รายงาน การทดลอง การประยุกต์ใช้งาน NETPIE

กลุ่มที่ 8

คณะผู้จัดทำ

นาย	จตุรภุช	โมรา	6430036121
นาย	จิรโชติ	ศรีจิรานนท์	6430043521
นาย	ธีรวัฒน์	เลิศอัมพรวิทย์	6430108721

เสนอ

รศ.ดร สุรีย์ พุ่มรินทร์

รายงานนี้เป็นส่วนหนึ่งของวิชา

2102447 Electronics Engineering Laboratory

ภาคเรียนที่ 2 ปีการศึกษา 2567

จุฬาลงกรณ์มหาวิทยาลัย

สารบัญ

1. ที่มาและความสำคัญ

การเดินทางเป็นสิ่งที่ผู้คนต้องพบเจอได้ในชีวิตประจำวัน โดยเฉพาะการเดินทางบนท้องถนนที่มียานพาหนะมากมาย ซึ่งเป็นสิ่งที่เราหลีกเลี่ยงไม่ได้ที่จะต้องพบเจอ ซึ่งหากเราเดินผ่านถนนอย่างไม่ระวังตัวก็อาจจะเกิดอุบัติเหตุได้ โดยเฉพาะขณะที่มีการใช้งานยานพาหนะบนถนน หรือตอนกลางคืน

ระบบตรวจวัดสภาพแวดล้อมบนท้องถนน จะช่วยแจ้งเตือนว่าช่วงเวลาไหน สามารถเดินข้ามถนนได้อย่างปลอดภัย หรือจะต้องระมัดระวังตัว หรือเป็นอันตราย โดยจะมีการแสดงผลผ่านสัญญาณไฟจราจร รวมทั้งแสดงผลผ่านหน้าจอ OLED และ ส่งข้อมูลโดยใช้ระบบ IoT ด้วย NETPIE เพื่อแจ้งข้อมูลแบบ Real Time

2. วัตถุประสงค์

1) เพื่อตรวจวัดสภาพแวดล้อมที่เกิดขึ้นว่าเหมาะสมกับการเดินข้ามถนนหรือไม่ โดยใช้ PIR และ LUX sensor

2) เพื่อแสดงผลข้อมูลสภาพแวดล้อมบนท้องถนน ผ่าน LED สัญญาณไฟจราจร และ หน้าจอ OLED

3) เพื่อประยุกต์การใช้งาน IoT ในการรายงานสภาพแวดล้อมบนท้องถนน โดยใช้ NETPIE

3. ประโยชน์ที่คาดว่าจะได้รับ

สามารถนำระบบการตรวจวัดสภาพแวดล้อมบนท้องถนนมาปรับใช้เพื่อแจ้งเตือนคนที่เดินข้ามถนนให้ระมัดระวังตัวมากขึ้น และหลีกเลี่ยงการเดินข้ามถนนในบริเวณที่สภาพแวดล้อมไม่เหมาะสม เพื่อลดโอกาสในการเกิดอุบัติเหตุบนท้องถนน

4. ขั้นตอนการทำงาน, Flow chart/Blackbox และ Schematic diagram

4.1 ขั้นตอนการทำงาน

การทำงานจะรับค่า 2 ค่า คือ

1. ค่า PIR เป็นค่าที่ตรวจจับวัตถุเคลื่อนที่อยู่ตรงหน้า หากไม่มีวัตถุเคลื่อนที่ จะมีค่าเป็น 0 หากมีวัตถุเคลื่อนที่ จะมีค่าเป็น 1
2. ค่า LUX เป็นค่าที่ตรวจวัดความสว่างของแสงในเวลานั้น

เงื่อนไขสำหรับการแสดงผล คือ

ถ้า PIR = 1 และ LUX < 50 ถือว่ามียานพาหนะแล่นผ่าน และสภาพแวดล้อมมืด ไม่สามารถมองเห็นรถได้ชัดเจน ซึ่งเป็นอันตราย หากเดินข้ามถนน สัญญาณไฟจราจรจะแสดงผลสีแดง

ถ้า PIR = 1 และ LUX >= 50 ถือว่ามียานพาหนะแล่นผ่าน และสภาพแวดล้อมสว่าง สามารถมองเห็นรถได้ชัดเจน สามารถเดินข้ามถนนได้ แต่ต้องระมัดระวังตัว สัญญาณไฟจราจรจะแสดงผลสีเหลือง

ถ้า PIR = 0 ถือว่าไม่มียานพาหนะแล่นผ่าน สามารถเดินข้ามถนนได้ สัญญาณไฟจราจรจะแสดงผลสีเขียว

โปรแกรมนี้เป็นการทำงานแบบ multitasking ประกอบด้วย 5 task ย่อย ได้แก่

TASK 1 อ่านค่า PIR จาก PIR Motion Sensor แล้วส่งค่าไปยัง NETPIE ทุก 1 วินาที

TASK 2 อ่านค่า LUX จาก TSL2561 LUX Sensor แล้วส่งค่าไปยัง NETPIE ทุก 1 วินาที

TASK 3 แสดงผล Traffic สัญญาณไฟจราจร ทุก 1 วินาที โดยมีเงื่อนไข ดังนี้

ถ้า PIR = 1 และ LUX < 50 แสดงผล Traffic สีแดง

ถ้า PIR = 1 และ LUX >= 50 แสดงผล Traffic สีเหลือง

ถ้า PIR = 0 แสดงผล Traffic สีเขียว

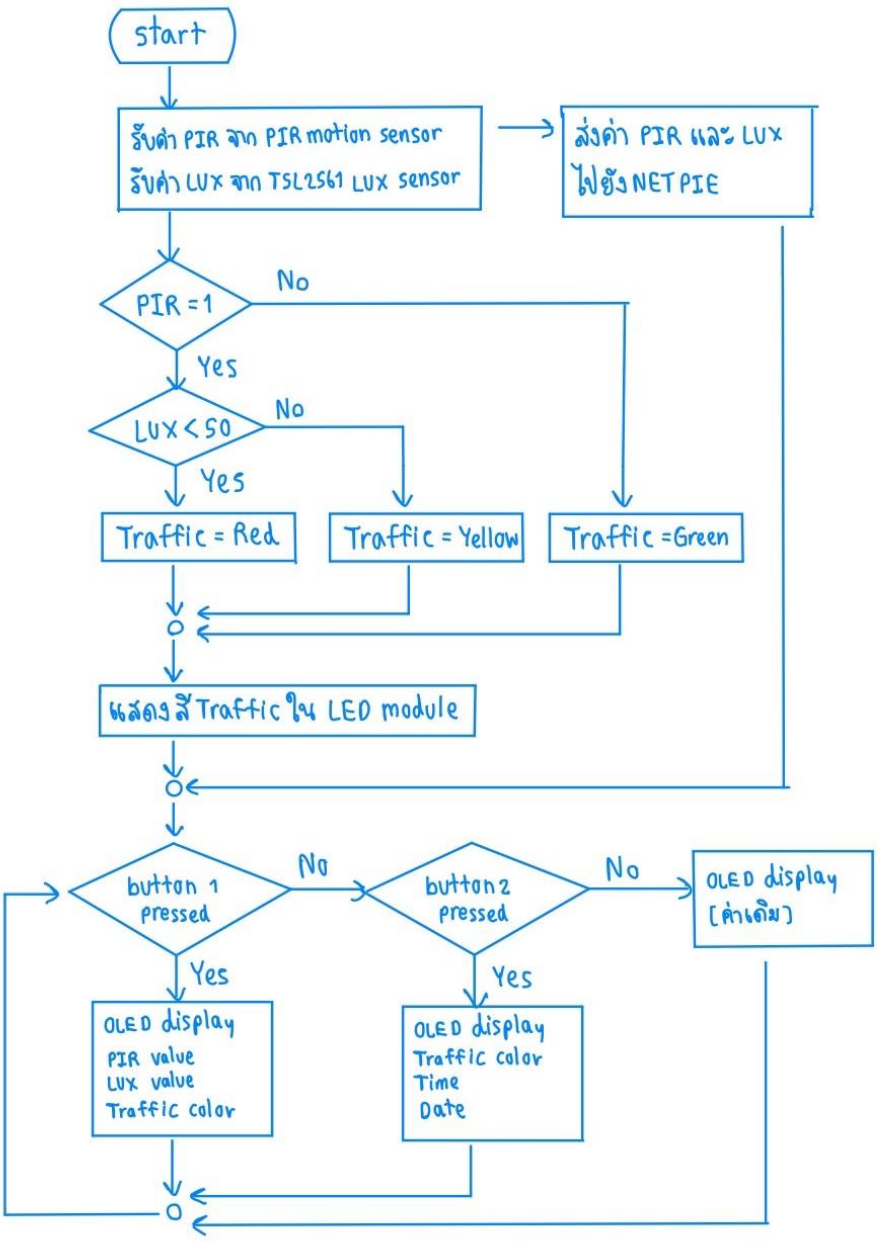
TASK 4 ตรวจสอบการเชื่อมต่อ NETPIE และ WIFI ทุก 1 วินาที

TASK 5 แสดงไอคอนและค่าต่างๆ และการเชื่อมต่อ WIFI ทุก 1 วินาที

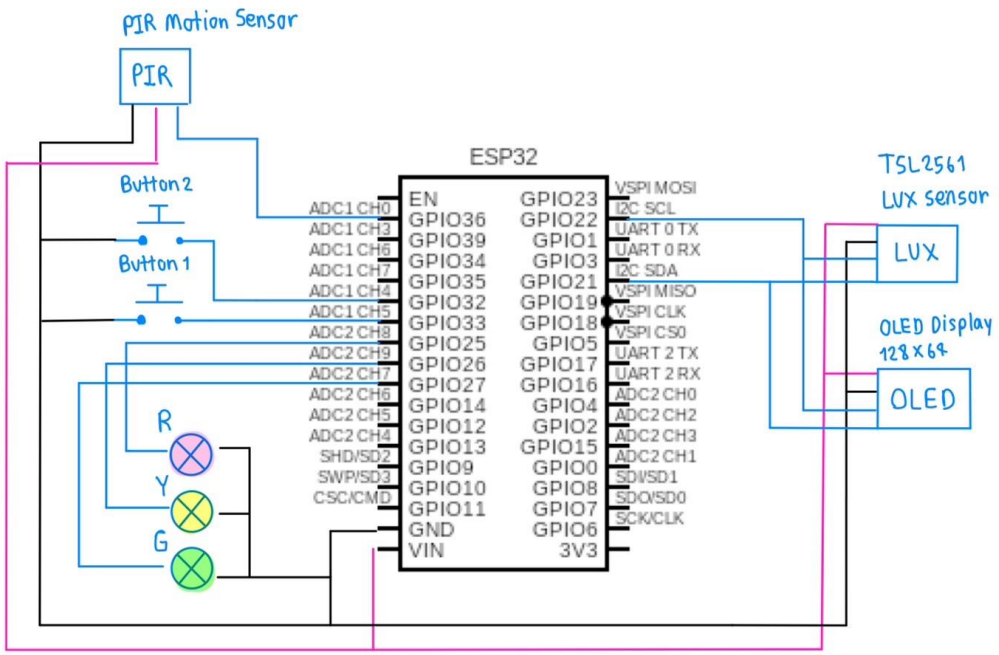
เมื่อกด BUTTON 1 แล้วปล่อย จะแสดงผล ค่า PIR, ค่า LUX, สีของสัญญาณไฟจราจร

เมื่อกด BUTTON 2 แล้วปล่อย จะแสดงผล สีของสัญญาณไฟจราจร, เวลาปัจจุบัน, วันที่ปัจจุบัน

4.2 Flowchart

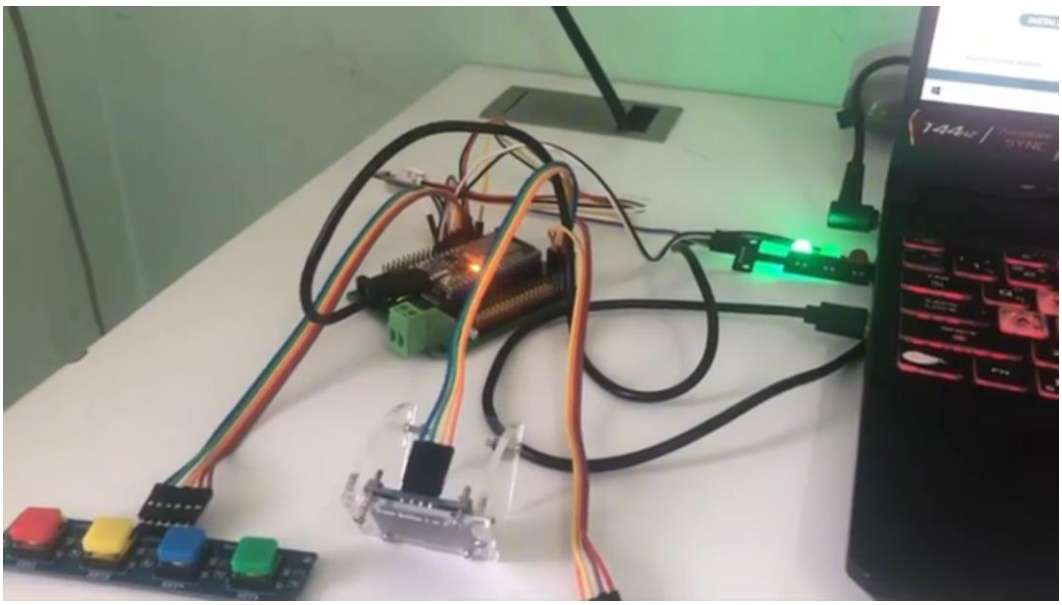


4.3 Schematic diagram

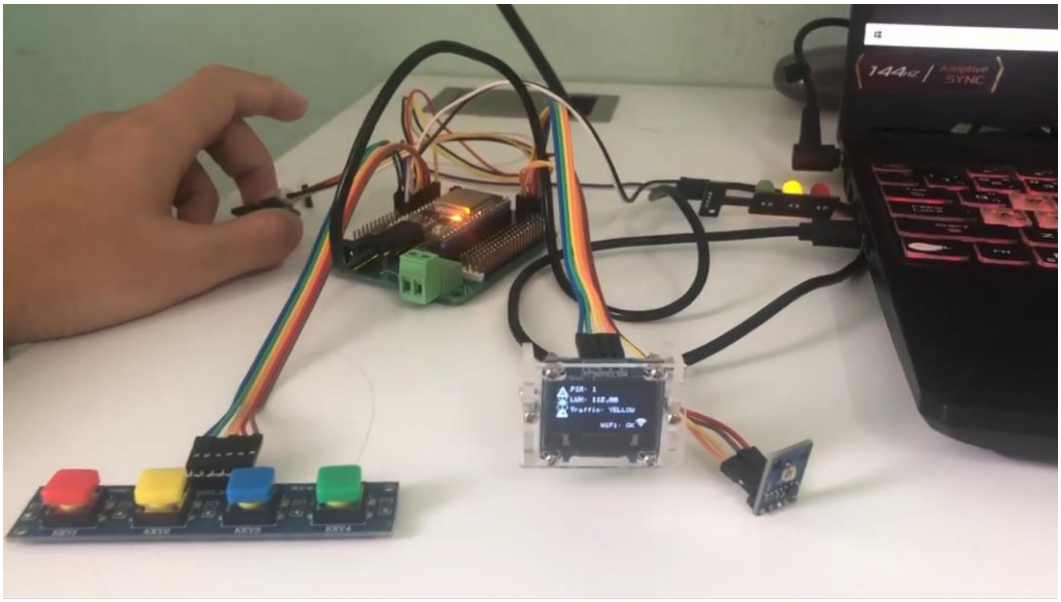


5. ผลการทดลอง

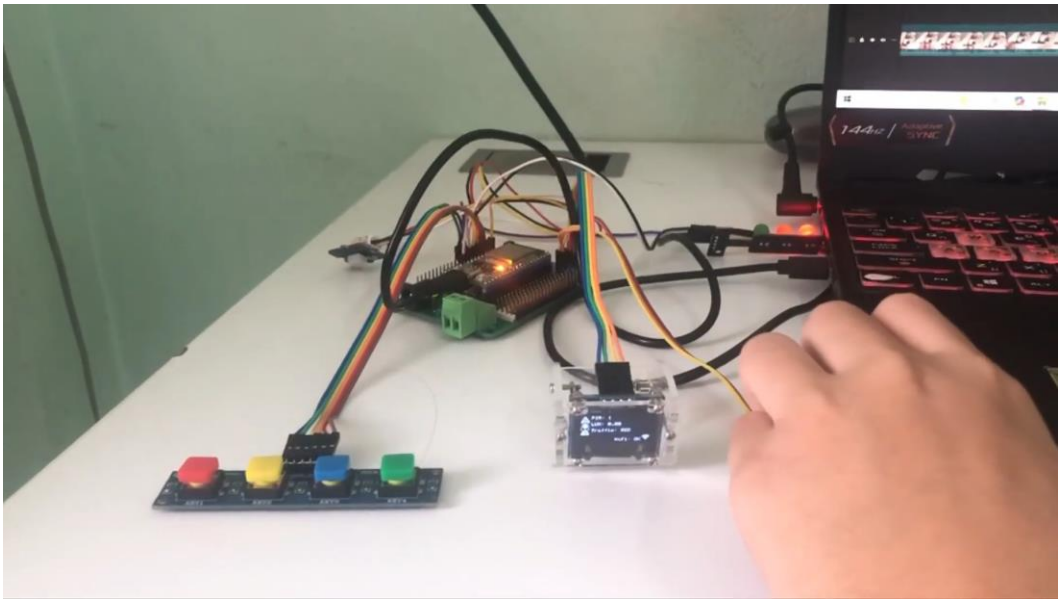
LED สีเขียวติดเมื่อ PIR มีค่าเป็น 0 หรือไม่มีการเคลื่อนไหวโดยรอบบริเวณเซนเซอร์



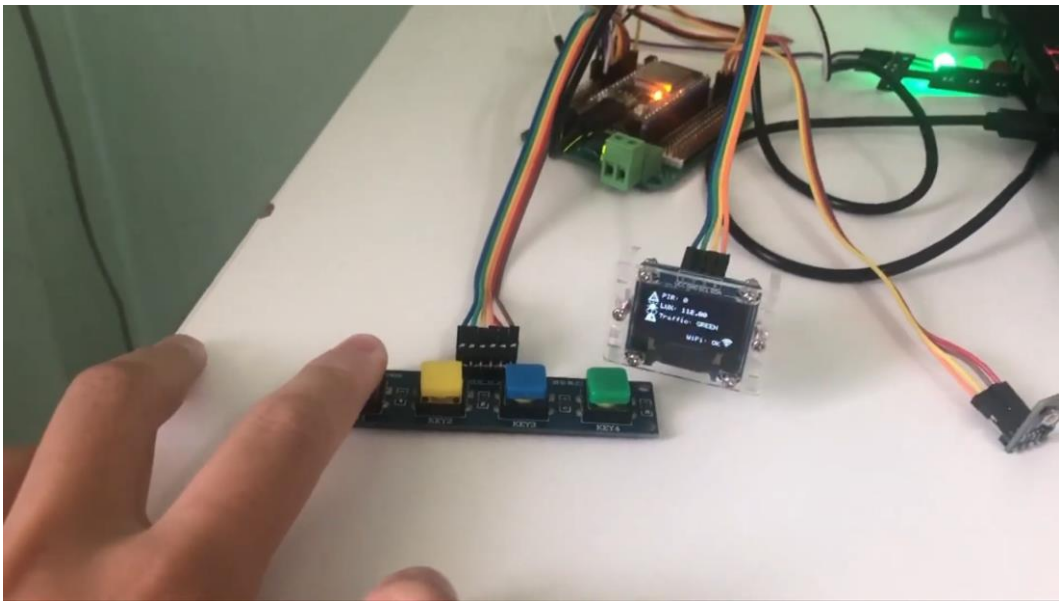
LED สีเหลืองติดเมื่อ PIR มีค่าเป็น 1 หรือก็คือมีการเคลื่อนไหวบริเวณเซนเซอร์ และ Lux sensor อ่านค่าได้มากกว่ากว่า50



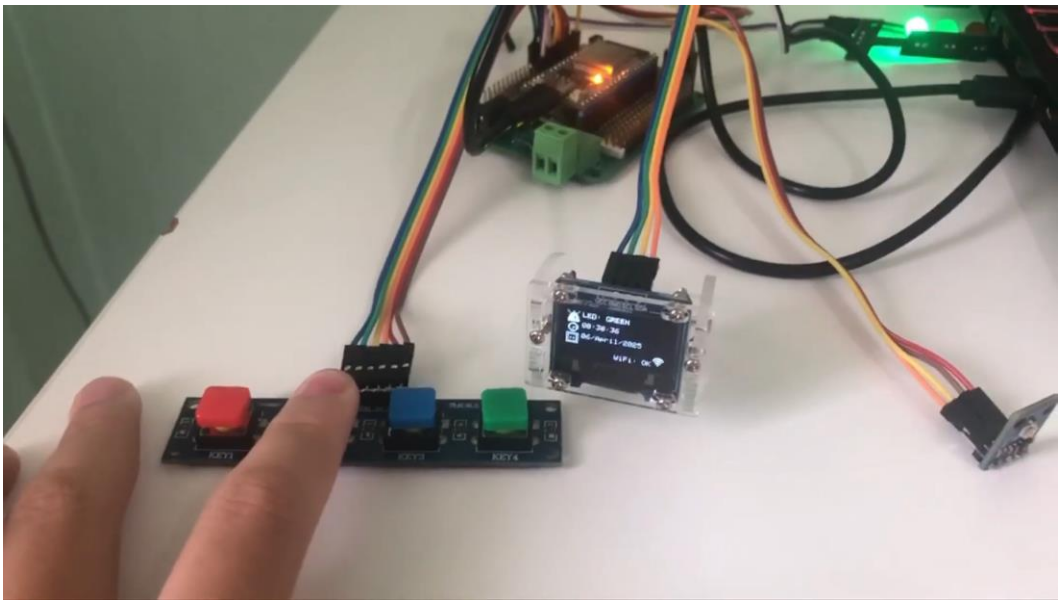
LED สีแดงติดเมื่อ PIR มีค่าเป็น 1 หรือก็คือมีการเคลื่อนไหวบริเวณเซนเซอร์ และ Lux sensor อ่านค่าได้น้อยกว่า 50



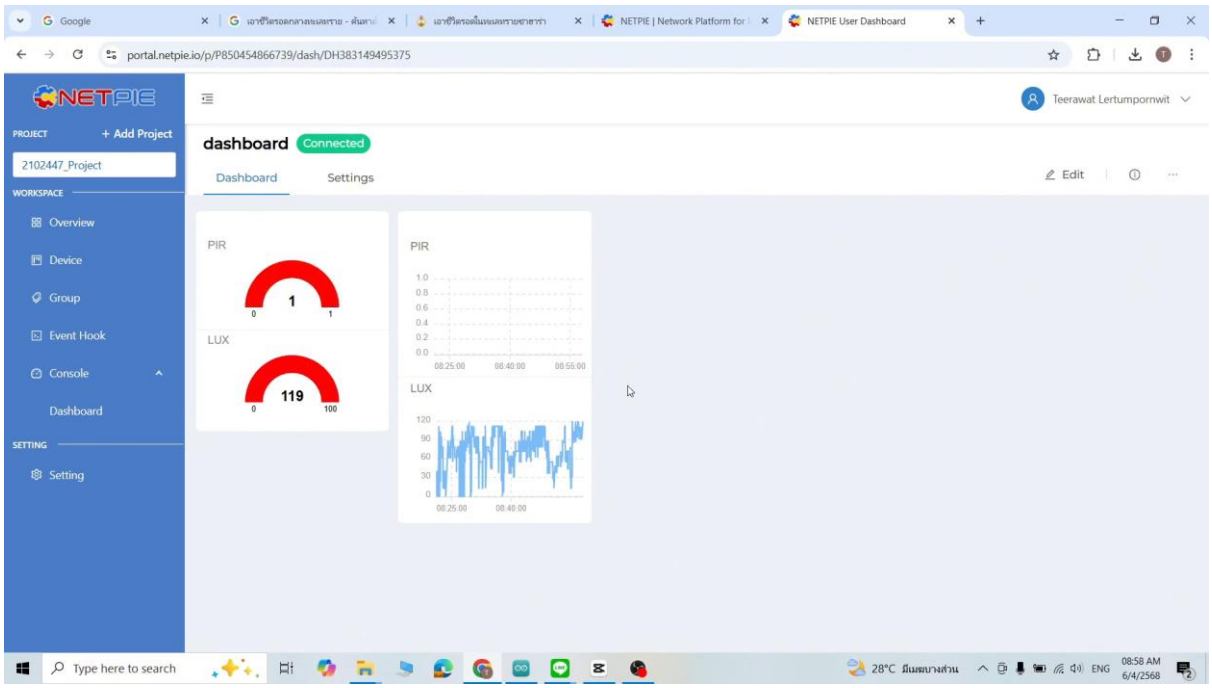
การแสดงผลหน้าจอ OLED ผ่านการกดปุ่มที่ 1 โดยแสดงค่า PIR LUX และ สีของไฟ LED



การแสดงผลหน้าจอ OLED ผ่านการกดปุ่มที่ 2 โดยแสดงสีของไฟ LED เวลา และวันที่



การแสดงค่าต่างๆที่อ่านได้จากเซนเซอร์แสดงบน NETPIE



6. อภิปรายผล และสรุปผล

จากผลการออกแบบและทดลอง พบว่าระบบที่ออกแบบสามารถทำงานได้ตามที่ออกแบบไว้ คือ ถ้า PIR = 1 และ LUX < 50 แสดงผล Traffic สีแดง ถ้า PIR = 1 และ LUX >= 50 แสดงผล Traffic สีเหลือง และถ้า PIR = 0 แสดงผล Traffic สีเขียว รวมถึง TASK อื่นๆก็สามารถทำงานได้ตามออกแบบ เช่น การแสดงค่าบน NETPIE และอื่นๆ โดยปัญหาที่พบเช่น ความไวของเซนเซอร์ PIR

7. แหล่งอ้างอิง

MINI PIR Sensor: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/2244/101020353_Web.pdf

LUX Sensor: handsontec.com/dataspecs/sensor/BH1750 Light Sensor.pdf

8.ภาคผนวก อธิบาย Source code

```
#include <Wire.h>
#include <Adafruit_TSL2561_U.h>
#include <WiFi.h>
#include <time.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <PubSubClient.h>
#include "credentials.h"
#include "iot_iconset_16x16.h"

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

#define OLED_RESET     -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define PIR_PIN 36 // GPIO36 สำหรับเซ็นเซอร์ PIR
#define OUTPUT_1 25 // RED LED (output)
#define OUTPUT_2 26 // YELLOW LED (output)
#define OUTPUT_3 27 // GREEn LED (output)
#define BUTTON_1 33
#define BUTTON_2 32

// NTP Time settings (GMT+7 for Thailand)
const char* ntpServer = "pool.ntp.org";
const long gmtOffset_sec = 3600 * 7;
const int daylightOffset_sec = 0;

Adafruit_TSL2561_Unified tsl =
Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);
WiFiClient espClient;
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
PubSubClient client(espClient);

int pirState = 0;
float lux = 0.0;
String led_color = "OFF";
char msg[100];
```

```
// ตัวแปรสถานะหน้าจอและสถานะปุ่มก่อนหน้า
int Switch_State = 0;
bool lastB1 = HIGH, lastB2 = HIGH;


void setup() {
  Serial.begin(115200);
  delay(100);

  pinMode(PIR_PIN, INPUT);
  pinMode(OUTPUT_1, OUTPUT);
  pinMode(OUTPUT_2, OUTPUT);
  pinMode(OUTPUT_3, OUTPUT);

  pinMode(BUTTON_1, INPUT_PULLUP);
  pinMode(BUTTON_2, INPUT_PULLUP);

  while (!tsl.begin()) {
    Serial.println(F("TSL2561 failed to begin"));
    delay(100);
  }

  while(!oled.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("OLED failed to begin"));
    delay(100);
  }


  oled.clearDisplay(); // clears the screen and buffer
  oled.setTextSize(1);
  oled.setTextColor(WHITE, BLACK);
  oled.setCursor(0,0);
  oled.println(F("OLED passed"));
  oled.display();
  delay(100);

  WiFi.begin(ssid, password);
  delay(100);

  client.setServer(mqtt_server, mqtt_port);
  client.connect(mqtt_client, mqtt_username, mqtt_password);
  Serial.println(F("NETPIE CONNECTED"));
  delay(100);
```

```
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

  xTaskCreate(vTaskPIR, "PIR", 4096, NULL, 1, NULL);
  xTaskCreate(vTaskLUX, "LUX", 4096, NULL, 1, NULL);
  xTaskCreate(vTaskTraffic, "Traffic", 4096, NULL, 1, NULL);
  xTaskCreate(vTaskConnectionCheck, "ConnectionCheck", 4096, NULL, 1,
NULL);
  xTaskCreate(vTaskOLED, "OLED", 8192, NULL, 1, NULL);
}

void loop() {
 // ใช้ FreeRTOS
}

// Task 1: Read PIR, send to NETPIE every 1 second
void vTaskPIR(void * pvParameters) {
  for (;;) {
    pirState = digitalRead(PIR_PIN); // อ่านค่า PIR

    if (client.connected()) {
      String payload = "{\"data\": {\"PIR\": " + String(pirState ?
"true" : "false") + "}}";
      payload.toCharArray(msg, (payload.length() + 1));
      client.publish("@shadow/data/update", msg);
    }
    vTaskDelay(1000 / portTICK_PERIOD_MS); //delay 1 seconds
  }
}

// Task 2: Read LUX, send to NETPIE every 1 second
void vTaskLUX(void * pvParameters) {
  for (;;) {
    sensors_event_t event;
    tsl.getEvent(&event); // อ่านค่า LUX จากเซนเซอร์
    lux = event.light; // เก็บค่าแสงในตัวแปร
    if (client.connected()) {
      String payload = "{\"data\": {\"LUX\": " + String(event.light, 2)
+ "}}";
      payload.toCharArray(msg, (payload.length() + 1));
      client.publish("@shadow/data/update", msg);
    }
```

```
    vTaskDelay(1000 / portTICK_PERIOD_MS); //delay 1 seconds
  }
}

// Task 3: Control LEDs based on PIR and LUX every 1 second
void vTaskTraffic(void * pvParameters) {
  for (;;) {
    sensors_event_t event;
    tsl.getEvent(&event);

    Serial.print("lux Sensor: ");
    Serial.println(lux);
    Serial.print("PIR Sensor: ");
    Serial.println(pirState);

    if (lux < 50 && pirState == HIGH) {
        digitalWrite(OUTPUT_1, HIGH);
        digitalWrite(OUTPUT_2, LOW);
        digitalWrite(OUTPUT_3, LOW);
        led_color = "RED";
        Serial.println("LED RED ON");
    } else if (lux >= 50 && pirState == HIGH) {
        digitalWrite(OUTPUT_1, LOW);
        digitalWrite(OUTPUT_2, HIGH);
        digitalWrite(OUTPUT_3, LOW);
        led_color = "YELLOW";
        Serial.println("LED YELLOW ON");
    } else {
        digitalWrite(OUTPUT_1, LOW);
        digitalWrite(OUTPUT_2, LOW);
        digitalWrite(OUTPUT_3, HIGH);
        led_color = "GREEN";
        Serial.println("LED GREEN ON");
    }
    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
}
// Task 4: Check WiFi and NETPIE connection every 1 seconds
void vTaskConnectionCheck(void * pvParameters) {
  for (;;) {
  if(client.connected()) {
    client.loop();
    client.publish("@shadow/data/update", msg);
```

```
  } else {
    if(WiFi.status() == WL_CONNECTED) {
      client.disconnect();
      client.connect(mqtt_client, mqtt_username, mqtt_password);
    } else {
      WiFi.disconnect();
      WiFi.begin(ssid, password);
    }
  }
    vTaskDelay(1000 / portTICK_PERIOD_MS);  //delay 1 seconds
  }
}

// Task 5: Display data on OLED every 0.5 second
void vTaskOLED(void *pvParameters) {
  for (;;) {
    bool b1 = digitalRead(BUTTON_1);
    bool b2 = digitalRead(BUTTON_2);

  // ตรวจจับขอบ HIGH → LOW เมื่อปุ่มถูกกด
    if (lastB1 == HIGH && b1 == LOW) {
      Switch_State = 1;
    } else if (lastB2 == HIGH && b2 == LOW) {
      Switch_State = 2;
    }

  // อัปเดตสถานะปุ่มล่าสุด
    lastB1 = b1;
    lastB2 = b2;
    delay(100);

    oled.clearDisplay();
    oled.setTextSize(1);
    oled.setTextColor(SSD1306_WHITE);
    oled.setCursor(0, 0);

    if (Switch_State == 1) {
      // Display PIR with icon
      oled.drawBitmap(0, 0, warning_icon16x16, 16, 16, 1);
      oled.setCursor(20, 0);
      oled.print(F("PIR: "));
      oled.println(pirState);
```

```
      // Display LUX with icon
      oled.drawBitmap(0, 16, sun_icon16x16, 16, 16, 1);
      oled.setCursor(20, 16);
      oled.print(F("LUX: "));
      oled.println(lux, 2);

      // Display Traffic with icon
      oled.drawBitmap(0, 32, siren_icon16x16, 16, 16, 1);
      oled.setCursor(20, 32);
      oled.print(F("Traffic: "));
      oled.println(led_color);

      // Display WiFi Status and Icon
      if (WiFi.status() == WL_CONNECTED) {
        oled.drawBitmap(112, 48, wifi1_icon16x16, 16, 16, 1);
        oled.setCursor(64, 56);
        oled.println("WiFi: OK");
      } else {
        oled.setCursor(64, 56);
        oled.println("WiFi: ...");
      }

    } else if (Switch_State == 2) {
      // Display Traffic with icon
      oled.drawBitmap(0, 0, siren_icon16x16, 16, 16, 1);
      oled.setCursor(20, 0);
      oled.print(F("LED: "));
      oled.println(led_color);

      struct tm timeinfo;
      if (getLocalTime(&timeinfo)) {

        // Display Time with icon
        oled.drawBitmap(0, 16, clock_icon16x16, 16, 16, 1);
        oled.setCursor(20, 16);
        char timeString[9];
        strftime(timeString, sizeof(timeString), "%H:%M:%S",
&timeinfo);
        oled.println(timeString); // แสดงเวลา HH:MM:SS

        // Display Date with icon
        oled.drawBitmap(0, 32, wallplug_icon16x16, 16, 16, 1);
```

```
        oled.setCursor(20, 32);
        char dateString[20];
        strftime(dateString, sizeof(dateString), "%d/%B/%Y",
&timeinfo);
        oled.println(dateString);

      } else {
        // Display Time Error
        oled.setCursor(20, 16);
        oled.println(F("Time Error"));
      }

      // Display WiFi Status and Icon
      if (WiFi.status() == WL_CONNECTED) {
        oled.drawBitmap(112, 48, wifi1_icon16x16, 16, 16, 1);
        oled.setCursor(64, 56);
        oled.println("WiFi: OK");
      } else {
        oled.setCursor(64, 56);
        oled.println("WiFi: ...");
      }

    } else {
      oled.println(F("Press Button 1 or 2"));
    }

    oled.display();

    vTaskDelay(500 / portTICK_PERIOD_MS); //delay 0.5 seconds
  }
}
```