# การใช้งาน Arduino เบื้องต้น

ผู้ช่วยศาสตราจารย์ศิริพงษ์  ฉายสินธ์

# คำนำ

บอร์ดไมโครคอนโทรเลอร์ได้มีการพัฒนาการใช้งานให้มีความสะดวกในการโปรแกรมการทำงานของตัวไมโครคอนโทรเลอร์ ทั้งในด้านการเขียนโปรแกรมและการถ่ายโอนข้อมูลไปยังไมโครคอนโทรเลอร์ แตกต่างจากเดิมที่พัฒนาโปรแกรมการทำงานด้วยภาษาแอสแซมบลี้ และต้องมีเครื่องมือเสริมในการการถ่ายโอนข้อมูล โดยบอร์ดไมโครคอนโทรลเลอร์ตระกูลหนึ่งที่ถูกพัฒนาให้ง่ายต่อการใช้งานนั่นคือบอร์ด Arduino ซึ่งเป็นที่นิยมใช้กันอย่างมาก และมีราคาไม่สูง ทำให้มีผู้สนใจที่จะนำมาเรียนรู้ในการเขียนโปรแกรมควบคุมการทำงาน และเชื่อมต่อกับอุปกรณ์ภายนอกเพื่อทำเป็นระบบควบคุม หรือวัดค่าและแสดงผล โดยในหนังสือเล่มนี้ได้เรียบเรียงเนื้อหาการใช้งาน ตั้งแต่การใช้งานเบื้องต้น จนถึงการต่อใช้งานในรูปแบบต่างๆ ที่จำเป็นเบื่องต้น เพื่อนำไปสู่การประยุกต์ใช้งานต่อไป ทั้งนี้หากหนังสือเล่มนี้มีข้อผิดพลาดส่วนใดผู้เขียนขอน้อมรับไปเพื่อปรับปรุงให้สมบรูณ์ยิ่งขึ้นต่อไป

ผู้ช่วยศาสตราจารย์ศิริพงษ์  ฉายสินธ์

มกคราคม  2562

# สารบัญ

หน้า

หน้า

หน้า

หน้า

# บทที่ 1

# Arduino

## 1.1 บอร์ด Arduino

Arduino (อาดุยโน่) ได้เกิดขึ้นมาในปี ค.ศ. 2005 เป็นไมโครคอนโทรลเลอร์ที่ถูกออกแบบเพื่อให้นักศึกษาที่ Interaction Design Institute Ivrea เมือง Ivrea ประเทศ Italy ได้ใช้ทดลองในการเรียนเพื่อลดค่าใช้จ่ายของนักศึกษาเนื่องจากบอร์ดไมโครคอนโทรลเลอร์ในสมัยนั้นมีราคาแพง และยังสามารถพัฒนาการทำงานได้ด้วยโปรแกรมที่แจกฟรี (Open Sorce) และรูปแบบการเขียนโปรแกรมควบคุมการทำงานก็ใช้โครงสร้างของภาษาซี ซึ่งเป็นสากลมีการใช้งานอย่างกว้างขวางมานาน ทำให้สามารถนำบอร์ด Arduino ไปประยุกต์ใช้งานที่เกี่ยวข้องกับอุปกรณ์อิเล็กทรอนิกส์ได้อย่างกว้างขวาง บอร์ด Arduino ได้ถูกพัฒนาอย่างต่อเนื่องมีให้เลือกใช้งานตามความต้องการของผู้พัฒนา เช่น บอร์ด Arduino UNO บอร์ด Arduino Nano บอร์ด Arduino Mega เป็นต้น ซึ่งบอร์ดแต่ละแบบก็จะมีสิ่งสนับสนุนที่อยู่บนบอร์ดแตกต่างกันออกไป นอกจากนี้ยังมีผู้พัฒนาบอร์ดสำหรับเชื่อมต่อกับบอร์ด Arduino ด้วย โดยจะเรียกว่า Shields เพื่อช่วยให้การนำบอร์ด Arduino ไปใช้งานสามารถทำงานได้สะดวกขึ้น

## 1.2 Arduino IDE

โปรแกรมสำหรับการควบคุมสั่งงานบอร์ด Arduino มีชื่อเรียกว่า Arduino IDE สามารถดาวน์โหลดได้ฟรีที่ https://www.arduino.cc/en/Main/Software ซึ่งมีให้เลือกใช้ได้หลายระบบปฏิบัติการ เมื่อดาวน์โหลดมาแล้วให้ดำเนินติดตั้งลงบนเครื่องคอมพิวเตอร์

Arduino Diecimila in Stoicheia

Arduino Duemilanove (rev 2009b)

Arduino UNO

Arduino Leonardo

Arduino Mega

Arduino MEGA 2560 R3 (front side)[a]

Arduino MEGA 2560 R3 (back side)[a]

Arduino Nano

Arduino Due (ARM Cortex-M3 core)

LilyPad Arduino (rev 2007)

https://en.wikipedia.org/wiki/Arduino#Official_boards : 2561

รูปที่ 1.1 ตัวอย่างบอร์ด Arduino แบบต่าง ๆ

Multiple shields can be stacked. In this example the top shield contains a solderless breadboard.

Screw-terminal breakout shield in a wing-type format

Adafruit Motor Shield with screw terminals for connection to motors

Adafruit Datalogging Shield with a Secure Digital (SD) card slot and real-time clock (RTC) chip

HackARobot Fabric Shield – designed for Arduino Nano to hook up motors and sensors such as gyroscope or GPS, and other breakout boards such as WiFi, Bluetooth, RF, etc.

https://en.wikipedia.org/wiki/Arduino#Official_boards : 2561

รูปที่ 1.2 ตัวอย่างบอร์ด Shields แบบต่าง ๆ

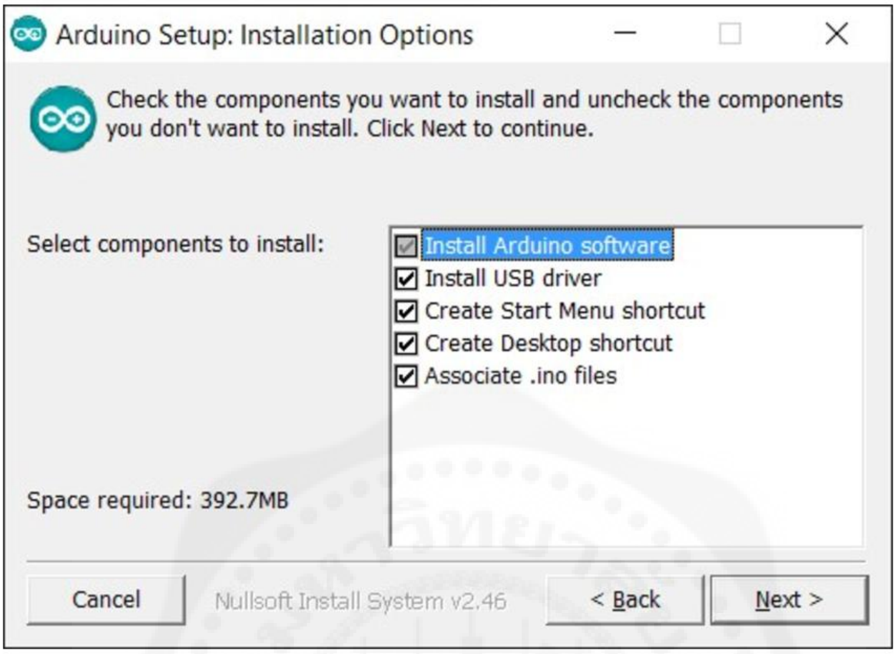

https://www.arduino.cc/en/Main/Software : 2561

รูปที่ 1.3 เวปไซต์ดาวน์โหลด Arduino IDE

ดับเบิ้ลคลิ๊กไฟล์ติดตั้งที่ดาวน์โหลดมาจะขึ้นหน้าต่างดังรูปที่ 1.4

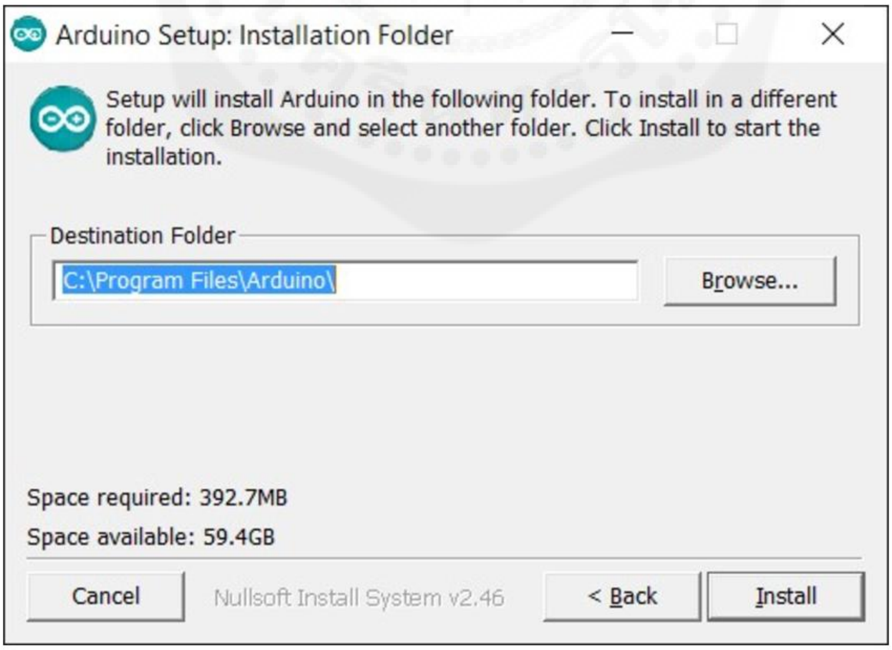

รูปที่ 1.4 ข้อตกลงการใช้โปรแกรม

เมื่อตอบเห็นด้วยจะปรากฏหน้าต่างดังรูปที่ 1.5



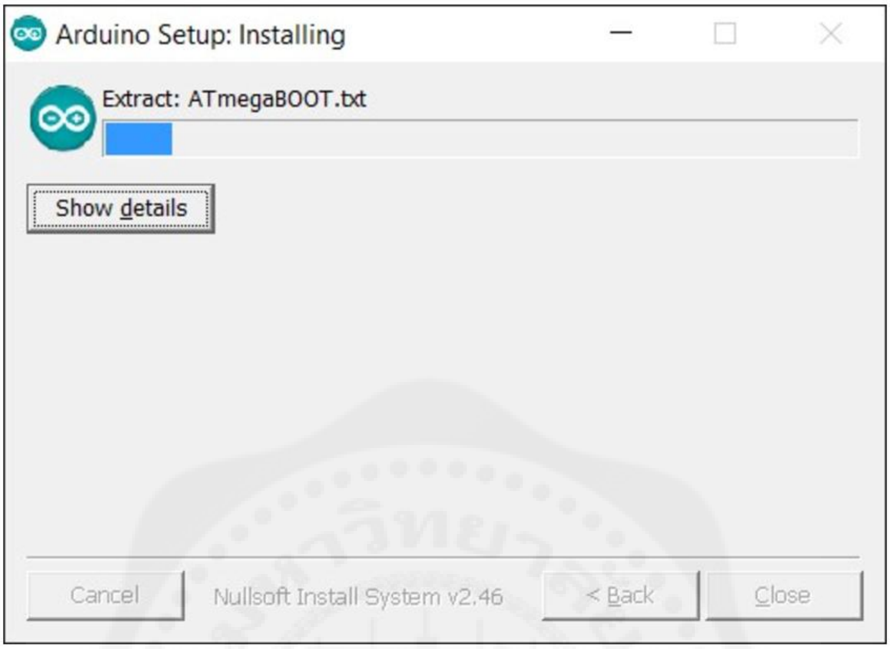

รูปที่ 1.5 เลือกออฟชั่นในการติดตั้งโปรแกรม

เมื่อเลือกออฟชั่นในการติดตั้งโปรแกรมเสร็จกดปุ่ม Next จะปรากฏหน้าต่างดังรูปที่ 1.6



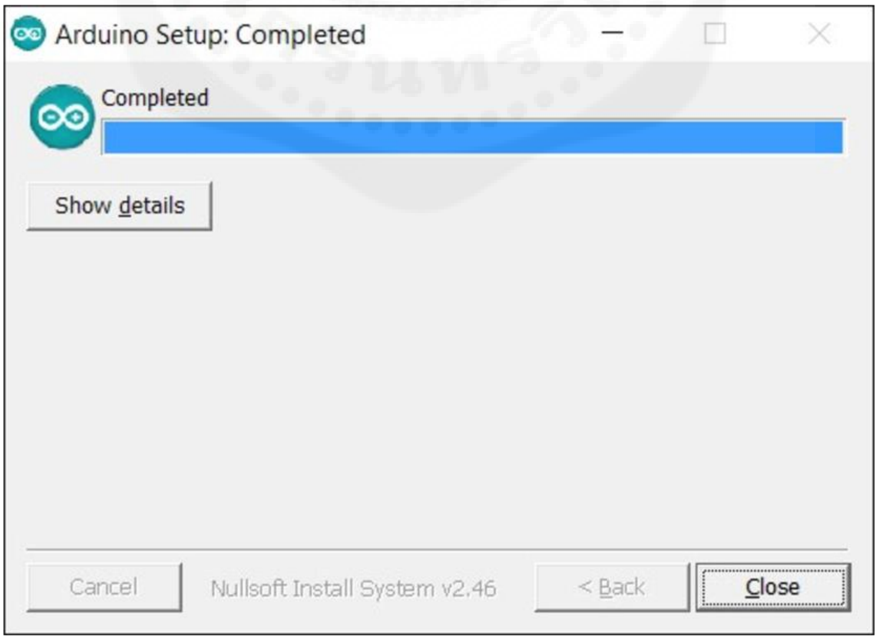

รูปที่ 1.6 เลือกโฟลเดอร์ที่จะติดตั้งโปรแกรม

เมื่อเลือกโฟลเดอร์ที่จะติดตั้งโปรแกรมเสร็จ กดปุ่ม Install จะปรากฏหน้าต่างดังรูปที่ 1.7




รูปที่ 1.7 โปรแกรมจะดำเนินการติดตั้ง

รอจนโปรแกรมติดตั้งเสร็จสิ้น กดปุ่ม Close




รูปที่ 1.7 โปรแกรมติดตั้งเรียบร้อย

เมื่อดำเนินการติดตั้งสำเร็จจะมีไอคอน Arduino ที่หน้า desktop ดับเบิ้ลคลิ๊กที่ไอคอนโปรแกรม จะเปิดขึ้นมาดังรูปที่ 1.8


รูปที่ 1.8 โปรแกรม Arduino IDE

รูปร่างหน้าตาของโปรแกรม Arduino มีรายละเอียดดังนี้

1. ทูลบาร์ แถบคำสั่งต่าง ๆ

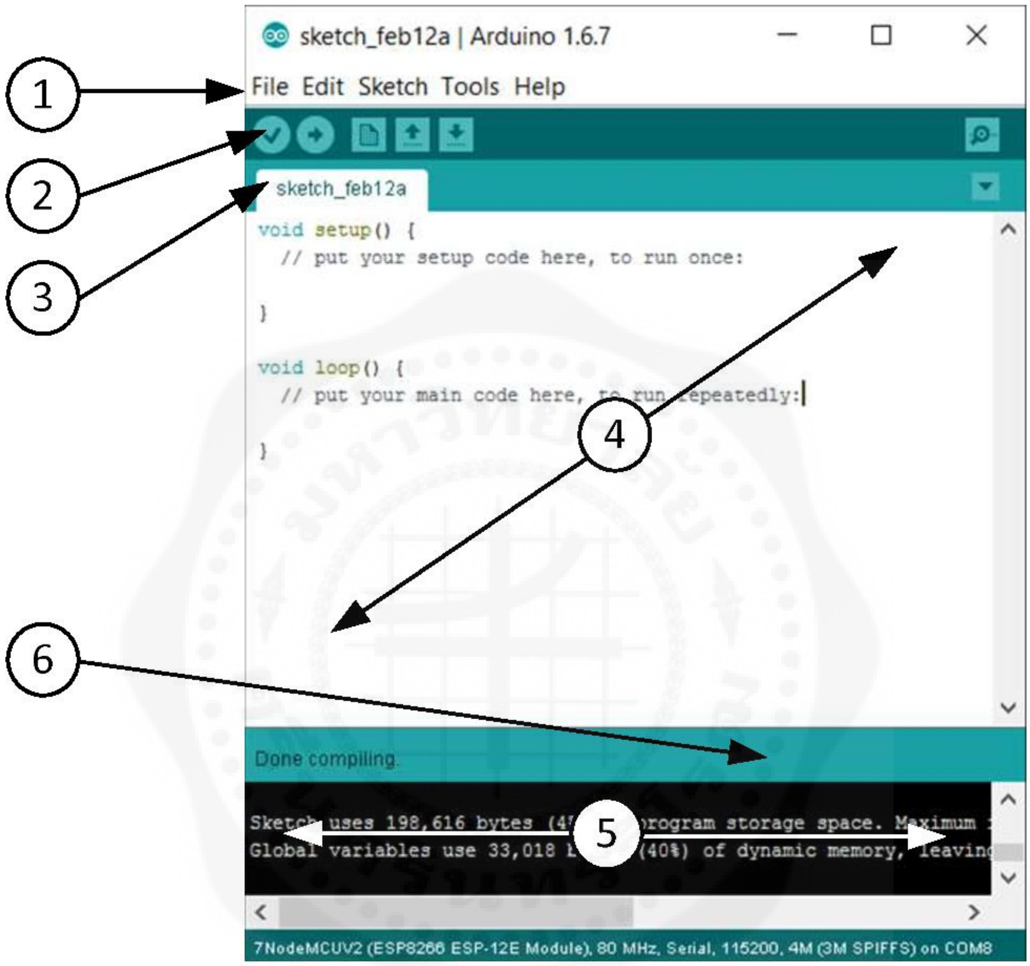
2. ปุ่ม Short cut คำสั่งที่ใช้บ่อย

3. แทป แสดงชื่อไฟล์ของโปรแกรมควบคุมการทำงาน ในกรณีเปิดหลายโปรแกรม

4. พื้นที่เขียนคำสั่งควบคุมการทำงาน

5. พื้นที่แสดงข้อความ การประมวลผลคำสั่งการทำงาน

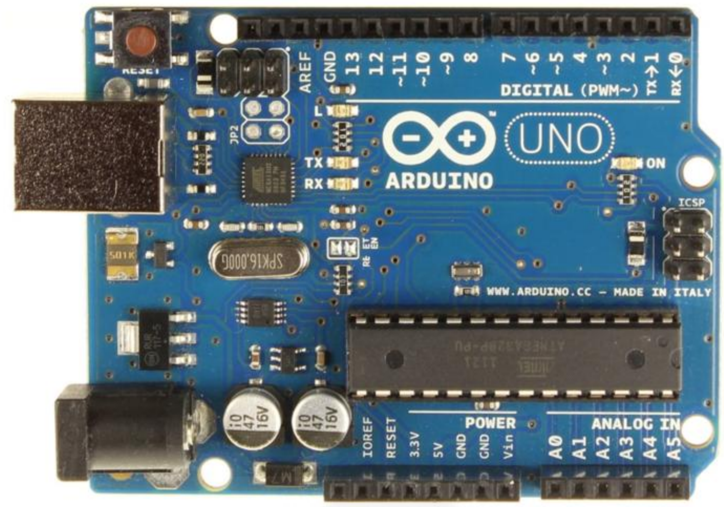
6. คอนโซลแสดงการทำงานของโปรแกรม Arduino IDE




รูปที่ 1.9 รายละเอียดหน้าตาโปรแกรม Arduino IDE

## 1.3 บอร์ด Arduino UNO

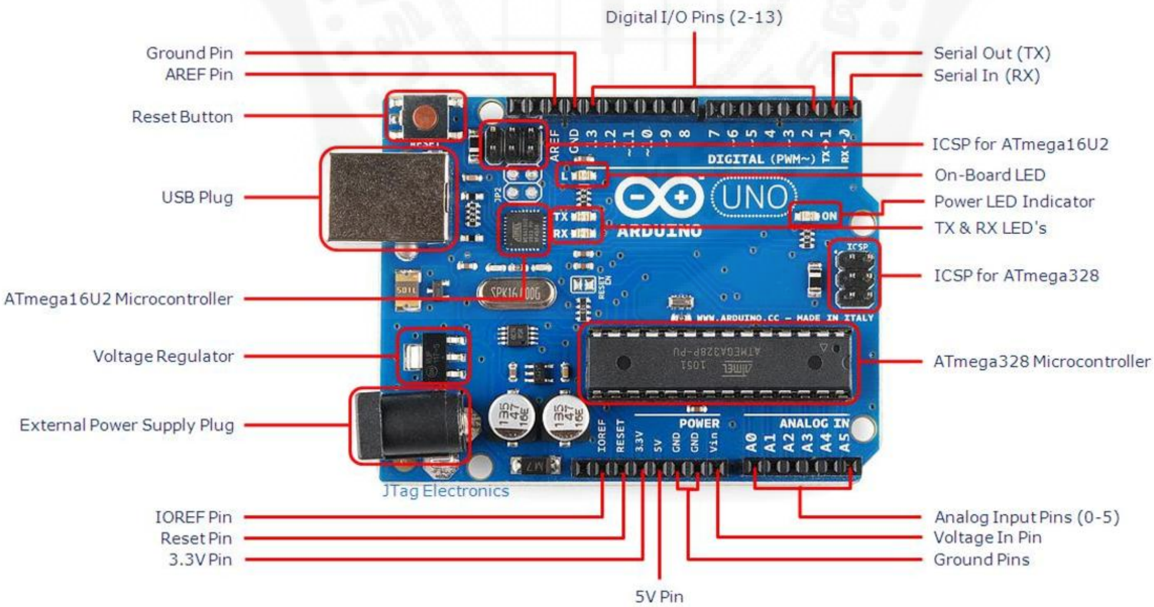
ในหนังสือเล่มนี้จะใช้บอร์ด Arduino UNO เป็นบอร์ดหลักในการใช้งานซึ่งมีลักษณะดังรูปที่ 1.10



https://www.arduino.cc/en/Guide/Windows : 2561

รูปที่ 1.10 บอร์ด Arduino UNO

และตำแหน่งใช้งานของขาต่างๆบนบอร์ด ดังรูป




http://www.jtagelectronics.com/?p=75 : 2561

รูปที่ 1.11 ตำแหน่งขาต่าง ๆ ของบอร์ด Arduino UNO

โดยแต่ละอุปกรณ์บนบอร์ดมีหน้าที่ดังนี้

**1. USB Plug :** ใช้สำหรับเชื่อมต่อกับ Computer ทางพอร์ต USB เพื่ออัปโหลดโปรแกรมเข้าบอร์ด Arduino และจ่ายไฟให้กับบอร์ด Arduino ด้วย

**2. Reset Button :** เป็นปุ่มสำหรับรีเซท โดยการกดปุ่มเมื่อต้องการให้บอร์ด Arduino เริ่มทำงานใหม่

**3. ICSP Port สำหรับ Atmega16U2 :** เป็นพอร์ตที่ใช้เพื่อโปรแกรมอัพเดทการทำงานของ Atmega16U2

**4. Digital I/O** : ขา D0 ถึง D13 นอกจากนี้ บางขาก็จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น ขา D0 และ D1 เป็นขา Tx และ Rx ของการส่งข้อมูลแบบอนุกรม (Serial) , ขา D3, D5, D6, D9, D10 และ D11 เป็นขาสำหรับส่งข้อมูลแบบ PWM (Pluse Width Modulator)

**5. ICSP Port สำหรับ Atmega328 :** เป็นพอร์ตที่ใช้เพื่อโปรแกรมอัพเดทการทำงานของ Atmega328

**6. Atmega328** : เป็น Microcontroller ที่ใช้บนบอร์ด Arduino UNO

**7. Analog I/O :** ตั้งแต่ขา A0 ถึง A5

**9. External Power Supply Plug :** รับไฟกระแสตรงจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V

**10. Atmega16U2 :** เป็น Microcontroller ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega16U2

**11. Voltage In Pin :** เป็นขาที่ใช้ป้อนแรงดันจากภายนอกให้กับบอร์ด Arduino โดยต้องมีแรงดันเท่ากับ 5V หากป้อนแรงดันผ่านทาง **External Power Supply Plug** ที่ขา Voltage In Pin ก็จะมีแรงดันเท่ากับ 5V จ่ายออกมา

**12. 5V Pin :** เป็นขาที่จ่ายแรงดันกระแสตรง 5V ออกมาเพื่อจ่ายให้กลับอุปกรณ์ภายนอก

**13. 3.3V Pin :** เป็นขาที่จ่ายแรงดันกระแสตรง 3.3V ออกมาเพื่อจ่ายให้กลับอุปกรณ์ภายนอก

**14. Ground Pins :** เป็นขา ground เพื่อใช้ต่อร่วมกับวงจรอื่นเพื่อให้ครบวงจร ตามหลักการวงจรไฟฟ้า

**15. IOREF Pin :** เป็นขาที่ใช้อ้างอิงสำหรับสัญญาณแรงดันที่ใช้เชื่อมต่อกับอุปกรณ์ภายนอก 5V หรือ 3.3V

**16. AREF Pin :** เป็นขาที่ใช้อ้างอิงสำหรับสัญญาณแรงดันอินพุทอนาล็อค

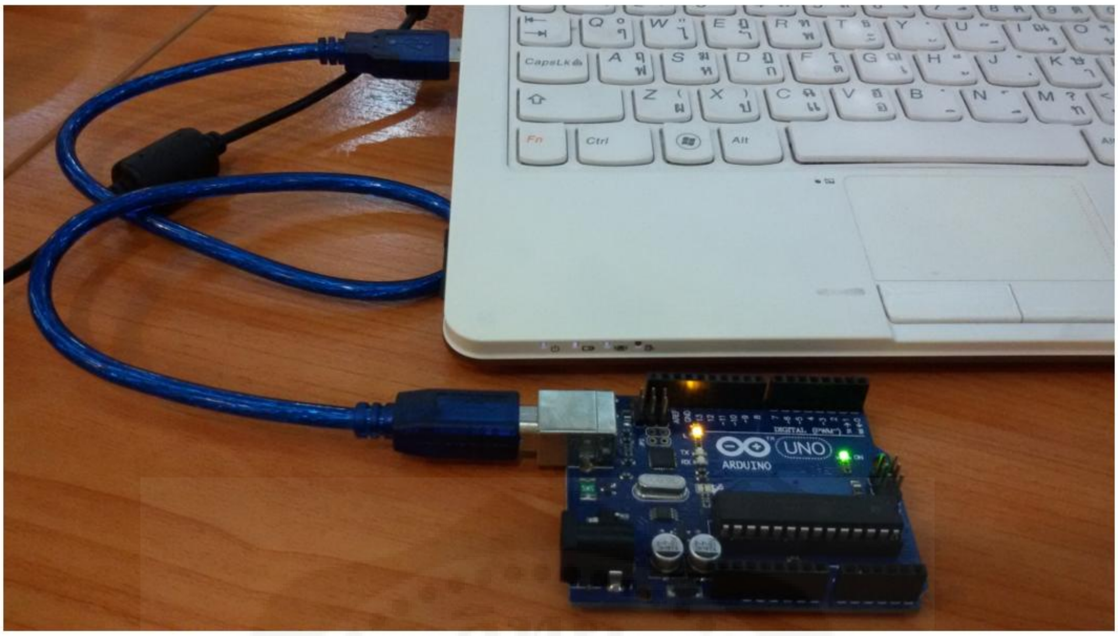
**17. Power LED Indicator :** LED แสดงว่าบอร์ดมีแรงดันจ่ายให้ทำงาน

**18. On-Board LED : LED** ที่ต่ออยู่กับขา D13 เมื่อขา D13 มีค่าลอจิก 1 LED จะสว่าง และจะดับลงเมื่อขา D13 มีค่าลอจิก 0

**19. TX & RX LEDs :** LED แสดงสถานะเมื่อบอร์ด Arduino มีการรับส่งข้อมูล เช่น เวลาอัพโหลดโปรแกรมจาก Aduino IDE ลงบอร์ด Arduino

## 1.4 เริ่มใช้งาน Arduino

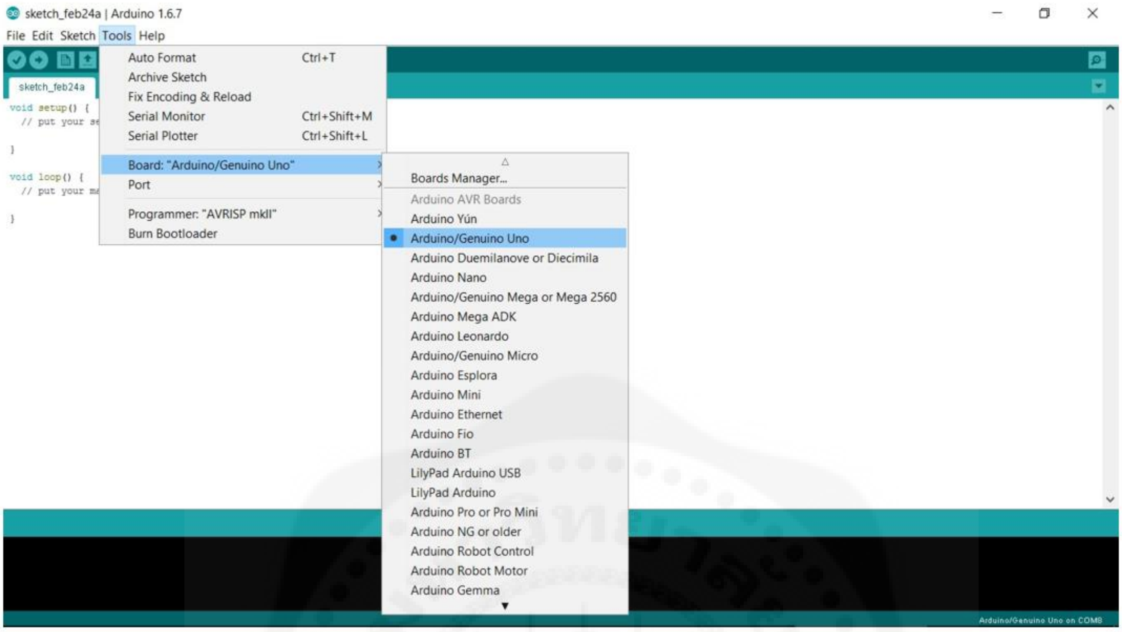
การใช้งาน Arduino เริ่มจากต่อบอร์ด Arduino เข้ากับคอมพิวเตอร์ผ่านทางพอร์ต USB ที่บอร์ด Arduino  LED ON (สีเขียว) จะสว่างแสดงว่าบอร์ด Arduino มีไฟมาเลี้ยงจากคอมพิวเตอร์ หลังจากนั้นเปิดโปรแกรม Arduino IDE ขึ้นมา



รูปที่ 1.12 ต่อบอร์ด Arduino เข้ากับคอมพิวเตอร์ผ่านทางพอร์ต USB

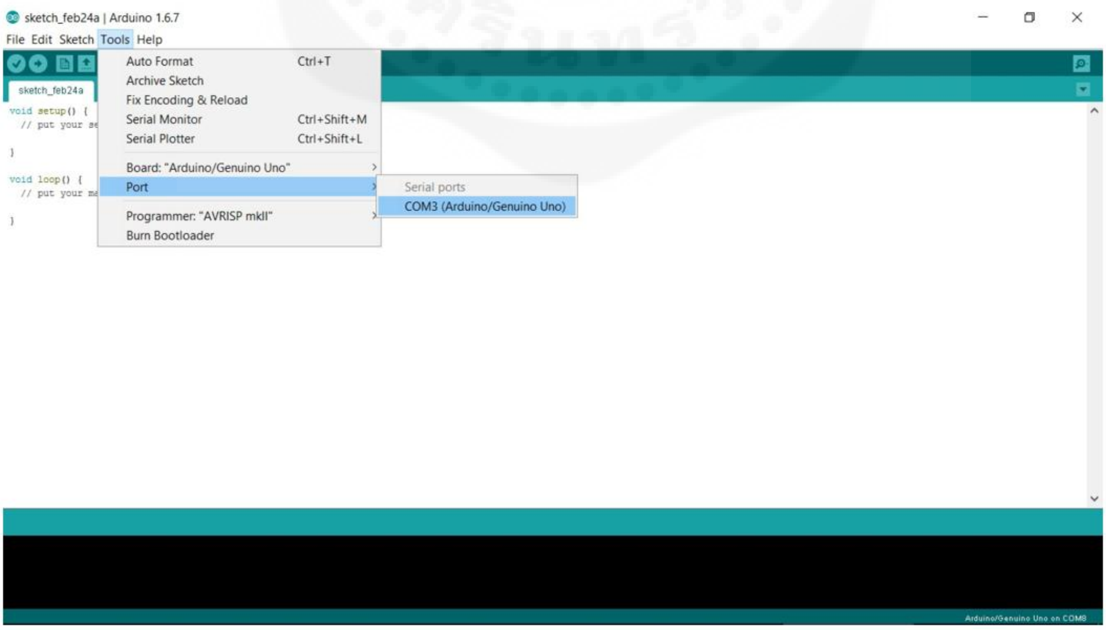

รูปที่ 1.13 โปรแกรม Arduino IDE

หลังจากนั้นเลือกบอร์ด Arduino UNO ที่เมนู Tools




รูปที่ 1.14 การเลือกบอร์ด Arduino UNO

เมื่อเลือกบอร์ด Arduino UNO ให้ทำการเลือก Com port ของคอมพิวเตอร์ที่เชื่อมต่อกับบอร์ด Arduino




รูปที่ 1.15 การเลือก Com port ของคอมพิวเตอร์ที่เชื่อมต่อกับบอร์ด Arduino

ที่โปรแกรม Arduino IDE จะแสดง Com port ของคอมพิวเตอร์ที่เชื่อมต่อกับบอร์ด Arduino ที่มุมจอขวาล่าง

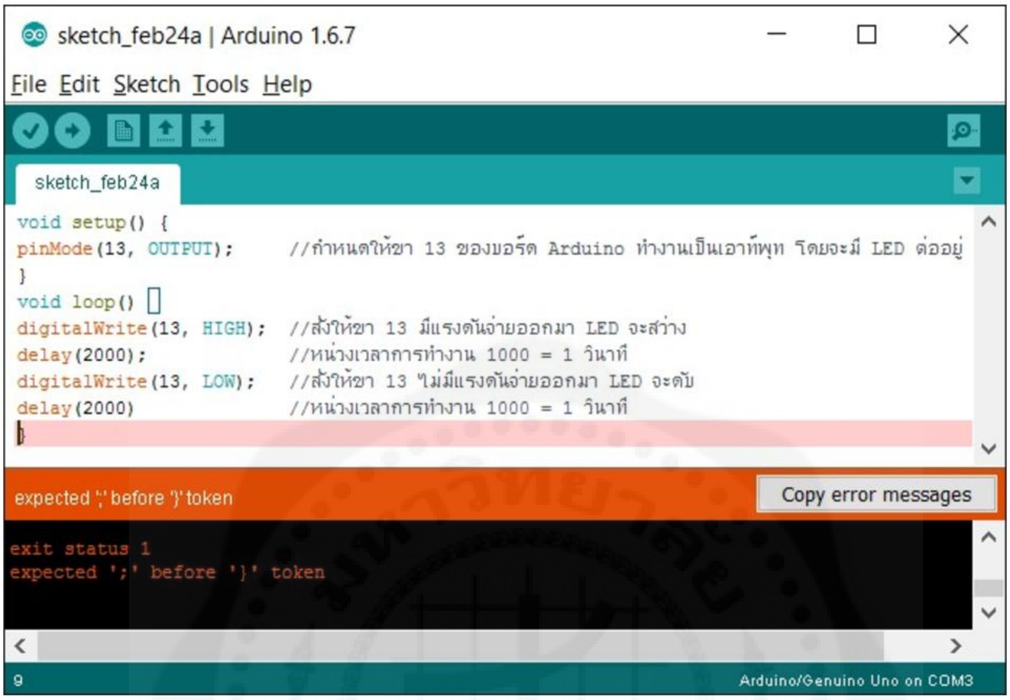

รูปที่ 1.16 Com port ของคอมพิวเตอร์ที่เชื่อมต่อกับบอร์ด Arduino

ทำการเขียนโปรแกรมที่พื้นที่เขียนโปรแกรม

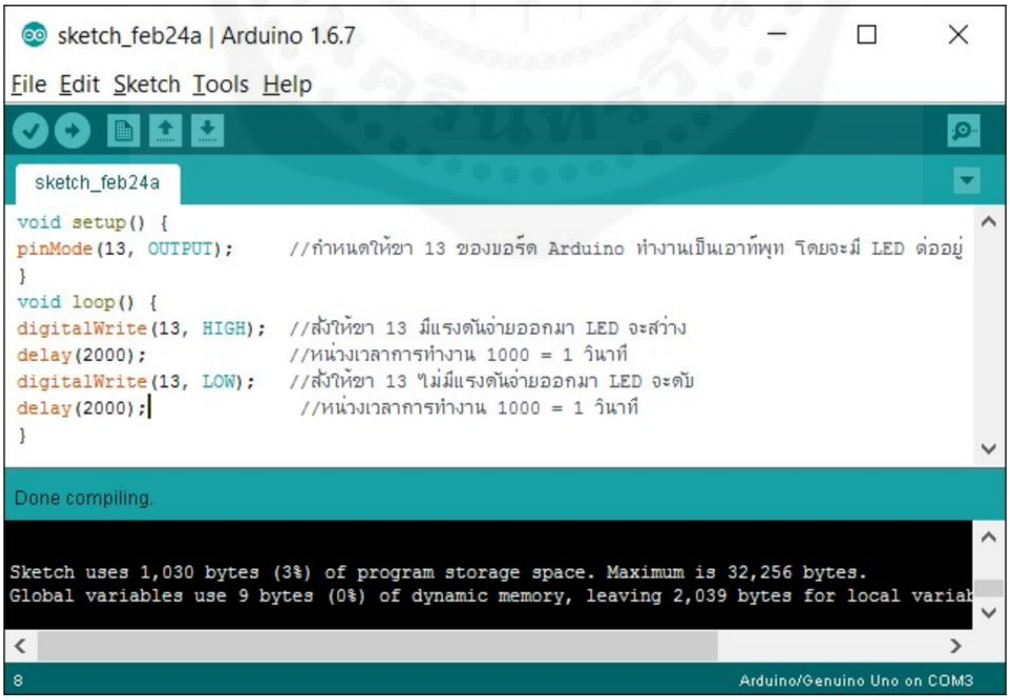

https://www.arduino.cc/en/Tutorial/Blink : 2561

รูปที่ 1.17 โปรแกรมตัวอย่าง

เมื่อเขียนโปแกรมเสร็จทำการตรวจสอบโปรแกรมที่เขียนมีข้อผิดพลาดหรือไม่ โดยกดที่ปุ่มเครื่องหมายถูก (Verify) หากมีข้อผิดพลาดจะมีข้อความแจ้งเตือนขึ้นมา

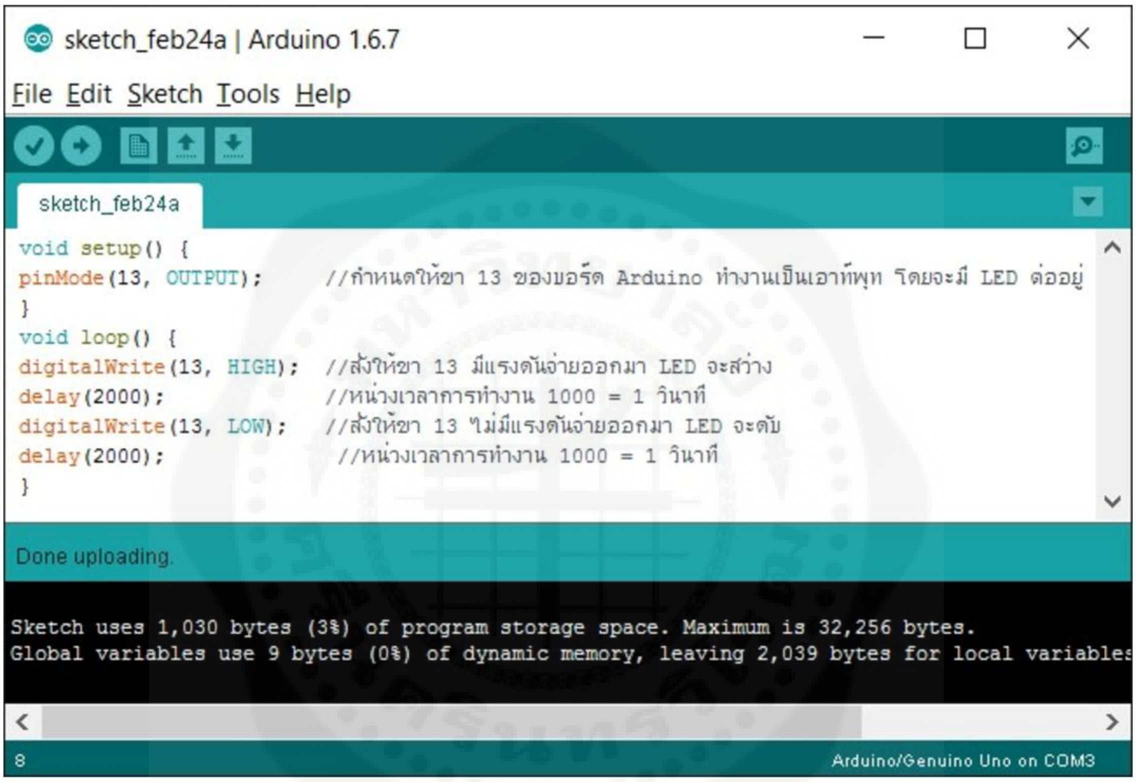



รูปที่ 1.18 เมื่อตรวจสอบโปรแกรมแล้วพบข้อผิดพลาด




รูปที่ 1.19 เมื่อตรวจสอบโปรแกรมแล้วไม่พบข้อผิดพลาด

เมื่อตรวจสอบโปรแกรมแล้วไม่พบข้อผิดพลาด ก็พร้อมที่จะอัพโหลดโปรแกรมลงบอร์ด Arduino โดยกดที่ปุ่มเครื่องหมายลูกศรหันขวา (Upload) เมื่ออัพโหลดเสร็จจะข้อความแจ้งว่าอัพโหลดเสร็จ (Done uploading) โดยระหว่างอัพโหลด สังเกต LED Tx และ Rx บนบอร์ด Arduino จะกระพริบสลับกัน




รูปที่ 1.20 เมื่อโปรแกรมอัพโหลดเสร็จ

เมื่อโปรแกรมอัพโหลดเสร็จ ที่บอร์ด Arduino LED ที่ขา 13 (สีส้ม) ก็จะสว่าง ดับ สลับกันไป ถ้าต้องการให้สว่าง/ดับไวขึ้น ก็ให้ลดค่าตัวเลขของคำสั่ง delay(2000) ลง แต่ถ้าต้องการให้สว่าง/ดับ ช้าลงก็ให้เพิ่มค่าตัวเลขของคำสั่ง delay


รูปที่ 1.21 บอร์ด Arduino เมื่อโปรแกรมอัพโหลดเสร็จ

## สรุป

บอร์อด Arduino เป็นบอร์ดไมโครคอนโทรลเลอร์ที่ถูกออกแบบเพื่อให้ใช้งานได้ง่ายขึ้นกว่าบอร์ดไมโครคอนโทรลเลอร์สมัยก่อน โดยสามารถพัฒนาโปรแกรมควบคุมการทำงานของบอร์ดด้วยโปรแกรม Arduino IDE บนเครื่องคอมพิวเตอร์ และถ่ายโอนโปรแกรมควบคุมการทำงานผ่านการเชื่อมต่อระหว่างคอมพิวเตอร์และบอร์อด Arduino ด้วยพอร์ทยูเอสบี ทำให้สะดวกในการพัฒนาโปรแกรมไม่ต้องถอดไอซีไมโครคอนโทรลเลอร์ออกมาถ่ายโอนโปรแกรมใส่เข้าเหมือนสมัยก่อน

## คำถาม

1. บอร์ด Arduino มีต้นกำเนิดจากประเทศใด
2. โปรแกรมที่ใช้อัพโหลดคำสั่งควบคุมบอร์ด Arduino คือโปรแกรมอะไร
3. บอร์ด Arduino UNO เมื่อเชื่อมต่อกับโปรแกรม Arduino IDE ในครั้งแรกจะต้องตั้งค่า

4. บอร์ด Arduino UNO ใช้ Microcontroller เบอร์อะไร

5. จากรูปที่ 1.20 หากต้องการหน่วงเวลา 5 วินาที จะต้องกำหนดค่า Delay เท่าไร

## เอกสารอ้างอิง

1. Marko Svaljek. (2015). Arduino Succinctly ($1^{st}$ ed.). North Carolina:

Syncfusion Inc.

2. Simon Monk. (2010). 30 Arduino Projects for The Evil Genius ($1^{st}$ ed.).

New York: McGraw-Hill

# บทที่ 2

# โปรแกรมภาษา C บน Arduino

โปรแกรมควบคุมการทำงานบอร์ด Arduino ใช้การเขียนโปรแกรมด้วยรูปแบบของภาษา C โดยทำการเขียนคำสั่งต่าง ๆ ในโปรแกรม Arduino IDE ซึ่งผู้เขียนโปรแกรมไม่จำเป็นต้องมีความรู้โปรแกรมแอสแซมบลี้เหมือนสมัยก่อน โดยรูปแบบการเขียนคำสั่งที่ใช้จะมีความคล้ายคลึงกับการเขียนคำสั่งโปรแกรมภาษา C/C++

## 2.1 โครงสร้างของโปรแกรม

เมื่อเปิดโปรแกรม Arduino IDE ขึ้นมา ในส่วนของพื้นที่สำหรับการเขียนโปรแกรมควบคุมจะมีฟังก์ชั่นถูกกำหนดขึ้นมาสองส่วน คือ void setup ( ) และ void loop ( ) โดยฟังก์ชั่นทั้งสองส่วน จะมีหน้าที่การทำงานที่แตกต่างกัน โดยฟังก์ชั่น void setup ( ) จะทำงานเพียงครั้งเดียวเมื่อเริ่มรันโปรแกรม ใช้สำหรับการกำหนดค่าเริ่มต้นต่าง ๆ ของโปรแกรม ส่วนฟังก์ชั่น void loop ( ) จะเป็นส่วนของโปรแกรมการทำงานควบคุมของบอร์ด Arduino โดยทั้งฟังก์ชั่น void setup ( ) และ void loop ( ) จะต้องเขียนให้อยู่ในรูปแบบที่กำหนดด้วยโปแกรมถึงจะคอมไพล์ผ่านเพื่ออัพโหลดลง Arduino

## 2.2 ฟังก์ชั่น Setup ( )

ฟังก์ชั่น Setup ( ) คือ ฟังก์ชั่นใช้ในการประกาศค่าเริ่มต้น ตำแหน่งพอร์ตที่ใช้งานรวมถึงฟังก์ชั่นที่อยู่ไลบรารี่ที่ใช้งาน เป็นฟังก์ชั่นที่ทำงานเพียงครั้งเดียว จะทำงานทุกครั้ง ที่มีการรีเซต หรือรีบูตเครื่องใหม่ เท่านั้น เช่น

```
void setup ( ){
    Serial.begin(9600); //มีใช้งานการสื่อสารรับส่งข้อมูลผ่านพอร์ตอนุกรมที่ความเร็ว 9600
บิท/วินาที
```

```
    pinMode(Pin7, INPUT); //กำหนดให้ขาดิจิตอลที่ 7 ทำงานในโหมดอินพุต
}
```

### 2.3 ฟังก์ชั่น Loop ( )

ฟังก์ชั่น Loop ( ) จะเป็นส่วนของโปรแกรมการทำงานควบคุมของบอร์ด Arduino หลังจากเรียกโปรแกรม setup() แล้วโปรแกรม loop ( ) จะทำงานต่อ และจะทำงานเป็นวนลูป เพื่อให้โปรแกรมทำงานตามสถานะต่างๆ เช่น

```
void loop( ) {
        digitalWrite(buzzPin, HIGH); // บัซเซอร์ดัง
        delay(5);
        digitalWrite(buzzPin, LOW); // บัซเซอร์เงียบ
        delay(5);
}
```

### 2.4 ฟังก์ชั่นทำงานเฉพาะ

ฟังชั่นเป็นกลุ่มคำสั่งที่มีชื่อเรียกและจะทำงานเมื่อฟังชั่นถูกเรียกใช้ ฟังชั่นที่ทำงานเฉพาะเขียนขึ้นเพื่อให้ทำงานที่ซ้ำๆกันเพื่อลดความยาวของคำสั่ง ในการสร้างฟังชั่นจะเริ่มต้นด้วยการกำหนดชนิดค่าที่ฟังชั่นจะคืนกลับให้โปรแกรมที่เรียกใช้ เช่น int สำหรับฟังก์ชั่นชนิด integer ถ้าฟังชั่นไม่มีการคืนค่าจะใช้คำว่า void รูปแบบการเชียนฟังก์ชั่นจะมีรูแปบบคือ ชื่อฟังชั่นและตามด้วยวงเล็บซึ่งจะเป็นค่าพารามิเตอร์ที่จะส่งค่าให้กับฟังก์ชั่น

```
type functionName(parameters)
{
statements;
}
```

ตัวอย่างการกำหนดฟังก์ชั่นโดยการกำหนดฟังก์ชั่นแบบ integer ชื่อ delayVal() ใช้เพื่อตั้งค่าหน่วงเวลาในโปรแกรมโดยใช้ค่าของตัวต้านทานปรับค่าได้ ซึ่งการทำงานของฟังชั่นจะกำหนดตัวแปร v และนำค่าของ ตัวต้านทานปรับค่าได้ซึ่งมีค่า 0-1023 มาเก็บไว้ใน v จากนั้นหารค่านั้นด้วย 4 เพื่อให้ได้ค่า 0-255 หลังจากนั้นจึงส่งค่ากลับไปยังโปรแกรมหลัก

```
int delayVal()
{
int v; // create temporary variable 'v'
v = analogRead(pot); // read potentiometer value
v /= 4; // converts 0 - 1023 to 0 - 255
return v;
}
```

**2.5 {} วงเล็บปีกกา (curly braces)**

วงเล็บปีกกาใช้เป็นเครื่องหมายเพื่อแสดงจุดเริ่มต้น และจุดสิ้นสุดของกลุ่มคำสั่งของฟังชั่นต่างๆ เช่น setup loop()  void loop() ดังตัวอย่าง

```
void loop( )
{
delay(5);
}
```

วงเล็บปีกกาเปิด { จะต้องตามด้วยวงเล็บปีกกาปิด } เสมอ และในโปรแกรมอาจมีวงเล็บปีกกาเปิดและปิดหลายๆอันก็ได้ แต่จะต้องมีจำนวนวงเล็บปีกกาเปิดและปิดเท่ากัน ถ้าจำนวนไม่เท่ากันจะทำให้เกิดข้อผิดพลาดเวลาคอมไพล์

### 2.6 ; เครื่องหมายเซมิโคลอน (SEMICOLON)

เครื่องหมายเซมิโคลอน ; จะต้องมีที่ส่วนท้ายของคำสั่งทุกบรรทัด หากลืมใส่เซมิโคลอนที่ท้ายบรรทัดจะทำให้เกิดข้อผิดพลาดเวลาคอมไพล์ ดังตัวอย่าง

```
void loop( )
{
int x = 13; // declare variable 'x' as the integer 13
delay(5);
}
```

### 2.7 /*....*/ กล่องคำอธิบาย (BLOCK COMMENTS)

กล่องคำอธิบายสามารถใส่ไว้ในเครื่องหมาย /*......คำอธิบาย....*/ โดยคำอธิบายที่อยู่ในเครื่องหมาย/*...*/ นี้ เวลาคอมไพล์ จะถูกไม่นำมาประมวลผล ซึ่งจะมีประโยชน์ในการอธิบายการทำงานของโปรแกรมเพื่อให้ง่ายในการทำความเข้าใจการทำงานแต่ละส่วนของโปรแกรม ซึ่งจะเริ่มด้วยเครื่องหมาย /* และจบด้วยเครื่องหมาย */

### 2.8 // คำอธิบาย LINE COMMENTS

การเขียนคำอธิบายคำสั่งบรรทัดเดียวจะใช้เครื่องหมาย // เป็นเครื่องหมายเริ่มต้นและสิ้นสุดด้วยการขึ้นบรรทัด ดังตัวอย่าง

```
int x = 13; // declare variable 'x' as the integer 13
```

คำอธิบายคำสั่งบรรทัดเดียวนิยมใช้ เนื่องจากช่วยอธิบายผลของคำสั่งบรรทัดนั้น ๆ

### 2.9 ตัวแปร (Valiable)

ตัวแปรเป็นชื่อ และที่เก็บค่าของตัวเลขที่ใช้งานในโปรแกรม ซึ่งค่าของตัวแปรสามารถเปลี่ยนแปลงได้ซึ่งจะตรงกันข้ามกับค่าคงที่ (Constant) ที่ไม่สามารถเปลี่ยนแปลงค่าได้ การใช้

งานตัวแปรต้องมีการกำหนดชนิดและอาจต้องกำหนดค่าเริ่มต้นด้วย ตัวอย่างด้านล่างนี้เป็นการกำหนดตัวแปรชื่อ inputVariable หลังจากนั้นนำค่าที่ได้จาก analog input pin 2 มาเก็บไว้

```
int inputVariable = 0; // declares a variable and assigns value of 0
inputVariable = analogRead(2); // set variable to value of anlog pin 2
```

จากตัวอย่างคำสั่งบรรทัดที่หนึ่ง ค่าของตัวแปร inputVariable สามารถเปลี่ยนแปลงได้ โดยบรรทัดแรกกำหนดให้เป็นข้อมูลชนิด int (ตัวเลขจำนวนเต็ม) โดยกำหนดให้มีค่าเริ่มต้นเท่ากับ 0 สำหรับบรรทัดที่สองกำหนดให้ตัวแปร inputVariable มีค่าเท่ากับข้อมูลจากขา analog ขา 2 ดังนั้นเมื่อตัวแปรถูกกำหนดขึ้นหรือมีค่าขึ้นมาใหม่เราสามารถตรวจสอบค่าของตัวแปรว่ามีค่าตามที่ต้องการ หรือเราจะนำค่านั้นไปใช้เลยก็ได้ ตัวอย่างด้านล่างนี้เป็นตัวอย่างที่ใช้งานตัวแปร คือตรวจสอบค่าตัวแปร inputVariable ว่ามีค่าต่ำกว่า 100 หรือไม่ ถ้ามีค่าต่ำกว่า 100 ให้กำหนดค่าตัวแปรให้เท่ากับ 100 แต่ถ้ามากกว่า 100 ก็ให้ใช้ค่านั้น หลังจากนั้นตั้งค่าหน่วงเวลาตามค่าตัวแปร inputVariable ที่ตรวจสอบ ซึ่งก็คือค่า 100 เป็นอย่างต่ำ

```
if (inputVariable < 100) // test variable if less than 100
{
inputVariable = 100; // if true assigns value of 100
}
delay(inputVariable); // uses variables as delay
```

### 2.10 การแจกแจงตัวแปร (variable declaration)

ตัวแปรที่จะนำไปใช้ในโปรแกรมจะต้องถูกระบุชนิดตัวแปรให้เป็น int, long, float, เป็นต้น ก่อนนำไปใช้ ซึ่งการระบุชนิดนี้จะทำเพียงครั้งเดียวในโปรแกรม แต่ค่าของตัวแปรจะเปลี่ยนแปลงไปตามการทำงานของโปรแกรม ตัวอย่างต่อไปนี้เป็นการระบุค่าตัวแปร inputVariable ให้เป็นตัวแปรจำนวนเต็ม หรือ integerและกำหนดเริ่มต้นให้เป็นศูนย์

int inputVariable = 0;

**2.11 ขอบเขตของตัวแปร**

ตัวแปรสามารถตั้งค่าตอนเริ่มต้นโปรแกรมก่อน void setup() หรือภายในฟังชั่น และอาจจะตั้งค่าตัวแปรภายในกลุ่มคำสั่ง for loop ก็ได้ ซึ่งการตั้งค่าตัวแปรในแบบต่างๆ จะมีผลถึงขอบเขตการใช้ตัวแปร ตัวแปรชนิด global เป็นตัวแปรที่โปรแกรมมองเห็นและใช้งานได้จากทุกฟังชั่นและทุกกลุ่มคำสั่งในโปรแกรม ตัวแปรนี้จะตั้งค่าที่ตอนเริ่มต้นโปรแกรมก่อนคำสั่ง setup() ตัวแปรชนิด local เป็นตัวแปรที่ตั้งค่าภายในฟังชั่นหรือภายในกลุ่มคำสั่ง for loop ตัวแปรชนิดนี้จะมองเห็นและใช้งานได้เฉพาะภายในฟังชั่นที่ตัวแปรถูกตั้งค่า ตัวอย่างด้านล่างนี้แสดงการตั้งค่าตัวแปรและขอบเขตการใช้งานของตัวแปร

```
int value; // 'value' is visible to any function
void setup();
{
// no setup needded
}
void loop();
{
for (int i=0; i<20;) // 'i is only visible inside the for - loop
{
i++;
}
float f; // 'f' is only visible inside loop
}
```

### 2.12 ตัวแปรชนิดต่างๆ

**byte**

ตัวแปร byte ใช้เก็บตัวเลขขนาด 8 bit ไม่มีทศนิยม มีค่า 0 – 255 ดังตัวอย่าง

```
byte someVariable = 180; // declares 'someVariable' as a byte type
```

**int**

ตัวแปร integer เป็นตัวแปรพื้นฐานที่เก็บตัวเลขโดยไม่มีจุดทศนิยม และเก็บค่าแบบ 16 bit มีค่าระหว่าง 32,767 ถึง -32,768 ดังตัวอย่าง

```
int someVariable = 1500; // declares 'someVariable' as an integer type
```

ตัวแปร integer จะมีค่าย้อนกลับ (roll over) ถ้าถูกใส่ค่าเกินจากค่าสูงสุดที่รับได้ เช่นถ้า x = 32767 และคำสั่งถัดมาให้เพิ่มค่า 1 ให้กับ x , x = x + 1 หรือ x++, ค่า x จะล้นหรือที่เรียกว่า roll over เป็นค่า -32768

**long**

ตัวแปร long เป็นตัวแปรจำนวนเต็มแบบขยายโดยไม่มีจุดทศนิยม เก็บค่าแบบ 32 bit มีค่าระหว่าง 2,147,483,647 ถึง -2,147,483,648 ดังตัวอย่าง

```
long someVariable = 90000; // declares 'someVariable' as a long type
```

**float**

ตัวแปร float เป็นตัวแปรที่มีจุดทศนิยม ตัวแปรนี้มีค่ามากกว่าค่าของตัวแปรจำนวนเต็ม เก็บค่าแบบ 32 bit มีค่าระหว่าง 3.4028235E+38 ถึง -3.4028235E+38 ดังตัวอย่าง

float someVariable = 3.14; // declares 'someVariable' as a floatint -point type

**arrays**

ตัวแปร arrays เป็นตัวแปรที่สามารถเข้าถึงได้ด้วยค่าตัวชี้หรือ index ค่าตัวแปรใน array อาจเรียกใช้โดยระบุชื่อ array และระบุตัวชี้ index number ตัวแปร array จะมี index เริ่มต้นจาก 0 ตัวแปร array จะต้องตั้งค่า ก่อนจะนำไปใช้งาน และจะกำหนดค่าเริ่มต้นหรือไม่ก็ได้ ดังตัวอย่าง

int myArray[] = {value0, value1, value2...}

เราอาจตั้งค่า array โดยกำหนดชนิดและขนาด แล้วค่อยใส่ค่าลงไปทีหลังก็ได้ เช่น

int myArrays[5]; // declares integer array w/ 6 positions

myArray[3] = 10; // assigns the 4th index the value 10

การดึงค่าจาก array ใช้การเรียกค่าจากตัวแปร array และระบุตำแหน่งตัวชี้ index

x = myArray[3]; // x now equals 10

**2.13 arithmetic**

ตัวกระทำการทางคณิตศาสตร์หมายถึง การบวก, ลบ, คูณ, และหาร ซึ่งจะให้ค่าผลรวม, ผลต่าง, ผลคูณ, หรือผลการหาร ของสองตัวดำเนินการ

y = y + 3;

x = x - 7;

i = j * 6;

r = r / 5

การทำงานของโปรแกรมจะเป็นการนำขนิดข้อมูลตัวแปรของตัวดำเนินการมาใช้ เช่น 9/4 ผลลัพท์จะเท่ากับ 2 แทนที่จะเท่ากับ 2.25 เนื่องจาก 9 และ 4 เป็นตัวแปรจำนวนเต็ม หรือ integer ซึ่งไม่สามารถเก็บค่าทศนิยมได้ และถ้าผลลัพท์ที่ได้มีค่ามากกว่าที่ตัวแปรจะรับได้ มันจะเกิดการ overflow หรือค่าย้อนกลับ ถ้ามีการคำนวนของชนิดตัวแปรที่ต่างกัน ตัวแปรที่มีขนาดใหญ่กว่าจะถูกนำมาใช้ เช่นการบวก ลบเลขจำนวนเต็มกับเลขทศนิยม การบวกลบ

ทศนิยมจะถูกนำมาใช้ ดังนั้นจึงควรเลือกตัวแปรที่มีขนาดที่ใหญ่พอที่จะเก็บผลลัพท์การคำนวน เราสามารถเปลี่ยนชนิดตัวแปรจากชนิดหนึ่งไปยังอีกชนิดหนึ่งได้ด้วยคำสั่ง (int)myFloat myFloat เช่น i = (int)3.6 จะเป็นการแปลงค่าให้ i มีค่าเป็น 3

### 2.14 compound assignment

การเขียนคำสั่งแบบย่อ (**compound assignment)** คือการเขียนคำสั่งเกี่ยวกับตัวแปร แล้วนำผลลัพท์ที่ได้ไปเก็บในตัวแปรนั้นเหมือนเดิม การเขียนคำสั่งแบบย่อที่ใช้ทั่วไป เช่น

x ++ เหมือนกับคำสั่ง x = x + 1

x -- เหมือนกับคำสั่ง x = x – 1

x += y เหมือนกับคำสั่ง x = x + y

x -= y เหมือนกับคำสั่ง x = x – y

x *= y เหมือนกับคำสั่ง x = x * y

x /= y เหมือนกับคำสั่ง x = x / y

### 2.15 comparison operator

การเขียนคำสั่งเปรียบเทียบตัวแปรหรือค่าคงที่กับค่าอื่นๆ ใช้เพื่อตรวจค่าว่ามีสถานะตรงตามที่กำหนดหรือไม่ ตัวอย่างต่อไปนี้เป็นคำสิ่งเปรียบเทียบตัวแปรสองตัว การเขียนคำสั่งเปรียบเทียบที่ใช้ทั่วไป เช่น

x == y หมายถึง x มีค่าเท่ากับ y

x != y หมายถึง x มีค่าไม่เท่ากับ y

x < y หมายถึง x มีค่าน้อยกว่า y

x > y หมายถึง x มีค่ามากกว่า y

x <= y หมายถึง x มีค่าน้อยกว่าหรือเท่ากับ y

x >= y หมายถึง x มีค่ามากกว่าหรือเท่ากับ y

### 2.16 logical operators

คำสั่งตรวจสอบทางลอจิกโดยใช้เปรียบเทียบตัวดำเนินการสองตัวและให้ผลลัพท์ จริง หรือ เท็จ ขึ้นอยู่กับตัวดำเนินการ โดยตัวดำเนินการทางลอจิกมี 3 ตัวคือ AND OR และ NOT คำสั่งตรวจสอบทางลอจิกดังตัวอย่าง

Logical AND :

if (x > 0 && x < 5) หมายถึง เป็นจริงเมื่อ x มากกว่า 0 และ x น้อยกว่า 5

Logical OR:

if (x > 0 || y < 5) หมายถึง เป็นจริงเมื่อ x มากกว่า 0 หรือ x น้อยกว่า 5

### 2.17 constants

ค่าที่กำหนดให้เป็นค่าคงที่ จะเรียกค่านี้ว่า constant ค่าคงที่มีไว้เพื่อกำหนดข้อมูลตัวเลขที่อยู่ในโปรแกรม

### 2.18 true/false

ค่าคงที่เราสามารถกำหนดให้เป็นข้อมูลชนิด boolean ใช้บอกระดับลอจิก 0 หมายถึง FALSE และระดับลอจิก 1 หมายถึง TRUE ดังตัวอย่าง

```
if (b == TRUE);

{

x = 5 ;
}
```

### 2.19 high/low

การกำหนดระดับลอจิกที่ขาไอซีว่าเป็น HIGH หรือ LOW โดยค่า HIGH จะแทนระดับลอจิก 1 หรือ 5 โวลท์และค่า LOW จะแทนระดับลอจิก 0 หรือ 0 โวลท์ ดังตัวอย่าง

digitalWrite(13, HIGH) ; หมายถึงกำหนดให้ขา 13 มีค่าเป็นลอจิก 1

### 2.20 input/output

เป็นการกำหนดค่าในฟังชั่น pinMode() เพื่อกำหนดหน้าที่ของขาไอซีให้ทำหน้าที่เป็นอินพุทหรือเอาท์พุท ดังตัวอย่าง

pinMode(13, OUTPUT) ; หมายถึงกำหนดให้ขา 13 ทำหน้าที่เป็นเอาท์พุท

### 2.21 if

คำสั่ง if ใช้ตรวจสถานะที่ต้องการว่าเป็นไปตามเงื่อนไขที่กำหนดหรือไม่

### 2.22 if… else

if… else เป็นการเพิ่มทางเลือกหากไม่เป็นไปตามเงื่อนไขแรก เช่น ตรวจสอบขาอินพุท ถ้าเป็น HIGH กำหนดให้ x = 5 แต่ถ้าขาอินพุทเป็น LOW กำหนดให้ x = 1

```
if (inputPin == HIGH)
{
X = 5 ;
}
else
{
X = 1 ;
}
```

### 2.23 for

คำสั่ง for ใช้เพื่อทำคำสั่งวนลูปหลาย ๆครั้ง ตามจำนวนที่กำหนด แล้วจึงออกจากวงลูป โดยมีรูปแบบดังนี้

```
for (initialization; condition ; expression)
{
doSomething;
}
```

initialization คือการกำหนดค่าเริ่มต้นของตัวแปรและหลังจากนั้นจะทำงานตามคำสั่งในลูปไปจนครบรอบ ค่า initialization จะเพิ่มค่าตามค่าที่กำหนดใน expression แล้วทดสอบรอบการทำงานตามเงื่อนไขหรือ condition ที่กำหนดไว้ ถ้าเงื่อนไขเป็นจริง ก็จะทำงานตามคำสั่งในลูปวนไปจนเงื่อนไขเป็นเท็จจึงจะออกจากลูป ตัวอย่างต่อไปนี้กำหนดค่าค่าเริ่มต้นให้ตัวแปร i เป็นศูนย์ ทดสอบค่า i มีค่าต่ำกว่า 20 ถ้าเป็นจริง เพิ่มค่า i ครั้งละ 1 และทำคำสั่งในวงเล็บปีกกา วนลูปไปจนกว่าค่า I จะเท่ากับ 20 จึงออกจากลูป

```
for (int i=0; i < 20; i++) ;
{
degitalWrite(13, HIGH) ;
}
```

### 2.24 while

คำสั่ง while เป็นคำสั่งที่จะทำงานวนลูปต่อเนื่องไปจนกว่าเงื่อนไขในวงเล็บจะเป็นเท็จ ซึ่งจะต้องมีสิ่งที่ทำให้มีค่าตัวแปรที่ใช้ทดสอบเกิดการเปลี่ยนค่า เช่นเพิ่มค่าตัวแปร หรือทดสอบตรวจสอบ sensor เป็นต้น โดยมีรูปแบบดังนี้

```
while (someVariable ?? value) // test if less than 200
{
doSomething; // executes enclosed statements
```

```
someVariable++; // increments variable by 1
}
```

### 2.25 do... while

คำสั่ง do ลูปจะทำงานและทดสอบสถานะที่ด้านท้ายของ loop คล้ายกับคำสั่ง while ยกเว้นการตรวจสอบสถานะเพื่อออกจากลูป จะทำที่ส่วนท้ายของลูป ดังนั้นคำสั่ง do ลูปจะทำงานอย่างน้อยหนึ่งครั้งเสมอ โดยมีรูปแบบดังนี้

```
do
{
doSomething;
}
while (someVariable ?? value);
```

ตัวอย่างต่อไปนี้อ่านค่า readsensor() แล้วเก็บค่าที่อ่านไว้ที่ตัวแปร x  จากนั้นหน่วงเวลา 50 mS และวนลูปจนกระทั่งตัวแปร x มีค่าน้อยกว่า 100

```
do
{
x = readSensor(); // assigns the value of readSensors() to x
delay(50); // delay 50 milliseconds
}
while (x < 100); // loop if x is less than 100
```

### 2.26 pinMode(pin, mode)

เป็นคำสั่งที่ใช้ในกลุ่ม void setup() เพื่อกำหนดให้ขาอาดูโน่ที่เป็นขารับส่งสัญญาณแบบดิจิทัลทำงานเป็นขารับสัญญาณ หรือขาส่งสัญญาณ โดยมีรูปแบบดังนี้

pingMode(pin, OUTPUT) ; // กำหนดให้ ‘pin’ เป็น output

**2.27 digitalRead(pin)**

เป็นคำสั่งที่ใช้อ่านค่าจากขาอาดูโน่ที่เป็นขารับส่งข้อมูลแบบดิจิทัล (digital pin) โดยข้อมูลที่ได้จะเป็น HIGH หรือ LOW โดยมีรูปแบบดังนี้

value = digitalRead(pin);  // กำหนดให้ ‘value’ มีค่าเท่ากับข้อมูลที่ได้จากขา ‘pin’

**2.28 digitalWrite(pin, value)**

เป็นคำสั่งที่ใช้ส่งค่าให้ขาอาดูโน่ ที่เป็นขารับส่งข้อมูลแบบดิจิทัล (digital pin) โดยข้อมูลที่ส่งจะเป็น HIGH หรือ LOW โดยมีรูปแบบดังนี้

digitalWrite(pin, HIGH); // กำหนดให้ขา ‘pin’ มีค่าเท่ากับ HIGH

**2.29 analogRead(pin)**

เป็นคำสั่งเพื่อใช้อ่านค่าจากขา Analog โดยบอร์ดอาดูโน่จะมีวงจร A/D ขนาด 10 bit เพื่อแปลงค่าแรงดันที่อ่านได้จากขา Analog โดยผลลัพท์จะเป็นเลขจำนวนเต็มค่า 0 – 1023 และขา Analog จะไม่เหมือนกับขา digital  คือไม่ต้องกำหนดตอนเริ่มต้นว่าเป็นขา INPUT หรือขา OUTPUT  โดยมีรูปแบบดังนี้

value = analogRead(pin); // กำหนดให้ ‘value’ มีค่าเท่ากับข้อมูลจากขา ‘pin’

**2.30 analogWrite(pin, value)**

เป็นคำสั่งที่ให้ค่าแรงดันที่ขา Analog เปลี่ยนไปตามข้อมูลที่กำหนด เพื่อให้ได้สัญญาณในรูปของ Pulse Width Modulation (PWM) ถ้าค่า 0 แรงดันที่ขา Analog จะเป็น 0 โวลท์คงที่ ส่วนค่า 255 จะเป็น 5 โวลท์คงที่เช่นกัน แต่ค่าระหว่าง 0 - 255 ค่าแรงดันที่ขา Analog จะเปลี่ยนไปเป็นค่าที่อยู่ระหว่าง 0 และ 5 โวลท์ โดยค่ายิ่งมากแรงดันที่ขา Analog ก็จะเข้าใกล้ 5 โวลท์ เช่น ค่า 64 จะทำให้เกิดแรงดัน 0 โวลท์เป็นเวลาสามในสี่ของคาบเวลาทั้งหมด และเป็น 5 โวลท์หนึ่งในสี่ของคาบเวลาทั้งหมด  แต่ถ้าค่าที่เขียนคือ 128 จะทำให้เกิดแรงดัน 0 โวลท์ครึ่งหนึ่งของคาบเวลาทั้งหมดและแรงดัน 5 โวลท์อีกครึ่งหนึ่งของคาบเวลาทั้งหมดา ถ้าเป็นค่า

192 จะทำให้เกิดแรง 0 โวลท์หนึ่งในสี่ของคาบเวลาทั้งหมดและแรงดัน 5 โวลท์ สามในสี่ของคาบเวลาทั้งหมด โดยมีรูปแบบดังนี้

```
analogWrite(pin, value); // เขียนค่าตามที่ขา ‘pin’ ตามค่าของ ‘value’
```

**2.31 delay(ms)**

เป็นคำสั่งที่ใช้หน่วงเวลาการทำงานของโปรแกรมเป็นเวลาตามค่าที่กำหนดเป็นมิลลิวินาที โดยค่า 1000 จะเท่ากับการหน่วงเวลา 1 วินาที

```
delay(1000); // หน่วงเวลา 1 วินาที
```

**2.32 millis()**

เป็นคำสั่งที่กำหนดให้ผลลัพท์ได้ค่าเวลาเป็นมิลลิวินาที โดยมีรูปแบบดังนี้

```
value = millis(); // กำหนดให้ ‘value’ มีค่าเท่ากับ millis()
```

**2.33 Serial.begin(rate)**

เป็นคำสั่งที่ใช้เปิดพอร์ทอนุกรมของอาดูโนให้ทำงาน และกำหนดค่าความเร็วในการรับส่งข้อมูล โดยปกติจะกำหนดไว้ที่ความเร็ว 9600 บิท/วินาที โดยมีรูปแบบดังนี้

```
void setup()
{
Serial.begin(9600); // เปิดพอร์ทอนุกรมและกำหนดค่าความเร็วในการรับส่งข้อมูลเท่ากับ
9600 บิท/วินาที
}
```

### 2.34 Serial.print(data)

เป็นคำสั่งที่ส่งข้อมูลให้กับพอร์ทอนุกรม และสามารถดูข้อมูลได้ที่เมนู Serial monitor ของโปรแรกม Arduino IDEได้ด้วย โดยมีรูปแบบดังนี้

Serial.print(analogvalue); // ส่งค่าของตัวแปร 'analogValue' ไปที่พอร์ทอนุกรม

### 2.35 Serial.println(data)

เป็นคำสั่งที่ส่งข้อมูลให้กับพอร์ทอนุกรม แล้วตามด้วยการขึ้นบรรทัดใหม่โดยอัตโนมัติ คำสั่งนี้จะมีผลเหมือนกับคำสั่ง Serial.print()และสามารถดูข้อมูลได้ที่เมนู Serial monitor ของโปรแรกม Arduino IDEได้ด้วย โดยมีรูปแบบดังนี้

Serial.println(analogvalue); // ส่งค่าของตัวแปร 'analogValue' ไปที่พอร์ทอนุกรม

**สรุป**

การเขียนโปรแกรมควบคุมการทำงาน Arduino จะใช้การเขียนโปรแกรมด้วยรูปแบบของภาษา C ซึ่งผู้เขียนโปรแกรมไม่จำเป็นต้องมีความรู้โปรแกรมแอสแซมบลี้เหมือนสมัยก่อน โดยรูปแบบของโปรแกรมของ Arduino ที่ใช้จะมีความคล้ายคลึงกับโปรแกรมภาษา C/C++

**คำถาม**

1. โปรแกรม Arduino IDE ในส่วนของพื้นที่สำหรับการเขียนโปรแกรมควบคุมจะมีฟังก์ชั่นที่ถูกกำหนดขึ้น 2 ส่วน คืออะไรบ้าง
2. void setup ( ) มีลักษณะการทำงานอย่างไร
3. void loop ( ) มีลักษณะการทำงานอย่างไร
4. ตัวแปรชนิดต่างๆ ในภาษาซีมีอะไรบ้าง และมีความหมายอย่างไร
5. คำสั่ง x == y มีความหมายอย่างไร

## อ้างอิง

1. Simon Monk. (2014). Programming Arduino™ Next Steps. New York: McGraw-Hill Education (Publisher).

2. Richard Blum. (2015). Sams Teach Yourself Arduino™ Programmingin 24 Hours. Indiana: Pearson Education, Inc.

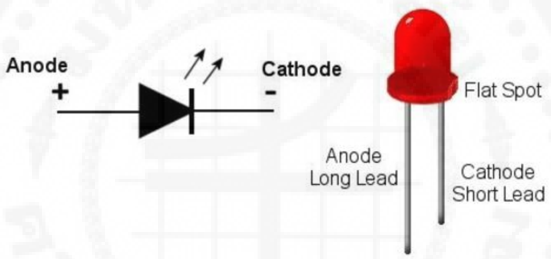
3. Brian w. evans. (2008). Arduino programming notebook. California: Creative Commons.

# บทที่ 3

# อุปกรณ์อินพุท เอาท์พุทพื้นฐาน

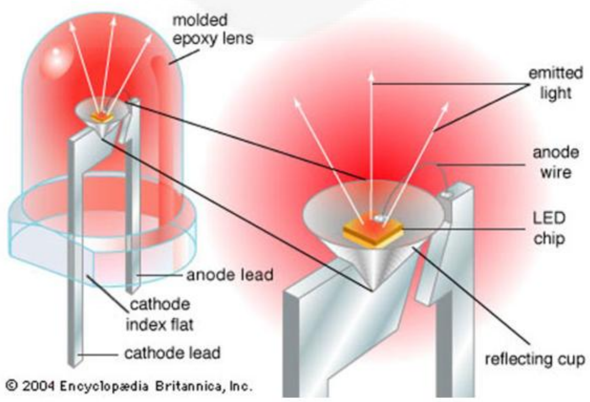
## 3.1 แอลอีดี (LED)

แอลอีดี หรือไดโอดเปล่งแสงเป็นไดโอดชนิดหนึ่งที่จะเปล่งแสงออกมาเมื่อมีการต่อไบอัสตรงโดยสีของแสงที่เปล่งออกมาจะขึ้นอยู่กับชนิดของสารเจือที่อยู่ในสารกึ่งตัวนำที่นำมาทำไดโอดเปล่งแสง และเปล่งแสงได้ใกล้ช่วงแสงอัลตราไวโอเลต ช่วงแสงที่มองเห็น และช่วงแสงอินฟราเรด สำหรับไดโอดเปล่งแสงจะมีสัญลักษณ์ดังรูปที่ 1. และโครงสร้างดังรูปที่ 2.




http://www.mainbyte.com/ti99/electronics/led.html : 2561

รูปที่ 3.1 สัญลักษณ์ของ LED

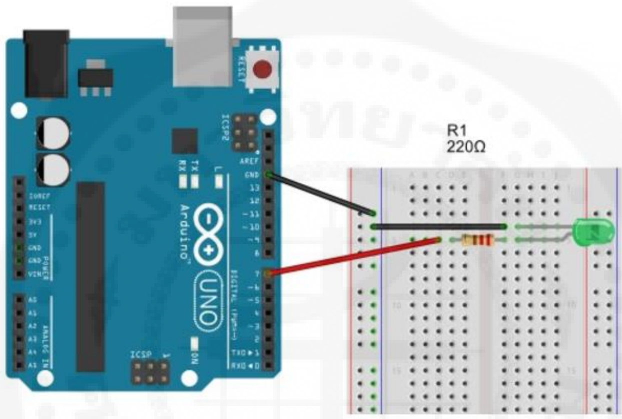



http://www.vcharkarn.com/varticle/38778 : 2561

รูปที่ 3.2 โครงสร้างของ LED

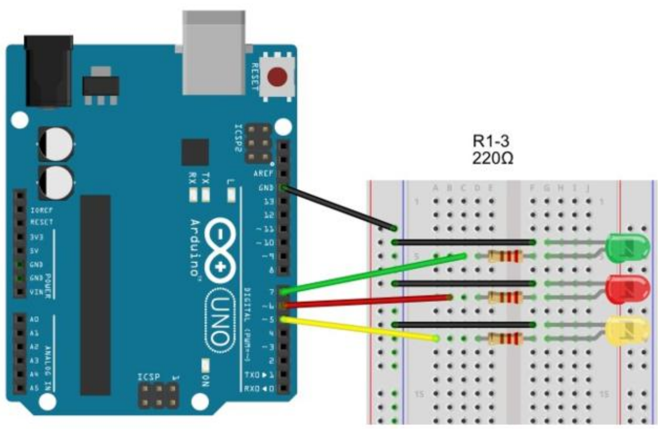
## 3.2 การต่อใช้งาน LED กับบอร์ด Arduino

สำหรับการต่อใช้งานกับบอร์ด Arduino จะใช้ LED เป็นอุปกรณ์ในการแสดงผลการทำงานนั้นคือเป็นอุปกรณ์เอาท์พุท โดยจะต้องต่อตัวต้านทานค่า 220 โอห์มต่ออนุกรมกับ LED เพื่อลดแรงดันให้กับ LED เนื่องจาก LED จะทำงานที่แรงดันประมาณ 1.8 V แต่บอร์ด Arduino จะจ่ายแรงดันออกมาที่ 5 V ไม่เช่นนั้น LED จะชำรุด รูปแบบการต่อตัวต้านทานค่า 220 โอห์มต่ออนุกรมกับ LED ดังรูปที่ 3.3




รูปที่ 3.3 ตัวต้านทานค่า 220 โอห์มต่ออนุกรมกับ LED เพื่อลดแรงดัน

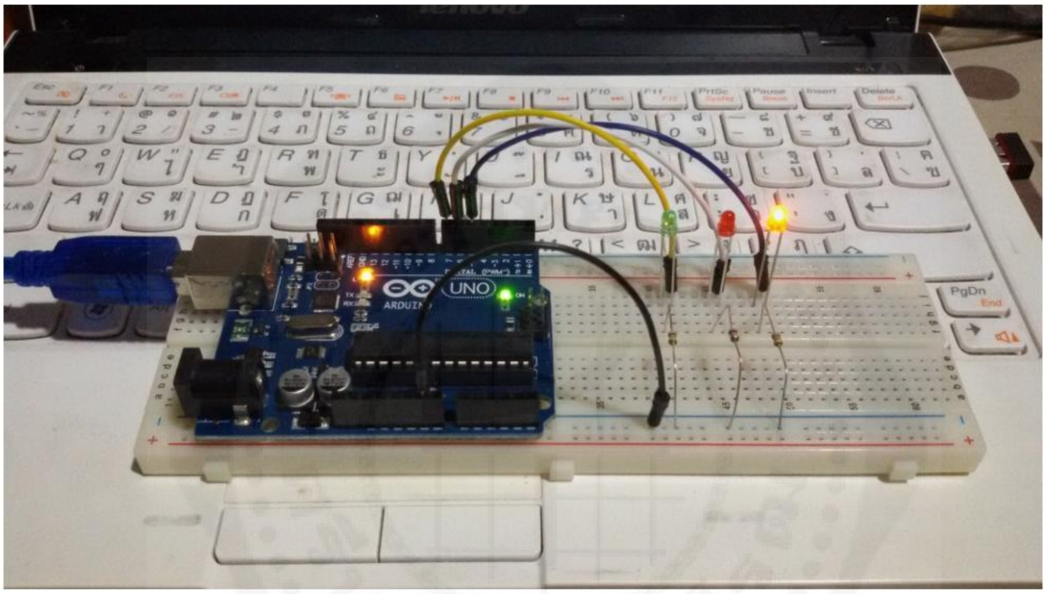
**ตัวอย่างที่ 1.** ต่อ LED 3 ดวง ให้ติดสว่างทีละดวงเรียงกันไปแล้ววนกลับมาใหม่ กำหนดให้ใช้ Digital I/O ขา 5, 6 และ 7 ของบอร์ด Arduino สำหรับต่อ LED




รูปที่ 3.4 การต่อวงจรตามตัวอย่างที่ 1.

```
void setup() {
pinMode(7, OUTPUT); // กำหนดตำแหน่ง LED เขียว
pinMode(6, OUTPUT); // กำหนดตำแหน่ง LED  แดง
pinMode(5, OUTPUT); // กำหนดตำแหน่ง LED เหลือง   }
void loop() {
// สั่งให้ LED เขียว สว่าง
digitalWrite(7, HIGH);
digitalWrite(6, LOW);
digitalWrite(5, LOW);
delay(1000);
// สั่งให้ LED แดง สว่าง
digitalWrite(7, LOW);
digitalWrite(6, HIGH);
digitalWrite(5, LOW);
delay(1000);
// สั่งให้ LED เหลือง สว่าง
digitalWrite(7, LOW);
```

```
digitalWrite(6, LOW);

digitalWrite(5, HIGH);

delay(1000);  }
```




รูปที่ 3.5 การต่อใช้งานจริงตามตัวอย่างที่ 1.

**การทำงานของโปรแกรม**

โปรแกรมจะสั่งให้ LED เขียวสว่างด้วยคำสั่ง digitalWrite(7, HIGH) และดับด้วยสั่ง digitalWrite(7, LOW) สั่งให้ LED แดงสว่างด้วยคำสั่ง digitalWrite(6, HIGH) และดับด้วยสั่ง digitalWrite(6, LOW) สั่งให้ LED เหลืองสว่างด้วยคำสั่ง digitalWrite(5, HIGH) และดับด้วยสั่ง digitalWrite(5, LOW)

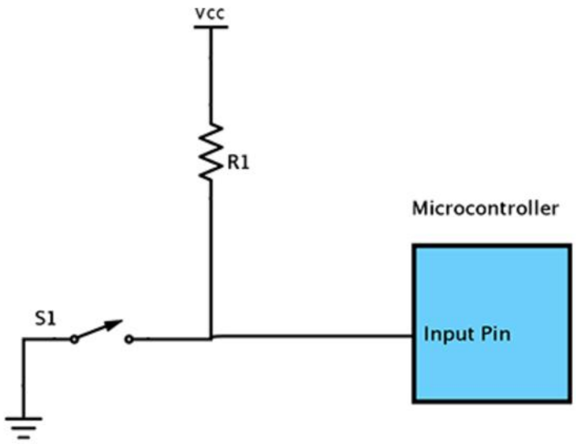
**3.3 สวิทช์กดติดปล่อยดับ** (Push Button)

http://www.porjaielectronicsolar.com/category/12/สวิทช์กด-push-button-switch : 2561

รูปที่ 3.6 สวิทช์กดติดปล่อยดับแบบต่างๆ

สวิทช์กดติดปล่อยดับ ส่วนใหญ่จะเป็นสวิทช์ขนาดเล็กหน้าสัมผัสในการต่อวงจรทนกระแสได้ไม่สูงนิยมนำมาใช้เป็นอุปกรณ์ในการกระตุ้นวงจรให้วงจรเปลี่ยนสภาวะในงานควบคุม เช่นการปิดเปิดการทำงานของวงจร โดยต่อใช้งานร่วมกับวงจรอิเล็กทรอนิกส์อีกทีหนึ่งเพื่อให้วงจรสามารถทำงานได้ ซึ่งถ้านำมาต่อใช้งานกับวงจรดิจิตอล จะต้องกำหนดสถานะของสวิทช์ไว้ก่อนการกด ซึ่งจะมีอยู่ด้วยกัน 2 แบบคือ แบบ Pull-up และแบบ Pull-down

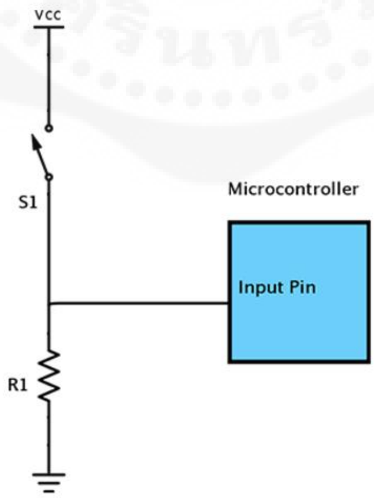
- แบบ Pull-up คือการต่อวงจรตัวต้านทานอนุกรมกับสวิทช์โดยการนำตัวต้านทานต่อเข้ากับแหล่งจ่าย Vcc เพื่อให้แรงดันที่จุดรับค่าอยู่ในสถานะลอจิก 1 ตลอดเวลาดังรูปที่ 3.7 และเมื่อกดสวิตซ์ กระแสไฟฟ้าจะไหลครบวงจรลงกราวด์ทำให้แรงดันที่จุดรับค่าอยู่ในสถานะลอจิก 0 ดังนั้นการทำงานลักษณะนี้จะเรียกว่า **Active Low** เพราะว่าจะเขียนโปรแกรมให้ทำงาน เมื่อได้รับสัญญาณลอจิกเป็น 0




http://www.electronicshub.org/applications-of-resistors/ : 2561

รูปที่ 3.7 วงจร Pull-up

- แบบ Pull-down คือการต่อวงจรตัวต้านทานอนุกรมกับสวิทช์โดยการนำตัวต้านทานต่อเข้าลงกราวด์เพื่อให้แรงดันที่จุดรับค่าอยู่ในสถานะลอจิก 0 ตลอดเวลาดังรูปที่ 3.8 และเมื่อกดสวิตซ์ กระแสไฟฟ้าจะไหลครบวงจรทำให้แรงดันที่จุดรับค่าอยู่ในสถานะลอจิก 1 ดังนั้นการทำงานลักษณะนี้จะเรียกว่า **Active High** เพราะว่าจะเขียนโปรแกรมให้ทำงาน เมื่อได้รับสัญญาณลอจิกเป็น 1

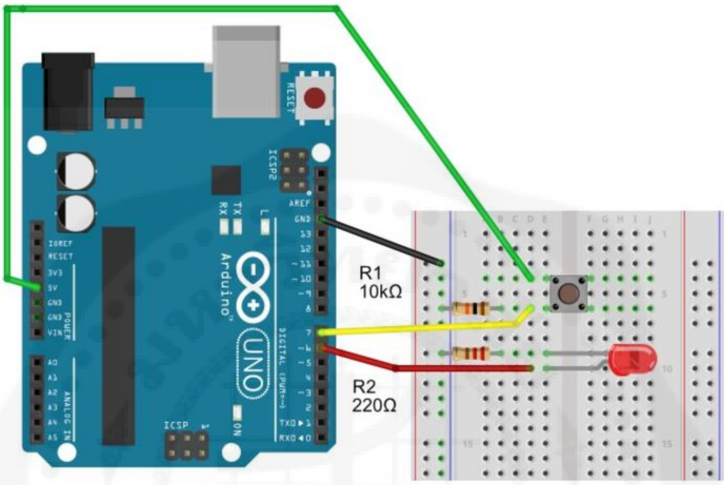



http://www.electronicshub.org/applications-of-resistors/ : 2561

รูปที่ 3.8 วงจร Pull-down

### 3.4 การต่อใช้งานสวิทช์กับบอร์ด Arduino

**ตัวอย่างที่ 2.** ใช้สวิทช์กดติดปล่อยดับต่อวงจรแบบ Pull-down เมื่อกดสวิทช์แล้วหลอด LED สว่าง กำหนดให้ใช้ Digital I/O ขา 6 ของบอร์ด Arduino สำหรับต่อ LED และ Digital I/O ขา 7 ของบอร์ด Arduino สำหรับต่อสวิทช์กดติดปล่อยดับ




รูปที่ 3.9 การต่อวงจรตามตัวอย่างที่ 2.

```
int buttonPin = 7; // กำหนดตำแหน่งสวิทช์

int ledPin =  6; // กำหนดตำแหน่ง

int buttonState = 0;  //กำหนดสภาวะเริ่มต้นของสวิทช์

void setup() {

pinMode(ledPin, OUTPUT);

pinMode(buttonPin, INPUT);  }

void loop() {
```

```
buttonState = digitalRead(buttonPin);

if (buttonState == HIGH) {

digitalWrite(ledPin, HIGH);  // LED สว่าง

} else {

digitalWrite(ledPin, LOW); //  LED ดับ  }

}
```

**การทำงานของโปรแกรม**

โปรแกรมจะสั่งให้ LED สว่างด้วยการเช็คสถานะการกดสวิทช์ที่กำหนดไว้ด้วยคำสั่ง buttonState = digitalRead(buttonPin) โดยการเปรียบเทียบสภาวะการกดสวิทช์จากคำสั่ง

if (buttonState == HIGH) { digitalWrite(ledPin, HIGH)}

else { digitalWrite(ledPin, LOW) }

**3.5 บัซเซอร์ (Buzzer)**

บัซเซอร์ คือลำโพงแบบแม่เหล็กหรือ แบบเปียโซโดยมีวงจรกำเนิดความถี่อยู่ภายในตัว เมื่อป้อนแรงดันให้ก็สามารถกำเนิดเสียงได้ด้วยตัวเอง

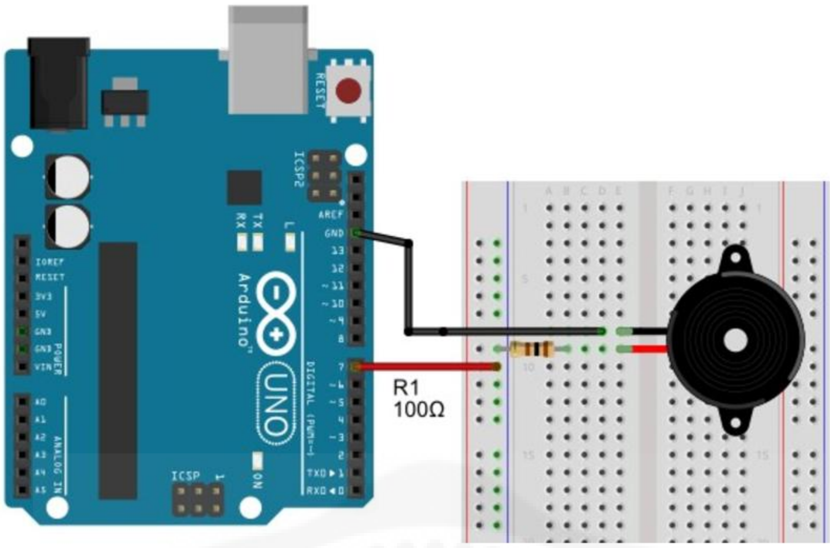
http://www.instructables.com/id/How-to-use-a-Buzzer-Arduino-Tutorial/ : 2561

รูปที่ 3.10 บัซเซอร์

**3.6 การต่อใช้งานสวิทช์กับบอร์ด Arduino**

การต่อใช้งานบัซเซอร์กับบอร์ด Arduino นั้นจะต้องต่อบัซเซอร์ให้ถูกขั้วแรงดันด้วย ถ้าบัซเซอร์เป็นชนิดที่ไม่มีสายไฟต่อมาให้จะมีสัญลักษณ์บอกขั้วแรงดันมาให้ ถ้าแบบที่มีสายไฟต่อมาด้วยสายสีแดงหมายถึงขั้วบวก สีดำหมายถึงขั้วลบ และจะต้องมีตัวต้านทานต่ออนุกรมกับบัซเซอร์ด้วยเพื่อลดแรงดันตกคร่อมบัซเซอร์ โดยทั่วไปจะใช้ตัวต้านทานที่มีค่าประมาณ 100 โอห์ม และบอร์ด Arduino สามารถสั่งให้บัซเซอร์กำเนิดเสียงได้ 2 วิธีคือการใช้คำสั่ง DigitalWrite กับการใช้คำสั่ง tone โดยคำสั่ง tone สามารถกำหนดค่าความถี่ที่จะให้บัซเซอร์ส่งเสียงได้

tone( pin number, frequency in hertz, duration in milliseconds);




รูปที่ 3.11 การต่อบัซเซอร์กับบอร์ด Arduino

**ตัวอย่างที่ 3.** สั่งให้บัซเซอร์กำเนิดเสียงและเงียบเสียง สลับกันไปในระยะเวลาเท่าๆ กัน ด้วยคำสั่ง DigitalWrite และกำหนดให้ขาขั้วบวกของบัซเซอร์ต่อเข้าที่ขา Digital I/O ขา 7 โดยต่อวงจตามรูปที่ 3.11

```
int buzzPin = 7; // กำหนดตำแหน่งบัซเซอร์

void setup() {

pinMode(buzzPin, OUTPUT);  }

void loop() {

digitalWrite(buzzPin, HIGH); // บัซเซอร์ดัง

delay(5);

digitalWrite(buzzPin, LOW); // บัซเซอร์เงียบ
```

```
delay(5);  }
```

**การทำงานของโปรแกรม**

โปรแกรมจะสั่งให้บัซเซอร์ส่งเสียงดังด้วยคำสั่ง digitalWrite(buzzPin, HIGH) และเงียบด้วยสั่ง digitalWrite(buzzPin, LOW)

**ตัวอย่างที่ 4.** สั่งให้บัซเซอร์กำเนิดเสียงที่ความถี่ 1.5kHz ด้วยคำสั่ง Tone และกำหนดให้ขาขั้วบวกของ
บัซเซอร์ต่อเข้าที่ขา Digital I/O ขา 7 โดยต่อวงจตามรูปที่ 3.11

```
int buzzPin = 7; // กำหนดตำแหน่งบัซเซอร์

void setup() {

pinMode(buzzPin, OUTPUT);

 }

void loop() {

tone(buzzPin, 1500, 1000);  // สั่งให้บัซเซอร์มีเสียงดังที่ความถี่ 1500Hz  }
```

**การทำงานของโปรแกรม**

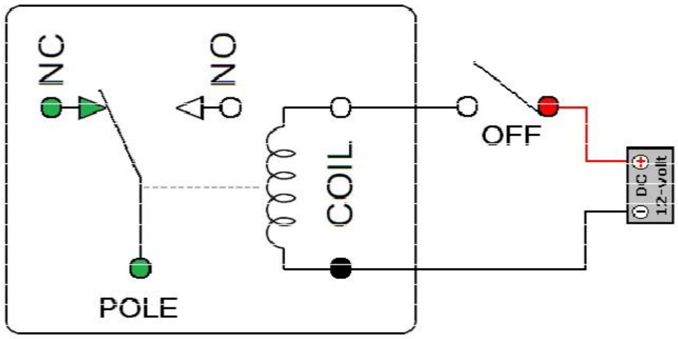
โปรแกรมจะสั่งให้บัซเซอร์ส่งเสียงดังด้วยคำสั่ง tone(buzzPin, 1500, 1000)

### 3.7 รีเลย์ (Relay)

เนื่องจากบอร์ด Arduino ไม่สามารถจ่ายกระแสไฟฟ้าที่มีค่ากระแสสูงๆ ได้ ซึ่งในกรณีถ้านำบอร์ด Arduino ไปใช้ควบคุมอุปกรณ์ดังกล่าวจะต้องการมีการต่ออุปกรณ์ช่วยเพื่อให้สามารถจ่ายกระแสไฟฟ้าที่มีค่าสูงๆ ได้อีกทีหนึ่ง ซึ่งส่วนใหญ่จะนิยมใช้รีเลย์เป็นอุปกรณ์ช่วยดังกล่าว โดยรีเลย์ทำหน้าที่เป็นสวิตซ์ตัด-ต่อวงจร โดยใช้หลักการแม่เหล็กไฟฟ้า และการที่จะให้รีเลย์ทำงานก็ต้องจ่ายแรงดันให้ตามขนาดที่กำหนด โดยเมื่อยังไม่จ่ายแรงดันกับรีเลย์หน้าสัมผัสรีเลย์จะอยู่ในสภาวะหน้าสัมผัสปิด (Normal Close : NC) ดังรูปที่ 13. และเมื่อจ่ายแรงดันให้กับรีเลย์ จะทำให้หน้าสัมผัสเปลี่ยนสภาวะมาเป็นหน้าสัมผัสเปิด (Normal Open : NO) ดังรูปที่ 14. และทันทีที่หยุดจ่ายแรงดันให้รีเลย์ก็จะกลับมาเป็นสภาวะหน้าสัมผัสปิดเหมือนเดิม ในการนำรีเลย์ไปใช้งาน จะสั่งงานให้รีเลย์ทำงานในสภาวะหน้าสัมผัสเปิด

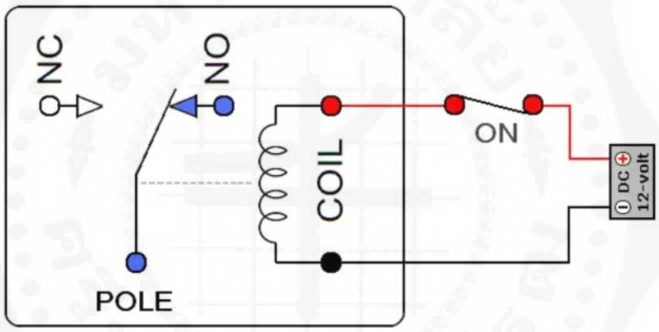
http://electronics.howstuffworks.com/relay.htm : 2561

รูปที่ 3.12 โครงสร้างภายในรีเลย์




http://www.zen22142.zen.co.uk/ronj/cpr.html : 2561

รูปที่ 3.13 สภาวะหน้าสัมผัสปิด (Normal Close : NC)

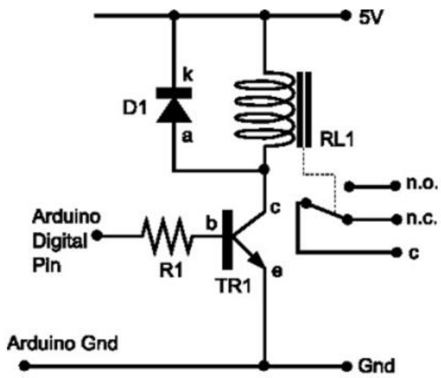



http://www.zen22142.zen.co.uk/ronj/cpr.html : 2561

รูปที่ 3.14 สภาวะหน้าสัมผัสเปิด (Normal Open : NO)

### 3.8 การต่อใช้งานรีเลย์กับบอร์ด Arduino

การนำรีเลย์ไปใช้งานกับบอร์ด Arduino ไม่สามารถนำไปต่อใช้งานได้ทันทีเนื่องจากบอร์ด Arduino ไม่สามารถจ่ายกระแสไฟได้พอที่จะสั่งให้รีเลย์ทำงาน ดังนั้นจึงต้องมีการต่อวงจรเพิ่มเติมอีกที่เพื่อให้บอร์ด Arduino สามารถใช้งานร่วมกับรีเลย์ได้ดังรูปที่ 3.15 และในปัจจุบันได้มีการจัดทำบอร์ดรีเลย์สำเร็จรูปเพื่อใช้กับบอร์ด Arduino ได้เลยดังรูปที่ 3.16



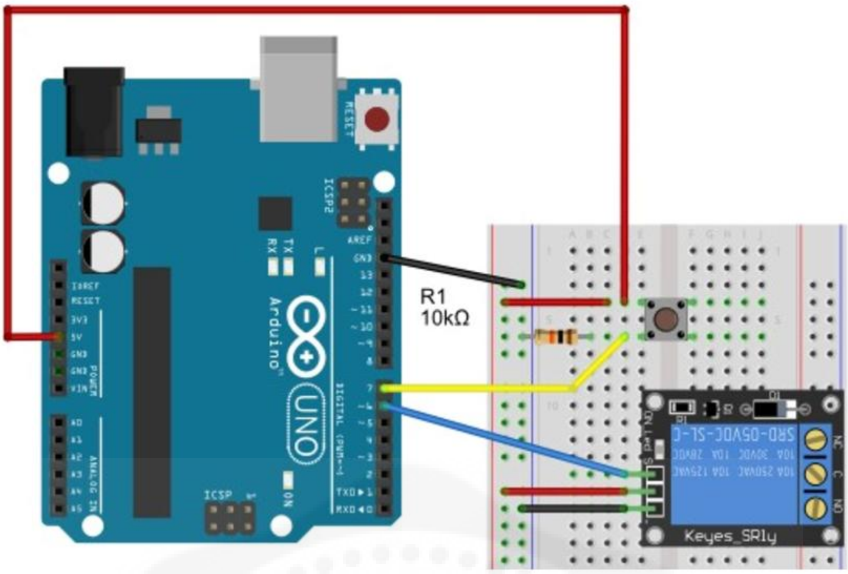

https://www.flickr.com/photos/anachrocomputer/3242387876 : 2561

รูปที่ 3.15 การต่อรีเลย์เพื่อใช้งานร่วมกับบอร์ด Arduino


https://www.robotics.org.za/index.php?route=product/product&product_id=160
8 : 2561

รูปที่ 3.16 บอร์ดรีเลย์สำเร็จรูปเพื่อใช้งานร่วมกับบอร์ด Arduino

**ตัวอย่างที่ 5.** ใช้สวิทช์กดติดปล่อยดับต่อวงจรแบบ Pull-down เมื่อกดสวิทช์แล้วรีเลย์ทำงาน และเมื่อกดสวิทช์อีกครั้ง รีเลย์หยุดทำงาน กำหนดให้ใช้ Digital I/O ขา 6 ของบอร์ด Arduino สำหรับต่อวงจรขับรีเลย์ และ Digital I/O ขา 7 ของบอร์ด Arduino สำหรับต่อสวิทช์กดติดปล่อยดับ




รูปที่ 3.17 การต่อวงจรตามตัวอย่างที่ 5.

```
int buttonPin = 7; // กำหนดตำแหน่งสวิทช์กดติดปล่อยดับ

int relayPin = 6; // กำหนดตำแหน่งควบคุมรีเลย์

int state = LOW;   // กำหนดสภาวะเริ่มต้นในการควบคุมรีเลย์

int stateButton ;   // ตั้งชื่อที่เก็บสภาวะการกดปุ่ม

int previous = LOW;    //  กำหนดสภาวะก่อนกดปุ่ม

long time = 0;         // กำหนดค่าเวลาสั่งการล่าสุด

long debounce = 200;   // กำหนดค่าเวลาเพื่อป้องกันสภาวะ bounce ของสวิทช์

void setup()  {

pinMode(buttonPin, INPUT);
```

```
pinMode(relayPin, OUTPUT);

}

void loop()  {

stateButton = digitalRead(buttonPin);

if (stateButton == HIGH && previous == LOW && millis() - time > debounce) {

if (state == HIGH){

state = LOW;

} else {

state = HIGH; }

time = millis();

}

digitalWrite(relayPin, state);

previous = stateButton;

}
```

**การทำงานของโปรแกรม**

โปรแกรมจะสั่งรีเลย์ทำงานจากการกดสวิทช์ โดยกดครั้งแรกรีเลย์ทำงาน กดครั้งต่อไปรีเลย์หยุดทำงาน โดยการใช้คำสั่งเปรียบเทียบเก็บค่าสภาวะการกดสวิทช์ครั้งก่อนกับครั้งล่าสุดด้วยชุดคำสั่ง

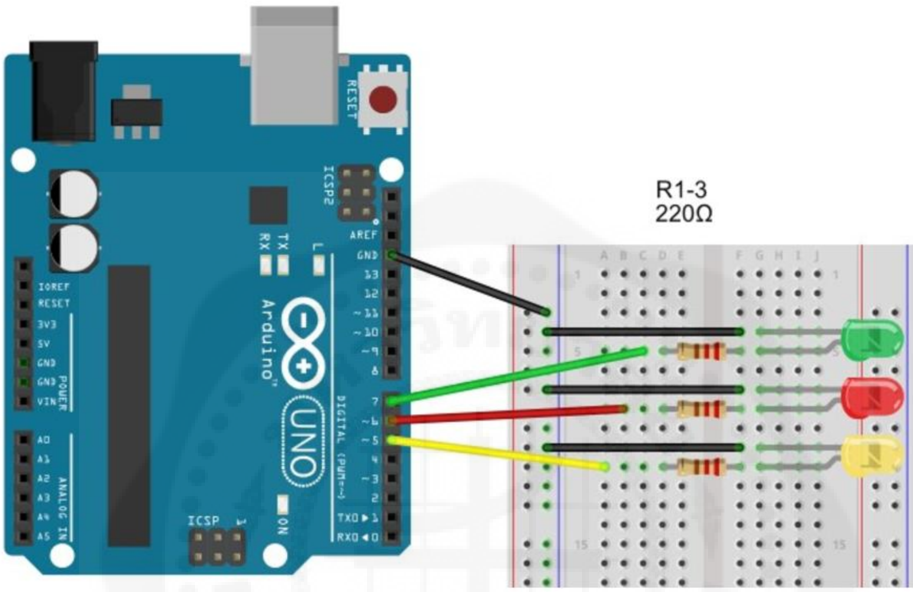
```
if (stateButton == HIGH && previous == LOW && millis() - time > debounce) {

if (state == HIGH){

state = LOW;

} else {

state = HIGH; }

time = millis();

}

digitalWrite(relayPin, state);

previous = stateButton;

}
```

**สรุป**

การต่ออุปกรณ์เพื่อให้บอร์ด Arduino รับ/ส่งค่าทางขา DIgital I/O นั้น โดยการรับค่าเข้ามา (ต่อกับอุปกรณ์ชนิดอินพุท) และการต่ออุปกรณ์เพื่อให้บอร์ด Arduino ส่งค่าออกมา (ต่อกับอุปกรณ์ชนิดเอาท์พุท) ประการแรกจะต้องกำหนดให้บอร์ด Arduino รับทราบว่าขาที่จะต่อใช้งานเป็นขารับข้อมูลด้วยคำสั่ง pinMode(หมายเลขขา, INPUT) หรือขาส่งข้อมูลด้วยคำสั่ง pinMode(xx, OUTPUT) ก่อนจึงจะใช้คำสั่ง digitalWrite(หมายเลขขา, สถานะ) (ส่งสถานะออก) หรือ digitalRead(หมายเลขขา, สถานะ) (อ่านสถานะเข้า)

**คำถาม**

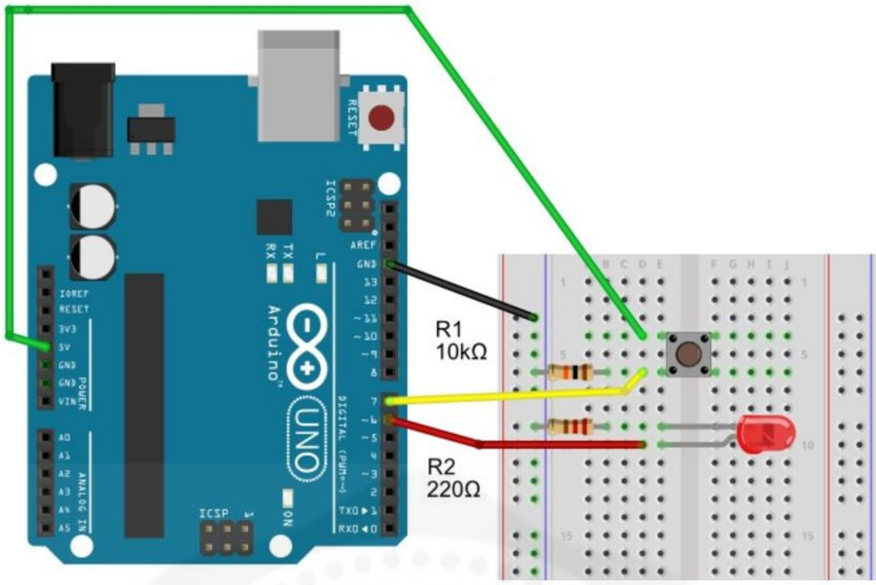
1. ต่อ LED 3 ดวง ให้ติดสว่างทีละดวงเรียงกันไปจนครบ แล้วย้อนวนกลับไป-มา กำหนดให้ใช้ Digital I/O ขา 5, 6 และ 7 ของบอร์ด Arduino สำหรับต่อ LED โดยต่อวงจรตามรูป




รูปที่ 3.18 การต่อวงจรตามคำถามข้อ 1.

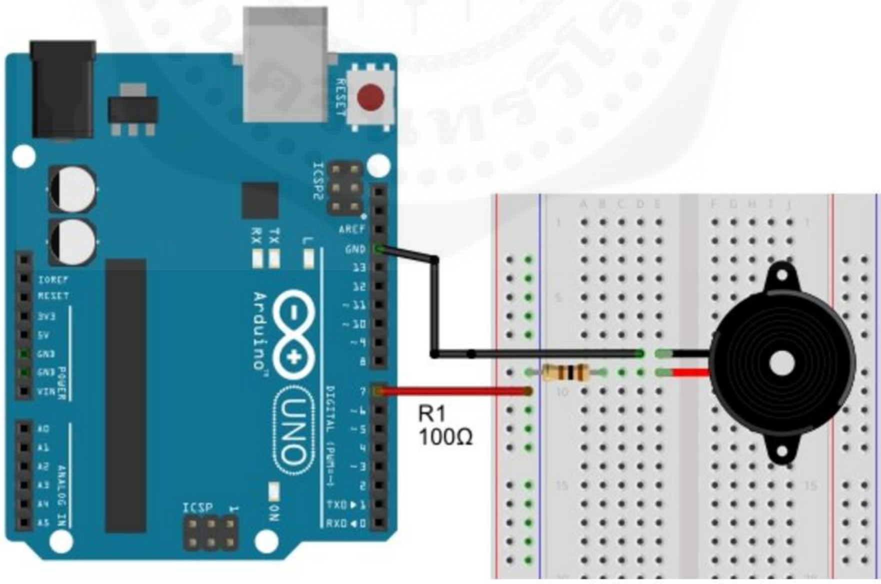
2. . ใช้สวิทช์กดติดปล่อยดับต่อวงจรแบบ Pull-down เมื่อกดสวิทช์แล้วหลอด LED ดับ กำหนดให้ใช้ Digital I/O ขา 6 ของบอร์ด Arduino สำหรับต่อ LED และ Digital I/O ขา 7 ของบอร์ด Arduino สำหรับต่อสวิทช์

โดยต่อวงจรตามรูป



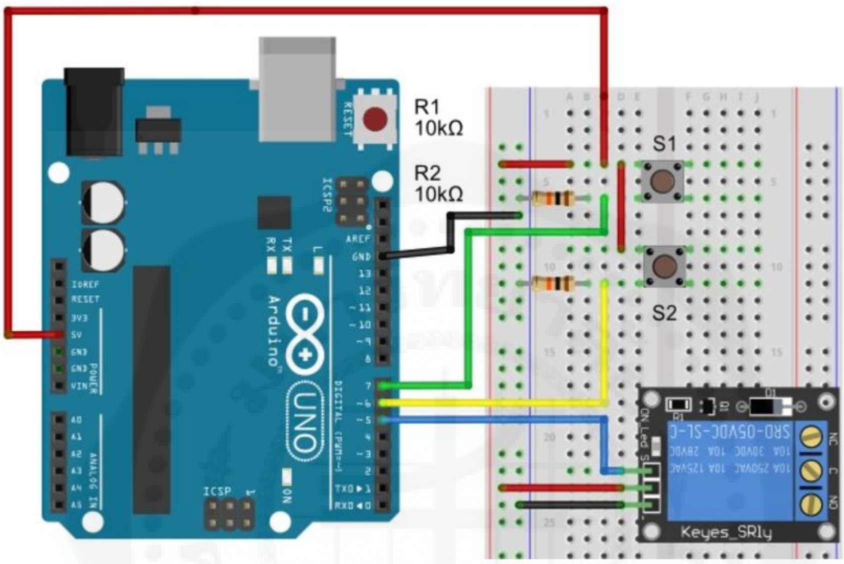

รูปที่ 3.19 การต่อวงจรตามคำถามข้อ 2.

3. สั่งให้บัซเซอร์กำเนิดเสียงที่ความถี่ 1.5kHz สลับกับความถี่ 500Hz ด้วยคำสั่ง Tone และกำหนดให้ขาขั้วบวกของบัซเซอร์ต่อเข้าที่ขา Digital I/O ขา 7 โดยต่อวงจรตามรูป




รูปที่ 3.20 การต่อวงจรตามคำถามข้อ 3.

4. ใช้สวิทช์กดติดปล่อยดับต่อวงจรแบบ Pull-down 2 ตัว ตัวที่ 1 สำหรับกดให้รีเลย์ทำงาน และตัวที่ 2 สำหรับกดให้รีเลย์หยุดทำงาน กำหนดให้ใช้ Digital I/O ขา 5 ของบอร์ด Arduino สำหรับต่อวงจรขับรีเลย์ และ Digital I/O ขา 6 และ 7 ของบอร์ด Arduino สำหรับต่อสวิทช์กดติดปล่อยดับ โดยต่อวงจรตามรูป




รูปที่ 21. การต่อวงจรของคำถามข้อ 4.

**เอกสารอ้างอิง**

1. Mark Svaljek. (2015). Arduino Succinctly. Morrisville, NC : Syncfusion Inc.

2. https://www.arduino.cc/en/Tutorial/Button : 2561

3. https://www.arduino.cc/en/Reference/Tone : 2561

4. https://th.wikipedia.org/wiki/รีเลย์ : 2561

# บทที่ 4

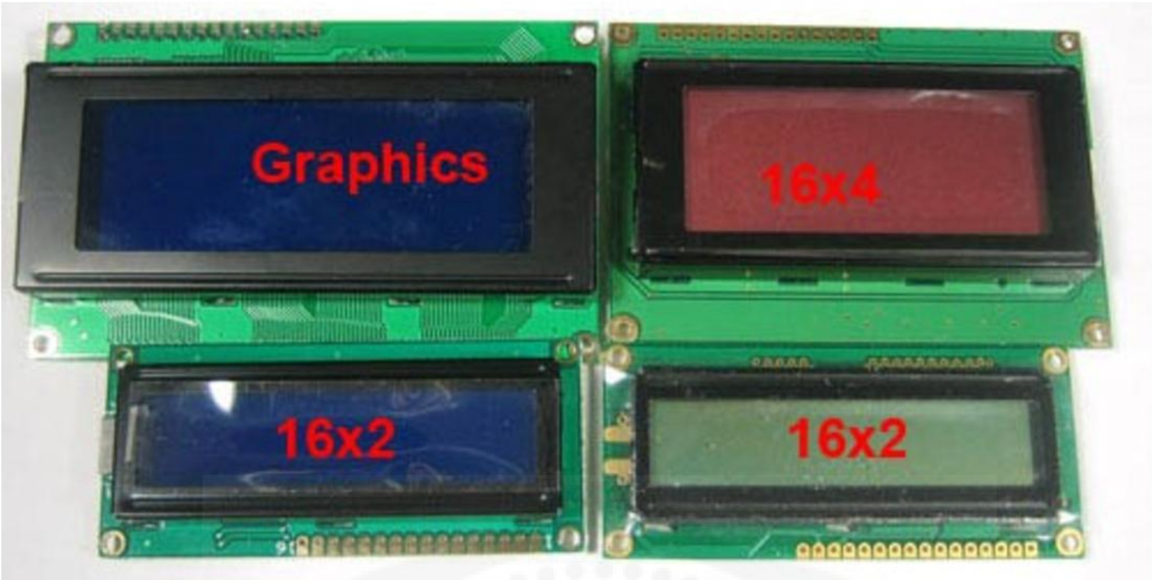
# การต่อใช้งานจอแสดงผลผลึกเหลว LCD

## 4.1 จอแสดงผลผลึกเหลว LCD

จอแสดงผลผลึกเหลว LCD เป็นชื่อย่อมาจากคำว่า Liquid Crystal Display เป็นจอที่ทำมาจากผลึกคริสตอลเหลว โดยด้านหลังจอจะมีไฟส่องสว่าง ซึ่งเรียกว่าไฟ Backlight เมื่อมีการจ่ายกระแสไฟฟ้าเข้าไปกระตุ้นที่จอผลึกคริสตอลเหลว ก็จะทำให้ผลึกโปร่งแสง ทำให้แสงที่มาจากไฟ Backlight แสดงขึ้นมาบนหน้าจอ โดยส่วนอื่นที่โดนผลึกปิดบังไว้จะมีสีที่แตกต่างกันตามสีของผลึกคริสตอล เช่น สีเขียว หรือ สีฟ้า ทำให้เมื่อมองไปที่จอก็จะพบกับตัวอักษรสีขาว ส่วนพื้นหลังจะเป็นสีตามสีของผลึกคริสตอล จอ LCD จะแบ่งเป็น 2 แบบใหญ่ๆตามลักษณะการแสดงผลดังนี้

4.1.1 **Character LCD** เป็นจอที่แสดงผลเป็นตัวอักษรตามช่องแบบตายตัว เช่น จอ LCD ขนาด 16x2 หมายถึงใน 1 บรรทัดมีตัวอักษร 16 ตัว และมีทั้งหมด 2 บรรทัด

4.1.2 **Graphic LCD** เป็นจอที่กำหนดได้ว่าจะให้แต่ละจุดบนหน้าจอกั้นแสง หรือปล่อยแสงผ่านออกไป ทำให้จอนี้สามารถสร้างรูปภาพขึ้นมาบนหน้าจอได้ การระบุขนาดจอ จะระบุเป็นจำนวนจุด (Pixels) ในแต่ละแนว เช่น 128x64 หมายถึงจอที่มีจำนวนจุดตามแนวนอน 128 จุด และมีจุดตามแนวตั้ง 64 จุด




http://www.circuitvalley.com/2011/12/two-wire-serial-lcd-16x2-graphics.html :

2561

รูปที่ 4.1 จอแสดงผลผลึกเหลว LCD

### 4.2 จอ Character LCD


http://circuits4you.com/2016/05/15/arduino-lcd-display/ :2561

รูปที่ 4.2 จอ Character LCD

ขาของ LCD จะมี 16 ขา โดยแต่ละขามีรายละเอียดดังนี้

–GND : Ground

-VCC : ไฟเลี้ยง LCD ขนาด +5VDC

-CONTRAST : ขาสำหรับปรับความสว่างของหน้าจอ LCD

-RS : Register select ใช้บอกให้ LCD Controller ว่า ข้อมูลที่ส่งให้ขา Data เป็นคำสั่งหรือข้อมูล

-RW : Read/Write ใช้กำหนดว่าจะอ่านหรือเขียนข้อมูลกับ LCD Controller

-EN : Enable ใช้กำหนดการทำงานให้กับ LCD Controller

-DB0-DB7 :  เป็นขาสัญญาณ Data ใช้เขียนหรืออ่านข้อมูล/คำสั่ง กับ LCD Controller

-BLACKLIGHT + : ไฟเลี้ยงหลอด BLACKLIGHT ขั้ว +

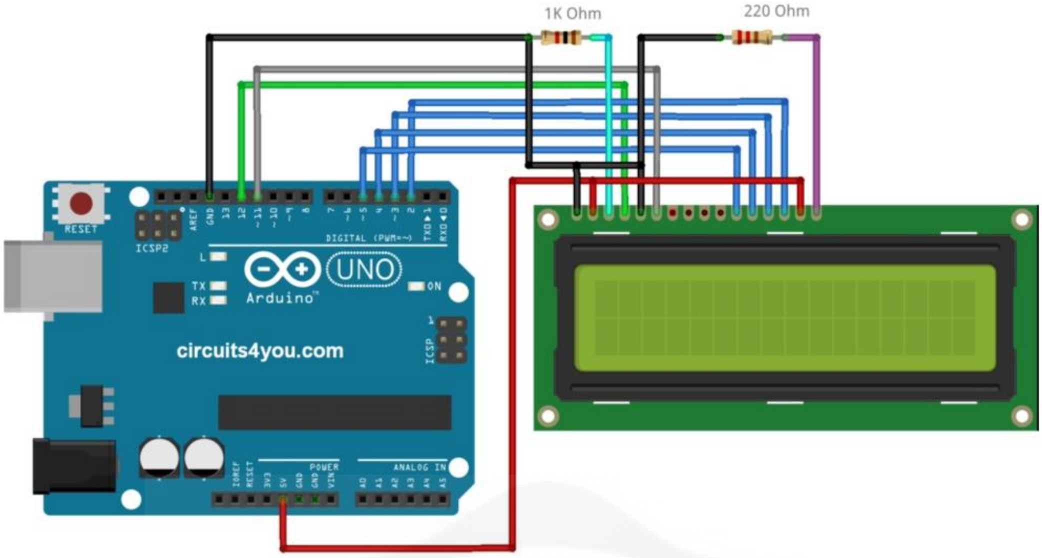
-BLACKLIGHT - : ไฟเลี้ยงหลอด BLACKLIGHT ขั้ว -

## 4.3 การเชื่อมต่อกับจอ Character LCD

การเชื่อมต่อจะมีด้วยกัน 2 แบบ คือ

### 4.3.1 การเชื่อมต่อแบบขนาน

เป็นการเชื่อมต่อจอ LCD เข้ากับบอร์ด Arduino โดยตรง โดยจะแบ่งเป็นการเชื่อมต่อได้ 2 แบบคือ แบบ 4 บิต และแบบ 8 บิต ซึ่งการเชื่อมต่อกับบอร์ด Arduino จะนิยมเชื่อมต่อแบบ 4 บิต เนื่องจากใช้สายในการเชื่อมต่อน้อยกว่า

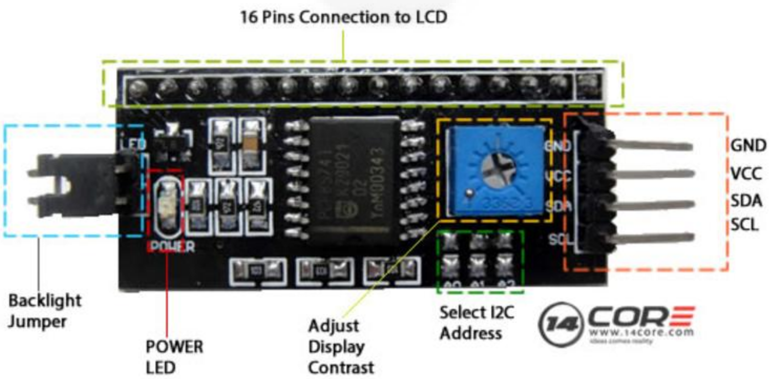



http://circuits4you.com/2016/05/15/arduino-lcd-display/ : 2561

รูปที่ 4.3 การเชื่อมต่อกับบอร์ด Arduino แบบขนาน เชื่อมต่อแบบ 4 บิต

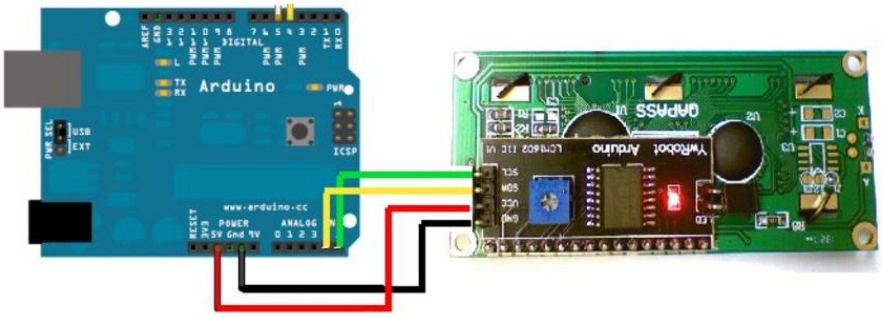
### 4.3.2 การเชื่อมต่อแบบอนุกรม

เป็นการเชื่อต่อกับจอ LCD เข้ากับบอร์ด Arduino โดยผ่านโมดูลที่ทำหน้าแปลงรูปแบบการเชื่อมต่อกับจอ LCD จากแบบขนาน มาเป็นการเชื่อมต่อแบบอนุกรม ซึ่งจะใช้สายเชื่อมต่อเพียง 4 เส้น โดยใช้โมดูล I2C Serial Interface ส่งข้อมูลด้วยโปรโตคอล I2C ก็ทำให้หน้าจอแสดงผลข้อความต่างๆ เหมือนการต่อ LCD แบบ ขนาน




http://www.14core.com/wiring-i2c-module-on-16x2-lcd-with-sclsda/ : 2561

รูปที่ 4.4 โมดูล I2C Serial Interface

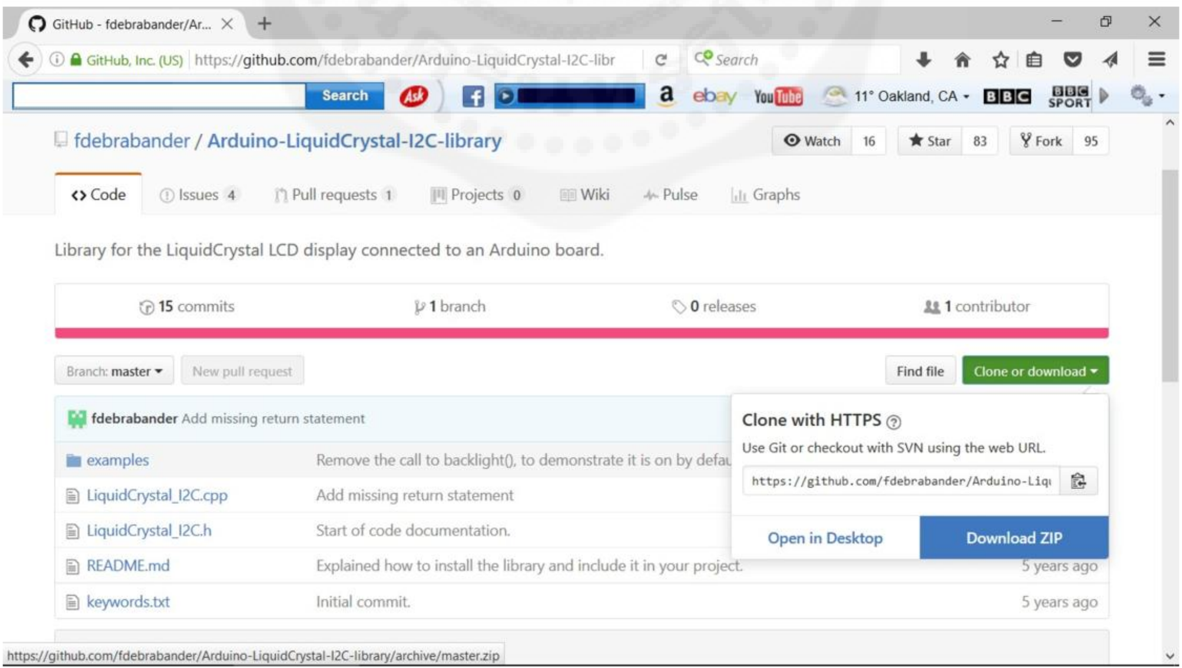



http://commandronestore.com/products/ba200.php : 2561

รูปที่ 4.5 การต่อโมดูล I2C Serial Interface กับบอร์ด Arduino และจอ LCD

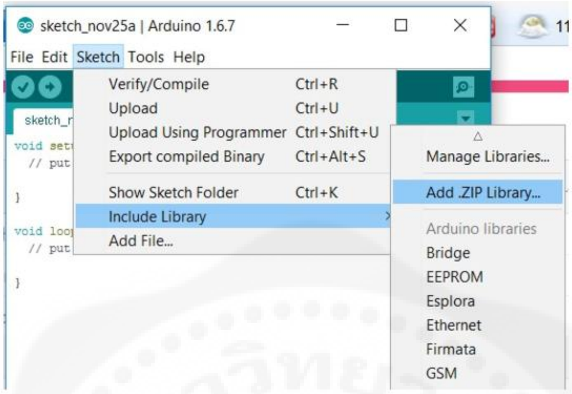
## 4.4 เริ่มใช้งาน Arduino IDE กับ I2C Serial Interface

ก่อนอื่นต้องติดตั้งไลบรารี่เสริมเพื่อให้จอ LCD ที่ต่อร่วมกับโมดูล I2C Serial Interface สามารถใช้งานร่วมกับบอร์ด Arduinoได้ โดยดาวน์โหลดไลบรารี่เสริมได้ที่ https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library ดังรูปที่ 4.6 โดยดาวน์เป็นไฟล์ Zip



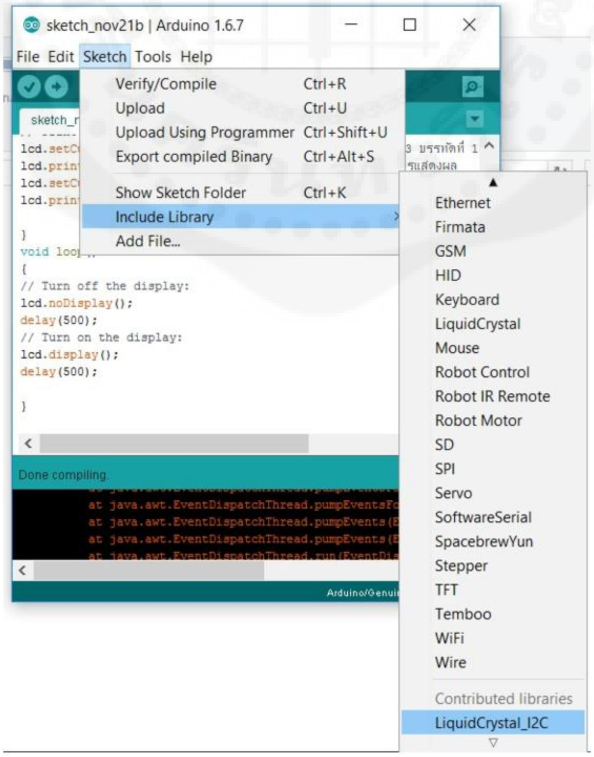

รูปที่ 4.6 เวปไซด์สำหรับดาวน์โหลดไลบรารี่เสริม

หลังจากนั้นให้ดำเนินการติดตั้งให้กับโปรแกรม Arduino IDE โดยกดไปที่ Sketch -> Include Library -> Add .Zip Library ดังรูปที่ 4.7 จากนั้นทำการหาตำแหน่งไฟล์ที่ดาวน์โหลดเก็บไว้ จากนั้นกด Open




รูปที่ 4.7 การติดตั้งไลบรารี่เสริม

เมื่อติดตั้งไลบรารี่เสริมสำเร็จ จะมีชื่อไลบรารี่เสริมในโปรแกรม Arduino IDE ดังรูปที่ 8.




รูปที่ 4.8 การติดตั้งไลบรารี่เสริมสำเร็จ

ในการเชื่อมต่ออุปกรณ์แบบ I2C ในการส่งข้อมูลไปให้อุปกรณ์ต่างๆ จำเป็นต้องทราบค่าแอดเดรสของอุปกรณ์ที่ต้องการจะสื่อสารกันก่อน ถึงจะส่งข้อมูลสื่อสารกันได้ โดยเราสามารถให้ค้นหาแอดเดรสของอุปกรณ์ที่ต้องการโปรแกรมข้างล่างนี้

```
#include <Wire.h>
void setup() {
Serial.begin (9600);
Serial.println ();
Serial.println ("I2C Addr. Scanning ...");
byte count = 0;
Wire.begin();
for (byte i = 8; i < 120; i++)  {
Wire.beginTransmission (i);
if (Wire.endTransmission () == 0)  {
Serial.print ("Found address: ");
Serial.print (i, DEC);
Serial.print (" (0x");
Serial.print (i, HEX);
Serial.println (")");
count++;
}  }
Serial.println ("Done.");
Serial.print ("Found ");
Serial.print (count, DEC);
Serial.println (" device(s).");
}
void loop() { }
```

ก่อนเขียนโปรแกรมแสดงผลที่จอ LCD มารู้จักฟังก์ชั่นสั่งงานจอ LCD เพื่อเราให้สามารถแสดงผลได้ตามที่เราต้องการ

ฟังก์ชั่น lcd.clear() > ใช้ล้างหน้าจอ เมื่อมีตัวอักษรใดๆอยู่บนหน้าจอ จะถูกล้างออกทั้งหมด

lcd.home() > ใช้ปรับให้เคเซอร์กลับไปอยู่ที่ตำแหน่งแรกด้านซ้าย เมื่อใช้คำสั่ง lcd.print() จะไปเริ่มแสดงผลทางด้านบนซ้าย

lcd.setCursor (ลำดับตัวอักษรนับจากทางซ้าย, บรรทัด) > ใช้ตั้งค่าเคเซอร์
เช่น lcd.setCursor(2, 0); หมายถึงเซ็ตเคเซอร์ไปตัวอักษรที่ 2 นับจากทางซ้าย และอยู่บรรทัดแรก เมื่อใช้คำสั่ง lcd.print() ตัวอักษรตัวแรกจะอยู่ลำดับที่ 3 นับจากทางซ้าย

lcd.write(ข้อมูลที่ต้องการเขียนออกไป) > ใช้สำหรับเขียนข้อมูลออกไปทีละตัวอักษร

lcd.print(ข้อมูลที่ต้องการให้เขียนออกไป [, รูปแบบข้อมูล]) > ใช้เขียนข้อมูลออกไปทั้งข้อความ

lcd.cursor() > ใช้สั่งให้แสดงเคเซอร์บนหน้าจอ

lcd.noCursor() > ใช้สั่งให้ไม่แสดงเคเซอร์บนหน้าจอ

lcd.display() > แสดงตัวอักษรบนหน้าจอ

lcd.noDisplay() > ปิดการแสดงตัวอักษรในหน้าจอ

lcd.scrollDisplayLeft() > เลือนตัวอักษรไปทางซ้าย 1 ตัว

lcd.scrollDisplayRight() > เลื่อนตัวอักษรไปทางขวา 1 ตัว

lcd.autoscroll() > เลื่อนตัวอักษรไปทางขวาอัตโนมัติหากใช้คำสั่ง lcd.print() หรือ lcd.write() เมื่อตัวอักษรเต็มหน้าจอ

lcd.noAutoscroll() > ปิดการเลื่อนตัวอักษรอัตโนมัติ

lcd.leftToRight() > เมื่อใช้คำสั่ง lcd.print() หรือ lcd.write() ตัวอักษรจะเขียนจากซ้ายไปขวา

lcd.rightToLeft() > เมื่อใช้คำสั่ง lcd.print() หรือ lcd.write() ตัวอักษรจะเขียนจากขวาไปซ้าย

**ตัวอย่างที่ 1. แสดงผล Character LCD 16×2**

```
#include <LiquidCrystal_I2C.h> //ประกาศ Library ของจอ I2C

// กำหนดค่า LCD แอดเดรสที่ตำแหน่ง 0x27 สำหรับ LCD แบบ 16 ตัวอักษร มี 2 บรรทัด

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()  {

// กำหนดค่า LCD

lcd.begin();  // การแสดงผลที่ LCD

lcd.setCursor(3,0);  //ฟังก์ชั่นในการกำหนดตำแหน่ง Cursor ที่ 3 บรรทัดที่ 1

lcd.print("--Hello--");  //ฟังก์ชั่นในการกำหนดข้อความที่ต้องการแสดงผล

lcd.setCursor(1,1);  //ฟังก์ชั่นในการกำหนดตำแหน่ง Cursor ที่ 1 บรรทัดที่ 2

lcd.print("<<<Thailand>>>"); ////ฟังก์ชั่นในการกำหนดข้อความที่ต้องการแสดงผล

}
```

void loop() { }

**ตัวอย่างที่ 2.** แสดงผลให้ Cursor กระพริบโดยใช้คำสั่ง Blink() และยกเลิกการกระพริบโดยใช้คำสั่ง noBlink()

```
#include <LiquidCrystal_I2C.h> //ประกาศ Library ของจอ I2C

// กำหนดค่า LCD แอดเดรสที่ตำแหน่ง 0x27 สำหรับ LCD แบบ 16 ตัวอักษร มี 2 บรรทัด

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {

// กำหนดค่า LCD

lcd.begin();  // การแสดงผลที่ LCD

lcd.print("Hello, Thailand");  //ฟังก์ชั่นในการกำหนดข้อความที่ต้องการแสดงผล

}

void loop()  {

// ปิดการกระพริบ Cursor

lcd.noBlink();

delay(500);

// เปิดการกระพริบ Cursor

lcd.blink();
```

```
delay(500);
}
```

**ตัวอย่างที่ 3.** แสดงผลให้ Cursor กระพริบแบบ underscore (_) โดยใช้คำสั่ง cursor() และยกเลิกการกระพริบโดยใช้คำสั่ง noCursor()

```
#include <LiquidCrystal_I2C.h> //ประกาศ Library ของจอ I2C
// กำหนดค่า LCD แอดเดรสที่ตำแหน่ง 0x27 สำหรับ LCD แบบ 16 ตัวอักษร มี 2 บรรทัด
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
// กำหนดค่า LCD
lcd.begin();  // การแสดงผลที่ LCD
lcd.print("Hello, Arduino"); //ฟังก์ชั่นในการกำหนดข้อความที่ต้องการแสดงผล
}
void loop()  {
// ปิดการกระพริบ Cursor
lcd.nocursor();
delay(500);
// เปิดการกระพริบ Cursor
```

```
Lcd.cursor();

delay(500);

}
```

**ตัวอย่างที่ 4.** แสดงผลให้ตัวอักษรกระพริบโดยใช้คำสั่ง display() และยกเลิกการกระพริบโดยใช้คำสั่ง noDisplay()

```
#include <LiquidCrystal_I2C.h> //ประกาศ Library ของจอ I2C

// กำหนดค่า LCD แอดเดรสที่ตำแหน่ง 0x27 สำหรับ LCD แบบ 16 ตัวอักษร มี 2 บรรทัด

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {

// กำหนดค่า LCD

lcd.begin();  // การแสดงผลที่ LCD

lcd.setCursor(3,0);  //ฟังก์ชั่นในการกำหนดตำแหน่ง Cursor ที่ 3 บรรทัดที่ 1

lcd.print("--Hello--");  //ฟังก์ชั่นในการกำหนดข้อความที่ต้องการแสดงผล

lcd.setCursor(1,1);  //ฟังก์ชั่นในการกำหนดตำแหน่ง Cursor ที่ 1 บรรทัดที่ 2

lcd.print("<<<Thailand>>>");  //ฟังก์ชั่นในการกำหนดข้อความที่ต้องการแสดงผล

}
```

```
void loop() {
// ปิดการแสดงผล
lcd.noDisplay();
delay(500);
// เปิดการแสดงผล
lcd.display();
delay(500);
}
```

**ตัวอย่างที่ 5.** กำหนดทิศทางการเขียนข้อความโดยใช้คำสั่ง leftToRight() และ rightToLeft() เพื่อตัวอักษรให้เคลื่อนที่ไปทางซ้าย หรือทางขวา

```
#include <LiquidCrystal_I2C.h> //ประกาศ Library ของจอ I2C
LiquidCrystal_I2C lcd(0x27, 16, 2);
// กำหนดค่า LCD แอดเดรสที่ตำแหน่ง 0x27 สำหรับ LCD แบบ 16 ตัวอักษร มี 2 บรรทัด
int Char = 'A';
void setup()  {
// กำหนดค่า LCD
lcd.begin();
```

```
// เปิดการแสดง Cursor
lcd.cursor();   }
void loop()  {
// กลับทิศทางการเคลื่อนที่ ที่อักษร M
if (Char == 'M')  {
// เคลื่อนตัวอักษรตัวถัดไป ไปทางขวา
lcd.rightToLeft();  }
// กลับทิศทางการเคลื่อนที่อีกครั้ง ที่อักษร S
if (Char == 'S')  {
// เคลื่อนตัวอักษรตัวถัดไป ไปทางซ้าย
lcd.leftToRight();
}
// รีเซ็ตการเคลื่อนที่ ที่ตัวอักษร Z
if (Char > 'Z') {
// ไปที่ตำแหน่ง (0,0)
lcd.clear();
lcd.home();
```

```
// เริ่มต้นใหม่ที่ตำแหน่ง 0

Char = 'A';

}

// แสดงผลตัวอักษร

lcd.write(Char);

delay(1000);

// เปลี่ยนตัวอักษรเป็นตัวถัดไป

char++;

}
```

**ตัวอย่างที่ 6.** กำหนดทิศทางการเคลื่อนที่ของข้อความโดยใช้คำสั่ง scrollDisplayLeft() และ scrollDisplayRight() เพื่อให้ข้อความเคลื่อนที่ไปทางซ้าย หรือทางขวา

```
#include <LiquidCrystal_I2C.h> //ประกาศ Library ของจอ I2C

LiquidCrystal_I2C lcd(0x27, 16, 2);

// กำหนดค่า LCD แอดเดรสที่ตำแหน่ง 0x27 สำหรับ LCD แบบ 16 ตัวอักษร มี 2 บรรทัด

void setup() {

lcd.begin();

lcd.print("Hello Thailand");
```

```
delay(500);
}
void loop() {
// เคลื่อนที่ไปทางซ้าย 14 ครั้ง
for (int positionCounter = 0; positionCounter < 14; positionCounter++)  {
// เคลื่อนที่ไปทางซ้าย 1 ตำแหน่ง
lcd.scrollDisplayLeft();
delay(400);
}
// เคลื่อนที่ไปทางขวา 30 ครั้ง
 for (int positionCounter = 0; positionCounter < 30; positionCounter++)  {
// เคลื่อนที่ไปทางขวา 1 ตำแหน่ง
lcd.scrollDisplayRight();
delay(400);
}
// เคลื่อนที่ไปทางซ้าย 15 ครั้ง
```

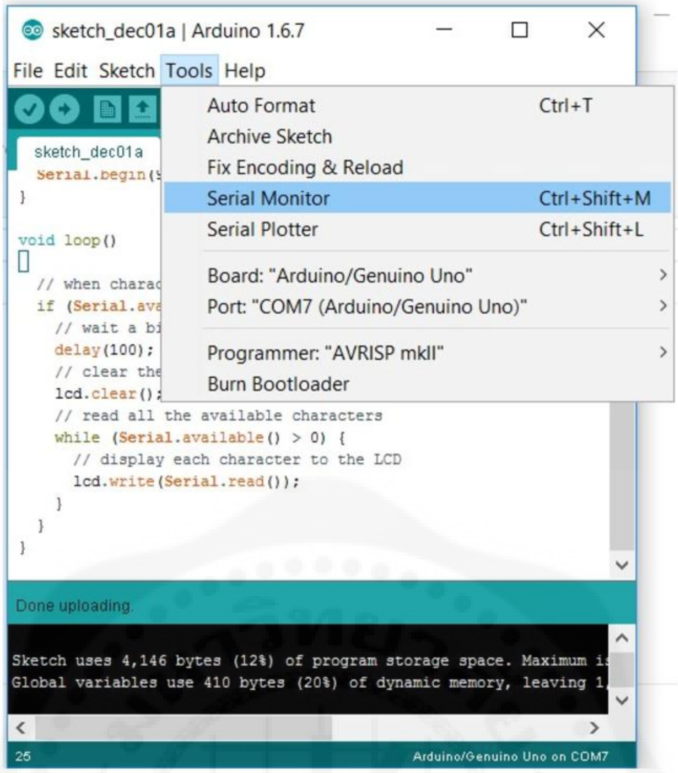
```
for (int positionCounter = 0; positionCounter < 15; positionCounter++) {
// เคลื่อนที่ไปทางซ้าย 1 ตำแหน่ง
lcd.scrollDisplayLeft();
delay(400);
}
delay(2000);
lcd.clear();
lcd.print("Hello Thailand");
delay(1000);
}
```

**ตัวอย่างที่ 7.** รับข้อความจาก Serial แล้วนำมาแสดงผลบน LCD

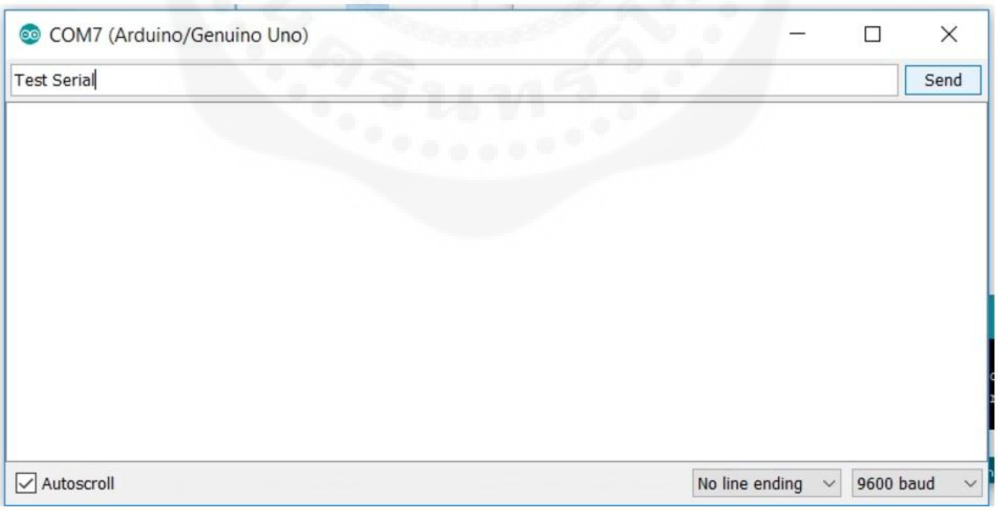
```
#include <LiquidCrystal_I2C.h> //ประกาศ Library ของจอ I2C
// กำหนดค่า LCD แอดเดรสที่ตำแหน่ง 0x27 สำหรับ LCD แบบ 16 ตัวอักษร มี 2 บรรทัด
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
```

```
// กำหนดตำแหน่ง column และ row ของ LCD
lcd.begin();
// กำหนดค่าการสื่อสารแบบอนุกรม
Serial.begin(9600);
}
void loop() {
// เมื่อตัวอักษรรับค่ามาจากพอร์ตอนุกรม
if (Serial.available()) {
delay(100);
lcd.clear();
// อ่านค่าตัวอักษรที่รับเข้ามา
while (Serial.available() > 0) {
// แสดงผลตัวอักษรที่รับเข้ามาที่ LCD
lcd.write(Serial.read());   }   }   }
```

เมื่ออัพโหลดโปรแกรมลงบนบอร์ด Arduino แล้ว ที่โปรแกรม Arduino IDE ให้เปิดหน้าต่าง Serial Monitor แล้วพิมพ์ข้อความลงในช่อง แล้วกดปุ่ม Send ข้อความที่พิมพ์จะไปปรากฏที่จอ LCD




รูปที่ 4.9 การเปิดหน้าต่าง Serial Monitor




รูปที่ 4.10 การส่งค่าจาก Serial Monitor แสดงผลที่จอ LCD

**ตัวอย่างที่ 8.** การกำหนดตำแหน่งของ Cursor ด้วยคำสั่ง setCursor()

```
#include <LiquidCrystal_I2C.h> //ประกาศ Library ของจอ I2C

// กำหนดค่า LCD แอดเดรสที่ตำแหน่ง 0x27 สำหรับ LCD แบบ 16 ตัวอักษร มี 2 บรรทัด

LiquidCrystal_I2C lcd(0x27, 16, 2); display

const int numRows = 2;

const int numCols = 16;

void setup()

{

// กำหนดตำแหน่ง column และ row ของ LCD

lcd.begin();

}

void loop()

{

// ลูปสำหรับ  ASCII 'A' ถึง ASCII 'Z'

for (int thisLetter = 'A'; thisLetter <= 'Z'; thisLetter++)
```

```
{
// ลูปสำหรับ column
for (int thisCol = 0; thisCol < numRows; thisCol++)
{
// ลูปสำหรับ row
for (int thisRow = 0; thisRow < numCols; thisRow++)
{
// กำหนดตำแหน่ง Cursor
lcd.setCursor(thisRow,thisCol);
// แสดงตัวอักษร
lcd.write(thisLetter);
delay(200);
} } } }
```

**ตัวอย่างที่ 10.** การใช้คำสั่ง autoscroll() เพื่อกำหนดให้ข้อความเลื่อนไปทางซ้าย

```
#include <LiquidCrystal_I2C.h> //ประกาศ Library ของจอ I2C
// กำหนดค่า LCD แอดเดรสที่ตำแหน่ง 0x27 สำหรับ LCD แบบ 16 ตัวอักษร มี 2 บรรทัด
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); display
void setup()
{
// กำหนดตำแหน่ง column และ row ของ LCD
lcd.begin();
}
void loop()
{
// กำหนดตำแหน่ง Cursor ที่ (0,0)
lcd.setCursor(0, 0);
// แสดงผลเลข 0 ถึง 9
for (int thisChar = 0; thisChar < 10; thisChar++) {
lcd.print(thisChar);
delay(500);
}
// กำหนดตำแหน่ง Cursor ที่ (16,1)
lcd.setCursor(16,1);
// แสดงผลแบบอัตโนมัติ
lcd.autoscroll();
```

```
// แสดงผลเลข 0 ถึง 9
for (int thisChar = 0; thisChar < 10; thisChar++) {
lcd.print(thisChar);
delay(500);
}
// ปิดแสดงผลแบบอัตโนมัติ
lcd.noAutoscroll();
// ล้างหน้าจอ LCD
lcd.clear();
}
```

**สรุป**

การใช้ต่อ LCD กับบอร์ด Arduino ในปัจจุบันสามารถลดความยุ่งยากในต่อใช้งานได้ด้วยการแปลงสัญญาณการเชื่อมต่อจากเดิมแบบขนานไปเป็นแบบอนุกรมด้วยโมดูล I2C Serial Interface โดยก่อนใช้งานต้องติดตั้งไลบรารี่เสริมเพื่อให้จอ LCD ที่ต่อร่วมกับโมดูล I2C Serial Interface สามารถใช้งานร่วมกับบอร์ด Arduinoได้

**คำถาม**

1. จงเขียนโปรแกรมให้แสดงข้อความที่ LCD บรรทัดที่ 1
2. จงเขียนโปรแกรมให้แสดงข้อความที่ LCD บรรทัดที่ 2
3. จงเขียนโปรแกรมให้แสดงข้อความที่ LCD บรรทัดที่ 1 กระพริบ

## เอกสารอ้างอิง

1. Adith Jagadish Boloor. (2015). Arduino by Example. Birmingham : Packt Publishing Ltd.

2. http://arduino.cc/en/Tutorial/LiquidCrystalBlink : 2561

3. http://arduino.cc/en/Tutorial/LiquidCrystalCursor : 2561

4. http://arduino.cc/en/Tutorial/LiquidCrystalDisplay : 2561

5. http://arduino.cc/en/Tutorial/LiquidCrystalTextDirection : 2561

6. http://arduino.cc/en/Tutorial/LiquidCrystalScroll : 2561

7. http://arduino.cc/en/Tutorial/LiquidCrystalSerial : 2561

8. http://arduino.cc/en/Tutorial/LiquidCrystalSetCursor : 2561

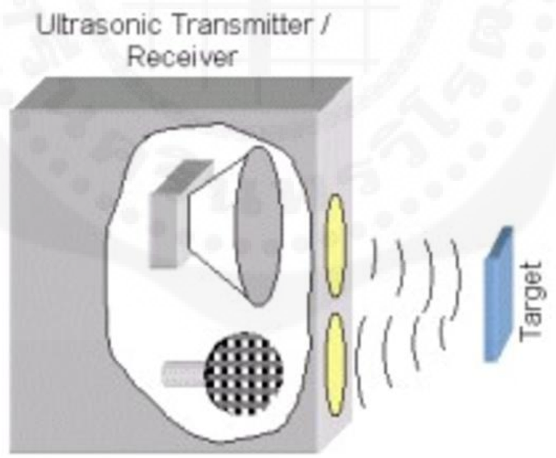
9. http://arduino.cc/en/Tutorial/LiquidCrystalAutoscroll : 2561

# บทที่ 5

# การวัดระยะด้วยคลื่นอัลตราโซนิค

## 5.1 เซนเซอร์คลื่นอัลตราโซนิค

เซ็นเซอร์คลื่นอัลตราโซนิคเป็นเซนเซอร์โดยอาศัยคลื่นเสียงที่มีความถี่สูงกว่า 20 kHz ซึ่งเป็นคลื่นในย่าน ที่มนุษย์จะไม่สามารถได้ยินเสียง เซนเซอร์คลื่นอัลตราโซนิคทำงานโดยอาศัยการกระจายคลื่นเสียงไปกระทบกับพื้นผิวของวัตถุซึ่งอาจเป็นของแข็งหรือของเหลว โดยคลื่นเสียงส่วนน้อยจะแทรกผ่านเข้าไปในตัววัตถุแต่คลื่นเสียงส่วนใหญ่ของคลื่นเสียงนี้จะสะท้อนกลับ ซึ่งช่วงเวลาของการสะท้อนกลับของคลื่นเสียงเป็นสัดส่วนโดยตรงกับระยะห่างระหว่างวัตถุกับเซนเซอร์ ทำให้เราสามารถหาระยะห่างของวัตถุได้




http://www.euclidres.com/motionSensors/motionSensors.html : 2561

รูปที่ 5.1 เซนเซอร์คลื่นอัลตราโซนิค

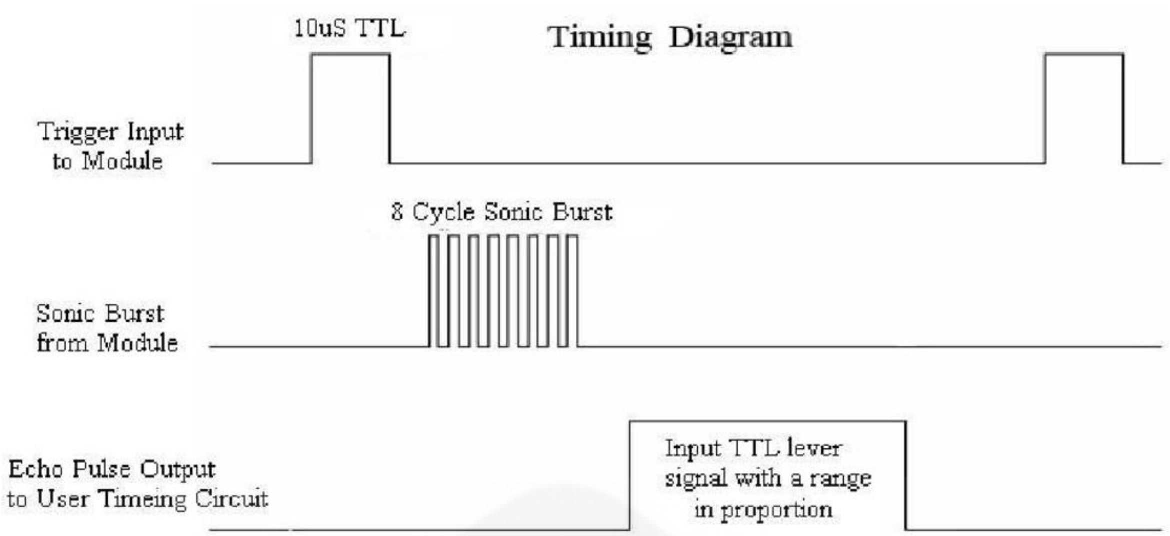
## 5.2 โมดูลเซนเซอร์คลื่นอัลตราโซนิค

ในปัจจุบันเซนเซอร์คลื่นอัลตราโซนิคถูกนำต่อร่วมกับอุปกรณ์อิเล็กทรอนิกส์เป็นโมดุลเซนเซอร์คลื่นอัลตราโซนิค ทำให้สามารถนำไปต่อใช้งานได้สะดวก ซึ่งโมดุลเซนเซอร์คลื่นอัลตราโซนิค จะมีจุดต่อใช้งานหลักตามรูปที่ 2. โมดูลเซนเซอร์คลื่นอัลตราโซนิคจะมีขาต่อใช้งาน 4 ขาคือ

1. Vcc แรงดันไฟเลี้ยงโมดูล 5V
2. Gnd
3. Trig ส่งสัญญาณให้โมดูลเซนเซอร์คลื่นอัลตราโซนิคส่ง คลื่นออกไป
4. Echo เมื่อโมดูลเซนเซอร์คลื่นอัลตราโซนิครับคลื่นสะท้อนกลับได้ จะส่งสัญญาณออกมาที่ขานี้


http://www.micropik.com/PDF/HCSR04.pdf : 2561

รูปที่ 5.2 โมดูลเซนเซอร์คลื่นอัลตราโซนิค




http://www.micropik.com/PDF/HCSR04.pdf : 2561

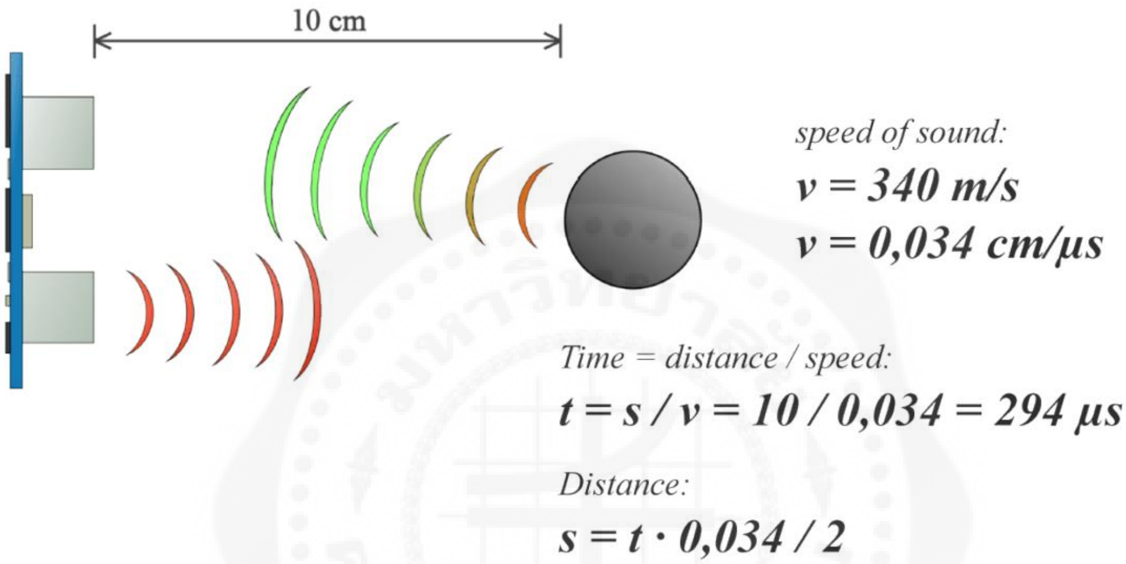
รูปที่ 5.3 Timing Diagram ของโมดูลเซนเซอร์คลื่นอัลตราโซนิค

จาก Timing Diagram จะต้องมีการส่งสัญญาณพัสส์สั้นๆประมาณ 10µS เข้าที่ขา Trig เพื่อเริ่มต้นวัดระยะทาง โมดูลเซนเซอร์คลื่นอัลตราโซนิคจะส่งคลื่นเสียงความถี่ 40 kHz ออกไป 8 พัสส์ แล้วรอสัญญาณเสียงสะท้อนกลับมา เมื่อมีสัญญาณเสียงสะท้อนกลับมาที่ขา Echo จะให้สัญญาณออกมาโดยเปลี่ยนสถานะจากลอจิก 0 เป็นลอจิก 1 โดยจะเป็นลอจิก 1 นานเป็นสัดสวนกับระยะทางที่วัดซึ่งระยะที่วัดจะหาได้จากสมการ

$$\text{ระยะทาง} = \frac{\text{ความกว้างของสัญญาณสะท้อนกลับ} \times 0.034(cm/\mu S)}{2} \quad \text{เซนติเมตร}$$

จากรูปที่ 5.4 ถ้าวัตถุห่างจากเซนเซอร์เป็นระยะ 10 ซม. และความเร็วในการเดินทางของเสียงเท่ากับ 340 m/s หรือ 0.034 cm/µs คลื่นเสียงจะต้องใช้เวลาเดินทางประมาณ 294 µS แต่ที่

ขา Echo pin กว่าจะได้รับสัญญาณสะท้อนกลับมาจะต้องคิดค่าเวลาเป็น 2 เท่าเนื่องจากคลื่นเสียงต้องเดินทางไปข้างหน้าหาวัตถุ และสะท้อนกลับมา ดังนั้นเพื่อที่ได้ระยะใน ซม. เราจะต้องคูณค่าเวลาที่ได้รับจากขา Echo pin ด้วย 0.034 และหาร ด้วย 2

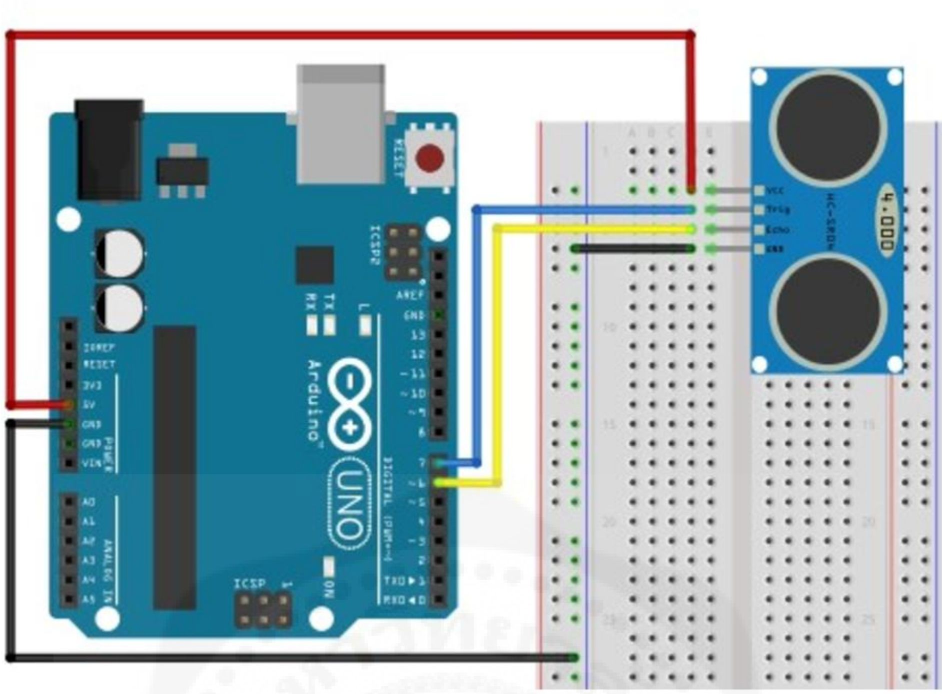



http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/ : 2561

รูปที่ 5.4 การหาระยะทางจากโมดูลเซนเซอร์คลื่นอัลตราโซนิค

### 5.3 การเชื่อมต่อกับบอร์ด Arduino

การเชื่อมต่อโมดูลเซนเซอร์คลื่นอัลตราโซนิคกับบอร์ด Arduino ทางขา Digital I/O ดังรูป



รูปที่ 5.4 การเชื่อมต่อโมดูลเซนเซอร์คลื่นอัลตราโซนิคกับบอร์ด Arduino

## 5.4 เริ่มใช้งาน Arduino IDE กับ โมดูลเซนเซอร์คลื่นอัลตราโซนิค

โมดูลเซนเซอร์คลื่นอัลตราโซนิคไม่ต้องติดตั้งไลบรารี่เสริม จึงใช้งานร่วมกับบอร์ด Arduino ได้เลย โดยก่อนเขียนโปรแกรมติดต่อกับโมดูลเซนเซอร์คลื่นอัลตราโซนิค มารู้จักคำสั่งและฟังก์ชั่นที่ใช้กับโมดูลเซนเซอร์คลื่นอัลตราโซนิค เพื่อให้เราสามารถใช้งานได้ตามที่เราต้องการ

**long ตัวแปร** > ก้านหนดข้อมูลเป็นตัวเลขที่มีความยาว

**ตัวแปร** = **pulseIn(ตำแหน่งขารับข้อมูล**, **HIGH**) > รับสัญญาณมาคำนวณระยะทาง

**ตัวแปร = ตัวแปร*0.034/2** > แปลงสัญญาณที่รับได้เป็นระยะทางหน่วยเป็น เซนติเมตร

**ตัวแปร = ตัวแปร*0.013/2** > แปลงสัญญาณที่รับได้เป็นระยะทางหน่วยเป็น นิ้ว

**ตัวอย่างที่ 1** การตรวจวัดระยะ เมื่อตรวจวัดได้ให้แสดงผลที่ Serial Monitor หน่วยที่วัดได้เป็นเซนติเมตรกำหนดให้ขาสัญญาณ Trig จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 7 และขาสัญญาณ Echo จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 6 ของบอร์ด Arduino โดยต่อวงจรตามรูปที่ 5.4

```
int trigPin = 7; //กำหนดขา 7 เป็นขา Trig

int echoPin = 6; //กำหนดขา 6 เป็นขา Echo

long duration, distance_cm;

void setup() {

Serial.begin (9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop() {

digitalWrite(trigPin, HIGH); // ส่งสัญญาณ Trig

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);
```

```
distance_cm= duration*0.034/2; // คำนวณระยะเป็นเซนติเมตร

Serial.print(distance_cm); // แสดงผล

Serial.println(" cm");

Serial.println("");

delay(1000);

}
```

**การทำงานของโปรแกรม**

โปรแกรมจะสั่งให้ส่งสัญญาณ Trig ออกไปด้วยคำสั่ง digitalWrite(trigPin, HIGH) 10 วินาที แล้วหยุดส่งด้วยคำสั่ง digitalWrite(trigPin, LOW) แล้วรอรับสัญญาณ pluse ที่จับการสะท้อนได้ด้วยคำสั่ง duration = pulseIn(echoPin, HIGH) แล้วนำค่าที่วัดได้มาคำนวณหาระยะ ด้วยคำสั่ง distance_cm= duration*0.034/2 ได้ค่าเป็นเซนติเมตร นำไปแสดงผล


รูปที่ 5.5 ผลการทดลองตามตัวอย่างที่ 1.

**ตัวอย่างที่ 2** การตรวจวัดระยะ เมื่อตรวจวัดได้ให้แสดงผลที่ Serial Monitor หน่วยที่วัดได้เป็นนิ้ว กำหนดให้ขาสัญญาณ Trig จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 7 และขาสัญญาณ Echo จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 6 ของบอร์ด Arduino โดยต่อวงจรตามรูปที่ 5.4

```
int trigPin = 7; //กำหนดขา 7 เป็นขา Trig

int echoPin = 6; //กำหนดขา 6 เป็นขา Echo

long duration, distance_inch;

void setup() {

Serial.begin (9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop() {

digitalWrite(trigPin, HIGH); // ส่งสัญญาณ Trig

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance_inch= duration*0.013/2; // คำนวณระยะเป็นนิ้ว
```

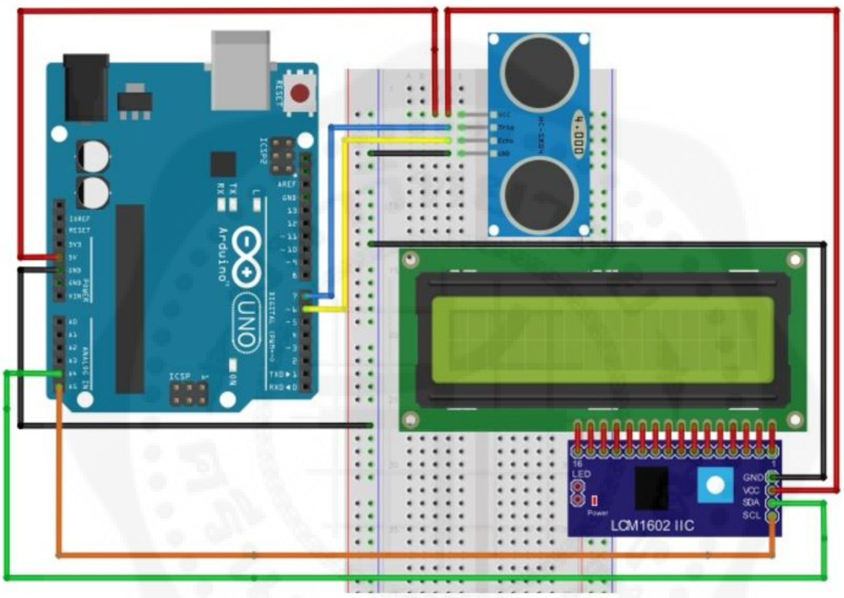
```
Serial.print(distance_inch);  // แสดงผล

Serial.print(" inch; ");

Serial.println("");

delay(1000);

}
```


รูปที่ 5.6 ผลการทดลองตามตัวอย่างที่ 2.

**การทำงานของโปรแกรม**

โปรแกรมจะสั่งให้ส่งสัญญาณ Trig ออกไปด้วยคำสั่ง digitalWrite(trigPin, HIGH) 10 วินาที แล้วหยุดส่งด้วยคำสั่ง digitalWrite(trigPin, LOW) แล้วรอรับสัญญาณ pluse ที่จับการสะท้อนได้ด้วยคำสั่ง duration = pulseIn(echoPin, HIGH) แล้วนำค่าที่วัดได้มาคำนวณหาระยะ ด้วยคำสั่ง distance_inch= duration*0.013/2 ได้ค่าเป็นนิ้ว นำไปแสดงผล

**ตัวอย่างที่ 3** การตรวจวัดระยะ เมื่อตรวจวัดได้ให้แสดงผลที่ LCD หน่วยที่วัดได้เป็นเซนติเมตรกำหนดให้ขาสัญญาณ Trig จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 7 และขาสัญญาณ Echo จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 6 ของบอร์ด Arduino โดยต่อวงจรตามรูปที่ 5.7




รูปที่ 5.7 วงจรสำหรับตัวอย่างที่ 3.

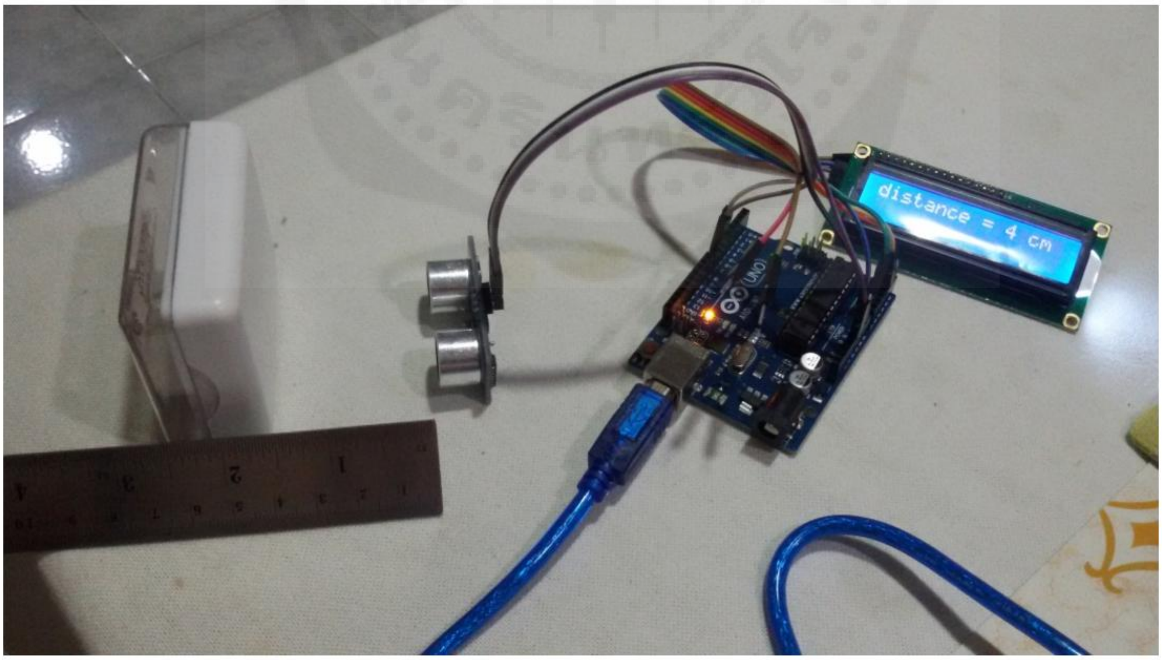
```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

int trigPin = 7; //กำหนดขา 7 เป็นขา Trig

int echoPin = 6; //กำหนดขา 6 เป็นขา Echo

long duration, distance_cm;
```

```
void setup() {
lcd.begin();
Serial.begin(9600);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}
void loop() {
digitalWrite(trigPin, HIGH); // ส่งสัญญาณ Trig
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance_cm= duration*0.034/2; // คำนวณระยะเป็นเซ็นติเมตร
// แสดงผลที่ LCD
lcd.clear();
lcd.setCursor(0,0);
lcd.print("distance = ");
lcd.print(distance_cm);
```

```
lcd.print(" cm");

delay(1000);

}
```

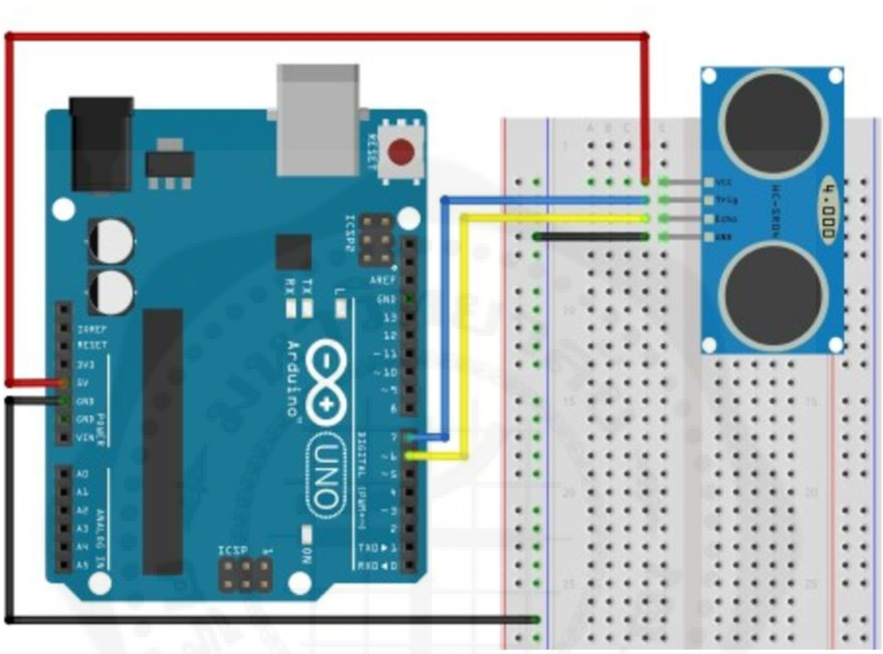
**การทำงานของโปรแกรม**

โปรแกรมจะสั่งให้ส่งสัญญาณ Trig ออกไปด้วยคำสั่ง digitalWrite(trigPin, HIGH) 10 วินาที แล้วหยุดส่งด้วยคำสั่ง digitalWrite(trigPin, LOW) แล้วรอรับสัญญาณ pluse ที่จับการสะท้อนได้ด้วยคำสั่ง duration = pulseIn(echoPin, HIGH) แล้วนำค่าที่วัดได้มาคำนวณหาระยะ ด้วยคำสั่ง distance_cm= duration*0.034/2 ได้ค่าเป็นเซนติเมตร นำไปแสดงผลที่ LCD




รูปที่ 5.7 ผลการทดลองตามตัวอย่างที่ 3.

**ตัวอย่างที่ 4** การตรวจวัดระยะ ให้ Ultrasonic ตรวจวัดระยะโดยเมื่อตรวจวัดระยะได้มากกว่า 10 เซนติเมตร ให้ LED ที่ขา 13 (สีส้ม) ของบอร์ด Arduino LED สว่าง กำหนดให้ขาสัญญาณ Trig จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 7 และขาสัญญาณ Echo จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 6 ของบอร์ด Arduino




รูปที่ 5.7 การต่อวงรสำหรับตัวอย่างที่ 4.

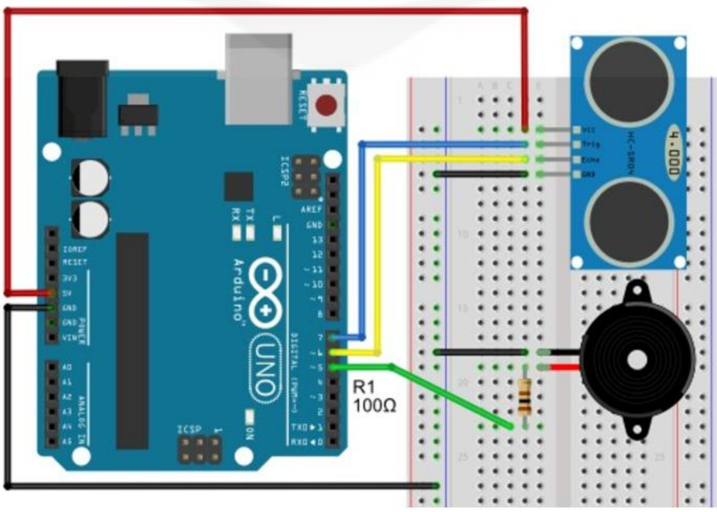
```
int trigPin = 7; //กำหนดขา 7 เป็นขา Trig

int echoPin = 6; //กำหนดขา 6 เป็นขา Echo

int ledPin= 13;

long duration, distance_inch, distance_cm;

void setup() {
```

```
Serial.begin (9600);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(ledPin, OUTPUT);
}
void loop() {
digitalWrite(trigPin, HIGH); // ส่งสัญญาณ Trig
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance_cm= duration*0.034/2; // คำนวณระยะเป็นเซ็นติเมตร
if (distance_cm < 10) {
digitalWrite(ledPin, LOW);  }
else {
digitalWrite(ledPin, HIGH);  }
delay(200);
}
```

**การทำงานของโปรแกรม**

โปรแกรมจะสั่งให้ส่งสัญญาณ Trig ออกไปด้วยคำสั่ง digitalWrite(trigPin, HIGH) 10 วินาที แล้วหยุดส่งด้วยคำสั่ง digitalWrite(trigPin, LOW) แล้วรอรับสัญญาณ pluse ที่จับการสะท้อนได้ด้วยคำสั่ง duration = pulseIn(echoPin, HIGH) แล้วนำค่าที่วัดได้มาคำนวณหาระยะ ด้วยคำสั่ง distance_cm= duration*0.034/2 ได้ค่าเป็นเซนติเมตร นำไปเปรียบเทียบด้วยคำสั่ง if (distance_cm < 10) หากระยะต่ำกว่า 10 เซนติเมตร LED pin 13 ดับ ถ้าระยะมากกว่า 10 เซนติเมตร LED pin 13 สว่าง

**ตัวอย่างที่ 5** การตรวจวัดระยะ ให้ Ultrasonic ตรวจวัดระยะโดยเมื่อตรวจวัดระยะได้น้อยกว่า 10 เซนติเมตรให้บัซเซอร์กำเนิดเสียงที่ความถี่ 500Hz และเมื่อตรวจวัดระยะได้น้อยกว่า 5 เซนติเมตรให้
บัซเซอร์กำเนิดเสียงที่ความถี่ 1500Hz กำหนดให้ขาสัญญาณ Trig จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 7 ขาสัญญาณ Echo จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 6 และบัซเซอร์ต่อเข้าที่ขา Digital I/O ขา 5 ของบอร์ด Arduino




รูปที่ 5.8 การต่อวงจรสำหรับตัวอย่างที่ 5.

```
int trigPin = 7; //กำหนดขา 7 เป็นขา Trig

int echoPin = 6; //กำหนดขา 6 เป็นขา Echo

int buzzPin = 5; // กำหนดตำแหน่งบัซเซอร์

long duration, distance_cm;

void setup() {

Serial.begin (9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(buzzPin, OUTPUT);

}

void loop() {

digitalWrite(trigPin, HIGH); // ส่งสัญญาณ Trig

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance_cm= duration*0.034/2; // คำนวณระยะเป็นเซ็นติเมตร

if (distance_cm < 10) {
```

```
tone(buzzPin, 500, 1000);  // สั่งให้บัซเซอร์มีเสียงดังที่ความถี่ 500Hz  }

if (distance_cm < 5) {

tone(buzzPin, 1500, 1000);  // สั่งให้บัซเซอร์มีเสียงดังที่ความถี่ 1500Hz  }

delay(200);

}
```

**การทำงานของโปรแกรม**

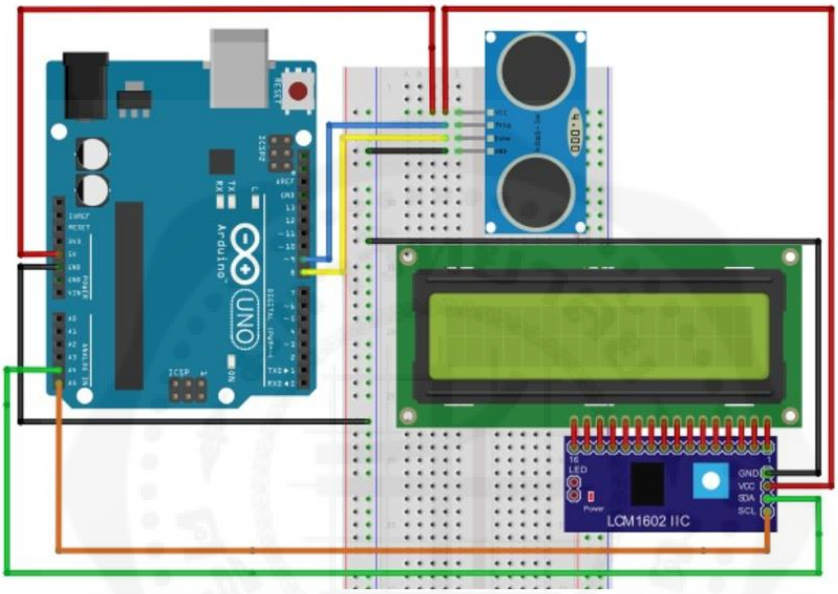
โปรแกรมจะสั่งให้ส่งสัญญาณ Trig ออกไปด้วยคำสั่ง digitalWrite(trigPin, HIGH) 10 วินาที แล้วหยุดส่งด้วยคำสั่ง digitalWrite(trigPin, LOW) แล้วรอรับสัญญาณ pluse ที่จับการสะท้อนได้ด้วยคำสั่ง duration = pulseIn(echoPin, HIGH) แล้วนำค่าที่วัดได้มาคำนวณหาระยะ ด้วยคำสั่ง distance_cm= duration*0.034/2 ได้ค่าเป็นเซนติเมตร นำไปเปรียบเทียบด้วยคำสั่ง if (distance_cm < 10) หากระยะต่ำกว่า 10 เซนติเมตร บัซเซอร์จะดังที่ความถี่ 500Hz ด้วยคำสั่ง tone(buzzPin, 500, 1000) ถ้าระยะกว่า5 เซนติเมตร บัซเซอร์จะดังที่ความถี่  1500Hz ด้วยคำสั่ง tone(buzzPin, 1500, 1000)

**สรุป**

โมดูลเซนเซอร์คลื่นอัลตราโซนิคเชื่อมต่อกับบอร์ด Arduino ทางขา Digital I/O โดยเขียนโปรแกรมควบคุมบอร์ด Arduino ให้ส่งสัญญาณ Trig ออกไปด้วยคำสั่ง digitalWrite(ขาtrig, HIGH) 10 วินาที แล้วหยุดส่งด้วยคำสั่ง digitalWrite(ขาtrig, LOW) แล้วรอรับสัญญาณ pluse ที่จับการสะท้อนได้ด้วยคำสั่ง duration = pulseIn(ขาecho, HIGH) แล้วนำค่าที่อ่านได้มาคำนวณตามความต้องการ

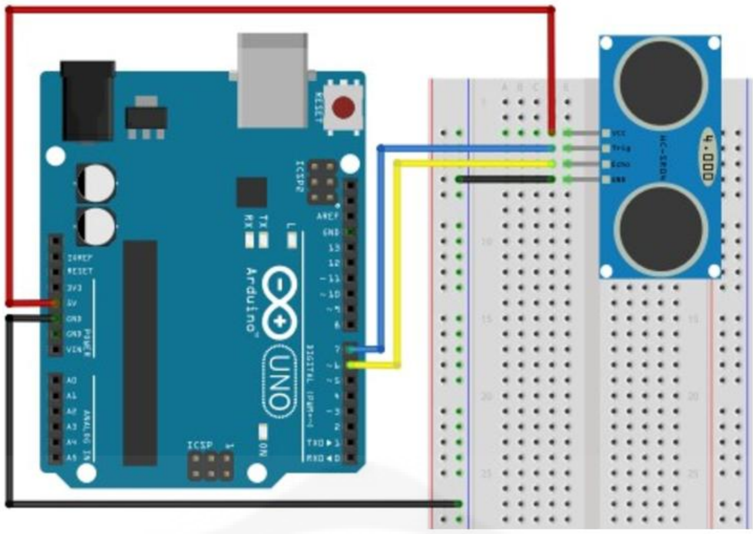
**คำถาม**

1. การตรวจวัดระยะ เมื่อตรวจวัดได้ให้แสดงผลที่ LCD หน่วยที่วัดได้เป็นนิ้วกำหนดให้ขาสัญญาณ Trig จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 9 และขาสัญญาณ Echo จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 8 ของบอร์ด Arduino



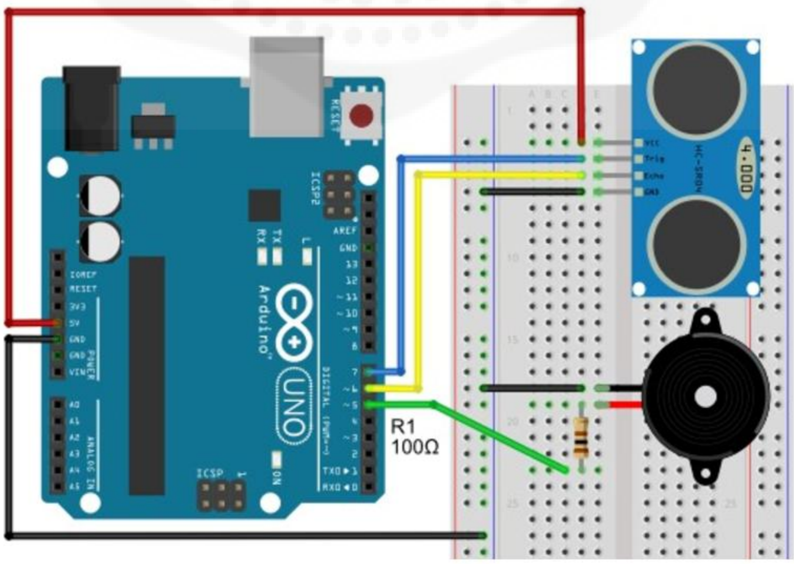

รูปที่ 5.9 การต่อวงจรสำหรับคำถามข้อ 1.

2. การตรวจวัดระยะ ให้ Ultrasonic ตรวจวัดระยะโดยเมื่อตรวจวัดระยะได้น้อยกว่า 15 เซนติเมตร ให้ LED ที่ขา 13 (สีส้ม) ของบอร์ด Arduino LED สว่าง กำหนดให้ขาสัญญาณ Trig จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 7 และขาสัญญาณ Echo จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 6 ของบอร์ด Arduino



รูปที่ 5.10 การต่อวงจรสำหรับคำถามข้อ 2.

3. การตรวจวัดระยะ ให้ Ultrasonic ตรวจวัดระยะโดยเมื่อตรวจวัดระยะได้น้อยกว่า 10 นิ้ว ให้บัซเซอร์กำเนิดเสียงที่ความถี่ 500Hz และเมื่อตรวจวัดระยะได้น้อยกว่า 5 นิ้ว ให้บัซเซอร์กำเนิดเสียงที่ความถี่ 1500Hz กำหนดให้ขาสัญญาณ Trig จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 7 ขาสัญญาณ Echo จาก Ultrasonic ต่อเข้าที่ขา Digital I/O ขา 6 และบัซเซอร์ต่อเข้าที่ขา Digital I/O ขา 5 ของบอร์ด Arduino




รูปที่ 5.11 การต่อวงจรสำหรับคำถามข้อ 3

## เอกสารอ้างอิง

1. Mark Svaljek. (2015). Arduino Succinctly. Morrisville, NC : Syncfusion Inc.

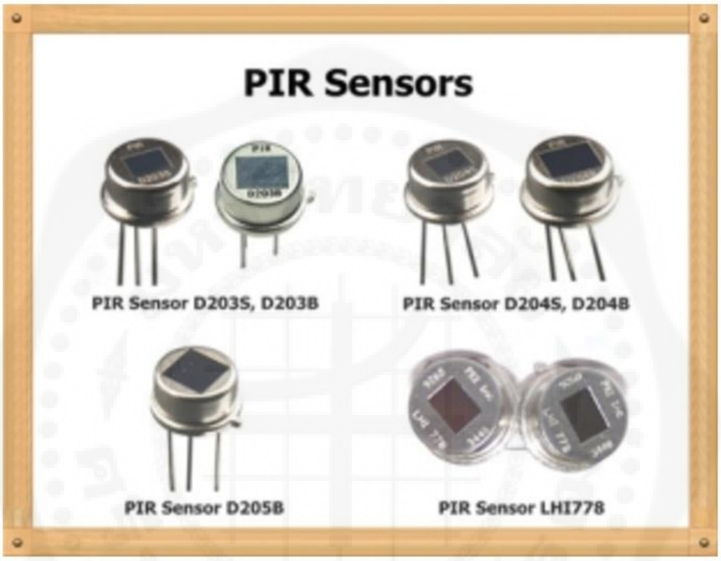
2. http://www.micropik.com/PDF/HCSR04.pdf : 2561

3. http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/ : 2561

# บทที่ 6

# เซนเซอร์ตรวจจับความเคลื่อนไหว PIR

## 6.1 เซนเซอร์ PIR




http://www.kksensor.com/pir-sensors.html : 2561

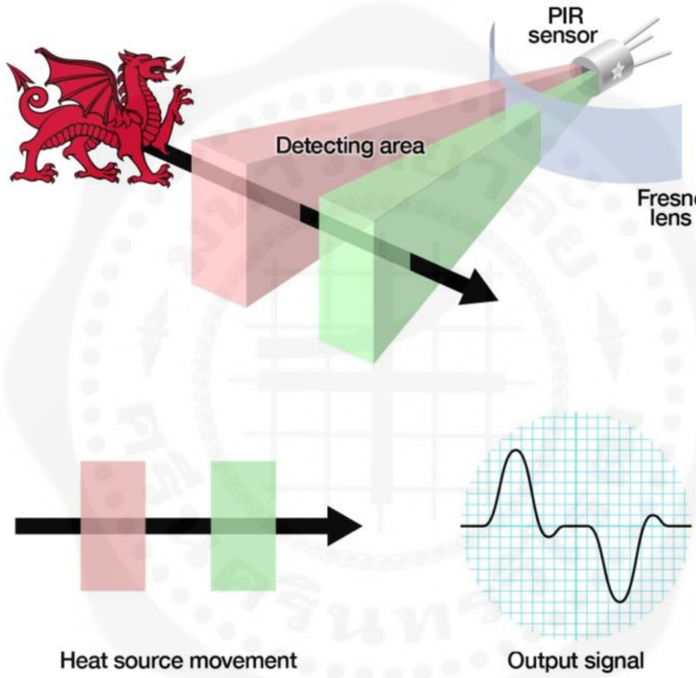
รูปที่ 6.1 เซนเซอร์ PIR

PIR ย่อมาจาก Passive infrared คืออุปกรณ์ตรวจจับความเคลื่อนไหว โดยใช้ปรากฏการณ์ Pyro electric effect ซึ่งหลักการคือการเปลี่ยนแปลงความร้อนจากรังสีอินฟราเรด ทำให้อะตอมในผลึกของวัสดุเกิดการเปลี่ยนแปลงตำแหน่ง ซึ่งทำให้ polarization เปลี่ยนเช่นกัน ทำให้ voltage มีค่าสูงขึ้นและนำมาแปลงเป็นเสียง หรือแสง สำหรับใช้เป็นเซนเซอร์ได้ จึงมีการนำเอา PIR มาประยุคใช้งานกันเป็นอย่างมากใช้เพื่อตรวจจับการเคลื่อนไหวของสิ่งมีชีวิต หรือ ตรวจจับการบุกรุกในงานรักษาความปลอดภัย หรือที่เรียกกันว่า PIR motion sensor นั่นเอง

## 6.2 การทำงานของเซนเซอร์ PIR

### 6.2.1 หลักการทำงานเซนเซอร์ PIR

เซนเซอร์ PIR จะทำงานซับซ้อนกว่ามากจากเซนเซอร์ชนิดอื่น ๆ เช่น Photocells เนื่องจากมีหลายตัวแปรที่มีผลต่อเซนเซอร์อินพุทและเอาท์พุท การอธิบายวิธีการทำงานของเซนเซอร์เป็นดังรูปที่ 6.1




https://learn.adafruit.com/assets/35647 : 2561

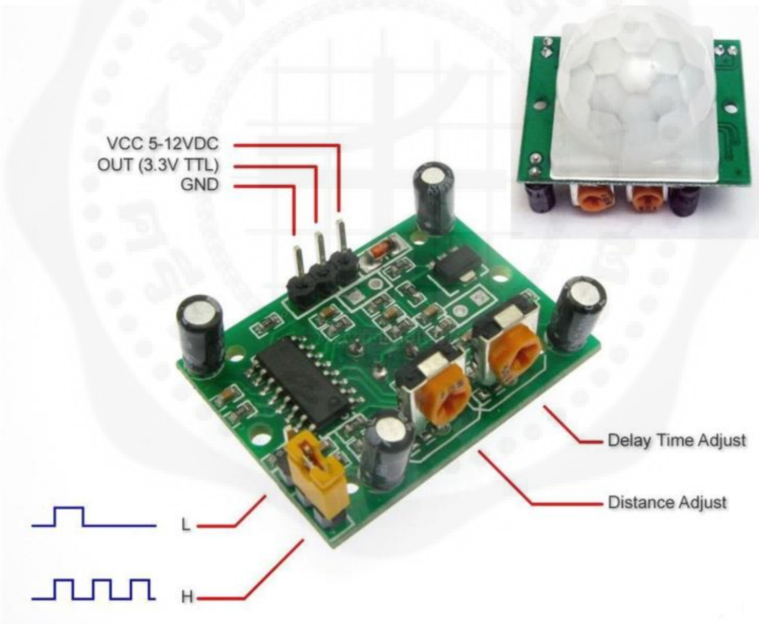
รูปที่ 6.2 การทำงานของ PIR

เซนเซอร์ PIR จะมีตัวตรวจจับ 2 ชุดในแต่ละชุดทำจากวัสดุพิเศษที่ไวต่อรังสีอินฟราเรด มีเลนส์ครอบทำหน้าที่กระจายแสงจากจุดโฟกัสให้กระจายเป็นลำออกไป เมื่อมี คน หรือ สัตว์ ซึ่งมีความร้อนจากร่างกายเคลื่อนที่ผ่านเข้ามาในพื้นที่ตัวจับ จากรังสีความร้อนของคน หรือสัตว์ที่แผ่ออกมารอบๆตัวซึ่งมีคลื่นรังสีอินฟราเรดอยู่ด้วย ทำให้อุณหภูมิในบริเวณนั้นมีการเปลี่ยนแปลง PIR จะสามารถตรวจจับคลื่นรังสีอินฟราเรดที่แพร่ออกมาจากสิ่งมีชีวิตได้ โดย

PIR จะรับคลื่นรังสีมายังตัวตรวจจับซึ่งจะเปลี่ยนพลังงานความร้อนจากรังสีอินฟราเรด เป็นพลังงานไฟฟ้า กล่าวคือเมื่อคน หรือสัตว์เคลื่อนที่ผ่านตัวตรวจจับรังสีอินฟราเรดตัวที่ 1 จะได้สัญญาณเอาท์พุทออกมาสูงกว่าแรงดันปรกติ และเมื่อคน หรือสัตว์เคลื่อนที่ผ่านตัวตรวจจับรังสีอินฟราเรดตัวที่ 2 จะได้แรงดันเอาท์พุทต่ำกว่าค่าแรงดันปรกติ

### 6.2.2 โมดูลเซนเซอร์ PIR

ในปัจจุบันเซนเซอร์ PIR ถูกนำต่อร่วมกับอุปกรณ์อิเล็กทรอนิกส์เป็นโมดุลเซนเซอร์ PIR ทำให้สามารถนำไปต่อใช้งานได้สะดวก ซึ่งโมดุลเซนเซอร์ PIR จะมีจุดต่อใช้งานหลักตามรูปที่ 6.3




http://www.lelong.com.my/hc-sr501-motion-sensor-arduino-ir-body-infrared-sensor-module-ginnieooi-158449329-2017-12-Sale-P.htm : 2561

รูปที่ 6.3 จุดต่อใช้งานโมดุล

ขา Output จะให้เอาท์พุทออกมาในรูปของสัญญาณดิจิตอล โดยสามารถเลือกลักษณะของสัญญาณได้จากการเลือกตำแหน่งจั๊มเปอร์ ซึ่งมีรูปแบบสัญญาณเอาท์พุท 2 แบบ คือ

- สัญญาณแบบคลื่นพัลส์ต่อเนื่อง (H)

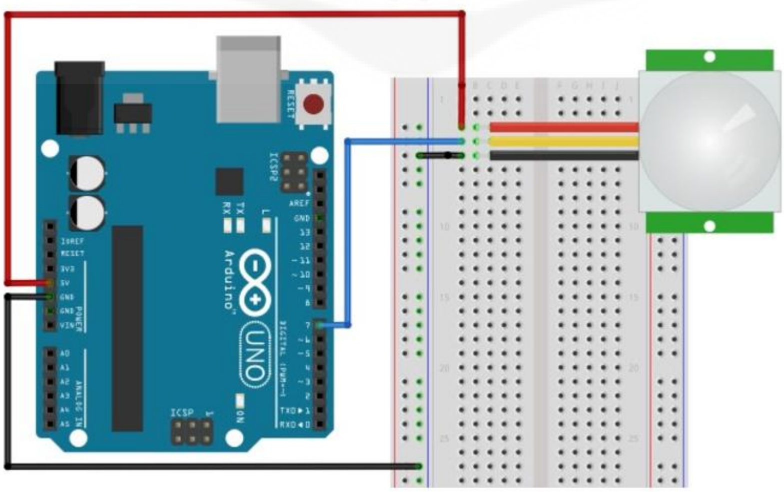
- สัญญาณลอจิก 1 ค้างไว้ จนกว่าจะไม่สามารถจับความเคลื่อนไหวได้ จึงจะกลับมาเป็นลอจิก 0 (L)

ปุ่มปรับ Delay Time Adjust ใช้สำหรับปรับการหน่วงเวลาตรวจจับ หากปรับหน่วงเวลานาน จะทำให้ค้างสถานะ 1 ไว้นาน เนื่องจากการตรวจจับไม่ได้ตอบสนองแบบทันทีทันใด เมื่อตรวจจับพบ จะให้สถานะทางเอาท์พุทเป็นลอจิก 1 ค้างไว้ เวลาที่ค้างไว้ขึ้นอยู่กับการปรับ Delay Time Adjust เมื่อหมดเวลาจึงตรวจจับอีกครั้ง หากตรวจจับไม่เจอก็จะให้สถานะเป็น 0 แต่หากจับเจอ จะค้างสถานะ 1 ไว้

ปุ่มปรับ Sensitivity Adjust ใช้ปรับความไว และระยะทางในการตรวจจับ

**6.3 การเชื่อมต่อกับบอร์ด Arduino**

การเชื่อมต่อโมดูลเซนเซอร์ PIR กับบอร์ด Arduino ทางขา Digital I/O ดังรูป



รูปที่ 6.4 การเชื่อมโมดูลเซนเซอร์ PIR กับบอร์อด Arduino

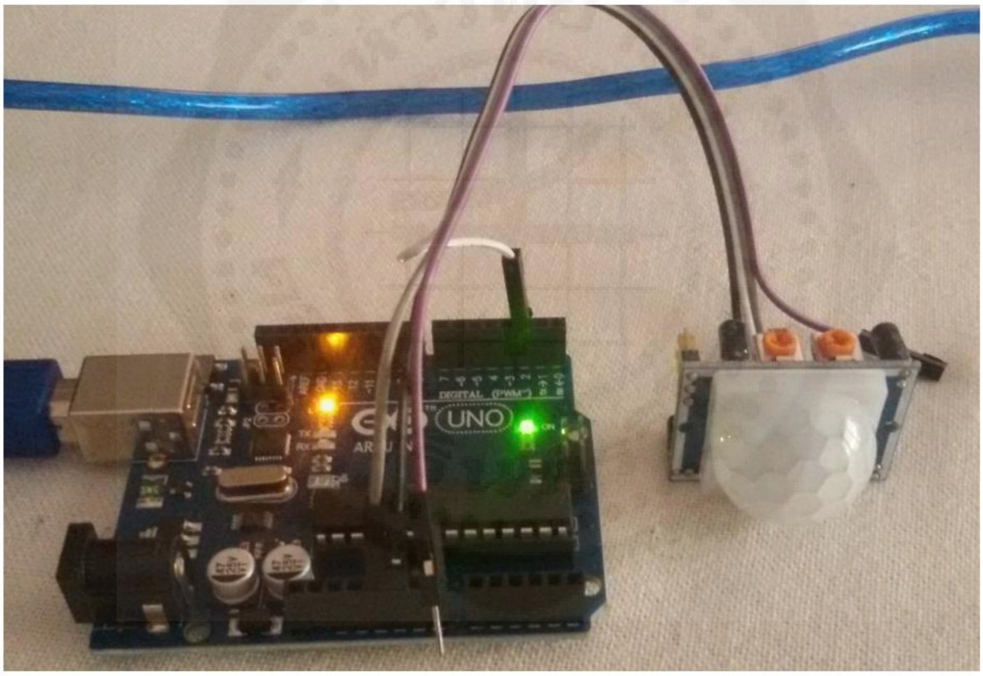
## 6.4 เริ่มใช้งาน Arduino IDE กับโมดูลเซนเซอร์ PIR

โมดูลเซนเซอร์ PIR ไม่ต้องติดตั้งไลบรารี่เสริมจึงใช้งานร่วมกับบอร์ด Arduinoได้เลย

**ตัวอย่างที่ 1** การตรวจจับความเคลื่อนไหว เมื่อตรวจจับได้ให้ LED ที่ขา 13 (สีส้ม) ของบอร์ด Arduino LED สว่าง และสัญญาณจาก PIR ต่อเข้าที่ขา Digital I/O ขา 7 ของบอร์ด Arduino โดยต่อวงจรตามรูปที่ 6.4

```
int ledPin= 13;

int inputPin= 7;

void setup() {

pinMode(ledPin, OUTPUT);

pinMode(inputPin, INPUT);

}

void loop() {

int value= digitalRead(inputPin);

if (value == HIGH)

{

digitalWrite(ledPin, HIGH);

delay(3000);
```

```
}
else
{
digitalWrite(ledPin, LOW);
delay(3000); }
}
```




รูปที่ 6.5 ผลการทดลองตามตัวอย่างที่ 1.

**การทำงานของโปรแกรม**

โปรแกรมจะรับสัญญาณการตรวจจับจาก PIR ด้วยคำสั่ง int value= digitalRead(inputPin) นำค่าที่รับได้มาตรวจสอบว่าเป็นลอจิก High หรือไม่ด้วยคำสั่ง if (value == HIGH) โดยถ้า PIR ตรวจจับได้จะส่งสัญญาณลอจิก High ออกมา ซึ่งถ้าใช่จะสั่งให้ LED ที่ขา 13 สว่าง

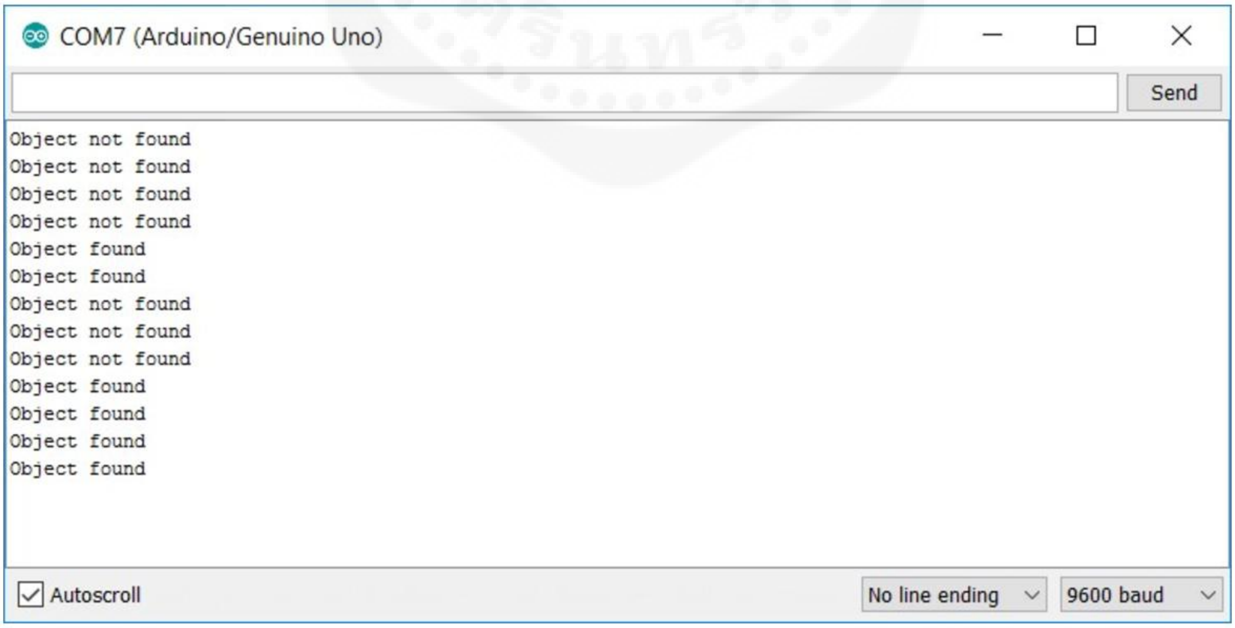
**ตัวอย่างที่ 2** การตรวจจับความเคลื่อนไหว เมื่อตรวจจับได้ให้แสดงผลที่ Serial Monitor โดยถ้าตรวจจับได้ให้แสดงข้อความ “Object found” แต่ถ้าตรวจจับไม่ได้ให้แสดงข้อความ “Object not found” และสัญญาณจาก PIR ต่อเข้าที่ขา Digital I/O ขา 7 ของบอร์ด Arduino โดยต่อวงจรตามรูปที่ 6.4

```
int inputPin= 7;

void setup() {

pinMode(inputPin, INPUT);

Serial.begin(9600);  }

void loop() {

int value= digitalRead(inputPin);

if (value == HIGH)  {

Serial.println("Object found");
```

```
delay(3000);   }

else  {

Serial.println("Object not found");

delay(3000);   }

}
```

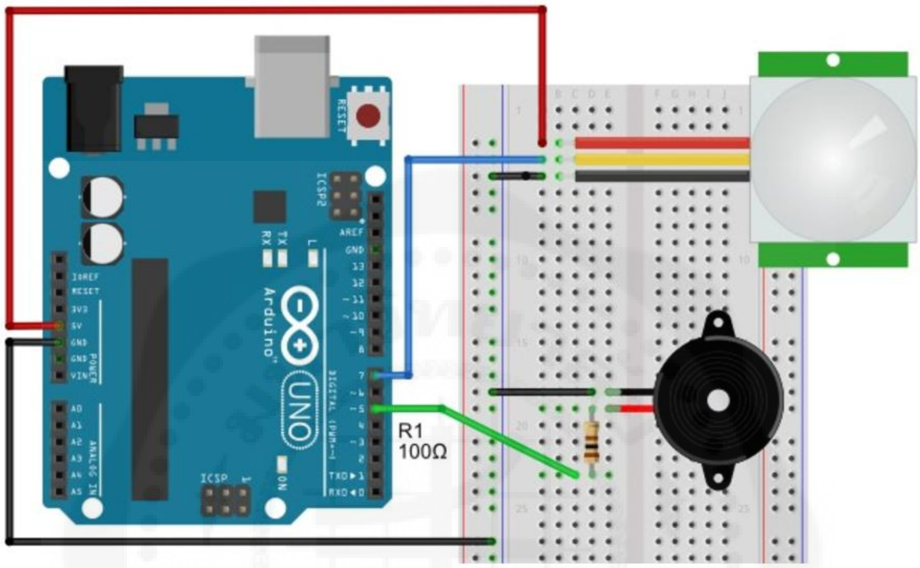
**การทำงานของโปรแกรม**

โปรแกรมจะรับสัญญาณการตรวจจับจาก PIR ด้วยคำสั่ง  int value= digitalRead(inputPin) นำค่าที่รับได้มาตรวจสอบว่าเป็นลอจิก High หรือไม่ด้วยคำสั่ง if (value == HIGH)  โดยถ้า PIR ตรวจจับได้จะส่งสัญญาณลอจิก High ออกมา ซึ่งถ้าใช่จะสั่งให้แสดงข้อความ Object found ที่ Serial Monitor แต่ถ้าตรวจจับไม่พบจะสั่งให้แสดงข้อความ Object not found




รูปที่ 6.6 ผลการทดลองตามตัวอย่างที่ 2.

**ตัวอย่างที่ 3** การตรวจจับความเคลื่อนไหว เมื่อตรวจจับได้ให้บัซเซอร์กำเนิดเสียงที่ความถี่ 1500Hz และสัญญาณจาก PIR ต่อเข้าที่ขา Digital I/O ขา 7 ของบอร์ด Arduino และบัซเซอร์ต่อเข้าที่ขา Digital I/O ขา 5 ของบอร์ด Arduino




รูปที่ 6.7 การต่อวงจรสำหรับตัวอย่างที่ 3.

```
int buzzPin = 5; // กำหนดตำแหน่งบัซเซอร์

int inputPin= 7; // PIR Input

void setup() {

pinMode(buzzPin, OUTPUT);

pinMode(inputPin, INPUT);

}
```

```
void loop() {

int value= digitalRead(inputPin);

if (value == HIGH) {

tone(buzzPin, 500, 1000);  // สั่งให้บัซเซอร์มีเสียงดังที่ความถี่ 500Hz

delay(3000) ;  }

else  {

tone(buzzPin, 0, 1000);  // สั่งให้บัซเซอร์มีเสียงดังที่ความถี่ 0Hz

delay(3000) ;  }

}
```

**การทำงานของโปรแกรม**

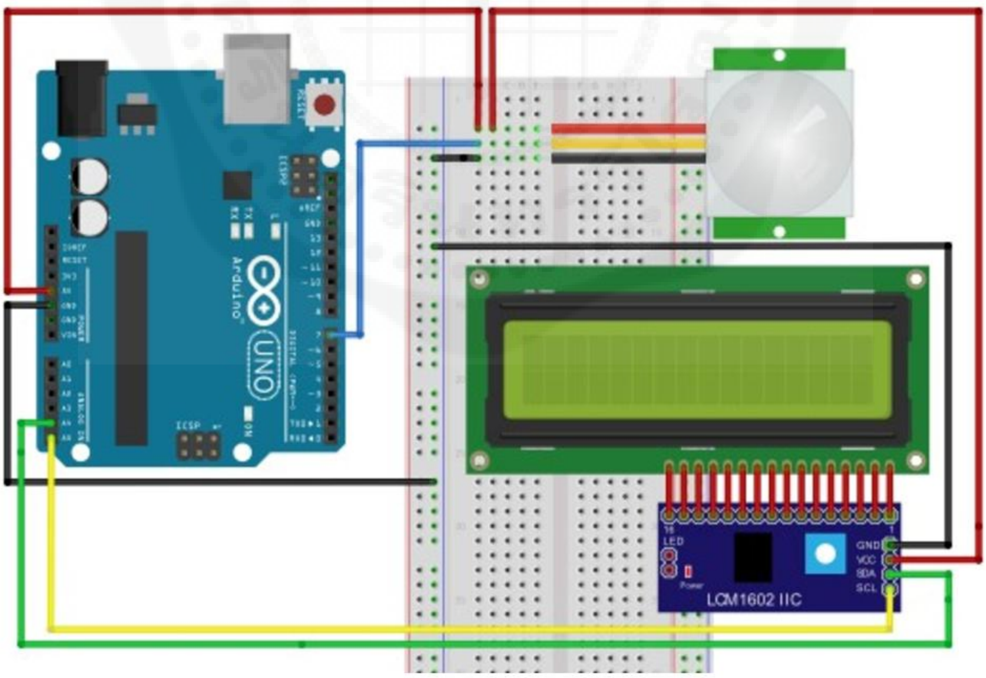
โปรแกรมจะรับสัญญาณการตรวจจับจาก PIR ด้วยคำสั่ง int value= digitalRead(inputPin) นำค่าที่รับได้มาตรวจสอบว่าเป็นลอจิก High หรือไม่ด้วยคำสั่ง if (value == HIGH) โดยถ้า PIR ตรวจจับได้จะส่งสัญญาณลอจิก High ออกมา ซึ่งถ้าใช่จะสั่งให้บัซเซอร์ส่งเสียงออกมา แต่ถ้าตรวจจับไม่พบจะสั่งให้บัซเซอร์
หยุดส่งเสียงออกมา

## สรุป

การต่อใช้งานโมดูลเซนเซอร์ PIR กับบอร์ด Arduino เราสามารถเชื่อมต่อได้ทางขา Digital I/O โดยเขียนโปรแกรมควบคุมบอร์ด Arduino ให้รับสัญญาณจากขาเอาท์พุทของโมดูลเซนเซอร์ PIR เข้ามาตรวจสอบสถานะลอจิก โดยนำผลจากการอ่านสถานะลอจิกที่ได้มาดำเนินการตามโปรแกรมที่ได้กำหนดไว้

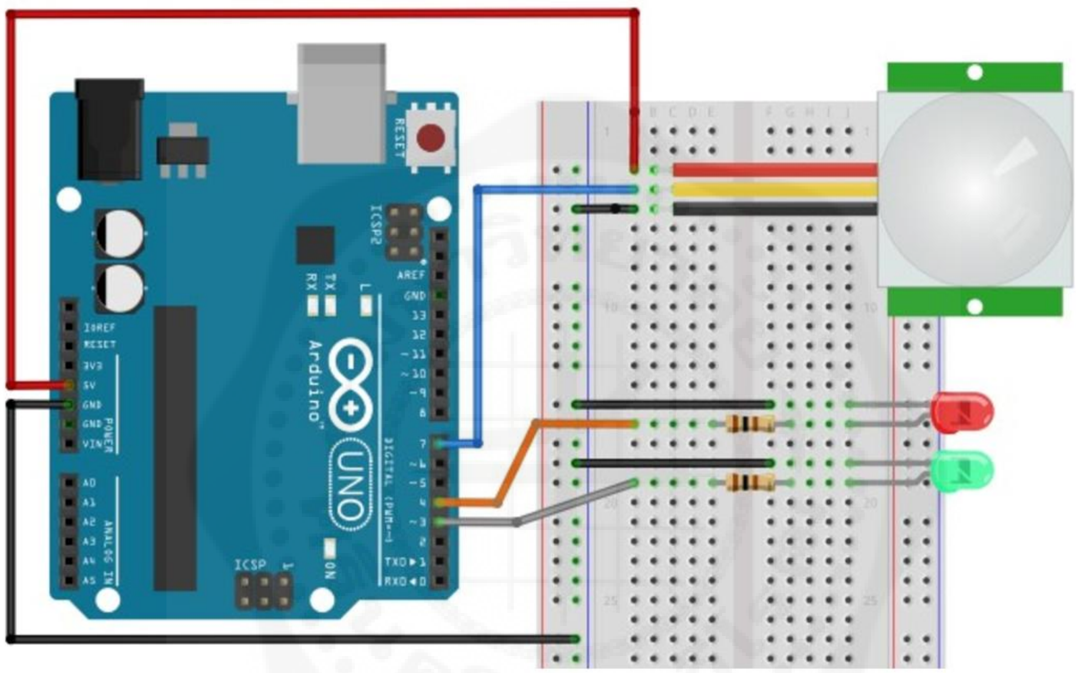
## คำถาม

1. การตรวจจับความเคลื่อนไหว เมื่อตรวจจับได้ให้แสดงผลที่ LCD โดยถ้าตรวจจับได้ให้แสดงข้อความ “Object found” แต่ถ้าตรวจจับไม่ได้ให้แสดงข้อความ “Object not found” และสัญญาณจาก PIR ต่อเข้าที่ขา Digital I/O ขา 2 ของบอร์ด Arduino



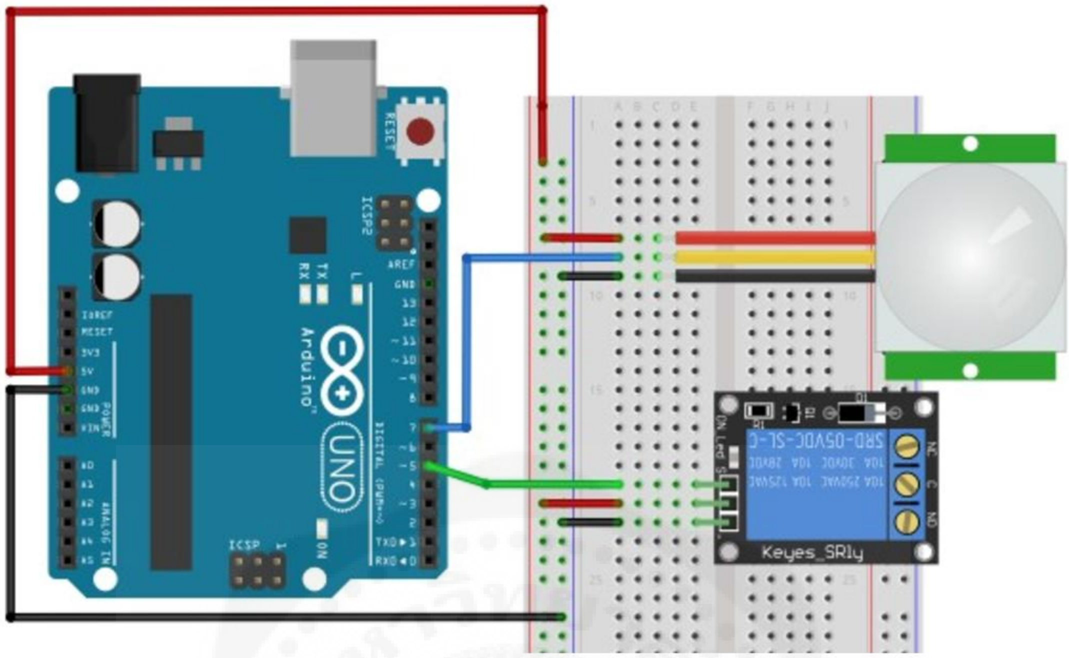

รูปที่ 6.8 การต่อวงจรสำหรับคำถามข้อ 1.

2. การตรวจจับความเคลื่อนไหว เมื่อตรวจจับได้ให้ LED ที่ขา Digital I/O ขา 3 ของบอร์ด Arduino LED สว่าง ถ้าตรวจจับไม่ได้ให้ LED ที่ขา Digital I/O ขา 4 ของบอร์ด Arduino LED สว่างและสัญญาณจาก PIR ต่อเข้าที่ขา Digital I/O ขา 7 ของบอร์ด Arduino



รูปที่ 6.7 การต่อวงจรสำหรับคำถามข้อ 2.

3. การตรวจจับความเคลื่อนไหว เมื่อตรวจจับได้ให้รีเลย์ทำงาน และสัญญาณจาก PIR ต่อเข้าที่ขา Digital I/O ขา 7 ของบอร์ด Arduino และ Digital I/O ขา 5 ของบอร์ด Arduino สำหรับต่อวงจรขับรีเลย์




รูปที่ 6.8 การต่อวงจรสำหรับคำถามข้อ 3.

## เอกสารอ้างอิง

1. Mark Svaljek. (2015). Arduino Succinctly. Morrisville, NC : Syncfusion Inc.

2. https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/overview : 2561

3. http://www.instructables.com/id/PIR-Motion-Sensor-Tutorial/ : 2561

# บทที่ 7

# การวัดอุณหภูมิและความชื้นด้วยเซนเซอร์ DHT11

## 7.1 เซนเซอร์ DHT11

เซนเซอร์ DHT11เป็นเซนเซอร์ที่ใช้สภาพแวดล้อมในอากาศโดยสามาถวัดค่าอุณหภูมิและความชื้นในอากาศ และให้เอาท์พุทออกมาในรูปของสัญญาณดิจิตอล สามารถนำไปเชื่อมต่อกับ Arduino ได้ง่าย และมีไลบรารี่ใช้งานกับ Arduino


https://wiki.microduino.cc/index.php/File:DHT11.jpg : 2561

รูปที่ 7.1 เซนเซอร์ DHT11

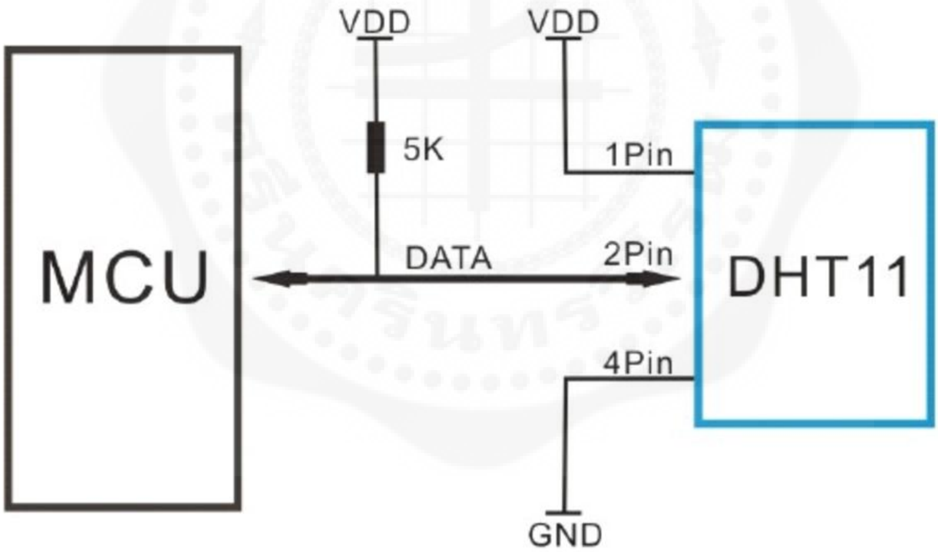
## 7.2 การทำงานของเซนเซอร์ DHT11

### 7.2.1 เซนเซอร์ DHT11 มีความสามารถในการวัดดังนี้

ย่านวัดความชื้น 20-90% RH โดยมีค่าความแม่นยำ ± 5% RH แสดงผลแบบ 8 บิต

ย่านวัดอุณหภูมิ 0 -50 องศาเซลเซียส โดยมีค่าความแม่นยำ ± 2 องศาเซลเซียส แสดงผลแบบ 8 บิต

### 7.2.2 การสื่อสารกับ Arduino



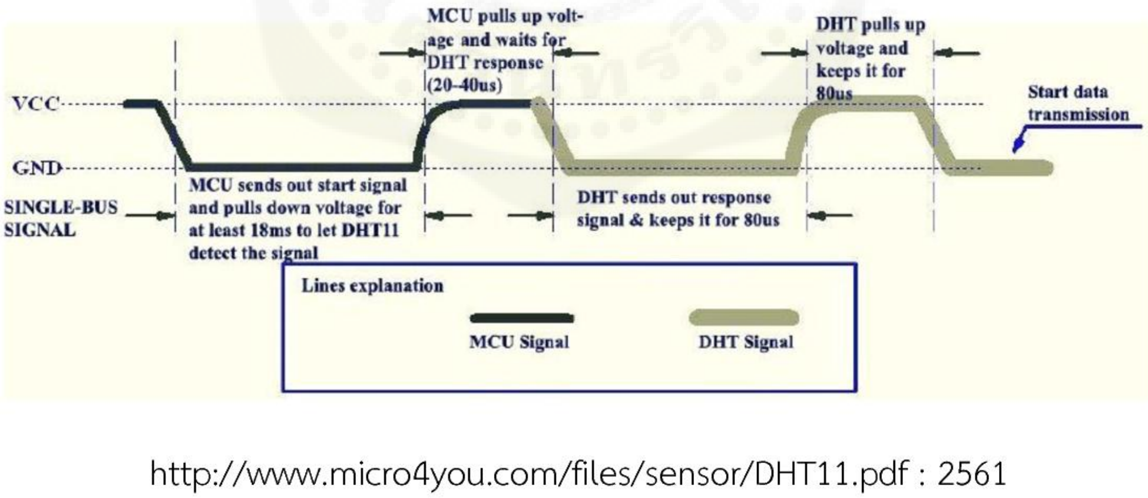

http://www.micro4you.com/files/sensor/DHT11.pdf : 2561

รูปที่ 7.2 การเชื่อมต่อกับ Microcontroller

ในการต่อใช้งานระหว่าง Sensor กับตัว Arduino จะต้องใช้ Pull up resistor ขนาดประมาณ 5k$\Omega$ โดยการต่อตัวต้านทานขนาด 5k$\Omega$ ไว้กับแหล่งจ่ายแรงดันและต่อเข้าไปที่ขา DATA ด้วย ซึ่งแต่ละขาของ DHT11 จะต่อตามรูปที่ 7.2 โดยใช้แหล่งจ่ายแรงดัน VDD จะมีขนาดแรงดัน 3-5.5 VDC

### 7.2.3 การส่งข้อมูลของ DHT11

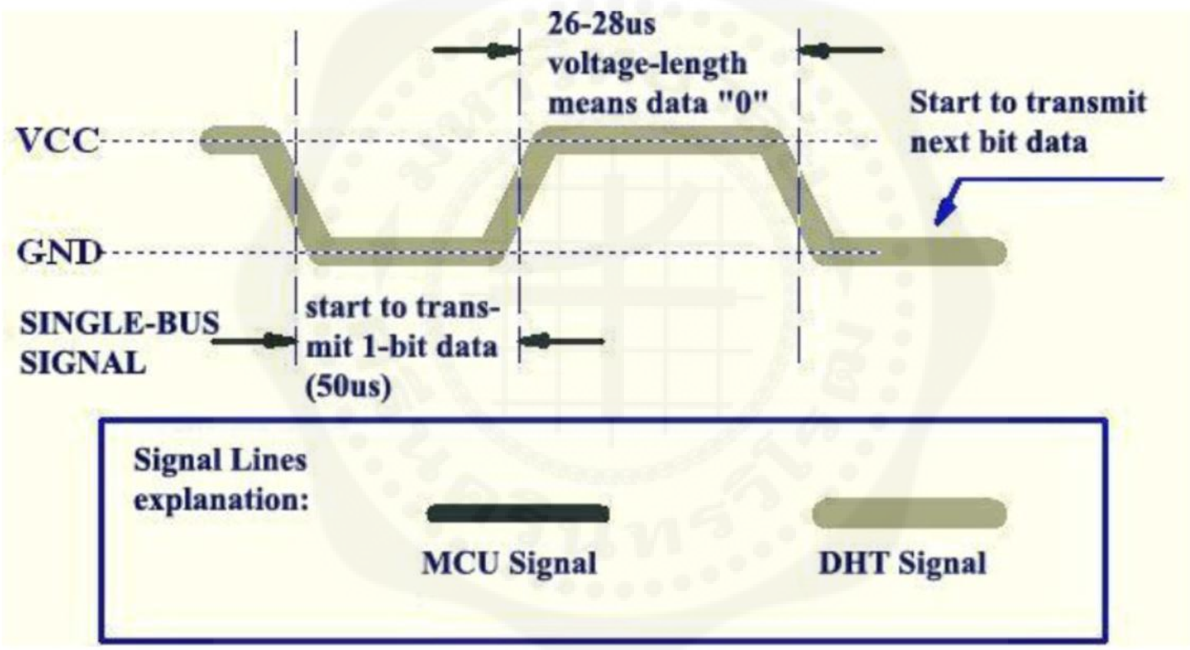
การส่งข้อมูลของ DHT11 ใช้การสื่อสารกับ MCU ด้วยวิธีการสื่อสารอนุกรมสองทางโดยใช้สายเส้นเดียว (Single-wire Two-way Serial interface) ในการสื่อสารโดยใช้สายเส้นเดียวนั้น อันดับแรก Arduino จะส่ง Start signal ด้วยแรงดันไฟฟ้าจากระดับสูงมาเป็นระดับต่ำอย่างน้อย 18µS ไปที่ DHT11 เพื่อให้ DHT11 รับทราบว่าจะเริ่มสั่งการแล้ว และรอต่อไปเป็นเวลาประมาณ 20-40µS เพื่อให้ DHT11ตอบกลับมา เมื่อ Arduino รับทราบว่า DHT11พร้อม DHT11 จะส่งแรงดันไฟฟ้าระดับต่ำกลับไปที่ Arduino การส่งแรงดันจาก DHT11 กลับไปจะใช้เวลา 80 µS จากนั้นจะรออีก 80µS ก่อนที่จะส่งข้อมูลบิต




http://www.micro4you.com/files/sensor/DHT11.pdf : 2561

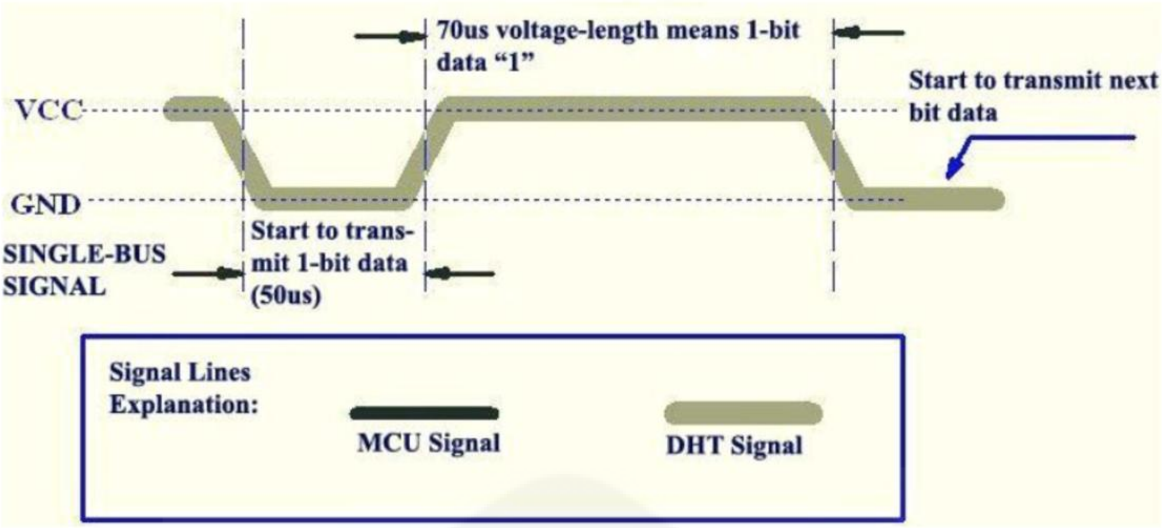
รูปที่ 7.3 สัญญาณการติดต่อช่วงเริ่มต้นระหว่าง Arduino กับ DHT11

การส่งข้อมูลเป็น "0" DHT11 จะปรับแรงดันลงต่ำนาน 50 ไมโครวินาที และปล่อยแรงดันเป็นระดับสูงนาน 26-28μS แต่ถ้าเป็นการส่งข้อมูลเป็น "1" DHT11จะปรับแรงดันลงต่ำนาน 50 μS และปล่อยแรงดันเป็นระดับสูงนาน 70 μS ในแต่ละชุดของข้อมูลที่ DHT11ส่งมา ตัว Arduino เมื่อรับข้อมูลแล้วจะต้องเอามาประมวลผลต่อว่าข้อมูลที่ส่งมานั้นมีค่าเป็นเท่าไหร่ โดยแต่ละชุดข้อมูลจะยาว 40 บิต และใช้เวลาส่งประมาณ 40mS ใน 40 บิตที่ส่งมาจะประกอบด้วย 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum



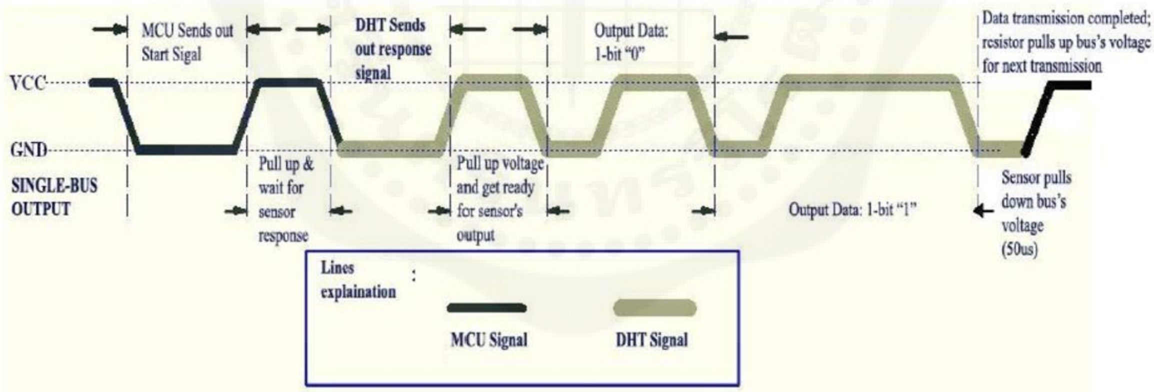

http://www.micro4you.com/files/sensor/DHT11.pdf : 2561

รูปที่ 7.4 การส่งข้อมูลเป็น “0”ระหว่าง Arduino กับ DHT11




http://www.micro4you.com/files/sensor/DHT11.pdf : 2561

รูปที่ 7.5 การส่งข้อมูลเป็น "1"ระหว่าง Arduino กับ DHT11

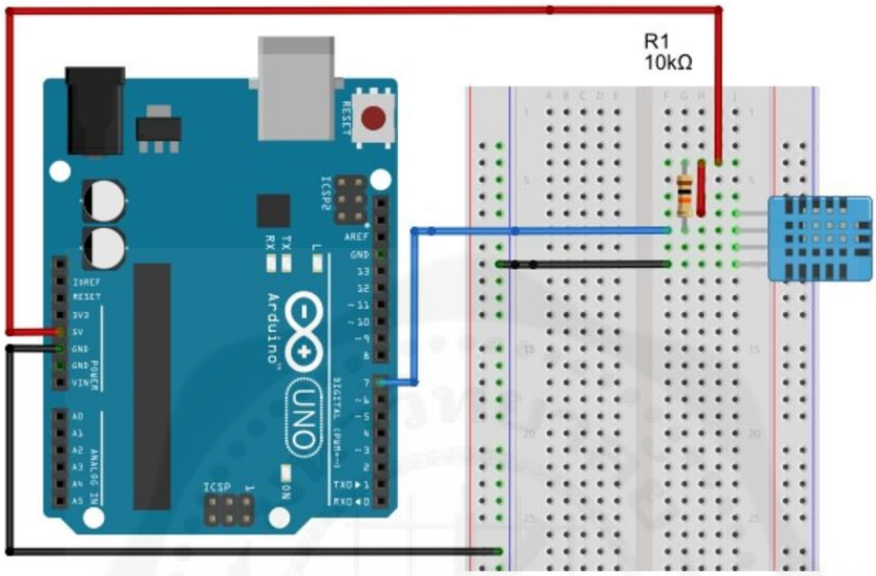



http://www.micro4you.com/files/sensor/DHT11.pdf : 2559

รูปที่ 7.6 การส่งชุดข้อมูลระหว่าง Arduino กับ DHT11

## 7.3 การเชื่อมต่อกับบอร์ด Arduino

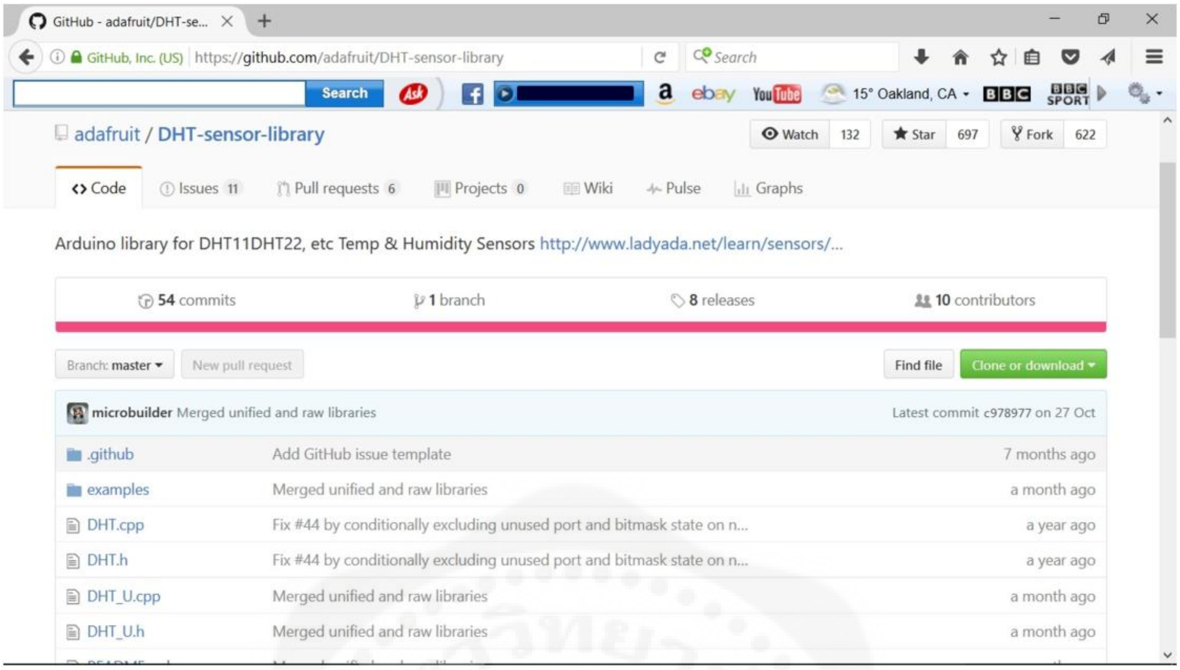
การเชื่อมต่อ DHT11 กับบอร์ด Arduino ทางขา Digital I/O ดังรูป




รูปที่ 7.7 การเชื่อมต่อ DHT11 กับบอร์ด Arduino

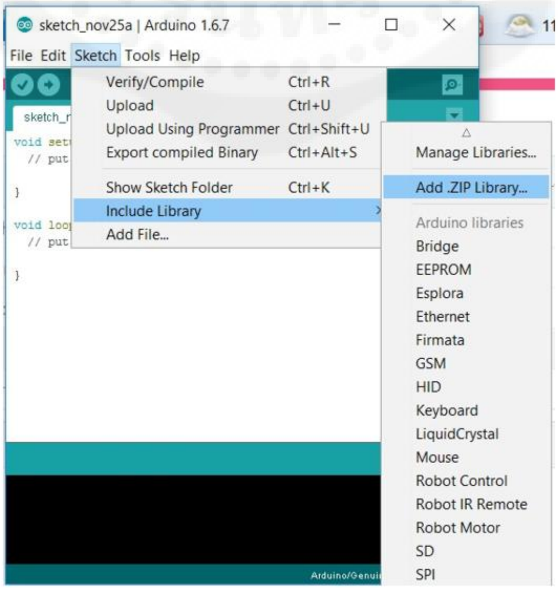
## 7.4 เริ่มใช้งาน Arduino IDE กับ DHT11

ก่อนอื่นต้องติดตั้งไลบรารี่เสริมเพื่อให้ DHT11 สามารถใช้งานร่วมกับบอร์ด Arduino ได้ โดยดาวน์โหลดไลบรารี่เสริมได้ที่ https://github.com/adafruit/DHT-sensor-library ดังรูปที่ 7.8 โดยดาวน์โหลดเป็นไฟล์ Zip



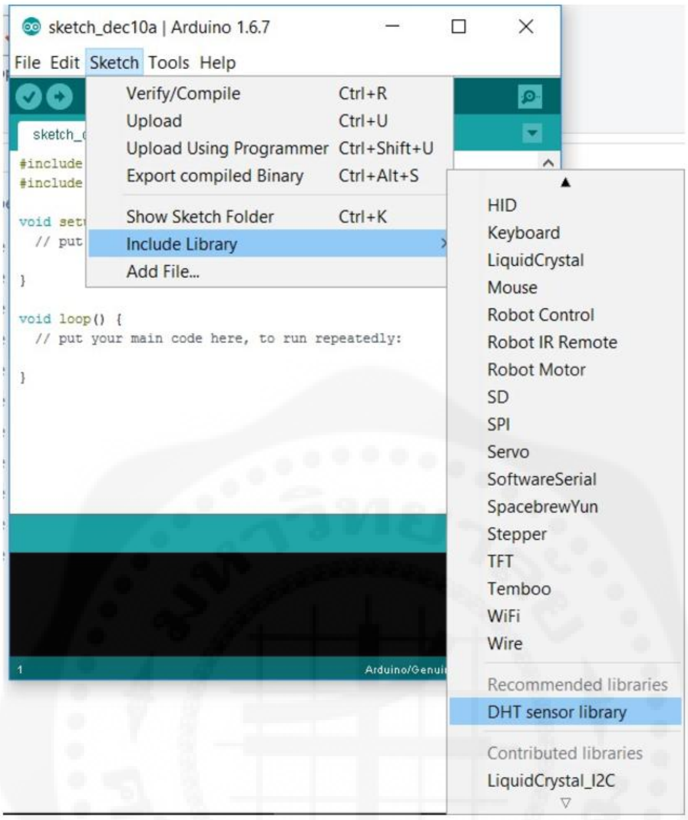

รูปที่ 7.8 เวปไซต์สำหรับดาวน์โหลดไลบรารี่เสริม

หลังจากนั้นให้ดำเนินการติดตั้งให้กับโปรแกรม Arduino IDE โดยกดไปที่ Sketch -> Include Library -> Add .Zip Library ดังรูปที่ 7.9 จากนั้นทำการหาตำแหน่งไฟล์ที่ดาวน์โหลดเก็บไว้ จากนั้นกด Open




รูปที่ 7.9 การติดตั้งไลบรารี่เสริม

เมื่อติดตั้งไลบรารี่เสริมสำเร็จ จะมีชื่อไลบรารี่เสริมในโปรแกรม Arduino IDE ดังรูปที่ 7.10




รูปที่ 7.10 การติดตั้งไลบรารี่เสริมสำเร็จ

ก่อนเขียนโปรแกรมรับค่าจากเซนเซอร์ DHT11 มารู้จักฟังก์ชั่นรับค่าจากเซนเซอร์ DHT11 เพื่อให้เราสามารถใช้งานได้ตามที่เราต้องการ

**float ตัวแปร = dht.readHumidity()** > รับค่าความชื้นค่าจากเซนเซอร์

**float ตัวแปร = dht.readTemperature()** > รับค่าอุณหภูมิจากเซนเซอร์เป็นองศาเซลเซียส

**float ตัวแปร = dht.readTemperature(true)** > รับค่าอุณหภูมิจากเซนเซอร์เป็นองศาฟาเรนไฮต์

**float ตัวแปร = dht.computeHeatIndex(f, h)** > อ่านค่าดัชนีความร้อนโดยให้ผลเป็นองศาฟาเรนไฮต์

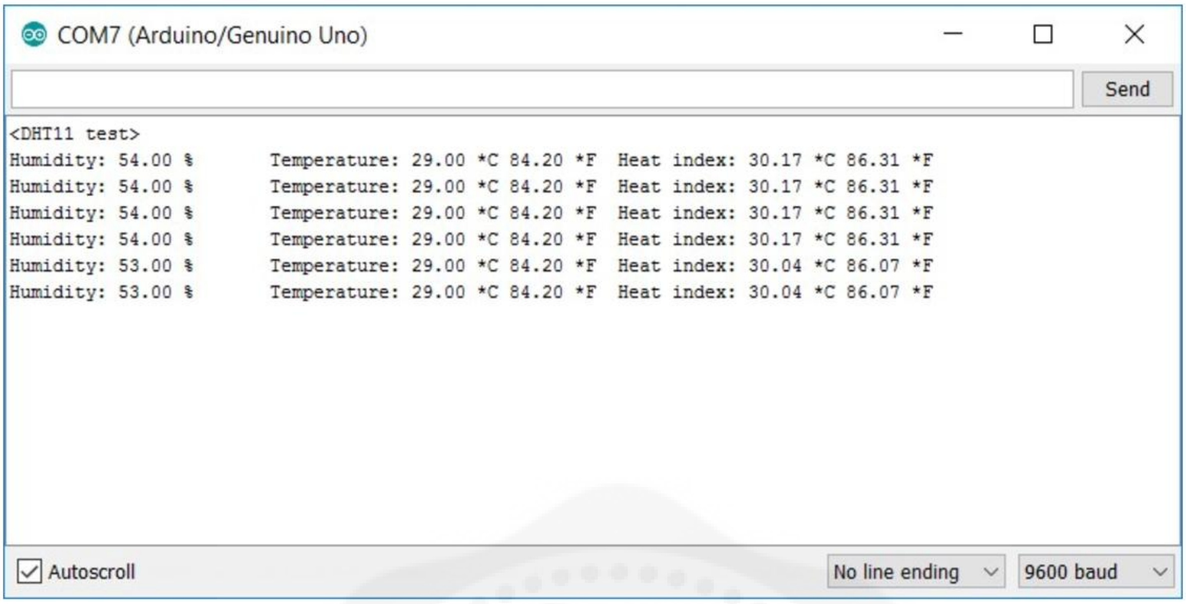
**float ตัวแปร = dht.computeHeatIndex(t, h, false)** > อ่านค่าดัชนีความร้อนโดยให้ผลเป็นองศาเซลเซียส

**ตัวอย่างที่ 1** การวัดความชื้น อุณหภูมิเป็นองศาเซลเซียสและองศาฟาเรนไฮต์ ดัชนีความร้อนโดยให้ผลเป็นองศาเซลเซียสและองศาฟาเรนไฮต์ โดยแสดงผลที่ Serial Monitor กำหนดให้ใช้ Digital I/O ขา 7 ของบอร์ด Arduino สำหรับต่อกับขา 2 ของ DTH11 ใช้วงจรตามรูปที่ 7.7

```
#include "DHT.h"

#define DHTPIN 7     // กำหนดให้ Pin 7 ของบอร์ด Arduino ต่อกับขา Data ของ DHT11

#define DHTTYPE DHT11  // กำหนดให้เซนเซอร์ตระกูล DHT เป็นเบอร์ DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

Serial.println("<DHT11 test>");

dht.begin();

}
```

```
void loop() {

// หน่วงเวลาระหว่างการอ่านค่า

delay(2000);

// อ่านค่าความชื้น

float h = dht.readHumidity();

// อ่านค่าอุณหภูมิเป็นองศาเซลเซียส

float t = dht.readTemperature();

// อ่านค่าอุณหภูมิเป็นองศาฟาเรนไฮต์

float f = dht.readTemperature(true);

// ตรวสอบการอ่านข้อมูล

if (isnan(h) || isnan(t) || isnan(f)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

// อ่านค่าดัชนีความร้อนโดยให้ผลเป็นองศาฟาเรนไฮต์

float hif = dht.computeHeatIndex(f, h);

// อ่านค่าดัชนีความร้อนโดยให้ผลเป็นองศาเซสเซียส
```

```
float hic = dht.computeHeatIndex(t, h, false);

Serial.print("Humidity: ");

Serial.print(h);

Serial.print(" %\t");

Serial.print("Temperature: ");

Serial.print(t);

Serial.print(" *C ");

Serial.print(f);

Serial.print(" *F\t");

Serial.print("Heat index: ");

Serial.print(hic);

Serial.print(" *C ");

Serial.print(hif);

Serial.println(" *F");

}
```

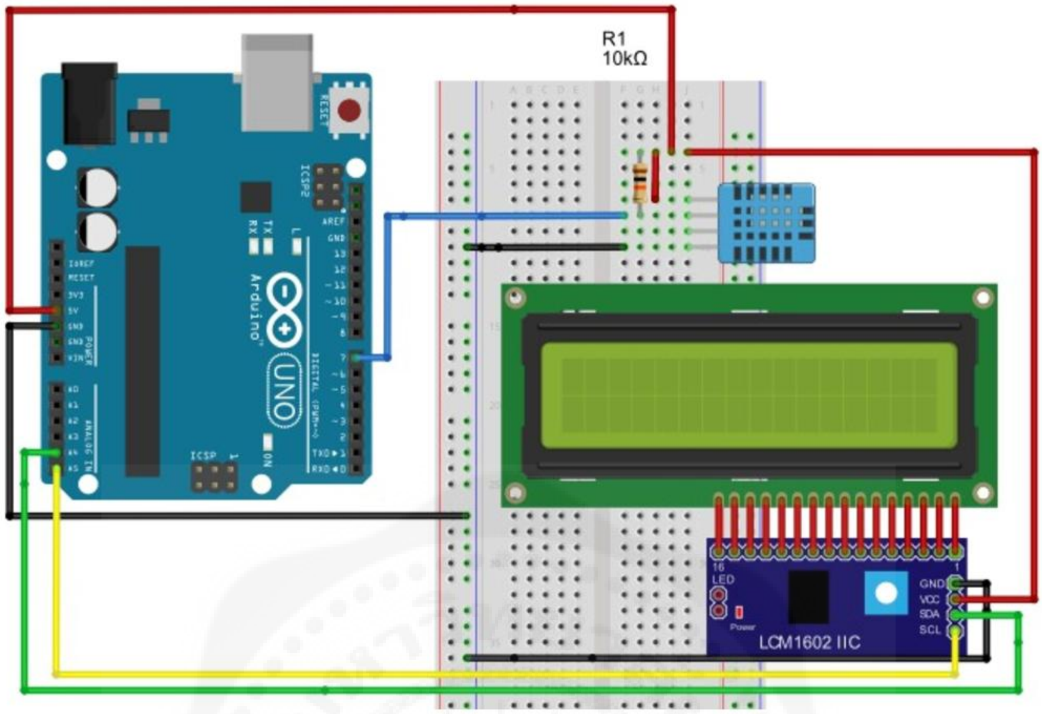



รูปที่ 7.11 ผลการทดลองตามตัวอย่างที่ 1.

**การทำงานของโปรแกรม**

โปรแกรมจะอ่านค่าความชื้นด้วยคำสั่ง float h = dht.readHumidity() อ่านค่าอุณหภูมิเป็นองศาเซลเซียสด้วยคำสั่ง float t = dht.readTemperature() อ่านค่าอุณหภูมิเป็นองศาฟาเรนไฮต์ด้วยคำสั่ง float f = dht.readTemperature(true) อ่านค่าดัชนีความร้อนโดยให้ผลเป็นองศาฟาเรนไฮต์ด้วยคำสั่ง float hif = dht.computeHeatIndex(f, h) อ่านค่าดัชนีความร้อนโดยให้ผลเป็นองศาเซสเซียสด้วยคำสั่งfloat hic = dht.computeHeatIndex(t, h, false) แสดงผลค่าที่อ่านได้ด้วยคำสั่ง Serial.print()

**ตัวอย่างที่ 2** การวัดความชื้น และอุณหภูมิเป็นองศาเซลเซียส โดยแสดงผลที่จอ LCD กำหนดให้ใช้ Digital I/O ขา 7 ของบอร์ด Arduino สำหรับต่อกับขา 2 ของ DHT11 และ LCD ใช้การเชื่อมต่อกับบอร์ด Arduno แบบ $I^2C$ ใช้วงจรตามรูปที่ 7.12

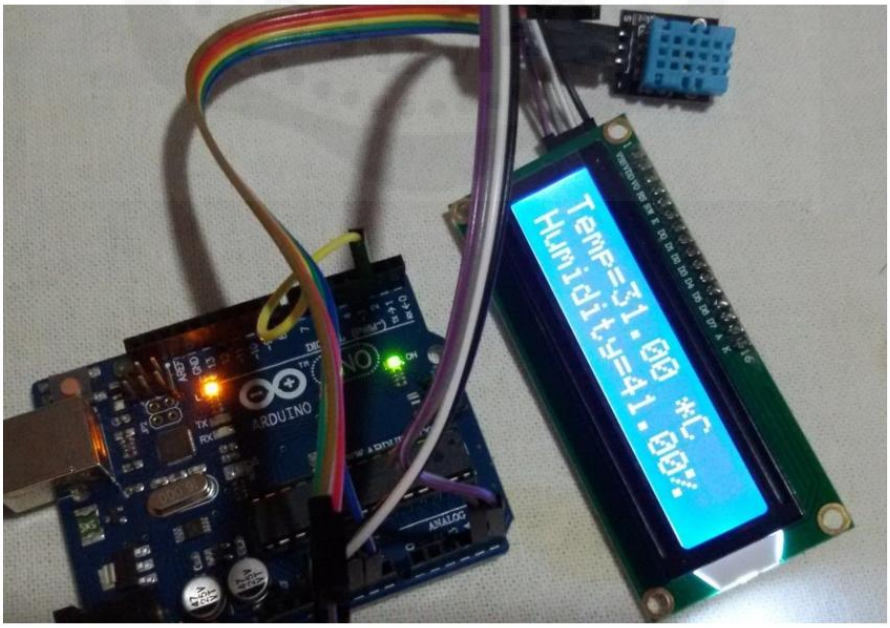



รูปที่ 7.12 การต่อวงจรสำหรับตัวอย่างที่ 2

```
#include "DHT.h"

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

#define DHTPIN 7

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

lcd.begin();
```

```
Serial.begin(9600);
dht.begin();
}
void loop() {
float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(t) || isnan(h)) {
lcd.clear();
lcd.print("Failed to read");
lcd.setCursor(0,1);
lcd.print("from DHT11");
delay(1000);
return;
}
else {
lcd.clear();
lcd.setCursor(0,0);
```

```
lcd.print("Temp=");

lcd.print(t);

lcd.print(" *C");

lcd.setCursor(0,1);

lcd.print("Humidity=");

lcd.print(h);

lcd.print("% ");

delay(1000);

}

}
```




รูปที่ 7.13 ผลการทดลองตามตัวอย่างที่ 2.

**การทำงานของโปรแกรม**

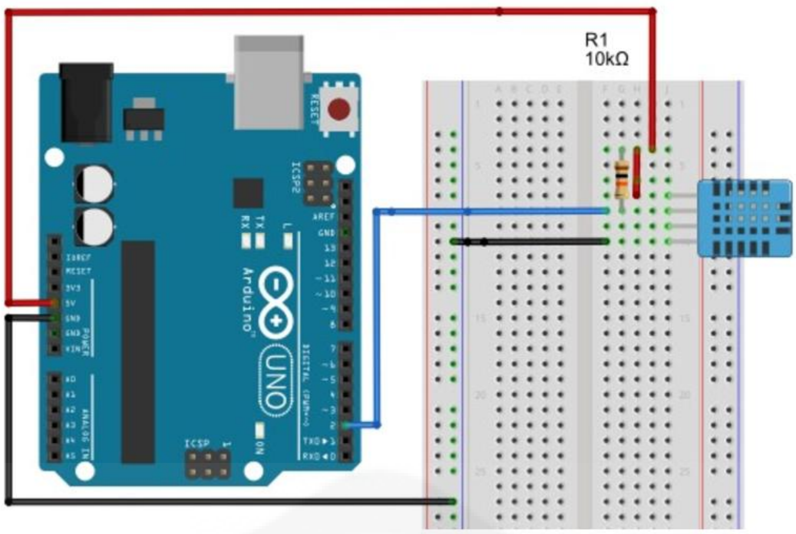
โปรแกรมจะอ่านค่าความชื้นด้วยคำสั่ง float h = dht.readHumidity() อ่านค่าอุณหภูมิเป็นองศาเซลเซียสด้วยคำสั่ง float t = dht.readTemperature() แสดงผลค่าที่อ่านได้ที่ LCD บรรทัดที่ 1ด้วยคำสั่ง lcd.setCursor(0,0) แสดงค่าอุณหภูมิด้วยคำสั่ง lcd.print(t) แสดงผลค่าที่อ่านได้ที่ LCD บรรทัดที่ 2 ด้วยคำสั่ง lcd.setCursor(0,1) แสดงค่าอุณหภูมิด้วยคำสั่ง lcd.print(h)

**สรุป**

การต่อใช้งานเซนเซอร์ DHT11 กับบอร์ด Arduino จะต้องติดตั้งไลบรารี่ก่อน เพื่อสามารถใช้งานร่วมกับบอร์ด Arduino ได้ โดยเซนเซอร์ทีใช้สภาพแวดล้อมในอากาศโดยสามาถวัดค่าอุณหภูมิ และความชื้นในอากาศ และให้เอาท์พุทออกมาในรูปของสัญญาณดิจิตอล

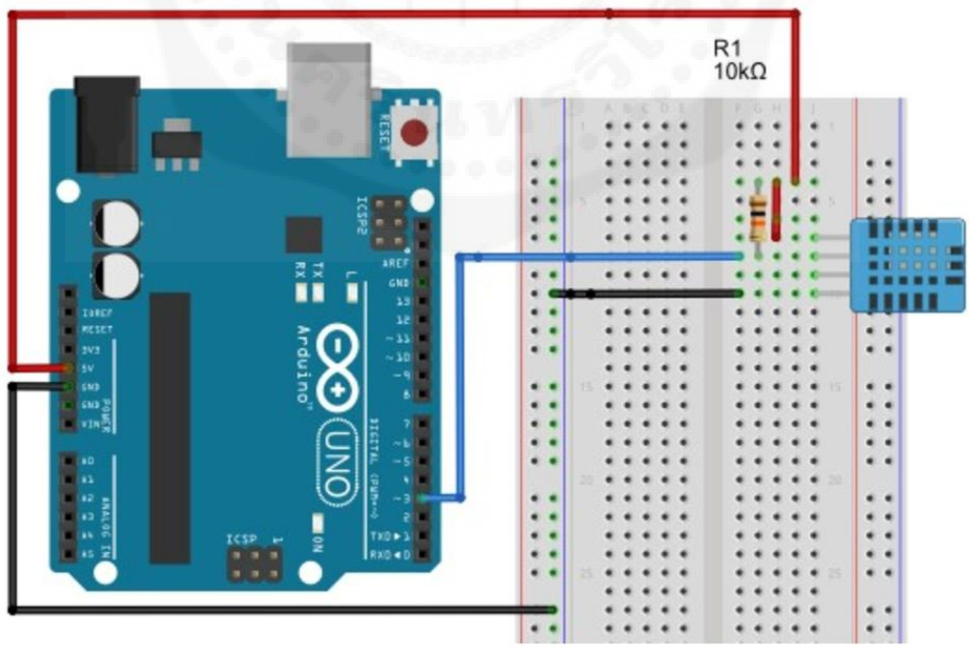
**คำถาม**

1. จงเขียนโปรแกรมเพื่อแสดงค่าความชื้น และอุณหูมิเป็นองศาฟาเรนไฮต์ ที่ Serial Monitor กำหนดให้ใช้ Digital I/O ขา 2 ของบอร์ด Arduino สำหรับต่อกับขา 2 ของ DHT11



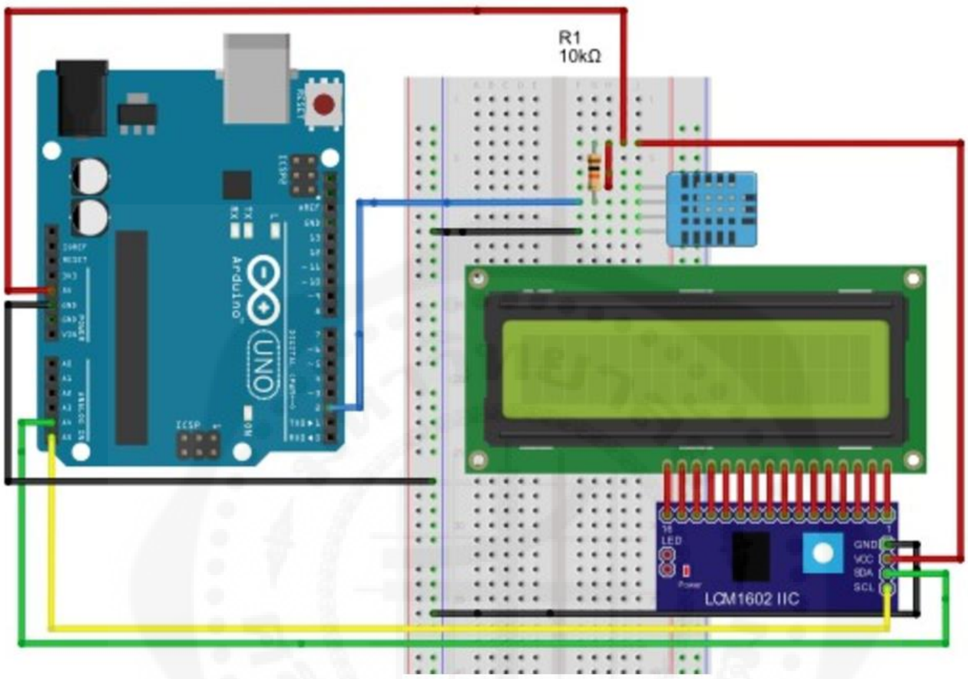

รูปที่ 7.14 การต่อวงจรสำหรับคำถามข้อ 1.

2. จงเขียนโปรแกรมเพื่อแสดงค่าอุณหภูมิเป็นองศาเซลเซียส และองศาฟาเรนไฮต์ ที่ Serial Monitor กำหนดให้ใช้ Digital I/O ขา 3 ของบอร์ด Arduino สำหรับต่อกับขา 2 ของ DHT11




รูปที่ 7.15 การต่อวงจรสำหรับคำถามข้อ 2.

3. จงเขียนโปรแกรมเพื่อแสดงค่าอุณหภูมิเป็นองศาเซลเซียส และองศาฟาเรนไฮต์ ที่จอ LCD กำหนดให้ใช้ Digital I/O ขา 2 ของบอร์ด Arduino สำหรับต่อกับขา 2 ของ DTH11 และ LCD ใช้การเชื่อมต่อกับบอร์ด Arduno แบบ $I^2C$




รูปที่ 7.16 การต่อวงจรสำหรับคำถามข้อ 3.

## เอกสารอ้างอิง

1. Mark Svaljek. (2015). Arduino Succinctly. Morrisville, NC : Syncfusion Inc.

2. http://playground.arduino.cc/main/DHT11Lib : 2561

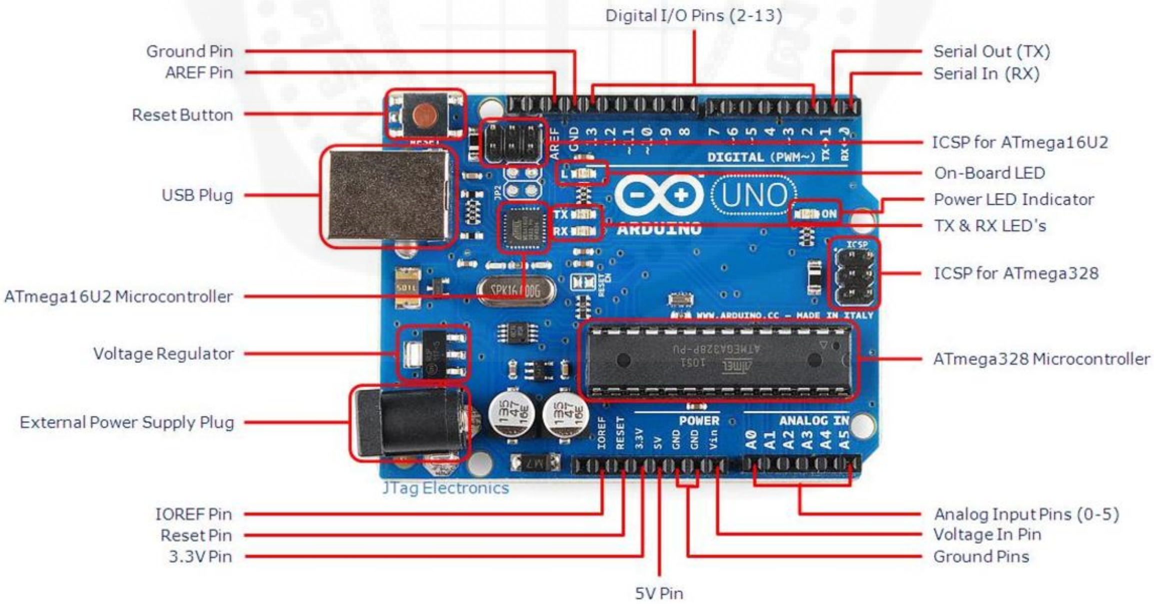
3. https://learn.adafruit.com/dht/using-a-dhtxx-sensor : 2561

# บทที่ 8

# การรับค่าอินพุทแบบอนาล็อก

## 8.1 อินพุทแบบอนาล็อก

การรับข้อมูลค่าแรงดันไฟฟ้าที่มีการเปลี่ยนแปลงแบบอนาล็อก คือค่าแรงดันมีการเพิ่มขึ้น หรือลดลงในแบบมีการเปลี่ยนแปลงค่าที่ไม่แน่นอนแต่อยู่ในช่วงที่กำหนด ซึ่งต่างจากค่าแรงดันที่เป็นแบบดิจิตอลซึ่งจะมีอยู่ 2 ระดับคือ 5V = ลอจิก 1 หรือ High และ 0V = ลอจิก 0 หรือ Low ซึ่งค่าแรงดันแบบอนาล็อกนี้จะไม่สามารถต่อเข้าที่ขา Digital I/O ของบอร์ด Arduino ได้ แต่ก็สามารถนำไปใช้งานกับบอร์ด Arduino โดยไม่ต้องแปลงค่าแรงดันดังกล่าวด้วยวงจร Analog to Digital เนื่องจากบอร์ด Arduino ได้ถูกออกแบบให้สามารถรับค่าแรงดันแบบอนาล็อกได้โดยตรง ซึ่งจะเรียกว่าขา Analog I/O ดังรูปที่ 8.1

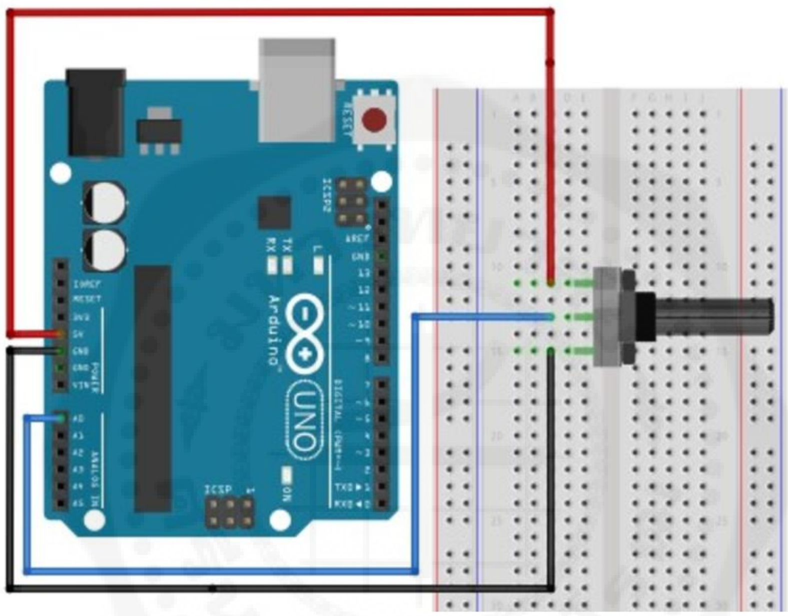



http://www.jtagelectronics.com/?p=75 : 2561

รูปที่ 8.1 ตำแหน่งขาต่าง ๆ ของบอร์ด Arduino UNO

จากรูปขา Analog I/O สำหรับบอร์ด Arduino UNO จะมีมาให้ทั้งหมด 6 ขา (A0 - A5) ซึ่งจะน้อยกว่าขา Digital I/O

### 8.2 การใช้ Analog I/O อ่านค่าการเปลี่ยนแปลง

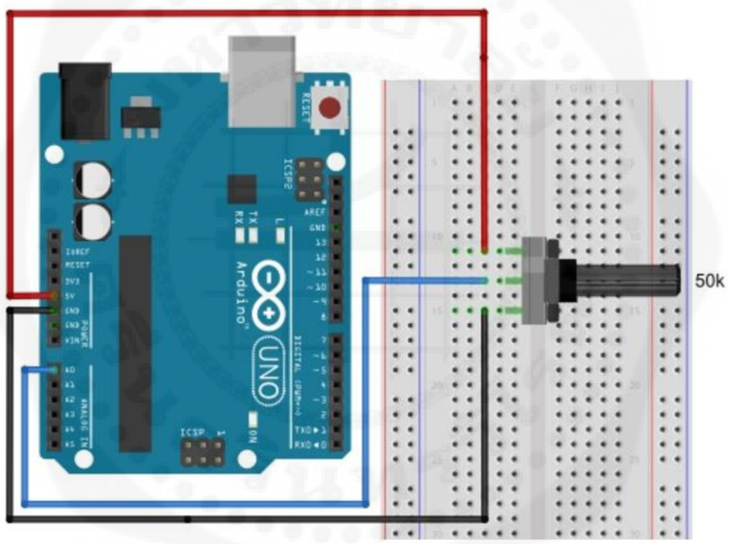


รูปที่ 8.2 การตัวต้านทานปรับค่าได้ป้อนค่าแรงดันให้ขา Analog I/O

จากรูปที่ 8.2 เป็นการต่อตัวต้านทานปรับค่าได้ในลักษณะวงจรแบ่งแรงดัน โดยขาริมด้านหนึ่งต่อกับแรงดันบวก 5V ขาริมอีกด้านต่อกับ GND ส่วนขากลางต่อเข้ากับขา Analog I/O ของบอร์ด Arduino เพื่อแบ่งแรงดันให้เป็นสัดส่วนตามการปรับค่าของตัวต้านทานปรับค่าได้ เพื่อให้บอร์ด Arduino ประมวลผลตามการเขียนโปรแกรม ซึ่งเราจะใช้ฟังก์ชั่น map ซึ่งเป็นฟังก์ชันที่ใช้เทียบอัตราส่วนตัวเลข ของค่าตัวเลข 2 ชุดที่แตกต่าง โดยค่าแรงดันอินพุทที่ป้อนเข้าบอร์ด Arduino ทางขา Analog I/O ซึ่งมีค่าแรงดันระหว่าง 0V ถึง 5V บอร์ด Arduino จะแปลงเป็นค่าตัวเลขระหว่าง 0 ถึง 1023 เนื่องจาก ADC ภายใน Arduino เป็น

ADC ขนาด 10-bit ดังนั้นถ้าเราต้องการให้แสดงผลเป็นค่าเท่าไร ก็จะใช้ฟังก์ชั่น map เป็นตัวช่วยในการเทียบอัตราส่วนเช่น map(val, 0, 1023, 0, 100) = แปลงค่า val จาก 0 ถึง 1023 เป็น 0 ถึง 100

**ตัวอย่างที่ 8.1** อ่านค่าแรงดันที่ขา Analog I/O ขา A0 ได้รับ โดยแสดงค่าที่ Serial Monitor เป็นเปอร์เซ็นต์ โดยถ้าอ่านได้สูงสุด 5V แสดงค่า 100% และต่ำสุด 0V แสดงค่า 0% ถ้าค่าแรงดันอื่นค่าเปอร์เซ็นต์ที่ได้ให้เป็นอัตราส่วนระหว่างค่าสูงสุดกับต่ำสุด




รูปที่ 8.3 วงจรสำหรับตัวอย่างที่ 1.

```
int inputPin = 0;

int read_value;

int value;

int previous_value;

void setup() {
```

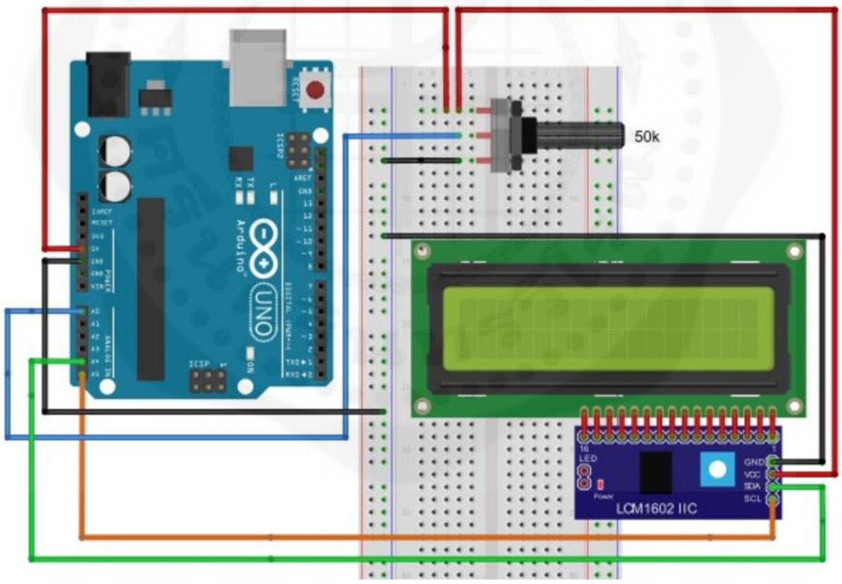
```
Serial.begin(9600);  }

void loop() {

read_value = analogRead(inputPin);

// ปรับค่าเป็นเปอร์เซนต์

value = map(read_value, 0, 1023, 0, 100);

if (value != previous_value) {

Serial.println(value);

previous_value = value;  }

delay(100); }
```


รูปที่ 8.4 ผลการทดลองวตัวอย่างที่ 1.

**การทำงานของโปรแกรม**

โปรแกรมจะอ่านค่าแรงดันด้วยคำสั่ง read_value = analogRead(inputPin) แล้วทำการแปลงข้อมูลจากอนาล็อกเป็นดิจิตอลด้วยคำสั่ง value = map(read_value, 0, 1023, 0, 100) แล้วจึงนำผลที่ได้ไปแสดงผล

**ตัวอย่างที่ 2.** อ่านค่าแรงดันที่ขา Analog I/O ขา A0 ได้รับ โดยแสดงค่าที่ LCD โดยค่าแรงดันอ่านได้สูงสุด 5V แสดงค่า 5V และต่ำสุด 0V แสดงค่า 0V ถ้าค่าแรงดันอื่น ค่าที่อ่านได้ให้มีทศนิยม 2 ตำแหน่ง




รูปที่ 8.5 วงจรสำหรับตัวอย่างที่ 2.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

int inputPin = 0;
```

```
float read_value;
float value;
float previous_value;
void setup() {
lcd.begin();
Serial.begin(9600);  }
void loop() {
read_value = analogRead(inputPin);
value = (read_value*5)/1023 ;
if (value != previous_value) {
// แสดงผล และเก็บค่าการเปรียบเทียบ
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Voltage = ");
lcd.print(value);
lcd.print(" V");
previous_value = value; }
```

delay(1000); }


รูปที่ 8.5 ผลการทดลองตัวอย่างที่ 2.

**การทำงานของโปรแกรม**

โปรแกรมจะอ่านค่าแรงดันด้วยคำสั่ง read_value = analogRead(inputPin) แล้วทำการแปลงข้อมูลจากอนาล็อกเป็นดิจิตอลด้วยคำสั่ง value = (read_value*5)/1023 แล้วจึงนำผลที่ได้ไปแสดงผลที่ LCD

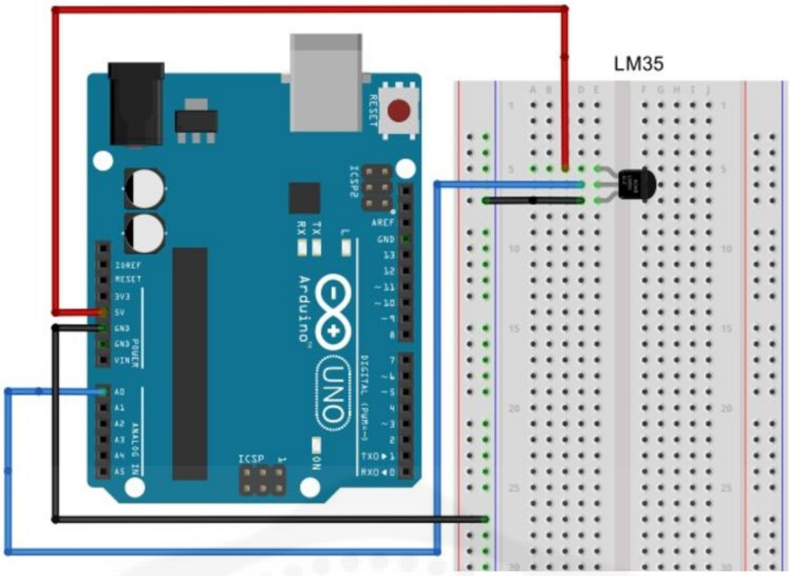
## 8.3 การวัดอุณหภูมิด้วยเซ็นเซอร์ LM35


- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates from 4 V to 30 V
- Less than 60-μA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

https://www.indiamart.com/proddetail/lm35-lm35dz-precision-temperature-sensor-14510861388.html/ : 2561

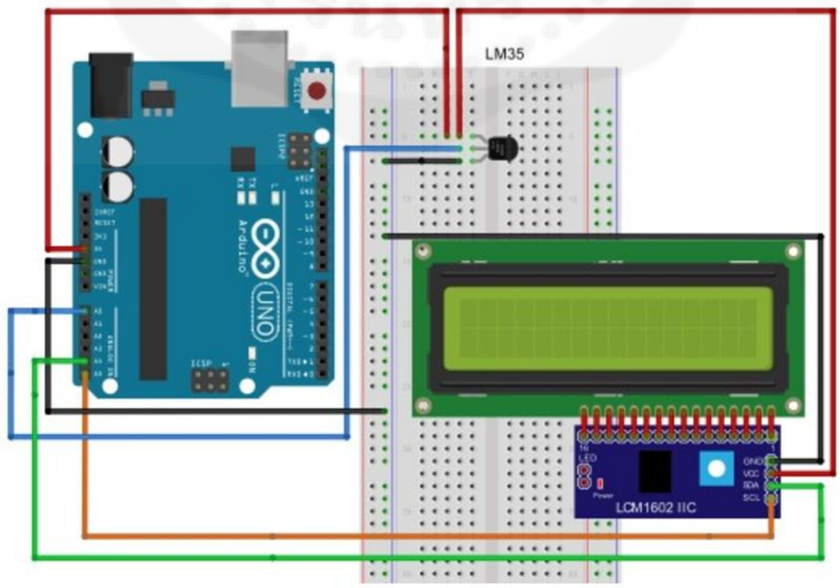
รูปที่ 8.6 เซ็นเซอร์ LM35

เซนเซอร์ LM35 เป็นเซ็นเซอร์วัดอุณหภูมิที่ให้ข้อมูลเอาท์พุทแบบอนาล็อกโดยจะให้เป็นค่าแรงดันตามอุณหภูมิที่เปลี่ยนแปลง เมื่ออุณหภูมิเพิ่มขึ้น 1 องศาแรงดันที่ขาเอาท์พุท LM35 จะเพิ่ม 10mV โดยมีค่าความแม่นยำอยู่ที่ $\pm 2^{O}$C นั้นคือหากตรวจจับอุณหภูมิที่ 35 องศาเซลเซียส แรงดันที่ขาเอาท์พุทของ LM35 จะมีค่า 350 mV ดังนั้นถ้านำมาต่อใช้งานกับบอร์ด Arduino จึงต้องต่อใช้งานกับขา Analog I/O ดังรูปที่ 8.7




รูปที่ 8.7 การต่อเซ็นเซอร์ LM35 กับ บอร์ด Arduino

**ตัวอย่างที่ 3.** ใช้เซ็นเซอร์ LM35 วัดอุณหภูมิ โดยค่าแรงดันเอาท์พุทต่อที่ขา Analog I/O ขา A0 ของบอร์ด Arduino โดยแสดงค่าอุณหภูมิที่วัดได้ที่ LCD ค่าอุณหภูมิที่วัดได้ให้มีทศนิยม 2 ตำแหน่ง

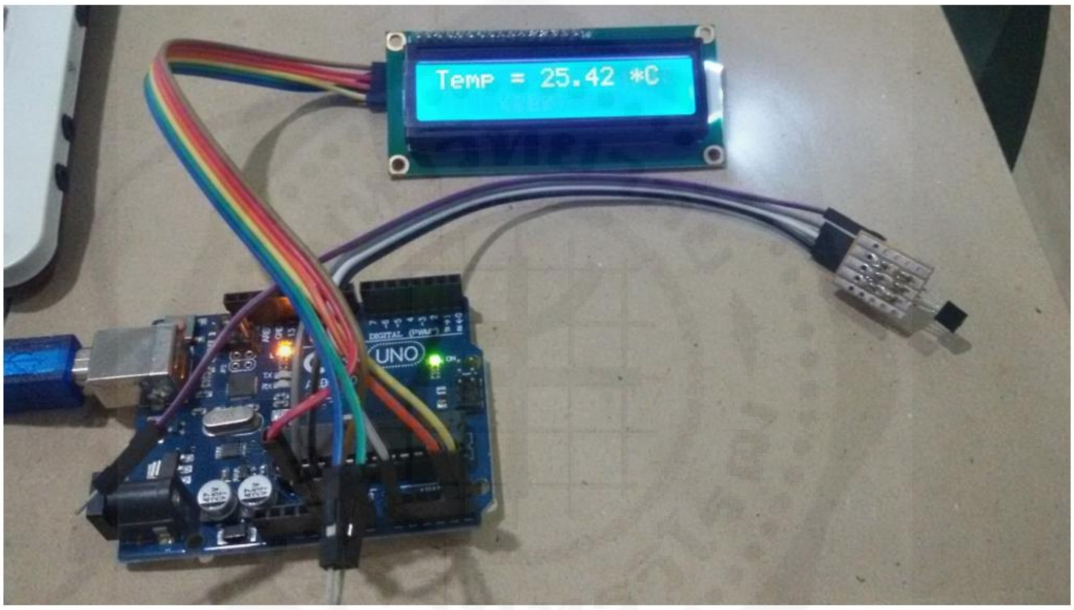



รูปที่ 8.8 วงจรสำหรับตัวอย่างที่ 3.

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
int inputPin = 0;
float read_Temp;
float Temp;
void setup() {
lcd.begin();
Serial.begin(9600);  }
void loop() {
read_Temp = analogRead(inputPin);
Temp = (read_Temp*5*100)/1023 ;
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Temp = ");
lcd.print(Temp);
lcd.print(" *C");
delay(500);  }
```

**การทำงานของโปรแกรม**

โปรแกรมจะอ่านค่าอุณหภูมิด้วยคำสั่ง read_Temp = analogRead(inputPin) แล้วทำการแปลงข้อมูลจากอนาล็อกเป็นดิจิตอลด้วยคำสั่ง Temp = (read_value*5*100)/1023 แล้วจึงนำผลที่ได้ไปแสดงผลที่ LCD




รูปที่ 8.9 ผลการทดลองตัวอย่างที่ 3.

## 8.4 การวัดความชื้นสัมพัทธ์ด้วยเซ็นเซอร์ HIH4030


**FEATURES**

- Tape and reel packaging allows for use in high volume pick and place manufacturing (1,000 units per tape and reel)
- Molded thermoset plastic housing
- Near linear voltage output vs %RH
- Laser trimmed interchangeability
- Low power design
- Enhanced accuracy
- Fast response time
- Stable, low drift performance
- Chemically resistant

https://www.sparkfun.com/products/9569 : 2561

รูปที่ 8.10 เซ็นเซอร์ HIH4030

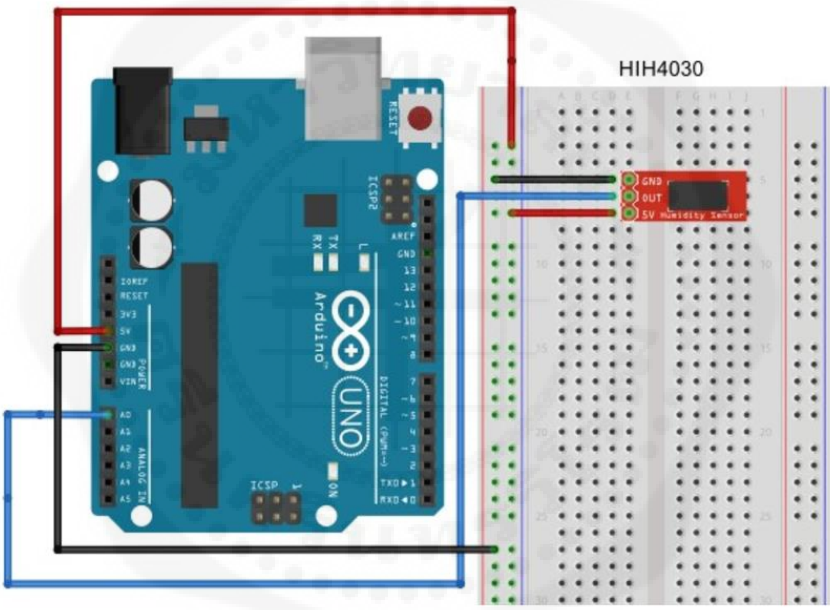
เซนเซอร์ HIH4030 เป็นเซ็นเซอร์วัดความชื้นสัมพัทธ์ (%RH) ที่ให้ข้อมูลเอาท์พุทแบบอนาล็อกในรูปแรงดันไฟตรงที่แปรผันตรงกับค่าความชื้นสัมพัทธ์ ซึ่งค่าแรงดันที่ได้จะใกล้เคียงเชิงเส้นมาก โดยค่าความชื้นในช่วง 0-100%RH จะแปลงเป็นแรงดันไฟตรงในช่วง 0.958V ถึง 4.065V และมีค่าความผิดพลาดอยู่ที่ ±3.5%RH เมื่อไฟเลี้ยงเซ็นเซอร์เท่ากับ 5V และ

อุณหภูมิห้อง 25$^{O}$C การนำไปใช้งานจะต้องนำค่าแรงดันที่วัดได้เป็นคำนวณหาค่าความชื้นสัมพัทธ์ในอากาศ อีกทีหนึ่งจากสมการ

$$\%RH = (Voltage - 0.958) / 0.0307 : T=25^{O}C$$

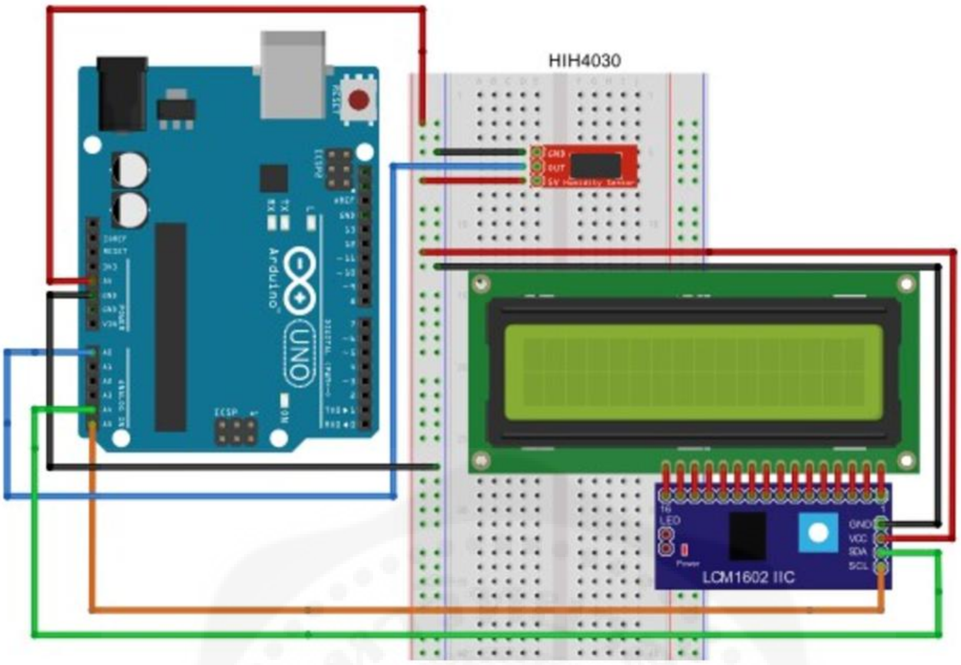
และ $\%True\ RH = (\%\ RH)/(1.0546 - 0.00216T)\ \ T\ in\ ^{O}C$

ดังนั้นถ้านำมาต่อใช้งานกับบอร์ด Arduino จึงต้องต่อใช้งานกับขา Analog I/O ดังรูปที่ 8.11




รูปที่ 8.11 การต่อเซ็นเซอร์ HIH4030 กับ บอร์ด Arduino

**ตัวอย่างที่ 4.** ใช้เซ็นเซอร์ HIH4030 วัดความชื้นสัมพัทธ์ โดยค่าแรงดันเอาท์พุทต่อที่ขา Analog I/O ขา A0 ของบอร์ด Arduino โดยแสดงค่าอุณหภูมิที่วัดได้ที่ LCD



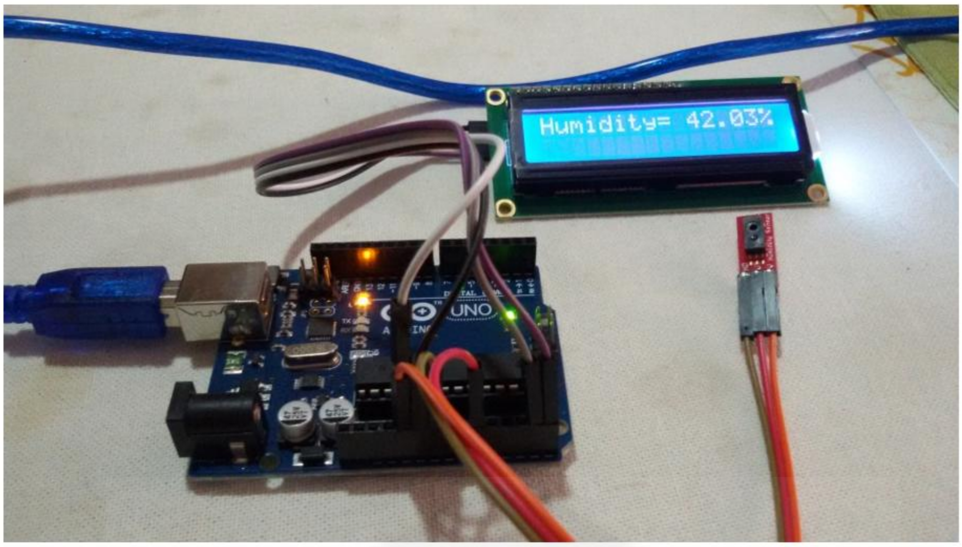

รูปที่ 8.12 วงจรสำหรับตัวอย่างที่ 4.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

int inputPin = 0;

float read_RH;

float RH;

void setup() {

lcd.begin();

Serial.begin(9600); }

void loop() {
```

```
read_RH = analogRead(inputPin);

RH = (((read_RH*5)/1023)-0.958)/0.0307 ;

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Humidity= ");

lcd.print(RH);

lcd.print("%");

delay(500);

}
```

**การทำงานของโปรแกรม**

โปรแกรมจะอ่านค่าแรงดันด้วยคำสั่ง read_value = analogRead(inputPin) แล้วทำการแปลงข้อมูลจากอนาล็อกเป็นดิจิตอลด้วยคำสั่ง RH = (((read_RH*5)/1023)-0.958)/0.0307 แล้วจึงนำผลที่ได้ไปแสดงผลที่ LCD



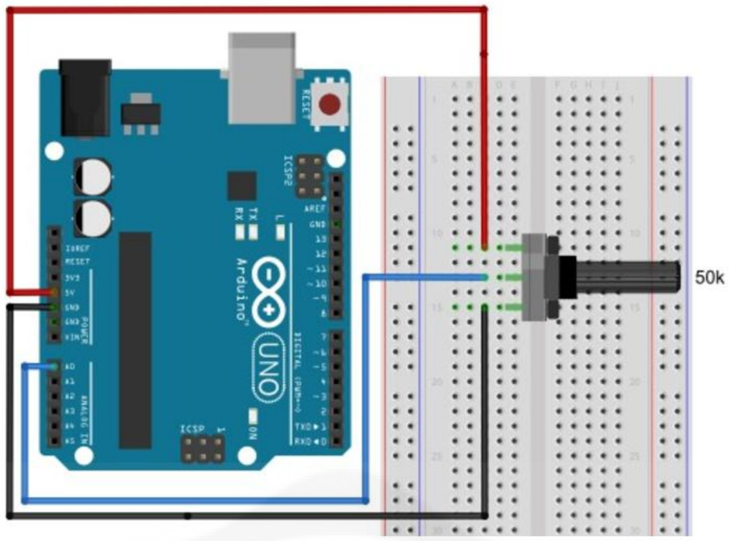

รูปที่ 8.13 ผลการทดลองตัวอย่างที่ 4.

## สรุป

บอร์ด Arduino การรับข้อมูลค่าแรงดันไฟฟ้าที่มีการเปลี่ยนแปลงแบบอนาล็อกได้ ซึ่งค่าแรงดันแบบอนาล็อกนี้จะไม่สามารถต่อเข้าที่ขา Digital I/O ของบอร์ด Arduino ได้ แต่ก็สามารถนำไปใช้งานกับบอร์ด Arduino ได้โดยไม่ต้องแปลงค่าแรงดันดังกล่าวด้วยวงจร Analog to Digital เนื่องจากบอร์ด Arduino ได้ถูกออกแบบให้มีวงจร Analog to Digital อยู่ภายใน ทำให้สามารถรับค่าแรงดันแบบอนาล็อกได้โดยตรง ซึ่งจะเรียกว่าขา Analog I/O

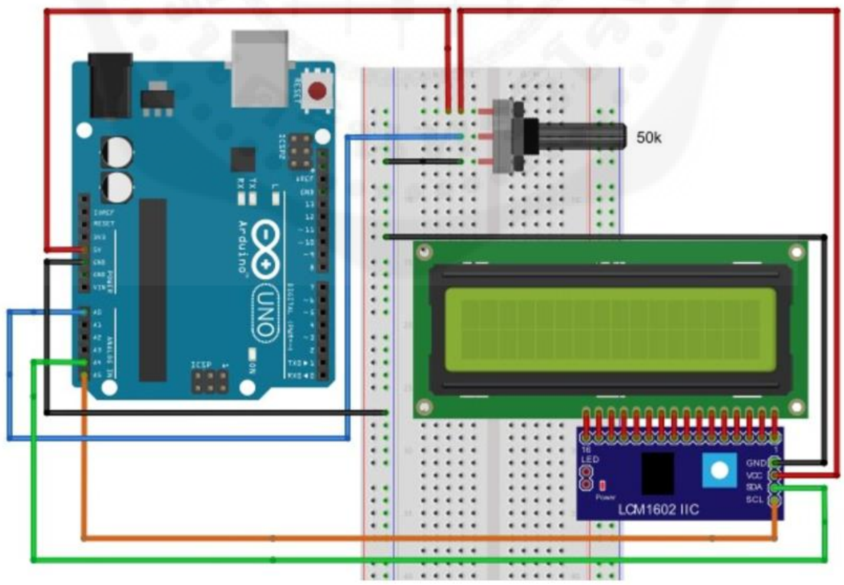
## คำถาม

1. อ่านค่าแรงดันที่ขา Analog I/O ขา A0 ได้รับ โดยแสดงค่าที่ Serial Monitor เป็นค่าแรงดันที่อ่านได้ โดยถ้าอ่านได้สูงสุด 5V แสดงค่า 5V และต่ำสุด 0V แสดงค่า 0V ถ้าค่าแรงดันอื่นค่าที่อ่านได้ให้มีทศนิยม 2 ตำแหน่ง



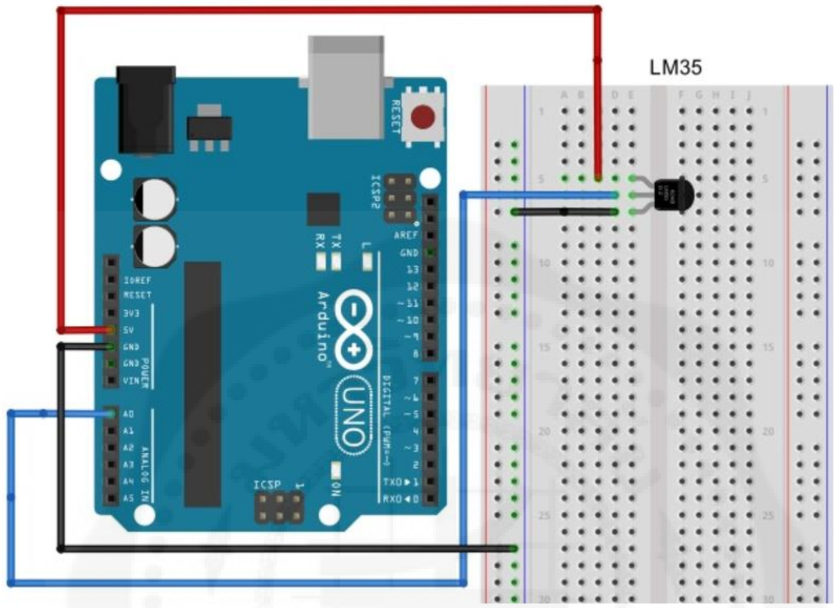

รูปที่ 8.14 การต่อวงจรสำหรับคำถามข้อ 1

2. อ่านค่าแรงดันที่ขา Analog I/O ขา A0 ได้รับ โดยแสดงค่าที่ LCD เป็นเปอร์เซ็นต์ โดยถ้าอ่านได้สูงสุด 5V แสดงค่า 100% และต่ำสุด 0V แสดงค่า 0% ถ้าค่าแรงดันอื่นค่าเปอร์เซ็นต์ที่ได้ให้เป็นอัตราส่วนระหว่างค่าสูงสุดกับต่ำสุด



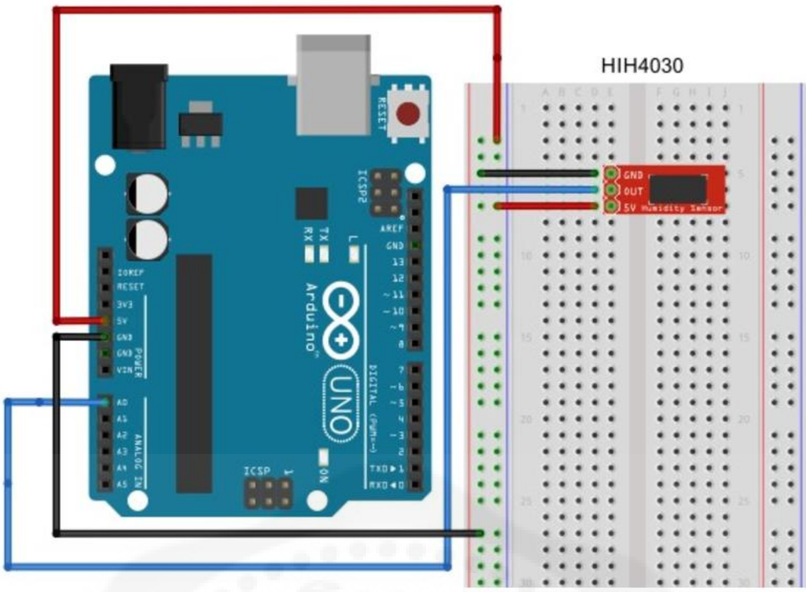

รูปที่ 8.15 การต่อวงจรสำหรับคำถามข้อ 2

3. ใช้เซ็นเซอร์ LM35 วัดอุณหภูมิ โดยค่าแรงดันเอาท์พุทต่อที่ขา Analog I/O ขา A0 ของบอร์ด Arduino โดยแสดงค่าอุณหภูมิที่วัดได้ที่ Serial Monitor ค่าอุณหภูมิที่วัดได้ให้มีทศนิยม 2 ตำแหน่ง




รูปที่ 8.16 การต่อวงจรสำหรับคำถามข้อ 3

4. ใช้เซ็นเซอร์ HIH4030 วัดความชื้นสัมพัทธ์ โดยค่าแรงดันเอาท์พุทต่อที่ขา Analog I/O ขา A0 ของบอร์ด Arduino โดยแสดงค่าอุณหภูมิที่วัดได้ที่ Serial Monitor




รูปที่ 8.17 การต่อวงจรสำหรับคำถามข้อ 4

## เอกสารอ้างอิง

1. Mark Svaljek. (2015). Arduino Succinctly. Morrisville, NC : Syncfusion Inc.

2. http://www.ti.com/lit/ds/symlink/lm35.pdf : 2561

3. http://playground.arduino.cc/Main/LM35HigherResolution : 2561

4. http://bildr.org/2012/11/hih4030-arduino/ : 2561

5. http://bradsduino.blogspot.com/2012/10/using-hih-4030-with-arduino-uno-to.html : 2561

6. https://www.sparkfun.com/datasheets/Sensors/Weather/SEN-09569-HIH-4030-datasheet.pdf : 2561

# บทที่ 9

# Real Time Clock

## 9.1 ไอซี DS1302

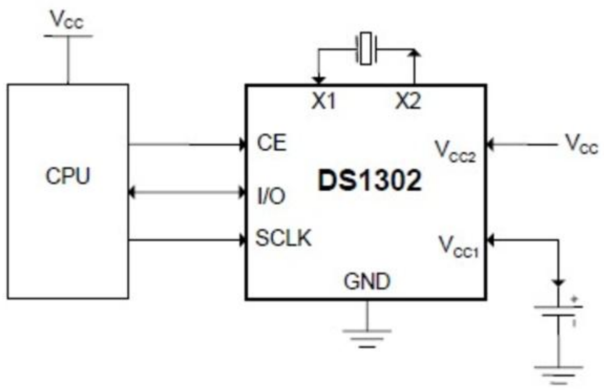

http://www.digole.com/index.php?productID=344 : 2560

รูปที่ 9.1 ไอซี ds1302

ไอซี DS1302 เป็นไอซีทำงานเป็นนาฬิกาให้ข้อมูลบอกเวลา วัน/เดือน/ปี ตามจริงโดยมีแบตเตอรี่ขนาดเล็กต่อสำรองไว้ เพื่อจ่ายแรงดันเลี้ยงตัวไอซีในกรณีที่หยุดจ่ายแรงดันให้วงจรหลักทำงาน ไอซี DS1302 ก็ยังทำงานเป็นนาฬิกาบอกเวลา วัน/เดือน/ปี อยู่ได้ นิยมนำไปใช้งานกับวงจรที่ต้องใช้ฐานเวลาที่เป็นปัจจุบันในการบันทึกข้อมูล เช่น อุปกรณ์จำพวก Data logger ไอซี DS1302 ใช้การสื่อสารข้อมูลแบบ 3 สาย (3-wire interface) ที่ประกอบด้วยขารับ-ส่งข้อมูล ขาสัญญาณนาฬิกา และขาเลือกให้ไอซีทำงาน แต่การสื่อสารข้อมูลแบบ 3 สายดังกล่าวไม่ใช้ตามมาตรฐาน $I^2C$ หรือ SPI

## 9.2 การต่อใช้งาน ไอซี DS1302

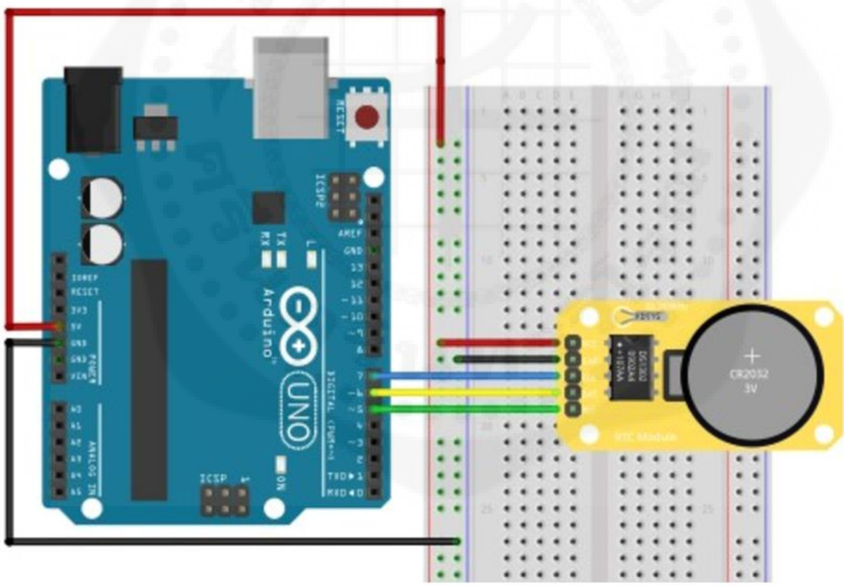
การต่อใช้งาน ไอซี DS1302 จะต้องมีการต่ออุปกรณ์ภายนอกร่วมด้วยถึงจะสามารถใช้งานได้ดังรูปที่ 9.2 ซึ่งจะต้องมี crystal ต่อที่ขา X1, X2 และแบตเตอรี่สำรองต่อที่ขา Vcc1 ด้วย




http://datasheets.maximintegrated.com/en/ds/DS1302.pdf : 2560

รูปที่ 9.2 การต่อใช้งานไอซี DS1302

การเชื่อมต่อไอซี DS1302 กับบอร์ด Arduino จะเชื่อมที่ขา Digital I/O ดังรูป



รูปที่ 9.2 การต่อใช้งานไอซี DS1302 กับบอร์ด Arduino

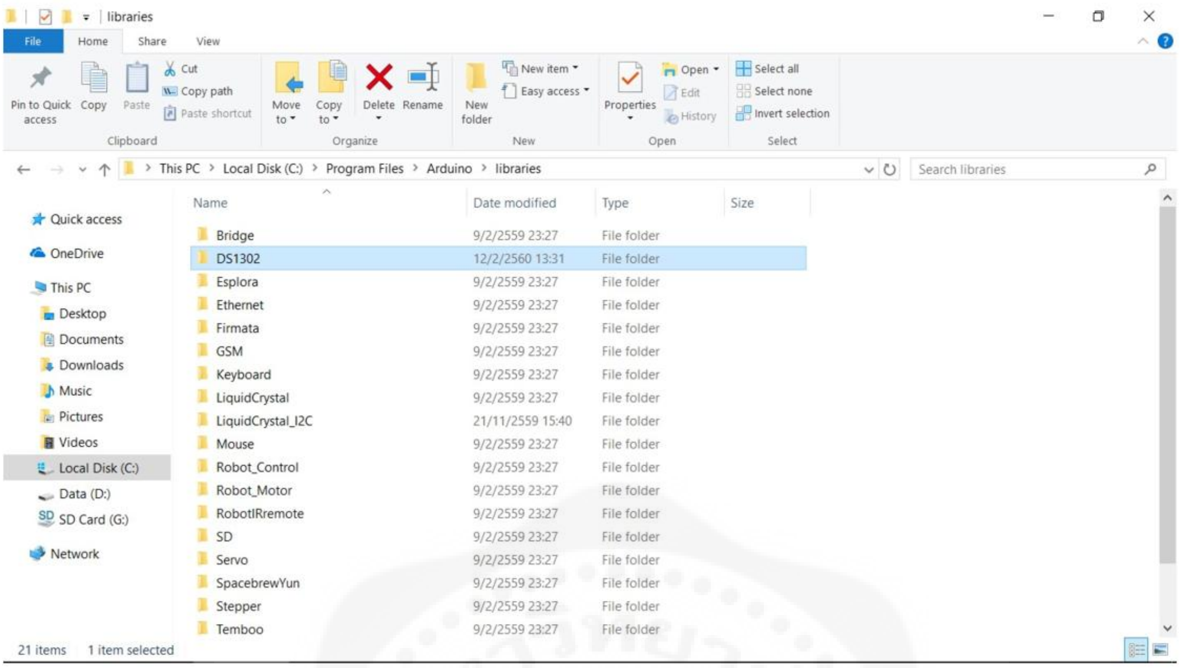
## 9.3 เริ่มใช้งาน Arduino IDE กับ DS1302

ก่อนอื่นต้องติดตั้งไลบรารี่เสริมเพื่อให้ DS1302 สามารถใช้งานร่วมกับบอร์ด Arduino ได้ โดยดาวน์โหลดไลบรารี่เสริมได้ที่

http://145.130.102.57/domoticx/arduino/libraries/ds13xx_ds32xx//DS1302 %20v2.2.7z ดังรูปที่ 9.4

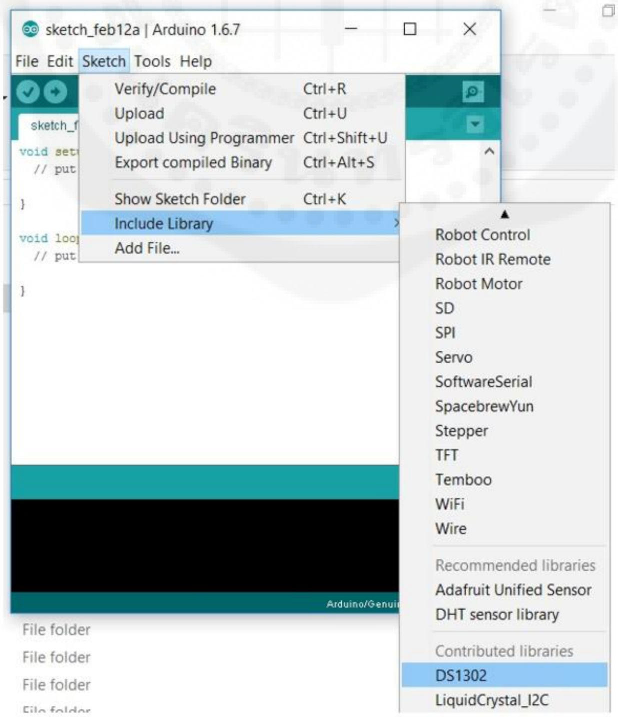

รูปที่ 9.4 เวปไซด์สำหรับดาวน์โหลดไลบรารี่เสริม

เนื่องจากไลบรารี่ไฟล์จะเป็นไฟล์บีบอัดนามสกุล .7z เมื่อดาว์นโหลดมาแล้วจะไม่สามารถติดตั้งในโปรแกรม Arduino IDE ได้โดยตรง จะต้องแตกไฟล์ดังกล่าวไว้ที่ไดร์ของคอมพิวเตอร์เสียก่อน หลังจากนั้นจึงจะดำเนินการติดตั้งไลบรารี่ให้กับโปรแกรม Arduino IDE โดยให้ก๊อบปี้โฟลเดอร์ที่แตกไฟล์ไลบรารี่ไว้ทั้งโฟลเดอร์ไปที่ C:\Programfiles\Arduino\Libraries\ ดังรูปที่ 9.5




รูปที่ 9.5 การติดตั้งไลบรารี่เสริม

เมื่อติดตั้งไลบรารี่เสริมสำเร็จ จะมีชื่อไลบรารี่เสริมในโปรแกรม Arduino IDE ดังรูปที่ 8.6



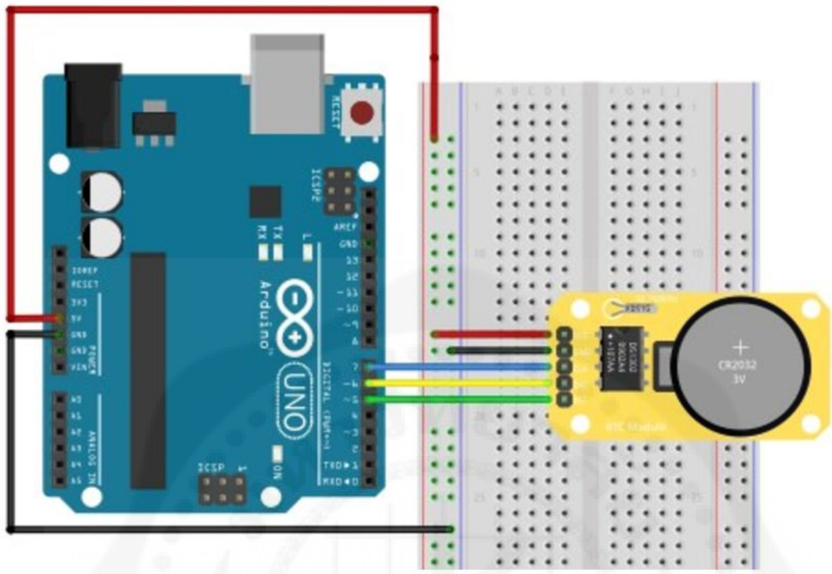

รูปที่ 9.6 การติดตั้งไลบรารี่เสริมสำเร็จ

ก่อนเขียนโปรแกรมรับค่าจากไอซี DS1302 มารู้จักฟังก์ชั่นที่จำเป็นสำหรับติดต่อกับไอซี DS1302 เพื่อให้เราสามารถใช้งานได้ตามที่เราต้องการ

| | |
|---|---|
| **rtc.halt(false);** | > กำหนด rtc ทำงาน |
| **rtc.writeProtect(false);** | > กำหนดให้สามารถเขียนข้อมูล ที่ rtc |
| **rtc.setDOW(FRIDAY);** | > ตั้งค่าวัน |
| **rtc.setDate(dd, mm, yyyy);** | > ตั้งต่า ว/ด/ป |
| **rtc.setTime(h, m, s);** | > ตั้งค่าเวลาแบบ 24 ชั่วโมง |
| **rtc.getDOWStr()** | > แสดงวัน |
| **rtc.getDateStr()** | > แสดง ว/ด/ป |
| **rtc.getTimeStr()** | > แสดงเวลา |
| **rtc.getMonthStr()** | > แสดงเดือน |

ในการใช้งานครั้งแรกเราจะต้องตั้งค่าเวลาให้กับไอซี DS1302 ก่อนเพียงครั้งเดียว ไอซี DS1302 ก็จะจำค่าไว้ได้ ดังตัวอย่างที่ 1 ซึ่งจะเป็นโปรแกรมการตั้งค่าเวลาให้กับไอซี DS1302

**ตัวอย่างที่ 1** เขียนโปรแกรมการตั้งค่าเวลาให้กับไอซี DS1302 กำหนดให้ใช้ Digital I/O ขา 5, 6 และ 7 ของบอร์ด Arduino สำหรับต่อกับขา CE, I/O และ CLOCK ของ DS1302



รูปที่ 9.7 วงจรสำหรับตัวอย่างที่ 1.

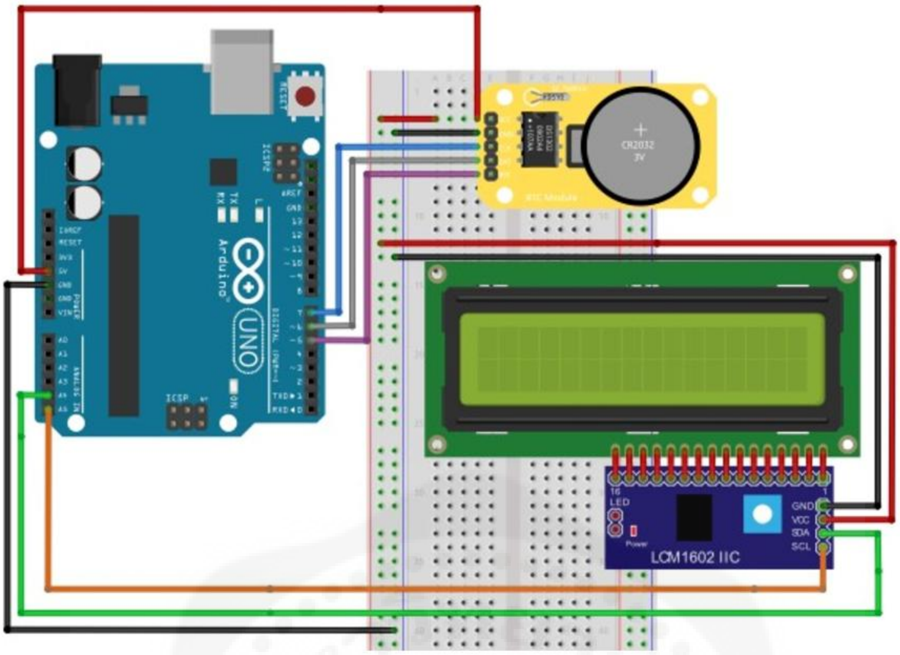
```
#include <DS1302.h>

//กำหนดขาต่อกับ DS1302 rtc([CE/RST], [I/O], [CLOCK]);

DS1302 rtc(5, 6, 7);

void setup() {

//กำหนดการทำงานในโหมด run, และยกเลิกการป้องกันการเขียนข้อมูล

rtc.halt(false);

rtc.writeProtect(false);
```

```
Serial.begin(9600);
//กำหนดค่าเวลา และ ว/ด/ป หากเริ่มใช้งาน DS1302 ครั้งแรก
rtc.setDOW(FRIDAY);        // ตั้งค่าวัน
rtc.setTime(12, 30, 0);     // ตั้งค่าเวลาแบบ 24 ชั่วโมง
rtc.setDate(12, 2, 2017);   // ตั้งค่า ว/ด/ป
}
void loop() {
}
```

**การทำงานของโปรแกรม**

โปรแกรมจะกำหนดค่าให้ DS1302 ด้วยคำสั่ง rtc.setDOW(FRIDAY) เพื่อตั้งค่าวัน คำสั่ง rtc.setTime(12, 30, 0); เพื่อตั้งค่าเวลา และคำสั่ง rtc.setDate(12, 2, 2017); เพื่อตั้งค่า ว/ด/ป

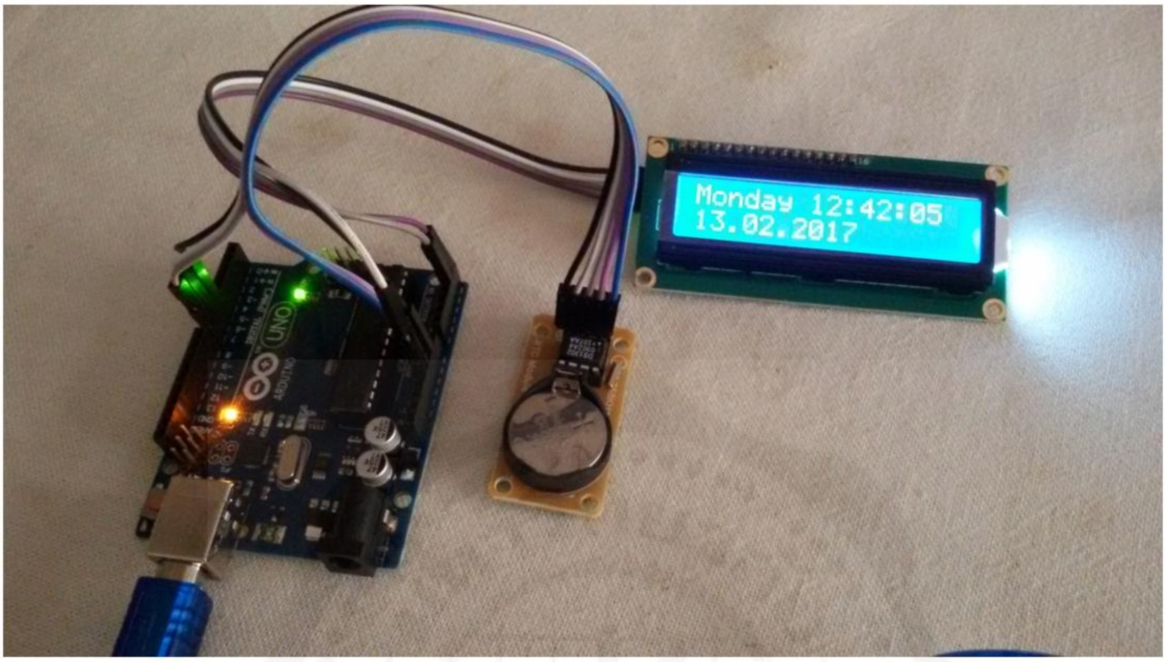
**ตัวอย่างที่ 2** เขียนโปรแกรมแสดงค่าวัน, ว/ด/ป และเวลาเวลาจากไอซี DS1302 กำหนดให้ใช้ Digital I/O ขา 5, 6 และ 7 ของบอร์ด Arduino สำหรับต่อกับขา CE, I/O และ CLOCK ของ DS1302 และ LCD ใช้การเชื่อมต่อกับบอร์ด Arduno แบบ $I^2C$ ใช้วงจรตามรูปที่ 9.8




รูปที่ 9.8 วงจรสำหรับตัวอย่างที่ 2.

```
#include <DS1302.h>

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

//กำหนดขาต่อกับ DS1302 rtc([CE/RST], [I/O], [CLOCK]);

DS1302 rtc(5, 6, 7);

void setup() {

//กำหนดการทำงานในโหมด run, และยกเลิกการป้องกันการเขียนข้อมูล

rtc.halt(false);
```

```
rtc.writeProtect(false);
lcd.begin();
Serial.begin(9600);
}
void loop() {
//แสดงค่าวัน
lcd.clear();
lcd.setCursor(0,0);
lcd.print(rtc.getDOWStr());
lcd.print(" ");
//แสดงค่าเวลา
lcd.print(rtc.getTimeStr());
lcd.setCursor(0,1);
//แสดงค่า ว/ด/ป
lcd.print(rtc.getDateStr());
delay (1000);
}
```



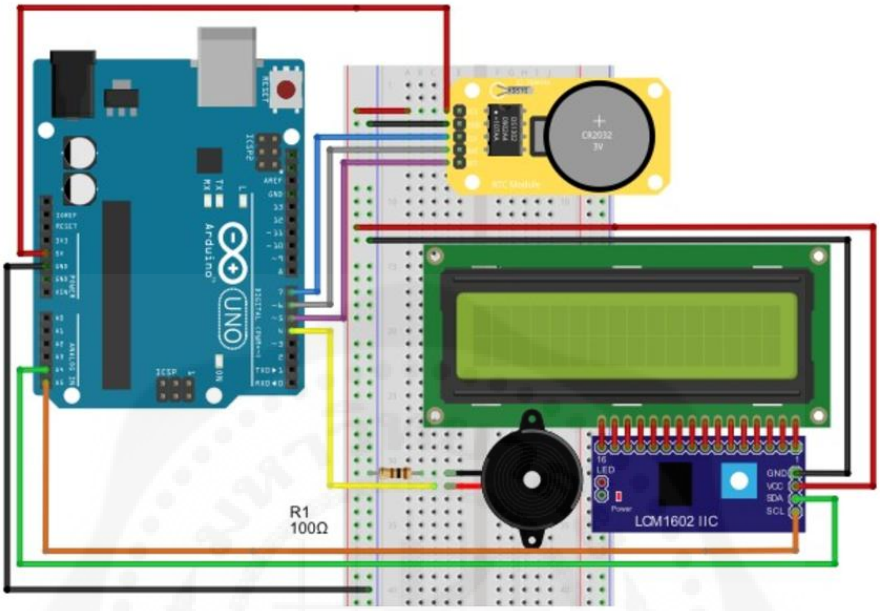

รูปที่ 9.9 ผลการทดลองตามตัวอย่างที่ 2.

**การทำงานของโปรแกรม**

โปรแกรมจะกำหนดให้แสดงผลที่ LCD บรรทัดที่ 1 ด้วยคำสั่ง lcd.setCursor(0,0) โดยแสดงค่าวันด้วยคำสั่ง lcd.print(rtc.getDOWStr()) แสดงค่าเวลาด้วยคำสั่ง lcd.print(rtc.getTimeStr()) กำหนดให้แสดงผลที่ LCD บรรทัดที่ 2 ด้วยคำสั่ง lcd.setCursor(0,1) แสดงค่า ว/ด/ป ด้วยคำสั่ง lcd.print(rtc.getDateStr())

**ตัวอย่างที่ 3** เขียนโปรแกรมแสดงค่าเวลา และ ว/ด/ป จากไอซี DS1302 โดยให้ตั้งค่าเวลาให้บัซเซอร์ดังทุก ๆ 1 นาที กำหนดให้ใช้ Digital I/O ขา 5, 6 และ 7 ของบอร์ด Arduino สำหรับต่อกับขา CE, I/O และ CLOCK ของ ds1302 กำหนดให้ขาขั้วบวกของบัซเซอร์ต่อเข้า

ที่ขา Digital I/O ขา 4 และ LCD ใช้การเชื่อมต่อกับบอร์ด Arduno แบบ $I^2C$ ใช้วงจรตามรูปที่ 9.10




รูปที่ 9.10 วงจรสำหรับตัวอย่างที่ 3.

```
#include <DS1302.h>

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

//กำหนดขาต่อกับ DS1302 rtc([CE/RST], [I/O], [CLOCK]);

DS1302 rtc(5, 6, 7);

Time t;

int buzzPin = 4;

void setup() {
```

```
//กำหนดการทำงานในโหมด run, และยกเลิกการป้องกันการเขียนข้อมูล
rtc.halt(false);
rtc.writeProtect(false);
lcd.begin();
Serial.begin(9600);
pinMode(buzzPin, OUTPUT);
}
void loop() {
t=rtc.getTime();
//แสดงค่าเวลา
lcd.clear();
lcd.setCursor(0,0);
lcd.print(rtc.getTimeStr());
//แสดงค่า ว/ด/ป
lcd.setCursor(0,1);
lcd.print(rtc.getDateStr());
if (t.sec == 00)
```

```
{
tone(buzzPin, 1400, 1000);
}
delay (1000);
}
```

**การทำงานของโปรแกรม**

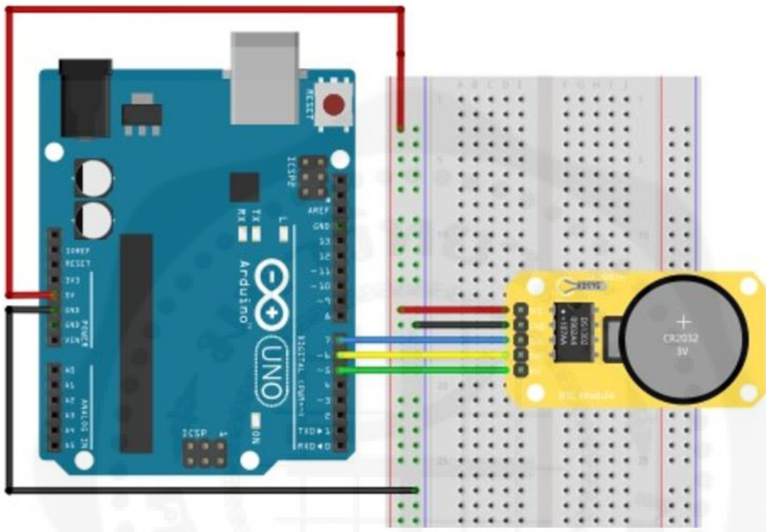
โปรแกรมจะกำหนดให้แสดงค่าที่ LCD บรรทัดที่ 1 ด้วยคำสั่ง lcd.setCursor(0,0) โดยแสดงค่าวันด้วยคำสั่ง lcd.print(rtc.getTimeStr() กำหนดให้แสดงค่าที่ LCD บรรทัดที่ 2 ด้วยคำสั่ง lcd.setCursor(0,1) แสดงค่า ว/ด/ป ด้วยคำสั่ง lcd.print(rtc.getDateStr() อ่านค่าเวลาเพื่อนำมาใช้เปรียบเทียบด้วยคำสั่ง t=rtc.getTime() และเปรียบเทียบให้เตือนทุก ๆ นาที ด้วยคำสั่ง if (t.sec == 00) โดยเสียงเตือนจากบัซเซอร์ใช้คำสั่ง tone(buzzPin, 1400, 1000);

**สรุป**

ไอซี DS1302 เป็นไอซีทำงานเป็นนาฬิกาให้ข้อมูลบอกเวลา วัน/เดือน/ปี ตามจริง การนำไปใช้งานจะต้องต่ออุปกรณ์ภายนอกร่วมด้วยถึงจะสามารถใช้งานได้ และจะต้องต่อแบตเตอรี่ขนาดเล็กเพื่อจ่ายไฟเลี้ยงไอซีไว้ และเมื่อนำมาใช้งานกับบอร์อด Arduino จะต้องติดตั้งไลบรารี่เพิ่มก่อน เพื่อให้ ds1302 สามารถใช้งานร่วมกับบอร์ด Arduino ได้

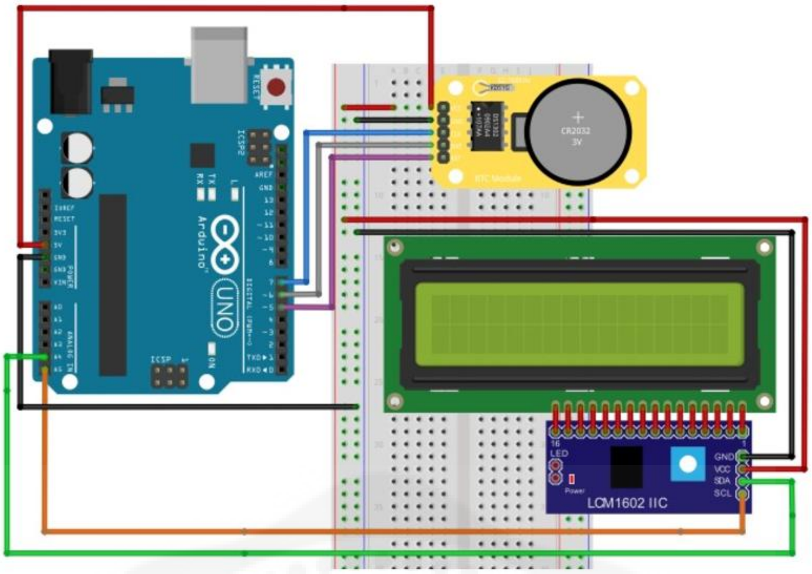
**คำถาม**

1. เขียนโปรแกรมแสดงค่าวัน, ว/ด/ป และเวลาจากไอซี ds1302 กำหนดให้ใช้ Digital I/O ขา 5, 6 และ 7 ของบอร์ด Arduino สำหรับต่อกับขา CE, I/O และ CLOCK ของ ds1302 โดยแสดงค่าเวลาที่ Serial Monitor



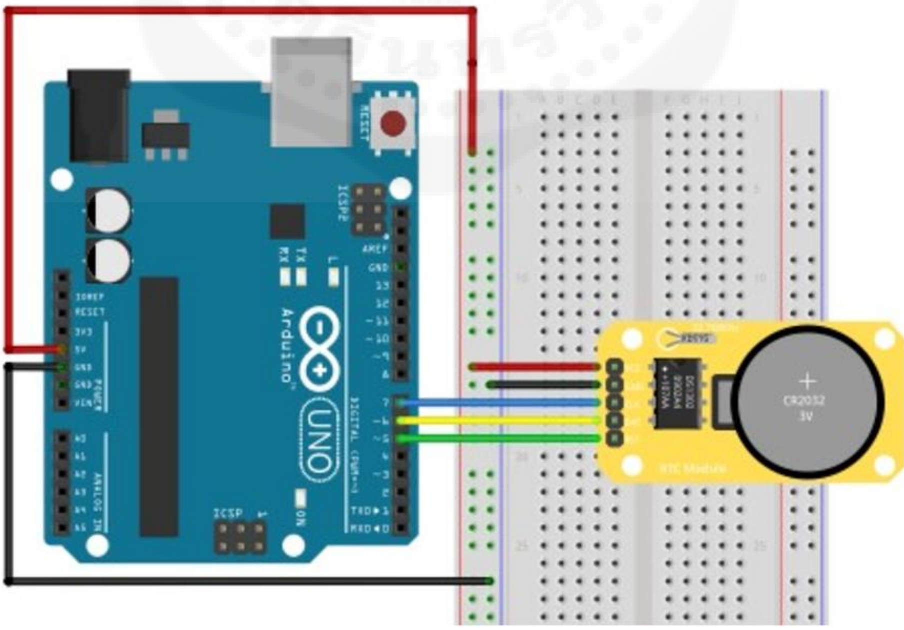
รูปที่ 9.11 การต่อวงจรสำหรับคำถามข้อ 1

2. เขียนโปรแกรมแสดงค่า ว/ด/ป และเวลาจากไอซี ds1302 กำหนดให้ใช้ Digital I/O ขา 5, 6 และ 7 ของบอร์ด Arduino สำหรับต่อกับขา CE, I/O และ CLOCK ของ ds1302 และ LCD ใช้การเชื่อมต่อกับบอร์ด Arduno แบบ $I^2C$ กำหนดให้แสดงผล ว/ด/ป ที่บรรทัดที่ 1 และเวลาที่บรรทัดที่ 2 ของ LCD




รูปที่ 9.12 การต่อวงจรสำหรับคำถามข้อ 2

3. เขียนโปรแกรมแสดงค่าเวลาจากไอซี ds1302 และให้ตั้งค่าเวลาให้ LED Digital I/O ขา 13 สว่างที่เวลา 13.00 น. และดับที่เวลา 13.05 น. กำหนดให้ใช้ Digital I/O ขา 5, 6 และ 7 ของบอร์ด Arduino สำหรับต่อกับขา CE, I/O และ CLOCK ของ ds1302 โดยแสดงค่าเวลาที่ Serial Monitor




รูปที่ 9.13 การต่อวงจรสำหรับคำถามข้อ 3

## เอกสารอ้างอิง

http://domoticx.com/arduino-rtc-tijdklok-ds1302/ : 2560

http://playground.arduino.cc/Main/DS1302RTC : 2560

http://datasheets.maximintegrated.com/en/ds/DS1302.pdf : 2560

# บทที่ 10

## การบันทึกข้อมูลด้วย SD Card

### 10.1 SD Card


http://nintendo.wikia.com/wiki/SD_Card : 2561

รูปที่ SD Card

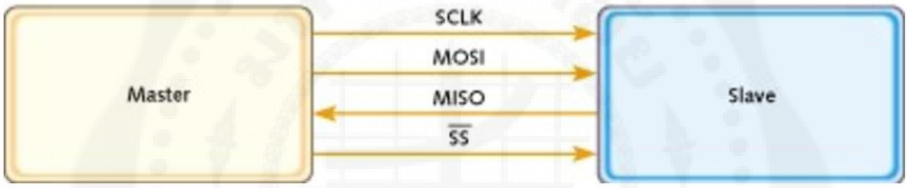
SD Card หรือ Secure Digital Card เป็นอุปกรณ์สำหรับจัดเก็บข้อมูลในรูปแบบการ์ดเป็นหน่วยความจำแบบไม่ลบเลือน (nonvolatile memory) หมายถึงข้อมูลในการ์ดจะไม่ลบหายไปถึงแม้ว่าจะหยุดจ่ายแรงดันไฟเลี้ยงแล้วก็ตาม

### 10.2 การสื่อสารข้อมูลแบบ SPI

การสื่อสารข้อมูลแบบ SPI หรือ Serial Peripheral Interface เป็นการสื่อสารอนุกรมแบบ Synchronous อีกแบบหนึ่ง ที่ทำงานโดยให้อุปกรณ์ตัวหนึ่งทำหน้าที่เป็นตัวหลัก (Master) และอุปกรณ์อีกตัวหนึ่งทำหน้าที่เป็นตัวรอง (Slave) โดยสามารถส่งข้อมูลได้ในแบบ Full-duplex นั่นคือสัญญาณติดสื่อสารกันระหว่าง Master และ Slave ได้อย่างต่อเนื่อง รูปแบบการสื่อสารข้อมูลแบบ SPI จะไม่ได้มาตรฐานกำหนดตายตัว จะใช้วิธีกำหนดการสื่อสาร

กันเอาเอง หรือดูจาก Datasheet ของอุปกรณ์ การสื่อสารข้อมูลแบบ SPI จะใช้สายสัญญาณ 4 เส้น ประกอบด้วย

- SCLK (Serial Clock) ใช้ส่งสัญญาณนาฬิกาจาก Master ไปหา Slave เพื่อกำหนดจังหวะการรับส่งข้อมูล
- MOSI (Master Out Slave In) ใช้ส่งข้อมูลจาก Master ไปหา Slave
- MISO (Master In Slave Out) ใช้รับข้อมูลจาก Slave มายัง Master
- \SS (Slave Select) ใช้ส่งสัญญาณ Low จาก Master ไปหา Slave ที่ต้องการติดต่อด้วย

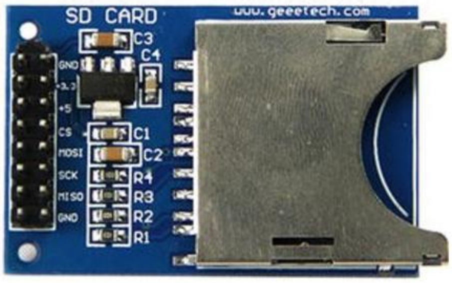



http://www.embedded.com/electronics-blogs/beginner-s-corner/4023908/Introduction-to-Serial-Peripheral-Interface : 2561

รูปที่ 10.2 การสื่อสารข้อมูลแบบ SPI

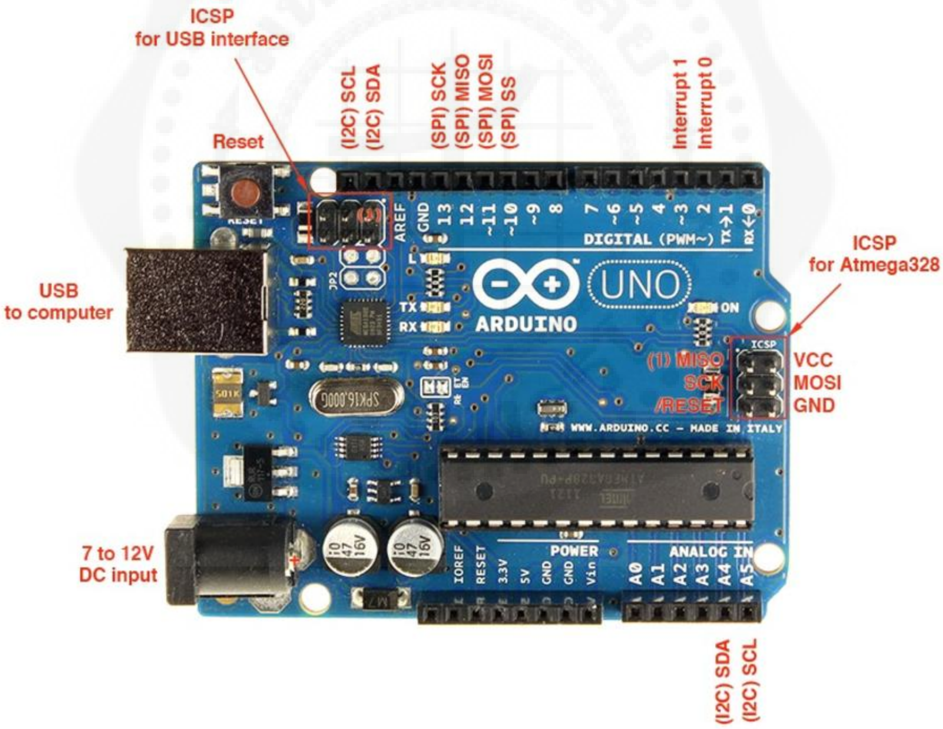
### 10.3 การใช้งาน SD Card กับ Arduino

การใช้งาน SD Card กับ Arduino จะเชื่อมต่อกันผ่านทางโมดูลที่ถูกออกแบบมาให้ใช้เชื่อมต่อกับบอร์ด Arduino ดังรูปที่ 10.3 โดยการเชื่อมต่อนั้นที่บอร์ด Arduino จะมีตำแหน่งการเชื่อมต่อแบบ SPI ที่ถูกกำหนดไว้อยู่แล้วในแต่ละรุ่นของบอร์ด Arduino โดยรูปที่ 10.4 จะเป็นตำแหน่งการเชื่อมต่อแบบ SPI ของบอร์ด Arduino UNO



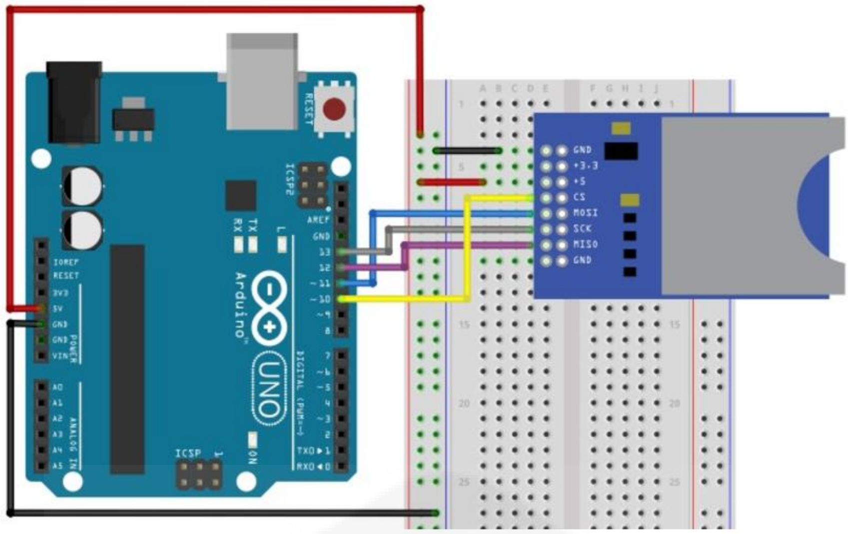

http://www.geeetech.com/wiki/index.php/Arduino_SD_card_Module : 2561

รูปที่ 10.3 SD Card โมดูล




https://forum.arduino.cc/index.php?topic=84190.0 : 2561

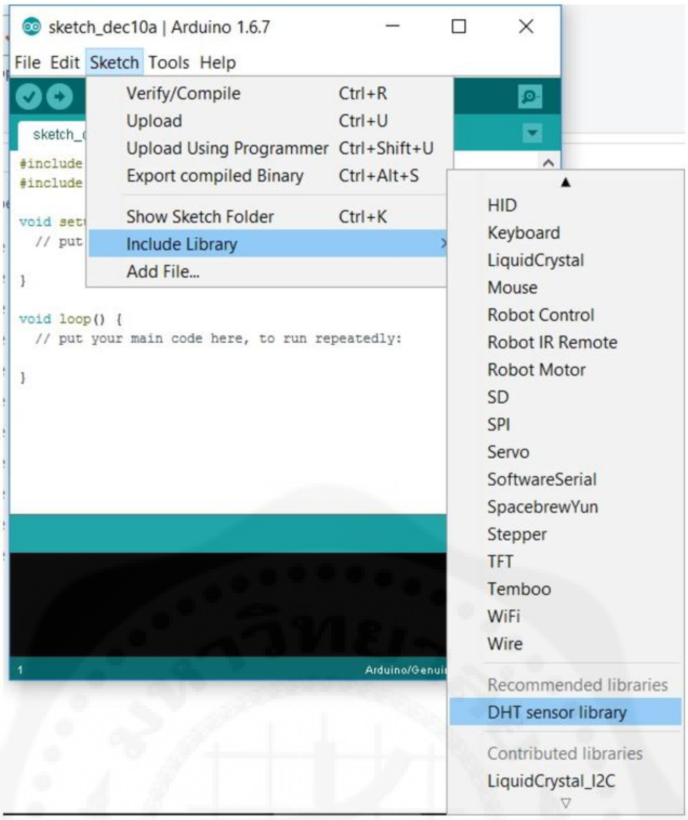
รูปที่ 10.4 ตำแหน่งการเชื่อมต่อแบบ SPI ของบอร์ด Arduino UNO




รูปที่ 10.4 การเชื่อมต่อบอร์ด Arduino กับโมดูล SD Card

## 10.4 เริ่มใช้งาน Arduino IDE กับ SD Card

การใช้งาน Arduino IDE กับ SD Card จะต้องมีการเรียกไลบรารี่มาใช้ร่วมในการเขียนโปรแกรม ซึ่งโปรแกรม Arduino IDE จะติดตั้งไลบรารี่ที่จะใช้กับ SD Card มาไว้อยู่แล้ง คือ SPI.h และ SD.h โดยสามารถตรวจสอบว่าโปรแกรม Arduino IDE ได้ติดตั้งไลบรารี่ทั้งสองไฟล์แล้วหรือไม่ ได้ดังรูปที่ 10.5

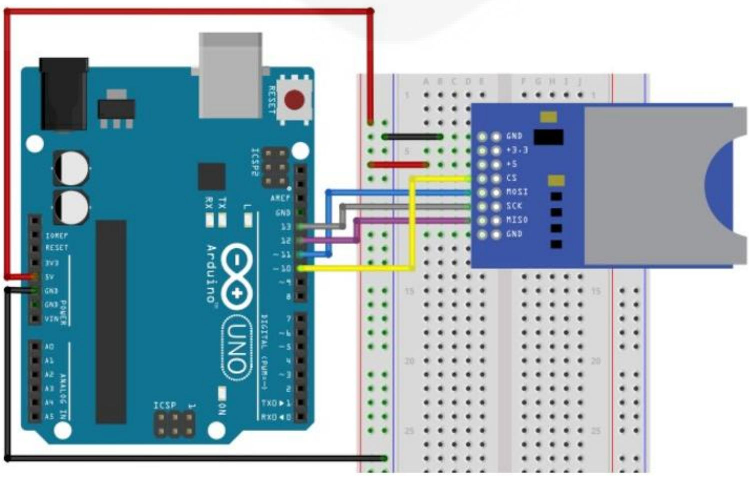



รูปที่ 10.5 ไลบรารี่เสริม SD และ SPI

ก่อนที่นำ SD Card มาใช้กับ Arduino จะต้องมี files system เป็นแบบ FAT32 หากไม่ใช่ ให้ทำการ Format SD Card นั้นก่อนโดยกำหนด file system เป็น FAT32 โดยสั่งได้จาก Windows ผ่าน Format Removable Disk ดังรูปที่ 10.6

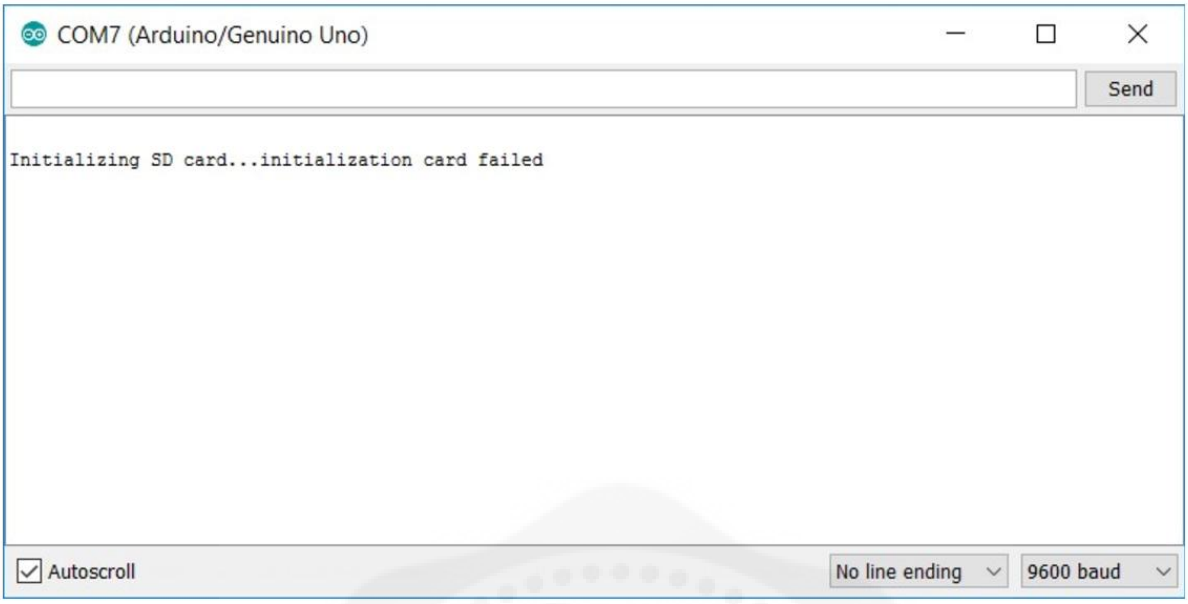

รูปที่ 10.6 การ Format SD Card ด้วยโปรแกรมของ Windows

**ตัวอย่างที่ 1** เขียนโปรแกรมทดสอบการต่อ SD card กับบอร์ด Arduino โดยเชื่อมต่อโมดูล SD Card ตามรูปที่ 10.7 โดยแสดงผลการเชื่อมต่อที่ Serial Monitor

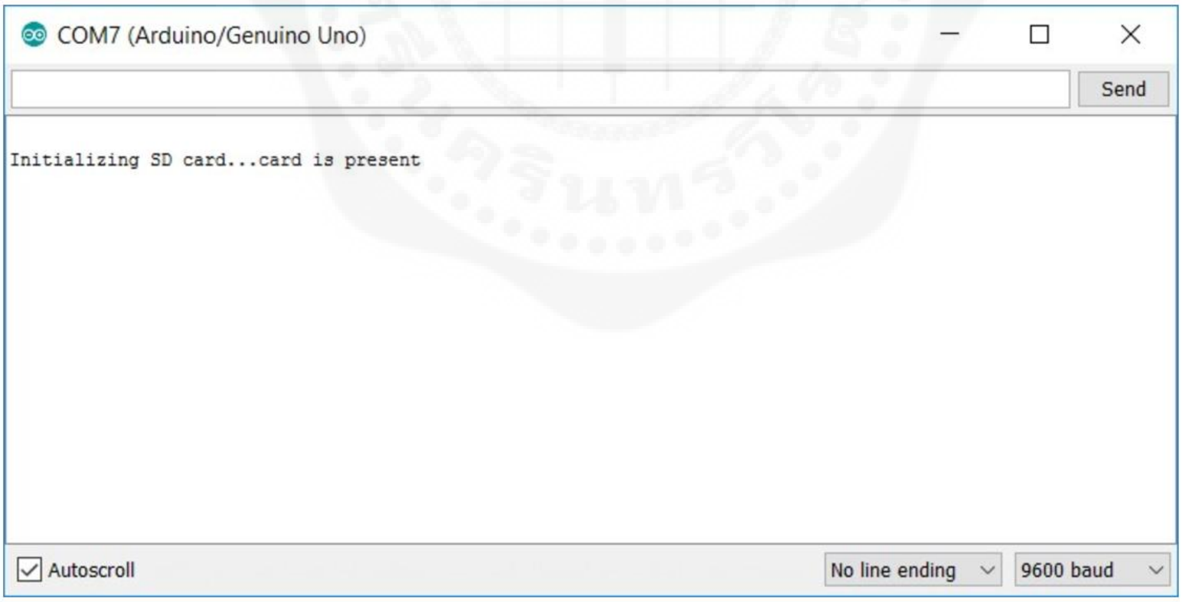


รูปที่ 10.7 การเชื่อมต่อบอร์ด Arduino กับโมดูล SD Card

```
#include <SPI.h>

#include <SD.h>

// กำหนดตัวแปรเพื่อใช้กับ SD utility library

int cardSelect = 10;

void setup() {

Serial.begin(9600);

while (!Serial) { ; // รอพอร์ทอนุกรมเชื่อมต่อ   }

Serial.print("\nInitializing SD card...");

if (!SD.begin(cardSelect)) {

Serial.println("initialization card failed");

return;

} else {

Serial.println("card is present");  }

}

void loop() { }
```




รูปที่ 10.8 ผลการทดลองตามตัวอย่างที่ 1. เมื่อบอร์ด Arduino ไม่สามารถติดต่อกับ SD Card ได้

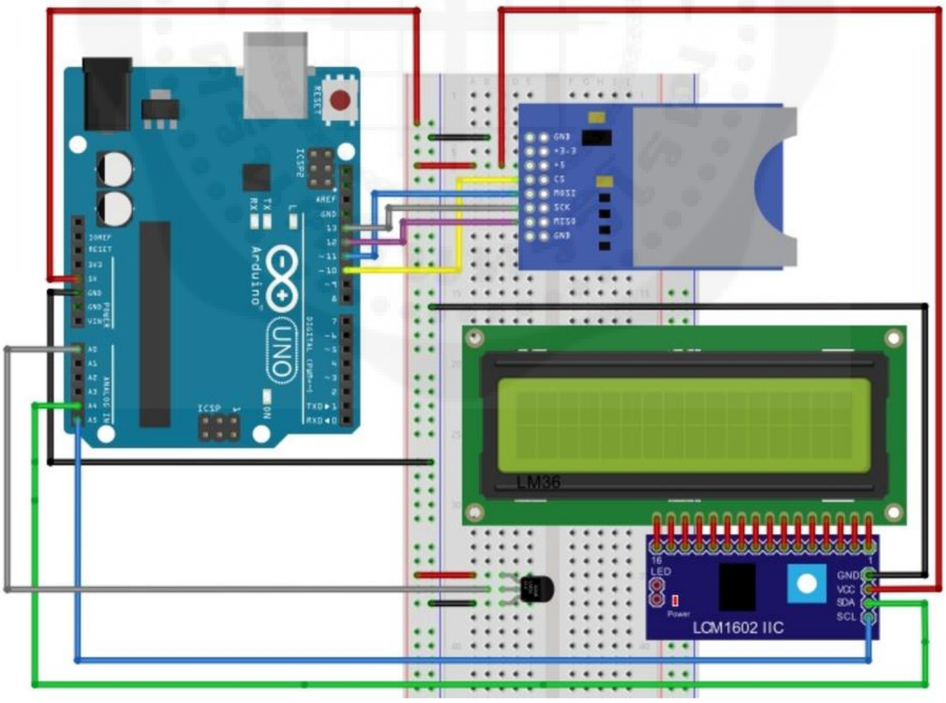



รูปที่ 10.9 ผลการทดลองตามตัวอย่างที่ 1. เมื่อบอร์ด Arduino สามารถติดต่อกับ SD Card ได้

**การทำงานของโปรแกรม**

โปรแกรมกำหนดให้ใช้สัญญาณเลือก SD Card ที่ขา Digital I/O ขา 10 เมื่อโปรแกรมเริ่มทำงานจะทำการทดสอบติดต่อกับการด้วยคำสั่ง if (!SD.begin(cardSelect)) ซึ่งถ้าติดต่อได้จะแสดงข้อความ card is present แต่ถ้าติดต่อไม่ได้จะแสดงข้อความ initialization card failed ที่หน้าต่าง Serial Monitor

**ตัวอย่างที่ 2** ใช้เซ็นเซอร์ LM35 วัดอุณหภูมิ โดยค่าแรงดันเอาท์พุทต่อที่ขา Analog I/O ขา A0 ของบอร์ด Arduino โดยแสดงค่าอุณหภูมิที่วัดได้ที่ LCD ค่าอุณหภูมิที่วัดได้ให้มีทศนิยม 2 ตำแหน่ง และบันทึกค่าอุณหภูมิที่วัดได้ไว้ใน SD card ด้วย




รูปที่ 10.10 วงจรสำหรับตัวอย่างที่ 2

```
#include <LiquidCrystal_I2C.h>

#include <SPI.h>

#include <SD.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

int inputPin = 0;

float read_Temp;

float Temp;

int cardSelect = 10;

void setup() {

lcd.begin();

Serial.begin(9600);

Serial.print("\nInitializing SD card...");

if (!SD.begin(cardSelect)) {

Serial.println("initialization card failed");

return;

} else {

Serial.println("card is present");
```

```
}

}

void loop() {

read_Temp = analogRead(inputPin);

Temp = (read_Temp*5*100)/1023 ;

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Temp = ");

lcd.print(Temp);

lcd.print(" *C");

delay(500);

//SD Card Write

File dataFile = SD.open("datalog.txt", FILE_WRITE);

if (dataFile) {

dataFile.print(Temp);

dataFile.println(" *C");

dataFile.close();
```

```
Serial.println(Temp);
}
}
```

**การทำงานของโปรแกรม**

โปรแกรมกำหนดให้ใช้สัญญาณเลือก SD Card ที่ขา Digital I/O ขา 10 เมื่อโปรแกรมเริ่มทำงานจะทำการทดสอบติดต่อกับ SD Card ด้วยคำสั่ง if (!SD.begin(cardSelect)) ซึ่งถ้าติดต่อได้จะแสดงข้อความ card is present แต่ถ้าติดต่อไม่ได้จะแสดงข้อความ initialization card failed ที่หน้าต่าง Serial Monitor ต่อจากนั้นโปรแกรมจะอ่านค่าอุณหภูมิด้วยคำสั่ง read_Temp = analogRead(inputPin) แล้วทำการแปลงข้อมูลจากอนาล็อกเป็นดิจิตอลด้วยคำสั่ง Temp = (read_value*5)/1023 แล้วจึงนำผลที่ได้ไปแสดงผลที่ LCD และทำการบันทึกค่าอุณหภูมิที่แสดงผลที่ LCD ลงใน SD Card ด้วย โดยจะต้องสร้างไฟล์ datalog.txt ไว้ใน SD Card ก่อน แล้วใช้คำสั่ง File dataFile = SD.open("datalog.txt", FILE_WRITE) เพื่อเปิดไฟล์ datalog.txt และบันทึกค่าอุณหภูมิลงใน SD Card ด้วยคำสั่ง dataFile.println(Temp)


datalog.txt - Notepad

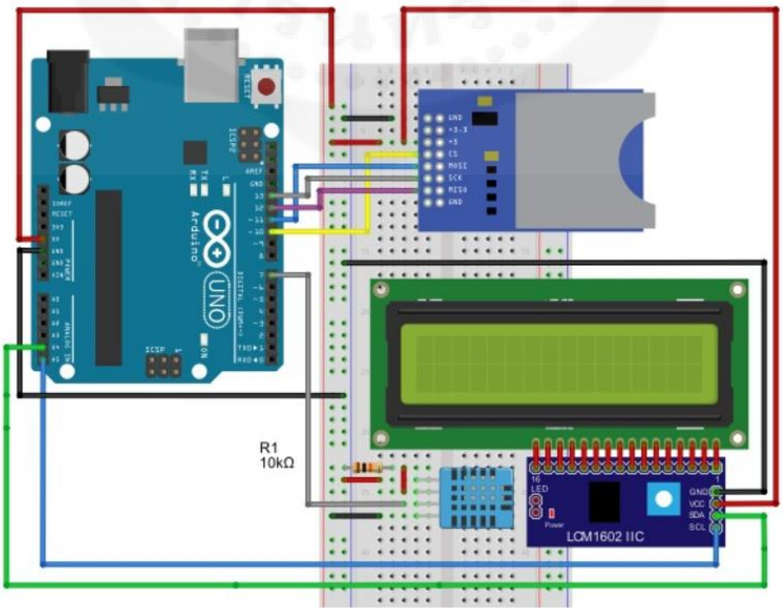
File Edit Format View Help

```
23.95 *C
24.93 *C
25.42 *C
26.39 *C
25.90 *C
25.90 *C
26.39 *C
25.42 *C
25.42 *C
25.90 *C
25.42 *C
25.42 *C
25.42 *C
25.42 *C
25.42 *C
24.93 *C
25.42 *C
25.42 *C
24.44 *C
25.42 *C
24.93 *C
24.44 *C
24.93 *C
24.44 *C
24.93 *C
```

Ln 25, Col 9

รูปที่ 10.11 ผลการทดลองตามตัวอย่างที่ 2. อุณหภูมิที่วัดได้ถูกบันทึกใน SD Card

**ตัวอย่างที่ 3** การวัดความชื้น และอุณหภูมิเป็นองศาเซลเซียส โดยแสดงผลที่จอ LCD กำหนดให้ใช้ Digital I/O ขา 7 ของบอร์ด Arduino สำหรับต่อกับขา 2 ของ DTH11 และบันทึกค่าอุณหภูมิที่วัดได้ไว้ใน SD card โดยให้ไฟล์ที่บันทึกเป็นนามสกุล .CSV




รูปที่ 10.12 วงจรสำหรับตัวอย่างที่ 3

ไฟล์นามสกุล .CSV โดย CSV เป็นคำย่อของ Comma Separated Value เป็นไฟล์ข้อความที่ใช้สำหรับเก็บข้อมูลในรูปแบบตาราง โดยใช้เครื่องหมายจุลภาค หรือคอมม่า (,) ในการแบ่งแต่ละคอลัมน์ โดยสามารถใช้ Microsoft Excel หรือ Notepad เปิดดูไฟล์ .CSV ได้

```
#include <LiquidCrystal_I2C.h>

#include <SPI.h>

#include <SD.h>

#include "DHT.h"

LiquidCrystal_I2C lcd(0x27, 16, 2);

#define DHTPIN 7

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

Sd2Card card;

int cardSelect = 10;

void setup() {

lcd.begin();

Serial.begin(9600);

dht.begin();

Serial.print("\nInitializing SD card...");
```

```
if (!SD.begin(cardSelect)) {
Serial.println("initialization card failed");
return;
} else {
Serial.println("card is present");  }
File dataFile = SD.open("datalog.csv", FILE_WRITE);
if(dataFile)  {
//Label Excel Collumn
dataFile.print("Temperature *C");
dataFile.print(",");
dataFile.println("Humidity %");
dataFile.close();  }
}
void loop() {
float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(t) || isnan(h)) {
```

```
lcd.clear();
lcd.print("Failed to read");
lcd.setCursor(0,1);
lcd.print("from DHT11");
delay(1000);
return;  }
else {
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Temp=");
lcd.print(t);
lcd.print(" *C");
lcd.setCursor(0,1);
lcd.print("Humidity=");
lcd.print(h);
lcd.print("% ");
delay(1000);  }
```

```
//SD Write
File dataFile = SD.open("datalog.csv", FILE_WRITE);
if (dataFile) {
dataFile.print(t);
dataFile.print(",");
dataFile.println(h);
dataFile.close();
Serial.println(t);  }
}
```

| | | | | |
|---|---|---|---|---|
| 1 | | | | |
| 2 | Temperature *C | Humidity % | | |
| 3 | 32 | 69 | | |
| 4 | 32 | 69 | | |
| 5 | 32 | 69 | | |
| 6 | 32 | 69 | | |
| 7 | 32 | 68 | | |
| 8 | 32 | 68 | | |
| 9 | 32 | 68 | | |
| 10 | 32 | 68 | | |
| 11 | 32 | 68 | | |
| 12 | 32 | 68 | | |
| 13 | 32 | 68 | | |
| 14 | 32 | 68 | | |

รูปที่ 10.13 ผลการทดลองตามตัวอย่างที่ 3. ไฟล์ .CSV ที่บันทึกใน SD Card

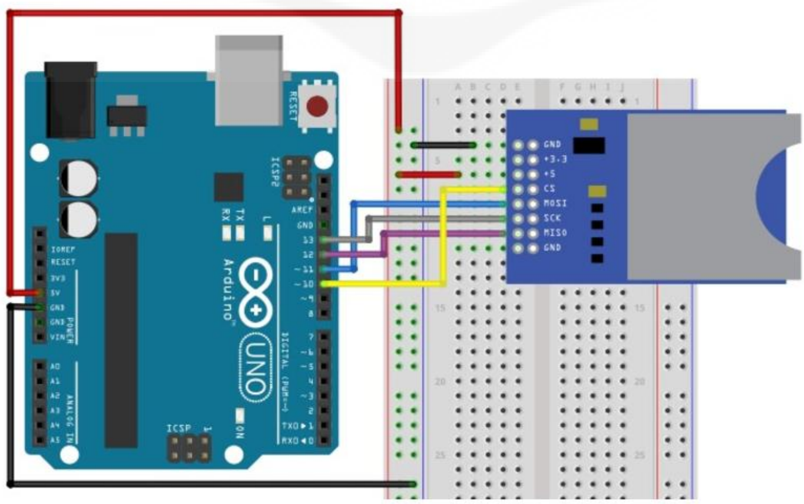
**การทำงานของโปรแกรม**

โปรแกรมกำหนดให้ใช้สัญญาณเลือก SD Card ที่ขา Digital I/O ขา 10 รับข้อมูลจาก DHT11 ที่ขา Digital I/O ขา 7 เมื่อโปรแกรมเริ่มทำงานจะทำการทดสอบติดต่อกับ SD

Card ด้วยคำสั่ง if (!SD.begin(cardSelect)) ซึ่งถ้าติดต่อได้จะแสดงข้อความ card is present แต่ถ้าติดต่อไม่ได้จะแสดงข้อความ initialization card failed ที่หน้าต่าง Serial Monitor ต่อจากนั้นโปรแกรมจะสร้างหัวตารางของไฟล์ .CSV ใน SD Card ด้วยคำสั่ง dataFile.print("Temperature *C") dataFile.print(",") และ dataFile.println("Humidity %") โดยจะต้องสร้างไฟล์ datalog.csv ใน SD Card เอาไว้ล่วงหน้าก่อน จากนั้นจะอ่านค่าอุณหภูมิด้วยคำสั่ง float t = dht.readTemperature() และค่าความชื้นด้วยคำสั่ง float t = dht.readTemperature() แล้วจึงนำผลที่ได้ไปแสดงผลที่ LCD และทำการบันทึกค่าที่วัดได้นั้นลงใน SD Card ด้วย โดยใช้คำสั่ง dataFile.print(t) dataFile.print(",") และ dataFile.println(h);

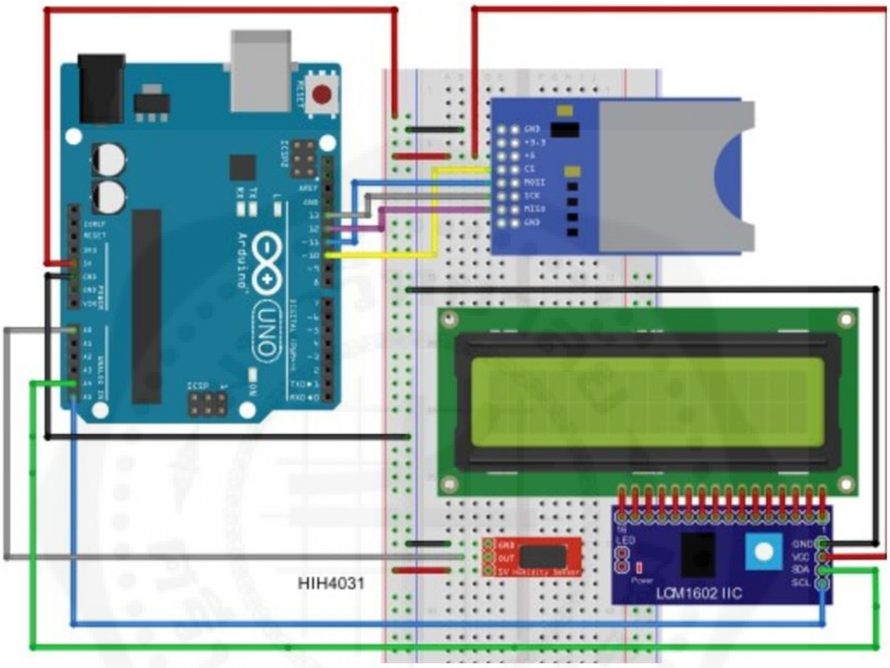
**คำถาม**

1. เขียนโปรแกรมทดสอบการต่อ SD card กับบอร์ด Arduino โดยเชื่อมต่อโมดูล SD Card โดยแสดงผลการเชื่อมต่อเป็นผลสำเร็จที่ให้ LED ที่ขา 13 (สีส้ม) ของบอร์ด Arduino LED สว่าง



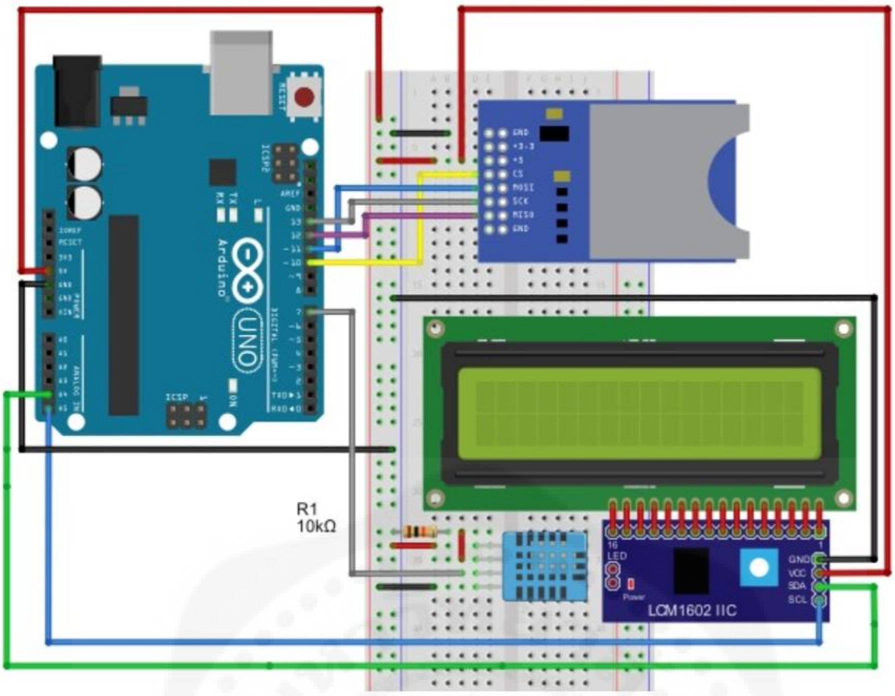

รูปที่ 10.14 การต่อวงจรสำหรับคำถามข้อ 1.

2. ใช้เซ็นเซอร์ HIH4030 วัดความชื้นสัมพัทธ์ โดยค่าแรงดันเอาท์พุทต่อที่ขา Analog I/O ขา A0 ของบอร์ด Arduino โดยแสดงค่าอุณหภูมิที่วัดได้ที่ LCD บันทึกค่าความชื้นสัมพัทธ์ ที่วัดได้ไว้ใน SD card ด้วย




รูปที่ 10.15 การต่อวงจรสำหรับคำถามข้อ 2.

3. จงเขียนโปรแกรมเพื่อแสดงค่าอุณหภูมิเป็นองศาเซลเซียส และองศาฟาเรนไฮต์ ที่จอ LCD กำหนดให้ใช้ Digital I/O ขา 2 ของบอร์ด Arduino สำหรับต่อกับขา 2 ของ DTH11 จอ LCD ใช้การเชื่อมต่อกับบอร์ด Arduno แบบ $I^2C$ และบันทึกค่าอุณหภูมิที่วัดได้ไว้ใน SD card โดยให้ไฟล์ที่บันทึกเป็นนามสกุล .CSV




รูปที่ 10.16 การต่อวงจรสำหรับคำถามข้อ 3.

## เอกสารอ้างอิง

1. Simon Monk. (2014). Programming Arduino Next Steps. New York : McGraw-Hill Education.
2. https://en.wikipedia.org/wiki/Comma-separated_values : 2561
3. http://www.embedded.com/electronics-blogs/beginner-s-corner/4023908/Introduction-to-Serial-Peripheral-Interface : 2561
4. https://www.arduino.cc/en/reference/SPI : 2561
5. https://www.arduino.cc/en/reference/SD : 2561
6. https://oscarliang.com/sd-card-arduino/ : 2561

## ดรรชนี

# ประวัติผู้แต่ง

## ผู้ช่วยศาสตราจารย์ศิริพงษ์  ฉายสินธ์

### ประวัติการศึกษา

- วศ.ม. (วิศวกรรมไฟฟ้า)    สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
- วศ.บ. (วิศวกรรมอิเล็กทรอนิกส์)   สถาบันเทคโนโลยีราชมงคล วิทยาเขตเทเวศร์

### ประวัติการทำงาน

- 2539 - 2549    อาจารย์ ประจำภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ องครักษ์
- 2549 - ปัจจุบัน  ผู้ช่วยศาสตราจารย์ ประจำภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์  มหาวิทยาลัยศรีนครินทรวิโรฒ องครักษ์

### ตำแหน่งอื่นๆ

- 2553 - 2562    รองคณบดีฝ่ายแผนและพัฒนา  คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ องครักษ์

# การใช้งาน Arduino เบื้องต้น