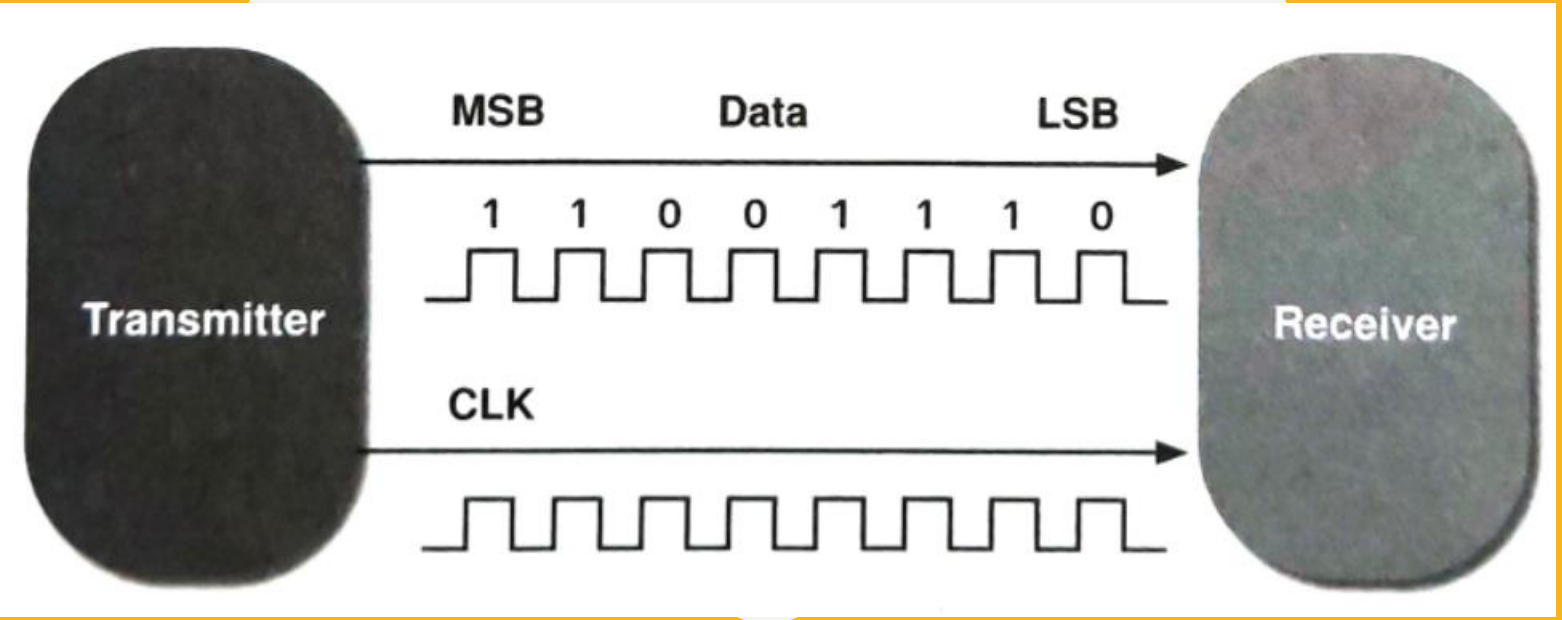
บทที่ 13
การรับส่งข้อมูลระหว่างอุปกรณ์



จุดประสงค์การเรียนรู้

1.ศึกษาการรับส่งข้อมูลระหว่างอุปกรณ์ผ่าน SPI บัส

2.ศึกษาการรับส่งข้อมูลระหว่างอุปกรณ์ผ่าน I2C บัส

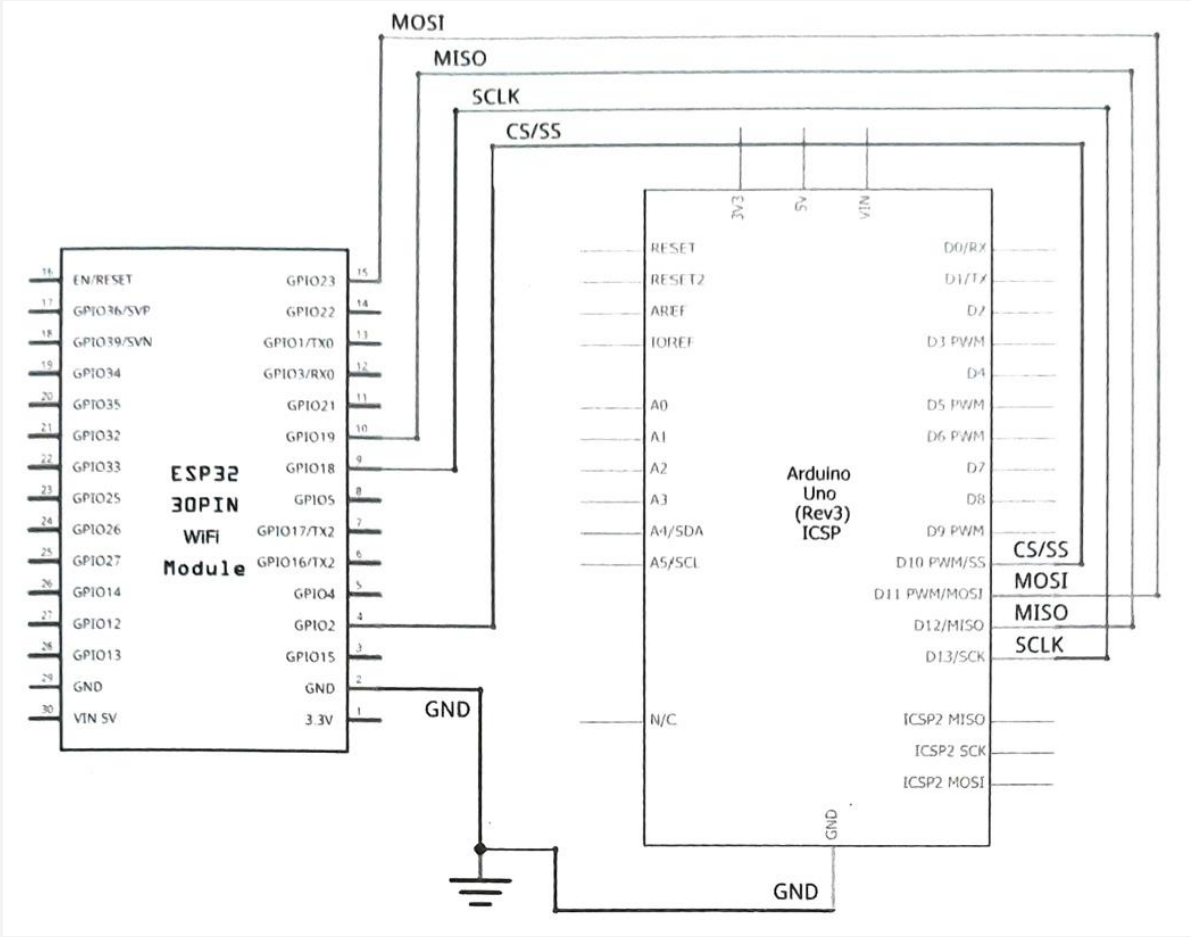
3.ศึกษาการแสดงข้อความออกทางโมดูล LCD

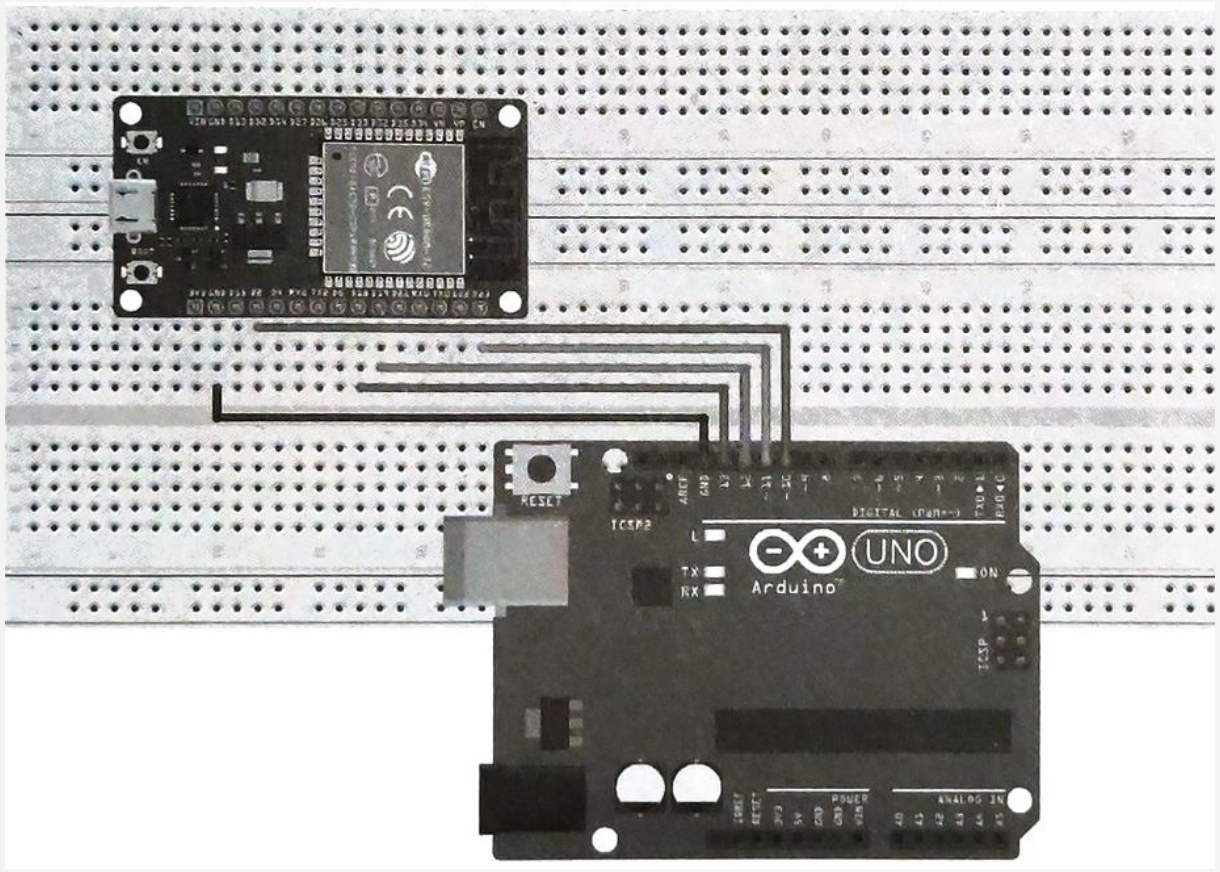
4.ศึกษาการปรับแต่งและควบคุมการแสดงผลบนหน้าจอ LCD

5.ศึกษาการแสดงข้อความออกทางโมดูลจอ OLED

6.ศึกษาการแสดงภาพกราฟิกออกทางโมดูลจอ OLED

การทดลองที่ 1 การรับส่งข้อมูลระหว่างอุปกรณ์ผ่าน SPI บัส




RESET
ICSP2
L
TX
RX
UNO
Arduino
ON
ICSP
POWER
ANALOG IN

โค้ดโปรแกรมสำหรับบอร์ด NodeMCU ที่เป็น Master

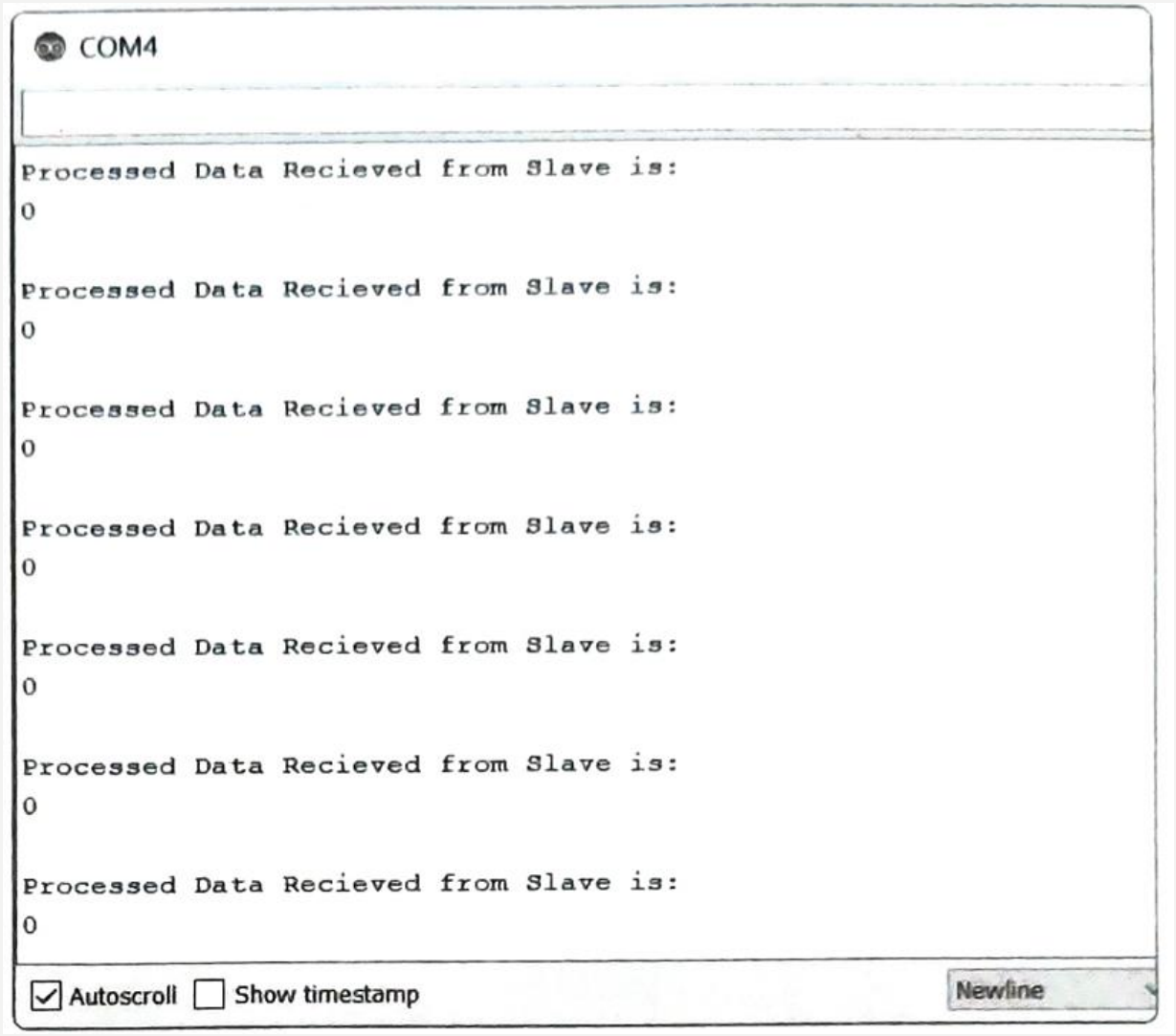
```
#include <SPI.h>                                    //เรียกใช้งานไลบรารี่ SPI สำหรับการสื่อสารผ่าน SPI
#define CS 2                                        //กำหนดให้ขา GPIO2 เป็นขา CS/SS

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
}
```

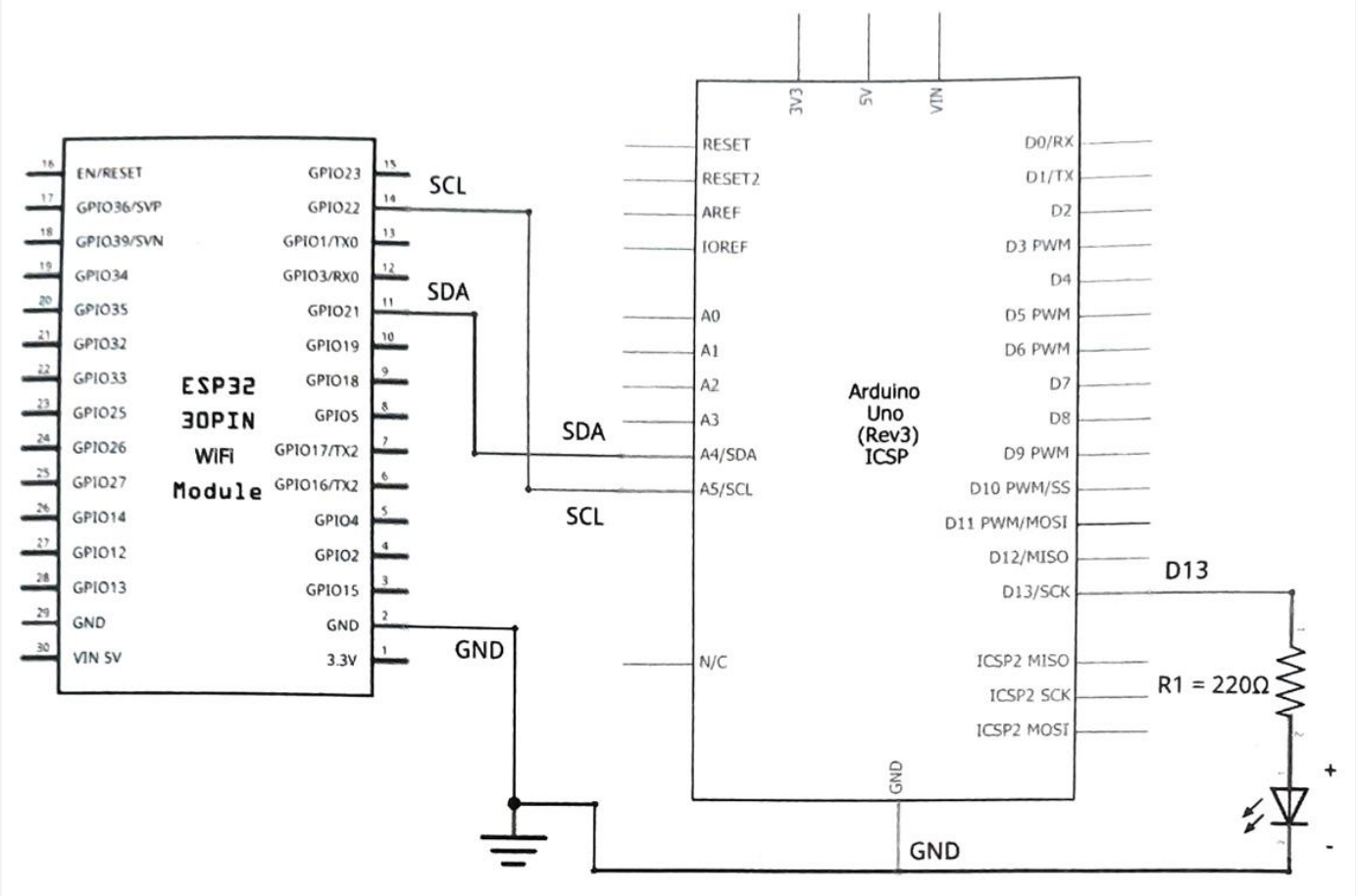
โค้ดโปรแกรมสำหรับบอร์ด Arduino Uno ที่เป็น Slave

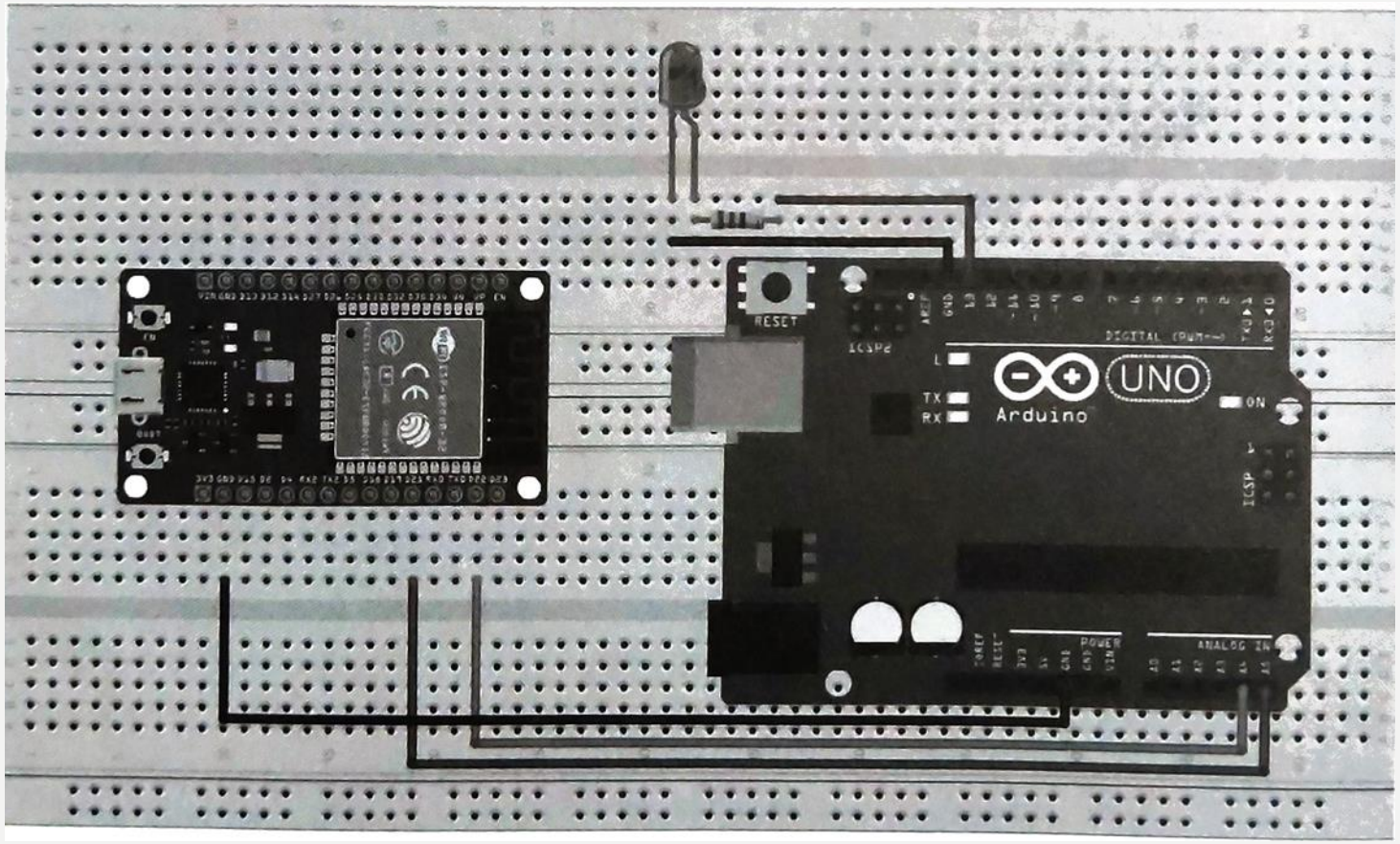
```
#include <SPI.h>                    //เรียกใช้งานไลบรารี่ SPI สำหรับการสื่อสารผ่าน SPI
volatile boolean process_it;        //ประกาศตัวแปร process_it ที่ใช้เก็บสถานะข้อมูลที่รับ
                                    มาจาก Master เพื่อรอการตรวจสอบว่าครบแล้วหรือยัง
byte a;                            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


COM4
Processed Data Recieved from Slave is:
0
Processed Data Recieved from Slave is:
0
Processed Data Recieved from Slave is:
0
Processed Data Recieved from Slave is:
0
Processed Data Recieved from Slave is:
0
Processed Data Recieved from Slave is:
0
Processed Data Recieved from Slave is:
0
Processed Data Recieved from Slave is:
0
Autoscroll
Show timestamp
Newline

การทดลองที่ 2 การรับส่งข้อมูลระหว่างอุปกรณ์ผ่าน I2C บัส




RESET
ICSP2
L
TX
RX
UNO
Arduino
ON
ICSP
DIGITAL (PWM~)
POWER
ANALOG IN

โค้ดโปรแกรมสำหรับบอร์ด NodeMCU ที่เป็น Master

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

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

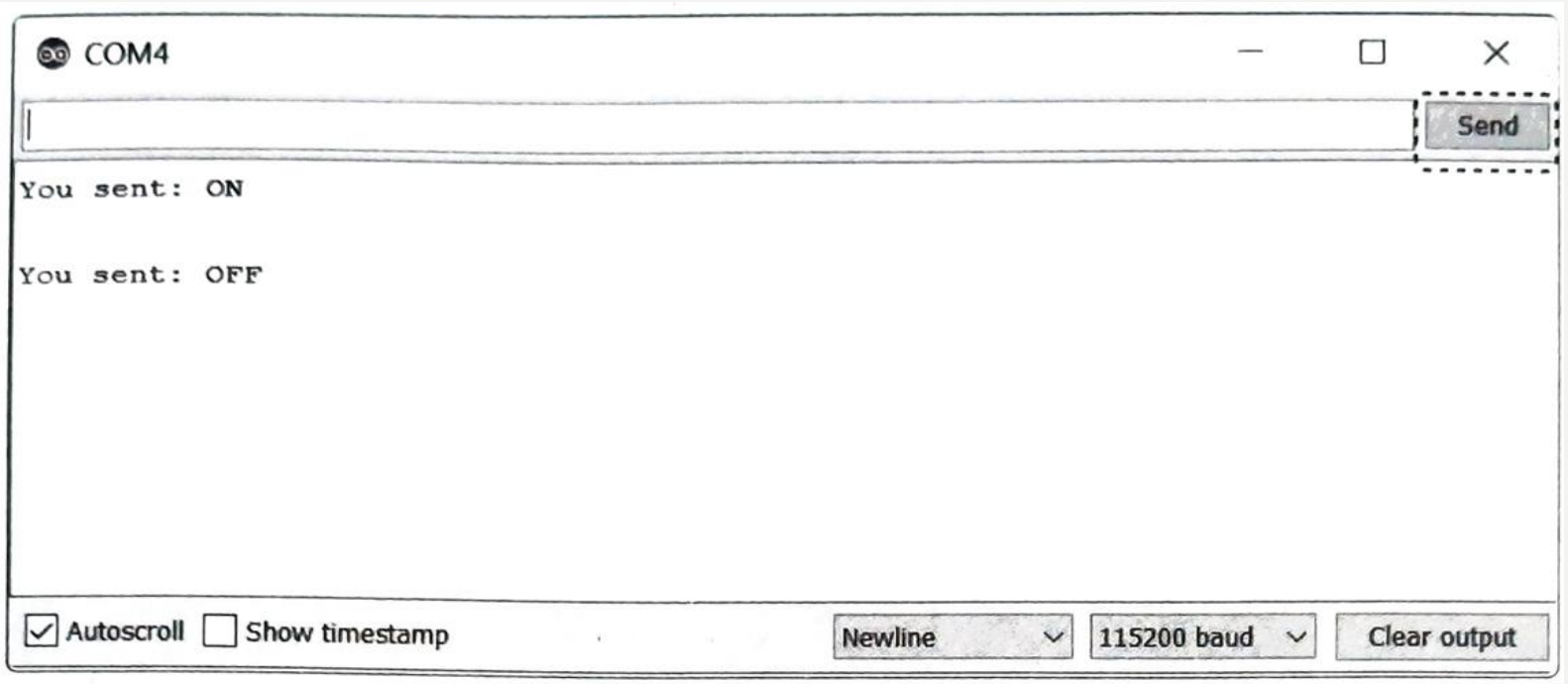
```
    Serial.println(data);            //แสดงข้อมูลที่อยู่ในตัวแปร data แล้วขึ้นบรรทัดใหม่
    Wire.endTransmission();          //ยกเลิกการส่งข้อมูลไปยังอุปกรณ์ Slave
  }
}
```

```
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
```

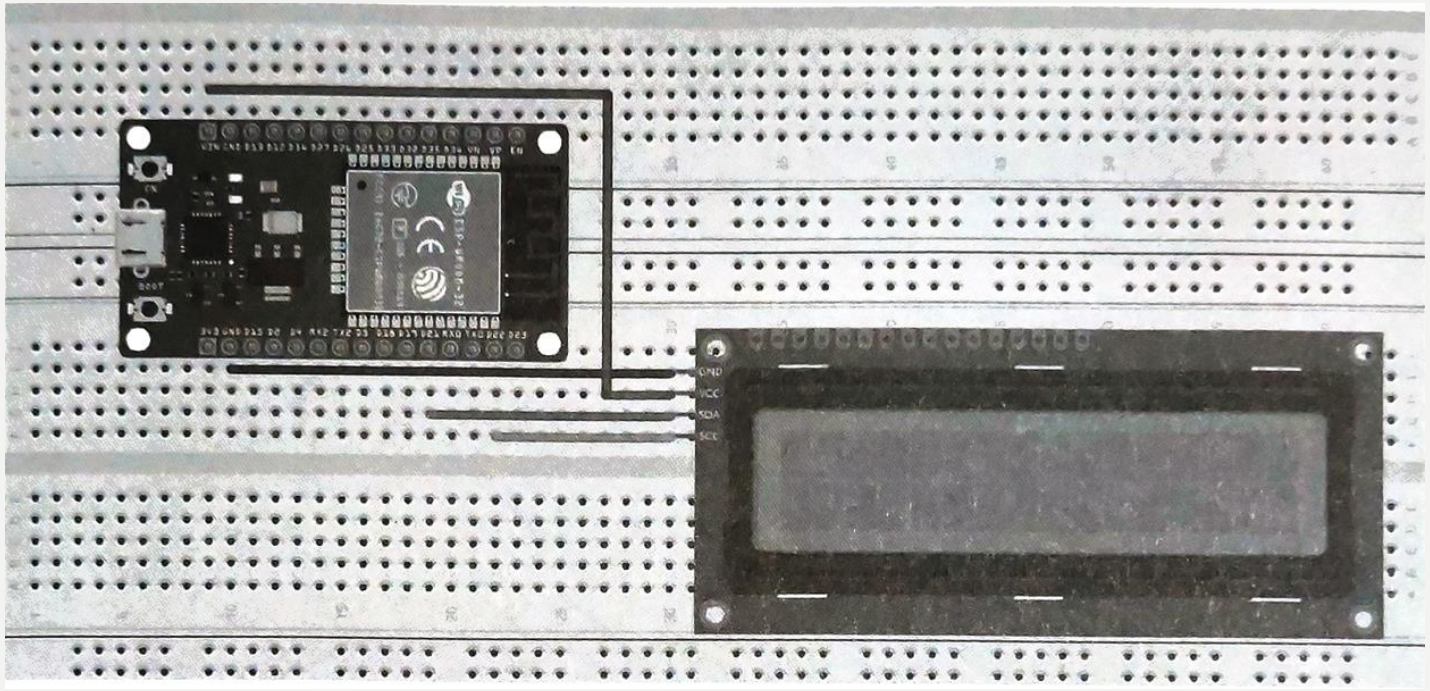
โค้ดโปรแกรมสำหรับบอร์ด Arduino Uno ที่เป็น Slave

```
#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)
}
```

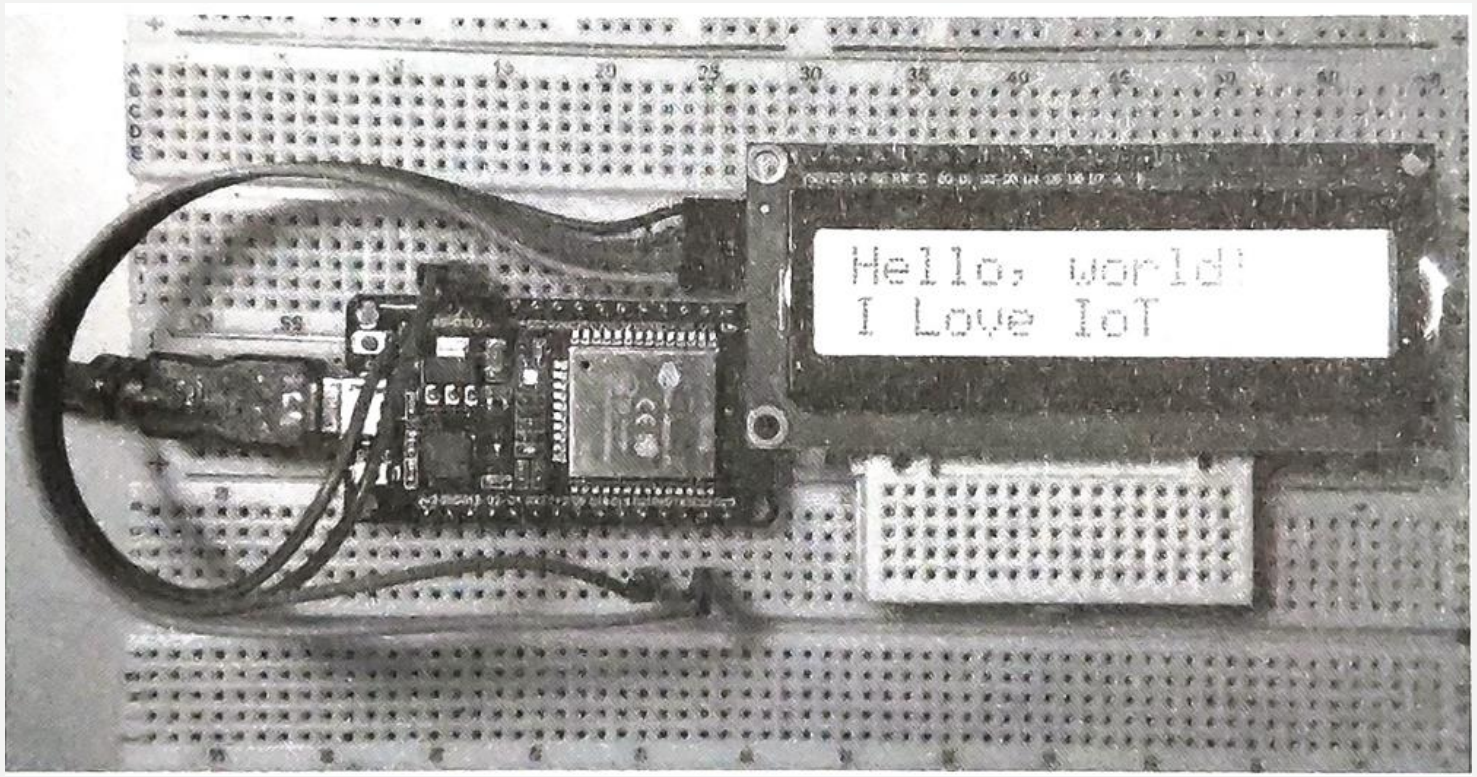
```
void loop() {
  Wire.onReceive(myHandler);      //เมื่ออุปกรณ์ Slave ได้รับข้อมูลแล้ว ให้ไปเรียกใช้ฟังก์ชั่น myHandler
  delay(300);                     //หน่วงรอเป็นเวลา 300 ms
}
void myHandler(int numByte) {     //ประกาศฟังก์ชั่น myHandler โดยผ่านค่าพารามิเตอร์ที่
                                  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
    }
  }
}
```


COM4
Send
You sent: ON
You sent: OFF
Autoscroll
Show timestamp
Newline
115200 baud
Clear output

การทดลองที่ 3 การแสดงข้อความออกทางโมดูล LCD



```
#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 แล้วนำไป
                                           เก็บไว้ในตัวแปร 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() {
}
```


Hello, world!
I Love IoT

การทดลองที่ 4 การปรับแต่งและควบคุมการแสดงผลบนหน้าจอ LCD

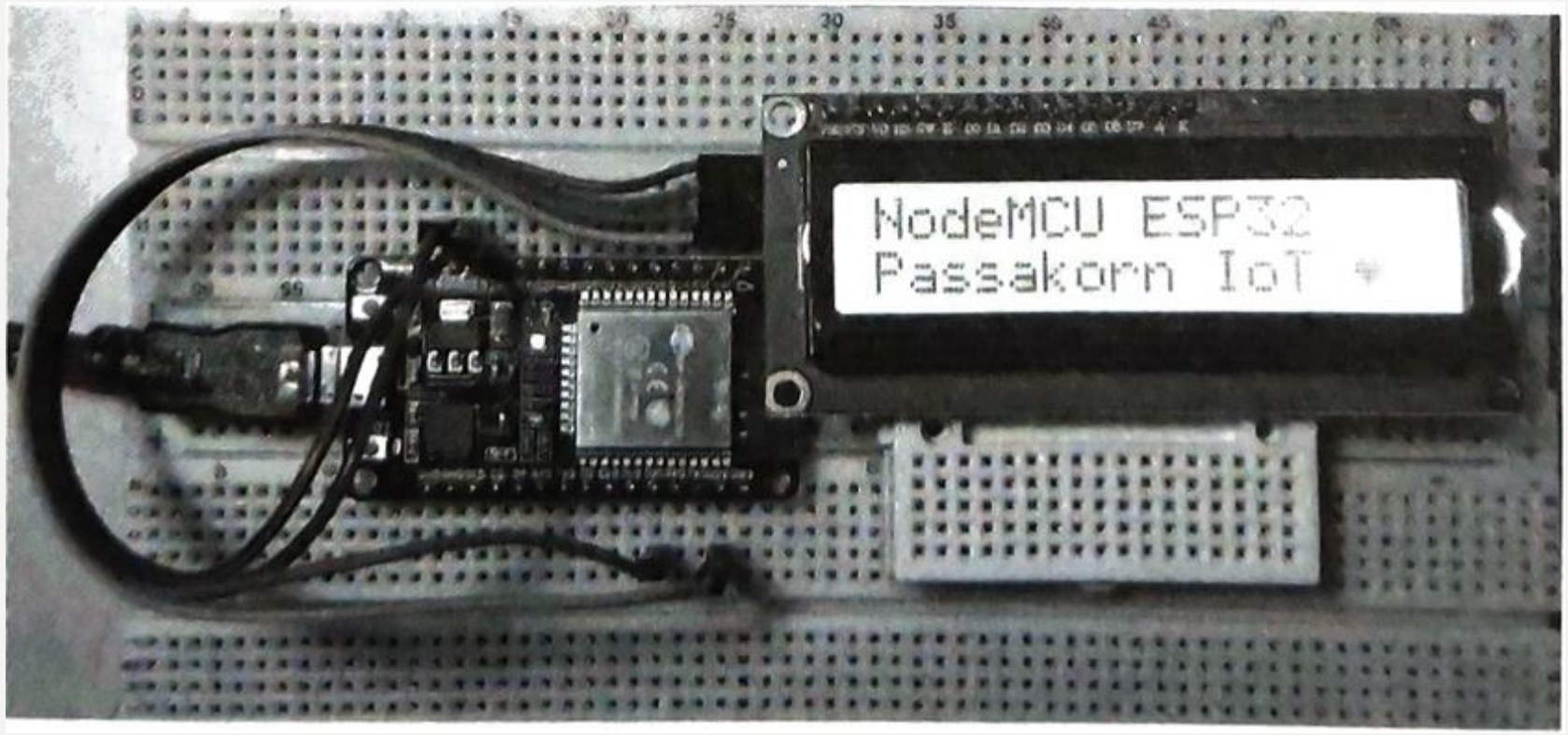
```
#include <Wire.h>                          //เรียกใช้ไลบรารี่ Wire.h สำหรับการสื่อสารแบบ I2C
#include <LiquidCrystal_I2C.h>             //เรียกใช้ไลบรารี่ LiquidCrystal_I2C.h เพื่อควบคุมการ
                                           แสดงผลบนจอ LCD

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
```

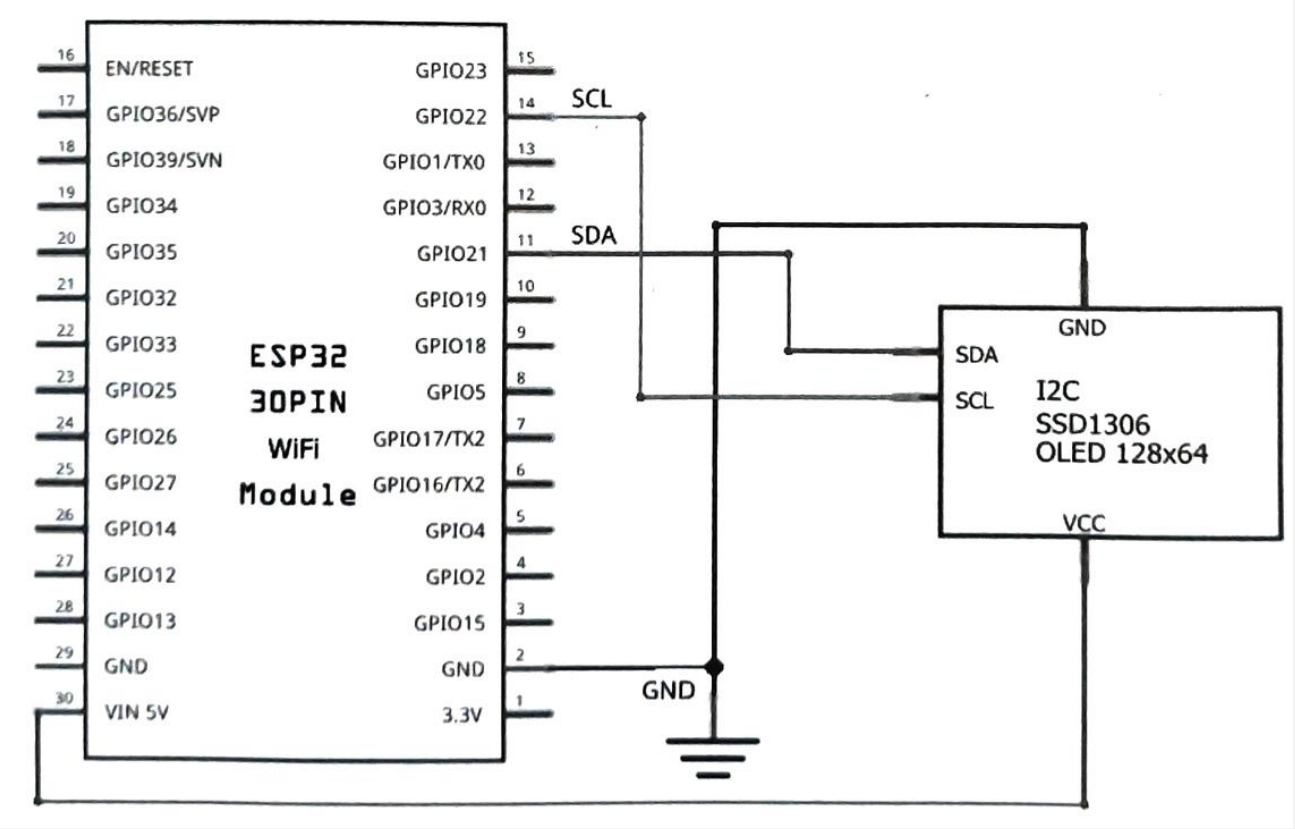
```
lcd.setCursor(10, 1);                  //เลื่อนเคอร์เซอร์ไปที่ตำแหน่งคอลัมน์ 10 แถว 1
lcd.print("IoT");                       //แสดงข้อความใน " " ออกทางหน้าจอ LCD
delay(2000);                           //หน่วงเวลารอ 2 วินาที

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
}
```

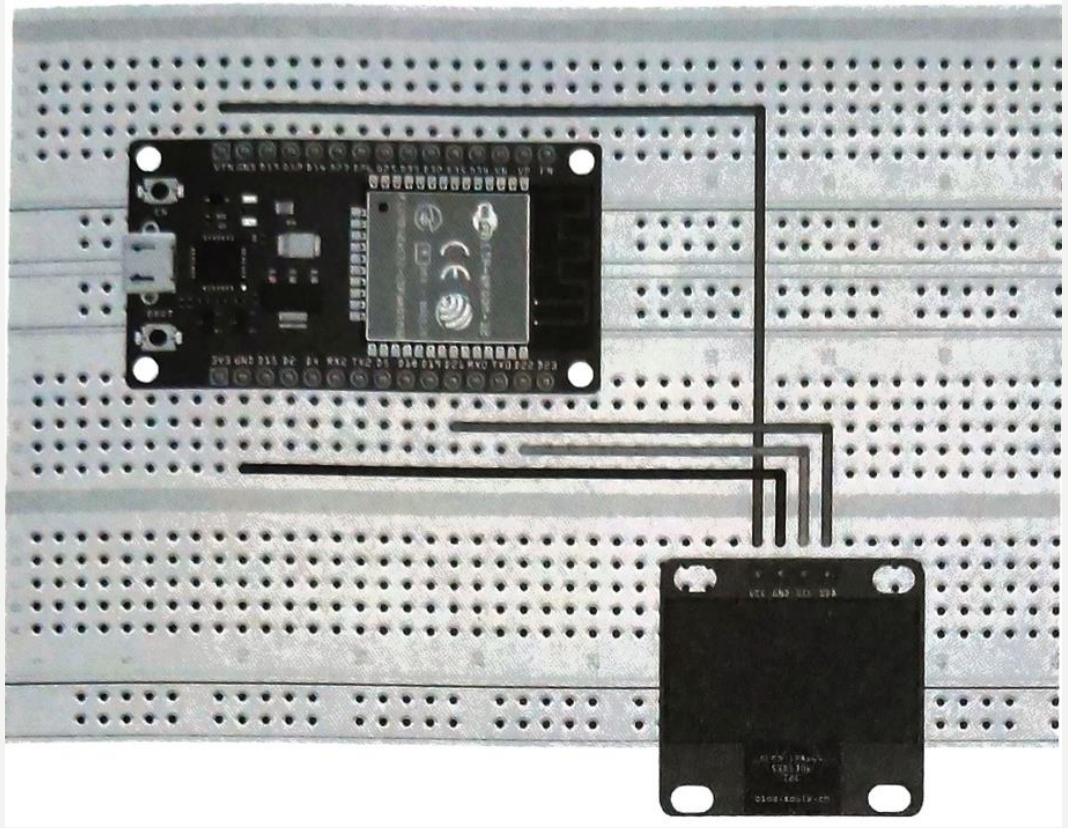
```
for (int i = 0; i <= 15; i++) {     //ใช้คำสั่งวนรอบทำซ้ำ โดยเริ่มนับจาก 0 ไปเรื่อยๆ จนถึง 15
  lcd.scrollDisplayLeft();          //เลื่อนการแสดงผลบนหน้าจอไปทางซ้าย
  delay(200);                       //หน่วงเวลารอ 0.2 วินาที
}
delay(2000);                        //หน่วงเวลารอ 2 วินาที
lcd.noDisplay();                    //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);
```

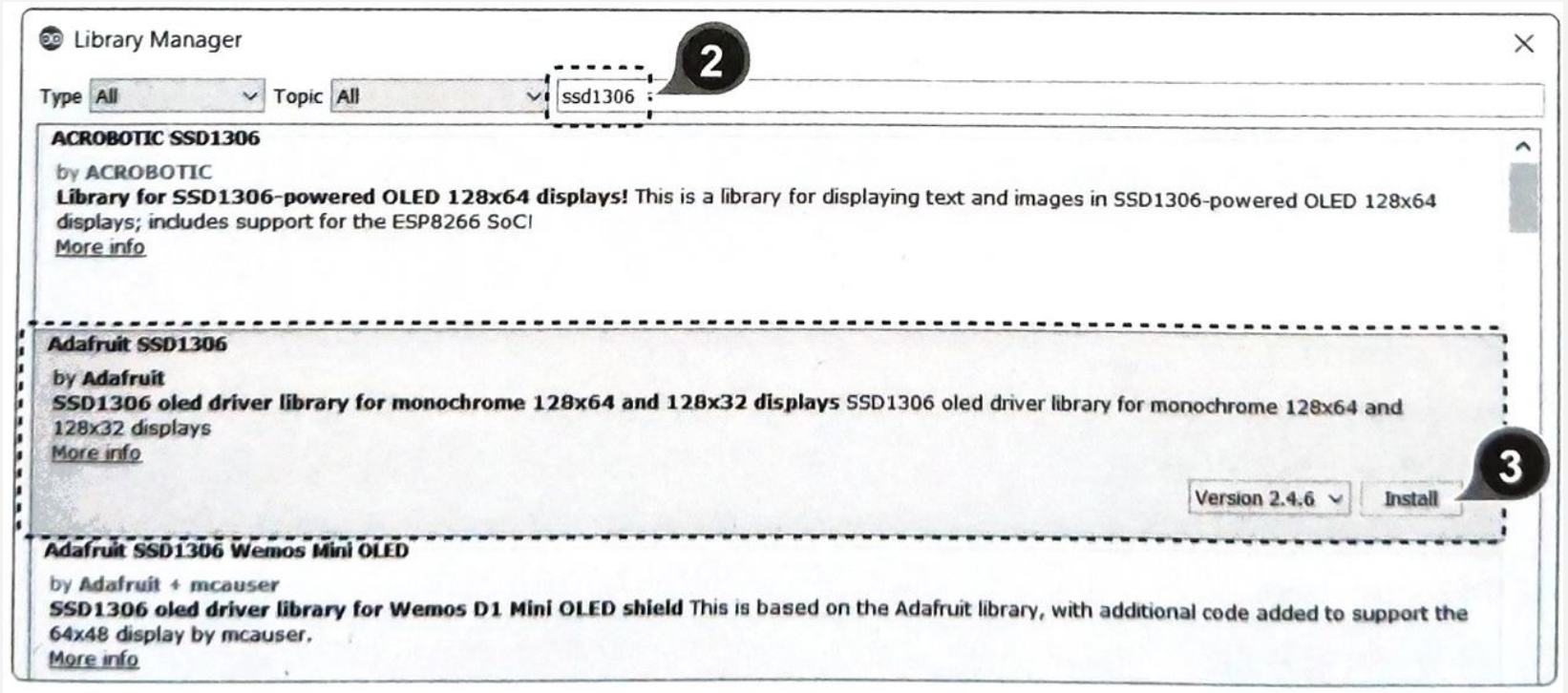
```
  lcd.noDisplay();
  delay(500);
  lcd.display();
  delay(1000);

  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
}
```

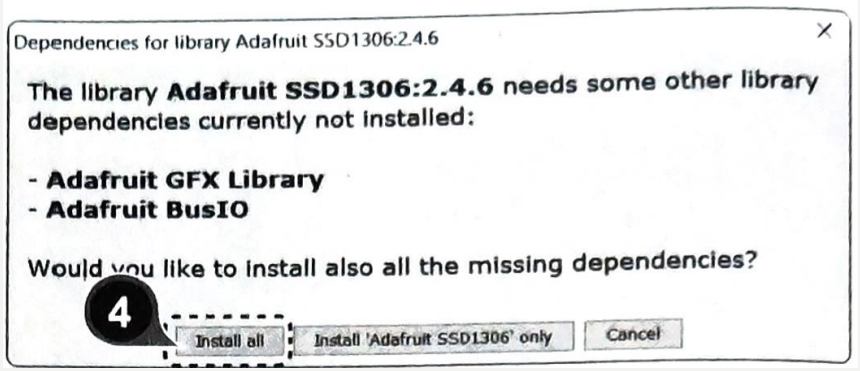

NodeMCU ESP32
Passakorn IoT

การทดลองที่ 5 การแสดงข้อความออกทางโมดูลจอ OLED



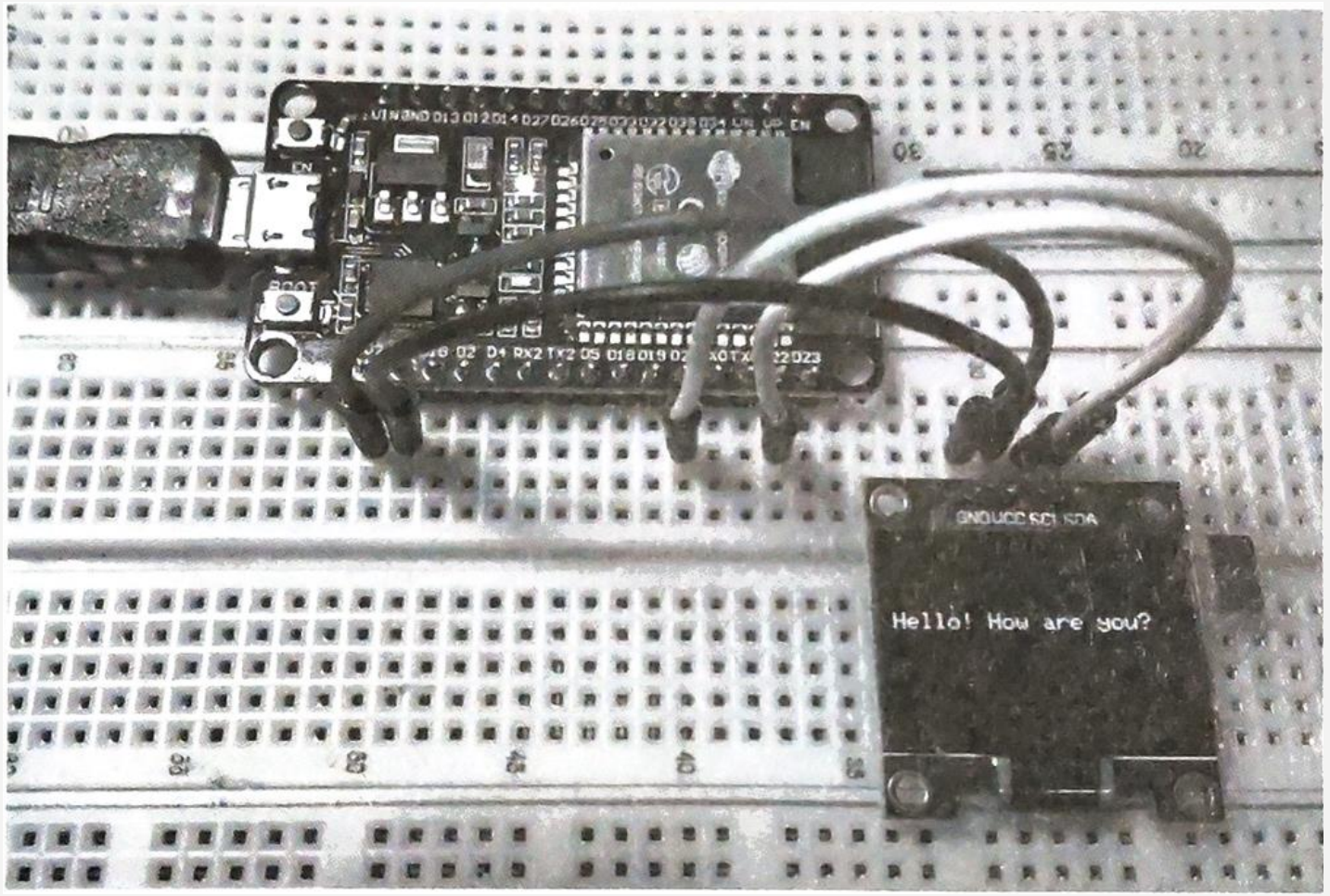



Library Manager
Type All
Topic All
ssd1306
2
ACROBOTIC SSD1306
by ACROBOTIC
Library for SSD1306-powered OLED 128x64 displays! This is a library for displaying text and images in SSD1306-powered OLED 128x64 displays; includes support for the ESP8266 SoC!
More info
Adafruit SSD1306
by Adafruit
SSD1306 oled driver library for monochrome 128x64 and 128x32 displays SSD1306 oled driver library for monochrome 128x64 and 128x32 displays
More info
Version 2.4.6
Install
3
Adafruit SSD1306 Wemos Mini OLED
by Adafruit + mcauser
SSD1306 oled driver library for Wemos D1 Mini OLED shield This is based on the Adafruit library, with additional code added to support the 64x48 display by mcauser.
More info


Dependencies for library Adafruit SSD1306:2.4.6
The library Adafruit SSD1306:2.4.6 needs some other library dependencies currently not installed:
- Adafruit GFX Library
- Adafruit BusIO
Would you like to install also all the missing dependencies?
4
Install all
Install 'Adafruit SSD1306' only
Cancel

```
#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() {
}
```


Hello! How are you?