

2102447 ปฏิบัติการวิศวกรรมอิเล็กทรอนิกส์
Electronic Engineering Laboratory

ผู้สอนประจำวิชา ผศ.ดร.สุรีย์ พุ่มรินทร์ และ อ.ดร.ณรงค์ ปันธนาธรรม
ผู้สอนปฏิบัติการ ณัทกร เกษมลาราม (ภาคการศึกษาต้น 2567)

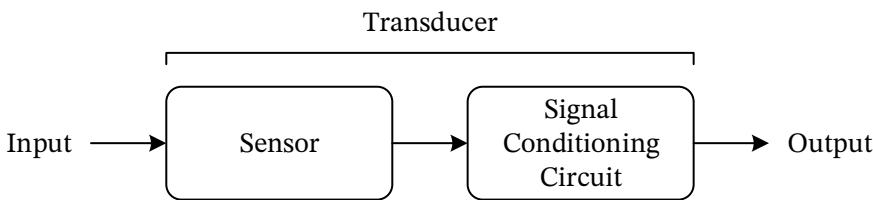


ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
ก่อตั้ง ๒๔๗๗
จ. พัฒกรน์มหาวิทยาลัย

อินเทอร์เน็ตของสรรพสิ่งส่วนฮาร์ดแวร์แพลตฟอร์ม
(Hardware Platform for Internet of Things)

1. บทนำ

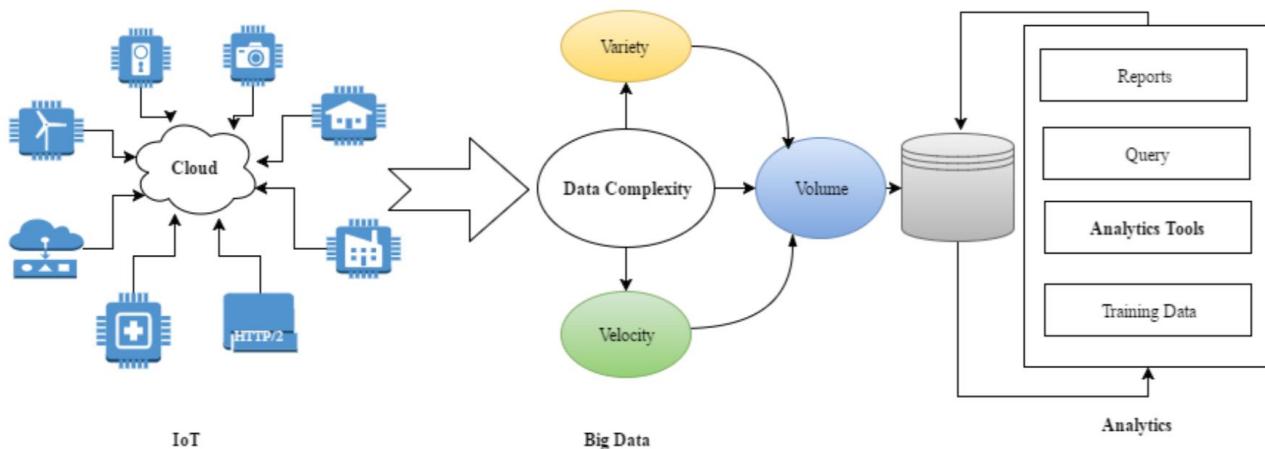
ตัวรับรู้ (Sensor) และตัวเปลี่ยนแปลง (Transducer) ทำหน้าที่แปลงพลังงานที่เกิดจากปรากฏการณ์ทางฟิสิกส์ไปเป็นพลังงานไฟฟ้า เช่น ตัวรับรู้ชนิดใช้แสงชนิดลำแสงผ่านตลอด (Through beam/broken beam) และข้อไฟฟ้าสำหรับวัดสัญญาณคลื่นไฟฟ้าหัวใจ (Electrocardiogram: ECG) ส่วนการทำงานของวงจรปรับสภาพสัญญาณ (Signal conditioning circuit) มีไว้สำหรับปรับลักษณะสัญญาณทางไฟฟ้าให้เหมาะสมกับวงจรปลายทาง โดยมีความสัมพันธ์แสดงได้ดังรูปที่ 1.1 โดยที่ขาเข้า (Input) เป็นพลังงานที่เกิดขึ้น และขาออก (Output) เป็นสัญญาณทางไฟฟ้า



รูปที่ 1.1 แผนภาพบล็อกของตัวเปลี่ยนแปลง (Transducer)

อินเทอร์เน็ต (Internet) ถูกพัฒนาตั้งแต่ปี พ.ศ. 2512 จากการเกิดเครือข่าย ARPANET (Advanced Research Projects Agency NETwork) ซึ่งเป็นเครือข่ายสำนักงานโครงการวิจัยชั้นสูงของกระทรวงกลาโหม ประเทศสหรัฐอเมริกา ต่อมาทั่วโลกได้พัฒนาอย่างต่อเนื่องจนกลายเป็นเครือข่ายที่มีใช้ในปัจจุบันอย่างแพร่หลาย นำมาสู่การกำเนิดอินเทอร์เน็ตของสรรพสิ่ง (Internet of Things: IoT) ซึ่งเป็นแนวคิดในการปรับปรุงสิ่งของหรือวัตถุ (Things) ที่มีวงจรอิเล็กทรอนิกส์เป็นส่วนประกอบ ให้สามารถเชื่อมต่อกับเครือข่าย เพื่อประมวลผล (Processing) และรับส่งข้อมูลระหว่างกันได้ (Communication) ทำงานร่วมกันอย่างเป็นระบบโดยไม่จำเป็นต้องอาศัยการปฏิสัมพันธ์กับมนุษย์ บนโครงสร้างพื้นฐาน (Infrastructure) ที่มีอยู่แล้ว เรียกว่าการทำงานอย่างอัตโนมัติ (Automation system) เมื่อตัวรับรู้เก็บข้อมูลและรับส่งพร้อมกันตลอดเวลาตามเวลาจริง (Real-time) ส่งผลให้เกิดข้อมูลในปริมาณสูง (Volume) และข้อมูลสามารถเกิดการเปลี่ยนแปลงอย่างรวดเร็ว (Velocity) และหลากหลาย (Variety) ซึ่งมีความน่าเชื่อถือและคุณภาพของข้อมูลที่แตกต่างกัน (Veracity) จนเกิดเป็นคุณลักษณะของข้อมูลทัต (Big data)

สำหรับปฏิบัติการในส่วนนี้ จะเป็นการใช้งานฮาร์ดแวร์และพัฒนาชุดคำสั่ง สำหรับการเก็บรวบรวมข้อมูล (Data acquisition) การวิเคราะห์และแปลความหมายข้อมูล (Data analysis and interpretation) และการนำเสนอข้อมูล (Data visualization) จากภาคีโครงสร้างพื้นฐานและแพลตฟอร์มอินเทอร์เน็ตของสรรพสิ่ง ความสัมพันธ์ระหว่างอินเทอร์เน็ตของสรรพสิ่ง ข้อมูลทั้ง แบบทั่วไป และกระบวนการวิเคราะห์ข้อมูลสัมพันธ์ แสดงดังแสดงรูปที่ 1.2



รูปที่ 1.2 ความสัมพันธ์ระหว่างอินเทอร์เน็ตของสรรพสิ่ง ข้อมูลทั้ง แบบทั่วไป และกระบวนการวิเคราะห์ข้อมูลสัมพันธ์

2. ฮาร์ดแวร์สำหรับอินเทอร์เน็ตของสรรพสิ่ง

2.1 ไมโครคอนโทรลเลอร์

สำหรับประมวลผลแบบคลาวด์ (Cloud computing) ในแต่ละจุดของอุปกรณ์อินเทอร์เน็ตของสรรพสิ่ง (Node devices) นอกจากจะรองรับอิเล็กทรอนิกส์ที่มีอยู่ภายในแล้ว ส่วนประมวลผลนิยมใช้ไมโครคอนโทรลเลอร์ (Microcontroller) ซึ่งประกอบด้วยหน่วยประมวลผลกลาง (Central processing unit : CPU) หน่วยความจำเข้าถึงแบบสุ่ม (Random access memory : RAM) หน่วยความจำแบบถาวร (Read-only memory : ROM) อินเตอร์เฟส และบัส (Interface and bus) เช่น ส่วนขาเข้าขาออก (Input/Output : I/O) เป็นต้น ไมโครคอนโทรลเลอร์มักถูกโปรแกรมให้อ่านค่าจากตัวรับสัญญาณ ประมวลผลระดับหนึ่ง และส่งข้อมูลออกไป ดังนั้นในการพิจารณาเลือกไมโครคอนโทรลเลอร์ให้เหมาะสมกับการใช้งาน ควรคำนึงถึงคุณสมบัติ และข้อจำกัดที่มี เช่น การใช้พลังงาน (Power consumption) หากใช้พลังงานที่น้อยอาจต้องแลกกับความสามารถในการประมวลผล และทรัพยากรับน้ำที่น้อยลง แหล่งข้อมูลในกลุ่มนักพัฒนา (Developer community) ที่เอื้อต่อการตั้งค่าตามจากปัญหาที่เกิดขึ้นในระหว่างการพัฒนา หรือการแก้ไขปัญหา

2.2 การเชื่อมต่อ กับเครือข่าย

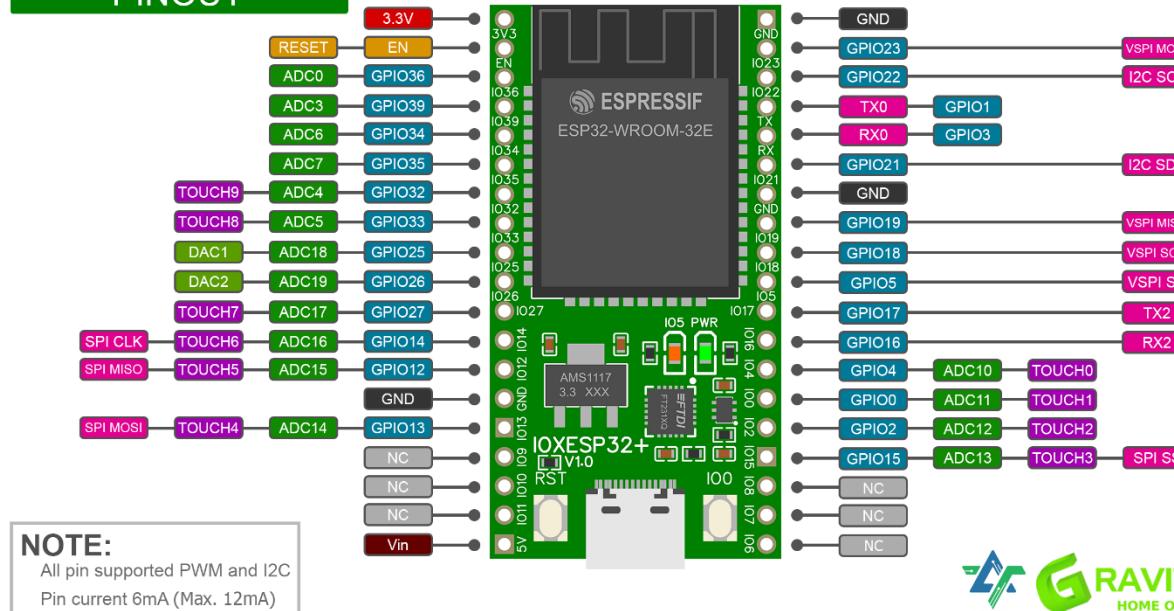
โพรโทคอล (Protocol) คือข้อกำหนดหรือข้อตกลงที่ประกอบด้วยกฎต่าง ๆ สำหรับการสื่อสารรูปแบบเฉพาะระหว่างอุปกรณ์ใดๆ เพื่อให้อุปกรณ์ที่มีคุณลักษณะแตกต่างกันสามารถสื่อสารกันได้ เช่น การสื่อสารระหว่างเครื่องแม่

ข่าย (Server) กับเครื่องลูกข่าย (Client) ผ่านเครือข่ายอินเทอร์เน็ต โทรศัพท์เคลื่อนที่นิยมใช้ในอินเทอร์เน็ตของสรรพสิ่ง ได้แก่ ไวไฟ (Wi-Fi: IEEE 802.11), TCP/IP (Transmission Control Protocol/Internet Protocol), Point-to-Point Protocol (PPP), Internet Protocol (IP), User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and Wireless Application Protocol (WAP) เป็นต้น สำหรับอุปกรณ์ บอร์ด IOXESP32+ ที่ใช้ในปฏิบัติการนี้ สามารถเชื่อมต่อไวไฟและบลูทูธกำลังงานต่ำ (Bluetooth Low Energy : BLE) ได้ในตัว โดยสามารถทำงานร่วมกับซอฟต์แวร์ไลบรารีบนแพลตฟอร์ม Arduino IDE ได้ทันที

2.3 บอร์ด IOXESP32+ (ESP32 by Espressif System)

IOXESP32+ เป็นบอร์ดพัฒนา ESP32 ใช้ไมโครคอนโทรลเลอร์ ESP32 ECO V3 System on Chip (SoC) แบบ 32 bit 2 CPU cores สัญญาณนาฬิกา 240 MHz มี IC Regulator จ่ายไฟ 3.3V 700mA และใช้ USB to TTL รุ่น FTDI (FT231XQ) ในการอัพโหลดโปรแกรมผ่านสาย USB-C มีความสามารถเชื่อมต่อเครือข่ายไร้สาย (Wireless communication) พร้อม WiFi (2.4G) และ Bluetooth 4.2 ขนาด Flash memory 4 MB ขนาด SRAM 520 kB Interface การเชื่อมต่อ I2C จำนวน 2 ช่อง, I2S จำนวน 2 ช่อง, SPI จำนวน 2 ช่อง, UART จำนวน 3 ช่อง, ADC จำนวน 16 ช่อง (หากเชื่อมต่อ WiFi จะใช้ได้ ADC ได้ 8 ช่อง), DAC จำนวน 2 ช่อง, CAN จำนวน 1 ช่อง นิสิตสามารถศึกษารายละเอียดของบอร์ด IOXESP32+ เพิ่มเติมได้ที่ https://docs.ioxesp32.com/ioxesp32_

IOXESP32+ PINOUT



รูปที่ 1.3 ผังตำแหน่งขาสัญญาณ (Pinout) ของบอร์ด IOXESP32+

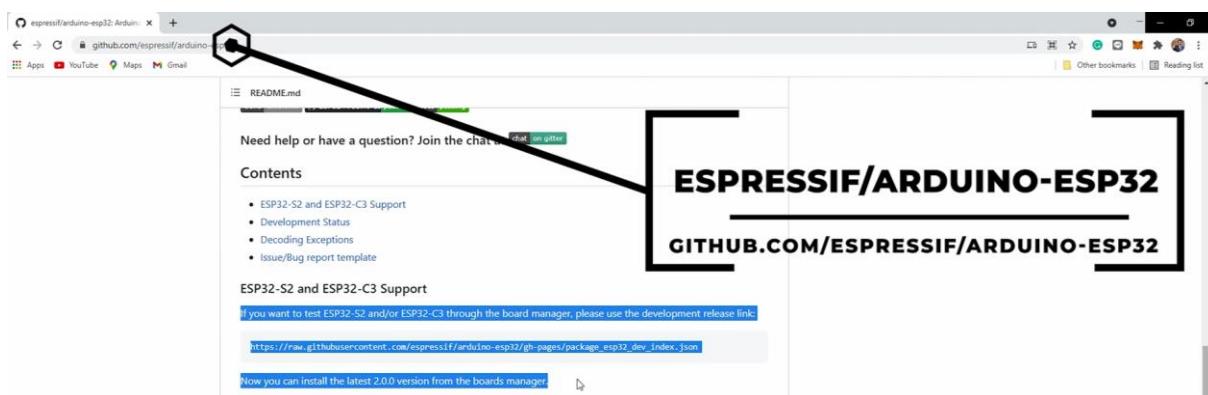
การทดลองที่ 1 | การติดตั้งชุดพัฒนาซอฟต์แวร์ (IDE) สำหรับ IOXESP32+ และการใช้ Push Button & OLED

*** มีสิ่งที่ต้องส่ง 2 รายการ คือ

1. Source Code ที่มีนามสกุลไฟล์ .ino เพียง 1 ไฟล์ ส่งผ่าน attachment box บน MyCourseVille
2. วีดีโอลิงก์แสดงการทำงาน ให้แนบ URL ของวีดีโอลิงก์ในช่องตอบคำถาม ส่งผ่าน text box บน MyCourseVille

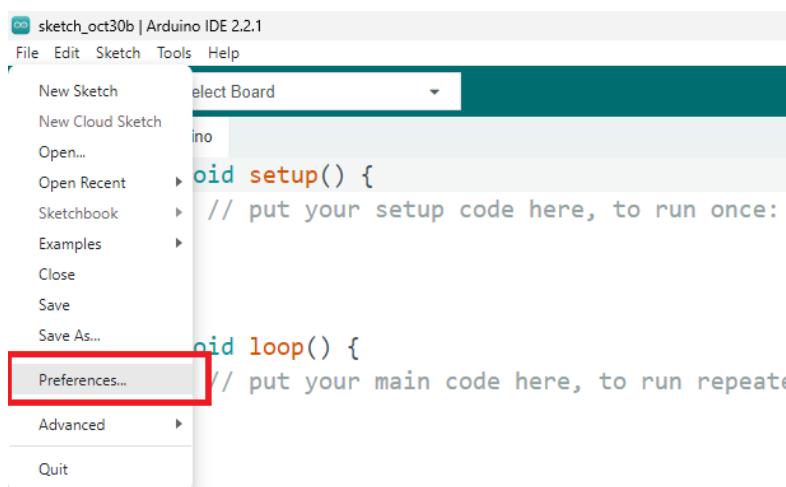
ขั้นตอนปฏิบัติ

1. ศึกษาวิธีการติดตั้งโปรแกรม Arduino IDE (แนะนำให้ระบบปฏิบัติการ Windows) จากคลิปวีดีโอประกอบรายวิชาที่ <https://www.youtube.com/watch?v=YZvR1Kbdghi> และทำความเข้าใจเกี่ยวกับ ESP32 Repository ที่ <https://github.com/espressif/arduino-esp32> เพื่อติดตั้ง Libraries ที่จำเป็น



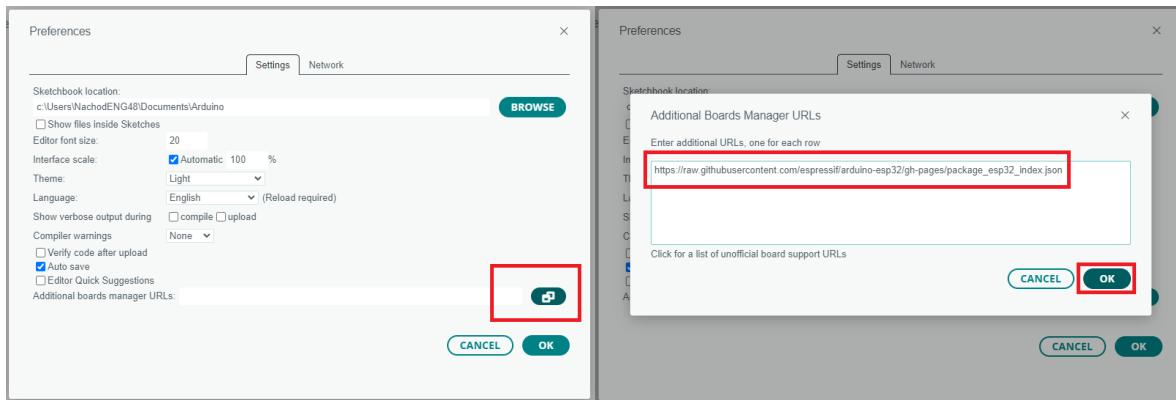
รูปที่ A1-1 ESP32 Repository <https://github.com/espressif/arduino-esp32>

2. เมื่อติดตั้งโปรแกรม Arduino IDE เสร็จเรียบร้อยแล้ว ให้ติดตั้ง ESP32 Repository เพิ่มบอร์ด ESP32 เข้าไปในโปรแกรม เริ่มจากคลิกที่ File >>> Preferences ดังรูปที่ A1-2



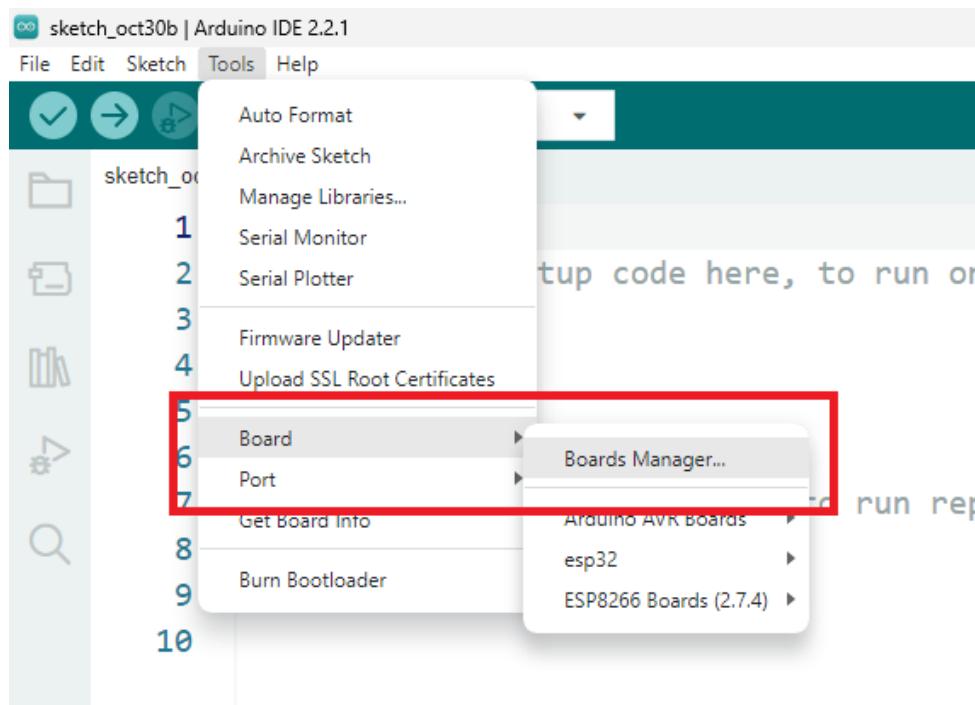
รูปที่ A1-2 การเปิด Preferences ของโปรแกรม Arduino IDE

3. ในหน้าต่าง Preferences ให้ไปที่ Additional Boards Manager URLs และคลิปที่ปุ่มดังรูปที่ A1-3 วาง Development release link ที่คัดลอกจาก ESP32 Repository และวิ่งกด OK ครั้ง

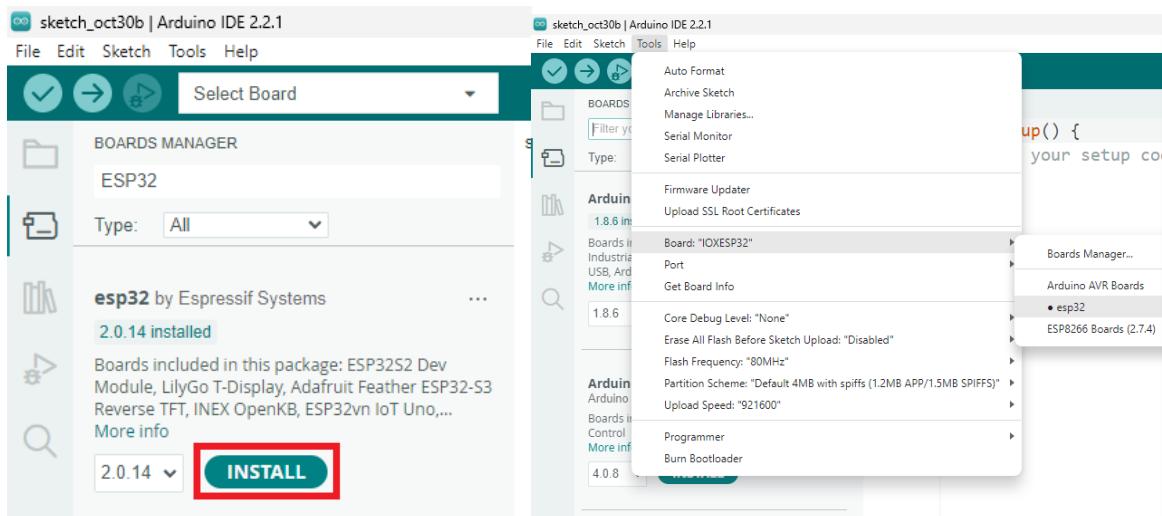


รูปที่ A1-3 ขั้นตอนการวาง Development release link ใน Additional Boards Manager URLs

4. เมื่อดำเนินการในข้อที่ 3 เสร็จแล้ว ให้คลิกที่ Tools >>> Board >>> Boards Manager... ดังรูปที่ A1-4 เพื่อเปิดหน้าต่าง Boards Manager ไปที่ช่องข้อความพิมพ์คำว่า ESP32 ตามด้วยกด Enter จะปรากฏดังรูปที่ A1-5 ให้กดปุ่ม Install รอจนกระทึ่ดติดตั้งเสร็จสิ้น จะขึ้นข้อความสีเขียว INSTALLED (ขั้นตอนนี้แนะนำให้เชื่อมต่อกับเครือข่ายที่มีความเสถียร และมีความเร็วอินเตอร์เน็ตที่สูง หากเกิดการขัดจังหวะขึ้น จะต้องรีบการติดตั้งใหม่ตั้งแต่ต้น)

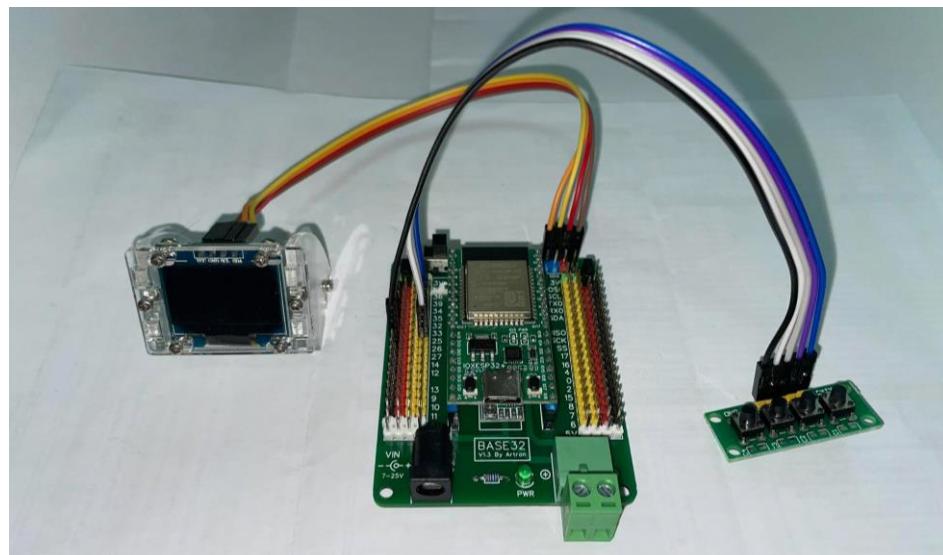


รูปที่ A1-4 การเปิดหน้าต่าง Boards Manager เพื่อติดตั้งบอร์ด ESP32



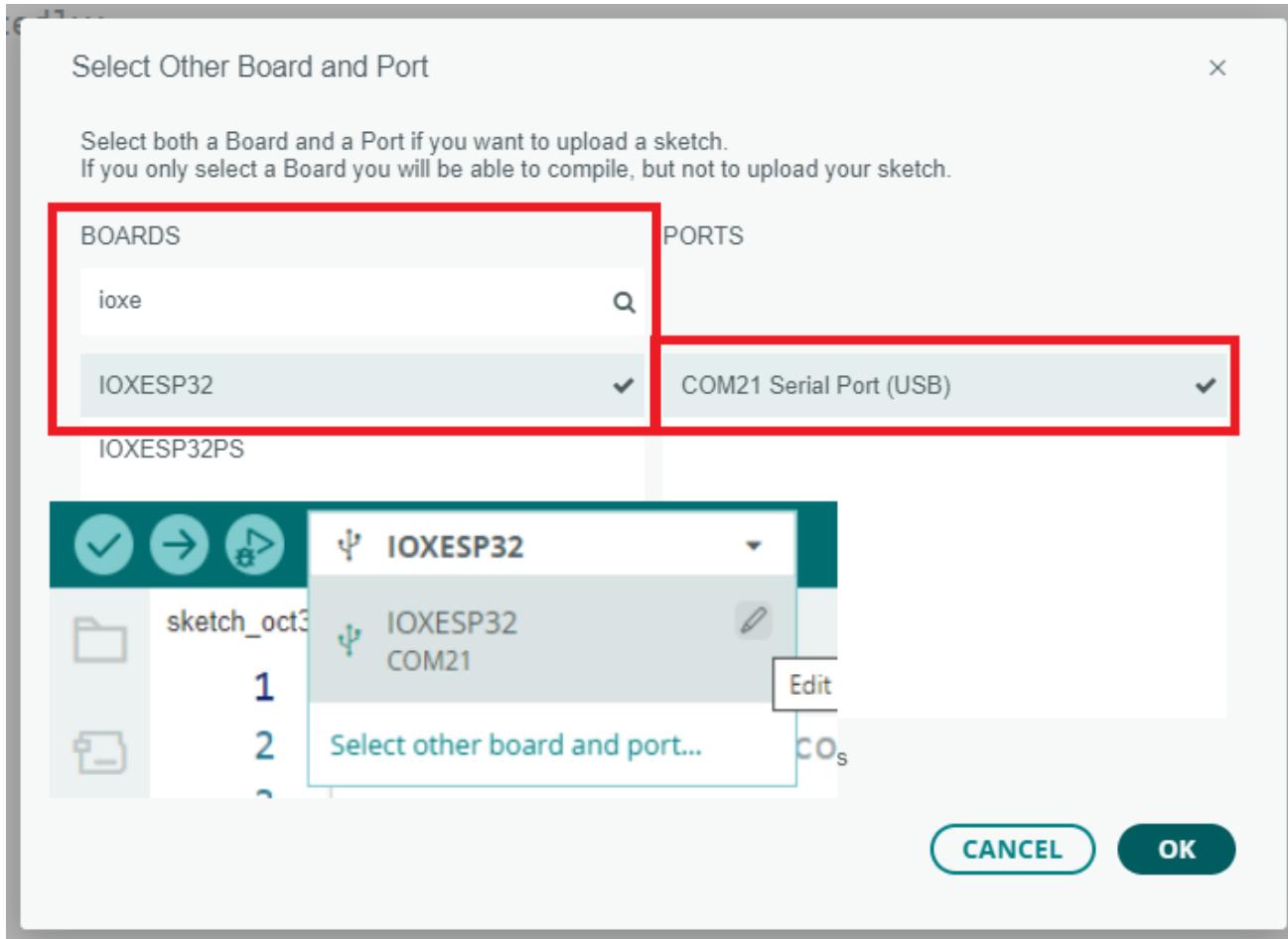
รูปที่ A1-5 หน้าต่าง Boards Manager ที่ติดตั้ง ESP32 Repository เรียบร้อยแล้ว

5. ต่อบอร์ดกับอุปกรณ์ต่างๆ ดังรูปที่ A1-6 โดยใช้ IOXESP32+ ร่วมกับ Expansion Board, 4 Channel Push Button และ OLED Panel 128x64 I2C โดยสามารถศึกษาวิธีการใช้งานหน้าจอ OLED ที่ <https://www.youtube.com/watch?v=2kckPMdi2XE> และการใช้งาน 4 Channel Push Button ที่ <https://www.youtube.com/watch?v=XsXG-DoFoMs> เพื่อทำการทดลองที่ 1



รูปที่ A1-6 การต่อบอร์ดกับอุปกรณ์ต่างๆ สำหรับทำการทดลองที่ 1

6. หลังจากต่อวงจรเรียบร้อยให้ต่อสาย USB-C เข้ากับบอร์ด IOXESP32+ และคอมพิวเตอร์ จากนั้นเข้าสู่โปรแกรม Arduino IDE จะสามารถเลือกบอร์ดดังรูปที่ A1-7 โดยการเข้าไปที่ Select Other Board and Port จากนั้นพิมพ์ในช่อง Boards ว่า IOXESP32 และเลือก Ports เป็น COMxx Serial Port (USB) ถ้าไม่ปรากฏให้ติดตั้ง Driver <https://www.silabs.com/documents/public/example-code/AN197SW.zip>



รูปที่ A1-7 การเลือกบอร์ด IOXESP32 และพอร์ตเพื่ออัปโหลดโปรแกรม

7. ทดสอบการทำงานของบอร์ดเบื้องต้น โดยอัปโหลดตัวอย่าง จากเมนู File >>> Examples >>> 01.Basics >>> Blink ให้แก้ชุดคำสั่งในหน้า Source code จาก LED_BUILTIN เป็น 5 ซึ่งเป็นหมายเลข IO5 ของบอร์ดที่มี LED built-in ติดตั้งอยู่ดังรูปที่ A1-8 และวิจัยกดปุ่มอัปโหลด  จากนั้นรอจนกระทิ้งขึ้นสถานะ Done uploading มุขยล่างของหน้าต่างโปรแกรม และสังเกตการกระพริบของ LED บนบอร์ด

```
27 #define LED_BUILTIN 5
28 void setup() {
29 // initialize digital pin LED_BUILTIN as an output.
30 pinMode(LED_BUILTIN, OUTPUT);
31 }
32
33 // the loop function runs over and over again forever
34 void loop() {
35 digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
36 delay(1000); // wait for a second
37 digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
38 delay(1000); // wait for a second
39 }
40
```



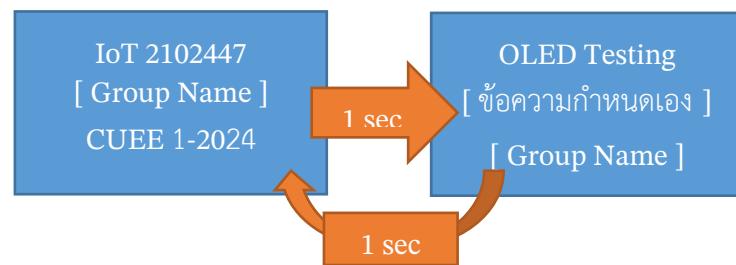
```
27 void setup() {
28 // initialize digital pin LED_BUILTIN as an output.
29 pinMode(5, OUTPUT);
30 }
31
32 // the loop function runs over and over again forever
33 void loop() {
34 digitalWrite(5, HIGH); // turn the LED on (HIGH is the voltage level)
35 delay(1000); // wait for a second
36 digitalWrite(5, LOW); // turn the LED off by making the voltage LOW
37 delay(1000); // wait for a second
38 }
```

รูปที่ A1-8 การแก้ Source code จาก LED_BUILTIN เป็น 5 เพื่อให้ LED แสดงผลได้

8. [การทดลองนี้ต้องส่ง .ino ไฟล์ที่ 1] เมื่อทดสอบการทำงานในข้อที่ 7 เรียบร้อยแล้ว ต่อไปคือการทดลองเกี่ยวกับหน้าจอแสดงผล OLED Panel 128x64 I2C โดยจะต้องเขียนโปรแกรมด้วย Source code ของตัวเองให้สามารถแสดงผลดังรูปที่ A1-9 ในหน้าจอแรกจะต้องแสดงคำว่า IoT 2102447 ตามด้วยชื่อกลุ่มใน myCourseVille และ CUCEE 2-2022 แสดงค้างไว้ 1 วินาที จากนั้นจึงเปลี่ยนหน้าจอที่แสดงคำว่า OLED Testing I2C Panel ตามด้วยชื่อกลุ่ม แสดงค้างไว้ 1 วินาที แล้วจึงวนกลับไปหน้าจอแรก บันทึกผลด้วยการถ่ายวิดีโออธิบายการทำงาน และส่ง Source code ที่มีนามสกุลไฟล์ .ino

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
...
void setup(){
    if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        Serial.println(F("SSD1306 allocation failed"));
        for(;;) // Don't proceed, loop forever
    }
    ...
}
void loop(){
    display...
    display...
    ...
    delay(...);
    display...
    display...
    delay(...);
}
```

Pseudocode



รูปที่ A1-9 แนวทางการเขียน Source code และข้อความบนหน้าจอที่ต้องทำส่ง

9. ส่งผลการทดลอง มีสิ่งที่ต้องส่ง 2 รายการ

*** มีสิ่งที่ต้องส่ง 2 รายการ คือ

1. Source Code ที่มีนามสกุลไฟล์ .ino เพียง 1 ไฟล์ ส่งผ่าน attachment box บน MyCourseVille
2. วีดีโอลิปแสดงการทำงาน ให้แนบ URL ของวีดีโอลิงในช่องตอบคำถาม ส่งผ่าน text box บน MyCourseVille

☆ จบการทดลองที่ 1 ☆

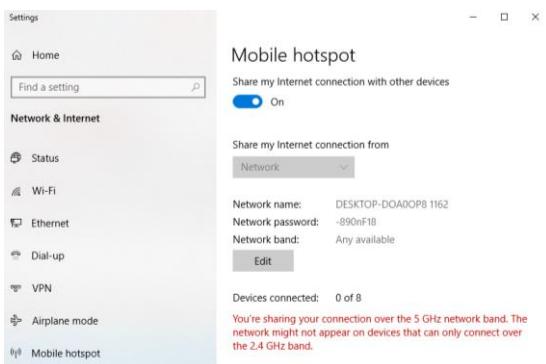
การทดลองที่ 2 | การเชื่อมต่อกับเครือข่ายไวไฟ (WiFi) และการเทียบฐานเวลาในประเทศไทย (NTP)

*** มีสิ่งที่ต้องส่ง 2 รายการ คือ

1. Source Code ที่มีนามสกุลไฟล์.ino เพียง 1 ไฟล์ ส่งผ่าน attachment box บน MyCourseVille
2. วีดิโอลิงค์แสดงการทำงาน ให้แนบ URL ของวีดิโอลิงค์ในช่องตอบคำถาม ส่งผ่าน text box บน MyCourseVille

ขั้นตอนปฏิบัติ

1. ให้เตรียม Personal Hotspot หรือ Mobile Hotspot จากอุปกรณ์ที่ทำงานบนระบบปฏิบัติการ iOS, iPadOS, Android, Windows 11 หรือระบบปฏิบัติการอื่น ๆ ที่รองรับ ดังรูปที่ A2-1 เพื่อจ่ายสัญญาณ WiFi ให้กับบอร์ด IOXESP32+ เพื่อเชื่อมต่ออินเตอร์เน็ตสำหรับดึงค่าเวลาเทียบฐานเวลาในประเทศไทย (NTP)



รูปที่ A2-1 การใช้งาน Mobile Hotspot บน Windows 11

2. ให้เขียนโปรแกรมเทียบฐานเวลา (NTP) ด้วยชอร์ซโค้ดตัวอย่างดังรูป A2-2 โดยให้แก้ไข ssid และ password ให้ตรงกับ Personal Hotspot หรือ Mobile Hotspot ของนิสิตไม่ เช่นนั้นจะไม่สามารถเชื่อมต่ออินเตอร์เน็ตได้ ให้ต่อสาย USB-C เข้ากับบอร์ด IOXESP32+ แล้วจึงกดปุ่มอปปโนหลัง จากนั้นรอจนกระทึ้งขึ้นสถานะ Done uploading มุมซ้ายล่างของหน้าต่างโปรแกรม จากนั้นจึงกดปุ่ม เพื่อเปิดหน้าต่าง Serial monitor เพื่อสังเกตการทำงานต่อไป

```
#include "WiFi.h"
#include <time.h>

// Replace with your network credentials
const char* ssid      = "[Your SSID]";
const char* password = "[Your Password]";

// Define NTP Client to get time
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 3600 * 7; // Thailand GMT+7
const int  daylightOffset_sec = 3600 * 0;

// Store time and date value here
char timeSeconds[3];
```

```

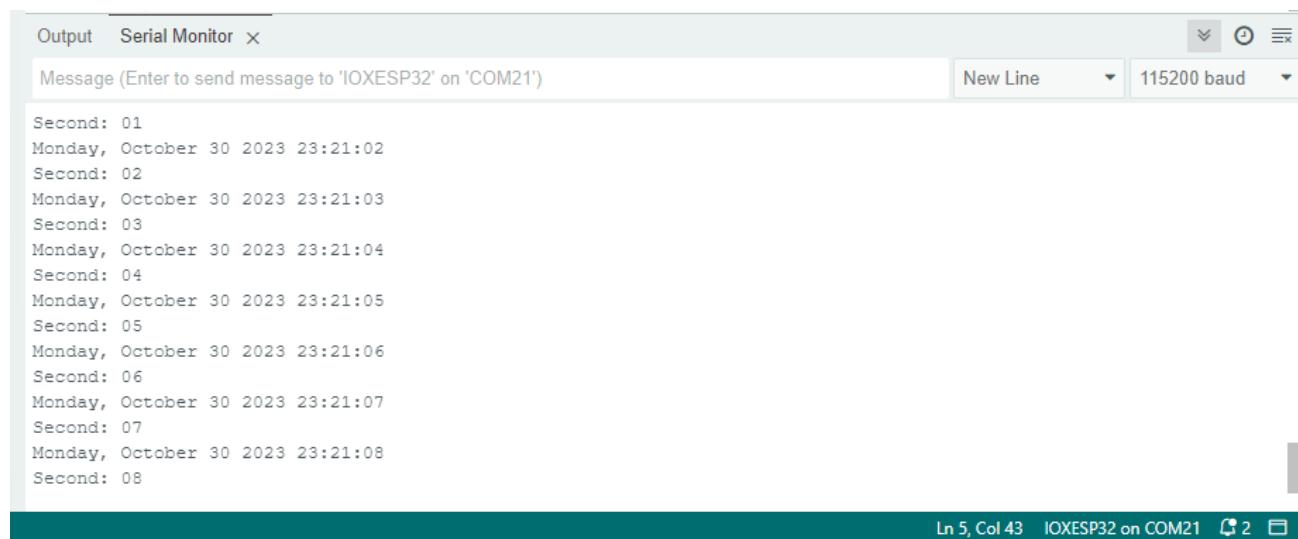
void setup() {
    // Initialize Serial Monitor
    Serial.begin(115200);
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    // Print local IP address and start web server
    Serial.println("");
    Serial.println("WiFi connected.");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());

    // Initialize a NTPClient to get time
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}
void loop() {
    struct tm timeinfo;
    if (!getLocalTime(&timeinfo)) {
        Serial.println("Failed to obtain time");
    }
    strftime(timeSeconds, 3, "%S", &timeinfo);
    Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");
    Serial.println("Second: " + String(timeSeconds));
    delay(1000);
}

```

รูปที่ A2-2 Source code สำหรับทดลอง ดึงค่าเวลาเทียบฐานเวลาในประเทศไทย (NTP)

*** แนะนำใช้ Library ของ NTPClient by Fabrice Weinberg version 3.2.1 ***



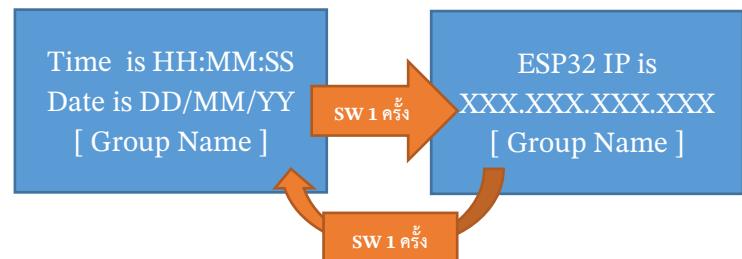
รูปที่ A2-3 เมื่อ ESP32 เชื่อมต่อ WiFi และดึงค่าเวลาเทียบฐานเวลาในประเทศไทย (NTP) สำเร็จ

3. [การทดลองนี้ต้องส่ง .ino ไฟล์ที่ 1] ให้ออกแบบการทำงานของอุปกรณ์ IoT โดยการใช้หน้าจอแสดงผล OLED Panel 128x64 I2C แสดงค่าเวลาเทียบฐานเวลาในประเทศไทย (NTP) ดังรูป A2-4 จากอินเตอร์เน็ต (เมื่อใช้การแสดงผลผ่าน Serial monitor) ซึ่งจะต้องแสดงผลตั้งต่อไปนี้

- บรรทัดแรก แสดงเวลาจาก NTP ประกอบด้วย ชั่วโมง : นาที : วินาที เช่น 14:0:0 หรือ 14:10:59 ถ้าทำให้แสดงแบบ 2-Digit 00:00:00 จะมี Bonus point เพิ่ม
- บรรทัดที่สอง แสดงวันที่จาก NTP ประกอบด้วย วัน / เดือน / ปี เช่น 26/10/21 ถ้าทำให้แสดงผลชื่อเดือนและปี 4 หลัก แบบ 26/October/2021 จะมี Bonus point เพิ่ม
- แสดงชื่อกลุ่มนิสิตที่ทำการทดลอง ตามรายชื่อกลุ่มนิสิตที่ปรากฏใน myCourseVille

เมื่อกดปุ่ม Push Button 1 ครั้ง ข้อมูลบนหน้าจอ OLED จะเปลี่ยนไปแสดงหมายเลข IP Address ของ IOXESP32+ ที่กำลังเชื่อมต่ออินเตอร์เน็ต ณ ปัจจุบัน และตามด้วยชื่อกลุ่มนิสิต สามารถประยุกต์ใช้จากฟังก์ชัน WiFi.localIP() ได้ โดยที่หน้าจอตั้งกล่าวจะติดค้างไว้จนกว่าจะมีการกดปุ่ม Push Button อีก 1 ครั้ง หน้าจอ OLED จะกลับไปแสดงเวลา วันที่ และชื่อกลุ่มนิสิตเดิม บันทึกผลด้วยวิดีโอหรือภาพการทำงาน และส่ง Source code ที่มีนามสกุลไฟล์ .ino

```
#include <SPI.h>
#include <Wire.h>
#include <WiFi.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
...
void setup(){
    ...
}
void loop(){
    ...
}
```



รูปที่ A2-4 แนวทางการเขียน Source code และข้อมูลบนหน้าจอที่ต้องทำส่ง

4. ส่งผลการทดลอง มีสิ่งที่ต้องส่ง 2 รายการ

*** มีสิ่งที่ต้องส่ง 2 รายการ คือ

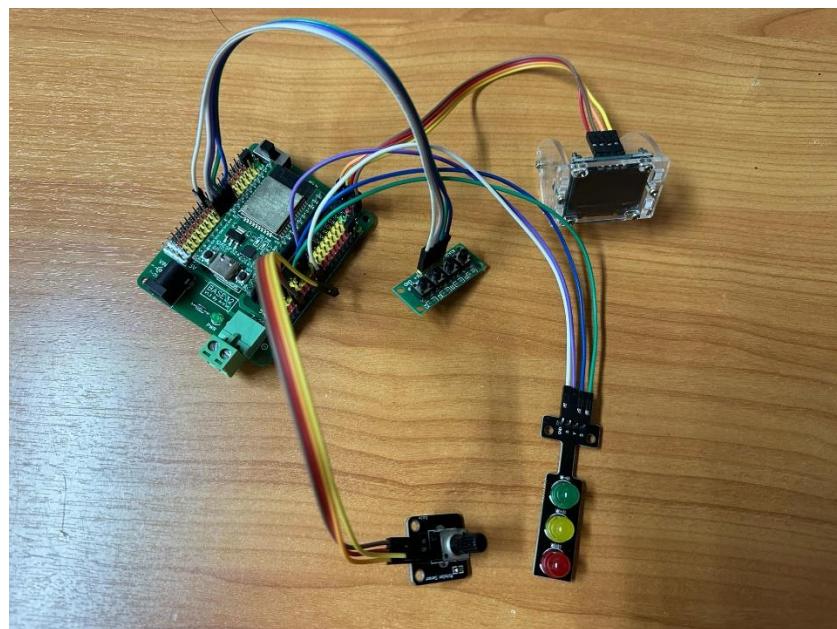
1. Source Code ที่มีนามสกุลไฟล์ .ino เพียง 1 ไฟล์ ส่งผ่าน attachment box บน MyCourseVille
2. วิดีโอลิงก์แสดงการทำงาน ให้แนบ URL ของวิดีโอลิงก์ในช่องตอบคำถาม ส่งผ่าน text box บน MyCourseVille

☆ จบการทดลองที่ 2 ☆

การทดลองที่ 3 | การทดลองควบคุมความสว่าง LED ด้วย PWM ผ่าน Potentiometer

ขั้นตอนปฏิบัติ

1. ต่อบอร์ดกับอุปกรณ์ต่างๆ เพิ่มเติมดังรูปที่ A1-6 โดยใช้ IOXESP32+ ร่วมกับ Expansion Board, Potentiometer ต่อเข้าช่องที่เป็น ADC (Analog to digital converter) และ 8mm LED (R/Y/G) module โดยชุดโมดูลหลอด LED จะเป็น common ground



รูปที่ A3-1 การต่อบอร์ดกับอุปกรณ์ต่างๆ สำหรับทำการทดลองที่ 3

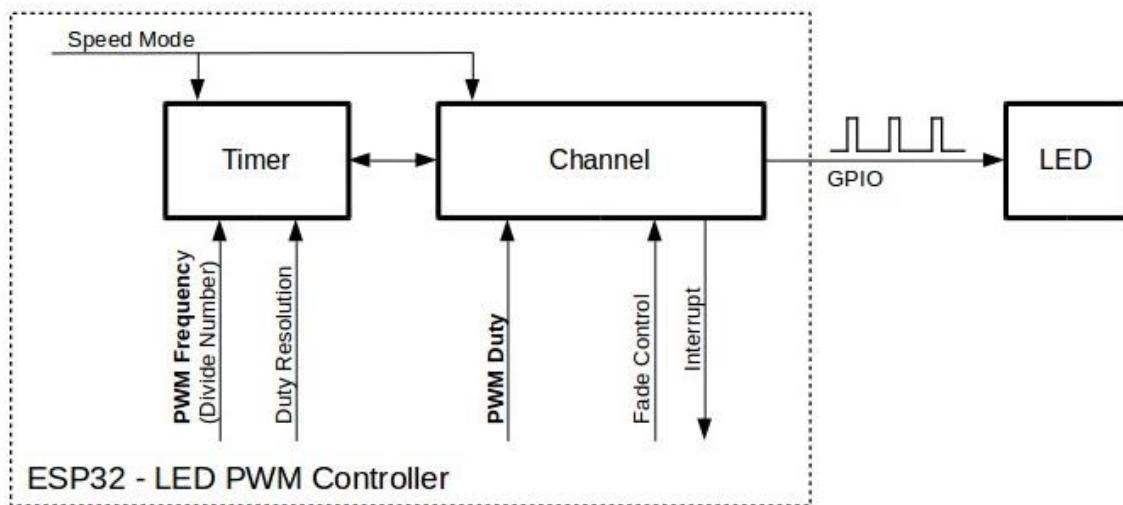
2. ให้นิสิตเขียนโปรแกรมควบคุมความสว่างหลอด LED ด้วย PWM ผ่าน Potentiometer โดยการกำหนดความถี่และเปลี่ยนแปลง Duty cycle ของสัญญาณขาออกของ LED ทั้ง 3 หลอด และแสดงค่า Duty cycle ออกทางหน้าจอ OLED panel ดังรูปที่ A3-2

```
#include <SPI.h>
#include <Wire.h>
#include <WiFi.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
...
void setup(){
    ...
}
void loop(){
    ...
}
```

PWM Duty Cycle
Value is XXX
[Group Name]

รูปที่ A3-2 แนวทางการเขียน Source code และข้อความบนหน้าจอที่ต้องทำส่ง

3. ในการเขียนคำสั่ง PWM (Pulse width modulation) บน ESP32 จะแต่งต่างจากบอร์ด Arduino ที่เป็นการกำหนด Pin นั้นโดยตรงด้วยคำสั่ง `analogWrite()` กำหนดค่าระหว่าง 0 ถึง 255 แต่จะเป็นการกำหนด Matrix bus หรือ PWM channel สามารถควบคุมได้สูงสุด 16 channel (0-15) กำหนดบิตของรูปคลื่นได้ละเอียดกว่า และสามารถปรับเปลี่ยนความถี่ได้ Arduino Uno ประมาณ 490Hz และ ESP32 ประมาณ 5,000 Hz



รูปที่ A3-3 การทำงานควบคุม PWM ของ ESP32

วิธีการเขียนโปรแกรมจะประกอบด้วย 3 ส่วน ส่วนแรกการกำหนดค่าตัวควบคุม PWM โดยที่ `PWM_CHANNEL` เป็นตัวกำหนดช่องในการควบคุม `PWM_FREQ` เป็นตัวกำหนดความถี่ ถ้าต้องการให้ใกล้เคียงกับ Arduino Uno ให้ตั้งเป็น 500 และ `PWM_RESOLUTION` เป็นการกำหนดความละเอียดของการเปลี่ยนแปลง PWM หากตั้งค่าเป็น 8 นั้นหมายความว่ามีค่าเป็น 0-255 หรือ 8 bit

```

const int PWM_CHANNEL = 0;
const int PWM_FREQ = 500;
const int PWM_RESOLUTION = 8;

const int MAX_DUTY_CYCLE = (int)(pow(2, PWM_RESOLUTION) - 1);

const int LED_1_OUTPUT_PIN = ...;
const int LED_2_OUTPUT_PIN = ...;
const int LED_3_OUTPUT_PIN = ...;
const int POT_PIN = ...;
```



ส่วนลัดมาการกำหนดขา GPIO สำหรับควบคุมหลอด LED จะใช้คำสั่งดังนี้ จะเขียนไว้ใน void setup()

```
ledcAttachChannel(uint8_t pin, uint8_t PWM_FREQ , uint8_t PWM_RESOLUTION,  
                  uint8_t PWM_CHANNEL);
```

ส่วนสุดท้ายการอ่านค่า Analog จาก Potentiometer ด้วยฟังก์ชัน analogRead(pin) มาใช้ในการควบคุม PWM ของหลอด LED โดยจะมีการใช้ฟังก์ชัน map(value, fromLow, fromHigh, toLow, toHigh) ซึ่งจะแปลงค่าที่อ่านได้จาก Potentiometer จาก 12 bit ไปเป็น 8bit Duty cycle สำหรับขับหลอด LED ผ่านฟังก์ชัน ledcWriteChannel(uint8_t channel, uint8_t dutycycle);

```
int dutyCycle = analogRead(POT_PIN);  
dutyCycle = map(dutyCycle, 0, 4095, 0, MAX_DUTY_CYCLE);  
ledcWriteChannel(PWM_CHANNEL, dutyCycle);  
delay(100);
```

4. ให้นิสิตประยุกต์ใช้ปุ่มกดในการเลือกควบคุม LED แต่ละสี เช่น เมื่อกดปุ่ม K1 จะควบคุมเฉพาะหลอดสีแดง (RED LED) กดปุ่ม K2 จะควบคุมเฉพาะหลอดสีเหลือง (YELLOW LED) กดปุ่ม K3 จะควบคุมเฉพาะหลอดสีเขียว (GREEN LED) และกดปุ่ม K4 จะควบคุมได้ทั้ง 3 หลอดพร้อมกัน (R/Y/G LED) และในแต่ละการทดลองในนิสิตลองปรับ Potentiometer เพิ่มลดความสว่างของทุกการทดลองและบันทึกวิดีโอลิป พูดอธิบายพฤติกรรมการทำงานให้ชัดเจน
5. ส่งผลการทดลอง มีสิ่งที่ต้องส่ง 4 รายการ

*** มีสิ่งที่ต้องส่ง 2 รายการ บน MyCourseVille คือ

1. Source Code ที่มีนามสกุลไฟล์ .ino เพียง 1 ไฟล์เท่านั้น ไม่ต้องบีบอัด ส่งผ่าน attachment box
2. วิดีโอลิปแสดงการทำงาน ให้พิมพ์ URL ของวิดีโอลงในช่องตอบคำถาม ส่งผ่าน text box

☆ จบการทดลองที่ 3 ☆

การทดลองที่ 4 | การทดลองปุ่มกด 4 Push Button Switch และการเขียนโปรแกรมลำดับ

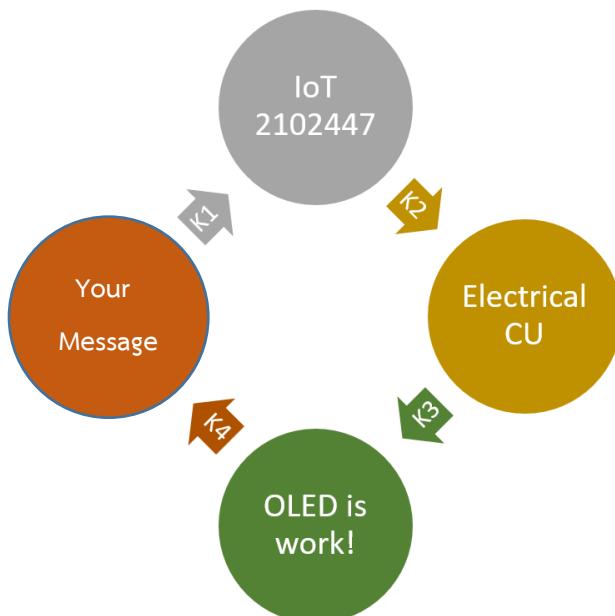
*** Special Bonus ***

ขั้นตอนปฏิบัติ

6. การทดลองเกี่ยวกับ 4 Channel Push Button ให้ศึกษาการทำงานแบบ Debouncing โดยสามารถดูตัวอย่าง จากเมนู File >>> Examples >>> Digital >>> Debounce เพื่อประยุกต์การทำงานของ Push Button ทั้ง 4 ได้แก่ปุ่ม K1, K2, K3 และ K4 ตามลำดับ ศึกษาการใช้งาน Internal Pull-up ได้จาก <https://www.youtube.com/watch?v=XsXG-DoFoMs> โดยที่หน้าจอ OLED Panel 128x64 I2C จะต้องแสดงผลเป็น State ดังต่อไปนี้

- เมื่อกดปุ่ม K1 ครั้ง หน้าจอจะต้องแสดง IoT 2102447
- เมื่อกดปุ่ม K2 1 ครั้ง หน้าจอจะต้องแสดง CUEE IoT
- เมื่อกดปุ่ม K3 1 ครั้ง หน้าจอจะต้องแสดง OLED is work!
- เมื่อกดปุ่ม K4 1 ครั้ง หน้าจอจะต้องแสดง ...เขียน Message ของตัวเอง...(Your Message)

โดยที่ไม่จำเป็นต้องเรียก State ซึ่งแต่ละ State สามารถข้ามไปยัง State อื่นๆ ได้ เช่น เมื่อยูที่ IoT 210244 เมื่อกด K4 1 ครั้ง หน้าจอจะต้องเปลี่ยนไปเป็น message ของตัวเอง หรือ เมื่อยูที่ OLED is work! เมื่อกด ปุ่ม K2 1 ครั้ง หน้าจอจะต้องเปลี่ยนไปเป็น CUEE IoT เมื่อเขียนโปรแกรมสำเร็จแล้วให้เรียนรู้และสังเกตการทำงานว่า Source Code ที่เขียนขึ้นนั้นทำงานตามรูปที่ A3-1 หรือไม่อย่างไร



ข้อกำหนดของ State machine

- เมื่อยูที่ IoT 2102447 สามารถกดปุ่ม K2, K3 หรือ K4 ได้
- เมื่อยูที่ Electrical CU สามารถกดปุ่ม K3, K4 หรือ K1 ได้
- เมื่อยูที่ OLED is work! สามารถกดปุ่ม K4, K1 หรือ K2 ได้
- เมื่อยูที่ Your Message สามารถกดปุ่ม K1, K2 หรือ K3 ได้

รูปที่ A4-1 ลำดับ State ของการกดปุ่ม 4 Channel Push Button และข้อความที่แสดงบนหน้าจอ OLED

7. ส่งผลการทดลอง มีสิ่งที่ต้องส่ง 3 รายการ

*** มีสิ่งที่ต้องส่ง 3 รายการ บน MyCourseVille คือ

1. แผนภาพสถานะการทำงานของระบบจากเงื่อนไขที่กำหนด รูปแบบ PDF ส่งผ่าน attachment box
2. Source Code ที่มีนามสกุลไฟล์ .ino เพียง 1 ไฟล์เท่านั้น ไม่ต้องบีบอัด ส่งผ่าน attachment box
3. วิดีโอลิปแสดงการทำงาน ให้พิมพ์ URL ของวิดีโอลงในช่องตอบคำถาม ส่งผ่าน text box

2102447 ปฏิบัติการวิศวกรรมอิเล็กทรอนิกส์

Electronic Engineering Laboratory

ผู้สอนประจำวิชา ผศ.ดร.สุรีย์ พุ่มรินทร์ และ อ.ดร.ณพวงศ์ ปันธนาธรรม
ผู้สอนปฏิบัติการ ณัทกร เกษมสำราญ (ภาคการศึกษาต้น 2567)



ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์

ก่อตั้ง ๒๔๗๗

จ. พัลงกรน์ มหาวิทยาลัย

การเก็บและนำเสนอภาพข้อมูล IoT ด้วย NETPIE

(IoT Data Collection and Visualization via NETPIE)

วัสดุและอุปกรณ์

- คอมพิวเตอร์ติดตั้งแอปพลิเคชัน Arduino
- บอร์ด IOXESP32+ และบอร์ดขยายขา
- โมดูลจอ OLED 0.96" 128x64 pixel
- โมดูลวัดอุณหภูมิ ความชื้น และความดัน BME280
- สาย USB, สายไฟ และอุปกรณ์อื่นๆ ที่เกี่ยวข้อง



"IoT Learning Kit" 2102447 Electronics Engineering Laboratory

ELECTRICAL
ENGINEERING
SINCE 1930
CHULALONGKORN UNIVERSITY



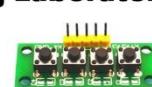
IOXESP32+ Detail



ESP32 Repository

Components list :

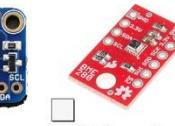
1.) IOXESP32+ V1.0 Embedded Board	1 each
2.) BASE32 Pinout Expansion Board	1 each
3.) OLED Display 128x64 0.96"	1 each
4.) OLED Housing Bracket	1 each
5.) Breadboard / Protoboard	1 each
6.) 4 SPST Switch Module	1 each
7.) 8mm LED (R/Y/G) Module	1 each
8.) Potentiometer Module	1 each
9.) USB-C to USB-A Cable	1 each
10.) Jumper Wire (Male to Male)	10 each
11.) Jumper Wire (Male to Female)	10 each
12.) Jumper Wire (Female to Female)	10 each



Sensor selection list :



TSL2561 Lux sensor



BME280 (Temp, Humid, Pressure)



PIR Motion Sensor



DS18B20 with Module

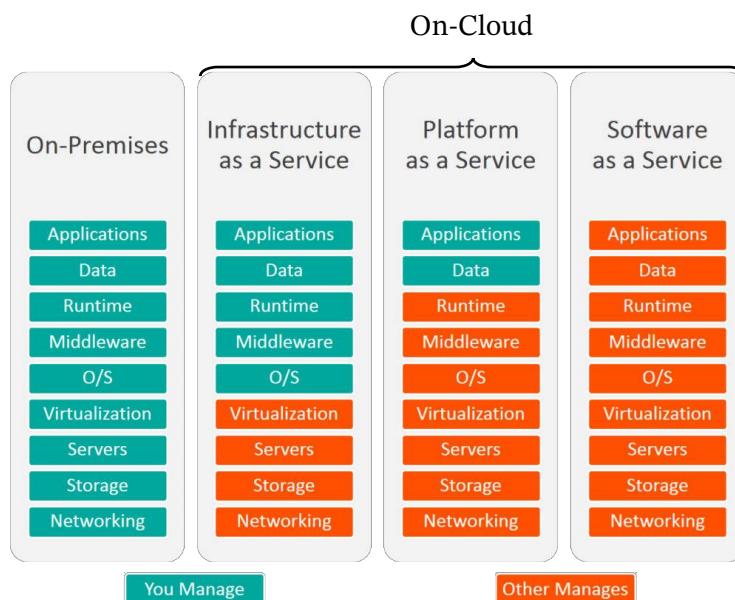


docs.loxesp32.com/loxesp32+

หมายเหตุ : ในกล่องไม่มีรายการที่ 5.) Breadboard / Protoboard ให้

1. บทนำ

การประมวลผลแบบคลุ่มเมฆ (cloud computing) เกิดขึ้นจากแบบจำลองธุรกิจ (business model) เพื่อการใช้ทรัพยากรยังมืออยู่อย่างจำกัดมาใช้งานร่วมกันให้เกิดประโยชน์สูงที่สุด โดยผู้ใช้งานไม่จำเป็นต้องเป็นเจ้าของฮาร์ดแวร์แต่จ่ายค่าบริการให้กับผู้ให้บริการเพื่อใช้บริการทรัพยากรในรูปแบบคล้ายกับการเช่า (subscription) เพื่อใช้งานทรัพยากรหรือซอฟต์แวร์ที่ต้องการ [Gollmann 2005.] โดยผู้ใช้งานนั้นไม่จำเป็นต้องมีความรู้ในเชิงเทคนิค รูปแบบการให้บริการอาจแบ่งตามขอบเขตความรับผิดชอบของผู้ให้บริการดังรูปที่ 1-1



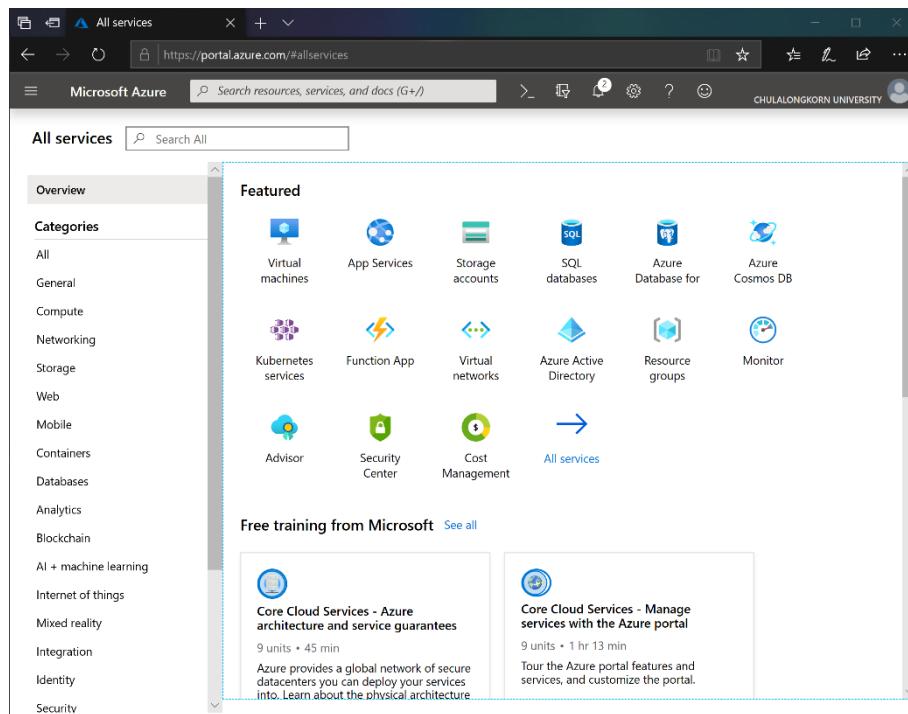
รูปที่ 1-1 ขอบเขตความรับผิดชอบในการบริหารจัดการและการซ่อมบำรุง

1.1 Software-as-a-Service

Software-as-a-Service (SaaS) คือการให้บริการซอฟต์แวร์สำเร็จรูปพร้อมใช้งานได้ทันทีโดยไม่จำเป็นต้องปรับแต่งใดๆ ข้อดีคือง่ายสำหรับผู้ใช้ที่ไม่มีความรู้ทางเทคนิค ข้อจำกัดคือไม่สามารถปรับแต่งหรือพัฒนาส่วนต่อขยายเพื่อเพิ่มฟังก์ชันการทำงานตามที่ต้องการด้วยตัวเองได้ หรือทำได้อย่างจำกัด

1.2 Platform-as-a-Service

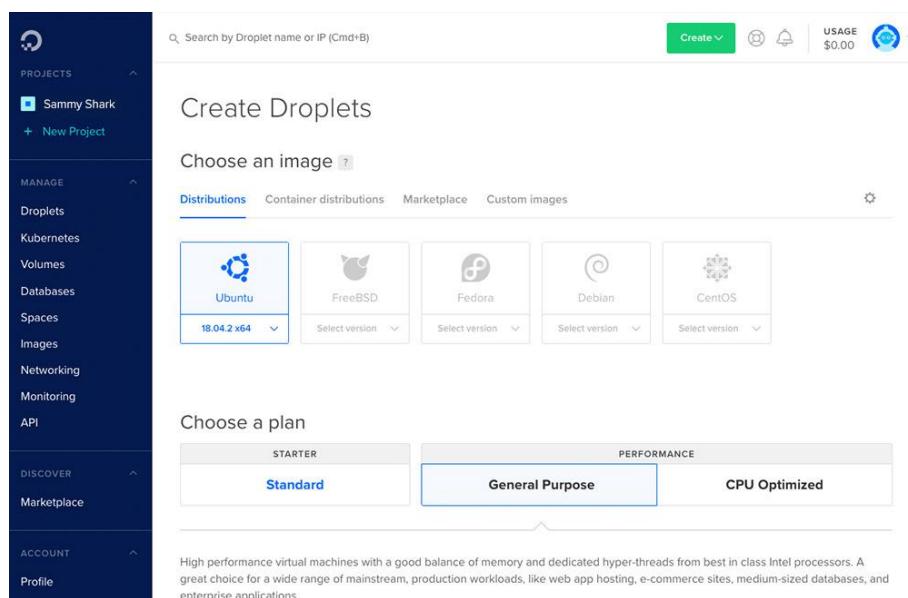
Platform-as-a-Service (PaaS) คือการให้บริการฮาร์ดแวร์และการปรับแต่งที่พร้อมสำหรับการพัฒนาแอปพลิเคชัน ผู้ให้บริการจะติดตั้งคอมไพล์เตอร์ (compiler) อินเทอร์พريเตอร์ (interpreter) ระบบจัดการฐานข้อมูล (database management system) เว็บเซิร์ฟเวอร์ (web server) รวมไปถึงแอปพลิเคชันอื่น ๆ ที่จำเป็น หมายความว่าผู้พัฒนาซอฟต์แวร์เพื่อทำงานเฉพาะอย่าง ข้อดีคือนักพัฒนาไม่จำเป็นต้องเตรียมสภาพแวดล้อมที่เหมาะสมสำหรับนักพัฒนาซอฟต์แวร์เพื่อทำงานเฉพาะอย่าง ข้อจำกัดคือนักพัฒนาไม่จำเป็นต้องใช้บริการแบบ PaaS (และ IaaS) ดังรูปที่ 1-2



รูปที่ 1-2 หน้าหลักของ Microsoft Azure Portal

1.3 Infrastructure-as-a-Service

Infrastructure-as-a-Service (IaaS) คือ การเช่าเซิร์ฟเวอร์เสมือน (virtual server) โดยผู้ให้บริการจะแบ่งทรัพยากร硬件และเครือข่ายให้ผู้ใช้บริการเสมือนได้เชิร์ฟเวอร์ส่วนตัวสำหรับใช้งาน ผู้ใช้สามารถติดตั้งซอฟต์แวร์หรือปรับแต่งสภาพแวดล้อมที่เหมาะสมได้เองดังตัวอย่างในรูปที่ 1-3 การเปรียบเทียบการให้บริการระหว่าง SaaS, PaaS และ IaaS สามารถแสดงได้ดังตารางที่ 1.1 และตัวอย่างผู้ให้บริการบางส่วนดังรูปที่ 1-4 และ 1-5



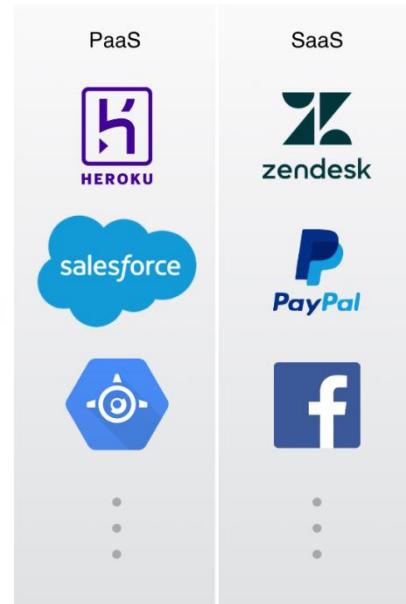
รูปที่ 1-3 หน้าจอหลักของ Digital Ocean ส่วนของผู้ใช้สำหรับจัดการบริการ IaaS

តារាងទី 1.1 ការប្រើប្រាស់ពីរប្រភេទការផ្តល់ប្រើប្រាស់ SaaS, PaaS និង IaaS

Features	IaaS	PaaS	SaaS
What you get	You get the infrastructure & pay accordingly .Freedom to use or install any OS, software or composition	Here you get what you demand. Software, hardware, OS, web environment. You get the platform to use & pay accordingly	Here you don't have to worry about anything. A pre-installed, pre-configured package as per your requirement is given and you only need to pay accordingly.
Importance	The basic layer of Computing	Top of IaaS	It is like a Complete package of services
Technical Difficulties	Technical knowledge required	You get the Basic setup but still the knowledge of subject is required.	No need to worry about technicalities. The SaaS provider company handles everything.
Deals with	Virtual Machines, Storage (Hard Disks), Servers, Network, Load Balancers etc	Runtimes (like java runtimes), Databases (like mySql, Oracle), Web Servers (tomcat etc)	Applications like email (Gmail, Yahoo mail etc), Social Networking sites (Facebook etc)
Popularity Graph	Popular among highly skilled developers, researchers who require custom configuration as per their requirement or field of research.	Most popular among developers as they can focus on the development of their apps or scripts. They don't have to worry about traffic load or server management etc.	Most popular among normal consumers or companies which reply on softwares such as email, file sharing, social networking as they don't have to worry about the technicalities.



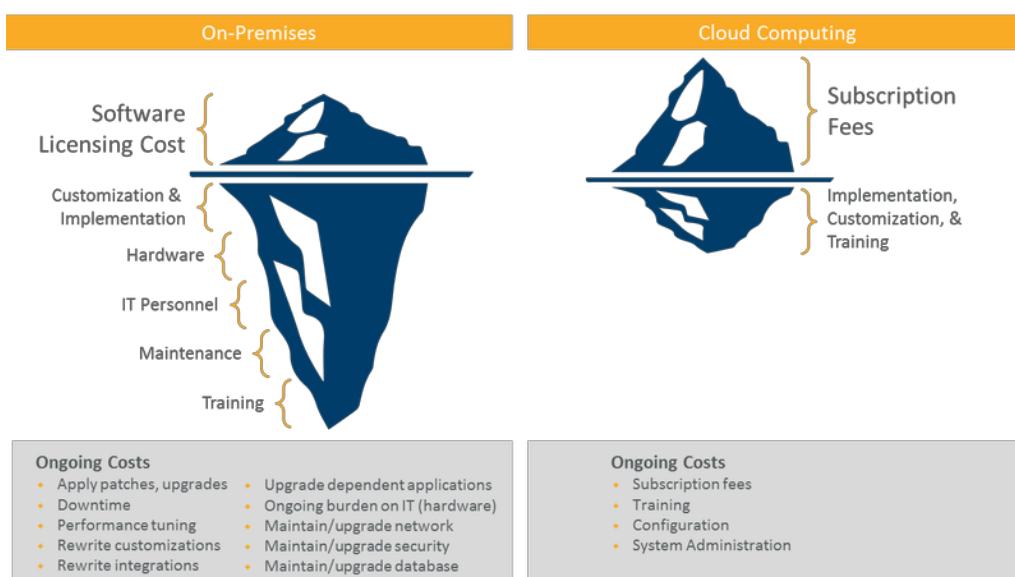
រូបទី 1-4 ត្រូវយកដំឡើងដូចខាងក្រោមនៃការ SaaS, PaaS និង IaaS



รูปที่ 1-5 ตัวอย่างผู้ให้บริการ SaaS, PaaS และ IaaS

1.4 เปรียบเทียบรายการค่าใช้จ่ายระหว่าง On-Cloud และ On-Premises

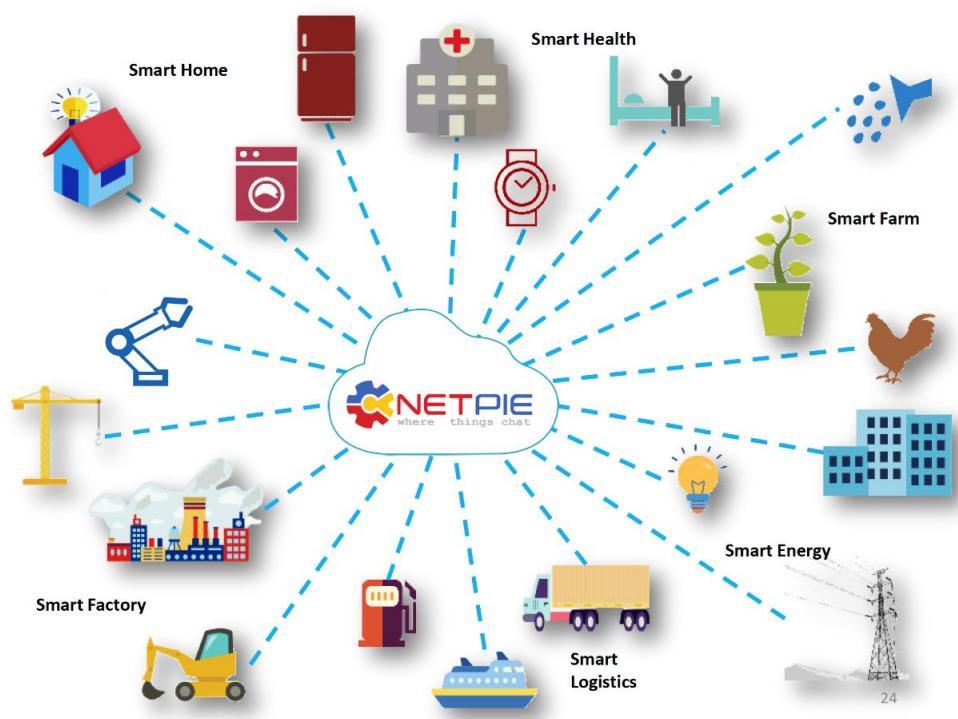
On-Premise คือ ระบบเซิร์ฟเวอร์ที่แบบดั้งเดิม (traditional) ที่ผู้ใช้จัดซื้อ ติดตั้ง บำรุงรักษา และอัปเกรดเอง อายุ่สมอ ผู้ใช้อาจว่าจ้างผู้ดูแลด้วยสัญญาบริการดูแลและบำรุงรักษา (Maintenance Service Agreement: MA) ซึ่งมีอายุประมาณ 3-5 ปี และเมื่อครบกำหนด หากธุรกิจของผู้ใช้งานเติบโต ระบบของผู้ใช้งานมีปริมาณงานมากขึ้นหรือต้องการใช้ทรัพยากรามากขึ้น ผู้ใช้มักต้องจัดซื้ออาร์ดแวร์ใหม่ที่มีประสิทธิภาพมากขึ้นกว่าเดิม การใช้บริการระบบเซิร์ฟเวอร์แบบ On-Cloud เป็นการผลักภาระภาระบำรุงรักษาไปยังผู้ให้บริการแทนแลกกับการจ่ายค่าบริการ รายการค่าใช้จ่ายระหว่างระบบเซิร์ฟเวอร์ทั้งสองรูปแบบแสดงดังรูปที่ 1-6



รูปที่ 1-6 เปรียบเทียบรายการค่าใช้จ่ายระหว่าง On-Premise กับ On-Cloud

2. NETPIE

NETPIE (Network Platform for Internet of Everything) อ่านว่า เน็ต-พาย เป็นชื่อเครื่องหมายการค้า ของเทอร์เน็ตของสรรพสิ่งที่ให้บริการโบรอกเกอร์ MQTT แบบกระจาย (distributed) ที่สามารถเชื่อมต่ออุปกรณ์ ハードแวร์ และแอปพลิเคชันที่เขียนโดยโปรแกรมภาษาใดๆ ได้ เพื่อรองรับการพัฒนาวัตกรรมอัจฉริยะด้านต่าง ๆ ดัง รูปที่ 2-1 ในช่วงต้น NETPIE ถูกวิจัยและพัฒนาโดยศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ต่อมาได้ถูกถ่ายทอดเทคโนโลยีด้วยการอนุญาตให้ใช้สิทธิ (licensing) แก่บริษัท เน็กซ์พาย จำกัด (NEXPIE) เพื่อพัฒนาต่อยอดและให้บริการเชิงพาณิชย์ ในขณะเดียวกัน NETPIE ยังคงให้บริการให้ฟรีสำหรับกลุ่มเมก เกอร์ นักเรียน นักศึกษา นักพัฒนา และอุตสาหกรรม SME ต่อไปภายใต้การสนับสนุนจากคณะกรรมการกิจการ กระจายเสียง กิจการโทรทัศน์และกิจการโทรคมนาคมแห่งชาติ หรือ กสทช.



รูปที่ 2-1 การเชื่อมต่ออุปกรณ์ハードแวร์ และแอปพลิเคชันต่าง ๆ ด้วย NETPIE

2.1 บัญชีผู้ใช้ NETPIE

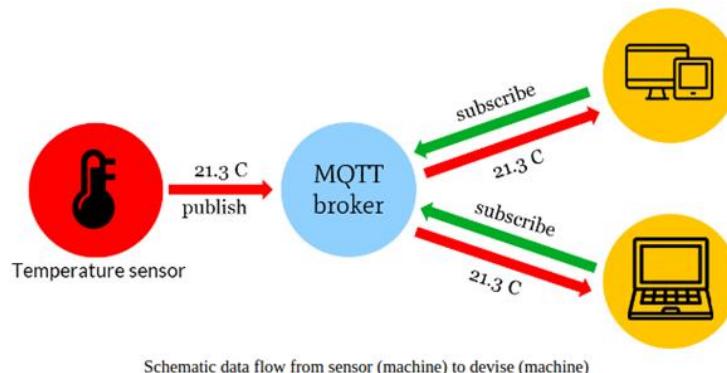
บัญชีผู้ใช้แบบฟรีความมีข้อจำกัด ดังนี้

Project	3	Projects
Connected devices	10	Devices
Real-time messages	9,000,000	Messages/month
Time-series data storage	1,000,000	Point-month
Shadow read/write	500,000	Operations/month
API call	800,000	Operations/month
Dashboard	3	Freeboards/project
Freeboard connection	3	Concurrent views

Trigger and action	5,000	Operation/month
--------------------	-------	-----------------

2.2 ໂພຣໂທຄອລ MQTT

MQTT (Message Queuing Telemetry Transport) เป็นໂພຣໂທຄອລມີສາປັຕຍກຣມແບບແຄລເອນຕໍ່/ເຊື່ອງ/ໄວ່ອ່າງ (client/server) ບນ້ຳປະປະຢຸກຕີໃຊ້ຈານຮູ້ອ້ານບັນສຸດໃນແບບຈຳລອງ OSI ໄຄລເອນຕໍ່ສາມາຮັດ Publish ຂໍ້ມູນໄປຢັ້ງໂບຮກເກອຮ໌ດ້ວຍການ Push ແລະສາມາຮັດ Subscribe ເພື່ອຮັບຂໍ້ມູນຈາກໂບຮກເກອຮ໌ດ້ວຍການ Request ດັ່ງແຜນກາພຽບທີ 2-2

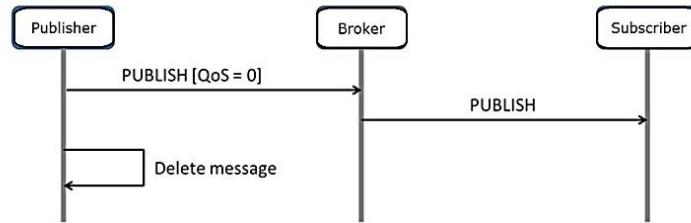


ຮູບທີ 2-2 ແຜນກາພແສດງຟັງກັນການຮັບ/ສ່າງຂອງ MQTT ໄຄລເອນຕໍ່ກັບໂບຮກເກອຮ໌ (ເຊື່ອງ/ໄວ່ອ່າງ)

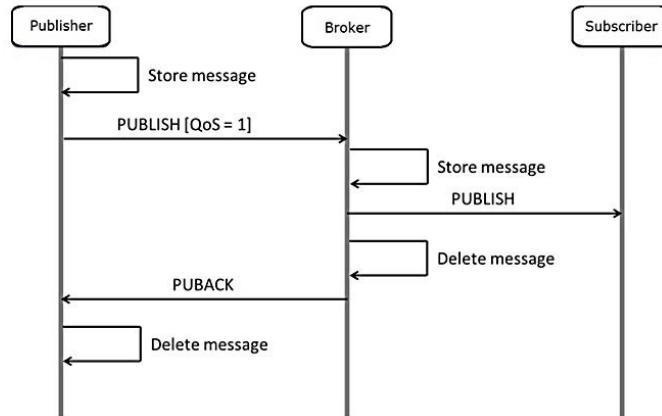
ເມື່ອເປົ້າຍບເຫັນ MQTT ກັບ HTTP (REST) ຈະພບວ່າ MQTT ມີຄວາມໄດ້ເປົ້າຍທີ່ໂບຮກເກອຮ໌ສາມາຮັດ Push ຂໍ້ຄວາມ (message) ໄປຢັ້ງໄຄລເອນຕໍ່ໄດ້ຕາມເຫດກາຮັນ (event-driven) ໃນຂະໜາດທີ່ເມື່ອໃໝ່ HTTP ຜັ່ງໄຄລເອນຕໍ່ຕ້ອງ Poll ຂໍ້ມູນເປັນຮະຍະ ຈະ ຊຶ່ງແຕ່ລະຄັ້ງຕ້ອງມີການສ່າງການເຂື່ອມຕ່ອງຂຶ້ນໃໝ່ແລະອາຈະໄມ້ມີຂໍ້ມູນໃໝ່ໄດ້ ໃຫ້ອັບເດັດ ທາກ ຕ້ອງການໃໝ່ຮບບທໍາງນາມແບບເວລາຈິງຍ່ອມໝາຍເຖິງຕ້ອງຕັ້ງຄາບເວລາການໂພລໃຫ້ສັ້ນ ເກີດ Overhead ໃນການຮັບ/ສ່າງຂໍ້ມູນ ແຕ່ລະຄັ້ງທໍາໃໝ່ການໃຊ້ຈານແບນດົວດົກທີ່ມີປະສິທິກາພລດລົງ ຈຶ່ງເປັນຂໍ້ໄດ້ເປົ້າຍຂອງໂພຣໂທຄອລ MQTT ເຫັນກັບ HTTP

2.3 ການຮັບປະກັນຄຸນກາພຂອງໂພຣໂທຄອລ MQTT

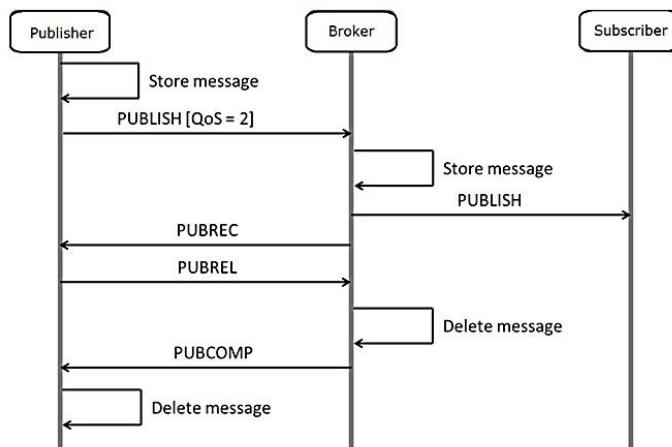
ການຮັບປະກັນຄຸນກາພ (Quality of Service: QoS) ການຈັດກາຮ່ອງທາງແບນດົວດົກທີ່ຂອງຮະບບເຄືອຂ່າຍ ສໍາຮັບໂພຣໂທຄອລ MQTT ໄຄລເອນຕໍ່ຈະເປັນຜູ້ກໍານົດຮະດັບຂອງບໍລິການສ່າງແລະຮັບຂໍ້ຄວາມຮູ້ອ່າງ QoS ທີ່ຕົນຕ້ອງການໃນແຕ່ລະ ລັບຂໍ້ອັນ (topic) ອີ່ Publish ຮູ້ອ່າງ QoS 0, ອີ່ອ່ານນ້ອຍຫົ່ງຄັ້ງ (At Least Once: QoS 1) ແລະ ລັບຂໍ້ອັນ (Exactly Once: QoS 2) ແສດກາຮັບ/ສ່າງຂໍ້ມູນໄດ້ດັ່ງຮູບທີ 2-3, 2-4 ແລະ 2-5 ຈຶ່ງສາມາຮັດເຮັດວຽກໃຊ້ແບນດົວດົກທີ່ຈາກນ້ອຍໄປມາກໄດ້ແກ່ QoS 0, QoS 1 ແລະ QoS 2 ຕາມລຳດັບ



รูปที่ 2-3 แผนผังการสื่อสารเพื่อ Publish ข้อความด้วย QoS 0



รูปที่ 2-4 แผนผังการสื่อสารเพื่อ Publish ข้อความด้วย QoS 1



รูปที่ 2-5 แผนผังการสื่อสารเพื่อ Publish ข้อความด้วย QoS 2

2.4 Device บน NETPIE (<https://docs.netpie.io/device-config.html>)

คลาสเอนต์หรือ Device บน NETPIE มีโครงแบบ (configuration) อยู่ 4 ส่วน คือ Shadow, Schema, Trigger and Event Hook, และ Feed

Shadow คือ ฐานข้อมูลเสมือนของ Device เป็นฐานข้อมูลเล็ก ๆ ที่มีคู่อยู่ Device ทุกตัว ใช้สำหรับเก็บข้อมูลต่าง ๆ เกี่ยวกับอุปกรณ์นั้น ๆ (shadow data) เช่น ข้อมูลที่เกิดจากเซนเซอร์ ข้อมูลการกำหนดองค์ประกอบต่าง ๆ เป็นต้น

Schema คือ แม่พิมพ์ข้อมูลที่กำหนดไว้เพื่อใช้กับ Shadow สำหรับ Device ที่ต้องมีการจัดการข้อมูลครวิสร้าง Schema ของข้อมูลเตรียมไว้ เสมือนเป็น Template ทำให้เซิร์ฟเวอร์สามารถตรวจสอบข้อมูล (data validation) แปลงข้อมูล (data transformation) เช่น เปลี่ยนหน่วยของข้อมูล เป็นต้น และ เก็บข้อมูลลงในฐานข้อมูลอนุกรมเวลา (timeseries database)

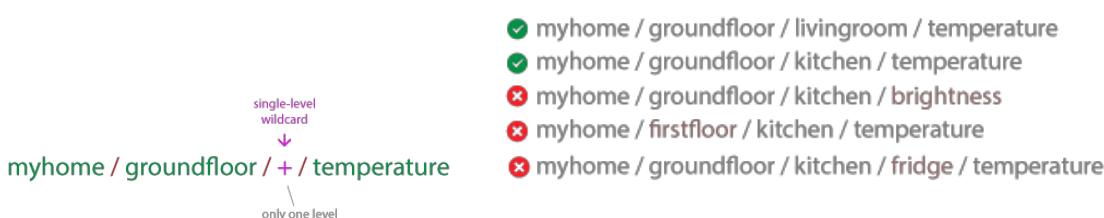
Trigger and Event Hook เป็นระบบที่ผูกการเปลี่ยนแปลงข้อมูลของ Shadow ตามเงื่อนไขที่ถูกตั้งค่าไว้ (trigger) เข้ากับการกระทำการนอก (event hook) เช่น การตั้งค่าแจ้งเตือนตามสถานะต่าง ๆ

Feed เป็นระบบจัดการและดูแลข้อมูลใน Timeseries Data เป็นต้นของแต่ละ Device ซึ่งจะแสดงในรูปแบบของกราฟเส้น แยกตามพิล์ดที่กำหนดใน Schema และยังสามารถดาวน์โหลดข้อมูลออกมายเป็นไฟล์ .csv ได้

2.4 Topic บน MQTT และบน NETPIE

Topic บน MQTT เป็นข้อมูลประเภท UTF-8 String ซึ่งมีรูปแบบเดียวกับพาท (path) หรือเส้นทางการเข้าถึงข้อมูลของระบบจัดเก็บไฟล์ คือสามารถจัดเป็นลำดับชั้นได้ด้วยการขึ้นต้นด้วย “/” ตัวอย่างเช่น myhome/groundfloor/livingroom/temperature เป็นต้น ซึ่งของหัวข้อที่นำไปใช้ได้ต้องขึ้นต้นด้วยตัวอักษรใดก็ได้หนึ่งตัวอักษร ยกเว้น “\$” เนื่องจากเป็นหัวข้อส่วนสำหรับใช้ภายในเบรกเกอร์เท่านั้น

โคลเลอนต์สามารถเลือก Publish หรือ Subscribe เฉพาะ Topic หรือหลาย Topic พร้อมกันโดยใช้ อักขระตัวแทน (wildcard character) โดยที่เครื่องหมาย “+” หมายถึงการระบุแบบหนึ่งระดับ (single-level) ส่วนเครื่องหมาย “#” หมายถึงการระบุแบบหลายระดับ (multi-level) ตัวอย่างเช่น กรณี myhome/groundfloor/+/temperature หมายถึง Topic ที่ขึ้นต้นด้วย myhome/groundfloor ลำดับชั้นถัดไป 1 ลำดับเป็นอะไรก็ได้ และลำดับชั้นสุดท้ายเป็น temperature ดังรูปที่ 2-6 กรณี myhome/groundfloor/# หมายถึง Topic ที่ขึ้นต้นด้วย myhome/groundfloor ทั้งหมด ดังรูปที่ 2-7



รูปที่ 2-6 การระบุแบบหนึ่งระดับ



รูปที่ 2-7 การระบุแบบหลายระดับ

Topic บน NETPIE มีการกำหนดเพิ่มเติมโดยแบ่งออกเป็น Message Topic และ Shadow Topic

Message Topic ใช้สำหรับ Publish หรือ Subscribe ข้อมูลเพื่อสื่อสารระหว่าง Device ที่อยู่ภายใต้ Group เดียวกัน Topic จะขึ้นต้นด้วย @msg เป็นลำดับชั้นแรก

Shadow Topic ใช้สำหรับจัดการ Shadow Data ของ Device มี Topic การ Publish และ Subscribe ดังนี้

Publish Topic	
@shadow/data/update	อัพเดทค่าใน Shadow Data โดยส่ง Payload ดังนี้ { "data":{ "field name 1": value 1, "field name 2": value 2, ..., "field name n": value n }}
@shadow/batch/update	อัพเดทค่าใน Shadow แบบเป็นชุดข้อมูล (shadow batch update)
@shadow/data/get	ร้องขอ Shadow Data ของตัวเองแบบทั้งหมด รอรับข้อมูลโดย Subscribe Topic @private/# หรือ @private/shadow/data/get/response

Subscribe Topic	
@private/shadow/data/get/response	รอรับ Shadow Data เมื่อมีการร้องขอข้อมูลไป
@private/shadow/batch/update/response	รอรับข้อความตอบกลับการอัพเดท Shadow แบบเป็นชุดข้อมูล
@private/#	รอรับทุกข้อมูล Topic ที่ขึ้นต้นด้วย @private/ รวมถึงข่าวสารต่างๆ ที่ Platform ต้องการแจ้งก็จะถูก Publish มาที่ Topic นี้

2.5 ไลบรารี PubSubClient (<https://github.com/knolleary/pubsubclient>)

PubSubClient คือซอฟต์แวร์ไลบรารีที่ทำงานเป็นโคล่อน์สำหรับทำการส่งรับ Message อย่างง่ายผ่านการ Publish หรือ Subscribe กับเซิร์ฟเวอร์ที่รองรับ MQTT โดยสามารถ Publish ที่ QoS 0 และ Subscribe ที่ QoS 0 หรือ 1 ฟังก์ชันบางส่วนในไลบรารี PubSubClient แสดงในตารางที่ 2.1

ตารางที่ 2.1 ฟังก์ชันบางส่วนในไลบรารี PubSubClient

ฟังก์ชัน	ต้นแบบฟังก์ชันและหน้าที่
constructor	PubSubClient(Client& client); ฟังก์ชันคอนสตรัคเตอร์ (constructor) สร้างวัตถุจากคลาส PubSubClient
setServer	PubSubClient& setServer(const char * domain, uint16_t port); กำหนดเซิร์ฟเวอร์ที่ต้องการเชื่อมต่อ
connect	bool connect(const char* clientId, const char* username, const char* password); เชื่อมต่อกับเซิร์ฟเวอร์
connected	bool connected(); ตรวจสอบสถานะการเชื่อมต่อกับเซิร์ฟเวอร์
disconnect	void disconnect(); ยกเลิกการเชื่อมต่อกับเซิร์ฟเวอร์
setCallback	PubSubClient& setCallback(MQTT_CALLBACK_SIGNATURE); ใช้กำหนดฟังก์ชันที่จะถูกเรียกเมื่อโคลอ่อนตัวรับ Message ใหม่
publish	bool publish(const char* topic, const char* payload); ส่ง Message ไปยัง Topic ที่กำหนด
subscribe	bool subscribe(const char* topic, uint8_t qos); สมัครติดตามการรับ Message ที่เกิดขึ้นใน Topic นั้น ๆ
unsubscribe	bool unsubscribe(const char* topic); ยกเลิกการรับ Message ที่เกิดขึ้นใน Topic นั้น ๆ
loop	bool loop(); เรียกใช้อย่างสม่ำเสมอเพื่อให้โคลอ่อนตัวประมวลผล Message ที่เข้ามาและยืนยันการเชื่อมต่อกับเซิร์ฟเวอร์

2.6 Freeboard บน NETPIE

Freeboard คือส่วนการนำเสนอดữาข้อมูล (data virtualization) ของ Device อย่างง่ายบน NETPIE สามารถแสดงผลได้หลากหลาย ทั้งแบบ Text, Gauge, Indicator Light, Button, Toggle, FeedView, Map เป็นต้น และใส่คำสั่ง Javascript สำหรับ Action ต่าง ๆ ได้

2.7 RESTful API บน NETPIE

RESTful API เป็นช่องทางสำหรับให้ Device เรียกใช้บริการด้วย HTTP Protocol ซึ่งหมายความว่าใช้เป็นช่องทางในการสนับสนุน (integration) ระบบต่างๆ ทั้งที่มีอยู่แล้วหรือกำลังจะพัฒนาขึ้นมาใหม่ โดยไม่จำกัดว่าจะต้องพัฒนาจากภาษาโปรแกรมใด (ทดสอบการทำงานของ API ได้ที่ <https://trial-api.netpie.io>)

3. Multitasking

Multitasking คือการทำงานมากกว่าหนึ่งงาน (task) พร้อมกัน โดยแนวคิดคือทุกงานจะต้องทำพร้อม ๆ กัน ในเวลาเดียวกัน (parallelism) แต่ในความเป็นจริงนั้นหน่วยประมวลผลหรือ CPU จะสลับการทำงานไปมาระหว่างงาน (context switching) อย่างรวดเร็วจนดูเหมือนสามารถทำงานหลายงานได้ภายในเวลาเดียวกัน สำหรับการทำงานแบบ Multitasking บน Arduino IDE สามารถทำได้ดังนี้

3.1 Multitasking โดยใช้ millis()

Multitasking โดยใช้ millis() ทำได้โดยกำหนดให้ทุกงานอยู่ภายใต้ loop() และอาศัยฟังก์ชัน millis() เพื่ออ่านค่าเวลาและตรวจสอบว่าแต่ละงานึงทำงานเดลาก่อนการทำงานอีกครั้งหนึ่งแล้วหรือไม่ ดังรูป 3-1 โดยแต่งานควรใช้เวลาสั้น ๆ และไม่ควรมีการวนลูปรอที่ใช้เวลานาน ๆ

```
void loop(){
    /* 250ms Task */
    currentMillis = millis();
    if(currentMillis - previousMillis1 > 250){
        previousMillis1 = currentMillis;
        /* Task 1 code here */
    }

    /* 500ms Task */
    currentMillis = millis();
    if(currentMillis - previousMillis2 > 500){
        previousMillis2 = currentMillis;
        /* Task 2 code here */
    }

    /* 1000ms Task */
    currentMillis = millis();
    if(currentMillis - previousMillis3 > 1000){
        previousMillis3 = currentMillis;
        /* Task 3 code here */
    }
}
```

รูปที่ 3-1 ตัวอย่าง Multitasking โดยใช้ millis() ทุก ๆ 250, 500 และ 1000 ms

3.2 Multitasking โดยใช้ FreeRTOS

FreeRTOS เป็นระบบปฏิบัติการสำหรับอุปกรณ์ Embedded ใช้ได้ในหลากหลายไมโครคอนโทรเลอร์ ซึ่งทาง Espressif ได้นำ FreeRTOS มาใส่ในชุดพัฒนาของ ESP32 ทำให้สามารถใช้งานใน Arduino IDE โดยตรงได้เลย การสร้าง Task สามารถทำได้โดยใช้ฟังก์ชัน xTaskCreate() ซึ่งมีรูปแบบการใช้งานดังนี้

```
void xTaskCreate(TaskFunction_t pvTaskCode,           // พังก์ชันที่จะไปเรียกทำงานในรูปแบบ Task
                  const char *pcName,             // ชื่อของ Task ที่สร้าง
                  int usStackDepth,            // พื้นที่บนแรมสำหรับตัวแปรแบบ Local ใน Task
                  void *pvParameters,          // ข้อมูลที่ต้องการส่งเข้าไปในพังก์ชัน Task
                  int uxPriority,              // ระดับความสำคัญของ Task
                  TaskHandle_t *pxCreatedTask); // ตัวแปรพอยเตอร์ที่จะนำ Task ไปควบคุมในขั้นตอนต่อไป
```

ภายใน Task ที่สร้างจะต้องมีการเรียกใช้ฟังก์ชันหน่วงเวลา vTaskDelay() เสมอเพื่อเปิดโอกาสให้ FreeRTOS ได้นำเวลาที่ถูกหน่วงไปทำงานใน Task อื่น ๆ ที่มีระดับความสำคัญรองลงมา หากภายใน Task ไม่มีการเรียกใช้ฟังก์ชันหน่วงเวลาอยู่เลยจะทำให้ Task อื่นไม่ทำงาน FreeRTOS เปิดให้สามารถสร้าง Task ได้ไม่จำกัด แต่จะถูกจำกัดจำนวนพื้นที่แรม โดยจะใช้ได้เท่าที่จอมีไว้เท่านั้น

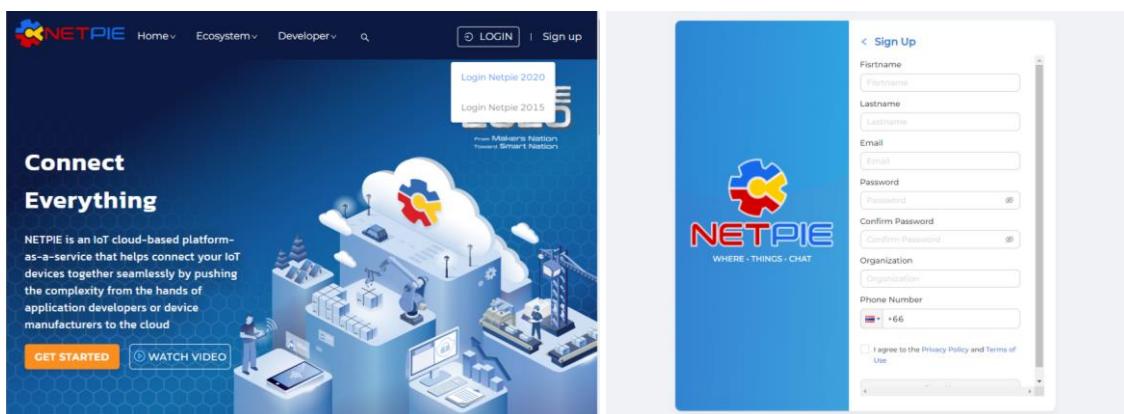
การทดลองที่ 0 | การลงทะเบียนบัญชีผู้ใช้ NETPIE และการติดตั้งไลบรารี PubSubClient บน Arduino IDE

สิ่งที่ต้องส่ง

ไม่มี

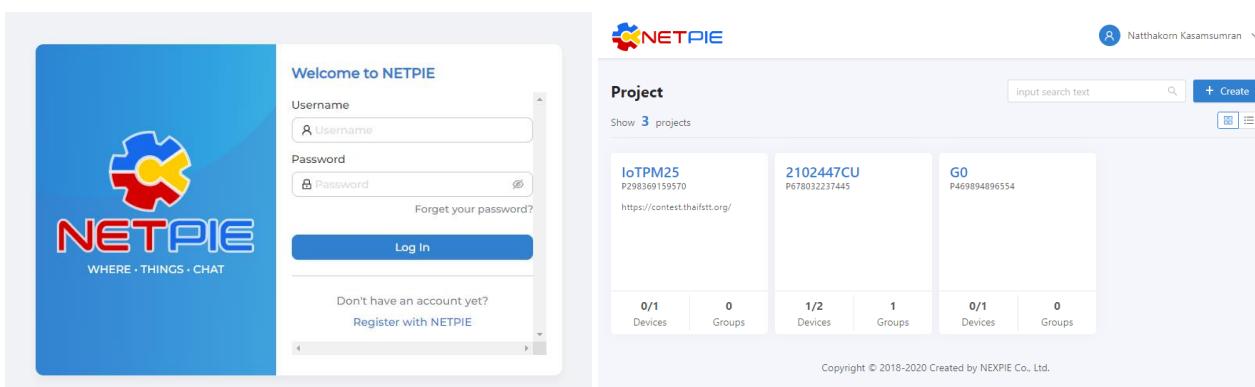
ขั้นตอนปฏิบัติ

- ไปที่เว็บไซต์ <https://netpie.io> จากนั้นเลือก Sign up ดังรูปที่ L0-1 กรอกข้อมูลให้ครบถ้วน (email กับ phone number ไม่สามารถแก้ไขภายหลังได้) จากนั้นรอรับ password ทาง email



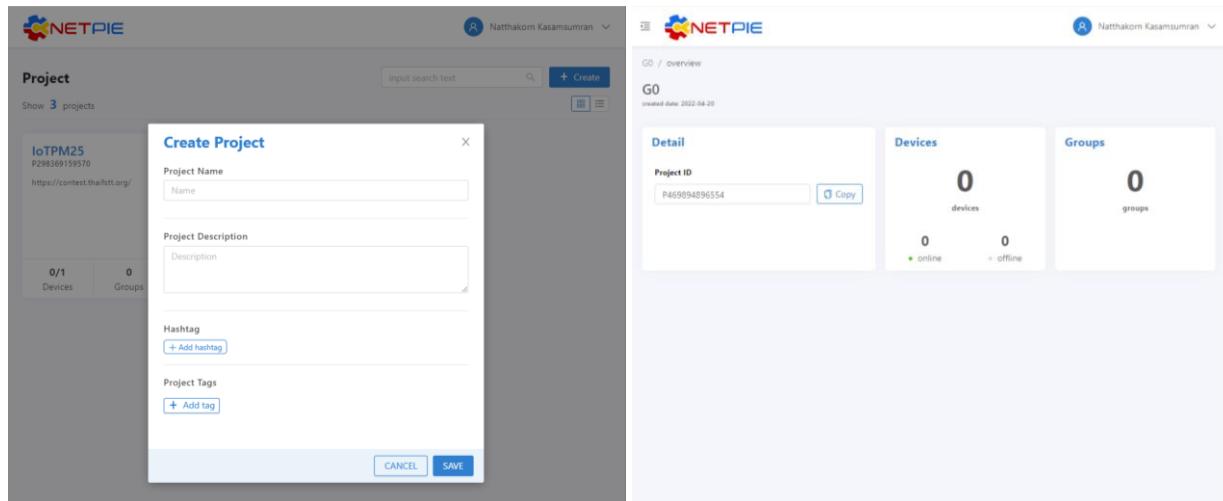
รูปที่ L0-1 หน้าของเว็บไซต์ netpie.io และหน้า Sign up

- ไปที่เว็บไซต์ <https://netpie.io> จากนั้นเลือก LOGIN เพื่อเข้าสู่ระบบ ดังรูปที่ L0-2 สามารถเปลี่ยน password ได้ โดยกดชื่อบัญชีด้านขวาบนแล้วเลือก profile



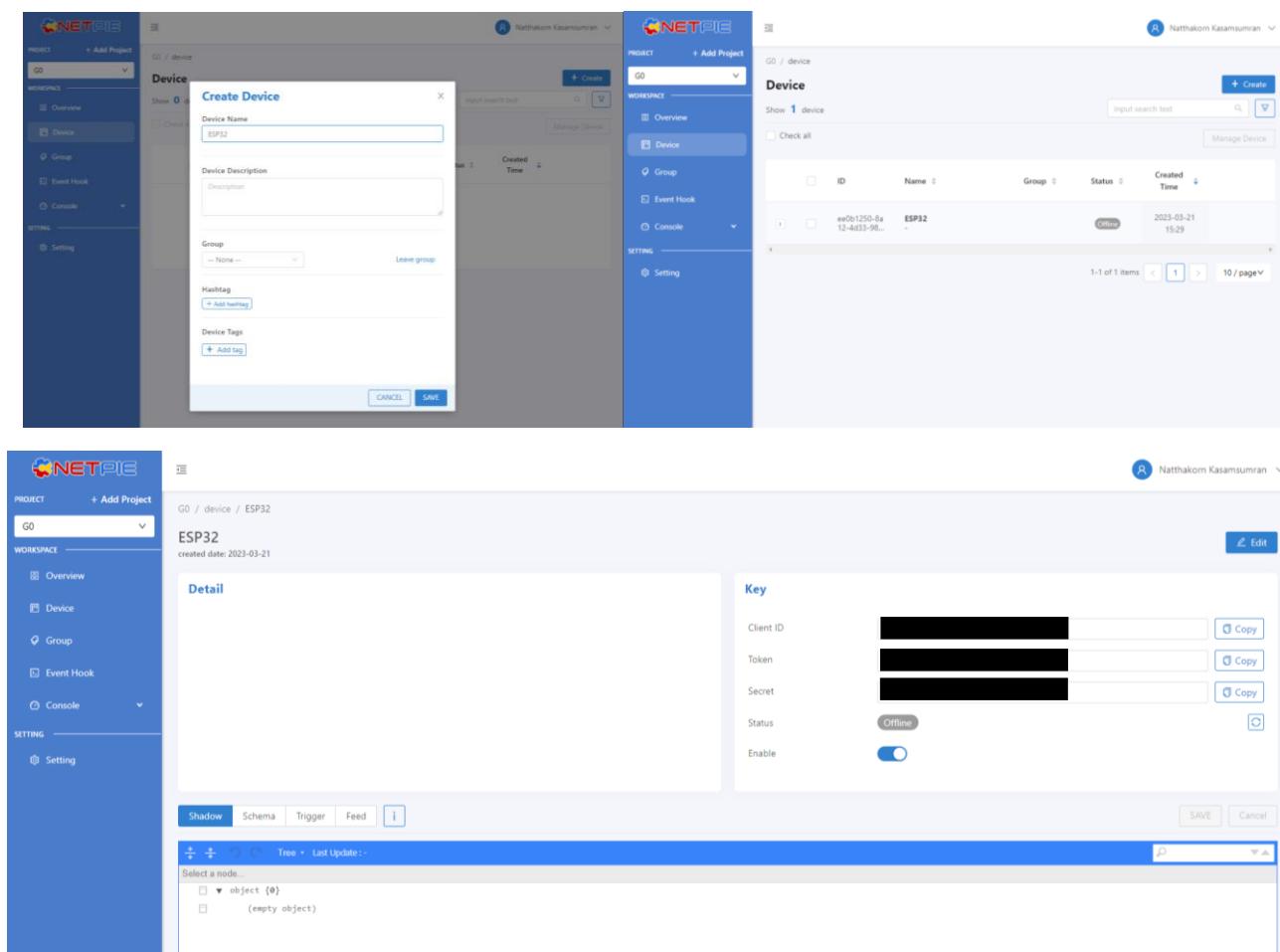
รูปที่ L0-2 หน้า LOGIN และหน้าหลักเมื่อเข้าสู่ระบบ

- กดปุ่ม “+” เพื่อสร้าง Project ตั้งชื่อ Project กดปุ่ม Create แล้วกดเลือก Project ที่สร้างไว้ ดังรูปที่ L0-3



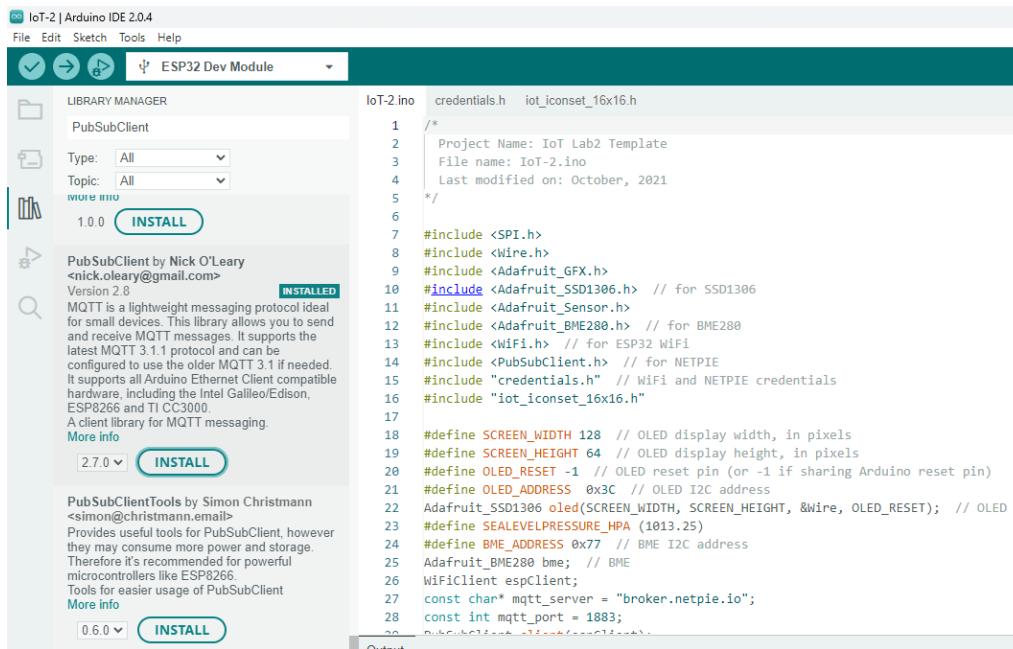
รูปที่ L0-3 การสร้าง Project บน NETPIE

4. กดเลือก Device List จากนั้นกดปุ่ม “+ Create” เพื่อสร้าง Device ตั้งชื่อ Device กดปุ่ม Create และกดเลือก Device ที่สร้างไว้ ดังรูปที่ L0-4 โดยแต่ละ Device จะมี Key คือ ClientID, Token, และ Secret ไว้ใช้สำหรับการเชื่อมต่อ



รูปที่ L0-4 การสร้าง Device ใน Project

5. ติดตั้งไลบรารี PubSubClient บน Arduino IDE โดยไปที่เมนู Sketch → Include Library → Manage Libraries ค้นหาคำว่า PubSubClient แล้วกดปุ่ม Install เพื่อติดตั้ง



รูปที่ L0-5 การติดตั้งไลบรารีด้วย Manage Libraries บน Arduino IDE 2.0



รูปที่ L0-6 การติดตั้งไลบรารีไลบรารี PubSubClient

*** แนะนำให้ใช้ PubSubClient by Nick O'Leary <nick.oleary@gmail.com> ***

เนื่องจากทดสอบแล้วสามารถสื่อสารกับ NETPIE 2020 ได้ปกติ หากนิสิตอยากรทดสอบ Library อื่นๆ ได้

☆ จบการทดลองที่ 0 ☆

การทดลองที่ 1 | การการเก็บและนำเสนอภาพข้อมูลด้วย NETPIE

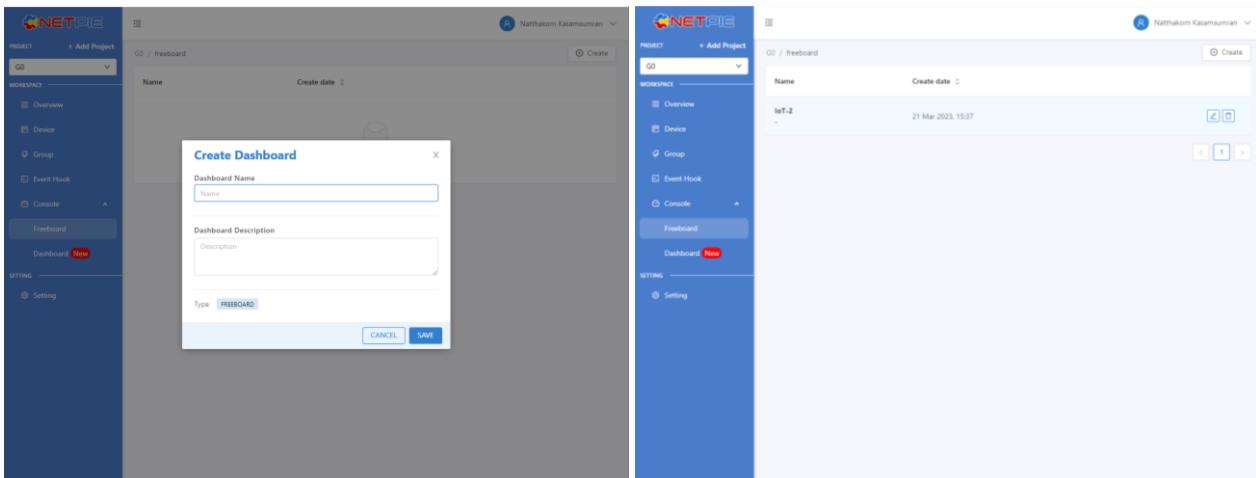
สิ่งที่ต้องส่ง

มีสิ่งที่ต้องส่ง 2 รายการ คือ

1. ไฟล์ IoT-2_1.zip โดยประกอบด้วยไฟล์ต่าง ๆ ที่แก้ไขยกเว้น credentials.h และ iot_iconset_16x16.h อย่างลีมแก้ไขไฟล์ schema.txt ด้วย โดยส่งใน attachment slot บน mycourseville
2. วิดีโอแสดงการทำงานของอุปกรณ์ โดยส่ง URL ของวิดีโອใน textbox บน mycourseville

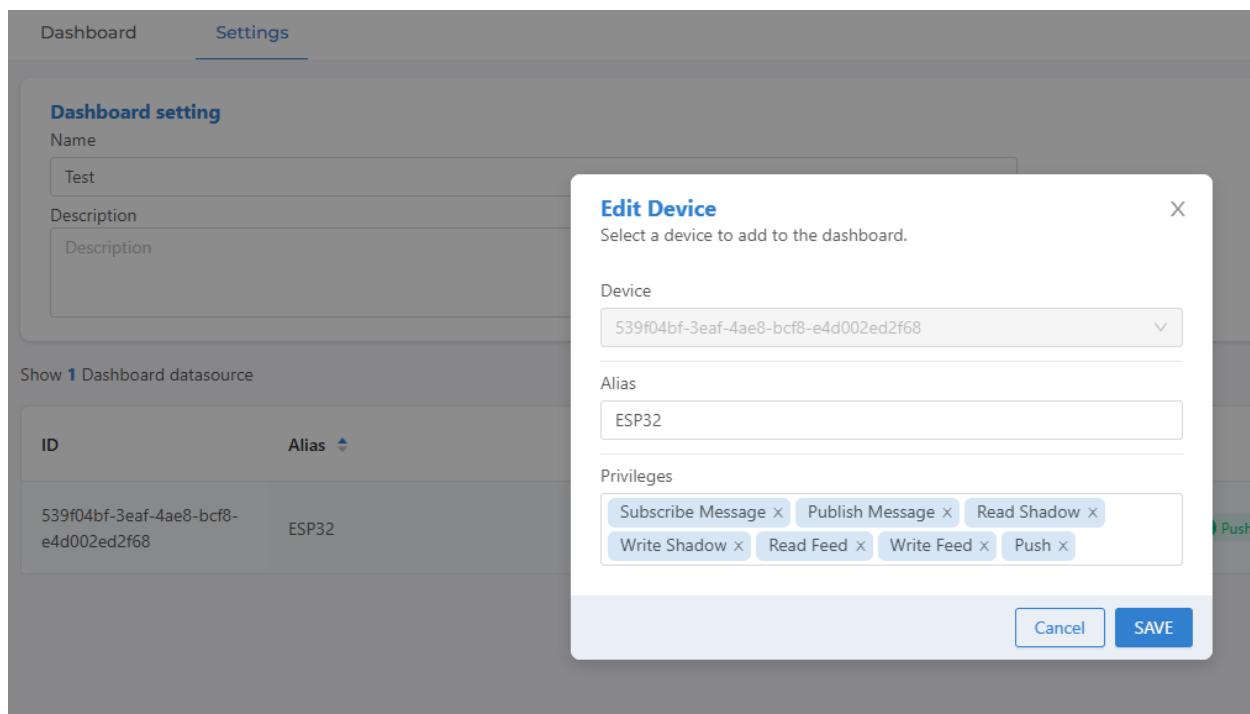
ขั้นตอนปฏิบัติ

1. ดาวน์โหลดไฟล์ IoT-2.zip โดยประกอบด้วย 6 ไฟล์ ได้แก่
 - a. Lab447-IoT-2.pdf ชีทแล็บนี้
 - b. IoT-2.ino โค้ดหลัก
 - c. credentials.h เก็บรหัสผ่าน WiFi และ NETPIE
 - d. iot_iconset_16x16.h เก็บไอคอน
 - e. schema.txt โค้ด Schema
 - f. MTask/Mtask.ino โค้ดตัวอย่าง Multitasking โดยใช้ FreeRTOS
2. ให้นิสิตต่อโมดูล OLED และโมดูล BME280 เข้ากับบอร์ด IOXESP32+ ผ่านทาง I2C และแก้ไข I2C Address ในไฟล์ IoT-2.ino (แนะนำควรใช้ I2C scanner ในการหา Address)
3. จากนั้นแก้ไข SSID และ password ที่มาจากการตั้งค่า WiFi Hotspot ในส่วนของ /* WiFi */ ของนิสิต และใน /*NETPIE*// ให้นำ ClientID, Token, และ Secret จากรูปที่ L0-4 มาใส่ในตัวแปรในไฟล์ credentials.h แนะนำให้ใช้ WiFi Hotspot เพื่อความสะดวกในการทดลองเปิดปิด WiFi (ถ้าเป็น iPhone ให้เปิด Maximum compatibility)
4. จากนั้นเข้าไปที่ Schema ของ Device ใน NETPIE เปลี่ยนมุมมองจาก Tree เป็น Code จากนั้น Copy โค้ดในไฟล์ schema.txt ใส่ลงไปแล้วกด Save
5. Upload โค้ดในไฟล์ IoT-2.ino ไปยังบอร์ด IOXESP32+ แล้วดูผลที่จอ OLED และส่วน Shadow กับ Feed ของ Device ใน NETPIE
6. กดเลือก Dashboard จากนั้นกดปุ่ม “+ Create” เพื่อสร้าง Dashboard ตั้งชื่อ Dashboard กดปุ่ม Save และกดเลือก Dashboard ที่สร้างไว้ ตั้งรูปที่ L1-1



รูปที่ L1-1 การสร้าง Dashboard ใน Project ของนิสิต

- เมื่อนิสิตเข้ามาในหน้า Dashboard และให้เลือก Setting จากนั้นกดปุ่ม “+ Add device” และเลือก Device ที่ได้สร้างไว้ก่อนหน้านี้ ส่วนของ Privileges ให้เลือก Subscribe Message, Publish Message, Read Shadow, Write Shadow, Read Feed, Write Feed และ Push และจึงกด Save ดังรูปที่ L1-2



รูปที่ L1-2 การกำหนดสิทธิ์การเข้าถึงของอุปกรณ์

- กลับมาที่หน้า Dashboard ให้กดปุ่ม Edit และจึงกดปุ่ม “+ Add Panel” นิสิตสามารถปรับขนาดความกว้างของ Panel ได้โดยการกดไอคอนรูปประแจ (Config) และกำหนดชื่อ Panel ใน Title และกำหนดความกว้างด้วยการเลือน Columns จากนั้นกดปุ่ม Done และจึงกดปุ่ม Save (คำเตือน ให้กดปุ่ม Save ทุกครั้งที่มีการแก้ไข Dashboard ไม่เช่นนั้นแล้วสิ่งที่ทำจะไม่ถูกบันทึก)

- ■ ■
9. จากนั้นให้ทำการเพิ่ม Widget โดยการกดไอคอน + เลือก Type = Gauge ให้กรอกรายละเอียดดังนี้ Title = Altitude, ให้นิสิตกดปุ่ม “+ Device” ใน Value แล้วจะได้ #["ชื่อ Device ของนิสิต"] แล้วกรอกให้เป็น Value = #["ชื่อ Device ของนิสิต"]["shadow"]["altitude"].toFixed(2), Units = m, Minimum = 0, Maximum = 200 ปรับ Human friendly number เป็น No แล้วกด Done แล้วจึงกด Save
 10. ให้นิสิตสร้าง Panel เพิ่ม กำหนดให้ Columns = 3 จากนั้นให้ทำการเพิ่ม Widget โดยการกดไอคอน + เลือก Type = Chart, Data Source = #["ชื่อ Device ของนิสิต"]["feed"], Filter = altitude แล้วกด Done
 11. กด Save ที่ Dashboard
 12. ให้นิสิตแก้ไขโค้ดในไฟล์ IoT-2.ino และ Schema ของ Device เพื่อรับอุณหภูมิ (Temperature) และความชื้น (Humidity) หน่วยเป็น %Rh (relative humidity) ของโมดูล BME280 เพิ่มเติม โดยทำการอ่านค่าแสดงไอคอนและค่าบนจอ OLED (อุณหภูมิหน่วยเป็นองศาเซลเซียส °C) ส่งค่าไปยัง NETPIE และแสดงผลบน Dashboard (อุณหภูมิหน่วยเป็นองศาฟาเรนไฮต์ °F) ทั้งแบบ Gauge และ Chart และแสดงคล้ายกับ Feed ในหน้า Device ของ NETPIE
 13. หลังจากเก็บค่ามาประมาณสัก 10 นาทีเป็นอย่างน้อย บันทึกวิดีโอการแสดงผลบนจอ OLED และแสดงผลบน Chart กับ Feed และการแสดงผลบนจอ OLED เมื่อปิด WiFi ไปสัก 20 วินาที และเปิด WiFi Hotspot กลับซึ่งชุดคำสั่งที่นิสิตปรับปรุงเพิ่มจะต้องสามารถเชื่อมต่อ WiFi ได้อย่างอัตโนมัติ และสามารถส่งค่าไปยัง NETPIE ได้หลังจากเชื่อมต่ออินเทอร์เน็ตได้โดยสมบูรณ์

☆ จบการทดลองที่ 1 ☆

*** ถ้าจอ OLED แสดงผลผิดเพี้ยง ให้ลองเอาไปต่อขาด้าน SPI ของโมดูล BME280 ไว้ ***

การทดลองที่ 2 | การประยุกต์ใช้ Multitasking

สิ่งที่ต้องส่ง

มีสิ่งที่ต้องส่ง 2 รายการ คือ

- ไฟล์ IoT-2_2.zip โดยประกอบด้วยไฟล์ต่าง ๆ ที่แก้ไขยกเว้น credentials.h และ iot_iconset_16x16.h โดยส่งใน attachment slot บน mycourseville
- วิดีโอแสดงการทำงานของอุปกรณ์ โดยส่ง URL ของวิดีโอด้วย textbox บน mycourseville

ขั้นตอนปฏิบัติ

- ให้นิสิตทดลอง Upload โค้ดในไฟล์ MTask.ino ไปยังบอร์ด IOXESP32+ แล้วสังเกตการทำงานของหน้าจอ OLED และ LED บนบอร์ด IOXESP32+ (LED onboard at GPIO 5)
- ให้นิสิตแก้ไขชุดคำสั่งจากการทดลองที่ 1 ให้เป็นแบบ Multitasking โดยใช้ FreeRTOS กำหนดให้
 - Task 1 ทำการอ่านค่าความดัน (Pressure) และความสูง (Altitude) และส่งค่าความดัน ณ ขณะนั้น และค่าความสูงเท่ากับ 0 ไปยัง NETPIE ทุก ๆ 3 วินาที โดยตรวจสอบการเชื่อมต่อ NETPIE ว่ายังอยู่ก่อนส่งค่าไปยัง NETPIE
 - Task 2 ทำเช่นเดียวกับ Task1 แต่กับค่าอุณหภูมิ (Temperature) ทุก ๆ 4 วินาที
 - Task 3 ทำเช่นเดียวกับ Task1 แต่กับค่าความชื้น (Humidity) ทุก ๆ 5 วินาที
 - Task 4 ตรวจสอบการเชื่อมต่อ NETPIE และ WiFi ทุก ๆ 7 วินาที
 - Task 5 แสดงไอคอนและค่าต่าง ๆ และการเชื่อมต่อ WiFi ทุก ๆ 1 วินาที
- บันทึกการแสดงผลบนจอ OLED และส่วน Feed ของ Device เมื่อปิด WiFi Hotspot ไปสัก 20 วินาที และเปิด WiFi Hotspot ให้ IOXESP32+ เชื่อมต่ออินเตอร์เน็ต และสังเกตลักษณะ Feed

☆ จบการทดลองที่ 2 ☆

2102447 ปฏิบัติการวิศวกรรมอิเล็กทรอนิกส์
Electronic Engineering Laboratory

ผู้สอนประจำวิชา ผศ.ดร.สุรีย์ พุ่มรินทร์ และ อ.ดร.ณรงค์ ปันธนาธรรม
ผู้สอนปฏิบัติการ ณัทกร เกษมสำราญ (ภาคการศึกษาต้น 2567)



การควบคุมและทริกเกอร์อุปกรณ์ IoT ด้วย NETPIE

(IoT Device Control and Trigger via NETPIE)

วัสดุและอุปกรณ์

- คอมพิวเตอร์ติดตั้งแอปพลิเคชัน Arduino IDE 2.3.3
- บอร์ด IOXESP32+ และบอร์ดขยาย BASE32
- โมดูลจอ OLED 0.96" 128x64 pixel (I²C protocol)
- สาย USB-A to USB-C, สายไฟ Jumper
- โมดูลวัดอุณหภูมิ ความชื้น และความดัน BME280 sensor (I²C protocol)
- โมดูลวัดความเข้มแสง TSL2561 Lux sensor (I²C protocol)
- โมดูลตรวจจับการเคลื่อนไหว PIR Motion sensor (Digital output)
- โมดูลโปรดวัดอุณหภูมิดิจิทัล DS18B20 (Digital 1-Wire)
- อุปกรณ์อิเล็กทรอนิกส์อื่นๆ

"IoT Learning Kit" 2102447 Electronics Engineering Laboratory

Components list :

1.) IOXESP32+ V1.0 Embedded Board	1 each
2.) BASE32 Pinout Expansion Board	1 each
3.) OLED Display 128x64 0.96"	1 each
4.) OLED Housing Bracket	1 each
5.) Breadboard / Protoboard	1 each
6.) 4 SPST Switch Module	1 each
7.) 8mm LED (R/Y/G) Module	1 each
8.) Potentiometer Module	1 each
9.) USB-C to USB-A Cable	1 each
10.) Jumper Wire (Male to Male)	10 each
11.) Jumper Wire (Male to Female)	10 each
12.) Jumper Wire (Female to Female)	10 each

Sensor selection list :

TSL2561 Lux sensor	BME280 (Temp, Humid, Pressure)
PIR Motion Sensor	DS18B20 with Module

IOXESP32+ PINOUT

docs.ioxesp32.com/ioxesp32+

หมายเหตุ : ในกล่องไม่มีรายการที่ 5.) Breadboard / Protoboard ให้

การทดลองที่ 1 | การควบคุมและทริกเกอร์อุปกรณ์ด้วย NETPIE

สิ่งที่ต้องส่ง

มีสิ่งที่ต้องส่ง 1 รายการ คือ

- วีดิโอแสดงการทำงานของอุปกรณ์ โดยส่ง URL ของวีดิโອใน textbox บน mycourseville

ขั้นตอนปฏิบัติ

- ดาวน์โหลดไฟล์ที่ IoT-3.zip โดยประกอบด้วย 7 ไฟล์ ได้แก่
 - Lab447-IoT-3.pdf ชีทแล็บนี้
 - IoT-3.ino โค้ดหลัก
 - credentials.h เก็บรหัสผ่าน WiFi และ NETPIE
 - iot_iconset_16x16.h เก็บไอคอน
 - schema.txt โค้ด Schema
 - trigger.txt โค้ด Trigger
 - eventhooks.txt โค้ด Event Hooks
- ต่อโมดูล OLED และโมดูล BME280 เข้ากับบอร์ด ESP32 ผ่านทาง I2C และแก้ไข I2C Address ในไฟล์ IoT-3.ino หากไม่ทราบให้ใช้ I2C Scanner เพื่อหา Address ของอุปกรณ์ก่อน
- แก้ไขรหัส WiFi และ NETPIE ในไฟล์ credentials.h และนำให้ใช้ WiFi Hotspot เพื่อความสะดวกในการทดลองเปิดปิด WiFi เพื่อตรวจสอบว่า Source code สามารถ Reconnect ได้หรือไม่
- ให้นิสิตไปที่เว็บไซต์ <https://netpie.io> จากนั้นเลือก Login เพื่อเข้าสู่ระบบเมื่อ้อนการทดลองก่อนหน้า
- เลือกที่ Device และกดเข้าไปที่ Device ที่สร้างไว้แล้ว เลือก Schema ของ Device ใน NETPIE เปลี่ยน มุมมองจาก Tree เป็น Code จากนั้น Copy โค้ดในไฟล์ schema.txt ใส่ลงไปแล้วกด Save
- Upload โค้ดในไฟล์ IoT-3.ino ไปยังบอร์ด ESP32 แล้วดูผลลัพธ์ที่จอ OLED และส่วน Shadow ของ Device ใน NETPIE หากบอร์ด ESP32 สามารถเชื่อมต่อ WiFi & NETPIE จะแสดงค่าใน Shadow
- ให้นิสิตเข้าที่ NETPIE 2020 กดเลือก Project ที่นิสิตสร้างไว้จากนั้นไปที่ Console และกดเลือก Dashboard จากนั้นกดปุ่ม “+ Create” เพื่อสร้าง Dashboard ตั้งชื่อ Dashboard Name (และกำหนด Description ถ้ามี) กดปุ่ม Save และกดเลือก Dashboard ที่สร้างไว้
- เมื่อนิสิตเข้ามาในหน้า Dashboard แล้วให้เลือก Setting จากนั้นกดปุ่ม “+ Add device” และเลือก Device ที่ได้สร้างไว้ก่อนหน้านี้ ส่วนของ Privileges ให้เลือก Subscribe Message, Publish Message, Read Shadow, Write Shadow, Read Feed, Write Feed และ Push และจึงกด Save

9. กลับมาที่หน้า Dashboard ให้กดปุ่ม Edit แล้วจึงกดปุ่ม “+ Add Panel” นิสิตสามารถปรับขนาดความกว้างของ Panel ได้โดยการกดไอคอนรูปประแจง (Config) และกำหนดชื่อ Panel ใน Title และกำหนดความกว้างด้วยการเลื่อน Columns จากนั้นกดปุ่ม Done และจึงกดปุ่ม Save (คำเตือน ให้กดปุ่ม Save ทุกครั้งที่มีการแก้ไข Dashboard ไม่เช่นนั้นแล้วสิ่งที่ทำจะไม่ถูกบันทึก)
10. จากนั้นทำการเพิ่ม Widget ให้นิสิตกดปุ่ม “+” (ปุ่มลำดับแรกใน Panel) จะปรากฏหน้าต่าง Widget ให้เลือก Type = Indicator Light และกำหนดค่าต่างๆ ดังนี้ Title กำหนดชื่อว่า LED Status ส่วนของ LIGHT ON COLOR กำหนดเป็น #["ชื่อ Device ของนิสิต"]["msg"]["led"]=="on" และ LIGHT OFF COLOR กำหนดเป็น #["ชื่อ Device ของนิสิต"]["msg"]["led"]=="off" ถัดมา ON TEXT กำหนดว่า ON และ OFF TEXT กำหนด OFF จากนั้นจึงกดปุ่ม Done และ Save ตามลำดับ
11. จากนั้นให้สร้างปุ่มกดเพิ่มเติม ให้นิสิตกดปุ่ม “+” (ปุ่มลำดับแรกใน Panel) จะปรากฏหน้าต่าง Widget ให้เลือก Type = Button และกำหนดค่าต่างๆ ดังนี้ BUTTON CAPTION เป็น ON ถัดมา LABEL เป็น LED ON สีของปุ่ม BUTTON COLOR เป็น Green และ ONCLICK ACTION กำหนดเป็น #["ชื่อ Device ของนิสิต"].publishMsg("led","on") จากนั้นจึงกดปุ่ม Done และ Save ตามลำดับ
12. ให้นิสิตกดปุ่ม “+” (ปุ่มลำดับแรกใน Panel) จะปรากฏหน้าต่าง Widget ให้เลือก Type เป็น Button และกำหนดค่าต่างๆ ดังนี้ BUTTON CAPTION เป็น OFF ถัดมา LABEL เป็น LED OFF สีของปุ่ม BUTTON COLOR เป็น Red และ ONCLICK ACTION กำหนดเป็น #["ชื่อ Device ของนิสิต"].publishMsg("led","off") จากนั้นจึงกดปุ่ม Done และ Save ตามลำดับ
13. เมื่อกด Save ที่ Dashboard และทดสอบกดปุ่ม ON และ OFF สังเกตสัญลักษณ์หลอดไฟบนหน้าจอ OLED และจะมีข้อความปรากฏเมื่อสถานะของหลอด LED on board หากนิสิตกำหนดการตั้งค่าต่างๆ ถูกต้องสมบูรณ์ ทั้งหน้าการแสดงผลบน Dashboard, หน้าจอ OLED และ LED on board จะทำงานสัมพันธ์กัน
- ข้อควรระวัง!** ใน Source Code IoT-3.ino สำหรับ LED_BUILTIN 5 ที่ติดมากับบอร์ด IOXESP32+ จะเป็นอุปกรณ์แบบ Active LOW นั้นหมายความว่าความถ้ามีการจ่าย Logic High ที่ขา 5 หลอดไฟจะดับ หากเป็น LOW หลอดไฟจะติดสว่าง ถ้านิสิตต้องการต่อหลอด LED ที่ GPIO อื่นๆ และอยากรทำเป็น Active HIGH คือจ่าย HIGH หลอดไฟติด และ LOW หลอดไฟดับ ตรงคำสั่ง digitalWrite(LED_BUILTIN, 0); แก้เป็น digitalWrite(27, 1); กรณีที่นิสิตต่อหลอด LED เข้ากับขาที่ 27 และ Common ground
14. ไปที่ <https://trial-api.netpie.io/> กด Authorize ใส่ Username (ClientID) และ Password (Token) และกด Authorize และกด Close ขั้นตอนนี้จะเป็นการทดสอบ RESTful API ของ NETPIE นิสิตสามารถศึกษาเพิ่มเติมจาก <https://docs.netpie.io/device-api.html>

15. กด PUT /device/message และกด Try it out กำหนด topic = led, Request body = on (or off) และกด Execute และดูผลเช่นเดิม หากส่งคำสั่งสำเร็จจะขึ้น 200 OK และควบคุม LED on board ได้ ดังรูป A1-1
16. กดเลือก Event Hooks จากนั้นกดปุ่ม “+ Create” เพื่อสร้าง Event Hook ตั้งชื่อว่า LED_ON กดปุ่ม Create และกดเลือก Event Hook ที่สร้างไว้ จากนั้น Copy โค้ดในไฟล์ eventhooks.txt ส่วน LED_ON ใส่ลงไป
17. นำ Authorization ใน Responses => Curl ดังรูป A1-2 จากการทดลองในข้อที่ 15 มาใช้แทนแล้วกด Save
18. ให้นิสิตสร้าง Device ตัวที่ 2 ตั้งชื่อว่า MQTTbox ในหน้า Device ของ NETPIE และให้ Copy ทั้ง Client ID, Token และ Secret เก็บไว้ เพื่อนำมาใช้ในการทดลองผ่านโปรแกรม MQTTbox และเลือกที่ Group ทำการจับกลุ่ม Device ทั้ง ESP32 และ MQTTbox เข้าด้วยกันโดยการกดปุ่ม “+ Create” จากนั้นเข้าไปด้านในเลือก Manage Device และทำการตีกถูก Device ทั้ง 2 ตัวให้อยู่กลุ่มเดียวกัน และจึงกด Save
19. จากนั้นไปที่ <https://netpie.io/guide> และเลื่อนหาหัวข้อ Must Download => Useful Programs => MQTTbox สามารถใช้ได้ทั้ง Windows และ MacOS เมื่อดาวโหลดเสร็จแล้วให้เปิดโปรแกรมดังกล่าว
20. เมื่อโปรแกรม MQTTbox ทำงานแล้วให้นิสิตเลือก Create MQTT Client จะปรากฏรูปที่ A1-3 ให้ทำการกำหนดค่าดังนี้ MQTT Client Name = “MQTTbox2ESP32” MQTT Client Id = “NETPIE Client ID”, Protocol = “mqtt/tcp”, Host = “broker.netpie.io”, Username = “NETPIE Token”, Password = “NETPIE Secret”, Append timestamp to MQTT client id? = No และกด Save
21. ไปที่หน้าต่างฝั่งซ้ายมือ Topic to subscribe กำหนดค่าดังนี้ Topic to publish = “@msg/led”, Payload Type = “Strings/JSON/XML/Characters”, Payload = on (หรือ off) และกด Publish ดูผลหลอดไฟที่บอร์ด ESP32 บนหน้าจอ OLED และ Dashboard จะต้องทำงานสัมพันธ์กัน
22. ไปที่หน้าต่างฝั่งขวา Topic to subscribe = “@msg/led” และกด Subscribe และกดเปิดปิด LED จากทั้ง Dashboard และ MQTTBox ดูผลที่ Subscriber ใน MQTTBox
23. ไปที่ส่วน Trigger ของ Device ใน NETPIE จากนั้น Copy โค้ดในไฟล์ trigger.txt ใส่ลงไปแล้วกด Save (สำหรับ NETPIE 2020 ก่อนปี 2565 แต่ถ้าหลังปี 2566 จะต้องสร้าง Trigger เอง โดยการกด + Add trigger จากนั้นตั้งชื่อ Trigger Title เช่น LED_ON กำหนด Event เป็น SHADOW.UPDATED ในส่วนของ Under conditions ให้กำหนดตามไฟล์ trigger.txt เช่น \$NEW.altitude > \$PREV.altitude ส่วน Action to ให้กำหนดไปยัง Event hook ที่สร้างขึ้น เช่น LED_ON และกด Add เมื่อนิสิตสร้างทุก Trigger ครบแล้วให้กด Enabled all นิสิตสามารถศึกษาเพิ่มเติมจาก <https://docs.netpie.io/device-config.html>
24. ทำเช่นข้อ 15-16 แต่เปลี่ยนจาก LED_ON เป็น LED_OFF จากนั้นดูผลเช่นเดิม

☆ จบการทดลองที่ 1 ☆

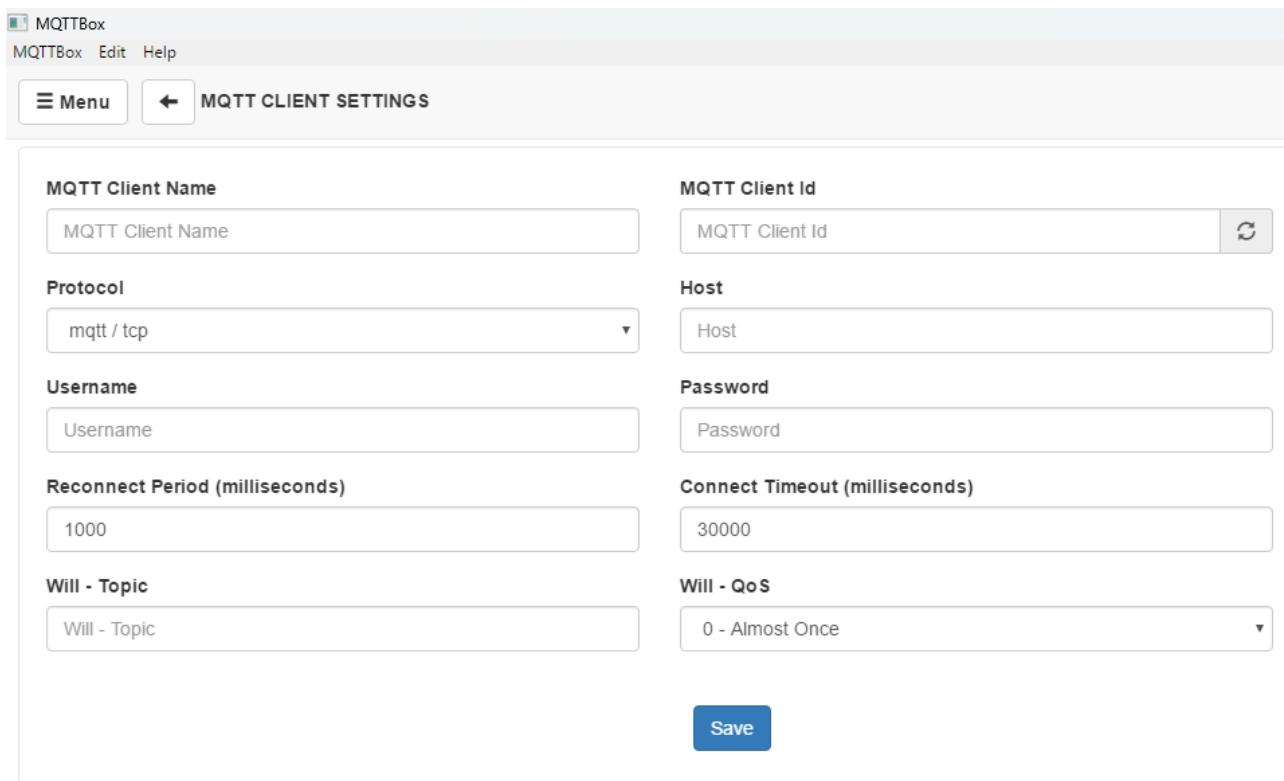
The screenshot shows the NETPIE Device API documentation. It features a navigation bar at the top with the title "NETPIE Device API 0.1.0 OAS3". Below the title, there's a note: "ทดลองและศึกษาการใช้งาน RESTful API ของ NETPIE Platform". The main content area is titled "Device API". It contains two examples:

- GET /device/status**: การขอสถานะเมื่อมอง Platform ของ Device (Device Status). This example shows a single parameter: "topic" with value "led".
- PUT /device/message**: Publish ข้อความ ไปที่ Topic ต่างๆ. This example shows a parameter "topic" with value "led" and a request body "off".

รูป A1-1 หน้าเว็บ NETPIE Device API สำหรับทดสอบการทำงานผ่าน RESTful API

The screenshot shows two windows side-by-side. On the left is the "NETPIE User Dashboard" showing a project named "2102447CU" with an "Event Hook" named "LED_ON". The "Code" tab of the event hook shows a curl command. On the right is the "NETPIE Device API" interface showing the "PUT /device/message" endpoint. The "Responses" section displays the curl command and its execution results, including the response code 200 and the response body "OK".

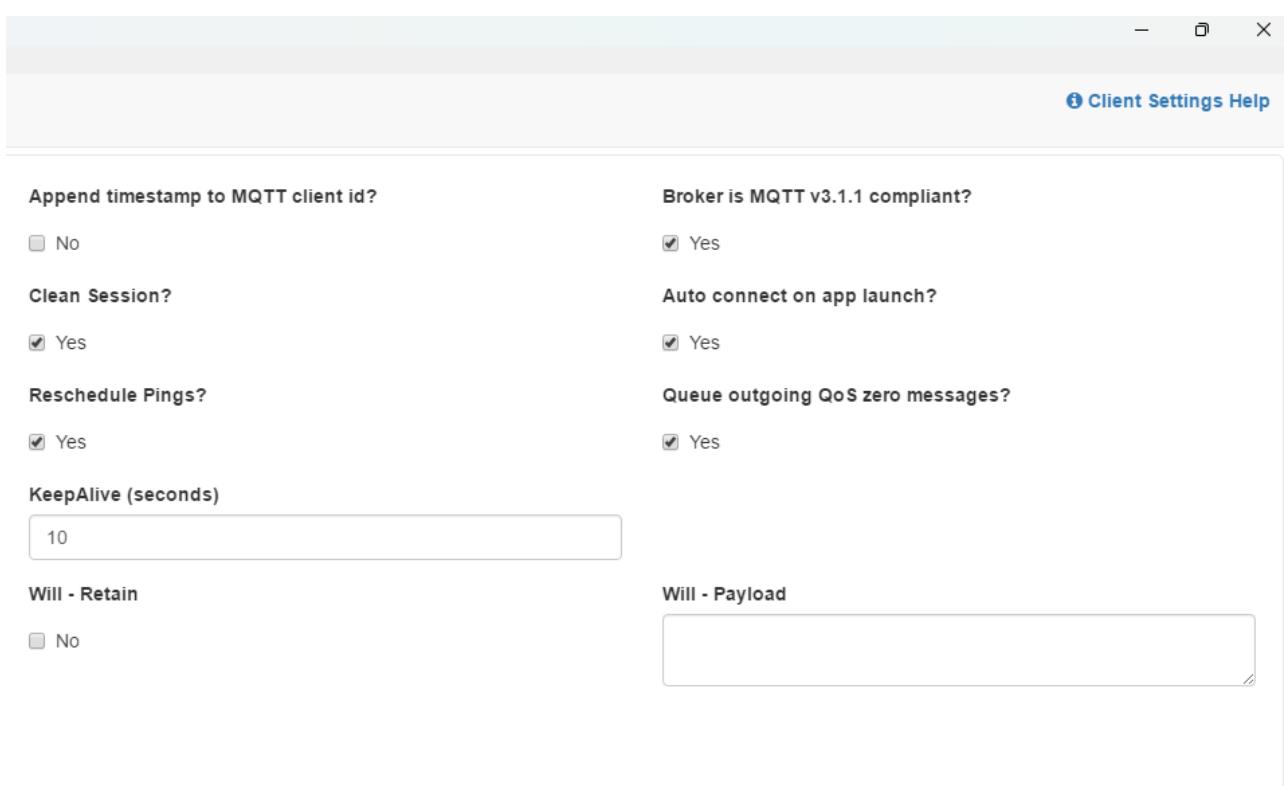
รูป A1-2 วิธีคัดลอก Authorization ในรูปแบบ BASE64ENCODE จาก Curl command ในหน้าเว็บ NETPIE Device API ซึ่งปกติแล้วนิสิตจะต้องนำ ClientID:Token ไปแปลงด้วย Base64 Encoder



The screenshot shows the MQTTClient Settings page of the MQTTBox application. The top navigation bar includes 'MQTTBox', 'Edit', and 'Help' buttons, along with a 'Menu' button and a back arrow. The main title is 'MQTT CLIENT SETTINGS'. The form contains the following fields:

- MQTT Client Name:** A text input field containing 'MQTT Client Name'.
- MQTT Client Id:** A text input field containing 'MQTT Client Id' with a refresh icon.
- Protocol:** A dropdown menu set to 'mqtt / tcp'.
- Host:** A text input field containing 'Host'.
- Username:** A text input field containing 'Username'.
- Password:** A text input field containing 'Password'.
- Reconnect Period (milliseconds):** An input field containing '1000'.
- Connect Timeout (milliseconds):** An input field containing '30000'.
- Will - Topic:** A text input field containing 'Will - Topic'.
- Will - QoS:** A dropdown menu set to '0 - Almost Once'.

A blue 'Save' button is located at the bottom center of the form. Below the form is a status bar with icons for minimize, maximize, and close, and a 'Client Settings Help' link.



This screenshot shows the 'Advanced Options' section of the MQTTBox settings. It includes the following configuration items:

- Append timestamp to MQTT client id?**: An unchecked checkbox labeled 'No'.
- Broker is MQTT v3.1.1 compliant?**: A checked checkbox labeled 'Yes'.
- Clean Session?**: A checked checkbox labeled 'Yes'.
- Auto connect on app launch?**: A checked checkbox labeled 'Yes'.
- Reschedule Pings?**: A checked checkbox labeled 'Yes'.
- Queue outgoing QoS zero messages?**: A checked checkbox labeled 'Yes'.
- KeepAlive (seconds):** An input field containing '10'.
- Will - Retain:** An unchecked checkbox labeled 'No'.
- Will - Payload:** A large text input area.

รูป A1-3 หน้าต่างในโปรแกรม MQTTbox ในการกำหนดค่าต่างๆ เพื่อให้เชื่อมต่อกับ NETPIE

การทดลองที่ 2 | การประยุกต์ใช้งาน NETPIE

สิ่งที่ต้องส่ง

มีสิ่งที่ต้องส่ง 2 รายการ คือ

- ไฟล์ IoT-3_Report.zip โดยประกอบด้วยไฟล์รายงานและไฟล์ต่าง ๆ ที่แก้ไขยกเว้น credentials.h และ iot_iconset_16x16.h โดยส่งใน attachment slot บน mycourseville
- วิดีโอแสดงการทำงานของอุปกรณ์ โดยส่ง URL ของวิดีโอด้วย textbox บน mycourseville

ขั้นตอนปฏิบัติ

- ให้นิสิตทดลองประยุกต์ใช้งาน NETPIE จากอุปกรณ์ที่ได้รับ เช่น
 - (ตัวอย่าง) ใช้ตัวรับรู้ 1 ตัวเก็บข้อมูลเพื่อนำมาแปลความหมาย (data interpretation) เช่น หากตรวจพบสิ่งของอุณหภูมิเมื่อเปิดปิดเครื่องปรับอากาศในห้อง หากความถี่การกินอาหารของสัตว์เลี้ยง โดยตรวจจับการเคลื่อนไหว
 - (ตัวอย่าง) ใช้ตัวรับรู้ตั้งแต่ 2 ตัวที่ต่างชนิดกันเก็บข้อมูลเพื่อหาความสัมพันธ์ (data correlation) เช่น หากความสัมพันธ์ระหว่างความสว่างของแสงกับเบอร์เซ็นต์ความชื้น
 - (ตัวอย่าง) ใช้ตัวรับรู้ตั้งแต่ 2 ตัวที่ชนิดเดียวกันเก็บข้อมูลเพื่อเปรียบเทียบ (data comparison) เช่น เปรียบเทียบอุณหภูมิจุดที่โดนแಡดและจุดที่ไม่โดนแಡด เปรียบเทียบความสว่างในแต่ละพื้นที่ ในการนี้ นิสิตอาจสร้าง Device อีกตัวหนึ่ง ส่ง Key ของ Device และโค้ดให้เพื่อนอีกกลุ่มที่อยู่คนละที่ซึ่งจะ ใส่รหัส WiFi และ Upload ลงบอร์ด ESP32 เพื่อเก็บข้อมูลมาเปรียบเทียบได้
 - ประยุกต์การควบคุมและทริกเกอร์อุปกรณ์เพื่อสร้างเป็นระบบบอตโนมัติขึ้นส่งผ่าน Line notify
- เขียนรายงานการทดลอง ประกอบด้วย
 - ที่มาและความสำคัญ
 - วัตถุประสงค์ อย่างน้อย 3 ข้อ
 - ประโยชน์ที่คาดว่าจะได้รับ
 - ขั้นตอนการทำงาน, Flow chart/Blackbox และ Schematic diagram
 - อภิรายผล และสรุปผล
 - แหล่งอ้างอิง
 - ภาคผนวก อธิบาย Source code

☆ จบการทดลองที่ 2 ☆

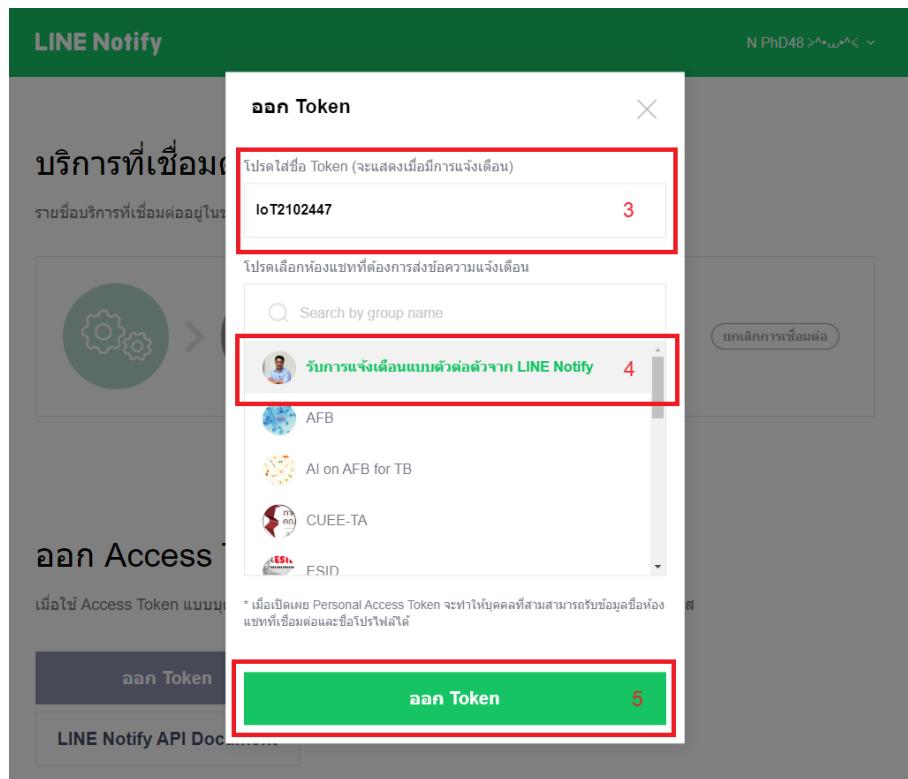
The screenshot shows the LINE Notify website interface. At the top, there's a green header bar with the text "LINE Notify". On the right side of the header, it says "N PhD48 >^•ω•^<". Below the header, there's a red-bordered box containing the message details. The message is from "Test" to "N PhD48" at 2023.11.13 16:37. The subject of the message is "การจัดการบริการที่ลงทะเบียน" (Registration Management Service). The message body contains the text "ออกจากระบบ" (Logout). The main content area below the header shows a message card with a gear icon, a recipient profile picture, and the message text.

ออก Access Token (สำหรับผู้พัฒนา)

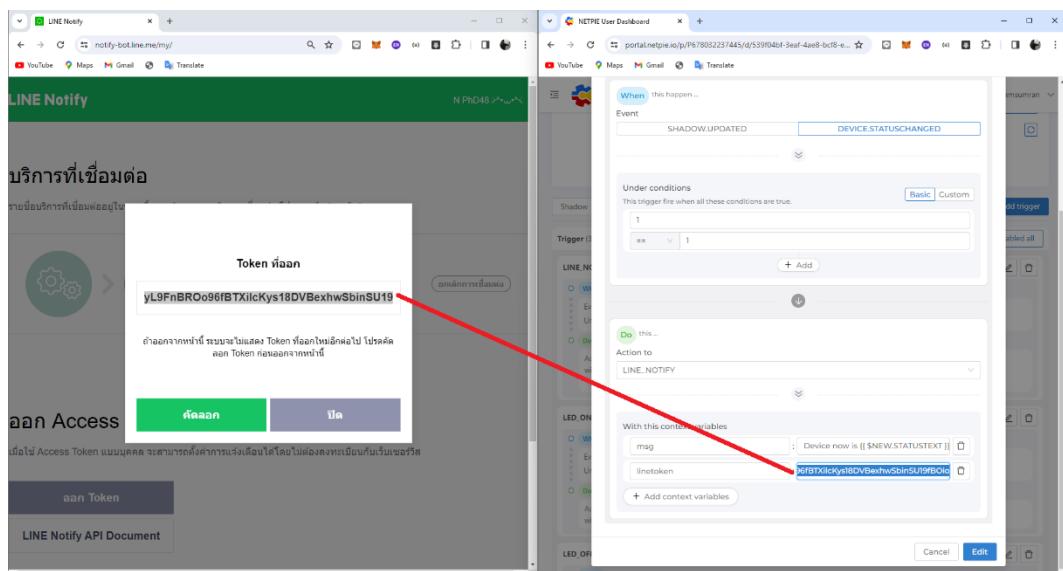
เมื่อใช้ Access Token แบบบุคคล จะสามารถตั้งค่าการแจ้งเตือนได้โดยไม่ต้องลงลงทะเบียนกับเว็บเซอร์วิส

The screenshot shows the "LINE Notify API Document" page. A large red box highlights the "ออก Token" (Generate Token) button, which is located in a grey rectangular area. Below the button, there is some descriptive text about the API document.

รูป A2-1 เว็บไซต์ <https://notify-bot.line.me/th/> สำหรับประยุกต์การควบคุมและทริกเกอร์อุปกรณ์ผ่าน Line notify หลังจากเข้ามาในหน้าเว็บใน Login ด้วย ID Line ของนิสิต และเลือกที่ “หน้าของฉัน” จากนั้นกด “ออก Token” เพื่อนำ Token ที่ได้ไปใส่ใน Trigger บน NETPIE 2020



รูป A2-2 เมื่อเข้าสู่หน้าต่างออก Token กำหนดชื่อ Token เป็นชื่อกลุ่มที่ปรากฏใน myCourseVille จากนั้นเลือก “รับการแจ้งเตือนแบบตัวต่อตัวจาก LINE Notify” หากต้องการรับการแจ้งเตือนแบบกลุ่ม นิสิตจะต้องสร้างกลุ่มแล้ว ดึงเพื่อนเข้ามาแล้วเลือกกลุ่มไลน์ที่ต้องการ เมื่อเรียบร้อยแล้วให้กด “ออก Token”



รูป A2-3 เมื่อได้ Token แล้วให้ทำการคัดลอกและบันทึกเก็บไว้ให้ดี เพราะ Token ที่ได้จะแสดงเพียงครั้งเดียวเท่านั้น จากนั้นไปที่หน้าเว็บ NETPIE 2020 เข้าไปในส่วนของ Trigger Line Notify ให้นำ Token ที่ได้รับวางในช่องของ line token ดังรูปนี้ นิสิตก็สามารถใช้งาน Line Notify ร่วมกับ NETPIE 2020 ได้