, SparkFun ESP8266 Thing, 4D Systems gen4 IoD Range, DOIT ESP-Mx DevKit (ESP8285), ESPDuino (ESP-13 Module), ESPresso Lite 2.0, NodeMCU 0.9 (ESP-12 Module), Phoenix

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting NETPIE2020 connection…");
    if (client.connect(mqtt_client, mqtt_username, mqtt_password)) {
      Serial.println("NETPIE2020 connected");
      client.subscribe("@msg/#");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
    InputCommand();
  }
}

void mqtt_callback(char* topic, byte* payload, unsigned int length) {
  Serial.print(F("# Message arrived ["));
```

7. In this case I'm working with NodeMCU then I choose NodeMCU 1.0 in esp8266



Schematics

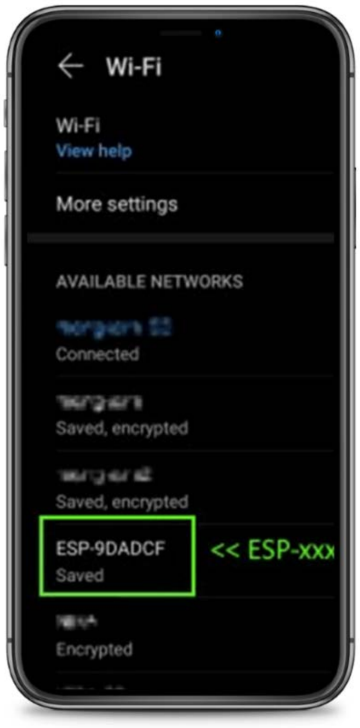


*****Be careful not to connect 3V and GND in reverse, otherwise it will damage the sensor*****

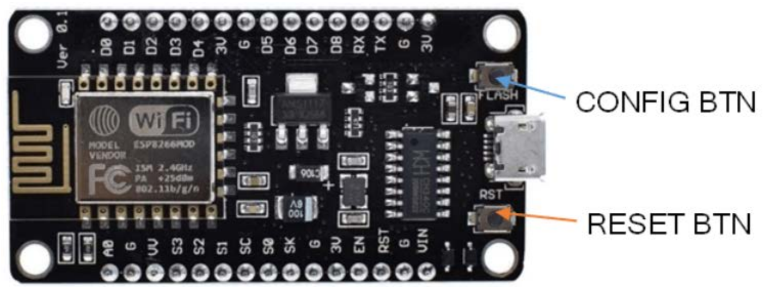
Wiring

NodeMCU	PMS7003	BME280	OLED	Button
3.3V	VCC	VCC	VCC	-
GND	GND	GND	GND	GND
GPIO5(D1)	-	SCL	SCL	-
GPIO4(D2)	-	SDA	SDA	-
GPIO0(D3)	-	-	-	VCC
GPIO14(D5)	Tx	-	-	-
GPIO12(D6)	Rx	-	-	-

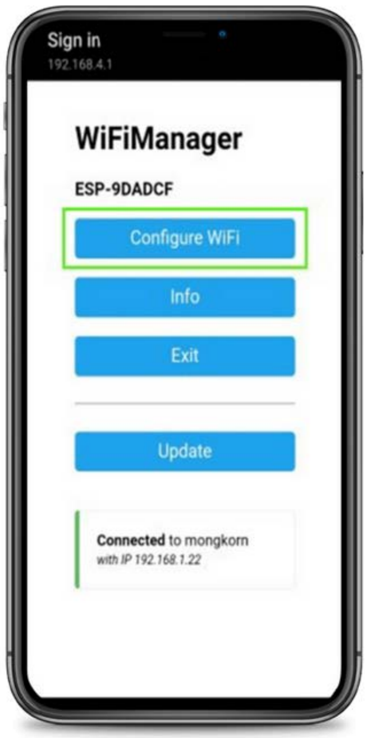
Connects the ESP8266 board to Wi-Fi.



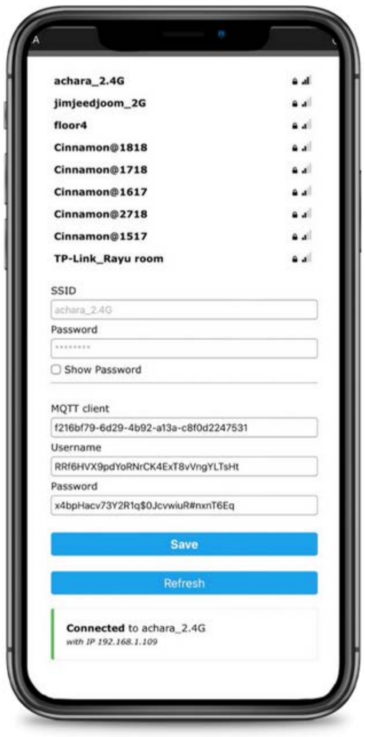
Step1 Press CONFIG button for 5 seconds to enter Wi-Fi configure mode. Board will work as Access point named ESP-xxxxxx. Use smart phone or notebook to connect to this access point.



Step2 Once connected, there will be a pop-up showing the Wi-Fi Manager page, press the "Configure Wi-Fi" button to set it up. If there is no pop-up, use a browser to open to the URL http://192.168.4.1.



Connected to mongkorn
with IP 192.168.1.22



SSID
achara_2.4G
Password
••••••••
☐ Show Password

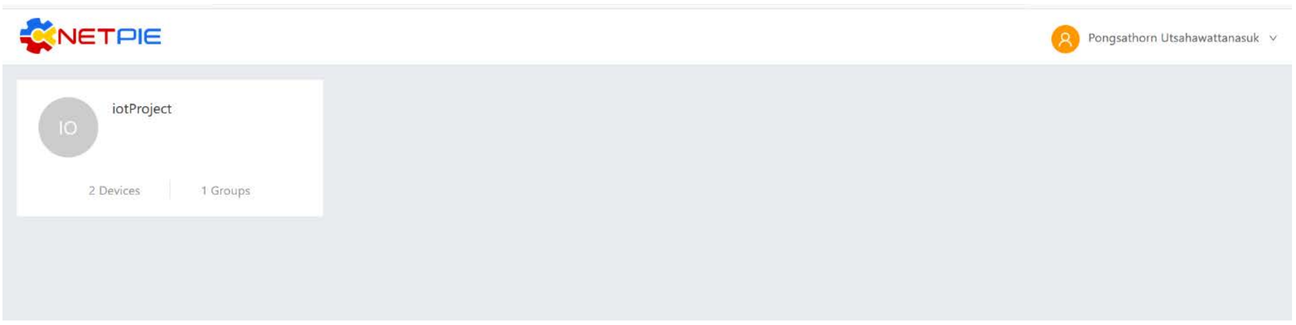
MQTT client
f216bf79-6d29-4b92-a13a-c8f0d2247531
Username
RRf6HVX9pdYoRNrCK4ExT8vVngYLTsHt
Password
x4bpHacv73Y2R1q$0JcvwiuR#nxnT6Eq

Connected to achara_2.4G
with IP 192.168.1.109

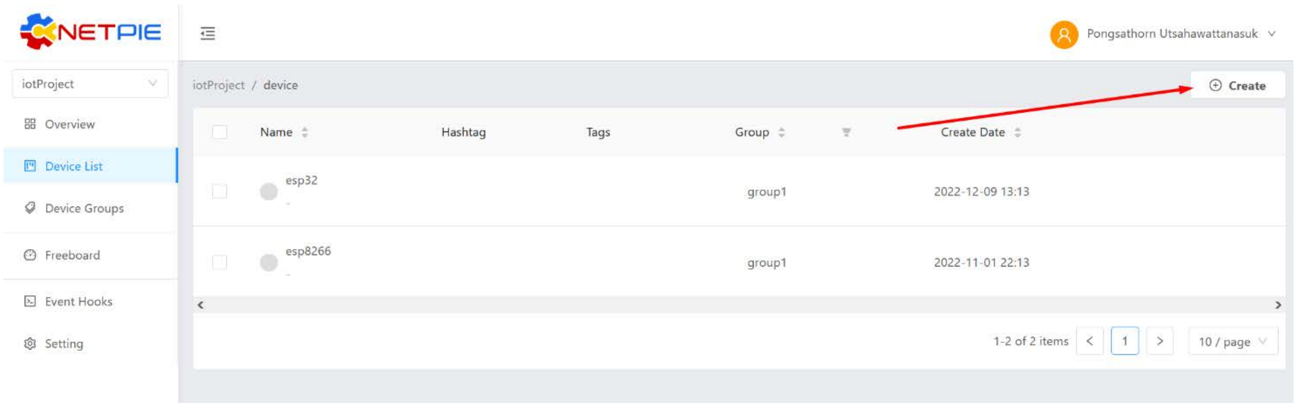
Step3

Settings: Wi-Fi: Tap to select Wi-Fi network or enter the name of the Wi-Fi network you want to use in the SSID field and enter the password. Then click save button

Connect to Netpie2020

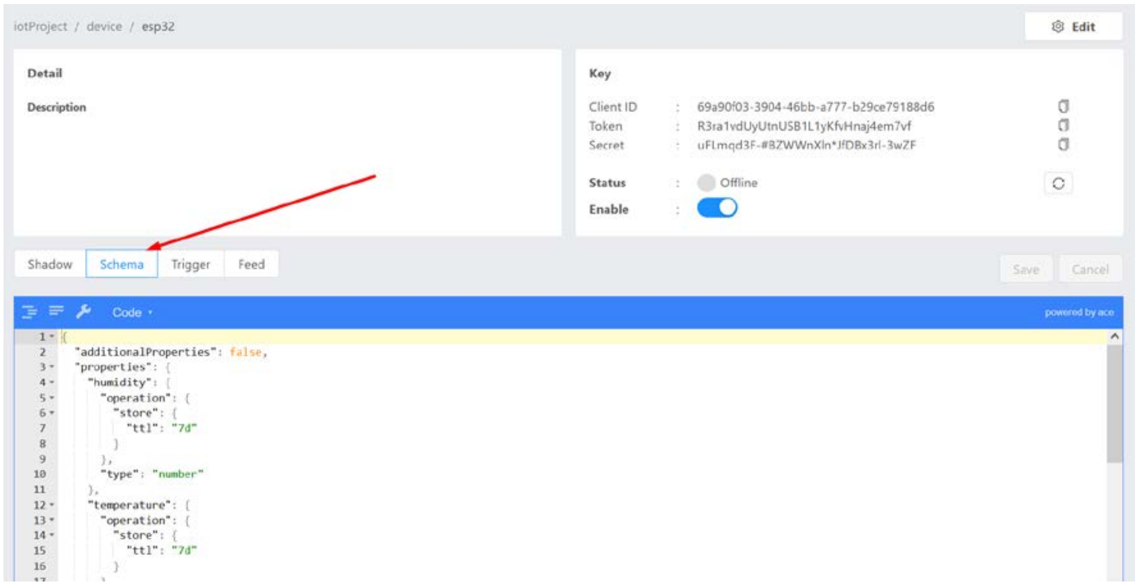


Step1 Create the project on Netpie2020.



	Name	Hashtag	Tags	Group	Create Date
	esp32			group1	2022-12-09 13:13
	esp8266			group1	2022-11-01 22:13

Step2 Create the device in that project. In order to get the ClientID, Token, and Secret.



Key

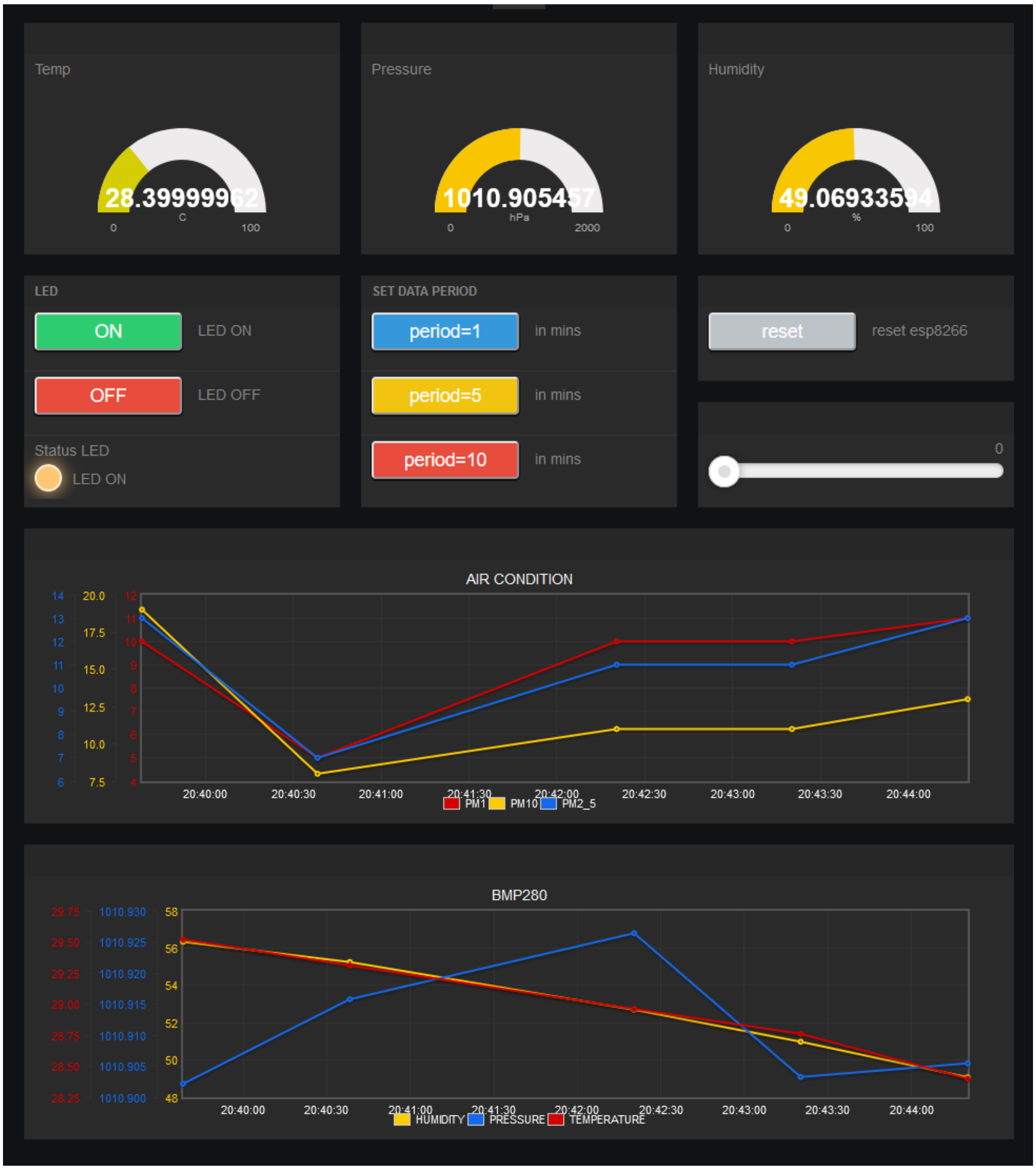
Client ID	69a90f03-3904-46bb-a777-b29ce79188d6
Token	R3ra1vdUyUtnUSB1L1yKfvHnaj4em7vf
Secret	uFLmqd3F-#BZWWnXln*JfDBx3rl-3wZF
Status	Offline
Enable	

```
{
  "additionalProperties": false,
  "properties": {
    "humidity": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "number"
    },
    "temperature": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
```

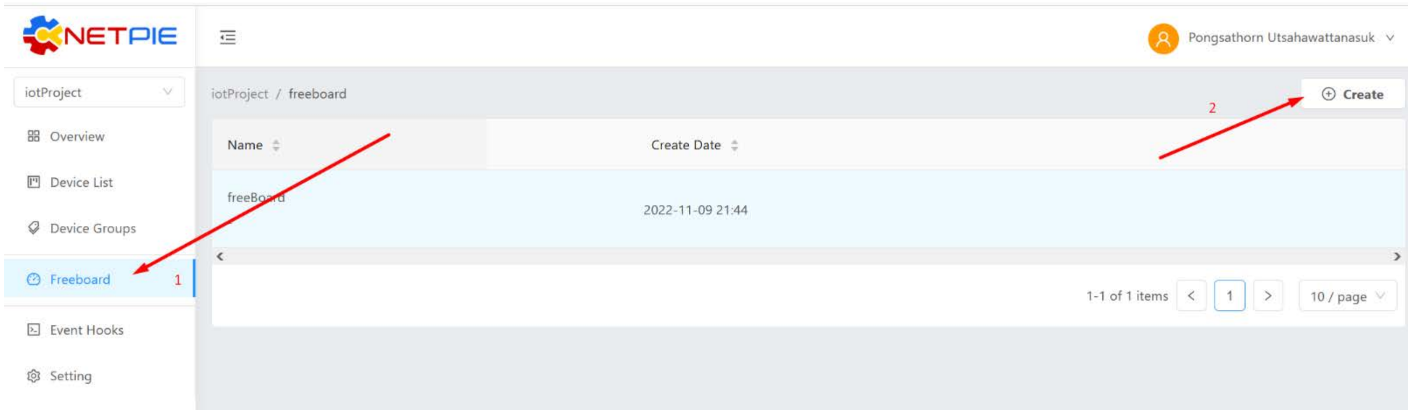
Step3 Add the data schema in order to store the data on Netpie shadow. I will provide my data schema below:

```
{
  "additionalProperties": false,
  "properties": {
    "humidity": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "number"
    },
    "temperature": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "number"
    },
    "pressure": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "number"
    },
    "pm1": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "number"
    },
    "pm2_5": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "number"
    },
    "pm10": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "number"
    },
    "led": {
      "operation": {
        "store": {
          "ttl": "7d"
        }
      },
      "type": "string"
    }
  }
}
```

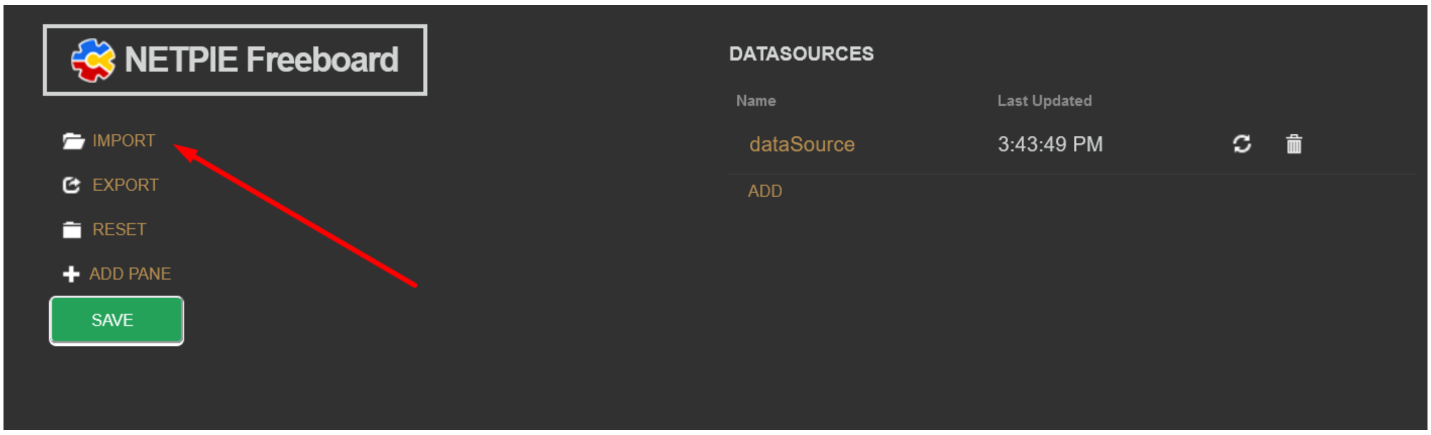
Netpie Freeboard



Step1 Create a New Freeboard by Login into the Netpie and click on Freeboard



Step2 Click on ADD in DATASOURCE



Step3 Upload the file in type of Jason but you need to change some information(I changed that data into red color)

```
{
 "version": "application/octet-stream",
 "allow_edit": true,
 "plugins": [],
 "panes": [
  {
   "width": 1,
   "row": {
    "3": 1
   },
   "col": {
    "3": 1
   },
   "col_width": 1,
   "widgets": [
    {
     "type": "gauge",
     "settings": {
      "title": "Temp",
      "value":
"datasources[\"dataSource\"][\"shadow\"][\"temperature\"]"
,
      "units": "C",
      "min_value": 0,
      "max_value": 100
     }
    }
   ]
  },
  {
   "width": 1,
   "row": {
    "3": 19
   },
   "col": {
    "3": 1
   },
   "col_width": "3",
   "widgets": [
    {
     "type": "nxpFeedView",
     "settings": {
      "title": "pm",
      "datasource":
"datasources[\"dataSource\"][\"feed\"]",
      "filter": "pm1,pm2_5,pm10",
      "type": "line",
      "color": "",
      "marker": true,
      "multipleaxis": true,
      "height_block": "240",
      "height": "4"
     }
    }
   ]
  },
  {
   "width": 1,
   "row": {
    "3": 1
   },
   "col": {
    "3": 2
   },
   "col_width": 1,
   "widgets": [
    {
     "type": "gauge",
     "settings": {
      "title": "Pressure",
      "value":
"datasources[\"dataSource\"][\"shadow\"][\"pressure\"]",
      "units": "hPa",
      "min_value": 0,
      "max_value": "2000"
     }
    }
   ]
  },
  {
   "width": 1,
   "row": {
    "3": 1
   },
   "col": {
    "3": 3
   },
   "col_width": 1,
   "widgets": [
    {
     "type": "gauge",
     "settings": {
      "title": "Humidity",
      "value":
"datasources[\"dataSource\"][\"shadow\"][\"humidity\"]",
      "units": "%",
      "min_value": 0,
      "max_value": 100
     }
    }
   ]
  },
  {
   "width": 1,
   "row": {
    "3": 9
   },
   "col": {
    "3": 1
   },
   "col_width": "3",
   "widgets": [
    {
     "type": "nxpFeedView",
     "settings": {
      "title": "bmp280",
      "datasource":
"datasources[\"dataSource\"][\"feed\"]",
      "filter": "temperature,pressure,humidity",
      "type": "line",
      "color": "",
      "marker": true,
      "multipleaxis": true,
      "height_block": "240",
      "height": "4"
     }
    }
   ]
  }
 ],
 "datasources": [
  {
   "name": "dataSource",
   "type": "netpiex_datasource",
   "settings": {
    "name": "dataSource",
    "deviceid": "f216bf79-6d29-4b92-a13a-c8f0d2247531",
    "devicetoken": "RRf6HVX9pdYoRNrCK4ExT8vVngYLTsHt",
    "feed": true,
    "feed_since_value": "2",
    "feed_since_unit": "hours",
    "feed_downsampling": "1",
    "feed_downsampling_unit": "minutes"
   }
  }
 ],
 "columns": 3,
 "theme": "default"
}
```

File name : air_mon_bme280_netpie

Section 1: Include libraries and declare global variables.

```
#include <ArduinoJson.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Ticker.h>
#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>
#include <PubSubClient.h>
#include <EEPROM.h>
#include <SoftwareSerial.h>
#include "pms7003.h"

#define BUILD_VERSION 20221208  //YYYYMMDD Pongsathorn Utsahawattanasuk 6210554784

//------------------------------WiFi/MQTT setup------------------------------
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
char mqtt_client[40] = "";
char mqtt_username[40] = "";
char mqtt_password[40] = "";

WiFiClient espClient;
PubSubClient client(espClient);
bool shouldSaveConfig = false;
uint8_t macAddr[6];

//------------------------------BME280------------------------------
Adafruit_BME280 bme;  // I2C

//------------------------------Command------------------------------
String cmdstring;
String parmstring;
int sepIndex;
bool noparm = 0;
//------------------------------PMS7003------------------------------
SoftwareSerial _serial(D5, D6);  // RX, TX
pms7003 pms(_serial, Serial);
//------------------------------OLED------------------------------
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
```

```
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

//-----------------------------timer-----------------------------
uint8_t is_1s = 0, sleep = 0;
unsigned int count_1s = 0;
char msg[100];
unsigned int send_period = 60;
Ticker ticker;
//-----------------------------btn-----------------------------
#define TRIGGER_PIN 0  // NodeMCU FLASH button for start wifi config portal
unsigned int btn_push_sec = 0, btn_push_loop = 0;
#define WIFI_CFG_BTN_PUSH_SEC 5
uint8_t pressed = 0;
boolean buttonState = LOW;

//-----------------------------eeprom-----------------------------
union {
  struct {
    uint8_t d0 : 1;
    uint8_t d1 : 1;
    uint8_t d2 : 1;
    uint8_t d3 : 1;
    uint8_t d4 : 1;
    uint8_t d5 : 1;
    uint8_t d6 : 1;
    uint8_t d7 : 1;
  } bit;
  uint8_t b8;
} crc = { .b8 = 0 };
```

Section 2: Create the function that will be trigger every second using the <Ticker.h> that I set in the void setup() `ticker.attach(1, tick_1s);`

```
void tick_1s() {
  is_1s = 1;
  count_1s++;
  sleep++;
}
```

Section 3: Callback Function when wifi-configuration portal is called.

```
void configModeCallback(WiFiManager* myWiFiManager) {
  Serial.println("# Entered config mode");
  Serial.println(WiFi.softAPIP());
  //if you used auto generated SSID, print it
  Serial.println(myWiFiManager->getConfigPortalSSID());
}
```

Section 4: Initialization for every sensor, Wi-Fi, MQTT and load the data from EEPROM by `load_config();`.

```
void setup() {
  Serial.begin(38400);
  _serial.begin(9600);

  ticker.attach(1, tick_1s);

  unsigned status = bme.begin(0x76);
  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println("SSD1306 allocation failed");
  } else {
    Serial.println("ArdinoAll OLED Start Work !!!");
  }
  display.clearDisplay();
  display.setTextColor(WHITE, BLACK);
  display.setCursor(0, 0);
  display.println(("Starting"));
  display.display();

  pinMode(TRIGGER_PIN, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);

  //wifi
  WiFi.mode(WIFI_STA);
  WiFi.begin();
  delay(1000);
  WiFi.printDiag(Serial);

  Serial.print(F("# Wait for Wifi connected "));
  int retry = 40;
  while (WiFi.status() != WL_CONNECTED && retry > 0) {
    Serial.print('.');
    retry--;
    delay(500);
    if (digitalRead(TRIGGER_PIN) == LOW) return;
  }
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println(" connected");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("!! NOT connect");
  }
  //mqtt
  display.println(mqtt_server);
  display.print(F("Port:"));
  display.println(mqtt_port);
  display.display();
  load_config();
  Serial.print(F("# mqtt_server: "));
  Serial.println(mqtt_server);
  Serial.print(F("# mqtt_port: "));
  Serial.println(mqtt_port);
  Serial.print(F("# mqtt_client: "));
  Serial.println(mqtt_client);
  Serial.print(F("# mqtt_username: "));
  Serial.println(mqtt_username);
  Serial.print(F("# mqtt_password: "));
  Serial.println(mqtt_password);
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(mqtt_callback);
}
```

Section 5: The operation in loop contains

1. If pushed the button more than 5 sec will active the Wi-Fi portal mode.
2. Every 3 minutes the OLED display will turn off. If turn on the OLED all time can cause the permanent damage.
3. If pushed the button once the OLED will turn on again.
4. If pushed the button once the OLED display mode will change.
5. The sensor is read every 1 second for bme280 but for pms7003 all the time.
6. The Jason object will be creating every send_ period values and publish to the Netpie2020 at `"@shadow/data/update"` and send_ period values can be adjusted via freeboard.

```
void loop() {
  //Serial.print("btn_push_loop = "); Serial.println(btn_push_loop);
  if (btn_push_sec >= WIFI_CFG_BTN_PUSH_SEC) {
    wifiConfigDisplay();
    startWifiConfig();
    Serial.println("Start Wifi Config");
    btn_push_sec = 0;
  }
  if (digitalRead(TRIGGER_PIN) == LOW) {
    btn_push_loop += 1;
```

```
      //Serial.print("btn_push_loop = "); Serial.println(btn_push_loop);
      if (5 < btn_push_loop < 25) {
        display.ssd1306_command(SSD1306_DISPLAYON);
        sleep = 1;
      }
    } else {
      btn_push_loop = 0;
      btn_push_sec = 0;
    }

    if (digitalRead(TRIGGER_PIN) == LOW && buttonState == LOW) {
      //oledDisplay(pressed);
      pressed++;
      buttonState = HIGH;
      if (pressed >= 2) pressed = 0;
    } else if (digitalRead(TRIGGER_PIN) == HIGH && buttonState == HIGH) {
      buttonState = LOW;
    }
    if (is_1s) {
      if (!client.connected()) {
        reconnect();
      }
      client.loop();

      if (digitalRead(TRIGGER_PIN) == LOW) {
        btn_push_sec += 1;
      }
      //Serial.print("btn_push_sec = "); Serial.println(btn_push_sec);
      //printValues();
      float temp = bme.readTemperature();
      float hum = bme.readHumidity();
      float pres = bme.readPressure() / 100.0F;

      oledDisplay(temp, pres, hum, pressed);
      is_1s = 0;
      //Serial.printf("pms7003(ug/m3) :{pm1:%d ,pm2_5:%d ,pm10:%d}\n", _pm1,
_pm25, _pm10);

      if (sleep % send_period == 0) {  //every 1,5,10 min
        //----------------------------------------
        //idea: https://arduinojson.org/book/serialization_tutorial6.pdf#page=12
        StaticJsonDocument<200> doc;

        JsonObject data = doc.createNestedObject("data");
        data["pm1"] = pms.pm1Value;
        data["pm2_5"] = pms.pm25Value;
        data["pm10"] = pms.pm10Value;
        data["humidity"] = hum;
        data["pressure"] = pres;
        data["temperature"] = temp;

        char buffer[256];
        serializeJson(doc, buffer);
        client.publish("@shadow/data/update", buffer);
        Serial.print("publish: ");
        serializeJson(doc, Serial);
        Serial.println();
        //----------------------------------------
        //      String data = "{\"data\": {\"humidity\":" + String(hum) +
  ",\"temperature\":" + String(temp) + ",\"pressure\":" + String(pres) + "}}";
        //      Serial.print("publish : "); Serial.println(data);
        //      data.toCharArray(msg, (data.length() + 1));
        //      client.publish("@shadow/data/update", msg);
      }
      if (sleep % 180 == 0) {  //every 3 min
        display.ssd1306_command(SSD1306_DISPLAYOFF);
        //      client.subscribe("@shadow/data/updated");
      }
    }
    InputCommand();
    pms.readSensor();
    delay(5);
  }
```

Section 6: Function for receive the command through Serial port

Example:

1. client= f216bf79-6d29-4b92-a13a-c8f0d2247531
2. username= RRf6HVX9pdYoRNrCK4ExT8vVngYLTsHt
3. password= x4bpHacv73Y2R1q$0JcvwiuR#nxnT6Eq
4. saveConfig

```
void InputCommand() {
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    input.trim();
    sepIndex = input.indexOf('=');
    if (sepIndex == -1) {
      cmdstring = input;
      noparm = 1;
    } else {
      cmdstring = input.substring(0, sepIndex);
      cmdstring.trim();
      parmstring = input.substring(sepIndex + 1);
      parmstring.trim();
      noparm = 0;
    }
    if (cmdstring.equalsIgnoreCase("client")) {
      if (noparm == 0) {
        for (int i = 0; i < 40; i++) mqtt_client[i] = 0;
        parmstring.toCharArray(mqtt_client, parmstring.length() + 1);
        Serial.print("New MQTT Client ID = ");
        Serial.println(mqtt_client);
      }
    } else if (cmdstring.equalsIgnoreCase("username")) {
      if (noparm == 0) {
        for (int i = 0; i < 40; i++) mqtt_username[i] = 0;
        parmstring.toCharArray(mqtt_username, parmstring.length() + 1);
        Serial.print("New MQTT Username = ");
        Serial.println(mqtt_username);
      }
    } else if (cmdstring.equalsIgnoreCase("password")) {
      if (noparm == 0) {
        for (int i = 0; i < 40; i++) mqtt_password[i] = 0;
        parmstring.toCharArray(mqtt_password, parmstring.length() + 1);
        Serial.print("New MQTT Password = ");
        Serial.println(mqtt_password);
      }
    } else if (cmdstring.equalsIgnoreCase("saveConfig")) {
      save_config();
      Serial.println("Save config");
    }
  }
}
```

Section 7: Function of EEPROM that return the size.

```
int get_config_size() {
  int s = 0;
  s = sizeof(s);
  s += sizeof(mqtt_client);
  s += sizeof(mqtt_username);
  s += sizeof(mqtt_password);
  return s;
}
```

Section 8: Function of EEPROM that save the mqtt data.

```
void save_config() {
  int s, ss;
  byte csum;
  int eeprom_size;
  s = get_config_size();
  ss = s + sizeof(csum);
  if (ss % 4 == 0) {
    eeprom_size = ss;
  } else {
    eeprom_size = ss + (4 - (ss % 4));
  }
  EEPROM.begin(eeprom_size);
  EEPROM.put(0, s);
  s = sizeof(s);
  EEPROM.put(s, mqtt_client);
  s += sizeof(mqtt_client);
  EEPROM.put(s, mqtt_username);
  s += sizeof(mqtt_username);
  EEPROM.put(s, mqtt_password);
  s += sizeof(mqtt_password);
  uint8_t bi;
  uint8_t out1;
  crc.b8 = 0;
  for (int i = 0, csum = 0; i < s; i++) {
    bi = EEPROM[i];
    for (int b = 0; b < 8; b++) {
      out1 = crc.bit.d7 ^ (bi & 0x01);
      crc.bit.d4 = crc.bit.d7 ^ crc.bit.d4;
      crc.b8 = crc.b8 << 1;
      crc.bit.d0 = out1;
      bi << 1;
    }
  }
  csum = crc.b8;
  EEPROM.put(s, csum);
  if (EEPROM.commit()) {
    Serial.printf("# EEPROM successfully committed. size=%d, csum=%02X\n", s,
csum);
  } else {
    Serial.println(F("# ERROR! EEPROM commit failed"));
  }
  EEPROM.end();
}
```

Section 9: Function of EEPROM that load the data of mqtt that was saved.

```
int load_config() {
  int s, ss, eeprom_size;
  byte csum, csum2;
  int ee_conf_size;
  s = get_config_size();
  ss = s + sizeof(csum);
  if (ss % 4 == 0) {
    eeprom_size = ss;
  } else {
    eeprom_size = ss + (4 - (ss % 4));
  }
  EEPROM.begin(eeprom_size);
  EEPROM.get(0, ee_conf_size);  //get size of config from eeprom
  if (ee_conf_size > s) {
    Serial.print(F("# Warning configuration size missmatch "));
    Serial.print(s);
    Serial.print("!=");
    Serial.println(ee_conf_size);
    return -1;
  }
  uint8_t bi;
  uint8_t out1;
  crc.b8 = 0;
  for (int i = 0, csum = 0; i < ee_conf_size; i++) {
    bi = EEPROM[i];
    for (int b = 0; b < 8; b++) {
      out1 = crc.bit.d7 ^ (bi & 0x01);
      crc.bit.d4 = crc.bit.d7 ^ crc.bit.d4;
      crc.b8 = crc.b8 << 1;
      crc.bit.d0 = out1;
      bi << 1;
    }
  }
  csum = crc.b8;
  EEPROM.get(ee_conf_size, csum2);
  if (csum != csum2) {
    Serial.print(F("# configuration checksum missmatch "));
    Serial.print(csum, HEX);
    Serial.print("!=");
    Serial.println(csum2, HEX);
    return -2;
  }
  Serial.print(F("# load EEPROM config ... "));
  do {
    s = sizeof(s);
    EEPROM.get(s, mqtt_client);
    s += sizeof(mqtt_client);
    EEPROM.get(s, mqtt_username);
    s += sizeof(mqtt_username);
    EEPROM.get(s, mqtt_password);
    s += sizeof(mqtt_password);
    if (s >= ee_conf_size) break;
  } while (false);
  Serial.println(F("Done"));
  EEPROM.end();
  return s;
}
```

Section 10: The function that checks if the MQTT is connected or not If not, it will be struck in this loop, and you can enter the MQTT data through the serial port.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting NETPIE2020 connection…");
    if (client.connect(mqtt_client, mqtt_username, mqtt_password)) {
      Serial.println("NETPIE2020 connected");
      client.subscribe("@msg/#");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
    InputCommand();
  }
}
```

Section 11: Function for the callback of MQTT. The data from the topic that received from the Netpie freeboard.

```
void mqtt_callback(char* topic, byte* payload, unsigned int length) {
  Serial.print(F("# Message arrived ["));
  Serial.print(topic);
  Serial.print("] ");
  String message;
  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }
  Serial.println(message);

  if (String(topic) == "@msg/led") {
    if (message == "on") {
      digitalWrite(LED_BUILTIN, 0);
      client.publish("@shadow/data/update", "{\"data\":{\"led\":\"on\"}}");
      Serial.println("LED ON");
    } else if (message == "off") {
      digitalWrite(LED_BUILTIN, 1);
      client.publish("@shadow/data/update", "{\"data\":{\"led\":\"off\"}}");
      Serial.println("LED OFF");
    }
  } else if (String(topic) == "@msg/reset") {
    if (message == "True") ESP.reset();
  } else if (String(topic) == "@msg/sendPeriod") {
    if (message == "1") {
      send_period = 60 * 1;
      client.publish("@shadow/data/update",
"{\"data\":{\"sendPeriod\":\"1\"}}");
      Serial.println("Send data period is set to 1 minute");
    } else if (message == "5") {
      send_period = 60 * 5;
      client.publish("@shadow/data/update",
"{\"data\":{\"sendPeriod\":\"5\"}}");
      Serial.println("Send data period is set to 5 minutes");
    } else if (message == "10") {
      send_period = 60 * 10;
      client.publish("@shadow/data/update",
"{\"data\":{\"sendPeriod\":\"10\"}}");
      Serial.println("Send data period is set to 10 minutes");
    }
  }
}
```

Section 12: OLED function that will active when enter the Wi-Fi portal mode.

```
void wifiConfigDisplay() {
  display.clearDisplay();
  display.setTextColor(WHITE, BLACK);
  display.setCursor(0, 10);
  display.setTextSize(2);
  display.println(" WIFI");
  display.println(" CONFIG...");
  display.drawRect(0, 0, 120, 50, WHITE);
  display.display();
  display.startscrollright(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrolldiagright(0x00, 0x07);
  delay(2000);
  display.startscrolldiagleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
}
```

Section 13: OLED function that contain 2 modes. First is for bme280 and second is for pms7003.

```
void oledDisplay(float temp, float pres, float hum, uint8_t oledMode) {
  display.clearDisplay();
  display.setTextColor(WHITE, BLACK);
  display.setCursor(0, 0);
  display.setTextSize(1);
  char wifi_status, mqtt_status;
  WiFi.status() == WL_CONNECTED ? wifi_status = 'Y' : wifi_status = 'N';
  client.connected() ? mqtt_status = 'Y' : mqtt_status = 'N';

  display.clearDisplay();
  display.setTextColor(WHITE, BLACK);
  display.setCursor(0, 0);
  display.setTextSize(1);
  switch (oledMode) {
    case 0:
      display.print("BME280");
      display.print(" WIFI:");
      display.print(wifi_status);
      display.print(" MQTT:");
      display.println(mqtt_status);
      display.println();
      display.print("Temp: ");
      display.setTextSize(2);
      display.print(temp);
      display.cp437(true);
      display.write(167);
```

```
        display.println("C");
        display.setTextSize(1);
        display.print("Pres: ");
        display.setTextSize(2);
        display.print(pres);
        display.println();
        display.setTextSize(1);
        display.print("Humi: ");
        display.setTextSize(2);
        display.print(hum);
        display.println(" %");
        display.setCursor(0, 40);
        display.setTextSize(1);
        display.print("(hPa)");
        display.display();
        break;
      case 1:
        display.cp437(true);
        display.print("PMS7003");
        display.print(" WIFI:");
        display.print(wifi_status);
        display.print(" MQTT:");
        display.println(mqtt_status);
        display.println();
        display.print("PM1:  ");
        display.setTextSize(2);
        display.print(pms.pm1Value);
        display.write(230);
        display.println("g/m3");
        display.setTextSize(1);
        display.print("PM2.5:");
        display.setTextSize(2);
        display.print(pms.pm25Value);
        display.write(230);
        display.println("g/m3");
        display.setTextSize(1);
        display.print("PM10: ");
        display.setTextSize(2);
        display.print(pms.pm10Value);
        display.write(230);
        display.println("g/m3");
        display.display();
        break;
    }
  }
```

Section 14: After the Wi-Fi portal is activated. You can enter the data through wifi connector or 192.168.4.1 and you can add more parameter to received from this portal such as mqtt data.

```
void startWifiConfig() {
  //WiFiManager
  //Local intialization. Once its business is done, there is no need to keep
it around
  WiFiManager wifiManager;
  WiFiManagerParameter custom_mqtt_client("client", "MQTT client",
mqtt_client, 40);
  WiFiManagerParameter custom_mqtt_mqtt_username("Username", "Username",
mqtt_username, 40);
  WiFiManagerParameter custom_mqtt_password("Password", "Password",
mqtt_password, 40);

  wifiManager.setTimeout(300);
  wifiManager.setAPCallback(configModeCallback);
  wifiManager.setSaveConfigCallback(saveConfigCallback);
  wifiManager.setShowInfoUpdate(false);
  wifiManager.addParameter(&custom_mqtt_client);
  wifiManager.addParameter(&custom_mqtt_mqtt_username);
  wifiManager.addParameter(&custom_mqtt_password);
  wifiManager.setBreakAfterConfig(true);
  wifiManager.setConnectTimeout(30);

  if (!wifiManager.startConfigPortal()) {
    Serial.println(F("# failed to connect or hit timeout"));
  }
}
```

Section 15: This function will be active when you click save in Wi-Fi portal.

```
Void saveConfigCallback() {
  Serial.println(“# Should save config”);
  shouldSaveConfig = true;
}
```

For pms7003.h and pms7003.cpp, In esp8266 there doesn’t have the library that support for pms7003. So, I get the reference from this website: **https://hpclab.blogspot.com/2020/02/esp8266-based-air-quality-monitoring.html** then i convert the read sensor function into cpp object-oriented program.

pms7003.h

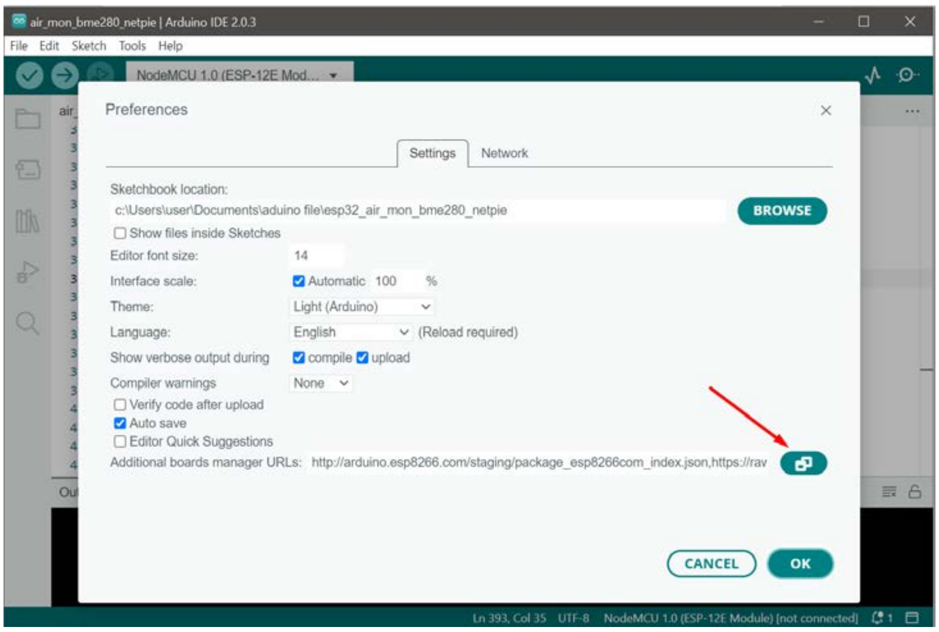
```
 //created by pongsathorn utsahawattanasuk 6210554784

#pragma once
#include "Arduino.h"
#define PMS7003_PREAMBLE_1 0x42  // From PMS7003 datasheet
#define PMS7003_PREAMBLE_2 0x4D
#define PMS7003_DATA_LENGTH 31

class pms7003 {
public:
  pms7003(Stream &serial);
  pms7003(Stream &serial, Stream &debug);
  int pm1Value, pm25Value, pm10Value;
  void readSensor();
private:
  int checksum;
  unsigned char pms[32] = {
    0,
  };
  Stream *_serial;
  Stream *_debug;
};
```

pms7003.cpp

```
#include "pms7003.h"

// Constructor
pms7003::pms7003(Stream &serial) {
  _serial = &serial;
}

pms7003::pms7003(Stream &serial, Stream &debug) {
  _serial = &serial;
  _debug = &debug;
}

void pms7003::readSensor() {
  checksum = 0;
  /**
     Search preamble for Packet
     Solve trouble caused by delay function
  */
  while (_serial->available() && _serial->read() != PMS7003_PREAMBLE_1 &&
_serial->peek() != PMS7003_PREAMBLE_2) {
  }
  if (_serial->available() >= PMS7003_DATA_LENGTH) {
    pms[0] = PMS7003_PREAMBLE_1;
    checksum += pms[0];
    for (int j = 1; j < 32; j++) {
      pms[j] = _serial->read();
      if (j < 30)
        checksum += pms[j];
    }
    _serial->flush();
    if (pms[30] != (unsigned char)(checksum >> 8)
        || pms[31] != (unsigned char)(checksum)) {
      _debug->println("Checksum error");
      return;
    }
    if (pms[0] != 0x42 || pms[1] != 0x4d) {
      _debug->println("Packet error");
      return;
    }
    pm1Value = makeWord(pms[10], pms[11]);
    pm25Value = makeWord(pms[12], pms[13]);
    pm10Value = makeWord(pms[14], pms[15]);
  }
}
```

Prepare the Arduino IDE with esp32

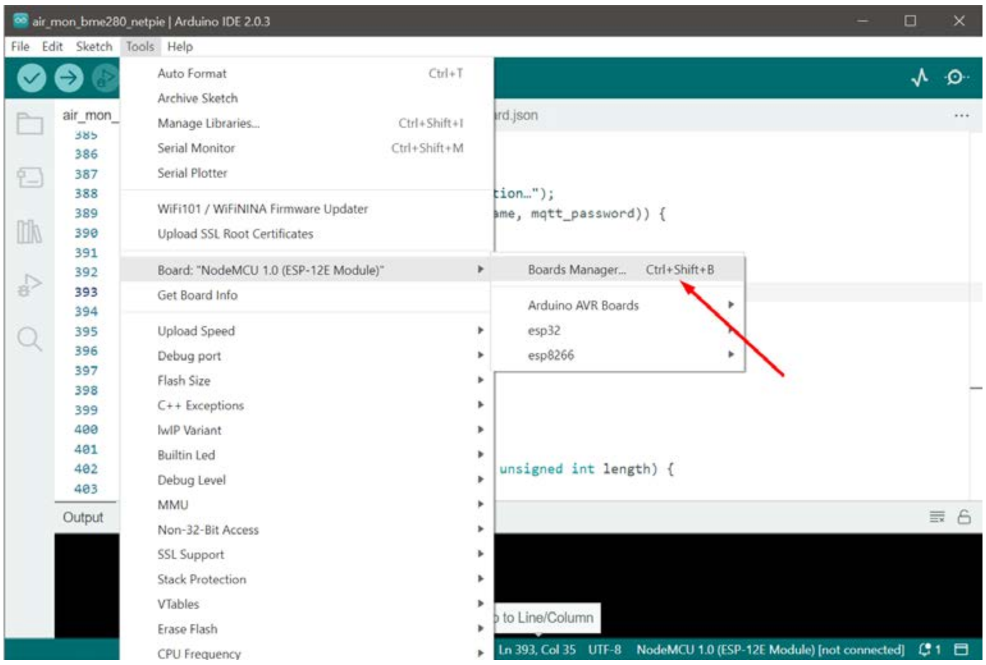
1. Download and install the Arduino IDE on your operating system.
2. Install the ESP32 add-on for the Arduino IDE. Go to **file** then **Preference**
3. Enter the URL in the Additional Board Manager field:
   `https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json`
4. Then click on OK button


air_mon_bme280_netpie | Arduino IDE 2.0.3
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12E Mod...
Preferences
Settings
Network
BROWSE
CANCEL
OK

Sketchbook location:
c:\Users\user\Documents\aduino file\esp32_air_mon_bme280_netpie
☐ Show files inside Sketches
Editor font size: 14
Interface scale: ☑ Automatic 100 %
Theme: Light (Arduino)
Language: English (Reload required)
Show verbose output during ☑ compile ☑ upload
Compiler warnings None
☐ Verify code after upload
☑ Auto save
☐ Editor Quick Suggestions
Additional boards manager URLs: http://arduino.esp8266.com/staging/package_esp8266com_index.json,https://rav

Ln 393, Col 35 UTF-8 NodeMCU 1.0 (ESP-12E Module) [not connected]

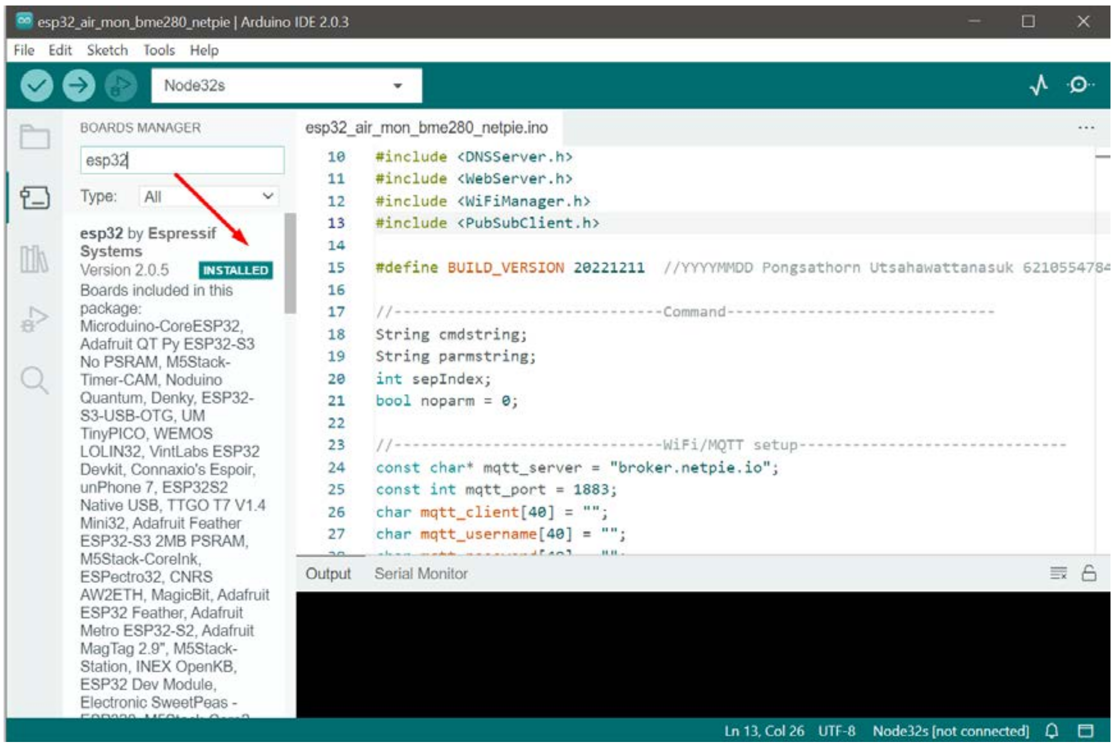
5. Go to **Tools** then **Board** then **Board Manager**


air_mon_bme280_netpie | Arduino IDE 2.0.3
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Board: "NodeMCU 1.0 (ESP-12E Module)"
Get Board Info
Upload Speed
Debug port
Flash Size
C++ Exceptions
IwIP Variant
Builtin Led
Debug Level
MMU
Non-32-Bit Access
SSL Support
Stack Protection
VTables
Erase Flash
CPU Frequency
Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
esp32
esp8266
Output

```
tion...");
ame, mqtt_password)) {
unsigned int length) {
```

Ln 393, Col 35 UTF-8 NodeMCU 1.0 (ESP-12E Module) [not connected]

6. Type esp32then install the board.



esp32_air_mon_bme280_netpie | Arduino IDE 2.0.3

File Edit Sketch Tools Help

Node32s

BOARDS MANAGER

esp32

Type: All

esp32 by Espressif Systems
Version 2.0.5 INSTALLED
Boards included in this package:
Microduino-CoreESP32, Adafruit QT Py ESP32-S3 No PSRAM, M5Stack-Timer-CAM, Noduino Quantum, Denky, ESP32-S3-USB-OTG, UM TinyPICO, WEMOS LOLIN32, VintLabs ESP32 Devkit, Connaxio's Espoir, unPhone 7, ESP32S2 Native USB, TTGO T7 V1.4 Mini32, Adafruit Feather ESP32-S3 2MB PSRAM, M5Stack-CoreInk, ESPectro32, CNRS AW2ETH, MagicBit, Adafruit ESP32 Feather, Adafruit Metro ESP32-S2, Adafruit MagTag 2.9", M5Stack-Station, INEX OpenKB, ESP32 Dev Module, Electronic SweetPeas -

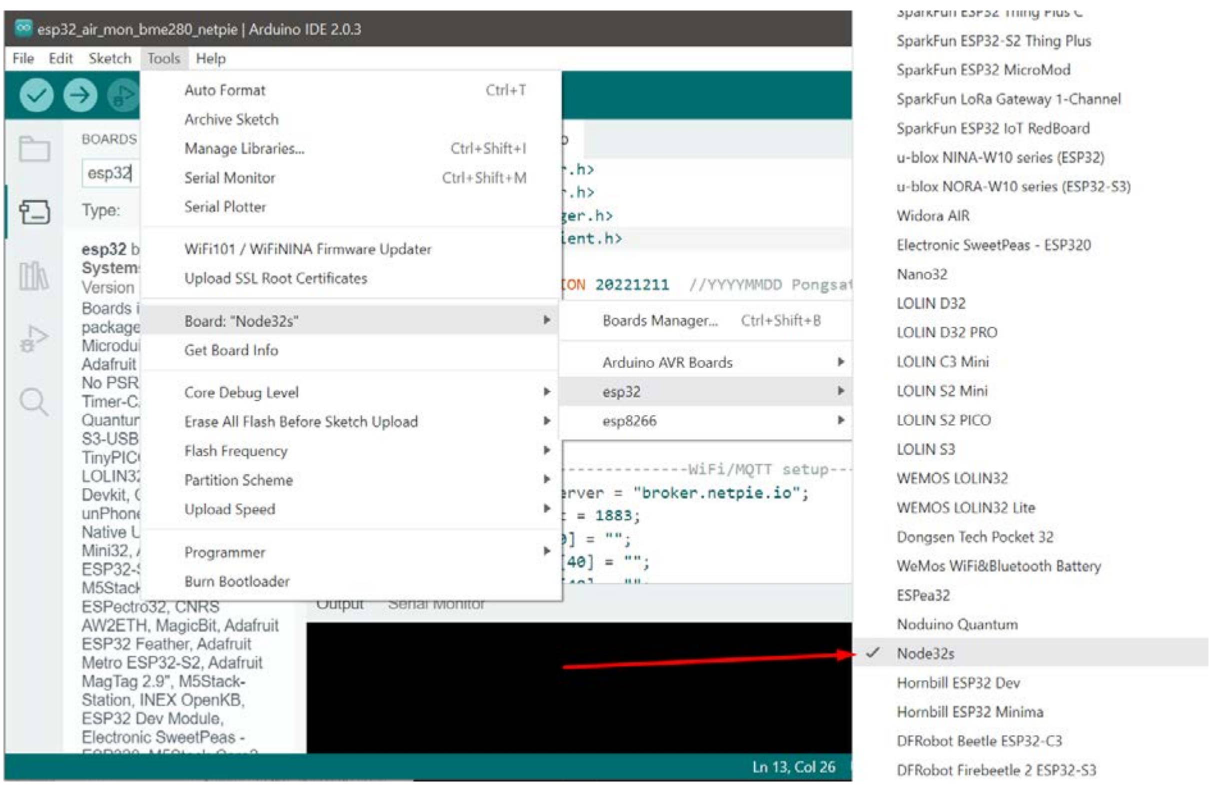
esp32_air_mon_bme280_netpie.ino

```
#include <DNSServer.h>
#include <WebServer.h>
#include <WiFiManager.h>
#include <PubSubClient.h>

#define BUILD_VERSION 20221211   //YYYYMMDD Pongsathorn Utsahawattanasuk 621055478

//----------------------------Command----------------------------
String cmdstring;
String parmstring;
int sepIndex;
bool noparm = 0;

//----------------------------WiFi/MQTT setup----------------------------
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
char mqtt_client[40] = "";
char mqtt_username[40] = "";
```

Output Serial Monitor

Ln 13, Col 26 UTF-8 Node32s [not connected]

7. In this case I'm working with NodeMCU then I choose Node32s in esp32



esp32_air_mon_bme280_netpie | Arduino IDE 2.0.3

File Edit Sketch Tools Help

Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Board: "Node32s"
Get Board Info
Core Debug Level
Erase All Flash Before Sketch Upload
Flash Frequency
Partition Scheme
Upload Speed
Programmer
Burn Bootloader

Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
esp32
esp8266

SparkFun ESP32 Thing Plus C
SparkFun ESP32-S2 Thing Plus
SparkFun ESP32 MicroMod
SparkFun LoRa Gateway 1-Channel
SparkFun ESP32 IoT RedBoard
u-blox NINA-W10 series (ESP32)
u-blox NORA-W10 series (ESP32-S3)
Widora AIR
Electronic SweetPeas - ESP320
Nano32
LOLIN D32
LOLIN D32 PRO
LOLIN C3 Mini
LOLIN S2 Mini
LOLIN S2 PICO
LOLIN S3
WEMOS LOLIN32
WEMOS LOLIN32 Lite
Dongsen Tech Pocket 32
WeMos WiFi&Bluetooth Battery
ESPea32
Noduino Quantum
✓ Node32s
Hornbill ESP32 Dev
Hornbill ESP32 Minima
DFRobot Beetle ESP32-C3
DFRobot Firebeetle 2 ESP32-S3

Ln 13, Col 26

Integrated the code to ESP32

because I want to add the Multitasking and use other method of EEPROM and In esp32 there is a library for pms7003 using PMS.h

```
#include <PMS.h>
#include <ArduinoJson.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Ticker.h>
#include <EEPROM.h>
#include <WiFi.h>
#include <DNSServer.h>
#include <WebServer.h>
#include <WiFiManager.h>
#include <PubSubClient.h>

#define BUILD_VERSION 20221211  //YYYYMMDD Pongsathorn Utsahawattanasuk
6210554784

//-----------------------------Command-----------------------------
String cmdstring;
String parmstring;
int sepIndex;
bool noparm = 0;

//-----------------------------WiFi/MQTT setup-----------------------------
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
char mqtt_client[40] = "";
char mqtt_username[40] = "";
char mqtt_password[40] = "";

WiFiClient espClient;
PubSubClient client(espClient);
bool shouldSaveConfig = false;
uint8_t macAddr[6];
//-----------------------------BME280-----------------------------
Adafruit_BME280 bme;
float temp, pres, hum;
//-----------------------------OLED-----------------------------
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

//------------------------------eeprom------------------------------
int eeinitflag = 0;  // EEPROM initialization flag
const int EEINITCODE = 7;
#define EE_INIT 200  // write EEINITCODE this address if EEPROM is initialized
//------------------------------EEPROM addresses -----------------------------
-
#define EE_MQTTCID_ADDR 0
#define EE_MQTTUSER_ADDR 40
#define EE_MQTTPWD_ADDR 80
bool eesaveflag = 0;

//------------------------------pms------------------------------
PMS pms(Serial1);  //GPIO1,GPIO3
PMS::DATA data;
uint8_t pm1, pm2_5, pm10;
//------------------------------timer------------------------------
uint8_t is_1s = 0, timeOut = 0;
unsigned int count_1s = 0;
char msg[100];
unsigned int send_period = 60;
Ticker ticker;
//------------------------------btn------------------------------
#define TRIGGER_PIN 0  // NodeMCU FLASH button for start wifi config portal
unsigned int btn_push_sec = 0, btn_push_loop = 0;
#define WIFI_CFG_BTN_PUSH_SEC 5
uint8_t pressed = 0;
boolean buttonState = LOW;
#define LED 2

void tick_1s() {
  is_1s = 1;
  count_1s++;
  timeOut++;
}

void configModeCallback(WiFiManager* myWiFiManager) {
  Serial.println("# Entered config mode");
  Serial.println(WiFi.softAPIP());
  //if you used auto generated SSID, print it
  Serial.println(myWiFiManager->getConfigPortalSSID());
}

//------------------------------FreeRtos------------------------------

TaskHandle_t Task0 = NULL;
```

```
void Task0_code(void* parameter) {  //core 0
  for (;;) {
    temp = bme.readTemperature();
    hum = bme.readHumidity();
    pres = bme.readPressure() / 100.0F;
    vTaskDelay(10);
  }
  vTaskDelete(NULL);
}

TaskHandle_t Task1 = NULL;

void Task1_code(void* parameter) {  //core 0
  for (;;) {
    pms.read(data);
    pm1 = data.PM_AE_UG_1_0;
    pm2_5 = data.PM_AE_UG_2_5;
    pm10 = data.PM_AE_UG_10_0;
    vTaskDelay(5);
  }
  vTaskDelete(NULL);
}

void freertos_init(void) {
  xTaskCreatePinnedToCore(Task0_code, "bme280", 10000, NULL, 3, &Task0, 1);
  xTaskCreatePinnedToCore(Task1_code, "pms7003", 10000, NULL, 2, &Task1, 1);
  xTaskCreatePinnedToCore(Task2_code, "oled", 10000, NULL, 1, &Task2, 1);
}

void setup() {
  Serial.begin(9600);   // for debug GPIO1,3
  Serial1.begin(9600);  //for pms GPIO2 or D4
  //EEPROM
  EEPROM.begin(256);
  eeinitflag = EEPROM.read(EE_INIT);
  if (eeinitflag == EEINITCODE) {
    Serial.println("Reading parameters from EEPROM");
    EEPROM_getparms();  // get parameters from EEPROM
  }

  ticker.attach(1, tick_1s);
  Serial.println("Strarting...");
  unsigned status = bme.begin(0x76);
  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println("SSD1306 allocation failed");
  } else {
    Serial.println("ArdinoAll OLED Start Work !!!");
  }
```

```
display.clearDisplay();
display.setTextColor(WHITE, BLACK);
display.setCursor(0, 0);
display.println(("Starting"));
display.display();

pinMode(TRIGGER_PIN, INPUT);
pinMode(LED, OUTPUT);

//wifi
WiFi.mode(WIFI_STA);
WiFi.begin();
delay(1000);
WiFi.printDiag(Serial);

Serial.print(F("# Wait for Wifi connected "));
int retry = 40;
while (WiFi.status() != WL_CONNECTED && retry > 0) {
  Serial.print('.');
  retry--;
  delay(500);
  if (digitalRead(TRIGGER_PIN) == LOW) return;
}
if (WiFi.status() == WL_CONNECTED) {
  Serial.println(" connected");
  Serial.println(WiFi.localIP());
} else {
  Serial.println("!! NOT connect");
}
//mqtt
display.println(mqtt_server);
display.print(F("Port:"));
display.println(mqtt_port);
display.display();
Serial.print(F("# mqtt_server: "));
Serial.println(mqtt_server);
Serial.print(F("# mqtt_port: "));
Serial.println(mqtt_port);
Serial.print(F("# mqtt_client: "));
Serial.println(mqtt_client);
Serial.print(F("# mqtt_username: "));
Serial.println(mqtt_username);
Serial.print(F("# mqtt_password: "));
Serial.println(mqtt_password);
client.setServer(mqtt_server, mqtt_port);
client.setCallback(mqtt_callback);

freertos_init();
```

```
}
// core1
void loop() {
  if (is_1s) {
    if (!client.connected()) {
      reconnect();
    }
    client.loop();
    is_1s = 0;
    if (timeOut % send_period == 0) {  //every 1,5,10 min
      //-----------------------------------------
      //idea: https://arduinojson.org/book/serialization_tutorial6.pdf#page=12
      StaticJsonDocument<200> doc;

      JsonObject data = doc.createNestedObject("data");

      data["pm1"] = pm1;
      data["pm2_5"] = pm2_5;
      data["pm10"] = pm10;

      data["humidity"] = hum;
      data["pressure"] = pres;
      data["temperature"] = temp;

      char buffer[256];
      serializeJson(doc, buffer);
      client.publish("@shadow/data/update", buffer);
      Serial.print("publish: ");
      serializeJson(doc, Serial);
      Serial.println();
    }
  }
  InputCommand();
  delay(5);
}

void InputCommand() {
  eesaveflag = 0;
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    if (input.equalsIgnoreCase("wifiPortal")) {
      startWifiConfig();
    }
    input.trim();
    sepIndex = input.indexOf('=');
    if (sepIndex == -1) {
      cmdstring = input;
      noparm = 1;
    } else {
      cmdstring = input.substring(0, sepIndex);
      cmdstring.trim();
      parmstring = input.substring(sepIndex + 1);
      parmstring.trim();
      noparm = 0;
    }
    if (cmdstring.equalsIgnoreCase("client")) {
      if (noparm == 0) {
        for (int i = 0; i < 40; i++) mqtt_client[i] = 0;
        parmstring.toCharArray(mqtt_client, parmstring.length() + 1);
        Serial.print("New MQTT Client ID = ");
        Serial.println(mqtt_client);
      }
    } else if (cmdstring.equalsIgnoreCase("username")) {
      if (noparm == 0) {
        for (int i = 0; i < 40; i++) mqtt_username[i] = 0;
        parmstring.toCharArray(mqtt_username, parmstring.length() + 1);
        Serial.print("New MQTT Username = ");
        Serial.println(mqtt_username);
      }
    } else if (cmdstring.equalsIgnoreCase("password")) {
      if (noparm == 0) {
        for (int i = 0; i < 40; i++) mqtt_password[i] = 0;
        parmstring.toCharArray(mqtt_password, parmstring.length() + 1);
        Serial.print("New MQTT Password = ");
        Serial.println(mqtt_password);
      }
    } else if (cmdstring.equalsIgnoreCase("saveConfig")) {
      EEPROM_saveparms();
      Serial.println("Save config");
    }
  }
}

// save parameters to EEPROM
void EEPROM_saveparms(void) {
  EEPROM.put(EE_MQTTCID_ADDR, mqtt_client);
  EEPROM.put(EE_MQTTUSER_ADDR, mqtt_username);
  EEPROM.put(EE_MQTTPWD_ADDR, mqtt_password);

  if (eeinitflag != EEINITCODE) EEPROM.write(EE_INIT, EEINITCODE);
  delay(10);
  EEPROM.commit();
  EEPROM.end();
  Serial.println("Parameters saved to EEPROM.");
  Serial.println("Press reset button.");
  eesaveflag = 1;
```

```
}

void EEPROM_getparms(void) {
  EEPROM.get(EE_MQTTCID_ADDR, mqtt_client);
  EEPROM.get(EE_MQTTUSER_ADDR, mqtt_username);
  EEPROM.get(EE_MQTTPWD_ADDR, mqtt_password);
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting NETPIE2020 connection…");
    if (client.connect(mqtt_client, mqtt_username, mqtt_password)) {
      Serial.println("NETPIE2020 connected");
      client.subscribe("@msg/#");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("try again in 5 seconds");
      delay(5000);
    }
    InputCommand();
  }
}

void mqtt_callback(char* topic, byte* payload, unsigned int length) {
  Serial.print(F("# Message arrived ["));
  Serial.print(topic);
  Serial.print("] ");
  String message;
  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }
  Serial.println(message);

  if (String(topic) == "@msg/led") {
    if (message == "on") {
      digitalWrite(LED, 0);
      client.publish("@shadow/data/update", "{\"data\":{\"led\":\"on\"}}");
      Serial.println("LED ON");
    } else if (message == "off") {
      digitalWrite(LED, 1);
      client.publish("@shadow/data/update", "{\"data\":{\"led\":\"off\"}}");
      Serial.println("LED OFF");
    }
  } else if (String(topic) == "@msg/reset") {
    if (message == "True") ESP.restart();
  } else if (String(topic) == "@msg/sendPeriod") {
    if (message == "1") {
      send_period = 60 * 1;
      client.publish("@shadow/data/update",
"{\"data\":{\"sendPeriod\":\"1\"}}");
      Serial.println("Send data period is set to 1 minute");
    } else if (message == "5") {
      send_period = 60 * 5;
      client.publish("@shadow/data/update",
"{\"data\":{\"sendPeriod\":\"5\"}}");
      Serial.println("Send data period is set to 5 minutes");
    } else if (message == "10") {
      send_period = 60 * 10;
      client.publish("@shadow/data/update",
"{\"data\":{\"sendPeriod\":\"10\"}}");
      Serial.println("Send data period is set to 10 minutes");
    }
  }
}
void wifiConfigDisplay() {
  display.clearDisplay();
  display.setTextColor(WHITE, BLACK);
  display.setCursor(0, 10);
  display.setTextSize(2);
  display.println(" WIFI");
  display.println(" CONFIG...");
  display.drawRect(0, 0, 120, 50, WHITE);
  display.display();
  display.startscrollright(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrolldiagright(0x00, 0x07);
  delay(2000);
  display.startscrolldiagleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
}

void oledDisplay(float temp, float pres, float hum, uint8_t oledMode) {
  display.clearDisplay();
  display.setTextColor(WHITE, BLACK);
  display.setCursor(0, 0);
  display.setTextSize(1);
  char wifi_status, mqtt_status;
  WiFi.status() == WL_CONNECTED ? wifi_status = 'Y' : wifi_status = 'N';
```

```
client.connected() ? mqtt_status = 'Y' : mqtt_status = 'N';

display.clearDisplay();
display.setTextColor(WHITE, BLACK);
display.setCursor(0, 0);
display.setTextSize(1);
switch (oledMode) {
  case 0:
    display.print("BME280");
    display.print(" WIFI:");
    display.print(wifi_status);
    display.print(" MQTT:");
    display.println(mqtt_status);
    display.println();
    display.print("Temp: ");
    display.setTextSize(2);
    display.print(temp);
    display.cp437(true);
    display.write(167);
    display.println("C");
    display.setTextSize(1);
    display.print("Pres: ");
    display.setTextSize(2);
    display.print(pres);
    display.println();
    display.setTextSize(1);
    display.print("Humi: ");
    display.setTextSize(2);
    display.print(hum);
    display.println(" %");
    display.setCursor(0, 40);
    display.setTextSize(1);
    display.print("(hPa)");
    display.display();
    break;
  case 1:
    display.cp437(true);
    display.print("PMS7003");
    display.print(" WIFI:");
    display.print(wifi_status);
    display.print(" MQTT:");
    display.println(mqtt_status);
    display.println();
    display.print("PM1:  ");
    display.setTextSize(2);
    display.print(data.PM_AE_UG_1_0);
    display.write(230);
    display.println("g/m3");
```

```
      display.setTextSize(1);
      display.print("PM2.5:");
      display.setTextSize(2);
      display.print(data.PM_AE_UG_2_5);
      display.write(230);
      display.println("g/m3");
      display.setTextSize(1);
      display.print("PM10: ");
      display.setTextSize(2);
      display.print(data.PM_AE_UG_10_0);
      display.write(230);
      display.println("g/m3");
      display.display();
      break;
  }
}

void startWifiConfig() {
  //WiFiManager
  //Local intialization. Once its business is done, there is no need to keep
it around
  WiFiManager wifiManager;
  //  WiFiManagerParameter custom_mqtt_client("client", "MQTT client",
mqtt_client, 40);
  //  WiFiManagerParameter custom_mqtt_mqtt_username("Username", "Username",
mqtt_username, 40);
  //  WiFiManagerParameter custom_mqtt_password("Password", "Password",
mqtt_password, 40);

  wifiManager.setTimeout(300);
  wifiManager.setAPCallback(configModeCallback);
  wifiManager.setSaveConfigCallback(saveConfigCallback);
  wifiManager.setShowInfoUpdate(false);
  //  wifiManager.addParameter(&custom_mqtt_client);
  //  wifiManager.addParameter(&custom_mqtt_mqtt_username);
  //  wifiManager.addParameter(&custom_mqtt_password);
  wifiManager.setBreakAfterConfig(true);
  wifiManager.setConnectTimeout(30);

  if (!wifiManager.startConfigPortal()) {
    Serial.println(F("# failed to connect or hit timeout"));
  }
}

void saveConfigCallback() {
  Serial.println("# Should save config");
  shouldSaveConfig = true;
}
```

Conclusion

I began the project with the esp8266 because I already had one, but I soon discovered that it couldn't multitask and that some libraries were unavailable. So, I tried to convert my code into esp32 code and then start working with multitasking. but because the PMS (dust sensor) doesn’t come along with the esp8266. So, I had a chance to work with a header and a C++ file.

I also tried to work with the FFT with the sound sensor because now that esp32 has two cores, the advance computation that consumes a lot of time will be moved to core 0. But, unfortunately, the sound sensor that I got, ky037, seems to not work well. I tried to adjust the potentiometer, but with the sound detection, it doesn’t make much difference in terms of signal amplitude.

Lastly, you can see that the EEPROM that I use in the esp8266 and esp32 is different. Firstly, I use the EEPROM code from my other work with esp8266, but after I use the same code with esp32 there is the problem with the data type. So, I use the same method as my teacher did in his code.

References

1. B. T.S., “ESP8266-based WIFI Air Quality Monitoring System using PMS7003 sensor,” *ESP8266-based WiFi air quality monitoring system using PMS7003 sensor*. [Online]. Available: https://hpclab.blogspot.com/2020/02/esp8266-based-wifi-air-quality.html. [Accessed: 11-Dec-2022].

2. BenoitBlanchon, “Efficient JSON serialization for embedded C++,” *ArduinoJson*. [Online]. Available: https://arduinojson.org/. [Accessed: 11-Dec-2022].