ใบงานที่ 13

การรับส่งข้อมูลระหว่างอุปกรณ์

จุดประสงค์การเรียนรู้

1.ศึกษาการรับส่งข้อมูลระหว่างอุปกรณ์ผ่าน SPI บัส

2.ศึกษาการรับส่งข้อมูลระหว่างอุปกรณ์ผ่าน I2C บัส

3.ศึกษาการแสดงข้อความออกทางโมดูล LCD

4.ศึกษาการปรับแต่งและควบคุมการแสดงผลบนหน้าจอ LCD

5.ศึกษาการแสดงข้อความออกทางโมดูลจอ OLED

6.ศึกษาการแสดงภาพกราฟิกออกทางโมดูลจอ OLED

เครื่องมือและอุปกรณ์การทดลอง

1.เครื่องไมโครคอมพิวเตอร์

2.บอรด์ NodeMCU ESP32

3.โปรแกรมการทดลอง

4.อุปกรณ์อิเล็กทอรนิกส์สำหรับทดลอง

การทดลองที่ 1 การรับส่งข้อมูลระหว่างอุปกรณ์ผ่าน SPI บัส

อุปกรณ์ในการทดลอง

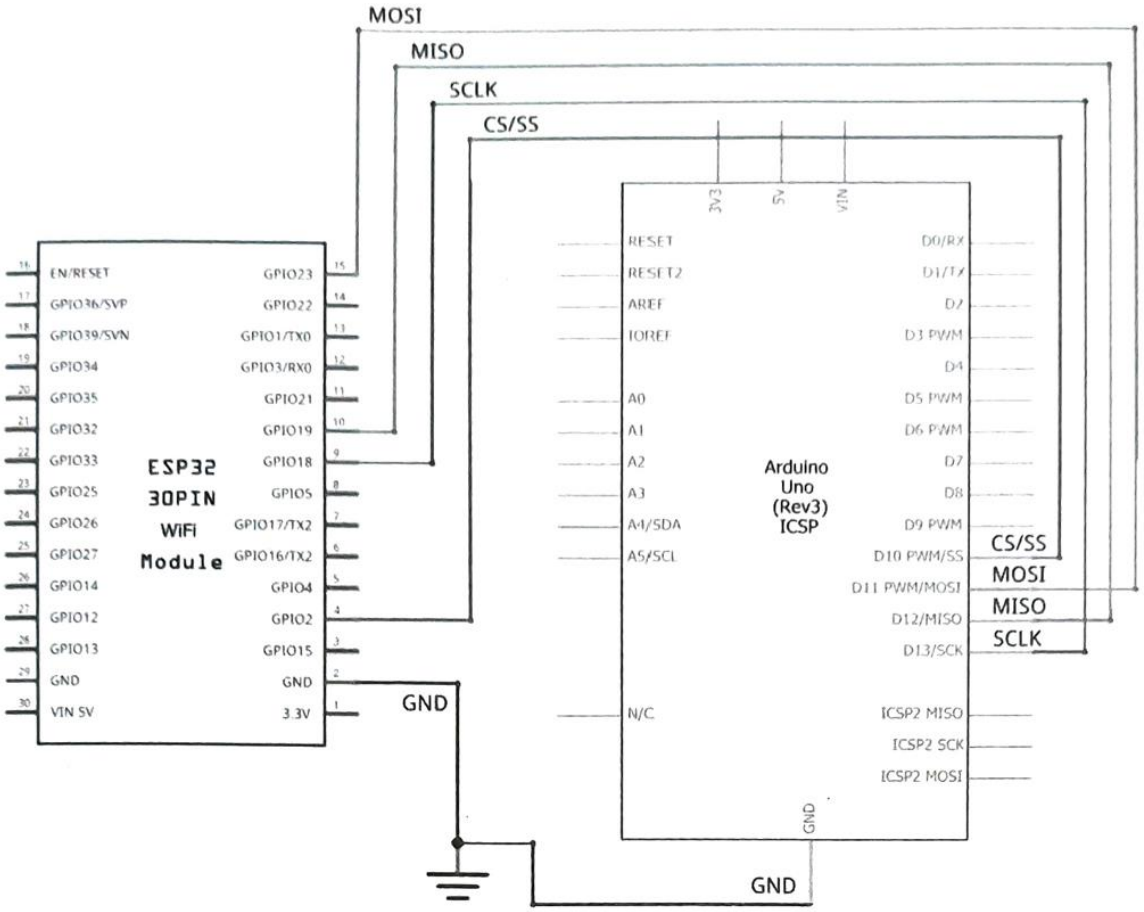
1.บอรด์ NodeMCU ESP32

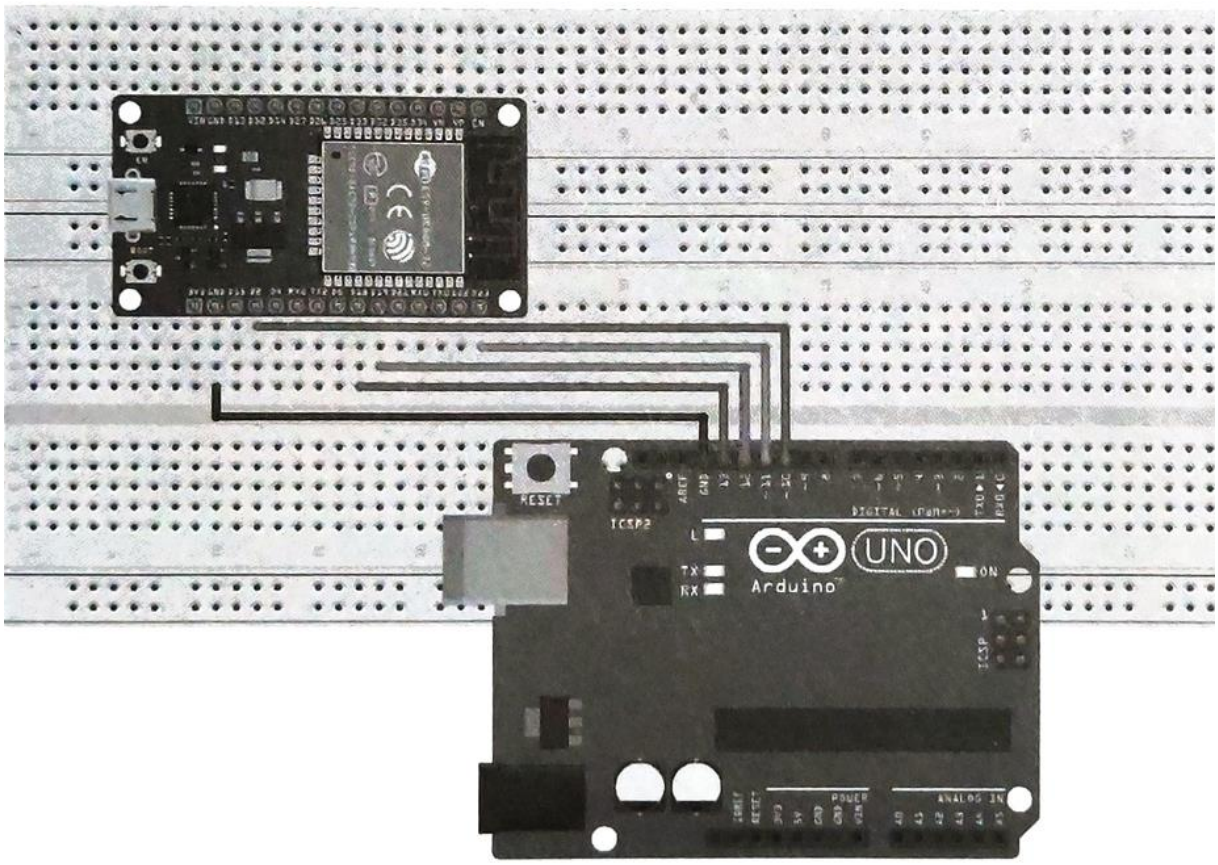
2.บอดร์ Arduino Uno R3

3.แผงต่อวงจร

4.สายไฟต่อวงจร

ประกอบวงจรตามรูป





ตัวอย่างนี้จะเป็นการทดลองส่งข้อมูลไปมาระหว่าง NodeMCU ESP32 กับ Arduino Uno โดยใช้ SPI บัส ซึ่งในที่นี้เลือกใช้บอร์ด NodeMCU ESP32 เป็น Master ส่งข้อมูลเป็นตัวเลข “2” ไปยังบอร์ด Arduino Uno ที่เป็น Slave ทุกๆ 1 วินาที ระหว่างนั้นอุปกรณ์ Slave ก็จะนำเอาข้อมูลตัวเลขที่รับมาไปคำนวณตามคำสั่ง โดยในที่นี้ให้คูณด้วย “8” ก่อนจะส่งผลลัพธ์ที่ได้กลับไปยังอุปกรณ์ Master หรือในที่นี้ก็คือบอร์ด NodeMCU ESP32 โดยเราจะใช้หน้าต่าง Serial Monitor ของ Master ตรวจดูผลลัพธ์ทั้งก่อนที่ข้อมูลจะถูกส่งไปยัง Slave และหลังจากที่ข้อมูลถูกคำนวณเสร็จแล้ว และถูกส่งกลับมายัง Master เพื่อพิสูจน์ว่า มีการรับส่งข้อมูลระหว่าง อุปกรณ์ Master และ Slave จริงๆ

ลงมือต่อวงจรตามรูป สำหรับบอร์ด NodeMCU ที่เป็น Master ในที่นี้จะใช้ขา SCLK, MISO, MOSI และ CS/SS ตรงตำแหน่งขา GPIO18, GPIO19, GPIO23 และ GPIO2 ส่วนบอร์ด Arduino Uno ที่เป็น Slave จะ ใช้ขา SCLK, MISO, MOSI และ CS/SS ตรงตำแหน่งขา D13, D12, D11 และ D10 การเชื่อมต่อเพียงแค่ใช้สาย สัญญาณเสียบโยงขาต่างๆ เข้าหากัน ดังรูป ซึ่งรวมถึงขา GND ที่ต้องต่อร่วมระหว่างกันเอาไว้ด้วย

โค้ดโปรแกรมสำหรับบอร์ด NodeMCU ที่เป็น Master

ทำหน้าที่ส่งข้อมูลตัวเลข “2” ผ่าน SPI บัสไปยังบอร์ด Slave และรับข้อมูลที่ผ่านการคำนวณตามคำสั่ง จาก Slave กลับมายัง Master เพื่อแสดงค่าออกทาง Serial Monitor

```
#include <SPI.h>                          //เรียกใช้งานไลบรารี่ SPI สำหรับการสื่อสารผ่าน SPI
#define CS 2                              //กำหนดให้ขา GPIO2 เป็นขา CS/SS

void setup() {
  pinMode(CS, OUTPUT);                    //กำหนดให้ขา CS/SS เป็น Output เพื่อใช้ติดต่อกับ Slave
  Serial.begin(115200);                   //เริ่มการรับส่งข้อมูลด้วยความเร็ว Baud Rate 115200
  digitalWrite(CS, HIGH);                 //กำหนดให้ขา CS/SS เป็น HIGH เลิกการติดต่อกับ Slave
  SPI.begin();                            //กำหนดสถานะเริ่มต้นใช้งานให้กับ SPI บัส
  SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE0));   //เริ่มต้น
                การใช้งาน SPI บัสด้วยการตั้งค่าความเร็วในการรับส่งข้อมูลสูงสุดที่ 1 MHz, กำหนดให้
                มีการเลื่อนบิตด้วยการส่งบิตจาก MSB ไปก่อน และกำหนดให้ใช้โหมดแบบ SPI_MODE0
}
```

```
void loop() {
  byte spi_dat;                          //ประกาศตัวแปร spi_dat ชนิด byte
  digitalWrite(CS, LOW);                 //กำหนดให้ขา CS/SS เป็น LOW เพื่อติดต่อกับ Slave
  SPI.transfer(0x02);                    //ส่งข้อมูลเป็นตัวเลข "2" ไปยัง Slave หรือ Arduino Uno
  digitalWrite(CS, HIGH);                //กำหนดให้ขา CS/SS เป็น HIGH เลิกการติดต่อกับ Slave
  delayMicroseconds(10);                 //หน่วงรอเป็นเวลา 10 µs หรือเท่ากับ 0.01 ms

  digitalWrite(CS, LOW);                 //กำหนดให้ขา CS/SS เป็น LOW เพื่อติดต่อกับ Slave
  spi_dat = SPI.transfer(0x00);          //รับข้อมูลตัวเลขที่เป็นผลลัพธ์จากการคำนวณจาก Slave
                                         มาเก็บไว้ในตัวแปร spi_dat
  digitalWrite(CS, HIGH);                //กำหนดให้ขา CS/SS เป็น HIGH เลิกการติดต่อกับ Slave
  Serial.println("Processed Data Recieved from Slave is: ");  //พิมพ์ข้อความใน " "
                                                               แล้วขึ้นบรรทัดใหม่
  Serial.print(spi_dat);                 //แสดงค่าตัวแปร spi_dat ออกทาง Serial Monitor
  Serial.println("\r\n");                //ขึ้นบรรทัดใหม่ 2 ครั้ง
  delay(1000);                           //หน่วงรอเป็นเวลา 1 ms
}
```

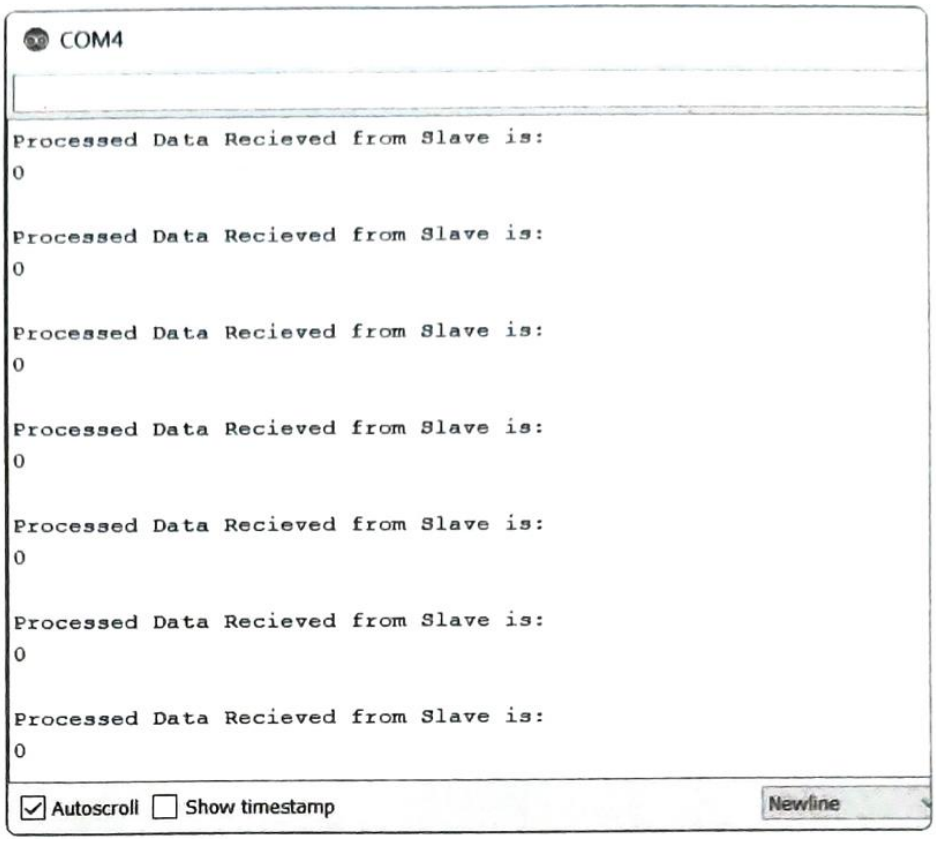
โค้ดโปรแกรมสำหรับบอร์ด Arduino Uno ที่เป็น Slave

ทำหน้าที่รับข้อมูลตัวเลขจากบอร์ด Master มาคำนวณตามคำสั่ง ด้วยการคูณด้วย “3” แล้วส่งผลลัพธ์ผ่าน SPI บัส กลับไปยังบอร์ด Master

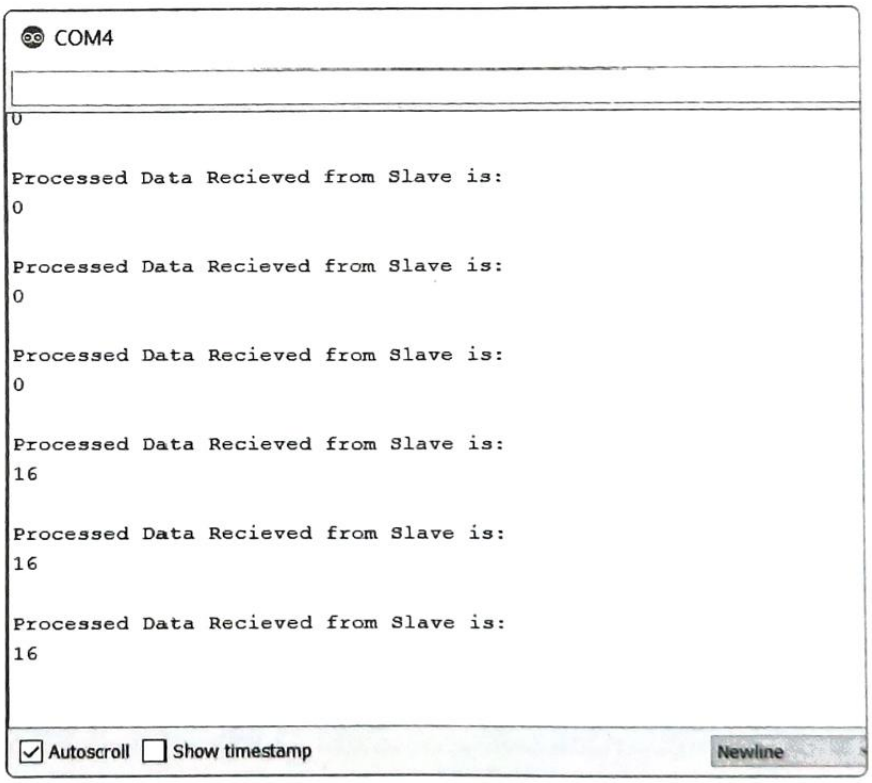
```
#include <SPI.h>                 //เรียกใช้งานไลบรารี่ SPI สำหรับการสื่อสารผ่าน SPI
volatile boolean process_it;     //ประกาศตัวแปร process_it ที่ใช้เก็บสถานะข้อมูลที่รับ
                                 มาจาก Master เพื่อรอการตรวจสอบว่าครบแล้วหรือยัง
byte a;                          //ประกาศตัวแปร a ชนิด byte

void setup (void) {
  Serial.begin (115200);
  SPCR |= bit (SPE);             //กำหนดให้บิต SPE ของรีจิสเตอร์ SPCR มีค่าเป็น 1 เพื่อเปิดใช้งาน SPI บัส
  pinMode(MISO, OUTPUT);         //กำหนดให้ขา MISO เป็น Output เพื่อใช้ส่งข้อมูลไปยัง Master
  process_it = false;            //ให้สถานะตัวแปร process_it มีค่าเป็น false เพื่อแสดงว่าข้อมูล
                                 ที่รับมายังไม่ครบหรือยังไม่มีข้อมูลเข้ามา
  SPCR |= bit (SPIE);            //กำหนดให้บิต SPIE ของรีจิสเตอร์ SPCR มีค่าเป็น 1 เพื่อเปิด
                                 ใช้งาน SPI Interrupt หรือจะใช้คำสั่ง SPI.attachInterrupt() ก็ได้
}

ISR (SPI_STC_vect) {             //ฟังก์ชั่นเริ่มต้นเมื่อ SPI Interrupt ถูกเปิดใช้งาน
```

```
  byte c = SPDR;          //ประกาศตัวแปร c เพื่อใช้เก็บข้อมูลแต่ละไบต์จากรีจิสเตอร์ SPDR
  a = c;                  //ให้ตัวแปร a มีค่าเท่ากับข้อมูลที่เก็บไว้ในตัวแปร c
  SPDR = c*8;             //นำข้อมูลจากตัวแปร c คูณด้วย 8 แล้วนำผลลัพธ์ที่ได้ไปเก็บไว้
                          ในรีจิสเตอร์ SPDR เพื่อรอให้ Master มาอ่านข้อมูลแล้วนำไป
                          แสดงออกทาง Serial Monitor
  process_it = true;      //เปลี่ยนสถานะการรับข้อมูลเป็น true เพื่อแสดงว่าข้อมูลที่รับมาครบแล้ว
}

void loop (void) {
  if (process_it) {       //ตรวจสอบสถานะว่ามีการรับข้อมูลมาจาก Master ครบแล้วหรือยัง
                          ถ้าครบแล้ว
    Serial.println("Recieved\r\n");   //พิมพ์ข้อความใน " " แล้วขึ้นบรรทัดใหม่ 2 ครั้ง
    process_it = false;               //หากยังไม่ครบ ให้เปลี่ยนสถานะการรับข้อมูลเป็น false
  }
}
```

จากนั้นให้เราทดสอบดูว่ามีการ ส่งข้อมูลจาก Master ผ่าน SPI บัสไปยัง Slave หรือไม่ ด้วยการดึงสาย MISO ที่เชื่อมต่อระหว่างขา GPIO19 ของ Master กับขา D12 ของ Slave ด้านใดด้านหนึ่งออก เพื่อยกเลิกการ ส่งข้อมูลหรือผลลัพธ์ที่ได้จากการคำนวณจาก Slave กลับมายัง Master จากนั้นไปที่หน้าต่าง Arduino IDE ที่ เราอัพโหลดโค้ดโปรแกรมให้กับบอร์ด Master หรือในที่นี้คือ NodeMCU ให้ เราเปิดหน้าต่าง Serial Monitor ขึ้นมา จะเห็นข้อความและผลลัพธ์ของตัวเลข เป็น “0” ดังรูป



ต่อมาให้เราทดสอบดูว่าข้อมูลที่ ถูกส่งจาก Master ไปยัง Slave ถูก นำไปคำนวณตามคำสั่งแล้วส่งผลลัพธ์ ของการคำนวณผ่าน SPI บัสกลับ มายัง Master หรือไม่ ด้วยการเสียบสาย MISO เมื่อสักครู่กลับเข้าที่เดิม ผลลัพธ์ที่ได้คือ ในหน้าต่าง Serial Monitor จะเห็นข้อความและผลลัพธ์ ของตัวเลขเป็น “16” ซึ่งก็คือค่าที่ได้ จากการคำนวณที่ Slave ส่งกลับมา ยัง Master นั่นเอง ดังรูป



บันทึกผลการทดลอง

.....

.....

.....

.....

การทดลองที่ 2 การรับส่งข้อมูลระหว่างอุปกรณ์ผ่าน I2C บัส

อุปกรณ์ในการทดลอง

1.บอรด์ NodeMCU ESP32

2.บอดร์ Arduino Uno R3

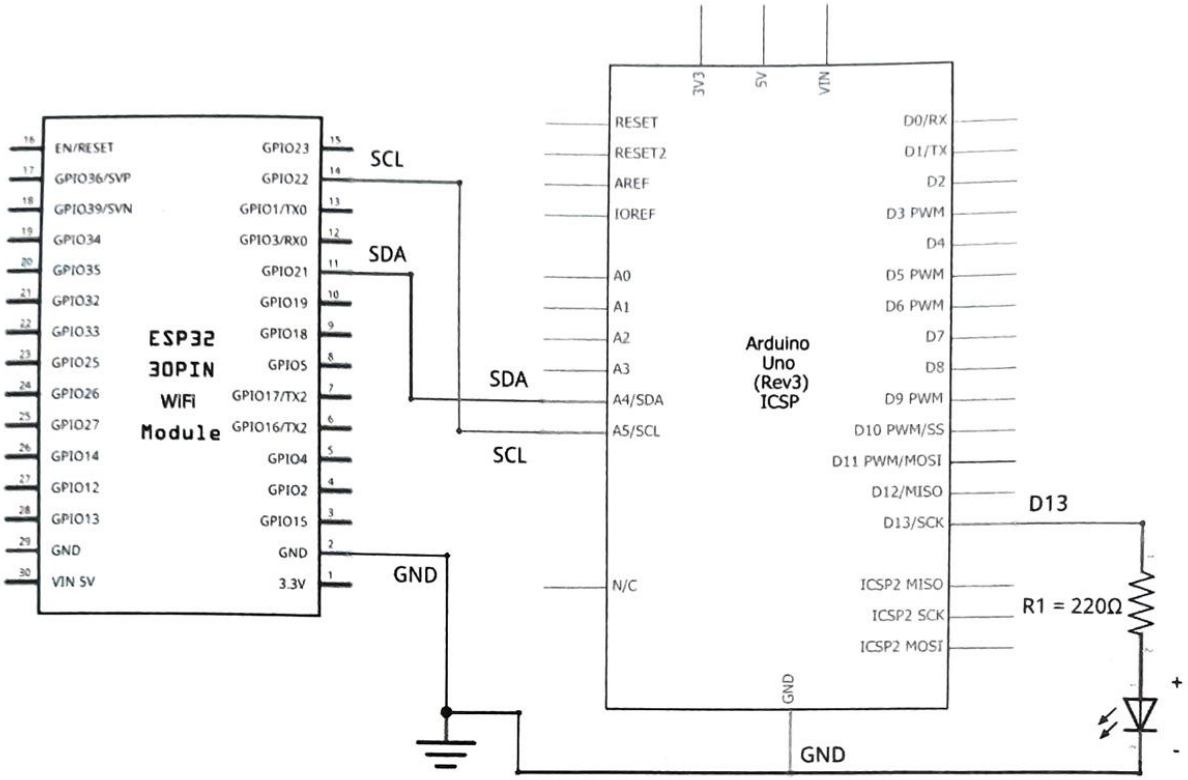
3.หลอดไฟ LED

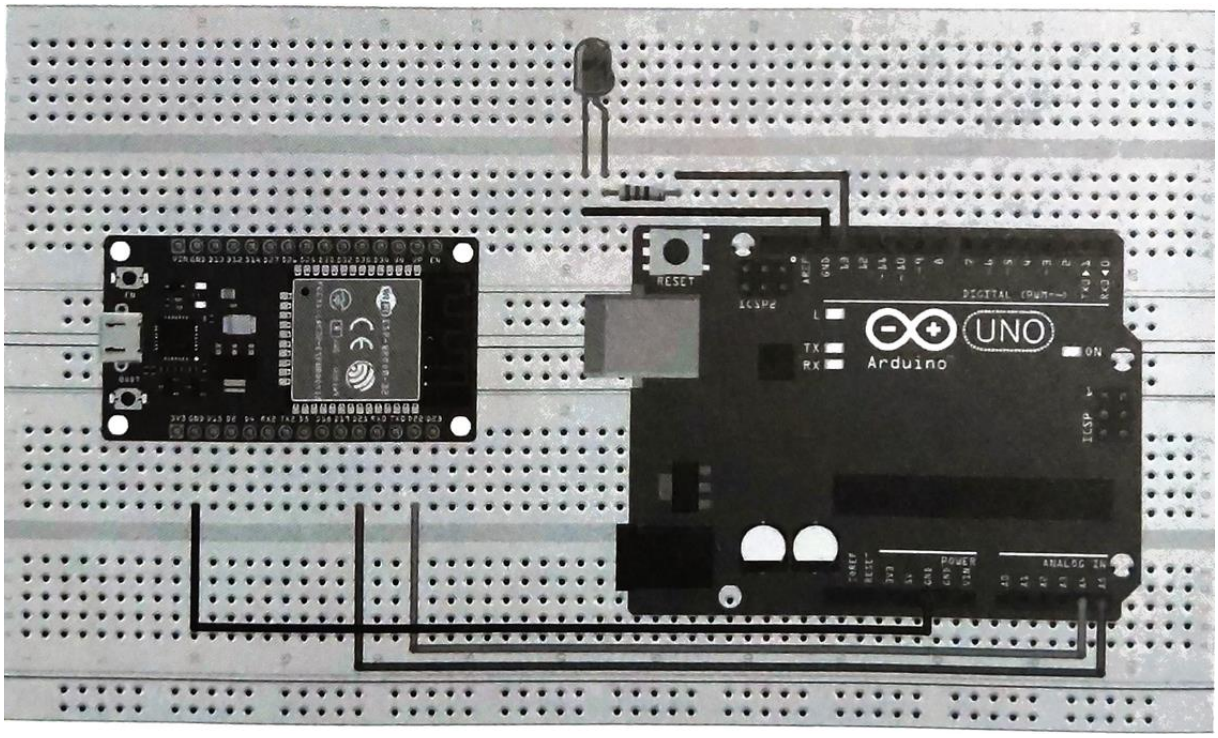
4.ตัวต้านทาน (Resistor)

5.แผงต่อวงจร

6.สายไฟต่อวงจร

ประกอบวงจรตามรูป





ลงมือต่อวงจรตามรูป สำหรับบอร์ด NodeMCU ที่เป็น Master ในที่นี้จะใช้ขา SCL และ SDA ตรงตำแหน่ง ขา GPIO22 และ GPIO21 ส่วนบอร์ด Arduino Uno ที่เป็น Slave จะใช้ขา SCL และ SDA ตรงตำแหน่งขา A5 และ A4 การเชื่อมต่อเพียงแค่ใช้สายสัญญาณเสียบโยงระหว่างขา GPIO22 กับ A5 สำหรับ SCL และขา GPIO21 กับ A4 สำหรับ SDA ดังรูป รวมถึงขา GND ที่ต้องต่อร่วมระหว่างกันเอาไว้ด้วย ส่วนหลอดไฟ LED ที่ใช้แสดงสถานะเปิด/ปิด ให้เอาขั้วบวกหรือ Anode (+) ซึ่งมีขาที่ยาวกว่า ต่อกับขา 13 ของบอร์ด Arduino Uno ที่เป็น Slave (ในที่นี้กำหนดให้เป็น Output) โดยให้มีตัวต้านทาน 2200 คั่นเอาไว้ สำหรับขั้วลบหรือ Cathode (-) ก็ให้ต่อกับกราวด์ (GND)

หลังจากลงมือเชื่อมต่อสายสัญญาณต่างๆ เรียบร้อยแล้ว ขั้นตอนต่อไปก็คือ การเขียนโค้ดและอัพโหลดโปรแกรม โดยเราจะต้องเขียนโค้ดขึ้นมา 2 ชุด ชุดแรกสำหรับอัพโหลดไปยังบอร์ด NodeMCU ที่เป็น Master และชุดสองสำหรับบอร์ด Arduino Uno ที่เป็น Slave ซึ่งมีรายละเอียดดังนี้

โค้ดโปรแกรมสำหรับบอร์ด NodeMCU ที่เป็น Master

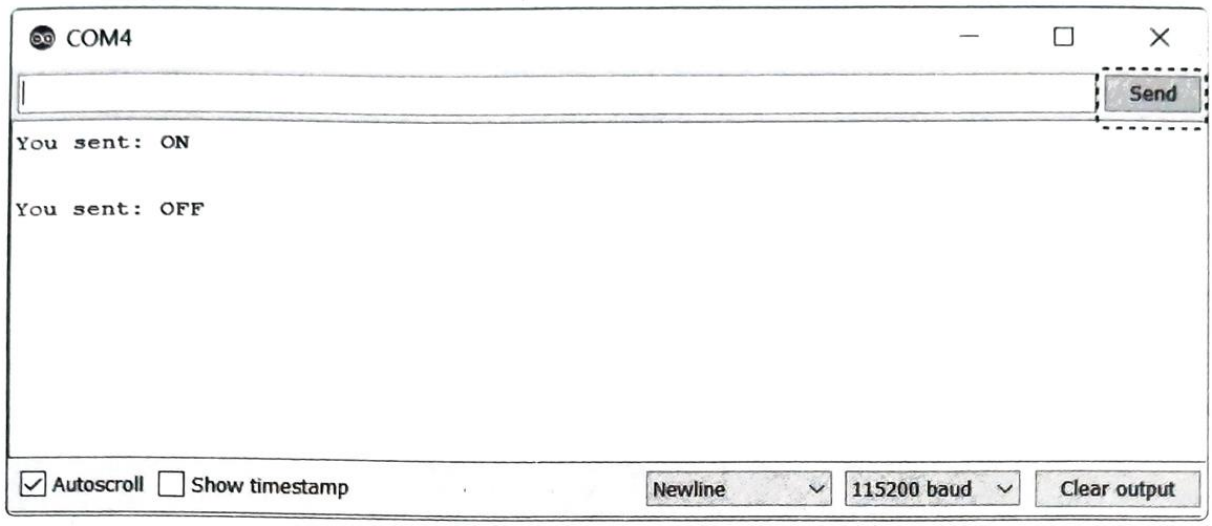
เป็นโค้ดที่ใช้ตรวจสอบตัวอักษรที่กรอกลงไปใน Serial Monitor แล้วส่งข้อมูลตัวอักษรดังกล่าวไปยังบอร์ด Arduino Uno ที่เป็น Slave ผ่านทาง I2C บัส

```
#include <Wire.h>                    //เรียกใช้ไลบรารี่ Wire.h สำหรับการสื่อสารแบบ I2C

void setup() {
  Serial.begin(115200);
  Wire.begin();                      //เริ่มต้นใช้งาน I2C กำหนดให้อุปกรณ์ปัจจุบันเป็น Master
}

void loop() {
  if(Serial.available()) {           //ตรวจสอบว่ามีข้อมูลถูกส่งผ่านเข้ามาทาง Serial Monitor หรือไม่
    String data = Serial.readString(); //ถ้ามี ข้อมูลนั้นจะถูกนำไปเก็บไว้ที่ตัวแปร data
    Wire.beginTransmission(11);      //เริ่มต้นติดต่อกับอุปกรณ์ Slave ด้วยแอดเดรสที่เรา
                                     กำหนดหรือระบุไว้ในคำสั่ง Wire.begin(11); ซึ่งอยู่ในโค้ด
                                     โปรแกรมสำหรับบอร์ด Arduino Uno ที่เป็น Slave หรือ
                                     ในที่นี้ก็คือ 11 หรือ 0xB นั่นเอง
    if(data.indexOf("ON") !=-1) {    //ตรวจสอบข้อมูลในตัวแปร data ถ้าเป็นข้อความ ON
      Wire.write('1');               //ให้ส่งข้อมูลเป็นตัวอักษร 1 ไปยังอุปกรณ์ Slave
    } else if (data.indexOf("OFF") !=-1) {   //ตรวจสอบข้อมูลในตัวแปร data
                                             ถ้าเป็นข้อความ OFF
      Wire.write('0');               //ให้ส่งข้อมูลเป็นตัวอักษร 0 ไปยังอุปกรณ์ Slave
    }
    Serial.print("You sent: ");      //แสดงข้อความใน " " ออกทาง Serial Monitor
    Serial.println(data);            //แสดงข้อมูลที่อยู่ในตัวแปร data แล้วขึ้นบรรทัดใหม่
    Wire.endTransmission();          //ยกเลิกการส่งข้อมูลไปยังอุปกรณ์ Slave
  }
}
```

โค้ดโปรแกรมสำหรับบอร์ด Arduino Uno ที่เป็น Slave

เป็นโค้ดที่ใช้แสดงสถานะของข้อมูลที่รับมาจากบอร์ด NodeMCU ที่เป็น Master ซึ่งถ้าข้อมูลที่รับมาเป็น “1” หรือมีการพิมพ์คำว่า “ON” จะทำให้สถานะของหลอดไฟ LED ที่เชื่อมต่ออยู่กับขา 13 ติด แต่ถ้าข้อมูล ที่รับมาเป็น “0” หรือพิมพ์คำว่า “OFF” จะทำให้สถานะของหลอดไฟ ดับ

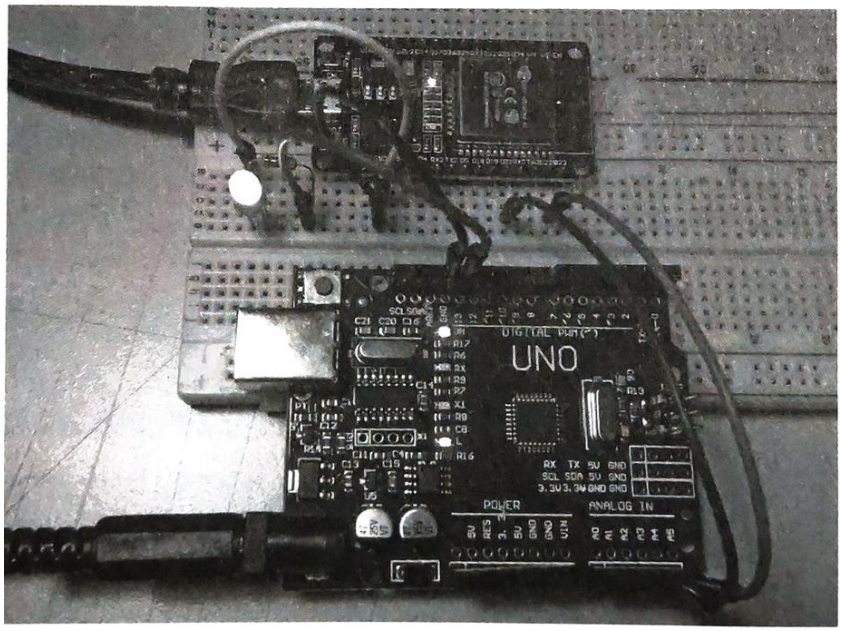
```
#include <Wire.h>

void setup() {
  pinMode(13, OUTPUT);          //กำหนดให้ขา 13 มีสถานะเป็น OUTPUT
  Wire.begin(11);               //กำหนดให้อุปกรณ์ปัจจุบันเป็น Slave ด้วยการติดต่อผ่านหมายเลข
                                แอดเดรสที่เรากำหนด ในที่นี้คือ 11 หรือ 0xB (ตามมาตรฐาน I2C หมายเลข
                                แอดเดรสที่เรากำหนดจะต้องมีขนาด 7-bit หรือเป็นค่าตัวเลขอะไรก็ได้ที่อยู่
                                ระหว่าง 0-127 แต่จะต้องไม่ซ้ำกัน และหากปล่อยวงเล็บให้ว่างไว้โดยไม่ใส่
                                แอดเดรสจะเป็นการกำหนดให้บอร์ดตัวนั้นทำหน้าที่เป็น Master)
}
void loop() {
  Wire.onReceive(myHandler);    //เมื่ออุปกรณ์ Slave ได้รับข้อมูลแล้ว ให้ไปเรียกใช้ฟังก์ชั่น myHandler
  delay(300);                   //หน่วงรอเป็นเวลา 300 ms
}
void myHandler(int numByte) {   //ประกาศฟังก์ชั่น myHandler โดยผ่านค่าพารามิเตอร์ที่
                                เป็นจำนวนไบต์ของข้อมูลที่ได้รับจากอุปกรณ์ Master
  while(Wire.available()) {     //ตรวจสอบว่ายังมีการส่งข้อมูลมาจากอุปกรณ์ Master
    char c = Wire.read();       //อ่านข้อมูลที่ได้รับเก็บไว้ในตัวแปร c
    if (c == '1') {             //ตรวจสอบค่าในตัวแปร c ถ้ามีค่าเป็นตัวอักษร 1
      digitalWrite(13, HIGH);   //กำหนดให้ขา 13 มีสถานะเป็น HIGH หลอดไฟติด
    } else if (c == '0') {      //หรือไม่เช่นนั้น ถ้าค่าในตัวแปร c มีค่าเป็นตัวอักษร 0
      digitalWrite(13, LOW);    //กำหนดให้ขา 13 มีสถานะเป็น LOW หลอดไฟดับ
    }
  }
}
```



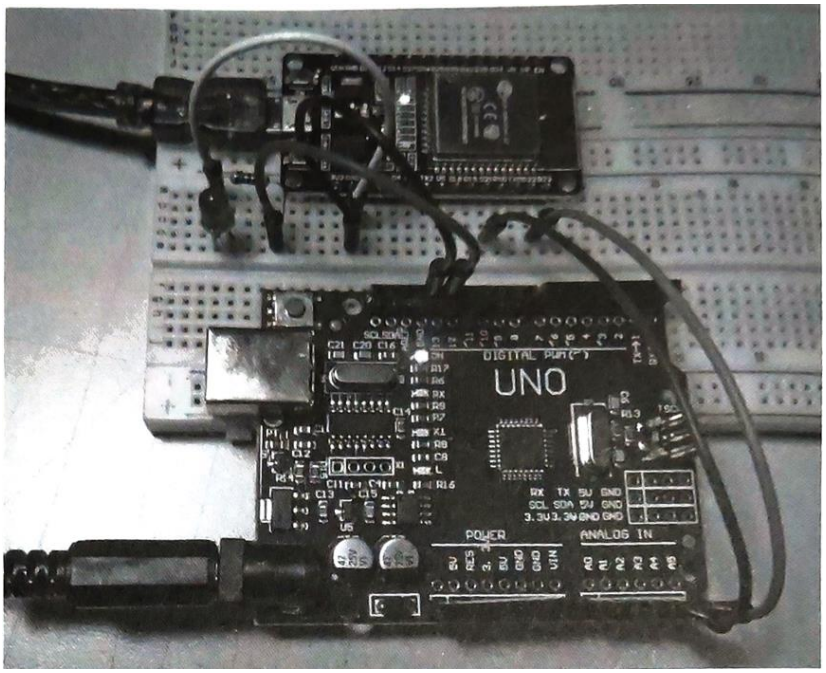
หลังจากอัพโหลดโค้ดโปรแกรมให้กับบอร์ดทั้ง 2 แล้ว จากนั้นเสียบสาย USB ให้กับบอร์ดทั้ง 2 เพื่อจ่ายไฟเลี้ยง

จากนั้นให้เราเปิดหน้าต่าง Serial Monitor ของบอร์ด NodeMCU ที่เป็น Master แล้วทดสอบด้วยการพิมพ์คำว่า “ON” แล้วกดคีย์ Enter หรือกดปุ่ม Send

ถ้าทุกอย่างถูกต้อง หลอดไฟ LED ที่เชื่อมต่ออยู่กับบอร์ด Arduino Uno ที่เป็น Slave จะติด (ดังรูป)



พิมพ์คำว่า “OFF” แล้วกดคีย์ Enter หรือกดปุ่ม Send ถ้าทุกอย่างถูกต้อง หลอดไฟ LED ที่เชื่อมต่ออยู่ กับบอร์ด Arduino Uno ที เป็น Slave จะดับ (ดังรูป)



บันทึกผลการทดลอง

.....

.....

.....

.....

การทดลองที่ 3 การแสดงข้อความออกทางโมดูล LCD

อุปกรณ์ในการทดลอง

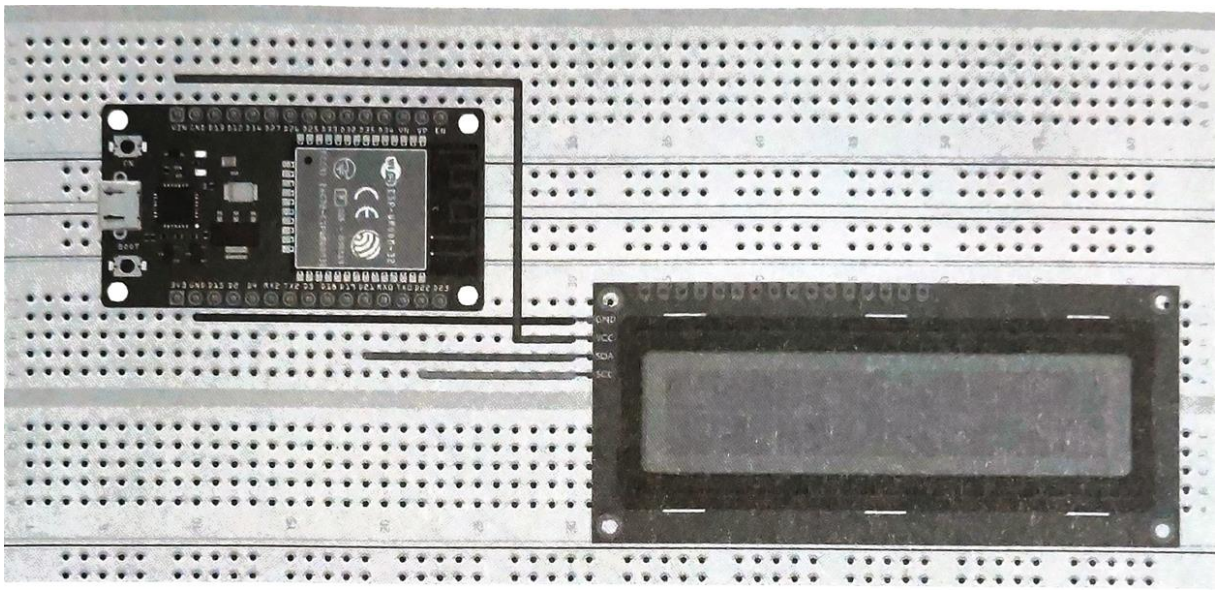
1.บอรด์ NodeMCU ESP32

2.โมดูลจอ LCD ขนาด 16x2

3.แผงต่อวงจร

4.สายไฟต่อวงจร

ประกอบวงจรตามรูป



ลงมือต่อวงจรตามรูป ซึ่งเป็นผังการต่อวงจรจากตัวอย่างที่แล้ว หลังจากเชื่อมต่อสายสัญญาณต่างๆ เสร็จ ขั้นตอนต่อไปก็คือ การเขียนโค้ดและอัพโหลดโปรแกรม แต่ในที่นี้เนื่องจากโมดูลจอ LCD จะเป็นอุปกรณ์ที่มี คอนโทรลเลอร์สำหรับควบคุมการแสดงผลมาให้อยู่แล้ว ดังนั้นการจะติดต่อกับโมดูลจอ LCD ได้นั้น จำเป็นจะ ต้องสั่งงานผ่านไลบรารี่ (Library) ที่ชื่อว่า LiquidCrystal_I2C และในกรณีนี้เนื่องจากบอร์ด MCU ที่เราใช้เป็น NodeMCU ESP32 ดังนั้นไลบรารี่ LiquidCrystal_I2C ที่ใช้ จะต้องสนับสนุนบอร์ด MCU นี้ด้วย เพราะฉะนั้นก่อนที่เราจะลงมือเขียนโค้ด เราจึงต้องมีการติดตั้งไลบรารี่ (Library) ดังกล่าวนี้ให้กับโปรแกรม Arduino IDE เสียก่อน ซึ่งการดาวน์โหลดและติดตั้งมีขั้นตอนง่ายๆ ดังนี้

1. ไปที่เว็บไซต์ https://github.com/nhatuan84/esp32-lcd หรือค้นหาแหล่งดาวน์โหลดจาก Google โดยใช้คำค้นหาว่า LiquidCrystal_I2C ESP32 ก็ได้ จากนั้นไฟล์ที่ดาวน์โหลดมาจะเป็นไฟล์ zip เช่น esp32-lcd-master.zip เป็นต้น

2. ติดตั้งไลบรารี่ (Library) ให้กับ Arduino IDE โดยใช้ตัว Manager ของ Arduino IDE ด้วยการคลิกที่เมนู Sketch - Include Library - Add .ZIP Library.... หรือจะใช้วิธีแบบ Manual โดยแตกซิป แล้วเปลี่ยนชื่อโฟลเดอร์เป็น LiquidCrystal_I2C จากนั้นค่อยก็อปปี้โฟลเดอร์นี้ไปวางไว้ที่ Documents\Arduino Libraries ก็ได้ แต่วิธีนี้หลังจากทำเสร็จอาจจะต้องปิดและเปิด Arduino IDE ใหม่ เพื่อให้โปรแกรมมอง เห็นไลบรารี่ที่เพิ่มเข้าไป

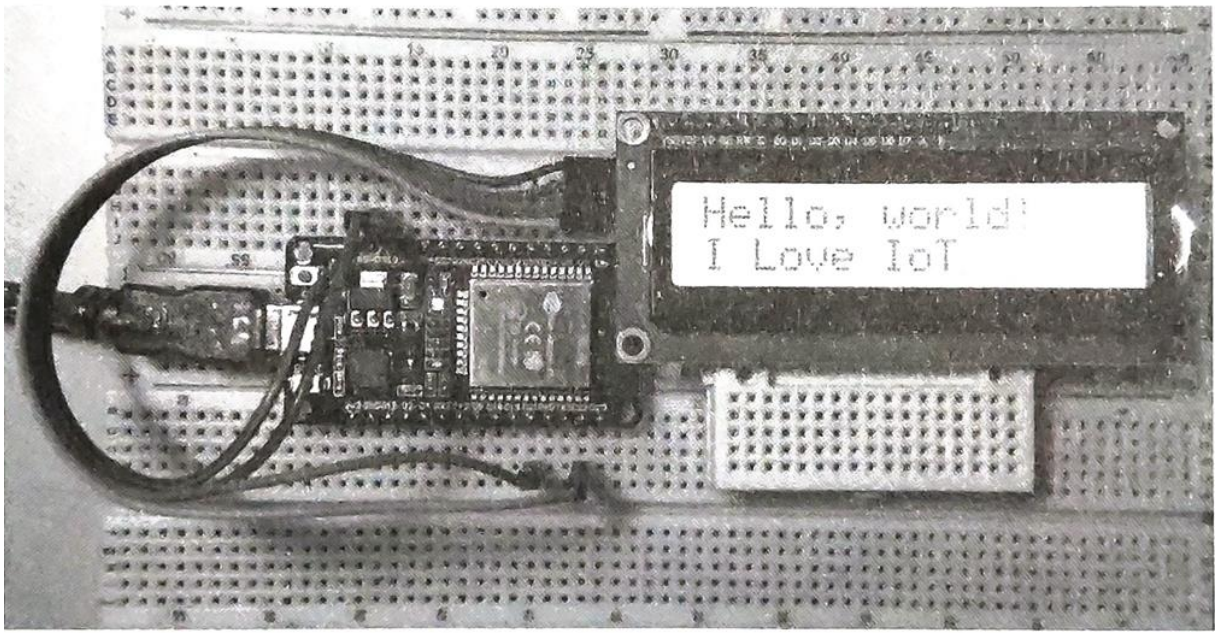
3. สุดท้ายหลังจากติดตั้งไลบรารี่ (Library) เสร็จ ก็ให้เข้าไปตรวจสอบที่ Arduino IDE ดูว่ามีชื่อไลบรารี่ที่เรา เพิ่งจะติดตั้งถูกเพิ่มเข้าไปรายชื่อไลบรารี่ทั้งหมดแล้วหรือยัง โดยคลิกไปที่เมนู Sketch » Include Library หรือจะคลิกดูที่เมนู File > Examples ก็ได้ ซึ่งในกรณีนี้จะต้องปรากฏชื่อไลบรารี่ว่า esp32-lcd-master ถ้าพบแล้วแสดงว่าการติดตั้งเสร็จเรียบร้อย พร้อมใช้งานแล้ว

หลังจากติดตั้งไลบรารี่ (Library) เพื่อให้บอร์ด NodeMCU ESP32 สามารถติดต่อกับโมดูลจอ LCD ได้ เสร็จเรียบร้อยแล้ว ก็ถึงเวลาที่เราจะต้องมาลงมือเขียนโค้ดและอัพโหลดโปรแกรมซักที

ในตัวอย่างนี้เริ่มต้นให้เราทดลองส่งข้อความ Hello, world! และ I Love IoT ไปแสดงบนหน้าจอ LCD โดยในที่นี้ จะใช้โค้ดโปรแกรมจากไฟล์ตัวอย่างที่ให้มา เริ่มต้นที่ Arduino IDE คลิกเมนู File » Examples » esp32-lcd-master » esp32lcd จะปรากฏหน้าต่างแสดงโค้ดโปรแกรม ดังนี้

```
#include <Wire.h>                          //เรียกใช้ไลบรารี่ Wire.h สำหรับการสื่อสารแบบ I2C
#include <LiquidCrystal_I2C.h>             //เรียกใช้ไลบรารี่ LiquidCrystal_I2C.h เพื่อควบคุมการแสดงผลบนจอ LCD

LiquidCrystal_I2C lcd(0x27, 16, 2);        //สร้างออบเจ็กต์จากคลาส LiquidCrystal_I2C แล้วนำไปเก็บไว้ในตัวแปร lcd โดยจะต้องผ่านค่าแอดเดรสของโมดูล LCD (ในที่นี้แอดเดรส คือ 0x27) และขนาดของหน้าจอ LCD (แสดงผล 2 แถว แถวละ 16 ตัวอักษร)

void setup() {
  lcd.begin(21, 22);                       //เริ่มต้นการสื่อสารกับโมดูลจอ LCD ผ่านขา SDA, SCL
  lcd.backlight();                         //เปิดไฟ backlight บนหน้าจอ LCD
  lcd.setCursor(0,0);                      //เลื่อนเคอร์เซอร์ไปที่ตำแหน่งคอลัมน์ 0 แถว 0
  lcd.print("Hello, world!");              //แสดงข้อความใน " " ออกทางหน้าจอ LCD
  lcd.setCursor(0,1);                      //เลื่อนเคอร์เซอร์ไปที่ตำแหน่งคอลัมน์ 0 แถว 1
  lcd.print("I Love IoT");                 //แสดงข้อความใน " " ออกทางหน้าจอ LCD
}

void loop() {
}
```

จากนั้นให้อัพโหลดโค้ดโปรแกรมลงไปบนบอร์ด จะเห็นข้อความปรากฏบนหน้าจอ LCD ดังรูป



บันทึกผลการทดลอง

.....

.....

.....

.....

การทดลองที่ 4 การปรับแต่งและควบคุมการแสดงผลบนหน้าจอ LCD

จากตัวอย่างที่ผ่านมา เราได้ทดลองเชื่อมต่อโมดูลจอ 16x2 Character LCD ที่มาพร้อม I2C LCD Controller เข้ากับบอร์ด NodeMCU ESP32 ซึ่งได้ทำการตรวจสอบหมายเลขแอดเดรสของโมดูลจอ LCD ที่ใช้ติดต่อผ่าน I2C บัส และได้ทดลองกันตัวอย่างโค้ดจากไลบรารี่ LiquidCrystal_I2C ที่ใช้แสดงข้อความ Hello, world! ออกทางหน้าจอของโมดูล LCD กันไปแล้ว มาถึงตัวอย่างนี้เราจะมาทดลองทำอะไรที่เกี่ยวข้องกับการแสดงผลบนหน้าจอ LCD ที่ซับซ้อนขึ้นไปอีก เช่น การแสดงข้อความ 2 บรรทัด, การแสดงตัวอักขระพิเศษ รูปหัวใจ, การเลื่อนข้อความไปทางซ้าย/ขวา, การควบคุมการปิด/เปิดแสดงข้อความและไฟ Backlight และการ ควบคุมตำแหน่งของเคอร์เซอร์ ซึ่งมีขั้นตอนต่างๆ ดังนี้

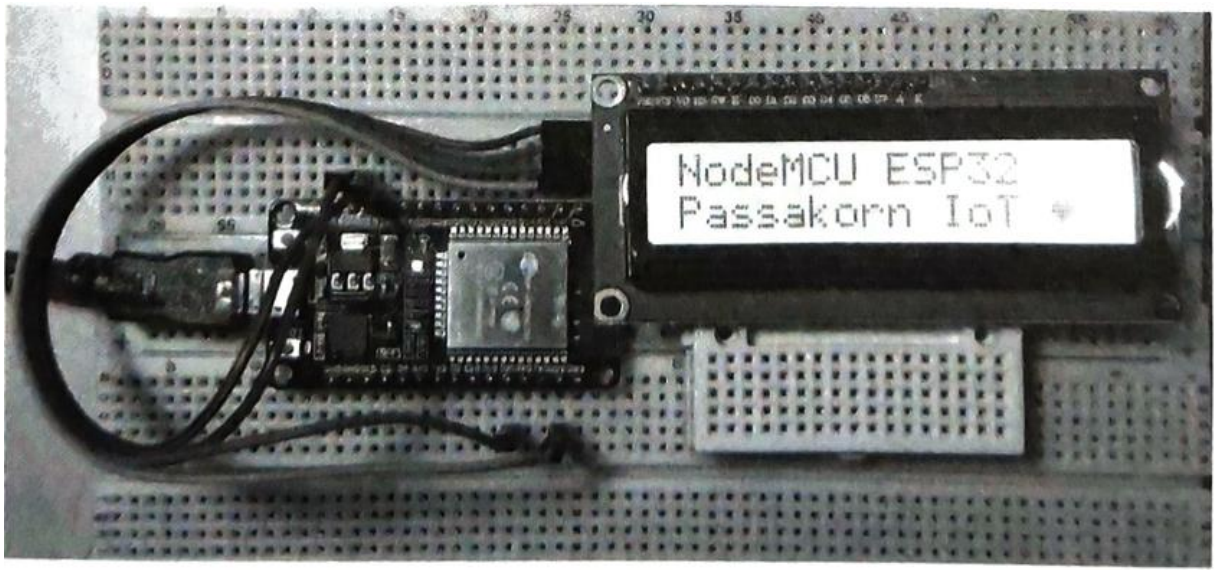
จากตัวอย่างที่แล้วการเชื่อมต่ออุปกรณ์ยังคงเหมือนเดิม เพียงแต่ให้เราเอาโค้ดโปรแกรมมาแก้ไข ซึ่งมีรายละเอียดดังนี้

```
#include <Wire.h>                          //เรียกใช้ไลบรารี่ Wire.h สำหรับการสื่อสารแบบ I2C
#include <LiquidCrystal_I2C.h>             //เรียกใช้ไลบรารี่ LiquidCrystal_I2C.h เพื่อควบคุมการ
                                           แสดงผลบนจอ LCD

LiquidCrystal_I2C lcd(0x27, 16, 2);        //สร้างออบเจ็กต์จากคลาส LiquidCrystal_I2C แล้ว
                                           นำไปเก็บไว้ในตัวแปร lcd โดยจะต้องผ่านค่าแอดเดรส
                                           ของโมดูล LCD (ในที่นี้แอดเดรสคือ 0x27) และขนาดของ
                                           หน้าจอ LCD (แสดงผล 2 แถว แถวละ 16 ตัวอักษร)
byte heart[8] = {0x00,0x0A,0x1F,0x1F,0x0E,0x04,0x00,0x00};  //เก็บค่าตัวอักขระพิเศษ
                                                            รูปหัวใจไว้ในตัวแปร heart

void setup() {
  lcd.begin(21, 22);                       //เริ่มต้นการสื่อสารกับโมดูลจอ LCD ผ่านขา SDA, SCL
}

void loop() {
  lcd.backlight();                         //เปิดไฟ backlight
  lcd.display();                           //เปิดการแสดงผลจอ LCD
  lcd.home();                              //เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นที่ตำแหน่งคอลัมน์ 0
                                           (ซ้ายสุด) แถว 0 (บรรทัดบน)
  delay(1000);                             //หน่วงเวลารอ 1 วินาที
  lcd.print("NodeMCU ESP32");              //แสดงข้อความใน " " ออกทางหน้าจอ LCD
  delay(2000);                             //หน่วงเวลารอ 2 วินาที
  lcd.setCursor(0, 1);                     //เลื่อนเคอร์เซอร์ไปที่ตำแหน่งคอลัมน์ 0 แถว 1 (บรรทัดล่าง)
  lcd.print("Passakorn");                  //แสดงข้อความใน " " ออกทางหน้าจอ LCD
  delay(2000);                             //หน่วงเวลารอ 2 วินาที
```

```
  lcd.setCursor(10, 1);              //เลื่อนเคอร์เซอร์ไปที่ตำแหน่งคอลัมน์ 10 แถว 1
  lcd.print("IoT");                  //แสดงข้อความใน " " ออกทางหน้าจอ LCD
  delay(2000);                       //หน่วงเวลารอ 2 วินาที

  lcd.createChar(0, heart);          //สร้างตัวอักขระพิเศษจากค่าในตัวแปร heart เก็บไว้ที่ 0
  lcd.setCursor(14, 1);              //เลื่อนเคอร์เซอร์ไปที่ตำแหน่งคอลัมน์ 14 แถว 1
  lcd.write(0);                      //แสดงตัวอักขระพิเศษที่เก็บไว้ที่ 0 ออกทางหน้าจอ LCD
  delay(2000);                       //หน่วงเวลารอ 2 วินาที

  for (int i = 0; i <= 15; i++) {    //ใช้คำสั่งวนรอบทำซ้ำ โดยเริ่มนับจาก 0 ไปเรื่อยๆ จนถึง 15
    lcd.scrollDisplayLeft();         //เลื่อนการแสดงผลบนหน้าจอไปทางซ้าย
    delay(200);                      //หน่วงเวลารอ 0.2 วินาที
  }
  for (int i = 0; i <= 15; i++) {    //ใช้คำสั่งวนรอบทำซ้ำ โดยเริ่มนับจาก 0 ไปเรื่อยๆ จนถึง 15
    lcd.scrollDisplayLeft();         //เลื่อนการแสดงผลบนหน้าจอไปทางซ้าย
    delay(200);                      //หน่วงเวลารอ 0.2 วินาที
  }
  delay(2000);                       //หน่วงเวลารอ 2 วินาที
  lcd.noDisplay();                   //ปิดการแสดงผลจอ LCD (ไม่แสดงข้อมูลใดๆ)
  delay(500);                        //หน่วงเวลารอ 0.5 วินาที
  lcd.display();                     //เปิดการแสดงผลจอ LCD (แสดงข้อมูล)
  delay(500);                        //หน่วงเวลารอ 0.5 วินาที
  lcd.noDisplay();
  delay(500);
  lcd.display();
  delay(500);
  lcd.noDisplay();
  delay(500);
  lcd.display();
  delay(1000);

  lcd.clear();                       //ลบข้อความบนหน้าจอ
  delay(1000);                       //หน่วงเวลารอ 1 วินาที
  lcd.noDisplay();                   //ปิดการแสดงผลจอ LCD (ไม่แสดงข้อมูลใดๆ)
  delay(1000);                       //หน่วงเวลารอ 1 วินาที
  lcd.noBacklight();                 //ปิดไฟ backlight
  delay(2000);                       //หน่วงเวลารอ 2 วินาที
}
```

การทำงานของโปรแกรม เริ่มต้นไฟ Backlight ของจอ LCD จะถูกเปิด พร้อมแสดงข้อความ Node MCU ESP32 ไว้ที่จุดเริ่มต้นหรือก็คือ คอลัมน์ซ้ายสุดของบรรทัดแรก รอเวลา 2 วินาที จากนั้นเคอร์เซอร์ถูกกำหนด ให้ไปรอที่ตำแหน่งคอลัมน์ซ้ายสุดของบรรทัดถัดมา พร้อมแสดงข้อความ Passakorn รอเวลา 2 วินาที ต่อมา เคอร์เซอร์ถูกกำหนดให้ไปรอที่ตำแหน่งคอลัมน์ 10 (นับจากคอลัมน์ซ้ายสุดคือ 0 มาทางขวาทีละ 1 จนถึงคอลัมน์ ที่ 10) ของบรรทัดเดิม พร้อมแสดงข้อความ IoT จากนั้นเคอร์เซอร์ถูกกำหนดให้ไปรอที่ตำแหน่งคอลัมน์ 14 ของบรรทัดเดียวกัน พร้อมแสดงตัวอักขระรูปหัวใจจากค่าที่เรากำหนด ก่อนจะเลื่อนการแสดงผลบนหน้าจอ ไปทางซ้ายจำนวน 15 คอลัมน์ ใช้เวลาคอลัมน์ละ 0.2 วินาที จากนั้นเลื่อนกลับมาทางขวาจำนวน 32 คอลัมน์ และสุดท้ายเลื่อนกลับไปทางซ้ายจำนวน 16 คอลัมน์ ซึ่งก็คือจุดเริ่มต้นในตอนแรก จากนั้นรอเวลา 2 วินาที แล้ว ตามด้วยการกะพริบข้อความทั้งหมดบนหน้าจอ 3 ครั้ง ก่อนจะลบข้อความและปิดการแสดงผลทั้งหมด สุดท้าย ปิดไฟ Backlight บนหน้าจอ และหลังจากนั้น 2 วินาที ก็จะเริ่มต้นการทำงานทั้งหมดใหม่อีกครั้ง วนไปเรื่อยๆ

หลังจากอัพโหลดโค้ดโปรแกรมลงไปบนบอร์ดเรียบร้อยแล้ว ผลลัพธ์ที่ได้จากการแสดงผลบนหน้าจอ LCD จะเป็นดังรูป



บันทึกผลการทดลอง

.....

.....

.....

.....

การทดลองที่ 5 การแสดงข้อความออกทางโมดูลจอ OLED

อุปกรณ์ในการทดลอง

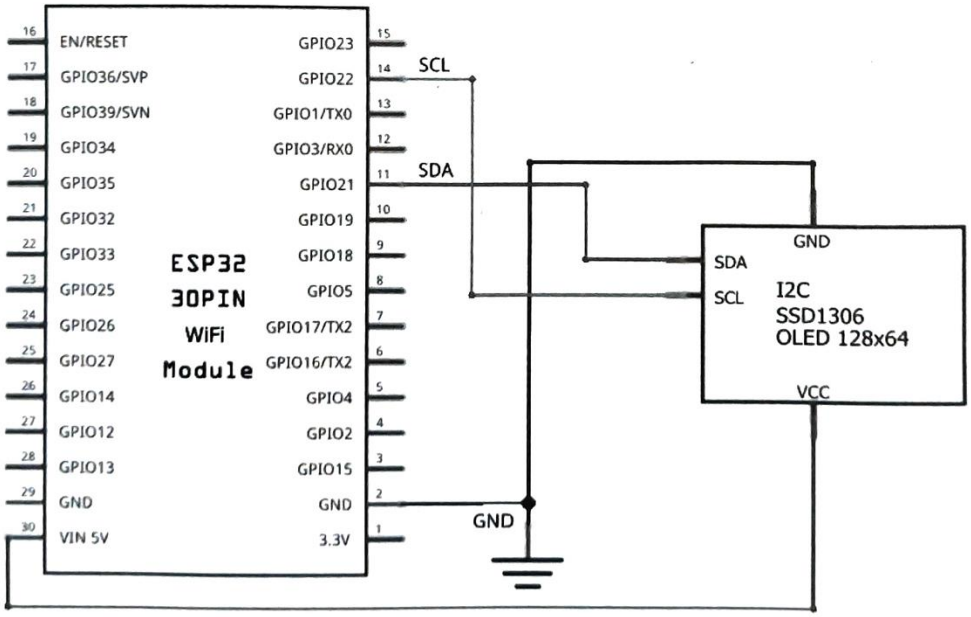
1.บอรด์ NodeMCU ESP32

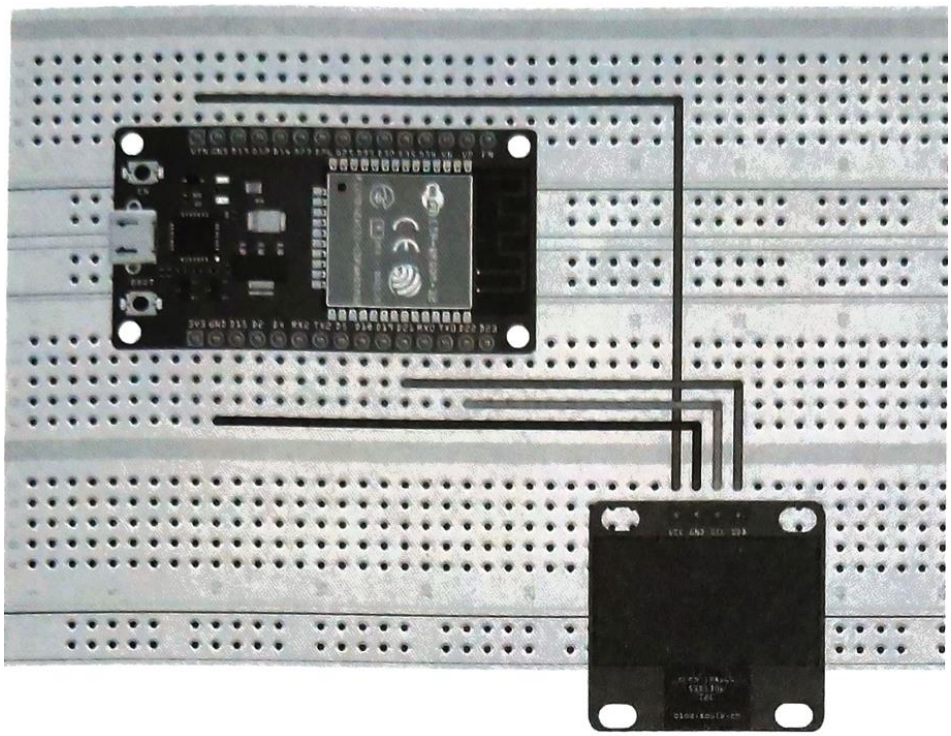
2.โมดูลจอ OLED

3.แผงต่อวงจร

4.สายไฟต่อวงจร

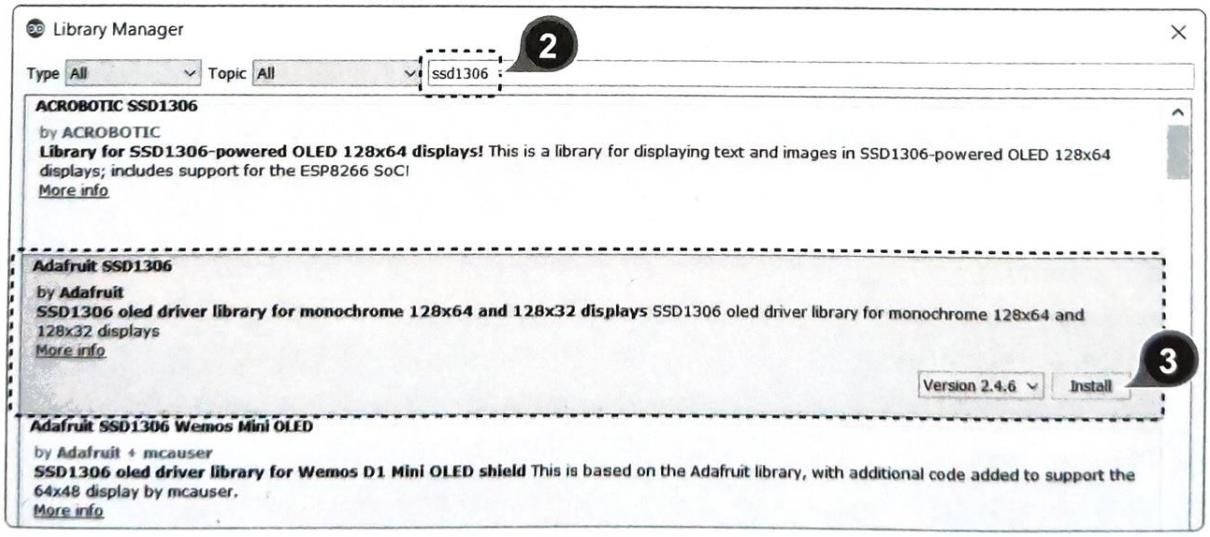
ประกอบวงจรตามรูป



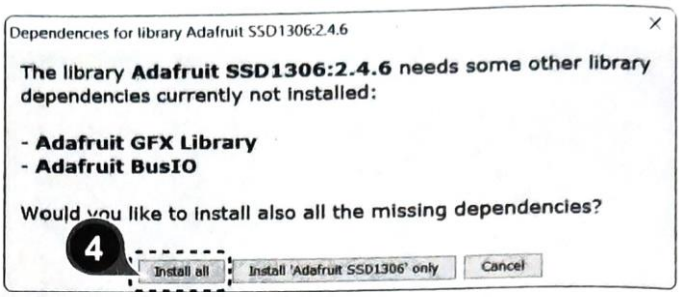


หลังจากลงมือเชื่อมต่อสายสัญญาณต่างๆ เรียบร้อยแล้ว ขั้นตอนต่อไปก็คือ การเขียนโค้ดและอัพโหลดโปรแกรม แต่ในที่นี้เนื่องจากโมดูลจอ OLED จะมีชิป IC Controller ที่ใช้ควบคุมการแสดงผลมาให้ในตัว เพราะฉะนั้นก่อนจะลงมือเขียนโค้ด เราจะต้องติดตั้งไลบรารี่ (Library) ที่จำเป็นให้กับ Arduino IDE เสียก่อน ซึ่งมีขั้นตอนสั้นๆ คือ

1. Arduino IDE คลิกเมนู Tools » Manage Libraries... หรือกดคีย์ Ctrl + Shift + I
2. ในช่อง Search พิมพ์คำว่า ssd1306 แล้วกดคีย์ Enter
3. ที่ไลบรารี่ Adafruit SSD1306 คลิกปุ่ม Install เพื่อติดตั้ง



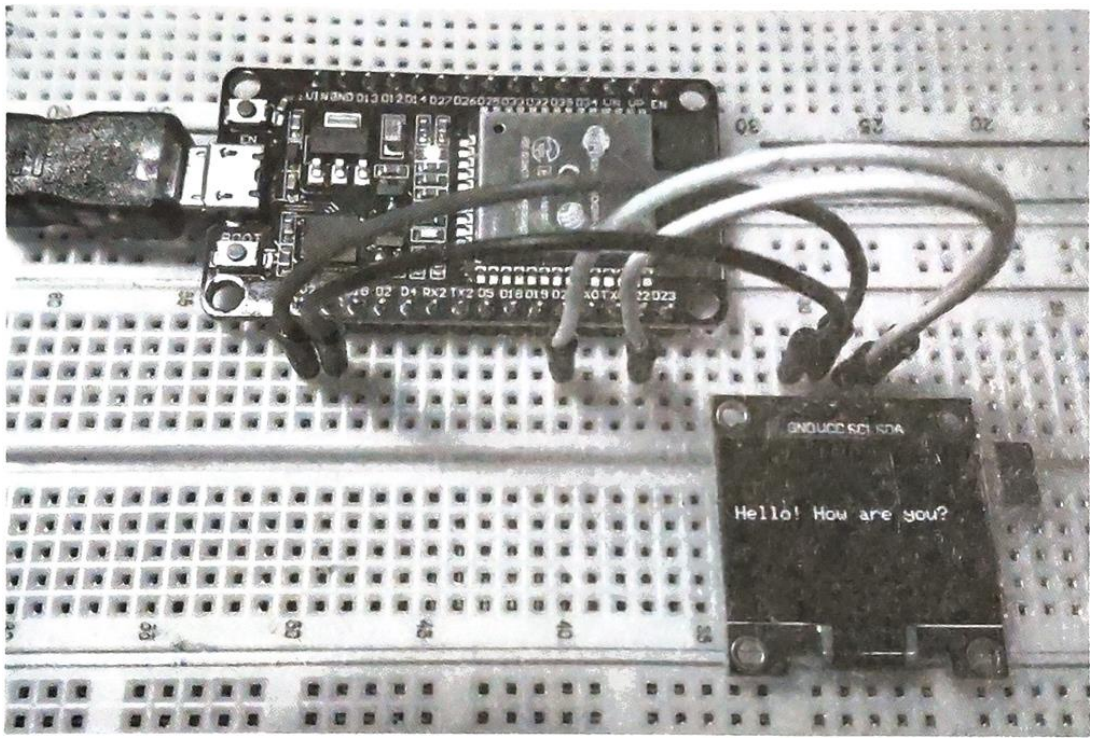
4. จะปรากฏหน้าต่างแจ้งว่าไลบรารี่นี้จำเป็น ต้องใช้งานไลบรารี่อื่นๆ เช่น Adafruit GFX Library และ Adafruit BusIO ร่วมด้วยซึ่งยังไม่ได้ถูกติดตั้ง ในที่นี้ให้เราเลือกว่า จะติดตั้งทั้งหมด คลิกปุ่ม Install all




```
#include <SPI.h>                          //เรียกใช้ไลบรารี่ SPI.h สำหรับการสื่อสารแบบ SPI
#include <Wire.h>                         //เรียกใช้ไลบรารี่ Wire.h สำหรับการสื่อสารแบบ I2C
#include <Adafruit_GFX.h>                 //เรียกใช้ไลบรารี่เพื่อควบคุมการแสดงผลบนจอ OLED
#include <Adafruit_SSD1306.h>             //เรียกใช้ไลบรารี่เพื่อควบคุมการแสดงผลบนจอ OLED

#define SCREEN_WIDTH 128                  //กำหนดขนาดความกว้างของหน้าจอแสดงผลเป็น pixel
#define SCREEN_HEIGHT 64                  //กำหนดขนาดความสูงของหน้าจอแสดงผลเป็น pixel
#define OLED_RESET 4                      //ประกาศให้ขา 4 เป็น OLED_RESET

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

```
void setup() {
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //เริ่มต้นใช้งาน I2C ผ่านแอดเดรส 0x3C
  display.clearDisplay();                    //เคลียร์หน้าจอ

//Display Text
  display.setTextSize(1);                    //กำหนดขนาดตัวอักษรเป็น 1
  display.setTextColor(WHITE);               //กำหนดสีตัวอักษรเป็นสีขาว (กรณีพื้นหลังเป็นสีเข้ม)
  display.setCursor(0, 32);                  //กำหนดตำแหน่งเคอร์เซอร์ไว้ที่พิกัด x = 0, y = 32
  display.println("Hello! How are you?");    //แสดงข้อความในวงเล็บออกทางหน้าจอ
                                             แล้วขึ้นบรรทัดใหม่
  display.display();                         //ดึงข้อมูลจากบัฟเฟอร์มาแสดงผลออกทางหน้าจอ
}
void loop() {
}
```

หลังจากอัพโหลดโค้ดโปรแกรมลงไปบนบอร์ดเรียบร้อยแล้ว ผลลัพธ์ที่ได้จะเป็นดังรูป

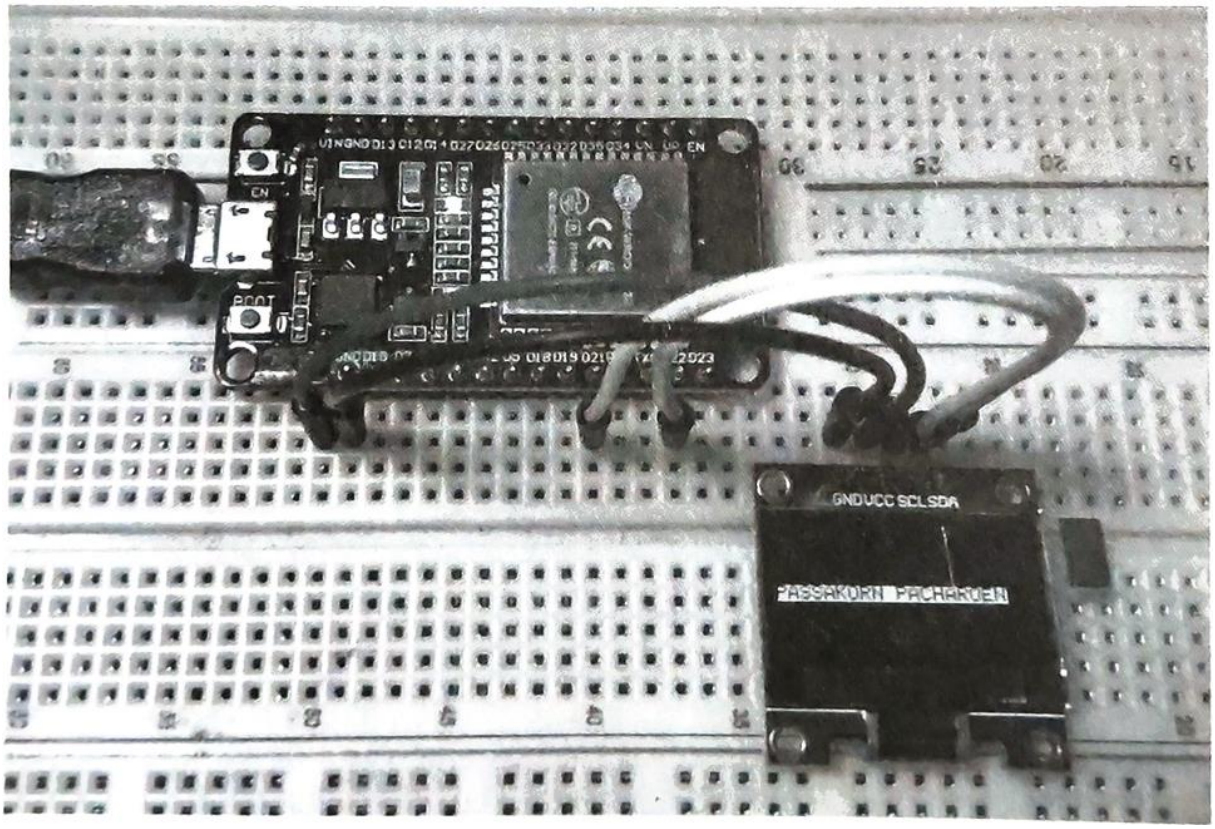


ต่อมาเราจะลองเพิ่มโค้ดเพื่อกลับสีตัวอักษรและพื้นหลังให้เป็นสีตรงกันข้ามดูบ้าง โดยจะให้แสดงผลลัพธ์ หลังจากแสดงข้อความก่อนหน้านี้ผ่านไปแล้ว 2 วินาที ให้เรานำโค้ดเดิมมาแก้ไข โดยใส่รายละเอียดดังในกรอบเพิ่มเติมเข้าไป

```
...
//Display Text
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 32);
```

```
  display.println("Hello! How are you?");
  display.display();
  delay(2000);                          //หน่วงรอเป็นเวลา 2 วินาที
display.clearDisplay();                 //เคลียร์หน้าจอ

//Display Inverted Text
  display.setTextColor(BLACK, WHITE);   //กำหนดสีตัวอักษรเป็นสีดำ พื้นหลังเป็นสีขาว
  display.setCursor(0, 32);             //กำหนดตำแหน่งเคอร์เซอร์ไว้ที่พิกัด
                                        x = 0, y = 32
  display.println("PASSAKORN PACHAROEN"); //แสดงข้อความในวงเล็บออกทางหน้าจอ
                                        แล้วขึ้นบรรทัดใหม่
  display.display();                    //ดึงข้อมูลจากบัฟเฟอร์มาแสดงผล
                                        ออกทางหน้าจอ
}

void loop() {}
```

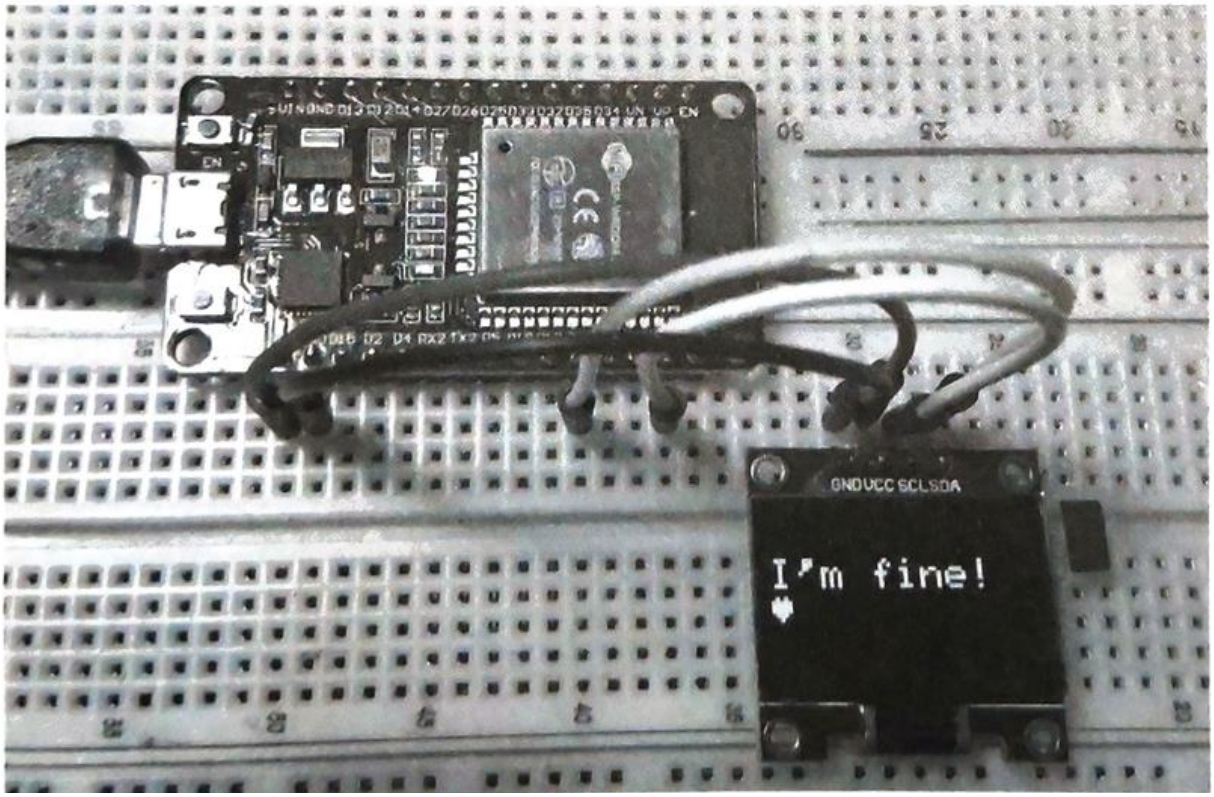
หลังจากอัพโหลดโค้ดโปรแกรมลงไปบนบอร์ดเรียบร้อยแล้ว ผลลัพธ์ที่ได้จะเป็นดังรูป



ต่อมาเราจะลองเพิ่มโค้ดเพื่อปรับขนาดตัวอักษรให้ใหญ่ขึ้น โดยจะให้แสดงผลลัพธ์หลังจากกลับสีตัวอักษร และพื้นหลังผ่านไปแล้ว 2 วินาที ให้เรานำโค้ดเดิมมาแก้ไข โดยใส่รายละเอียดดังในกรอบเพิ่มเติมเข้าไป

```
...
//Display Inverted Text
  display.setTextColor(BLACK, WHITE);
  display.setCursor(0, 32);
  display.println("PASSAKORN PACHAROEN");
  display.display();
  delay(2000);                          //หน่วงรอเป็นเวลา 2 วินาที
  display.clearDisplay();               //เคลียร์หน้าจอ

//Changing Font Size
  display.setTextColor(WHITE);          //กำหนดสีตัวอักษรเป็นสีขาว
  display.setCursor(0, 24);             //กำหนดตำแหน่งเคอร์เซอร์ไว้ที่พิกัด x = 0, y = 24
  display.setTextSize(2);               //กำหนดขนาดตัวอักษรเป็น 2
  display.print("I'm fine! ");          //แสดงข้อความในวงเล็บออกทางหน้าจอ
  display.write(3);                     //เลข 3 แทนรหัสตัวอักษร ASCII ที่มีสัญลักษณ์เป็นรูปหัวใจ
  display.display();                    //ดึงข้อมูลจากบัฟเฟอร์มาแสดงผลออกทางหน้าจอ
}

void loop() {}
```

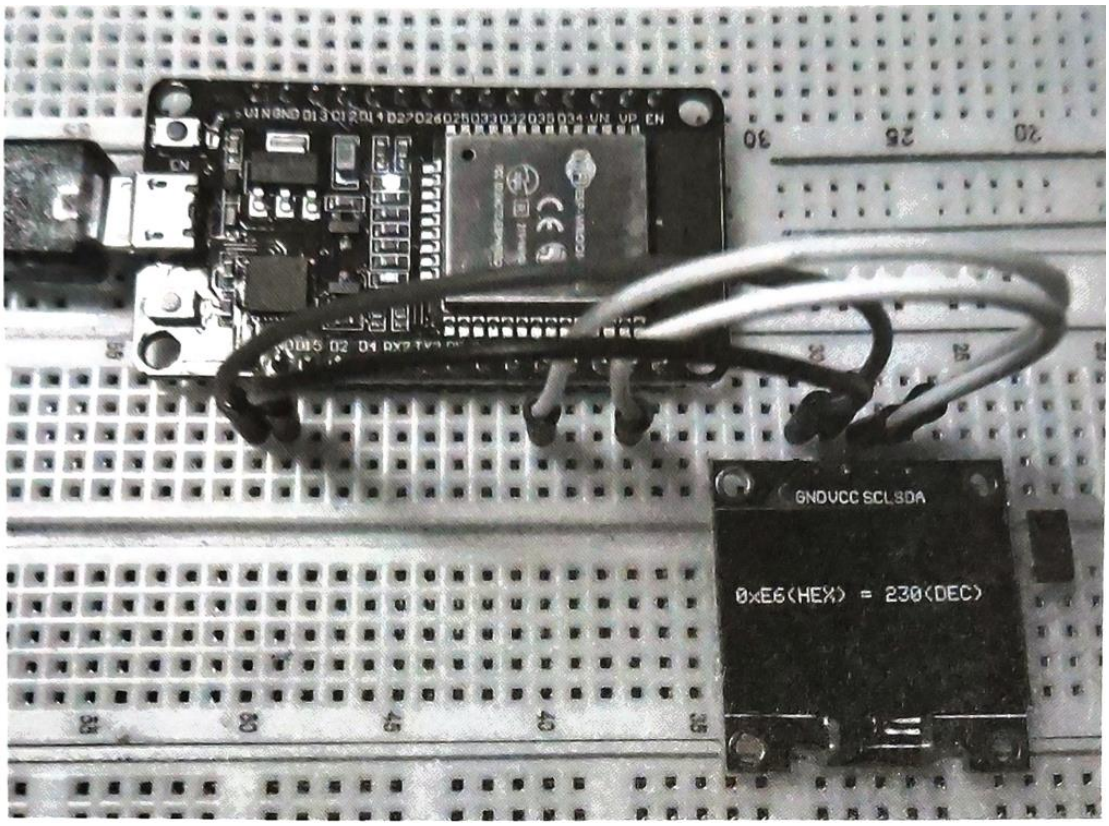
หลังจากอัพโหลดโค้ด โปรแกรมลงไปบนบอร์ด เรียบร้อยแล้ว ผลลัพธ์ที่ได้ จะเป็นดังรูป



ต่อมาเราจะลองเพิ่มโค้ดเพื่อแปลงค่าตัวเลข 0xE6 จากตัวเลขฐาน 16 ไปเป็นฐาน 10 พร้อมทั้งปรับขนาด ตัวอักษรให้เล็กลง โดยจะให้แสดงผลลัพธ์หลังจากที่ก่อนหน้านี้ ได้ปรับขนาดตัวอักษรให้ใหญ่ขึ้นผ่านไปแล้ว 2 วินาที ให้เรานำโค้ดเดิมมาแก้ไข โดยใส่รายละเอียดดังในกรอบเพิ่มเติมเข้าไป

```
...
//Changing Font Size
  display.setTextColor(WHITE);
  display.setCursor(0, 24);
  display.setTextSize(2);
  display.print("I'm fine ");
  display.write(3);
  display.display();
  delay(2000);                          //หน่วงรอเป็นเวลา 2 วินาที
  display.clearDisplay();               //เคลียร์หน้าจอ

  //Convert Base Number HEX to DEC
  display.setTextSize(1);               //กำหนดขนาดตัวอักษรเป็น 1
  display.setCursor(0, 32);             //กำหนดตำแหน่งเคอร์เซอร์ไว้ที่พิกัด x = 0, y = 32
  display.print("0x");                  //แสดงข้อความในวงเล็บออกทางหน้าจอ
  display.print(0xE6, HEX);             //แสดงค่า 0xE6 เป็นตัวเลขฐาน 16
  display.print("(HEX) = ");            //แสดงข้อความในวงเล็บออกทางหน้าจอ
  display.print(0xE6, DEC);             //แสดงค่า 0xE6 เป็นตัวเลขฐาน 10
  display.println("(DEC)");             //แสดงข้อความในวงเล็บออกทางหน้าจอ
  display.display();                    //ดึงข้อมูลจากบัฟเฟอร์มาแสดงผลออกทางหน้าจอ
}

void loop() {}
```

หลังจากอัพโหลดโค้ดโปรแกรมลงไปบนบอร์ดเรียบร้อยแล้ว ผลลัพธ์ที่ได้จะเป็นดังรูป



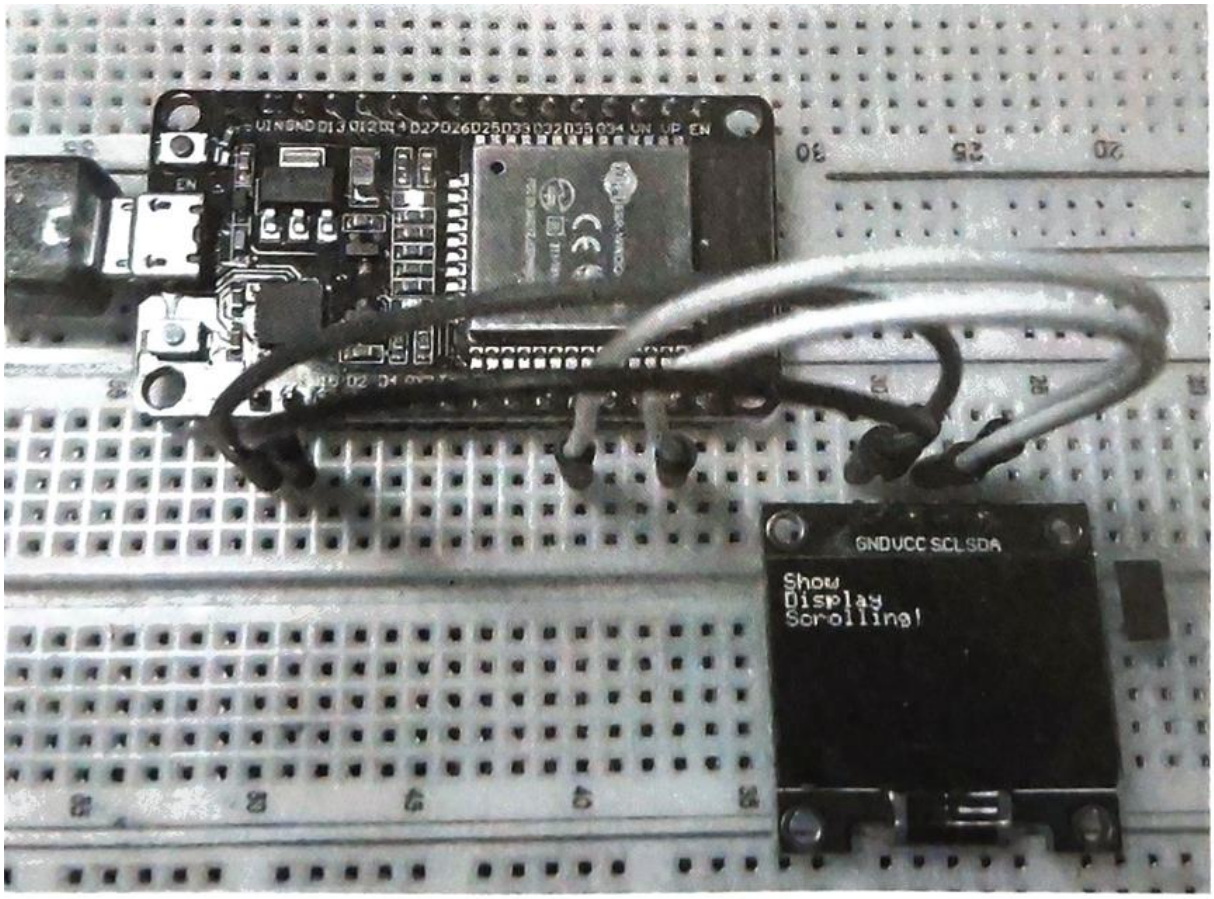
ต่อมาเราจะลองเพิ่มโค้ดเพื่อเลื่อนตำแหน่งข้อความทั้ง 3 บรรทัดบนหน้าจอไปทางขวา, ซ้าย, ทแยงขวา และซ้าย โดยจะให้แสดงผลลัพธ์หลังจากที่ได้แสดงการแปลงค่าตัวเลขผ่านไปแล้ว 2 วินาที ให้เรานำโค้ดเดิมมา แก้ไข โดยใส่รายละเอียดดังในกรอบเพิ่มเติมเข้าไป

```
...
  //Convert Base Number HEX to DEC
  display.setTextSize(1);
  display.setCursor(0, 32);
  display.print("0x");
  display.print(0xE6, HEX);
  display.print("(HEX) = ");
  display.print(0xE6, DEC);
  display.println("(DEC)");
  display.display();
```

```
  delay(2000);                    //หน่วงรอเป็นเวลา 2 วินาที
  display.clearDisplay();         //เคลียร์หน้าจอ

//Show Display Scrolling
  display.setCursor(0, 0);        //กำหนดตำแหน่งเคอร์เซอร์ไว้ที่พิกัด x = 0, y = 0
```

```
  display.println("Show");                    //พิมพ์ข้อความในวงเล็บออกทางหน้าจอ
  display.println("Display");                 //พิมพ์ข้อความในวงเล็บออกทางหน้าจอ
  display.println("Scrolling!");              //พิมพ์ข้อความในวงเล็บออกทางหน้าจอ
  display.display();                          //ดึงข้อมูลจากบัฟเฟอร์มาแสดงผลออกทางหน้าจอ
  display.startscrollright(0x00, 0x07);       //เลื่อนทั้ง 8 pages หรือทั้งหน้าจอไปทางขวา
  delay(4500);                                //หน่วงรอเป็นเวลา 4.5 วินาที
  display.stopscroll();                       //หยุดการเลื่อนหน้าจอ
  delay(1000);                                //หน่วงรอเป็นเวลา 1 วินาที
  display.startscrollleft(0x00, 0x07);        //เลื่อนทั้ง 8 pages หรือทั้งหน้าจอไปทางซ้าย
  delay(4500);                                //หน่วงรอเป็นเวลา 4.5 วินาที
  display.stopscroll();                       //หยุดการเลื่อนหน้าจอ
  delay(1000);                                //หน่วงรอเป็นเวลา 1 วินาที
  display.clearDisplay();                     //เคลียร์หน้าจอ
}

void loop() {}
```

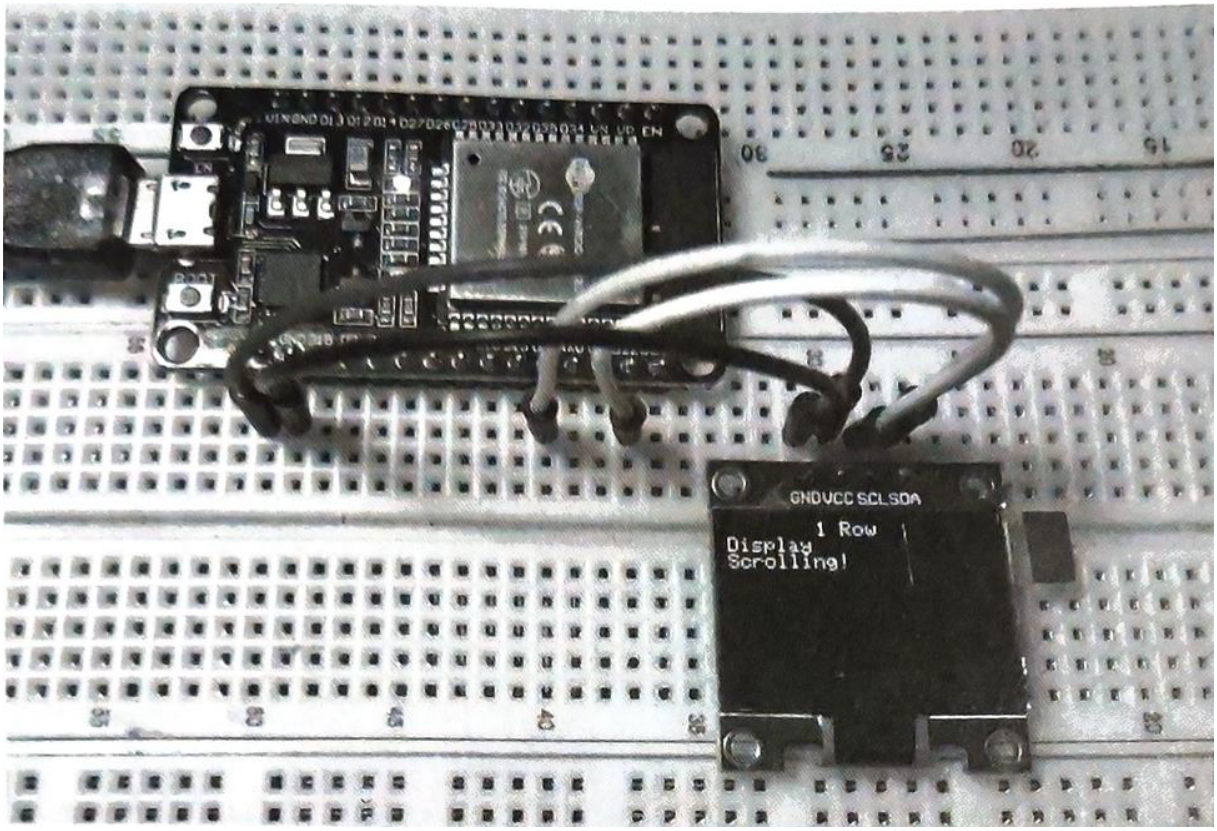
หลังจากอัพโหลดโค้ดโปรแกรมลงไปบนบอร์ดเรียบร้อยแล้ว ผลลัพธ์ที่ได้ จะเป็นดังรูป



ต่อมาเราจะลองเพิ่มโค้ดเพื่อเลื่อนเฉพาะข้อความในบรรทัดแรกไปทางขวา ให้เรานำโค้ดเดิมมาแก้ไข โดยใส่รายละเอียดดังในกรอบเพิ่มเติมเข้าไป

```
...
//Show Display Scrolling
  display.setCursor(0, 0);
  display.println("Show");
  display.println("Display");
  display.println("Scrolling!");
  display.display();
  display.startscrollright(0x00, 0x07);
  delay(4500);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x07);
  delay(4500);
  display.stopscroll();
  delay(1000);
  display.clearDisplay();
//1 Row Display Scrolling
  display.setCursor(0, 0);                  //กำหนดตำแหน่งเคอร์เซอร์ไว้ที่พิกัด x = 0, y = 0
  display.println("1 Row");                 //พิมพ์ข้อความในวงเล็บออกทางหน้าจอ
  display.println("Display");               //พิมพ์ข้อความในวงเล็บออกทางหน้าจอ
  display.println("Scrolling!");            //พิมพ์ข้อความในวงเล็บออกทางหน้าจอ
  display.display();
  display.startscrollright(0x00, 0x00);     //เลื่อนเฉพาะ pages 1 หรือบรรทัดแรกไปทางขวา
  delay(6000);
  display.stopscroll();                     //หยุดการเลื่อนหน้าจอ
  delay(1000);
  display.startscrollleft(0x00, 0x00);      //เลื่อนเฉพาะ pages 1 หรือบรรทัดแรกไปทางขวา
  delay(6000);
  display.stopscroll();                     //หยุดการเลื่อนหน้าจอ
  delay(1000);                              //หน่วงรอเป็นเวลา 6 วินาที
  display.clearDisplay();                   //เคลียร์หน้าจอ
}

void loop() {}
```

หลังจากอัพโหลดโค้ดโปรแกรมลงไปบนบอร์ดเรียบร้อยแล้ว ผลลัพธ์ที่ได้จะเป็นดังรูป

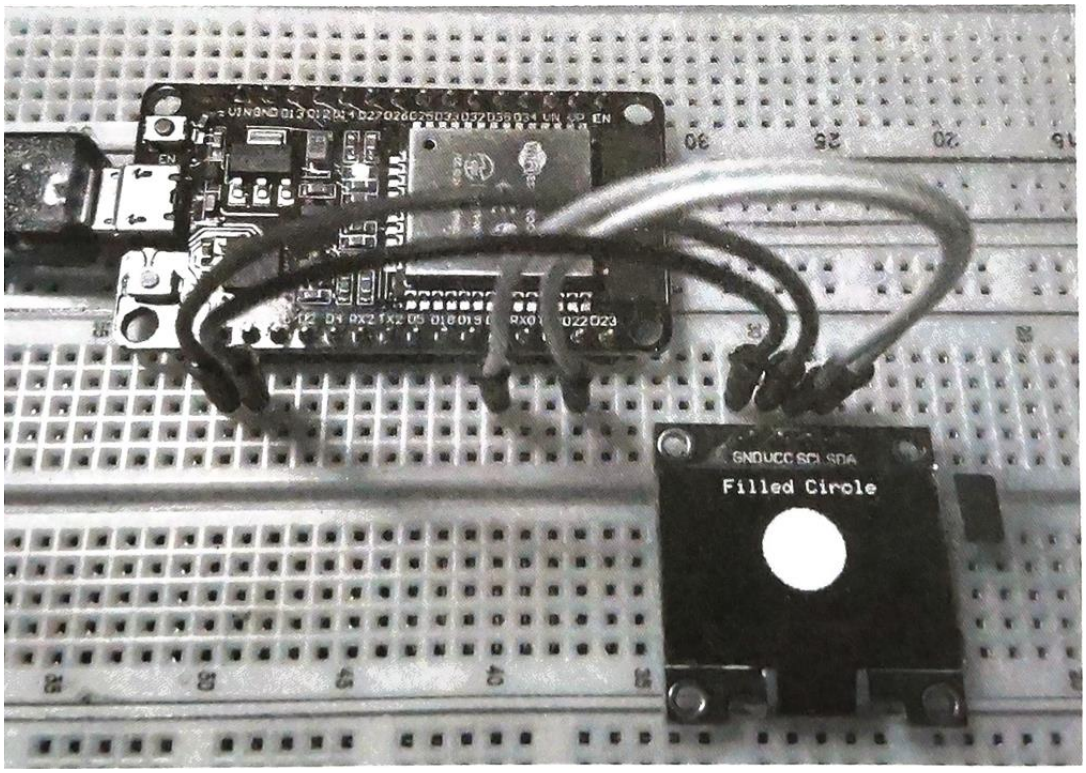


สุดท้ายเราจะลองเพิ่มโค้ดเพื่อวาดรูปพื้นที่วงกลมวางไว้กลางหน้าจอพร้อมแสดงข้อความกำกับไว้ที่ด้านบน ให้เรานำโค้ดเดิมมาแก้ไข โดยใส่รายละเอียดดังในกรอบเพิ่มเติมเข้าไป

```
...
//1 Row Display Scrolling
  display.setCursor(0, 0);
  display.println("1 Row");
  display.println("Display");
  display.println("Scrolling!");
  display.display();
  display.startscrollright(0x00, 0x00);
  delay(6000);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x00);
  delay(6000);
  display.stopscroll();
```

```
  delay(1000);
  display.clearDisplay();

//Drawing Filled Circle
  display.setCursor(27, 0);              //กำหนดตำแหน่งเคอร์เซอร์ไว้ที่พิกัด x = 27, y = 0
  display.setTextColor(WHITE);           //กำหนดสีตัวอักษรเป็นสีขาว
  display.println("Filled Circle");      //แสดงข้อความในวงเล็บออกทางหน้าจอ
                                         แล้วขึ้นบรรทัดใหม่
  display.fillCircle(64, 35, 20, WHITE); //วาดรูปพื้นที่วงกลมสีขาวที่มีจุดศูนย์กลางอยู่ที่
                                         พิกัด x = 64, y = 35 และมีรัศมี 20
  display.display();                     //ดึงข้อมูลจากบัฟเฟอร์มาแสดงผลออกทางหน้าจอ
  delay(2000);                           //หน่วงรอเป็นเวลา 2 วินาที
  display.clearDisplay();                //เคลียร์หน้าจอ
}

void loop() {}
```

หลังจากอัพโหลดโค้ดโปรแกรมลงไปบนบอร์ดเรียบร้อยแล้ว ผลลัพธ์ที่ได้จะเป็นดังรูป



บันทึกผลการทดลอง

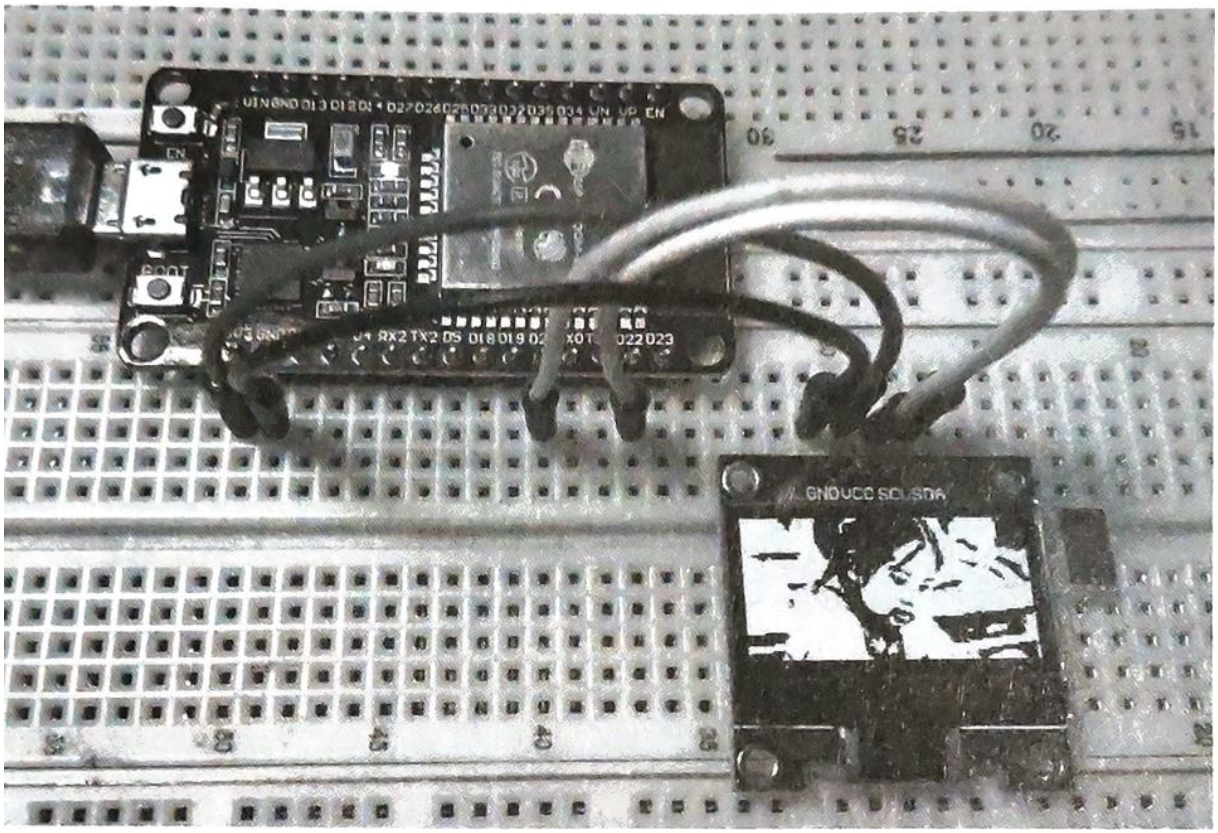
.....

.....

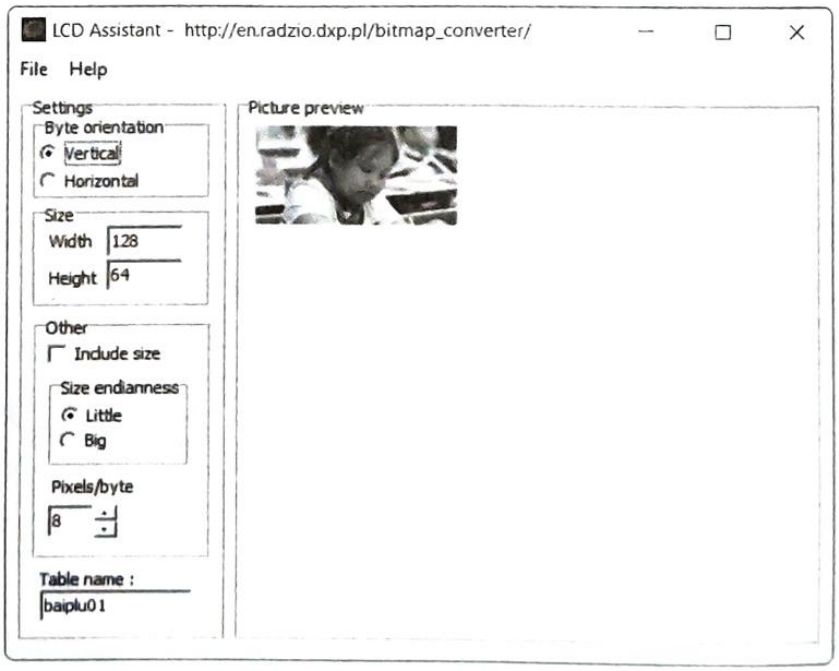
.....

.....

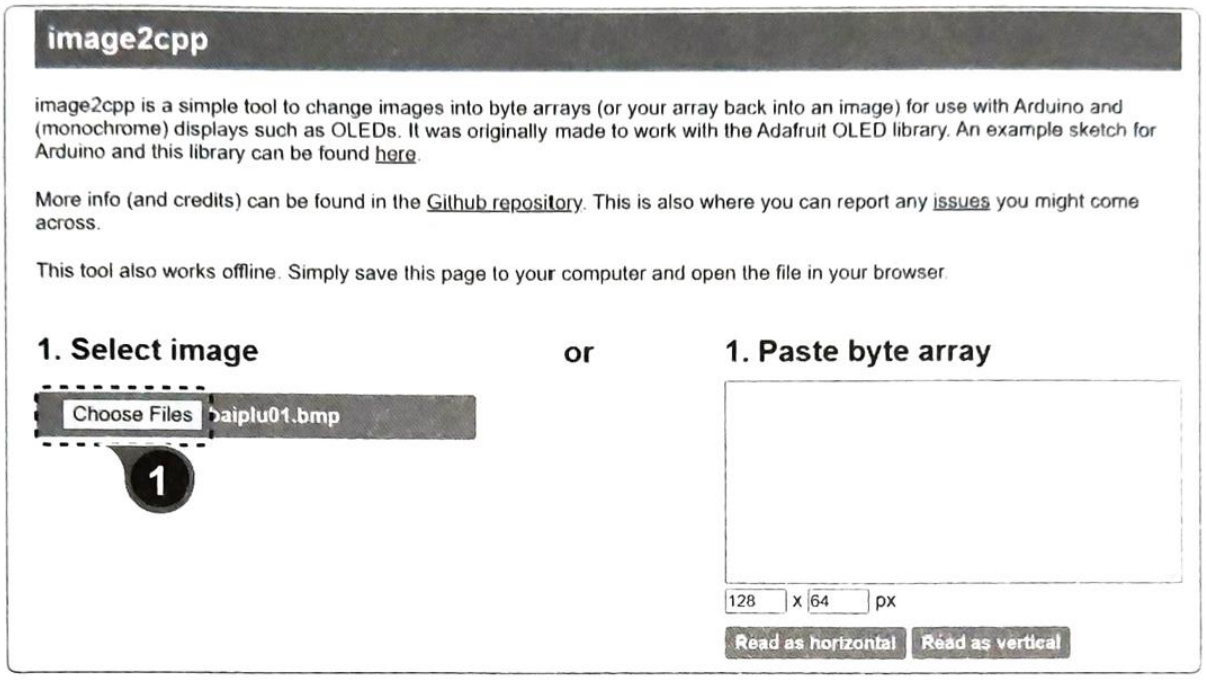
การทดลองที่ 6 การแสดงภาพกราฟิกออกทางโมดูลจอ OLED



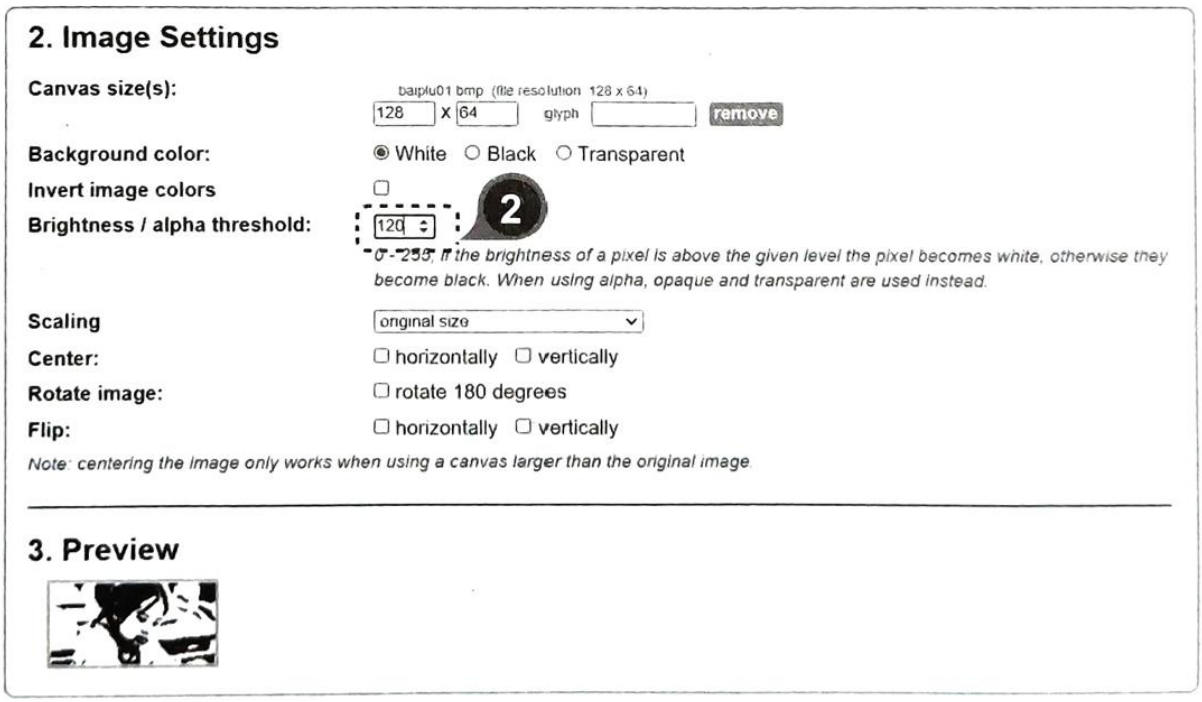
ก่อนอื่นเราจะต้องมีการเตรียม ภาพเสียก่อน โดยภาพนี้จะถูกนำไปแปลงให้เป็นโค้ดตัวแปรอาร์เรย์ เพื่อ ที่จะนำไปใช้ในโค้ดโปรแกรม สำหรับ ภาพที่นำมาใช้จะเป็นภาพอะไรก็ได้ แต่จะต้องมีการปรับให้มีขนาดเท่ากับ หน้าจอคือ 128x64 pixels และมีการ ปรับสีให้เป็นขาว-ดำ (Black & White) ก่อน จากนั้นจึงค่อย Save ภาพให้เป็น ไฟล์ Bitmap (.bmp) แล้วนำไปแปลง ให้เป็นโค้ด สำหรับนำไปใช้เป็นตัวแปร อาร์เรย์ ซึ่งการแปลงจะมีให้เลือกใช้อยู่ 2 วิธี



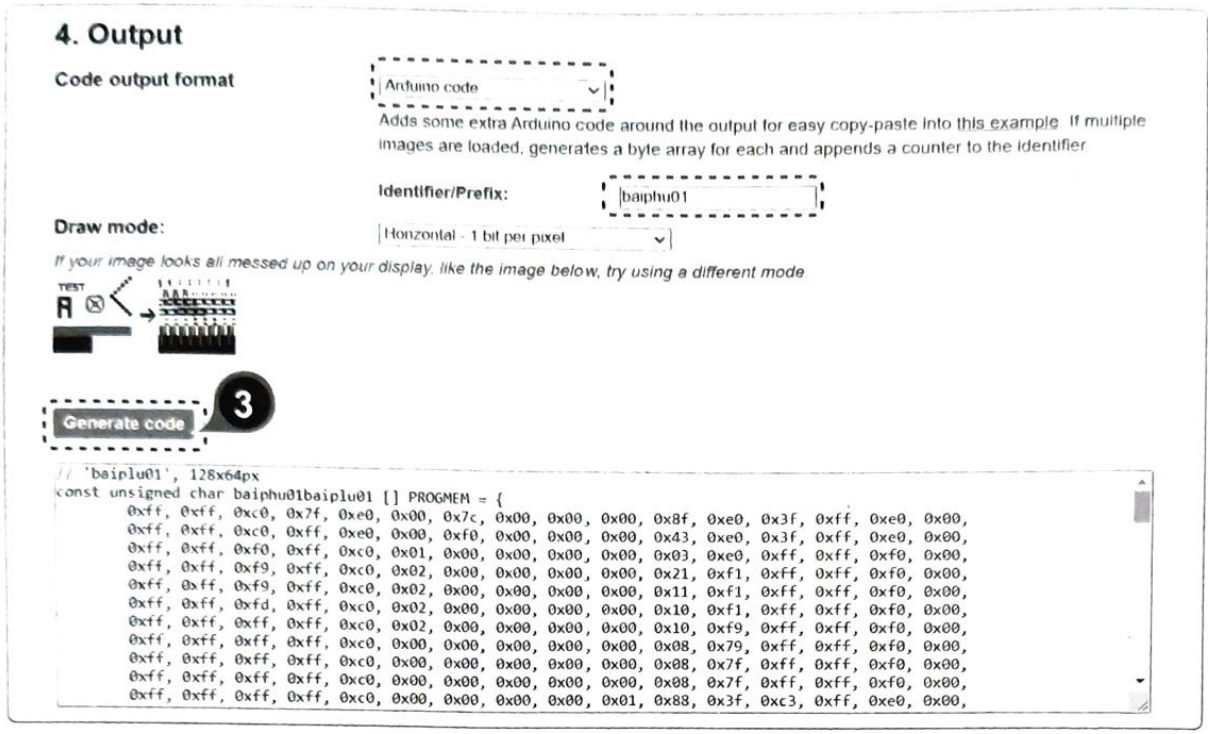
วิธีแรกจะเป็นการใช้โปรแกรม LCD Assistant ที่สามารถดาวน์โหลดจากเว็บไซต์ http://en.radzio.dxp.p//bitmap_converter มาใช้งานได้ฟรี และวิธีที่สองจะเป็นการใช้เว็บแอพพลิเคชั่นที่ชื่อว่า image2cpp ซึ่งสามารถเรียกใช้งานในแบบ Offline ได้ เพียงแค่ Save หน้าเพจเก็บไว้ในคอมพิวเตอร์ แล้วเปิดไฟล์ผ่าน Browser เพื่อเรียกใช้งาน แต่ในที่นี้เพื่อความสะ ที่นี้เพื่อความสะดวกจะเลือกใช้วิธีที่สองในแบบ Online โดยเราได้เตรียมภาพที่ เป็นไฟล์ .bmp ขนาด 128x64 pixels ที่มีการปรับสีเป็นขาว-ดำเตรียมไว้แล้ว (ในการเตรียมภาพจะใช้โปรแกรม อะไรก็ได้ เช่น Photoshop, Paint ฯลฯ แล้วแต่ถนัด) สำหรับขั้นตอนในการแปลงภาพเป็นโค้ดตัวแปรอาร์เรย์ ด้วย image2cpp มีดังนี้



1. ไปที่เว็บไซต์ http://javt.github.io/image2cpp/ คลิกปุ่ม Choose Files เพื่อเลือกภาพที่เตรียมไว้



2. กำหนดรายละเอียดให้กับภาพที่ Brightness/alpha threshold ให้เราเพิ่มหรือปรับลดตัวเลข เพื่อกำหนดความสว่างที่เหมาะสมให้กับภาพ โดยจะเห็นความเปลี่ยนแปลงของภาพในแบบเรียลไทม์ได้จาก Preview ในขั้นตอนที่ 3



3. ขั้นตอนสุดท้าย ที่ Code output format เลือกรูปแบบโค้ด ในที่นี้เลือก Arduino code และตั้งชื่อที่ จะใช้ระบุเป็นตัวชี้มายังภาพในช่อง Identifier จากนั้นคลิกปุ่ม Generate code เราจะได้โค้ดตัวแปรอาร์เรย์ที่จะนำไปใส่ในโค้ดโปรแกรม ดังรูป

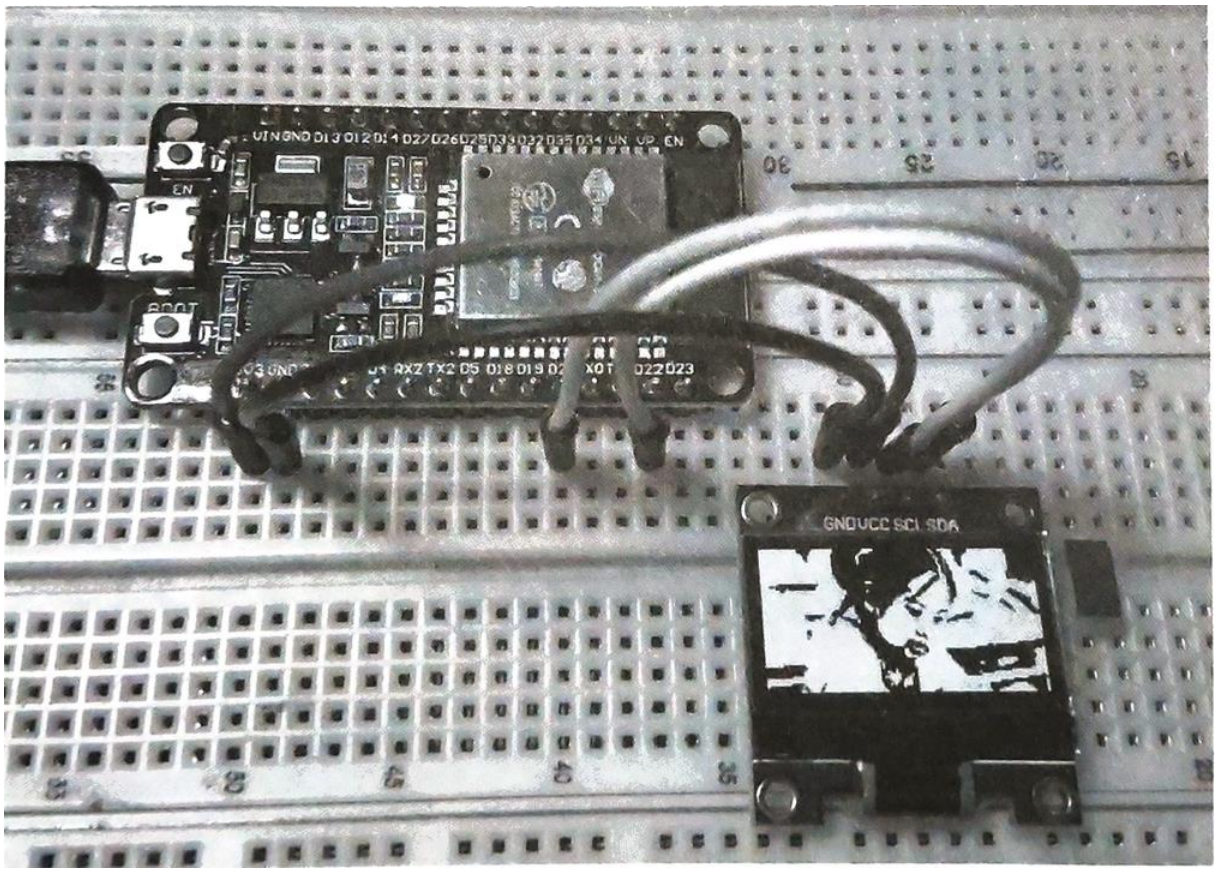
สำหรับอุปกรณ์ที่ใช้และการต่อวงจรก็คงไม่ต้องปรับเปลี่ยนอะไร เพราะเหมือนกับตัวอย่างที่แล้วมาทุกประการ ทีนี้ก็ถึงเวลาที่เราจะต้องมาลงมือเขียนโค้ดและอัพโหลดโปรแกรมกันแล้ว สำหรับรายละเอียดของโค้ดโปรแกรมมีดังนี้

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET 4

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

const unsigned char baiplu01 [] PROGMEM = {
```

(ให้ copy โค้ดตัวแปรอาร์เรย์ที่ Generate ได้ ที่อยู่ภายในกรอบทั้งหมด มาวางหรือ paste ไว้ที่นี่)

```
};
void setup() {
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  display.clearDisplay();
  display.drawBitmap(0, 0, baiplu01, 128, 64, WHITE);   //เรียกใช้ฟังก์ชั่น drawBitmap()
                                                        เพื่อแสดงผลภาพ จากข้อมูลที่เป็นตัวแปรอาร์เรย์ในชื่อ
                                                        baiplu01 โดยเริ่มจากพิกัด x = 0, y = 0 (มุมซ้ายบน)
                                                        มาสิ้นสุดที่พิกัด x = 128, y = 64 (มุมขวาล่าง) ด้วยสีขาว
  display.display();                                    //ดึงข้อมูลจากบัฟเฟอร์มาแสดงผลออกทางหน้าจอ
}

void loop() {
}
```

หลังจากอัพโหลดโค้ดโปรแกรมลงไปบนบอร์ดเรียบร้อยแล้ว ผลลัพธ์ที่ได้จะเป็นดังรูป



บันทึกผลการทดลอง

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....