

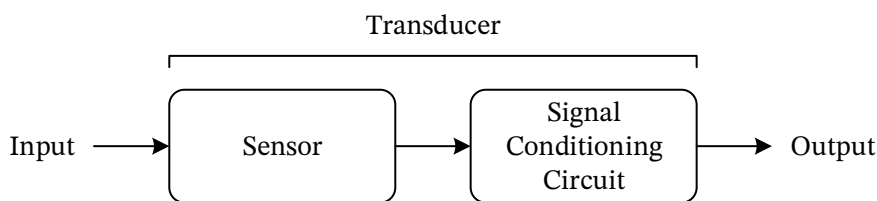


## อินเทอร์เน็ตของสรรพสิ่งส่วนฮาร์ดแวร์แพลตฟอร์ม

## (Hardware Platform for Internet of Things)

## 1. บทนำ

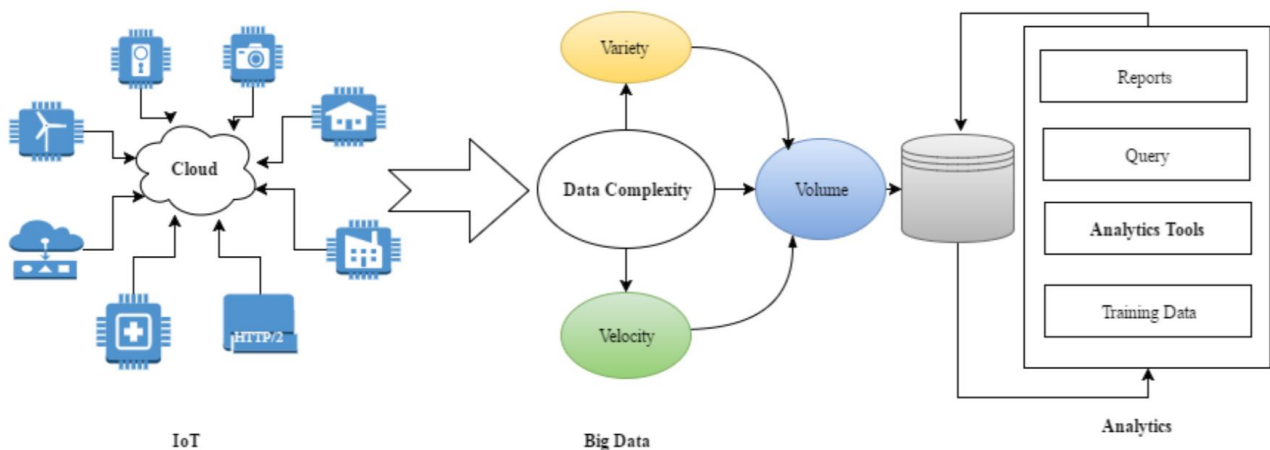
ตัวรับรู้ (Sensor) และตัวเปลี่ยนแปร (Transducer) ทำหน้าที่แปลงพลังงานที่เกิดจากปรากฏการณ์ทางฟิสิกส์ไปเป็นพลังงานไฟฟ้า เช่น ตัวรับรู้ชนิดใช้แสงชนิดลำแสงผ่านตลอด (Through beam/broken beam) และชีวไฟฟ้าสำหรับวัดสัญญาณคลื่นไฟฟ้าหัวใจ (Electrocardiogram: ECG) ส่วนการทำงานของวงจรปรับสภาพสัญญาณ (Signal conditioning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



รูปที่ 1.1 แผนภาพบล็อกของตัวเปลี่ยนแปร (Transducer)

อินเทอร์เน็ต (Internet) ถูกพัฒนาตั้งแต่ปี พ.ศ. 2512 จากการเกิดเครือข่าย ARPANET (Advanced Research Projects Agency NETwork) ซึ่งเป็นเครือข่ายสำนักงานโครงการวิจัยขั้นสูงของกระทรวงกลาโหม 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) จนเกิดเป็นคุณลักษณะของข้อมูลมหัต (Big data)

สำหรับปฏิบัติการในส่วนนี้ จะเป็นการใช้งานฮาร์ดแวร์และพัฒนาชุดคำสั่ง สำหรับการเก็บรวบรวมข้อมูล (Data acquisition) การวิเคราะห์และแปลความหมายข้อมูล (Data analysis and interpretation) และการนำเสนอข้อมูล (Data visualization) จากอาศัยโครงสร้างพื้นฐานและแพลตฟอร์มอินเทอร์เน็ตของสรรพสิ่ง ความสัมพันธ์ระหว่างอินเทอร์เน็ตของสรรพสิ่ง ข้อมูลมหัต และกระบวนการวิเคราะห์ข้อมูลสัมพันธ์ แสดงดังแสดงรูปที่ 1.2



รูปที่ 1.2 ความสัมพันธ์ระหว่างอินเทอร์เน็ตของสรรพสิ่ง ข้อมูลมหัต และกระบวนการวิเคราะห์ข้อมูลสัมพันธ์

## 2. ฮาร์ดแวร์สำหรับอินเทอร์เน็ตของสรรพสิ่ง

### 2.1 ไมโครคอนโทรลเลอร์

สำหรับประมวลผลแบบคลาวด์ (Cloud computing) ในแต่ละจุดของอุปกรณ์อินเทอร์เน็ตของสรรพสิ่ง (Node devices) นอกจากวงจรอิเล็กทรอนิกส์ที่มีอยู่ภายในแล้ว ส่วนประมวลผลนิยมใช้ไมโครคอนโทรลเลอร์ (Microcontroller) ซึ่งประกอบด้วยหน่วยประมวลผลกลาง (Central processing unit : CPU) หน่วยความจำเข้าถึงแบบสุ่ม (Random access memory : RAM) หน่วยความจำแบบถาวร (Read-only memory : ROM) อินเทอร์เฟซและบัส (Interface and bus) เช่น ส่วนขาเข้าขาออก (Input/Output : I/O) เป็นต้น ไมโครคอนโทรลเลอร์มักถูกโปรแกรมให้อ่านค่าจากตัวรับรู้ ประมวลผลระดับหนึ่ง และส่งข้อมูลออกไป ดังนั้นในการพิจารณาเลือกไมโครคอนโทรลเลอร์ให้เหมาะสมกับการใช้งาน ควรคำนึงถึงคุณสมบัติ และข้อจำกัดที่มี เช่น การใช้พลังงาน (Power consumption) หากใช้พลังงานที่น้อยอาจต้องแลกกับความสามารถในการประมวลผล และทรัพยากรบนที่น้อยลง แหล่งข้อมูลในกลุ่มนักพัฒนา (Developer community) ที่เอื้อต่อการตั้งคำถามจากปัญหาที่เกิดขึ้นในระหว่างการพัฒนา หรือการแลกเปลี่ยนความคิดเห็นสำหรับแก้ไขปัญหา

### 2.2 การเชื่อมต่อกับเครือข่าย

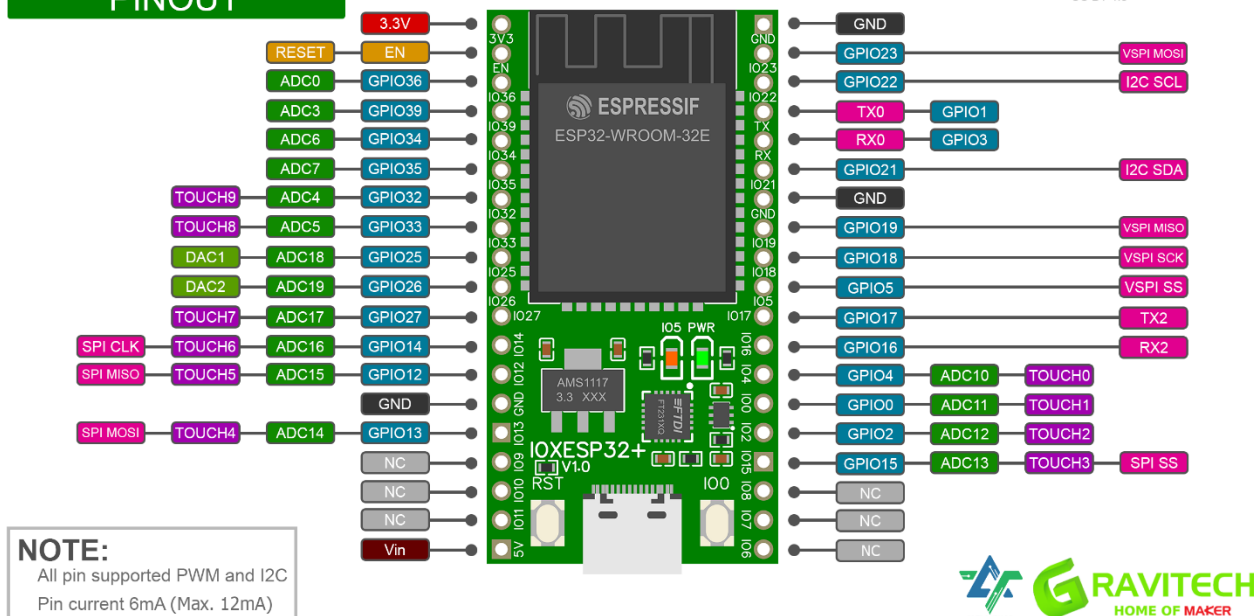
โพรโทคอล (Protocol) คือข้อกำหนดหรือข้อตกลงที่ประกอบด้วยกฎต่าง ๆ สำหรับการสื่อสารรูปแบบเฉพาะระหว่างอุปกรณ์ใดใด เพื่อให้อุปกรณ์ที่มีคุณลักษณะแตกต่างสามารถสื่อสารกันได้ เช่น การสื่อสารระหว่างเครื่องแม่

ข่าย (Server) กับเครื่องลูกข่าย (Client) ผ่านเครือข่ายอินเทอร์เน็ต โพรโทคอลที่นิยมใช้ในอินเทอร์เน็ตของสรรพสิ่ง 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) ในการอัปโหลดโปรแกรมผ่านสาย 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+

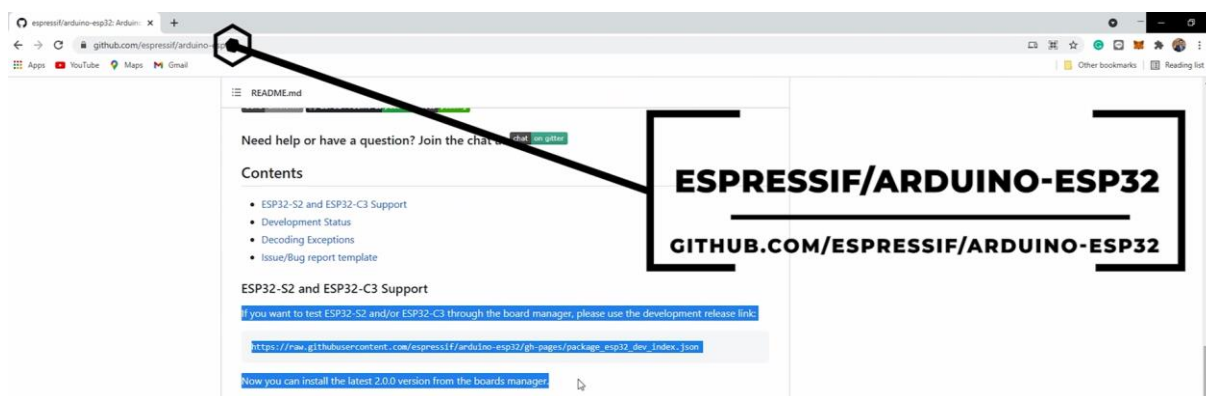
## การทดลองที่ 1 | การติดตั้งชุดพัฒนาซอฟต์แวร์ (IDE) สำหรับ IOXESP32+ และการใช้ Push Button & OLED

\*\*\* มีสิ่งที่ต้องส่ง 2 รายการ คือ

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

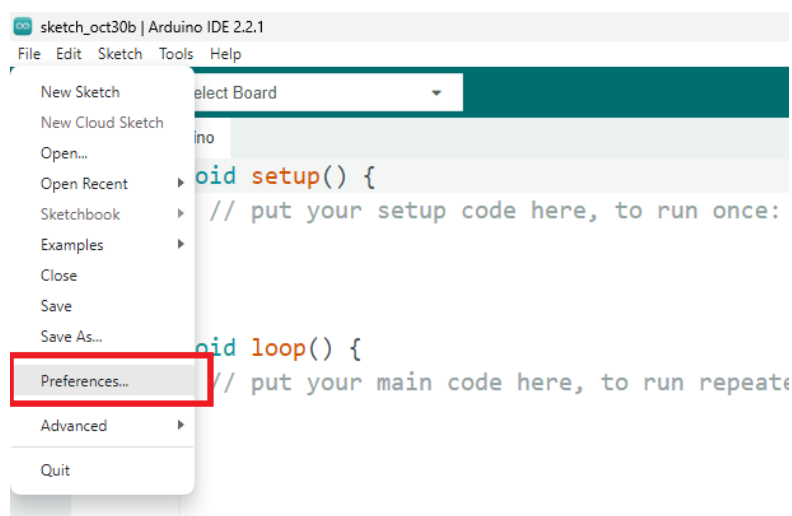
### ขั้นตอนปฏิบัติ

1. ศึกษาวิธีการติดตั้งโปรแกรม Arduino IDE (แนะนำใช้ระบบปฏิบัติการ Windows) จากคลิปวิดีโอประกอบรายวิชาที่ <https://www.youtube.com/watch?v=YZvR1KbdghI> และทำความเข้าใจเกี่ยวกับ ESP32 Repository ที่ <https://github.com/espressif/arduino-esp32> เพื่อติดตั้ง Libraries ที่จำเป็น



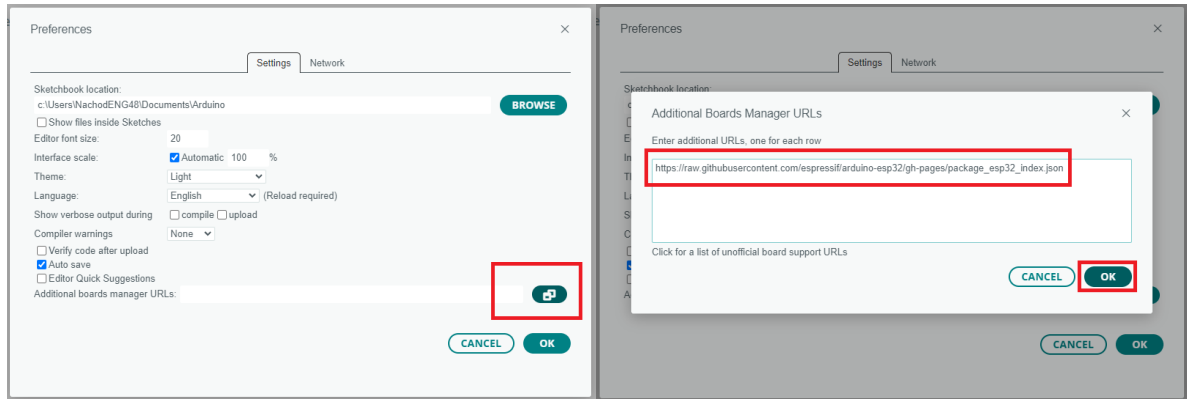
รูปที่ A1-1 ESP32 Repository <https://github.com/espressif/arduino-esp32>

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



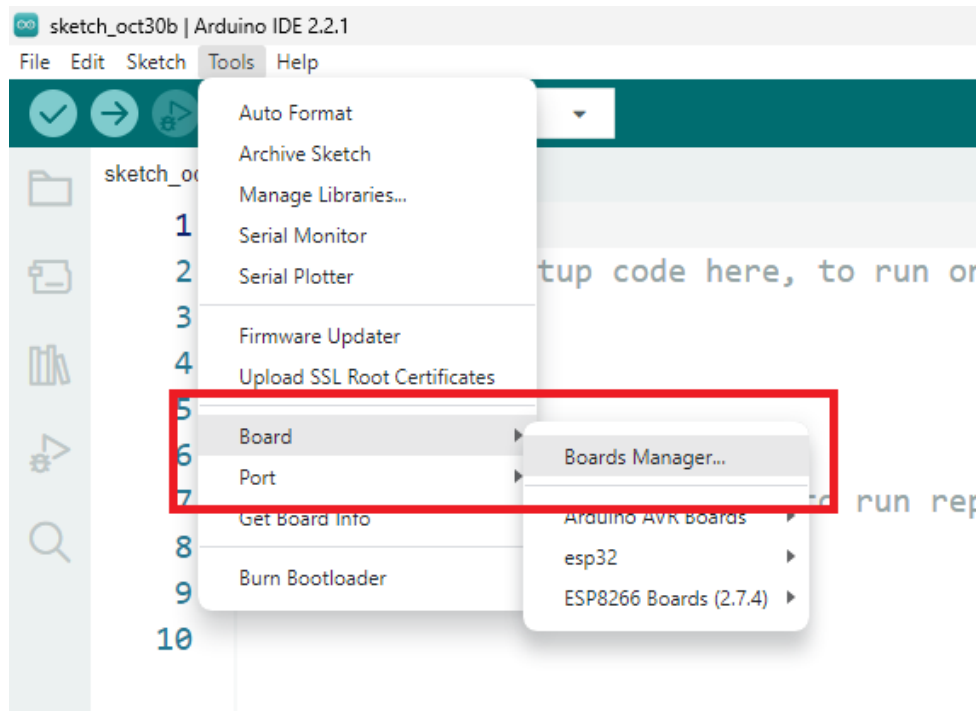
รูปที่ A1-2 การเปิด Preferences ของโปรแกรม Arduino IDE

3. ในหน้าต่าง Preferences ให้ไปที่ Additional Boards Manager URLs และคลิกที่ปุ่มดังรูปที่ A1-3 วาง Development release link ที่คัดลอกจาก ESP32 Repository แล้วจึงกด OK 2 ครั้ง

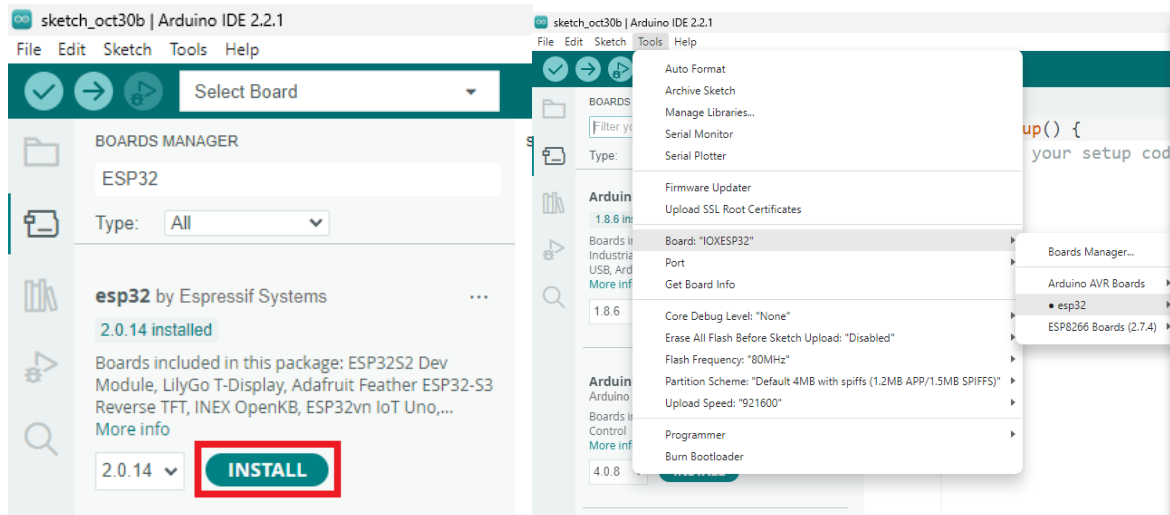


รูปที่ A1-3 ขั้นตอนการวาง Development release link ใน Additional Boards Manager URLs

4. เมื่อดำเนินการในข้อที่ 3 เสร็จแล้ว ให้คลิกที่ Tools >>> Board >>> Boards Manager... ดังรูปที่ A1-4 เพื่อเปิดหน้าต่าง Boards Manager ไปที่ช่องข้อความพิมพ์คำว่า ESP32 ตามด้วยกด Enter จะปรากฏดังรูปที่ A1-5 ให้กดปุ่ม Install รอจนกระทั่งติดตั้งเสร็จสิ้น จะขึ้นข้อความสีเขียว INSTALLED (ขั้นตอนนี้นแนะนำให้เชื่อมต่อกับเครือข่ายที่มีความเสถียร และมีความเร็วอินเทอร์เน็ตที่สูง หากเกิดการขัดจังหวะขึ้น จะต้องเริ่มการติดตั้งใหม่ตั้งแต่ต้น)

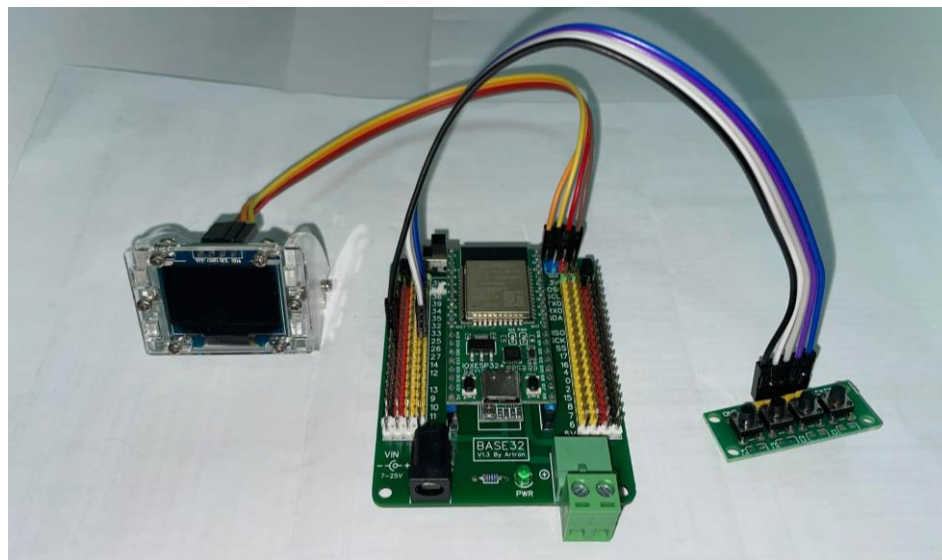


รูปที่ A1-4 การเปิดหน้าต่าง Boards Manager เพื่อติดตั้งบอร์ด ESP32



รูปที่ A1-5 หน้าต่าง Boards Manager ที่ติดตั้ง ESP32 Repository เรียบร้อยแล้ว

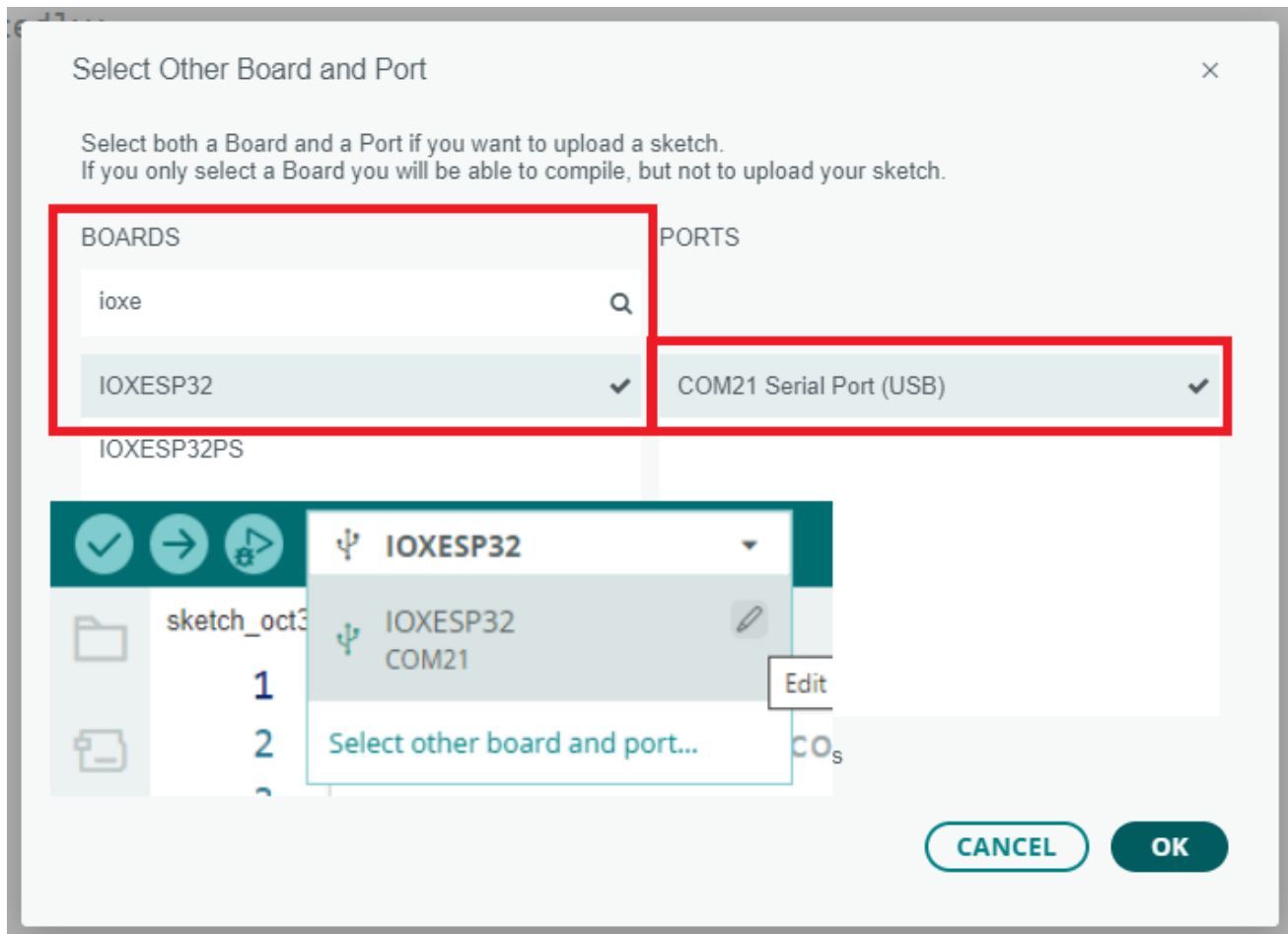
5. ต่อบอร์ดกับอุปกรณ์ต่างๆ ดังรูปที่ A1-6 โดยใช้ IOXESP32+ ร่วมกับ Expansion Board, 4 Channel Push Button และ OLED Panel 128x64 I2C โดยสามารถศึกษาวิธีการใช้งานหน้าจอ OLED ที่ <https://www.youtube.com/watch?v=2kckPMdi2XE> และการใช้งาน 4 Channel Push Button ที่ <https://www.youtube.com/watch?v=XsXG-DoFoMs>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




รูปที่ A1-6 การต่อบอร์ดกับอุปกรณ์ต่างๆ สำหรับทำการทดลองที่ 1

6. หลังจากต่อวงจรเรียบร้อยแล้วให้ต่อสาย USB-C เข้ากับบอร์ด IOXESP32+ และคอมพิวเตอร์ จากนั้นเข้าสู่โปรแกรม Arduino IDE จะสามารถเลือกบอร์ดดังรูปที่ A1-7 โดยการเข้าไปที่ Select Other Board and Port จากนั้นจึงพิมพ์ในช่อง 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 ให้แก่ชุดคำสั่งในหน้า Source code จาก LED\_BUILTIN เป็น 5 ซึ่งเป็นหมายเลข IO5 ของบอร์ดที่มี LED built-in ติดตั้งอยู่ดังรูปที่ A1-8 แล้วจึงกดปุ่มอัปโหลด  จากนั้นรอนจนกระทั่งขึ้นสถานะ Done uploading มุมซ้ายล่างของหน้าต่างโปรแกรม แล้วสังเกตการกะพริบของ 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

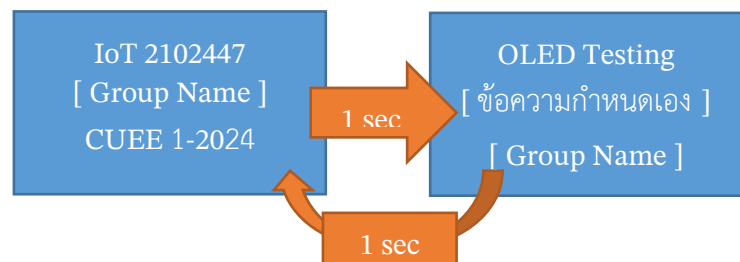
8. [การทดลองนี้ต้องส่ง .ino ไฟล์ที่ 1] เมื่อทดสอบการทำงานในข้อที่ 7 เรียบร้อยแล้ว ต่อไปคือการทดลองเกี่ยวกับหน้าจอแสดงผล OLED Panel 128x64 I2C โดยจะต้องเขียนโปรแกรมด้วย Source code ของตัวเองให้สามารถแสดงผลดังรูปที่ A1-9 ในหน้าจอแรกจะต้องแสดงคำว่า IoT 2102447 ตามด้วยชื่อกลุ่มใน myCourseVille และ CUEE 2-2022 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>
```

Pseudocode

```
...
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(...);
}
```



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. วิดีโอคลิปแสดงการทำงาน ให้แนบ URL ของวิดีโอลงในช่องตอบคำถาม ส่งผ่าน text box บน MyCourseVille

☆ จบการทดลองที่ 1 ☆



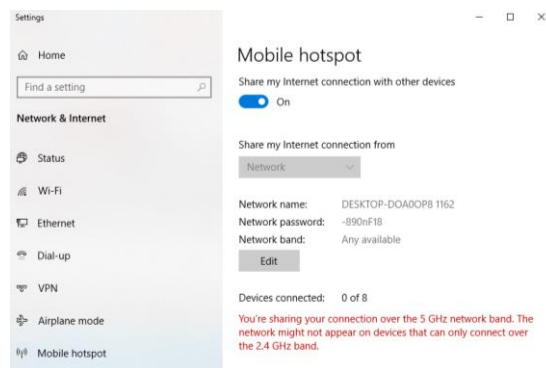
## การทดลองที่ 2 | การเชื่อมต่อกับเครือข่ายไวไฟ (WiFi) และการเทียบฐานเวลาในประเทศไทย (NTP)

### \*\*\* มีสิ่งที่ต้องส่ง 2 รายการ คือ


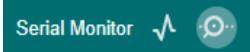
1. Source Code ที่มีนามสกุลไฟล์ .ino เพียง 1 ไฟล์ ส่งผ่าน attachment box บน MyCourseVille
2. วิดีโอคลิปแสดงการทำงาน ให้แนบ URL ของวิดีโอลงในช่องตอบคำถาม ส่งผ่าน text box บน MyCourseVille

### ขั้นตอนปฏิบัติ

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



รูปที่ A2-1 การใช้งาน Mobile Hotspot บน Windows 11

2. ให้เขียนโปรแกรมเทียบฐานเวลา (NTP) ด้วยซอร์สโค้ดตัวอย่างดังรูป A2-2 โดยให้แก้ไข ssid และ password ให้ตรงกับ Personal Hotspot หรือ Mobile Hotspot ของนิสิตไม่เช่นนั้นจะไม่สามารถเชื่อมต่ออินเทอร์เน็ตได้ ให้ต่อสาย USB-C เข้ากับบอร์ด IOXESP32+ แล้วจึงกดปุ่มอัปโหลด  จากนั้นรอจนกระทั่งขึ้นสถานะ 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)

\*\*\* แนะนำให้ Library ของ NTPClient by Fabrice Weinberg version 3.2.1 \*\*\*

The screenshot shows the Serial Monitor window with the following output:

```

Second: 01
Monday, October 30 2023 23:21:02
Second: 02
Monday, October 30 2023 23:21:03
Second: 03
Monday, October 30 2023 23:21:04
Second: 04
Monday, October 30 2023 23:21:05
Second: 05
Monday, October 30 2023 23:21:06
Second: 06
Monday, October 30 2023 23:21:07
Second: 07
Monday, October 30 2023 23:21:08
Second: 08

```

The status bar at the bottom indicates: Ln 5, Col 43 IOXESP32 on COM21.

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

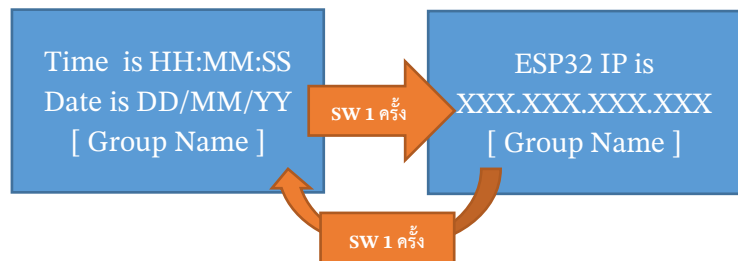
3. [การทดลองนี้ต้องส่ง .ino ไฟล์ที่ 1] ให้ออกแบบการทำงานของอุปกรณ์ IoT โดยการใช้หน้าจอแสดงผล OLED Panel 128x64 I2C แสดงค่าเวลาเทียบฐานเวลาในประเทศไทย (NTP) ดังรูป A2-4 จากอินเทอร์เน็ต (ไม่ใช่การแสดงผลผ่าน Serial monitor) ซึ่งจะต้องแสดงผลดังต่อไปนี้

- 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 ถ้าทำให้แสดงผล  
ชื่อเดือนและปี 4 หลัก แบบ 26/October/2021 จะมี Bonus point เพิ่ม
- แสดงชื่อกลุ่มของนิสิตที่ทำการทดลอง ตามรายชื่อกลุ่มที่ปรากฏใน myCourseVille

เมื่อกดปุ่ม Push Button 1 ครั้ง ข้อความบนหน้าจอ OLED จะเปลี่ยนไปแสดงหมายเลข IP Address ของ IOXESP32+ ที่กำลังเชื่อมต่ออินเทอร์เน็ต ณ ปัจจุบัน และตามด้วยชื่อกลุ่มของนิสิต สามารถประยุกต์ใช้จาก ฟังก์ชัน WiFi.localIP() ได้ โดยที่หน้าจอดังกล่าวจะติดค้างไว้จนกว่าจะมีการกดปุ่ม Push Button อีก 1 ครั้ง หน้าจอ OLED จะกลับไปแสดงเวลา วันที่ และชื่อกลุ่มตามเดิม บันทึกผลด้วยการถ่ายวิดีโออธิบายการทำงาน 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 และข้อความบนหน้าจอที่ต้องทำส่ง

4. ส่งผลการทดลอง มีสิ่งที่ต้องส่ง 2 รายการ

\*\*\* มีสิ่งที่ต้องส่ง 2 รายการ คือ

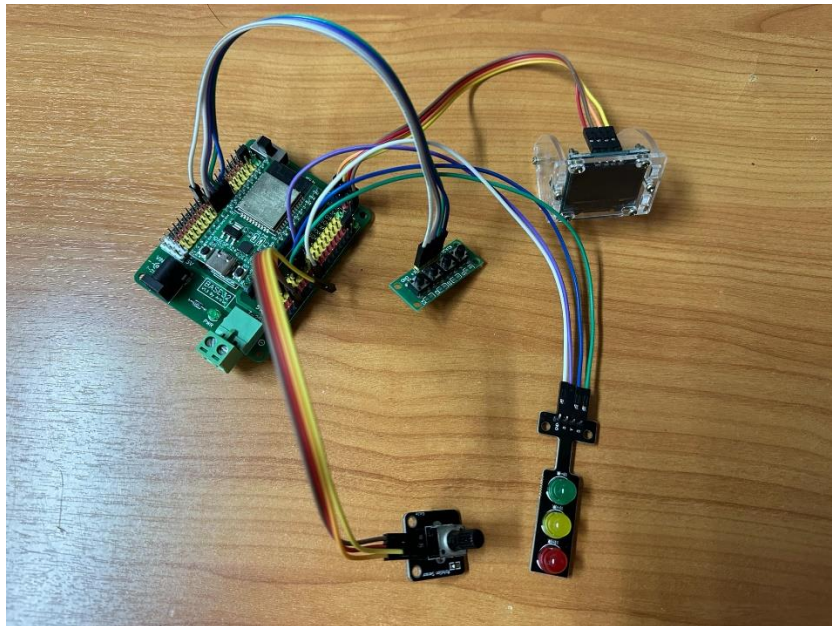
1. Source Code ที่มีนามสกุลไฟล์ .ino เพียง 1 ไฟล์ ส่งผ่าน attachment box บน MyCourseVille
2. วิดีโอคลิปแสดงการทำงาน ให้แนบ URL ของวิดีโอในช่องตอบคำถาม ส่งผ่าน text box บน MyCourseVille

★ 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

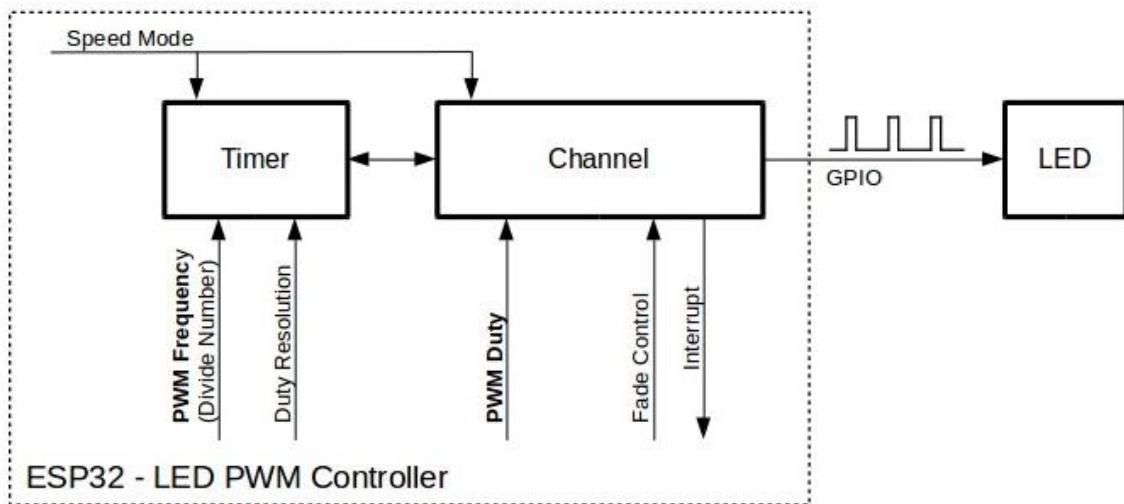
```
#include <SPI.h>
#include <Wire.h>
#include <WiFi.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
...
void setup(){
    ...
}

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 จะแตกต่างจากบอร์ด 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 นั้นหมายความว่ามีความถี่เป็น 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 = ...;
```

ส่วนถัดมาการกำหนดขา 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 เพิ่มลดความสว่างของทุกการทดลองและบันทึกวิดีโอคลิป 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. วิดีโอคลิปแสดงการทำงาน ให้พิมพ์ URL ของวิดีโอลงในช่องตอบคำถาม ส่งผ่าน text box

☆ จบการทดลองที่ 3 ☆

## การทดลองที่ 4 | การทดลองปุ่มกด 4 Push Button Switch และการเขียนโปรแกรมลำดับ

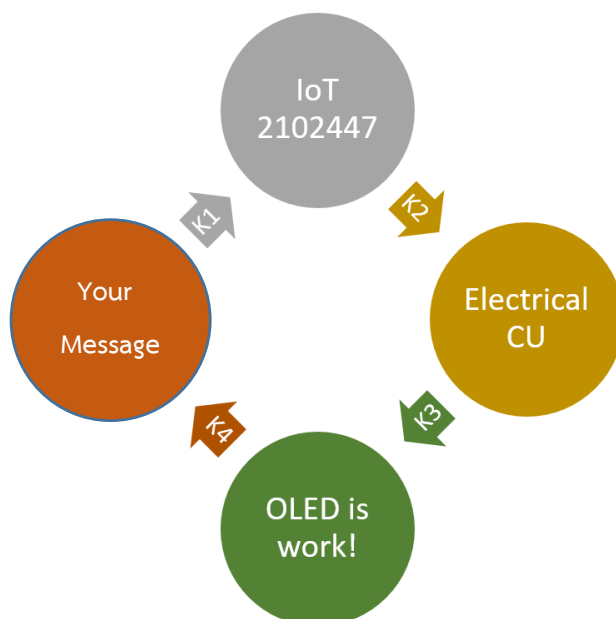
## \*\*\* Special Bonus \*\*\*

## ขั้นตอนปฏิบัติ

6. การทดลองเกี่ยวกับ 4 Channel Push Button ให้ศึกษาการทำงานแบบ Debouncing โดยสามารถดูตัวอย่างจากเมนู 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 1 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)

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



## ข้อกำหนดของ State machine

- เมื่ออยู่ที่ IoT 2102447 สามารถกดปุ่ม K2, K3 หรือ K4 ได้
- เมื่ออยู่ที่ Electrical CU สามารถกดปุ่ม K3, K4 หรือ K1 ได้
- เมื่ออยู่ที่ OLED is work! สามารถกดปุ่ม K4, K1 หรือ K2 ได้
- เมื่ออยู่ที่ 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. วิดีโอคลิปแสดงการทำงาน ให้พิมพ์ URL ของวิดีโอลงในช่องตอบคำถาม ส่งผ่าน text box