หน่วยที่ 10 การใช้งาน ARDUINO กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต

สาระสำคัญ

เซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์ เป็นเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์แบบดิจิทัล และเชื่อมต่อด้วยสัญญาณเพียงเส้นเดียวแบบสองทิศทาง (bidirectional) ใช้แรงดันไฟเลี้ยงได้ในช่วง 3.3V ถึง 5.2V สามารถวัดค่าอุณหภูมิได้ในช่วง -40 ถึง 80°C ความละเอียดในการวัดอุณหภูมิและความชื้น คือ 0.5°C และ 0.1%RH ในการทำงานของโมดูลอัลตร้าโซนิค จะอาศัยการส่งคลื่นเสียงความถี่สูงออกไปประมาณ 40kHz และจับเวลาในการเดินทางของคลื่นเสียงเมื่อไปและกลับมาหลังจากสะท้อนวัตถุกีดขวาง

เนื้อหาสาระการเรียนรู้

10.1 การอ่านข้อมูลจากโมดูลวัดอุณหภูมิและความชื้น DHT21 (AM2301)

10.2 การอ่านข้อมูลจากโมดูลวัดอุณหภูมิและความชื้น DHT22 / AM2302

10.3 การใช้งานโมดูลวัดอุณหภูมิและความชื้นสัมพัทธ์ SHT11

10.4 การใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 วัดระยะห่างด้วยคลื่นอัลตร้าโซนิค

10.5 การใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 หลายชุด

จุดประสงค์การเรียนรู้

จุดประสงค์ทั่วไป

1. เพื่อให้มีความรู้ความเข้าใจเกี่ยวกับการใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต
2. เพื่อให้สามารถนำความรู้ไปประยุกต์ใช้ในการเขียนโปรแกรมกำหนดการทำงานให้งาน Arduino
3. เพื่อให้ตระหนักถึงความสำคัญของการใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต

จุดประสงค์เชิงพฤติกรรม

1. อธิบายความรู้พื้นฐานเกี่ยวกับโมดูลวัดอุณหภูมิและความชื้น DHT21 (AM2301) ได้
2. ใช้งานโมดูลวัดอุณหภูมิและความชื้น DHT21 (AM2301) ได้
3. อธิบายความรู้พื้นฐานโมดูลวัดอุณหภูมิและความชื้น DHT22 / AM2302 ได้
4. ใช้งานโมดูลวัดอุณหภูมิและความชื้น DHT22 / AM2302 ได้
5. อธิบายความรู้พื้นฐานโมดูลวัดอุณหภูมิและความชื้นสัมพัทธ์ SHT11 ได้
6. ใช้งานโมดูลวัดอุณหภูมิและความชื้นสัมพัทธ์ SHT11 ได้
7. อธิบายความรู้พื้นฐานการใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 ได้
8. ใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 วัดระยะห่างด้วยคลื่นอัลตร้าโซนิคได้
9. ใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 หลายชุดได้

แบบทดสอบหลังเรียน หน่วยที่ 10

เรื่อง การใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต

เรื่อง การใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต **ใช้เวลา** 20 **นาที**

วิชา ไมโครคอนโทรลเลอร์เบื้องต้น **รหัสวิชา** (2127-2107)

ระดับชั้น ประกาศนียบัตรวิชาชีพ (ปวช.) **สาขาวิชา** เมคคาทรอนิกส์

****

คำชี้แจง 1. แบบทดสอบมีทั้งหมด 10 ข้อ (10 คะแนน)

2. ให้ผู้เรียนเลือกคำตอบที่ถูกที่สุดแล้วกาเครื่องหมายกากบาท (✕) ลงในกระดาษคำตอบ

1. โมดูล DHT21 (AM2301) เป็นเซนเซอร์วัดวัดอุณหภูมิและความชื้นสัมพัทธ์แบบใด
   - ก. ขนาน
   - ข. แอนะล็อก
   - ค. แบบดิจิตอล
   - ง. อนุกรม
2. DHT21 (AM2301) มีขาเชื่อมต่ออะไรบ้าง
   - ก. VCC, GND และ SCA
   - ข. VCC, GND และ SLA
   - ค. VCC, GND และ SDA
   - ง. VCC, GND และ SDR
3. DHT21 (AM2301) ต่อสายเชื่อมต่ออย่างไร
   - ก. สายสีแดงสำหรับ GND สายสีดำคือ VCC และสายสีเหลืองคือ SDA
   - ข. สายสีแดงสำหรับ VCC สายสีดำคือ GND และสายสีเหลืองคือ SDA
   - ค. สายสีแดงสำหรับ VCC สายสีดำคือ SDA และสายสีเหลืองคือ GND
   - ง. สายสีแดงสำหรับ SDA สายสีดำคือ GND และสายสีเหลืองคือ VCC
4. DHT22 / AM2302 ขาสัญญาณดิจิทัลเพียงเส้นเดียวในการเชื่อมต่อแบบ
   - ก. บิตอนุกรมสองทิศทาง (serial data, bi-directional)
   - ข. บิตอนุกรมหนึ่งทิศทาง (serial data, one-directional)
   - ค. บิตผสมสองทิศทาง (serial data, bi-directional)
   - ง. บิตขนานสองทิศทาง (serial data, bi-directional)

5. ค่า RH ที่อ่านได้คืออะไร

ก. ความชื้นสัมพันธ์ (Relative Humidity - RH)

ข. ความชื้นสัมพัทธ์ (Relative Humidity - RH)

ค. ความชื้นสัมประสิทธิ์ (Relative Humidity - RH)

ง. ความชื้นสัมผัส (Relative Humidity - RH)

6. โมดูล SHT11 ใช้แรงดันไฟเลี้ยงเท่าใด

ก. (Vcc): +1V .. +3.6V

ข. (Vcc): +0.5V .. +3.6V

ค. (Vcc): +2V .. +3.6V

ง. (Vcc): +1.5V .. +3.6V

7. โมดูล SHT11 ติดต่อสื่อสารแบบบัสแบบใด

ก. I2C

ข. Uart

ค. AVI

ง. SPI

8. โมดูล SHT11 ช่วงค่าความชื้นสัมพัทธ์ (Humidity Operating Range) คือ

ก. -1 ถึง 100 % RH

ข. 0 ถึง 100 % RH

ค. 10 ถึง 100 % RH

ง. -5 ถึง 100 % RH

9. โมดูลตรวจจับสัญญาณอินพุต HC-SR04 ใช้สำหรับทำอะไร

ก. วัดระยะทางด้วยคลื่นอัลตร้าซูมเมอร์

ข. วัดระยะทางด้วยคลื่นอัลตร้าไวโอเลต

ค. วัดระยะทางด้วยคลื่นอัลตร้าบีม

ง. วัดระยะทางด้วยคลื่นอัลตร้าโซนิค

10. โมดูล HC-SR04 ใช้คลื่นเสียงความถี่ประมาณ

ก. 40kHz

ข. 30kHz

ค. 20kHz

ง. 10kHz

หน่วยที่ 10
การใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต

10.1 การอ่านข้อมูลจากโมดูลวัดอุณหภูมิและความชื้น DHT21 (AM2301)

เซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์แบบดิจิทัลโมเดล AM2301 / DHT21 เป็นเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์แบบดิจิทัล และเชื่อมต่อด้วยสัญญาณเพียงเส้นเดียวแบบสองทิศทาง (bidirectional) ใช้แรงดันไฟเลี้ยงได้ในช่วง 3.3V ถึง 5.2V สามารถวัดค่าอุณหภูมิได้ในช่วง -40 ถึง 80°C ความละเอียดในการวัดอุณหภูมิและความชื้น คือ 0.5°C และ 0.1%RH และมีความแม่นยำ ±0.5°C และ ±3%RH ตามลำดับ ใช้ขาเชื่อมต่อเพียง 3 ขา ได้แก่ VCC, GND และ SDA (Serial Data) ในการอ่านข้อมูลแต่ละครั้ง จะอ่านข้อมูลทั้งหมด 40 บิต แบ่งเป็น 16 บิตสำหรับค่าความชื้น 16 บิตสำหรับค่าอุณหภูมิ และ 8 บิตสำหรับตรวจสอบค่า Parity Bits เพื่อดูว่าอ่านค่าได้ถูกต้องหรือไม่ โดย Arduino Sketch เพื่ออ่านค่าจากเซนเซอร์ดังกล่าว และนำมาแสดงผล

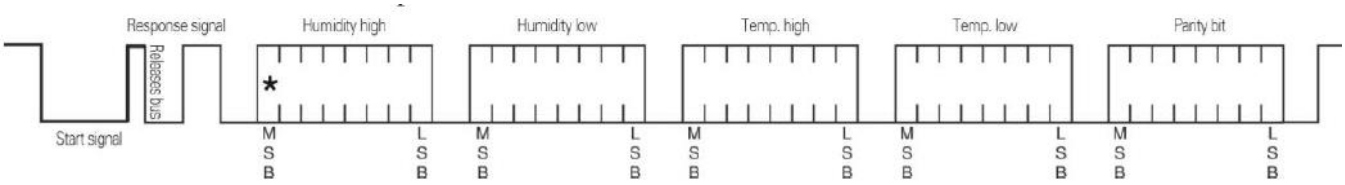
รูปที่ 10.1 แสดงโมดูลเซนเซอร์วัดอุณหภูมิและความชื้นสัมพัทธ์ โมเดล AM2301 (ด้านหลัง)
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

Pin	Color	Name	Description
1	Red	VDD	Power (3.3V–5.2V)
2	Yellow	SDA	Serial data, Dual-port
3	Black	GND	Ground
4		NC	Empty

ตารางที่ 10.1 แสดงขาสำหรับการเชื่อมต่อของโมดูล AM2301

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

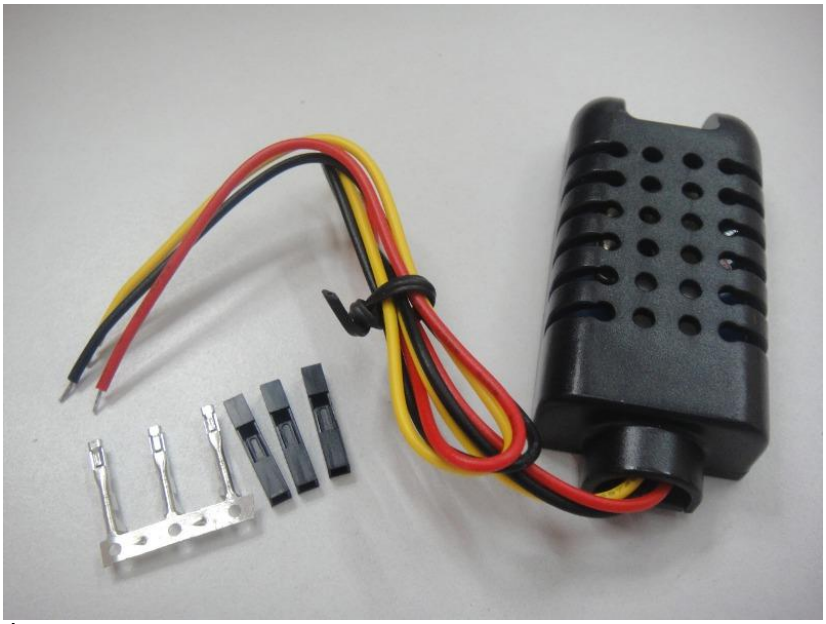
สายสีแดงสำหรับ VDD (ป้อนแรงดัน +5V) สายสีดำคือ GND และสายสีเหลืองคือ SDA



รูปที่ 10.2 แสดงโปรโตคอลในการรับส่งข้อมูลโดยใช้สายสัญญาณเส้นเดียวของโมดูล AM2301

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

ศึกษารายละเอียดเพิ่มเติมได้จากเอกสารของผู้ผลิต: am2301_datasheet.pdf



รูปที่ 10.3 แสดงตัวอย่างคอนเนคเตอร์ (female housing and crimp pins)

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

10.2 การอ่านข้อมูลจากโมดูลวัดอุณหภูมิและความชื้น DHT22 / AM2302

อุปกรณ์เซนเซอร์สำหรับวัดอุณหภูมิและความชื้นสัมพัทธ์ (Temperature & Relative Humidity Sensor) เป็นอุปกรณ์ที่สามารถนำมาประยุกต์ใช้งานทางด้านระบบสมองกลฝังตัวได้หลากหลาย เช่นการวัดและควบคุมอุณหภูมิและความชื้น ระบบบันทึกข้อมูลเกี่ยวกับอุณหภูมิและความชื้นในห้อง เป็นต้น อุปกรณ์ประเภทนี้แตกต่างกันตามผู้ผลิต ราคา ความแม่นยำ ความละเอียดในการวัด การให้ค่าแบบดิจิทัลหรือแบบแอนะล็อก เป็นต้น การทดลองใช้งานโมดูล DHT22 / AM2302 ซึ่งมีราคาถูก ให้ค่าเป็นแบบดิจิทัล ใช้ขาสัญญาณดิจิทัลเพียงเส้นเดียวในการเชื่อมต่อแบบบิตอนุกรมสองทิศทาง (Serial Data, Bi-Rirectional) โดยนำมาเชื่อมต่อกับ Arduino เพื่ออ่านค่าจากเซนเซอร์

10.2.1 ข้อมูลเชิงเทคนิค (Techinal details)

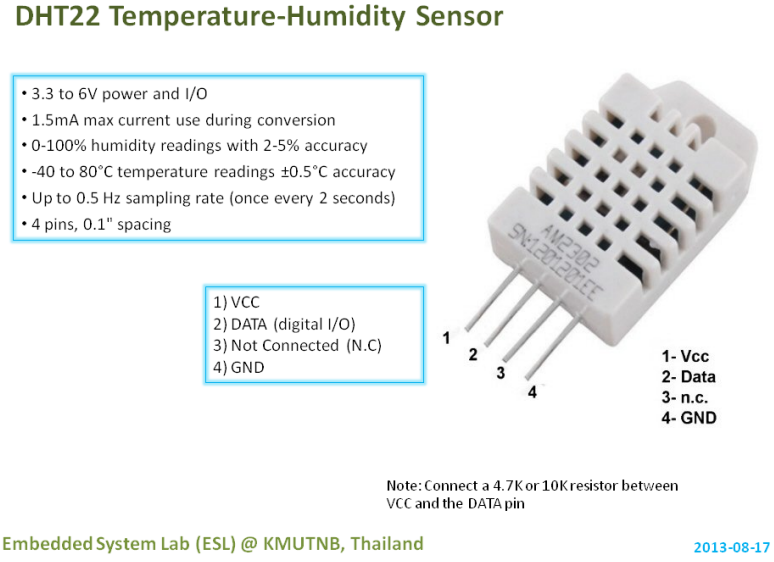
- ใช้แรงดันไฟเลี้ยงได้ในช่วง: 3.3V ถึง 5.5V DC (ดังนั้นจึงใช้ได้กับ 3.3V และ 5V)
- วัดอุณหภูมิได้ในช่วง: -40 to 80 °C (±0.5 °C accuracy)
- วัดความชื้นสัมพัทธ์ได้ในช่วง: 0 - 100 RH% (2 - 5% accuracy)
- อัตราการวัดสูงสุด: 0.5Hz
- คอนเนกเตอร์แบบ 4 ขา (0.1" / 2.54mm spacing)

  Pin 1 = VCC

  Pin 2 = SDA (Serial data, bidirectional)

  Pin 3 = N.C. (Not Connect)

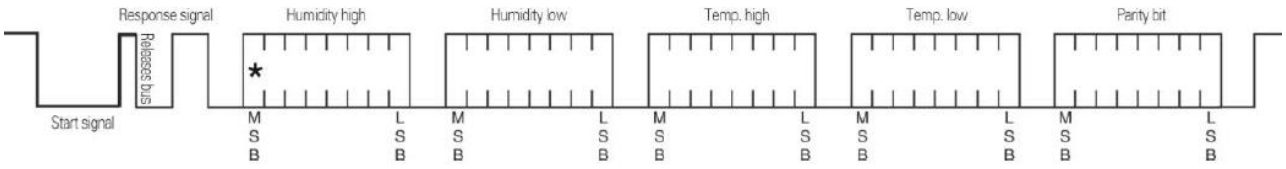
  Pin 4 = GND



รูปที่ 10.4 โมดูล DHT22 / AM2302
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

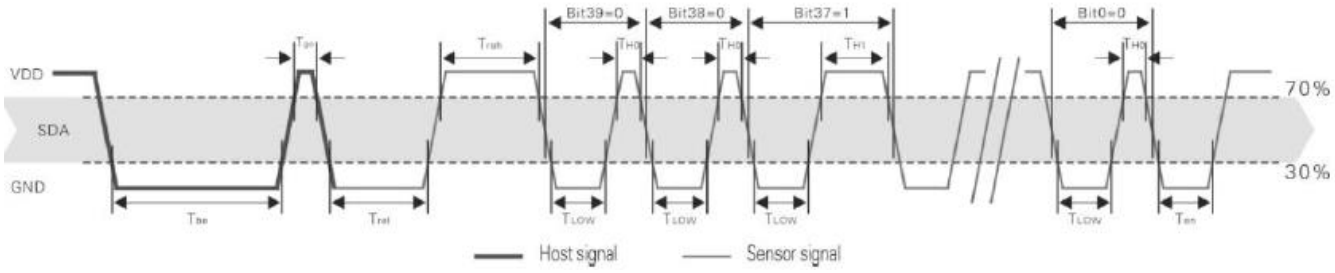
ในการอ่านข้อมูลจากไอซีนั้น จะใช้ขาสัญญาณเพียงเส้นเดียวคือ DATA (หรือ SDA) แบบสองทิศทาง และในสถานะปรกติสัญญาณ DATA จะเป็น HIGH ในการอ่านข้อมูลแต่ละครั้ง ไมโครคอนโทรลเลอร์จะต้องกำหนดให้ขา DATA เป็นเอาต์พุต และสร้างบิต START ซึ่งจะต้องเป็น LOW อย่างน้อย 800 usec จากนั้นจึงให้เป็น HIGH อย่างน้อย 20 μsec หลังจากนั้นเป็นการรอการตอบกลับ (Response) และจากไอซีขา DATA จะถูกต้องเปลี่ยนเป็นอินพุต

เริ่มต้นของการตอบกลับไอซี จะดึงสัญญาณลงเป็น LOW และปล่อยให้เป็น HIGH ช่วงละ 80 μsec โดยประมาณ (เรียกว่า Response Bit) จากนั้นจึงจะเป็นการส่งข้อมูลทีละบิต รวม 40 บิต (ช่วง LOW ตามด้วยช่วง HIGH) ช่วง LOW ของแต่ละบิต จะกว้างเท่ากัน แต่จะต่างกันในช่วง HIGH สำหรับบิตที่มีค่าเป็น 0 หรือ 1 (ใช้ความกว้างช่วง HIGH ในการจำแนกค่าของบิต)



รูปที่ 10.5 แสดงลำดับของข้อมูลบิตในการอ่านค่าจากไอซีทั้งหมด
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

สองไบต์แรกสำหรับความชื้น สองไบต์ต่อมาสำหรับอุณหภูมิ และไบต์สุดท้ายเป็น Checksum หรือ Parity Bits



รูปที่ 10.6 แสดงลำดับของข้อมูลบิตในการอ่านค่าจากไอซีและความกว้างของช่วง LOW และ HIGH
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

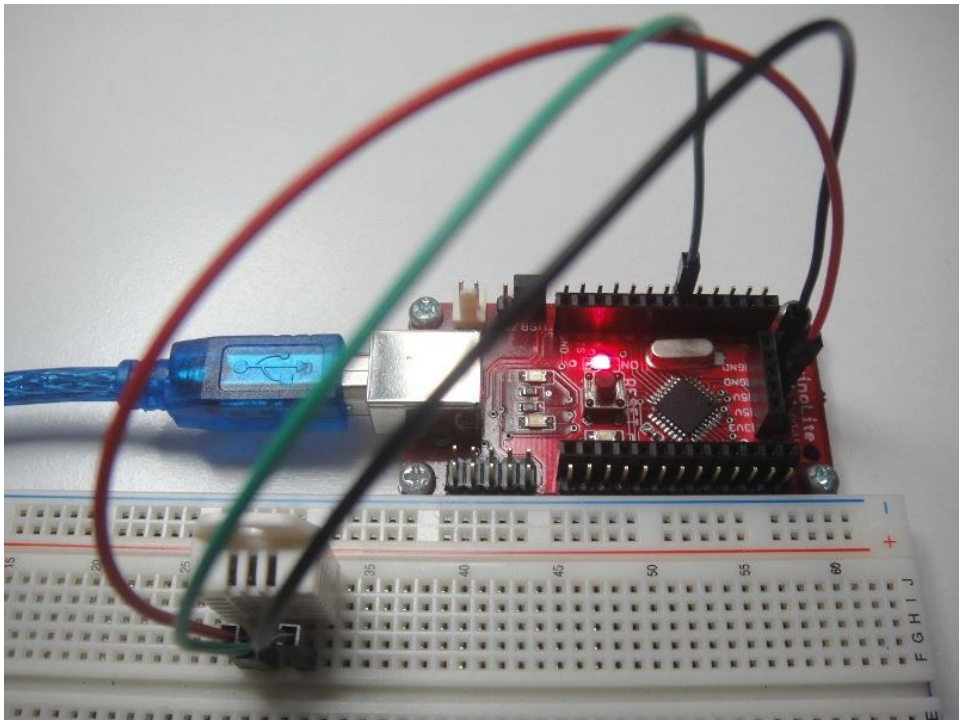
โปรแกรมที่ 10.1 Arduino Sketch 1

การอ่านค่าจากโมดูล DHT22 / AM2302 ด้วยบอร์ด Arduino แล้วนำค่าที่ได้แสดงผลผ่านทาง Serial

Monitor ของ Arduino IDE (ตั้งค่า baudrate = 115200) แล้วทำขั้นตอนซ้ำ

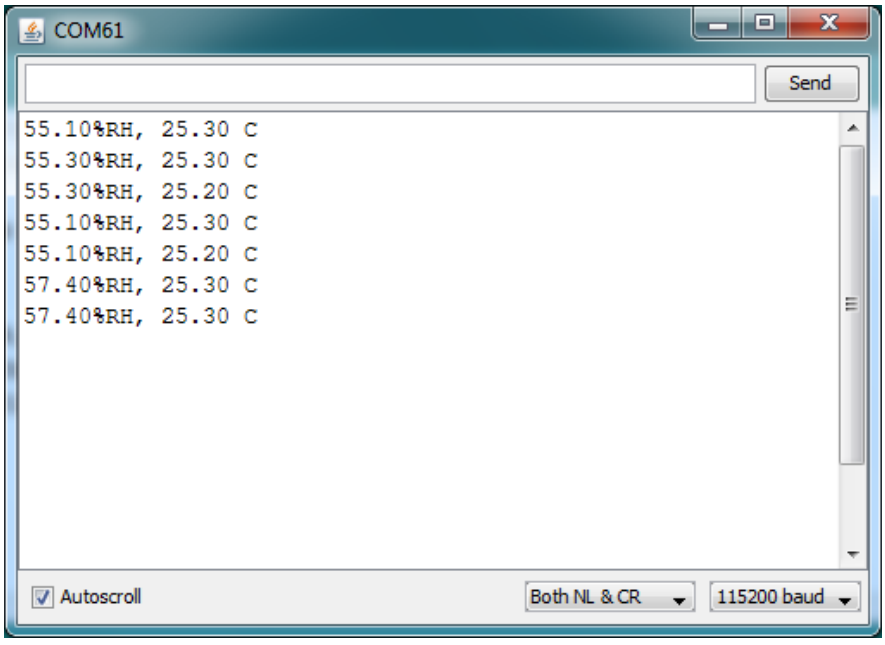
```
        // Author: RSP @KMUTNB
        // Date: 16-Aug-2013
        // Target Board: Arduino Uno (ATmega328P, 5V, 16MHz)
        // Arduino IDE: version 1.0.5

        // Demonstrate how to read data from DHT22 (AM2302) -- a digital
        // relative humidity and temperature sensor manufactured by
        // Aosong Electronics Co.,Ltd.
        const byte DATA_PIN = 5; // connected to the DATA pin of DHT22 (AM2302)
void setup() {
        pinMode( DATA_PIN, INPUT );
        digitalWrite( DATA_PIN, HIGH ); // enable internal pull-up
        Serial.begin( 115200 ); // use serial port (baudrate = 115200)
        }
        byte data[5];
void loop() {
         int count = 0;
        byte i=0, j=0;
        byte new_state, state = HIGH;
  for (byte x=0; x < 5; x++) {
        data[x] = 0; // clear data buffer
        }
        pinMode( DATA_PIN, OUTPUT ); // change direction to output
        digitalWrite( DATA_PIN, LOW ); // output low (send the start bit)
        delayMicroseconds( 1000 );
        digitalWrite( DATA_PIN, HIGH ); // output high
        delayMicroseconds( 40 );
        pinMode( DATA_PIN, INPUT ); // change direction to input
        digitalWrite( DATA_PIN, HIGH ); // enable internal pull-up
        // AM2302 will send a response signal of 40-bit data that
        // represent the relative humidity and temperature information to MCU.
```

```
        unsigned long t1, t0 = micros();
 while (1) {
        new_state = digitalRead( DATA_PIN );
        if ( state != new_state ) {
        t1 = micros();
if ( (state == HIGH) && (i > 2) ) {
        byte b = ( (t1-t0) > 40 ) ?  1 : 0;
        data[j/8] <<= 1;
        data[j/8] |= b;
        j++;
        }
        i++;
        state = new_state;
        t0 = t1;
        count = 0;
} else {
        count++;
if ( count > 1000 ) // timeout
         break;
        }
        }
        byte check_sum = 0x00;
for (byte x=0; x < 4; x++) {
        check_sum += data[x];
        }
if ( check_sum != data[4] ) {
        Serial.println( "CHECKSUM error" );
} else {
        Serial.print( ((data[0] << 8) | data[1])/10.0 );
        Serial.print( "%RH, " );
        Serial.print( ((data[2] << 8) | data[3])/10.0 );
        Serial.println( " C" ); } delay(2500);}
```



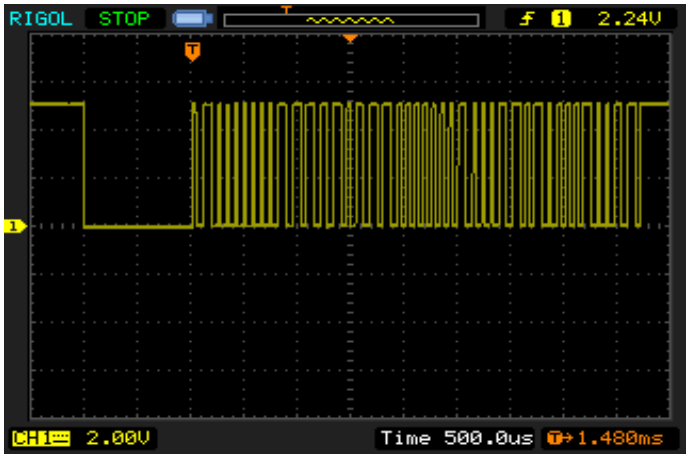
รูปที่ 10.7 การต่อวงจรการใช้งาน DHT 22 กับ Arduino บอร์ด

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)



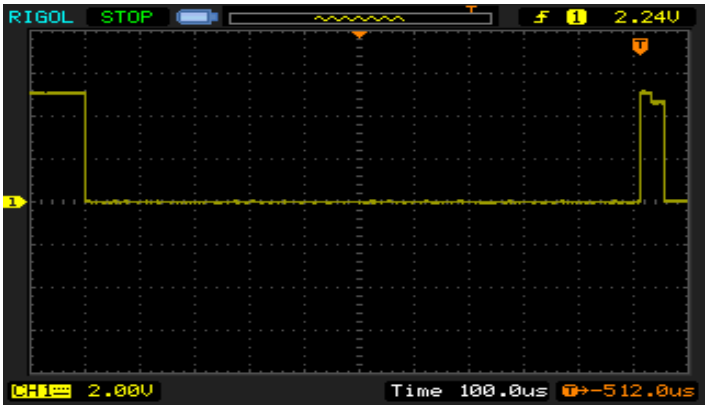
รูปที่ 10.8 ค่าที่วัดได้เมื่อดูผ่านทาง Serial Monitor

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)



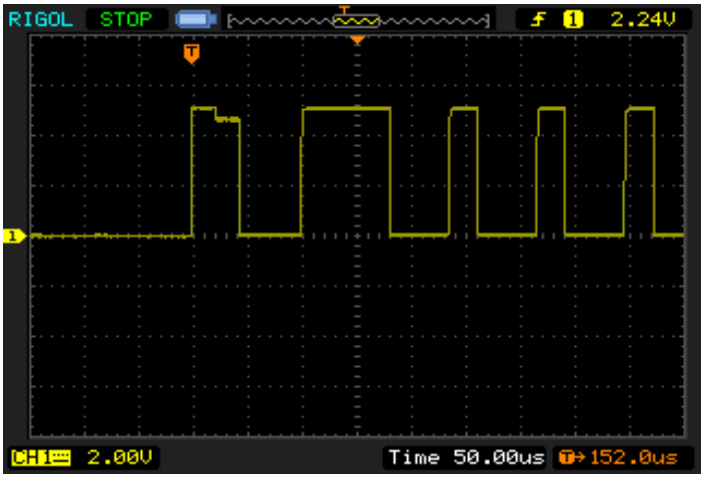
รูปที่ 10.9 สัญญาณ DATA เมื่อวัดด้วยออสซิลโลสโคป

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)



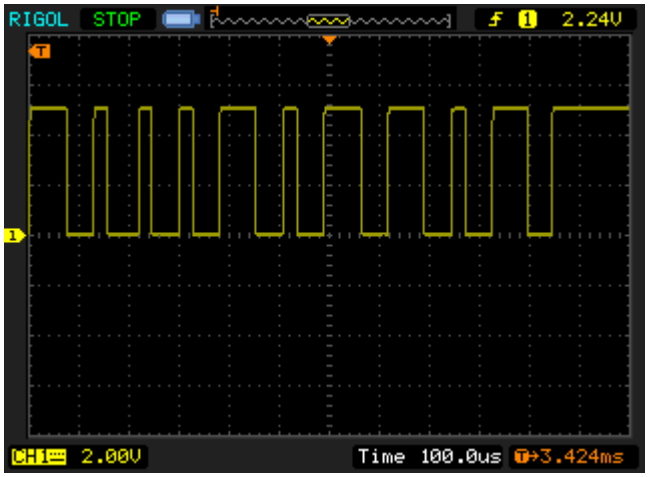
รูปที่ 10.10 แสดงให้เห็นช่วง LOW (Start) ประมาณ1000 μsec

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)



รูปที่ 10.11 ช่วง LOW และ HIGH ของ Start Bit

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)



รูปที่ 10.12 ช่วง LOW และ HIGH ของ Response Bit

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

โปรแกรมที่ 10.2 Arduino Sketch 2

วิธีสร้างอินเทอร์รัพท์ EINT0 และเขียน ISR (Interrupt Service Routine) เพื่อใช้ในการอ่านข้อมูลที่ถูกส่งมาจากไอซี DHT22

Sourcecode: dht22_eint0_reading.ino

```
// Author: RSP @KMUTNB
// Date: 22-April-2014
// Target Board: Arduino Uno (ATmega328P, 5V, 16MHz)
// Arduino IDE: version 1.0.5
// Description:
//   This Arduino Sketch demonstrates how to read data from DHT22 (AM2302)
//   -- a digital relative humidity and temperature sensor manufactured by
//   Aosong Electronics Co.,Ltd.
//   It utilizes the EINT0 interrupt and the corresponding ISR to read
//   data bits from the IC after sending the start bit.
const byte DATA_PIN = 2; // connected to the DATA pin of DHT22 (AM2302)
volatile boolean flag = false;
volatile uint8_t data[5];
volatile uint8_t bit_count = 0;
void eint_isr() { // ISR for EINT0
  static uint32_t tH, tL = 0L;
  if ( digitalRead( DATA_PIN ) ) { // HIGH
```

```
        tH = micros();
         if ( bit_count >= 42 ) {
        flag = true;
        bit_count = 0;
        }
} else { // LOW
        tL = micros();
        uint8_t b = ((tL - tH) > 40) ? 1 : 0;
 if ( bit_count >= 2 ) { // skip the first two bits (start and response bits)
         uint8_t byte_index = (bit_count-2)/8;
         data[ byte_index ] <<= 1;
         data[ byte_index ] |= b;
        }
        bit_count++;
        }
        }
void setup() {
        pinMode( DATA_PIN, INPUT );
        digitalWrite( DATA_PIN, HIGH );         // enable internal pull-up
        Serial.begin( 115200 );                 // use serial port (baudrate = 115200)
        }
void dht22_send_start_bit() {
        bit_count = 0;
        flag = false;
        pinMode( DATA_PIN, OUTPUT );            // change direction to output
        digitalWrite( DATA_PIN, LOW );          // output low (send the start bit)
        delayMicroseconds( 1000 );
        digitalWrite( DATA_PIN, HIGH );         // output high
        delayMicroseconds( 40 );
        pinMode( DATA_PIN, INPUT );             // change direction to input
        digitalWrite( DATA_PIN, HIGH );         // enable internal pull-up
        attachInterrupt( 0, eint_isr, CHANGE ); }  // enable EINT0 interrupt
```

```
        boolean dht22_read_data( int16_t *humidity, int16_t *temperature ) {
 if (!flag) { return false; } // data not available
        flag = false;  // clear flag
        detachInterrupt( 0 ); // disable EINT on data pin
        uint8_t check_sum = 0x00;
for ( int x=0; x < 4; x++) { // calculate checksum
        check_sum += data[x];
        }
if ( check_sum == data[4] ) {
        *humidity    = (data[0] << 8) | data[1];
        *temperature = (data[2] << 8) | data[3];
        return true; // checksum OK
        }
        return false; // checksum error
        }
        char buf[20]; // used for sprintf()
void loop() {
         int16_t h, t;
        dht22_send_start_bit();
        while (!flag) { delay(10); }
        if ( dht22_read_data( &h, &t ) ) {
        sprintf( buf, "%d.%d%cRH, %d.%d C", h/10, h%10, '%', t/10, t%10 );
        Serial.println( buf );
 } else {
        Serial.println( "DHT22 Checksum error!" );
        }
        delay(2500);
        }
```

10.3 การใช้งานโมดูลวัดอุณหภูมิและความชื้นสัมพัทธ์ SHT11

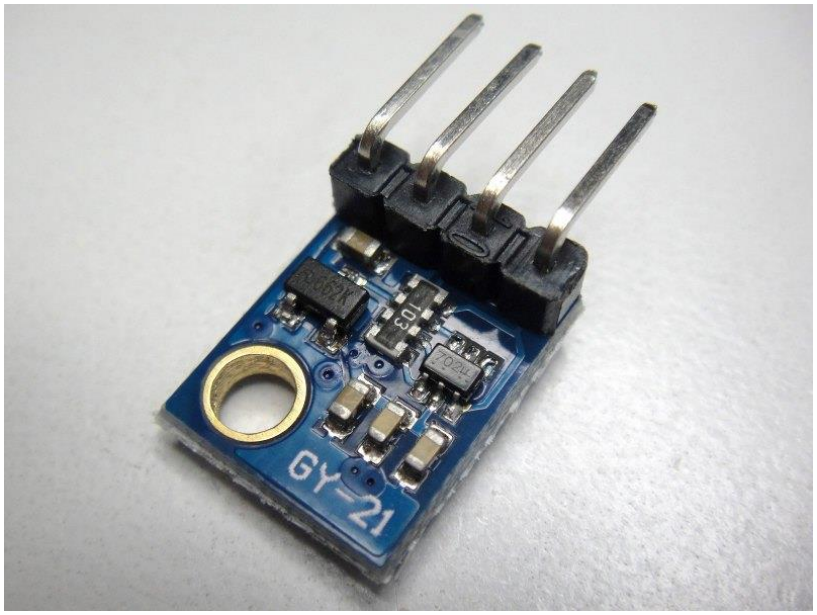
การทดลองใช้งาน **SHT11** ซึ่งเป็นเซนเซอร์สำหรับวัดค่าอุณหภูมิและความชื้นสัมพัทธ์ ผลิตโดยบริษัท **Sensirion** โดยนำมาใช้งานร่วมกับบอร์ด Arduino และเขียนโค้ด C++ เพื่อสร้างเป็นไลบรารี่ (Library) ไว้ใช้งาน

คำสำคัญ / Keywords: *SHT11, Digital Temperature and Relative Humidity Sensor, Arduino*

ปัจจุบันมีโมดูลเซนเซอร์สำหรับวัดค่าอุณหภูมิและความชื้นสัมพัทธ์ที่ให้ข้อมูลแบบดิจิทัลจากหลายผู้ผลิต ชิป **HTU21D** ของบริษัท Measurement Specialties Inc. ก็เป็นตัวเลือกหนึ่งสำหรับนำมาทดลองใช้ได้ แต่เนื่องจากชิปมีขนาดเล็ก แนะนำให้ใช้โมดูลประเภท "Breakout Board" ตัวอย่างของโมดูลที่สะดวกต่อการนำมาทดลองใช้งาน เช่น โมดูล **GY-21** ซึ่งมีราคาถูกและผลิตในประเทศจีน นอกจากนี้ยังมีโมดูลของบริษัท Adafruit และ Sparkfun ที่มีราคาสูงกว่า แต่มีคุณภาพและรายละเอียดของวงจรแตกต่างกันไป

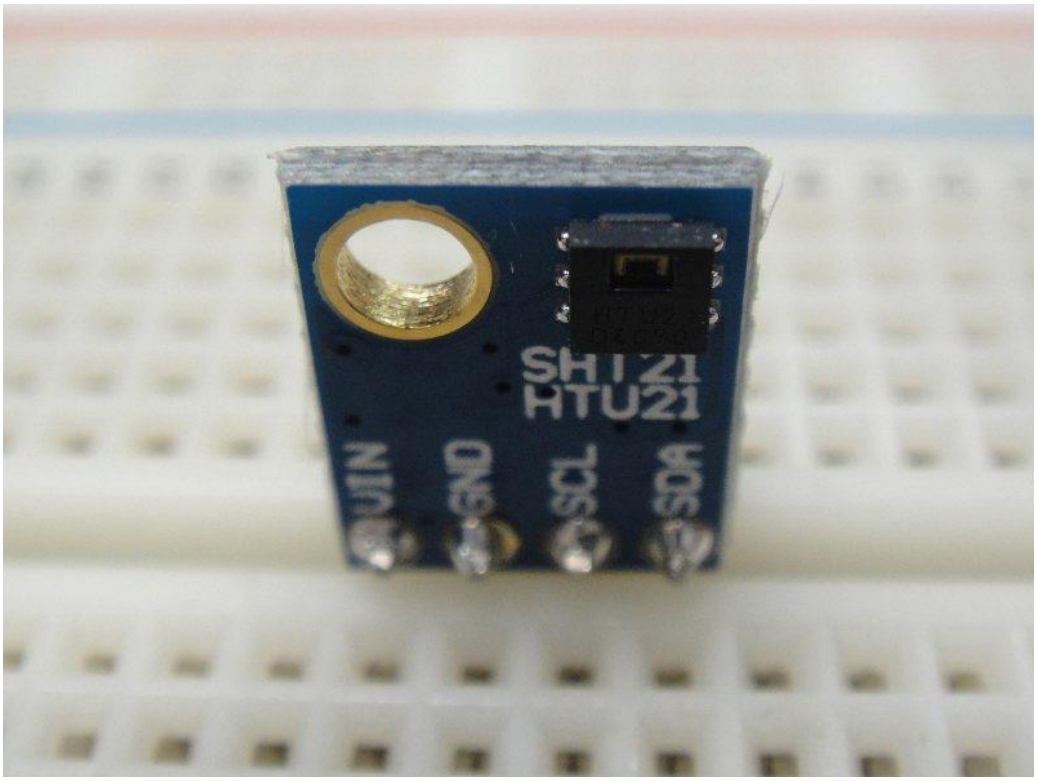
ข้อมูลเชิงเทคนิคที่สำคัญเกี่ยวกับโมดูล HTU21D

- ใช้แรงดันไฟเลี้ยง (Vcc): +1.5V .. +3.6V
- ติดต่อสื่อสารแบบบัส I2C (ใช้ความเร็ว 400kHz ได้)
- ใช้หมายเลขที่อยู่ 0x40 (hex) เพื่อเขียนหรืออ่านข้อมูลในรีจิสเตอร์ภายใน
- ช่วงค่าความชื้นสัมพัทธ์ (Humidity Operating Range): 0 .. 100 %RH
- ช่วงค่าอุณหภูมิ (Temperature Operating Range): -40 to +125°C
- ความละเอียดในการวัดความชื้นสัมพัทธ์ได้ถึง 12 บิต (ใช้เวลาในการวัดไม่เกิน 16 msec)
- ความละเอียดในการวัดอุณหภูมิได้ถึง 14 บิต (ใช้เวลาในการวัดไม่เกิน 50 msec)
- ความคลาดเคลื่อน ±3%RH, ±0.4°C tolerance @25°C (20%RH to 80%RH)



รูปที่ 10.13 โมดูล GY-21 HTU21D Breakout Board
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

โมดูล GY-21 มีไอซี 662K แปลงแรงดัน +5V ให้เป็น +3.3V ดังนั้นขา VIN สามารถป้อน +5V ได้ แต่ขา SDA และ SCL จะมีแรงดันลอจิกเป็น +3.3V



รูปที่ 10.14 โมดูล GY-21 (ด้านหลัง) สามารถมองเห็นชิป HTU21D
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

โปรแกรมที่ 10.3 Arduino Sketch 3

การอ่านค่าจากโมดูล GY-21 HTU21D Breakout Board โดยเชื่อมต่อกับบอร์ด Arduino ให้ต่อขา A4 (SDA) และ A5 (SCL) ผ่าน Logic Level Shifter ไปยังขา SDA และ SCL ของโมดูล GY-21 ตามลำดับ แล้วป้อนแรงดันไฟเลี้ยง 3.3V และต่อ GND เข้ากับระบบ

Sourecode: htu21d_demo.ino

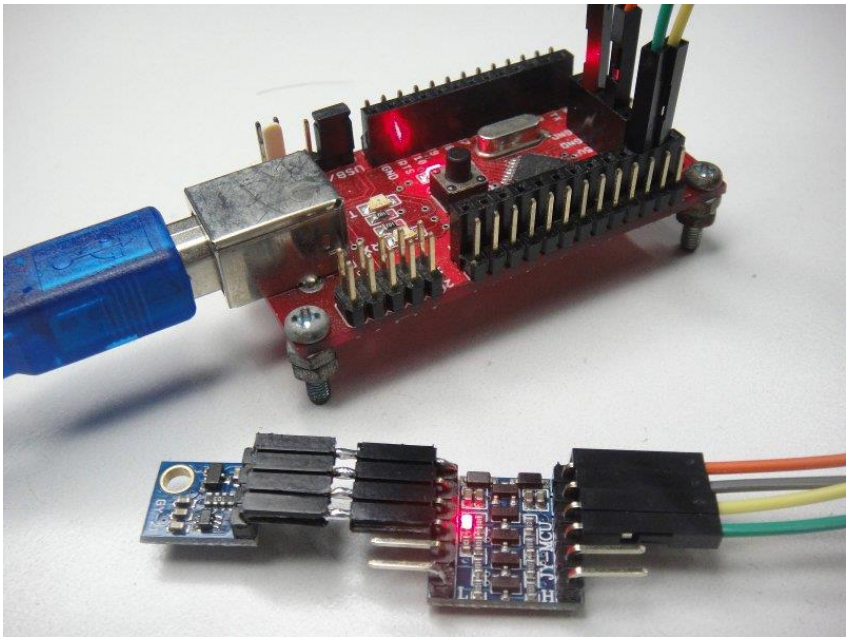
```
// Author: RSP @ Embedded System Lab (ESL), KMUTNB, Thailand
// Date: 2015-05-29
// Board: Arduino with ATmega168/328P (5V/16MHz)
// Arduino IDE: version 1.0.6
// Description:
//   This Arduino Sketch demonstates how to read relative humidity
//   and temperature values from the GY-21 HTU21D sensor module
//   by using an Arduino board.
#include <Wire.h> // use the Wire library
```

```
#define I2C_SLAVE_ADDR     (0x40)   // the 7-bit slave address of the HTU
//#define HTU21_READTEMP       0xE3 // Trigger Temperature Measurement, Hold Master
//#define HTU21_READHUM        0xE5 // Trigger Humidity Measurement, Hold Master
#define HTU21_READTEMP       0xF3
        // Trigger Temperature Measurement, No Hold Master
#define HTU21_READHUM        0xF5
        // Trigger Humidity Measurement, No Hold Master
#define HTU21_WRITEREG       0xE6 // Write User Register
#define HTU21_READREG        0xE7 // Read User Register
#define HTU21_RESET          0xFE // Soft Reset
// global variables
float temp, humid;
// used to hold the current temperature and relative humidty values
char sbuf[32];               // used for sprintf()
void setup() {
Serial.begin( 115200 );  // initialze the Serial
Wire.begin();            // initialize the Wire library before using the I2C bus
TWBR = 12;               // use 400kHz for I2C frequency
delay(1000);
i2c_scan();              // perform I2C slave device scanning
htu21d_reset();          // reset the HTU21D devie
}
void loop() {
Serial.print( "HTU21: " );
temp = htu21d_read_temperature();
 Serial.print( temp );
 Serial.print( " deg.C, " );
 humid = htu21d_read_humidity();
  humid += (25 - temp)*(-0.1); // compensated %RH
 Serial.print( humid );
  Serial.println( " %RH" );
delay(2500);
```

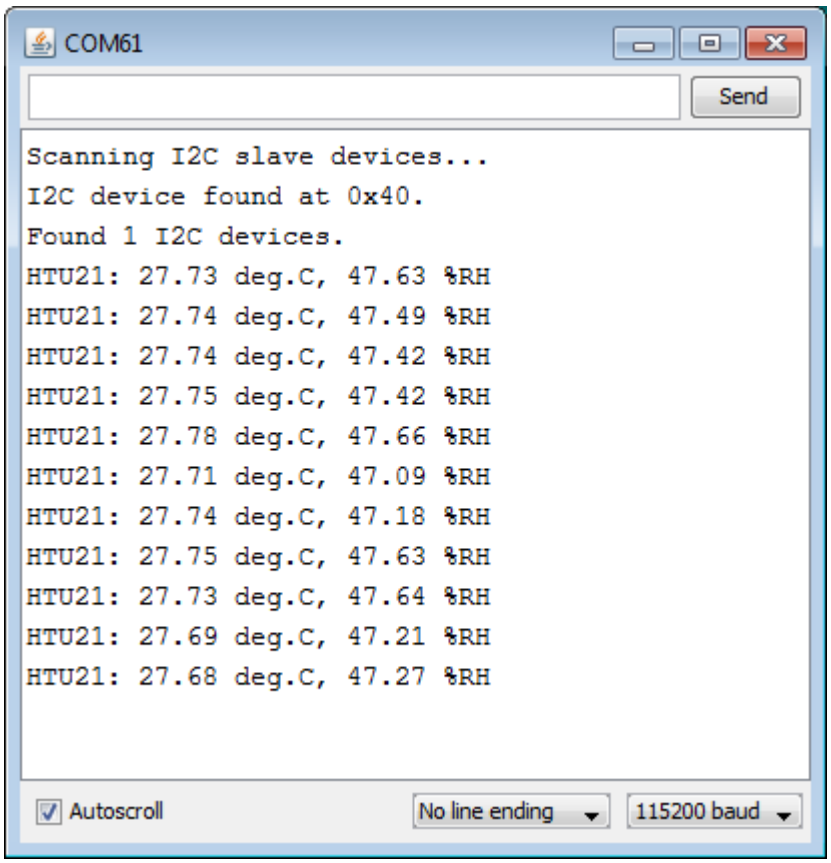
```
        }
void i2c_scan() { // scan I2C devices
        uint8_t count = 0;
        Serial.println( "Scanning I2C slave devices..." );
        delay(1);
for( uint8_t addr=0x01; addr <= 0x7f; addr++ ) {
        Wire.beginTransmission( addr );
 if ( Wire.endTransmission() == 0 ) {
        sprintf( sbuf, "I2C device found at 0x%02X.", addr );
        Serial.println( sbuf );
        count++;
        }
         }
if ( count > 0 ) {
        sprintf( sbuf, "Found %d I2C devices.", count );
} else {
        sprintf( sbuf, "No I2C device found." );
        }
        Serial.println( sbuf );
        }
        uint8_t htu21d_read_reg() {
        uint8_t data = 0x00;
        Wire.beginTransmission( I2C_SLAVE_ADDR );
        Wire.write( HTU21_READREG );
        Wire.endTransmission();
        Wire.requestFrom( I2C_SLAVE_ADDR, 1 );
if ( Wire.available() == 1 ) {
        data = Wire.read();
        }
        return data;
        }
```

```
void htu21d_write_reg( uint8_t data ) {
        Wire.beginTransmission( I2C_SLAVE_ADDR );
        Wire.write( HTU21_WRITEREG );
        Wire.write( data );
        Wire.endTransmission();
        }
        uint16_t htu21d_read_data( uint8_t reg_addr ) {
        uint16_t value = 0;
        uint8_t data[3];
        Wire.beginTransmission( I2C_SLAVE_ADDR );
        Wire.write( reg_addr );
        Wire.endTransmission();
        delay(50); // delay for 50 msec at least
        Wire.requestFrom( I2C_SLAVE_ADDR, 3 );
        uint32_t ts = millis();
while ( Wire.available() < 3 ) {
if ( (millis() - ts) > 10 ) {
        return value;
        }
        }
for ( uint8_t i=0; i < 3; i++ ) {
        data[i] = Wire.read();
        }
        value = data[0];
        value = (value << 8) | (data[1] & 0xFC);
if ( crc8( data, 2 ) ^ data[2] ) { // CRC8 checksum error
        Serial.println( "CRC8 checksum error" );
        value = 0;
        }
        return value;
        }
```

```
        float htu21d_read_temperature() {
        uint16_t T = htu21d_read_data( HTU21_READTEMP );
        // temperature = -46.85 + 175.72 * T/2^16 (see datasheet)
        return (-46.85 + (175.72 * T)/65536);
        }
        float htu21d_read_humidity(void) {
        uint16_t RH = htu21d_read_data( HTU21_READHUM );
         // rel_humidity = -6 + 125 * RH/2^16  (see datasheet)
        return (-6 + (125.0 * RH)/65536);
        }
void htu21d_reset(){
        Wire.beginTransmission( I2C_SLAVE_ADDR );
        Wire.write( HTU21_RESET );
        Wire.endTransmission();
        delay(15);
        }
        #define CRC8_INIT     0x00
#define CRC8_POLY     0x31 //  0x31 = 0b00110001 <=> x^8 + x^5 + x^4 + x^0
uint8_t crc8( uint8_t *data, uint16_t num_bytes ) { // calculate CRC-8 checksum
        uint8_t crc = CRC8_INIT;
for ( uint16_t i=0; i < num_bytes; i++ ) {
        crc ^= data[i];
 for ( uint8_t b=0; b < 8; b++ ) {
 if (crc & 0x80) {
        crc = (crc << 1) ^ CRC8_POLY;
        }
else {
        crc <<= 1;
        }}}
         return crc;
        }
```



รูปที่ 10.15 แสดงการต่อวงจรทดลองโดยใช้บอร์ด Arduino (328P, 5V,16MHz) และ โมดูล GY-21
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)



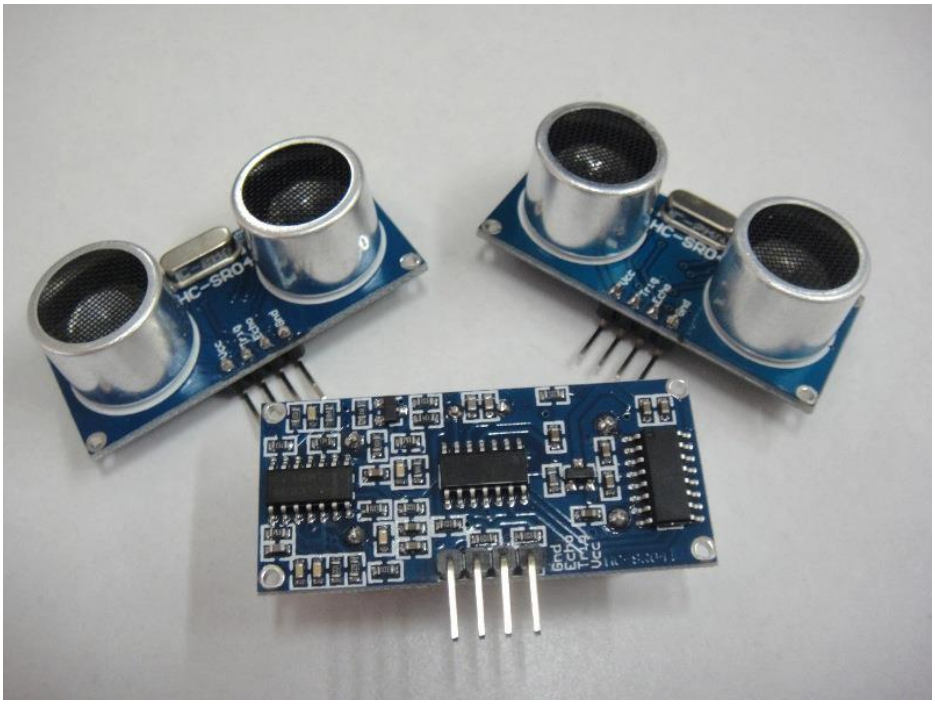
รูปที่ 10.16 ค่าที่อ่านได้จากโมดูล GY-21
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

10.4 การใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 วัดระยะห่างด้วยคลื่นอัลตร้าโซนิค

โมดูลสำหรับวัดระยะห่างด้วยคลื่นอัลตราโซนิค มีให้เลือกใช้งานแตกต่างกันไปแล้วแต่ผู้ผลิต คุณภาพ และราคา มีราคาถูกไม่กี่ร้อยบาท ไปจนถึงราคาเป็นพันบาท สามารถนำไปใช้ประยุกต์ใช้งานได้ เช่นการตรวจจับสิ่งกีดขวางสำหรับหุ่นยนต์เคลื่อนที่ บทความนี้จะกล่าวถึงการทดลองใช้งานโมดูล HC-SR04 วัดระยะห่างด้วยคลื่นอัลตร้าโซนิค ร่วมกับบอร์ด Arduino ในลักษณะเป็นโมดูลตรวจจับสัญญาณอินพุตแบบหนึ่ง

10.4.1 หลักการทำงานของโมดูลวัดระยะห่างด้วยคลื่นอัลตร้าโซนิค

โมดูล HC-SR04 เป็นอุปกรณ์อิเล็กทรอนิกส์ราคาถูก สำหรับวัดระยะห่างด้วยคลื่นอัลตราโซนิค (ใช้คลื่นเสียงความถี่ ประมาณ 40kHz) มีสองส่วนหลักคือ ตัวส่งคลื่นที่ทำหน้าที่สร้างคลื่นเสียงออกไปในการวัดระยะแต่ละครั้ง ("Ping") แล้วเมื่อไปกระทบวัตถุหรือสิ่งกีดขวาง คลื่นเสียงถูกสะท้อนกลับมายังตัวรับแล้วประมวลผลด้วยวงจรอิเล็กทรอนิกส์ภายในโมดูล ถ้าจับเวลาในการเดินทางของคลื่นเสียงในทิศทางไปและกลับ และถ้าทราบความเร็วเสียงในอากาศ ก็จะสามารถคำนวณระยะห่างจากวัตถุกีดขวางได้



รูปที่ 10.17 อุปกรณ์ HC-SR04

(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

โมดูล HC-SR04 ทำงานที่แรงดันประมาณ +5V (4.5V ถึง +5.5V) โดยป้อนให้ขา VCC และ GND โมดูลนี้มีขาสัญญาณดิจิทัล TRIG (อินพุต) และ ECHO (เอาต์พุต) ที่นำไปเชื่อมต่อกับไมโครคอนโทรลเลอร์ได้ อย่างเช่น Arduino ในการวัดระยะห่างแต่ละครั้ง จะต้องสร้างสัญญาณแบบ Pulse ที่มีความกว้าง (Pulse Width) อย่างน้อย 10 usec ป้อนให้ขา TRIG และหลังจากนั้นให้วัดความกว้างของสัญญาณช่วง HIGH จากขา ECHO ถ้าวัตถุอยู่ใกล้ความกว้างของสัญญาณ Pulse ที่ได้ก็จะน้อย แต่ถ้าวัตถุอยู่ไกลออกไป ก็จะได้ค่าความกว้างของสัญญาณ Pulse ที่

มากขึ้น การเลือกใช้งานโมดูลประเภทนี้ มีประเด็นที่สำคัญ เช่นช่วงระยะห่างของการวัด ความกว้างของมุมเมื่อคลื่นเสียงเดินทางออกไปจากตัวส่ง (เรียกว่า Beam Angle) นอกจากนั้นการสะท้อนกลับของคลื่นเสียงที่วัตถุกีดขวาง ขนาดและรูปทรงของวัตถุ และการสะท้อนกลับของเสียงจากหลายทิศทาง หรือต่างระยะกัน ก็มีผลต่อความถูกต้องหรือความผิดพลาดในการวัดค่าระยะห่างได้เช่นกัน

ข้อมูลเชิงเทคนิคของโมดูล HC-SR04

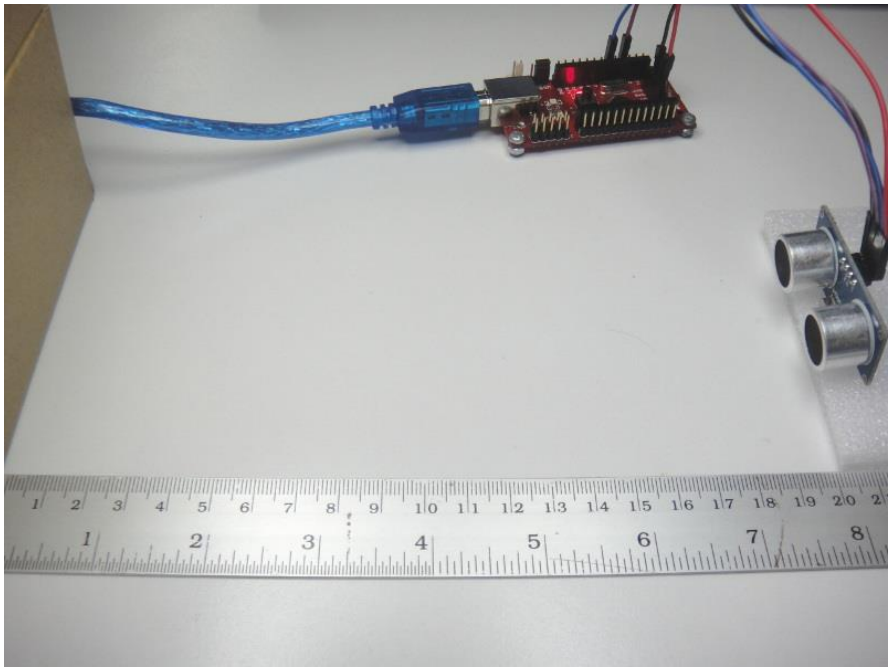
- ใช้แรงดันประมาณ +5V
- กินกระแสประมาณ 15mA
- ช่วงการวัดระยะทาง (measurement range): ประมาณ 4cm ถึง 4m
- ความกว้างเชิงมุมในการวัด (measuring angle): 15 องศา
- ความกว้างของสัญญาณ Pulse สำหรับ Trigger: 10 usec
- ระดับแรงดันลอจิกสำหรับขา TRIG และ ECHO: 5V TTL

โปรแกรมที่ 10.4 Arduino Sketch 4

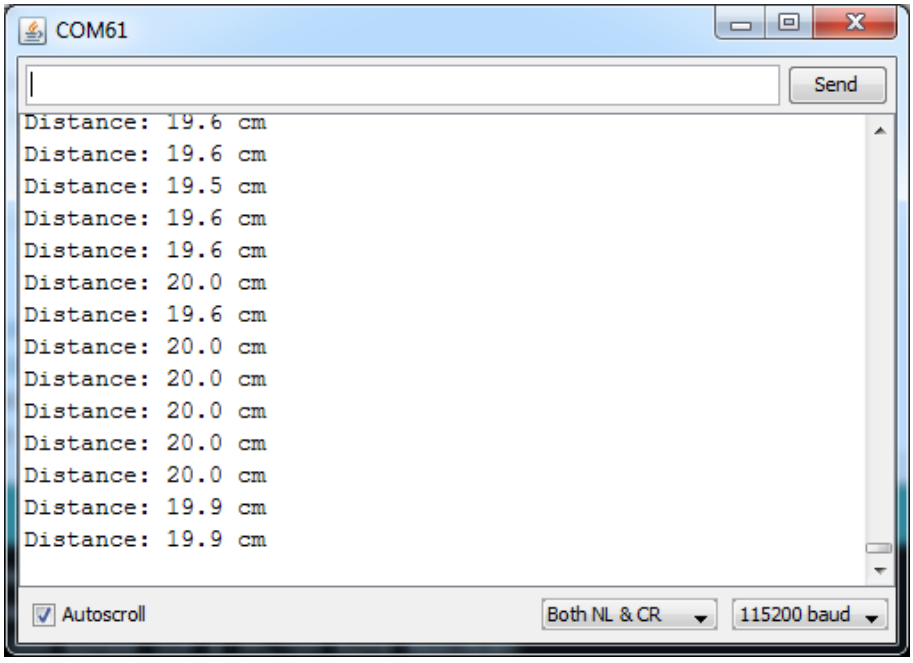
โค้ดนี้จะทำให้ Arduino ส่งสัญญาณ Pulse ความกว้างอย่างน้อย 10 usec ออกไปที่ขาเอาต์พุต TRIG จากนั้นจึงวัดความกว้างของสัญญาณ Pulse ที่เป็นอินพุตจากขา ECHO โดยใช้คำสั่ง pulseIn() ซึ่งเป็นคำสั่งของ Arduino และจะได้ค่าเป็นจำนวนเต็ม (หน่วยเป็นไมโครวินาที) จากนั้นนำค่าที่ได้มาคำนวณเป็นระยะทาง (หน่วยเป็นมิลลิเมตร) ในกรณีที่ได้ค่ามากกว่า 4000 (เกิน 4m หรือ 400cm) จะทำการวัดค่าใหม่ จนกว่าจะได้ค่าระยะห่างไม่เกิน 4000 เมื่อได้ค่าที่ถูกต้อง จะแสดงค่าทาง Serial แล้วเว้นระยะเวลาประมาณ 300 msec (0.3 วินาที) แล้วทำขั้นตอนซ้ำ

```
// Author: RSP @ ESL (Embedded System Lab), KMUTNB
// Date: 16-Jul-2013
// Target Board: Arduino Uno (ATmega328P, 5V, 16MHz)
// Arduino IDE: version 1.0.5
// HC-SR04 Ultrasonic module (using VCC=5V)
#define ECHO_PIN  3  // Echo Pin (Input) -- from the ECHO pin of HC-SR04
#define TRIG_PIN  5  // Trigger Pin (Output) -- to the TRIG pin of HC-SR04
void setup() {
pinMode( TRIG_PIN, OUTPUT );
pinMode( ECHO_PIN, INPUT );
digitalWrite( TRIG_PIN, LOW ); // output LOW to the TRIG pin
Serial.begin( 115200 ); // initialize serial, use baudrate=115200
}
```

```
void loop() {
        unsigned long duration_usec;
        unsigned long distance_mm;
        // v = 340 m/s = (340 * 100)/10^6 cm/usec = 34/1000 cm/usec
        // 2*d = v*t => d = v*t/2 = (17*t)/1000 cm = (17*t)/100 mm.
while (1) {
        duration_usec = ping();
        distance_mm = (17*duration_usec)/100;
 if ( distance_mm > 4000 ) { // out of range (beyond 4 meters)
        // Serial.println( "Out of range!" );
        continue;
        }
        Serial.print( "Distance: " );
        Serial.print( distance_mm / 10 );
        Serial.print( '.' );
        Serial.print( distance_mm % 10 );
        Serial.println( " cm" );
        break;
        }
        delay(300);
        }
        unsigned long ping() {
        // send a pulse (at least 10 usec long) to the TRIG pin
        digitalWrite( TRIG_PIN, HIGH );
        delayMicroseconds( 12 );
        digitalWrite( TRIG_PIN, LOW );
        // measure the ECHO pulse width (in microseconds)
        unsigned long duration_usec = pulseIn( ECHO_PIN, HIGH );
        return duration_usec;
        }
```



รูปที่ 10.18 การวัดระยะห่างจากวัตถุกีดขวางในระยะใกล้ (20 cm)
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)



รูปที่ 10.19 ค่าที่วัดได้เมื่อดูผ่านทาง Serial Monitor ของ Arduino IDE
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

โปรแกรมที่ 10.5 Arduino Sketch 5

สาธิตการใช้อินเทอร์รัพท์ภายนอก (External Interrupt) หมายเลข 0 (ซึ่งตรงกับขา D2 ของบอร์ด Arduino Uno) เพื่อดูการเปลี่ยนแปลงที่ขาอินพุต ซึ่งรับสัญญาณมาจากขา ECHO หลังจากส่งที่สัญญาณแบบ Pulse เป็นเอาต์พุตที่ขา TRIG ออกไป

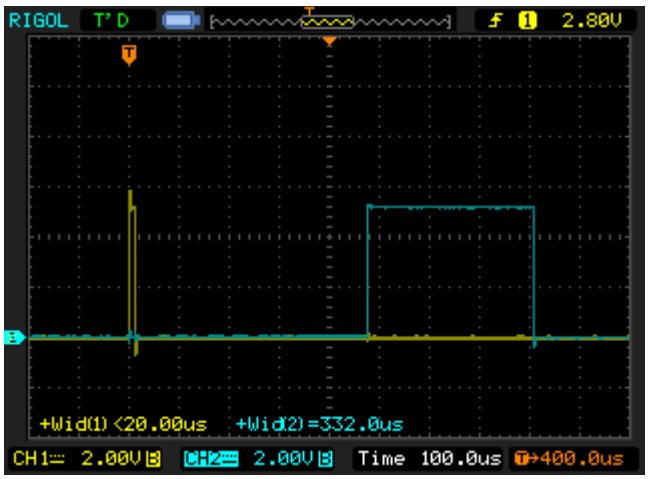
```
// Author: RSP @ ESL (Embedded System Lab), KMUTNB, Bangkok/Thailand
// Date: 24-May-2015
// Target Board: Arduino Uno (ATmega328P, 5V, 16MHz)
// Arduino IDE: version 1.0.6
// Ultrasonic module: HC-SR04 (VCC=+5V)
// Description:
//   This Arduino sketch shows how to measure the distance (in cm.)
//   from the obstacle using an ultrasonic sensor.
//   The Arduino sends the TRIG signal on the D4 pin.
//   The ECHO signal is connected to the D2 pin which will raise
//   an external interrupt every time the logic level of D2 pin is changed.
#define MAX_DISTANCE_IN_MM       (4000)   // max. valid value for distance
#define DURATION_TO_DISTANCE(x)  ((17*(x))/100) // usec -> mm.
const int ECHO_PIN = 2; // D2 pin (External Interrupt 0)
const int TRIG_PIN = 4; // D4 pin
const int LED_PIN = 13; // D13 pin
// global variables
volatile uint32_t tH, tL, pulse_width = 0;
void setup(){
    pinMode( LED_PIN, OUTPUT );
    digitalWrite( LED_PIN, LOW );
    pinMode( ECHO_PIN, INPUT );
    pinMode( TRIG_PIN, OUTPUT );
    digitalWrite( TRIG_PIN, LOW );
    // use the External Interrupt 0
    attachInterrupt( 0, eint_isr, CHANGE ); // D2 pin (EINT0)
    Serial.begin( 115200 );  // initialize the Serial port
    delay( 1000 );}
```

```
        uint16_t dist_mm; // distance in mm.
        char sbuf[32];    // used for sprintf()
        uint16_t read_ultrasonic_sensor() {
        digitalWrite( LED_PIN, HIGH );
        digitalWrite( TRIG_PIN, HIGH );
        delayMicroseconds( 12 );
        digitalWrite( TRIG_PIN, LOW );
        pulse_width = 0;
while (pulse_width == 0 ) {}  // wait until pulse_width > 0
        digitalWrite( LED_PIN, LOW );
        return DURATION_TO_DISTANCE( pulse_width );
        }
void loop() {
        d_ultrasonic_sensor();
        if ( dist_mm > MAX_DISTANCE_IN_MM ) {
        Serial.println( "Out of range." );
} else {
        sprintf( sbuf, "Distance: %d.%1d cm", (dist_mm/10), (dist_mm%10) );
        Serial.println( sbuf );
        }
        delay( 200 );
        }
        volatile uint32_t timestamp;
void eint_isr() { // ISR for Ext. Interrupt
        timestamp = micros();  // read the timestamp (in microseconds)
        if ( digitalRead( ECHO_PIN ) ) { // HIGH
        tH = timestamp;
        } else { // LOW
        tL = timestamp;
        pulse_width = (tL-tH);
        }}
```

10.5 การใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 หลายชุด

วิธีการวัดระยะห่างจากวัตถุด้วยโมดูลอัลตร้าโซนิค HC-SR04 (Ultrasonic Distance Sensor) ซึ่งเป็นโมดูลเซนเซอร์ราคาถูก โดยนำมาต่อใช้งานร่วมกับไอซี PCF8574A (I2C) และบอร์ด Arduino Uno และสามารถใช้งานโมดูล HC-SR04 ได้พร้อมกันถึง 4 ตัว

คำสำคัญ / Keywords: *HC-SR04, Ultrasonic Distance Sensors, Multiple Ultrasonic Sensor Reading*การทำงานของโมดูลอัลตร้าโซนิค จะอาศัยการส่งคลื่นเสียงความถี่สูงออกไป เช่น ประมาณ 40kHz และจับเวลาในการเดินทางของคลื่นเสียงเมื่อไปและกลับมาหลังจากสะท้อนวัตถุกีดขวาง โดยปรกติ จะใช้ไมโครคอนโทรลเลอร์อย่างเช่น บอร์ด Arduino สร้างสัญญาณแบบ Pulse ที่มีความกว้างอย่างน้อย 10 ไมโครวินาที ซึ่งต่อกับขา TRIG ของโมดูล และรอดูว่ามีสัญญาณแบบ Pulse ตอบกลับจากโมดูลที่ขา ECHO แล้วจึงวัดความกว้างของสัญญาณ Pulse ดังกล่าว นำค่าที่ได้มาคำนวณเป็นระยะห่างจากวัตถุ



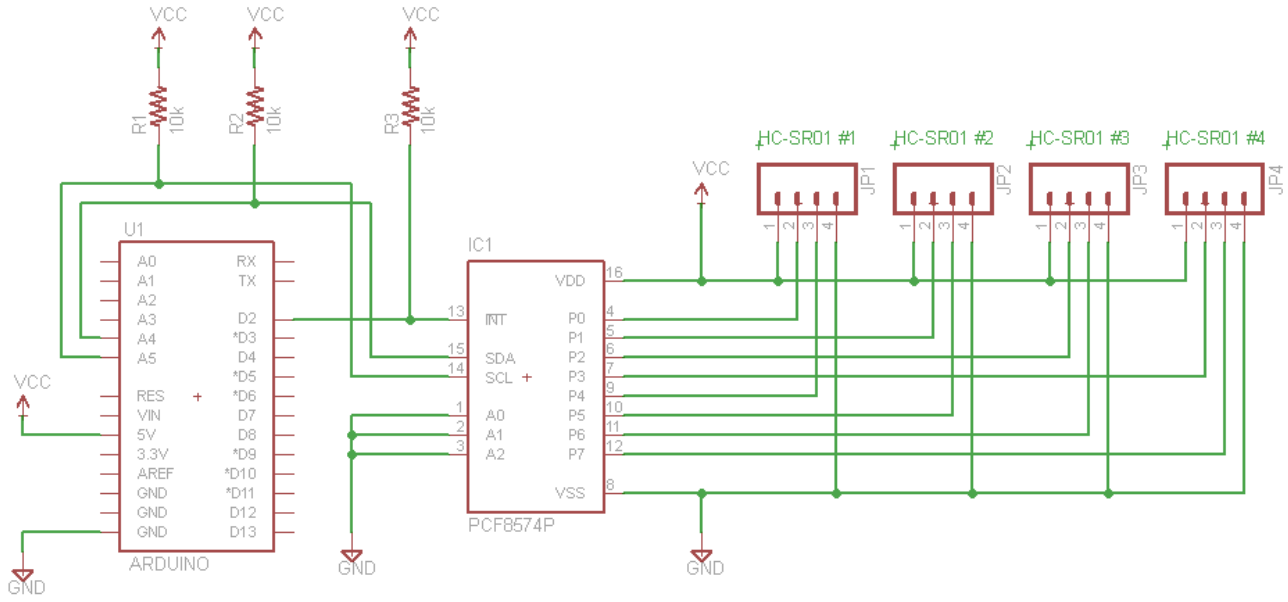
รูปที่ 10.20 แสดงตัวอย่างคลื่นสัญญาณที่วัดได้ด้วยเครื่องออสซิลโลสโคป TRIG (<20usec)
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

การวัดความกว้างของสัญญาณ Pulse อาจใช้คำสั่ง pulseIn() ของ Arduino Library หรือใช้หลักการทำงานของอินเทอร์รัพท์ภายนอก (External Interrupt) แล้วจับเวลา (มีฐานเวลาในการนับ เช่นหน่วยเป็นไมโครวินาที) เมื่อเกิดขอบขาขึ้นและขอบขาลงของสัญญาณอินพุตที่เข้ามาตามลำดับ

ถ้าต้องการใช้งานโมดูล HC-SR04 มากกว่าหนึ่งตัว โดยอ่านค่าจากโมดูลทีละตัว และนำมาต่อใช้งานร่วมกับ บอร์ด Arduino Uno จะมีวิธีการใดบ้าง ลองมาดูวิธีการต่อไปนี้ซึ่งใช้ไอซี PCF8574A I2C I/O Port Expander จำนวน 1 ตัว ที่เชื่อมต่อด้วยบัส I2C (ขา SDA และ SCL) โดยแบ่งพอร์ต I/O ขนาด 8 บิต เป็น 2 ส่วน ส่วนแรกเป็น 4 บิตแรก ใช้สำหรับเอาต์พุต (ขา P0..P3) และส่วนที่สองเป็น 4 บิตที่เหลือใช้เป็นอินพุต (ขา P4..P8) โดยนำไปต่อกับขา TRIG และ ECHO ของโมดูล HC-SR04 ได้ทั้งหมด 4 ตัว

การสร้างสัญญาณ TRIG จะกระทำไปทีละโมดูลตามลำดับ โดยการส่งค่าผ่านบัส I2C ให้ไปออกทางเอาต์พุต P0, P1, ..., P4 ของไอซี PCF8574A ไปตามลำดับ และจะมีเพียงขาเอาต์พุตเดียวในช่วงเวลาหนึ่งที่แอคทีฟ (active) หรือมีสัญญาณ Pulse ออกมา และเชื่อมต่อกับขา TRIG ของโมดูลที่ต้องการเลือกใช้งาน

เมื่อมีสัญญาณแบบ Pulse จากขา ECHO ของโมดูลตอบกลับมา (เข้าที่ขา P4, P5, ..., P8) จะทำให้สัญญาณ /INT ของไอซีมีการเปลี่ยนแปลงเชิงลอจิก (ต่อตัวต้านทานที่ขานี้ไว้แบบ Pull-up, ทำงานแบบ Active-Low) ถ้าวัดความกว้างของสัญญาณ /INT ในช่วงที่เป็น LOW จะสามารถนำไปคำนวณระยะห่างจากวัตถุที่วัดได้ ถ้านำขา /INT เป็นต่อกับขา D2 ของ Arduino Uno ก็สามารถใช้งานอินเทอร์รัพท์ภายนอก (External Interrupt 0) ได้



รูปที่ 10.21 ผังวงจรในการต่อวงจรโดยใช้บอร์ด Arduino Uno, PCF8574A และ HC-SR04 (ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

โปรแกรมที่ 10.6 Arduino Sketch 6

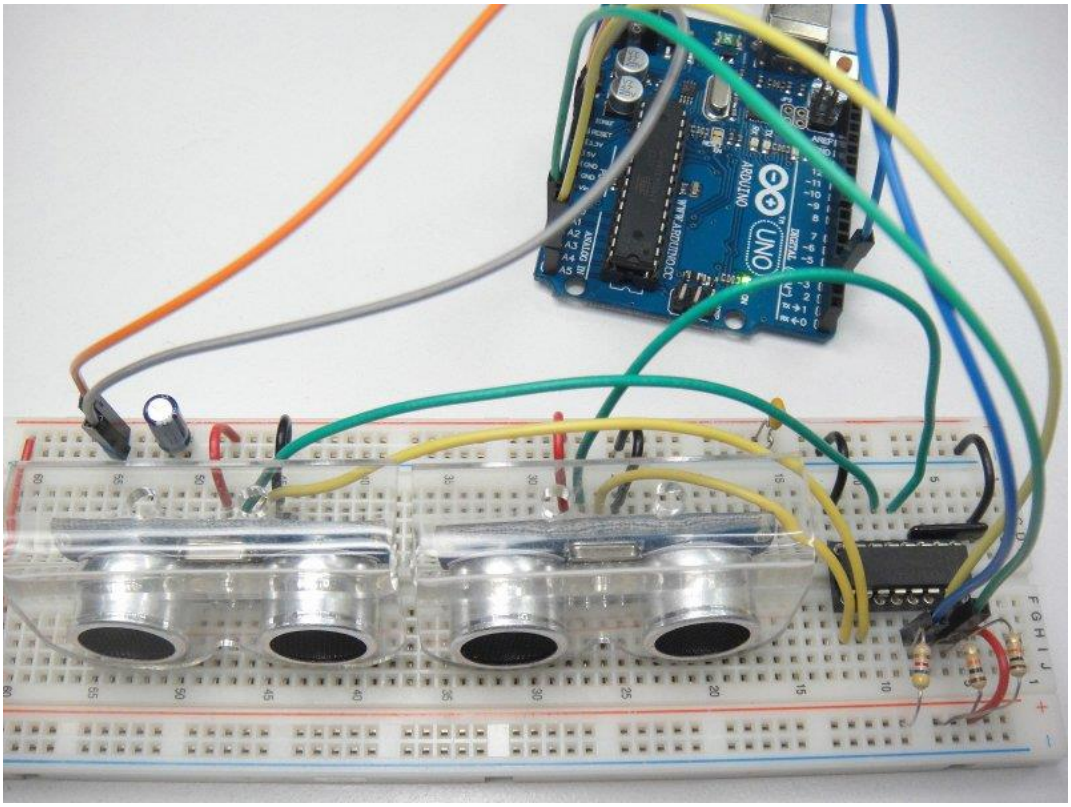
สาธิตการวัดระยะห่างจากวัตถุกีดขวาง โดยใช้โมดูลอัลตร้าโซนิค จำนวน 2 ชุด และต่อวงจรตามผังวงจรที่ให้ไว้

```
// Author: RSP @ Embedded System Lab (ESL), KMUTNB, Bangkok/Thailand
// Date: 2015-05-25
// Board: Arduino with ATmega168/328P (5V/16MHz)
// Arduino IDE: version 1.0.6
// Description:
//   This Arduino Sketch demonstrates how to use an Arduino Uno
```

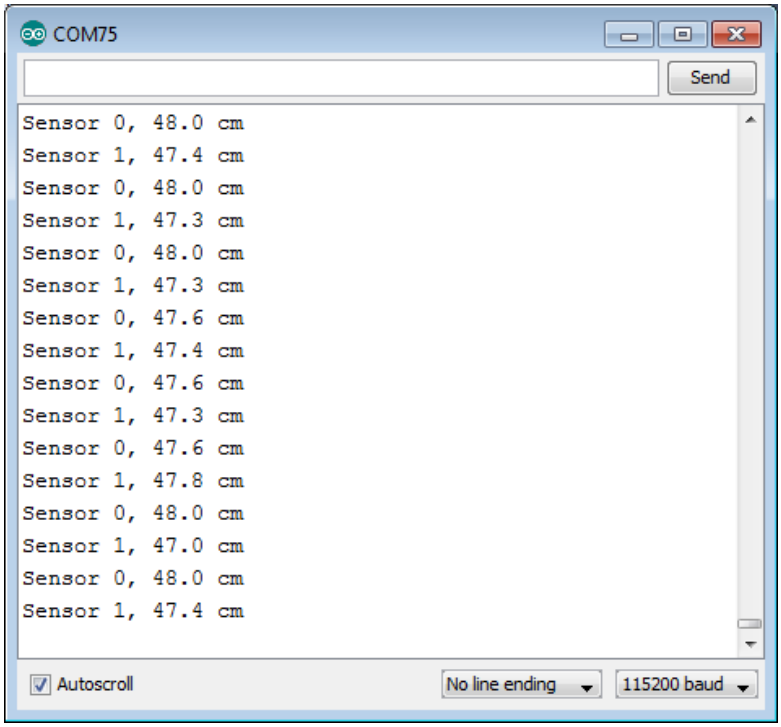
```
//   and a PCF8574A chip to interface with multiple ultrasonic distance
//   sensor modules (up to 4 modules).


#include <Wire.h> // use the Wire library
// connect address pins: A0=0 (GND), A1=0 (GND), A2=0 (GND)
#define ADDR_BITS (0B000) // A0=0,A1=0,A2=0
#define I2C_SLAVE_ADDR ((0B0111000) | ADDR_BITS) // 7-bit address
#define MAX_DISTANCE_IN_MM       (4000)   // max. valid value for distance
#define DURATION_TO_DISTANCE(x)  ((17*(x))/100) // usec -> mm.
const int ECHO_PIN = 2; // D2 pin (External Interrupt 0)
const int TRIG_PIN = 4; // D4 pin
const int LED_PIN = 13; // D13 pin
volatile uint32_t tH, tL, pulse_width = 0;
uint16_t dist_mm;       // distance in mm.
char sbuf[32];          // used for sprintf()
void setup() {
pinMode( LED_PIN, OUTPUT );
Wire.begin();
TWBR = 12; // for 400kHz
writeOutput(0xF0);
attachInterrupt( 0, eint_isr, CHANGE ); // D2 pin (EINT0)
Serial.begin( 115200 );
delay(1000);
}
void writeOutput( byte value ) { // write one byte
Wire.beginTransmission( I2C_SLAVE_ADDR );
Wire.write( value );
Wire.endTransmission();
}
byte readInput( void ) { // read one byte
byte data = 0xff;
```

```
        Wire.requestFrom( I2C_SLAVE_ADDR, 1 );
        delayMicroseconds(4);
if ( Wire.available() ) {
        data = Wire.read();}
        return data;
        }
        uint16_t read_ultrasonic_sensor( int pin ) {
        digitalWrite( LED_PIN, HIGH);
        writeOutput( 0xF0 | (1 << pin) );
        writeOutput( 0xF0 );
        pulse_width = 0;
while ( pulse_width == 0 ) {}  // wait until pulse_width > 0
        digitalWrite( LED_PIN, LOW );
        return DURATION_TO_DISTANCE( pulse_width );
        }
void loop() {
        for (int i=0; i < 2; i++ ) {
        dist_mm = read_ultrasonic_sensor( i );
if ( dist_mm > MAX_DISTANCE_IN_MM ) {
        Serial.println( "Out of range." );
} else {
        sprintf( sbuf, "Sensor %d, %d.%1d cm", i, (dist_mm/10), (dist_mm%10) );
        Serial.println( sbuf );
        }
        delay(5);
        }
        delay( 250 );
        }
        volatile uint32_t timestamp;
void eint_isr() { // ISR for Ext. Interrupt
        timestamp = micros();  // read the timestamp (in microseconds)
if ( digitalRead( ECHO_PIN ) == LOW ) {
```

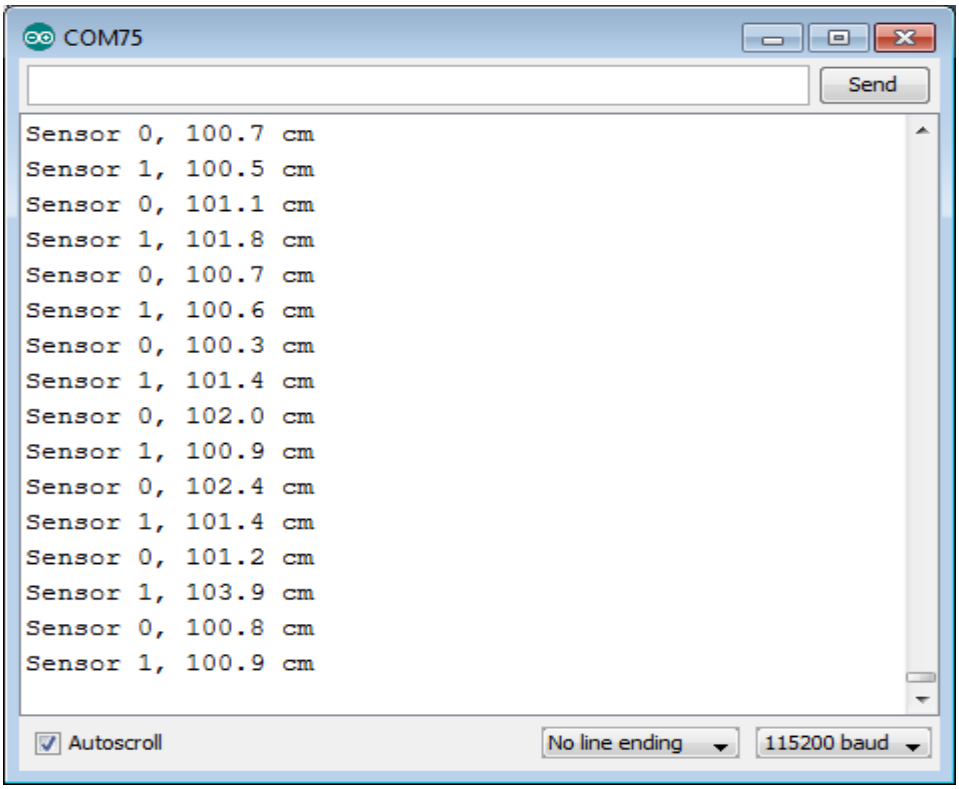
```
        tH = timestamp;
} else { // HIGH
        tL = timestamp;
        pulse_width = (tL-tH);}}
```



รูปที่ 10.22 แสดงการต่อวงจรทดลองโดยใช้บอร์ด Arduino Uno, ไอซี PCF8574A และโมดูล HC-SR04 (ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)



รูปที่ 10.21 แสดงค่าที่อ่านได้จากโมดูล HC-SR04 ทั้งสองตัว เมื่อหันหน้าไปทางเดียวกัน
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)



รูปที่ 10.22 แสดงค่าที่อ่านได้จากโมดูล HC-SR04 ทั้งสองตัว
(ที่มา www.Cpre.kmutnb.ac.th/esl/learning/index.php?article)

ที่ระยะห่างมากขึ้นจากวัตถุกีดขวาง เมื่อหันหน้าไปทางเดียวกัน และวัดระยะห่างจากสิ่งกีดขวางเดียวกัน จากวิธีการและรูปแบบการต่อวงจรดังที่ได้นำเสนอไป สามารถใช้ไอซี PCF8574A จำนวน 1 ตัว ได้กับโมดูล HC-SR04 ได้ถึง 4 ตัว ถ้าจะใช้งานโมดูล HC-SR04 เพิ่มอีก 4 ตัว จะต้องเพิ่มไอซี PCF8574A อีกหนึ่งตัว และนำมาต่อเข้ากับบัส I2C แต่จะต้องกำหนดแอดเดรสของอุปกรณ์ (Slave Device Address) ให้แตกต่างกัน และขา /INT ของไอซีตัวที่สอง จะต้องนำไปต่อกับขา External Interrupt อีกขาหนึ่งของ Arduino

สรุปเนื้อหาสาระสำคัญ

ปัจจุบันมีโมดูลเซนเซอร์สำหรับวัดค่าอุณหภูมิและความชื้นสัมพัทธ์ที่ให้ข้อมูลแบบดิจิทัลจากหลายผู้ผลิต ชิป HTU21D ของบริษัท Measurement Specialties Inc. ก็เป็นตัวเลือกหนึ่งสำหรับนำมาทดลองใช้ได้ แต่เนื่องจากชิปมีขนาดเล็ก แนะนำให้ใช้โมดูลประเภท "Breakout Board" ตัวอย่างของโมดูลที่สะดวกต่อการนำมาทดลองใช้งาน เช่นโมดูล GY-21 ซึ่งมีราคาถูกและผลิตในประเทศจีน นอกจากนี้ยังมีโมดูลของบริษัท Adafruit และ Sparkfun ที่มีราคาสูงกว่า แต่มีคุณภาพและรายละเอียดของวงจรแตกต่างกันไป ถ้าต้องการใช้งานโมดูล HC-SR04 มากกว่าหนึ่งตัว โดยอ่านค่าจากโมดูลทีละตัว และนำมาต่อใช้งานร่วมกับบอร์ด Arduino Uno ซึ่งใช้ไอซี PCF8574A I2C I/O Port Expander จำนวน 1 ตัว ที่เชื่อมต่อด้วยบัส I2C (ขา SDA และ SCL) โดยแบ่งพอร์ต I/O ขนาด 8 บิต เป็น 2 ส่วน ส่วนแรกเป็น 4 บิตแรก ใช้สำหรับเอาต์พุต (ขา P0..P3) และส่วนที่สองเป็น 4 บิตที่เหลือใช้เป็นอินพุต (ขา P4-P8) โดยนำไปต่อกับขา TRIG และ ECHO ของโมดูล HC-SR04 ได้ทั้งหมด 4 ตัว

เรื่อง การใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต ใช้เวลา 20 นาที

****

คำชี้แจง 1. แบบฝึกหัดมีทั้งหมด 2 ตอน ประกอบด้วยตอนที่ 1 และตอนที่ 2 (20 คะแนน)

2. แบบฝึกหัดตอนที่ 1 เป็นคำถามแบบถูก-ผิด มีทั้งหมด 20 ข้อ (10 คะแนน)

3. แบบฝึกหัดตอนที่ 2 เป็นคำถามแบบปรนัย มีทั้งหมด 10 ข้อ (10 คะแนน)

คำชี้แจง ให้ผู้เรียนกาเครื่องหมายถูก ✓ ในข้อที่คิดว่าถูก และกาเครื่องหมายผิด ✗ ในข้อที่คิดว่าผิด

คุณสมบัติของโมดูลวัดอุณหภูมิและความชื้น DHT22 / AM2302

_____ 1. อุปกรณ์เซนเซอร์สำหรับวัดอุณหภูมิและความชื้น

_____ 2. วัดอุณหภูมิได้ในช่วง: -40 to 100 °C (±0.5 °C accuracy)

_____ 3. วัดความชื้นสัมพัทธ์ได้ในช่วง: 0 - 80 RH% (2 - 5% accuracy)

_____ 4. อัตราการวัดสูงสุด: 50 Hz

_____ 5. คอนเนกเตอร์แบบ 4 ขา (0.1" / 2.54mm spacing)

_____ 6. Pin 1 = VCC

_____ 7. Pin 2 = SDA (Serial data, bidirectional)

_____ 8. Pin 3 = SCA

_____ 9. Pin 4 = GND

_____10. ใช้แรงดันไฟเลี้ยงได้ในช่วง: 3.3V ถึง 5.5V DC (ดังนั้นจึงใช้ได้กับ 3.3V และ 5V)

แบบฝึกหัดตอนที่ 2

1. โมดูล DHT21 (AM2301) เชื่อมต่อสายสัญญาณอย่างไร
   - ก. เชื่อมต่อด้วยสัญญาณเส้นเดียวแบบสองทิศทาง (bidirectional)
   - ข. เชื่อมต่อด้วยสัญญาณสองเส้นแบบสองทิศทาง (bidirectional)
   - ค. เชื่อมต่อด้วยสัญญาณสามเส้นแบบสองทิศทาง (bidirectional)
   - ง. เชื่อมต่อด้วยสัญญาณสี่เส้นแบบสองทิศทาง (bidirectional)
2. DHT21 (AM2301) ในการอ่านข้อมูลแต่ละครั้ง จะอ่านข้อมูลทั้งหมด 40 บิต แบ่งเป็น
   - ก. 8 บิตสำหรับค่าความชื้น 16 บิตสำหรับค่าอุณหภูมิ และ 16 บิตสำหรับตรวจ parity bits
   - ข. 16 บิตสำหรับค่าความชื้น 16 บิตสำหรับค่าอุณหภูมิ และ 8 บิตสำหรับตรวจ parity bits
   - ค. 16 บิตสำหรับค่าความชื้น 8 บิตสำหรับค่าอุณหภูมิ และ 16 บิตสำหรับตรวจ parity bits
   - ง. 16 บิตสำหรับค่าความชื้น 16 บิตสำหรับค่าอุณหภูมิ และ 16 บิตสำหรับตรวจ
3. DHT22 / AM2302 มีลำดับของข้อมูลบิตในการอ่านค่าจากไอซีทั้งหมดอย่างไร
   - ก. 5 ไบต์ (40 บิต)
   - ข . 6 ไบต์ (40 บิต)
   - ค. 7 ไบต์ (40 บิต)
   - ง. 8 ไบต์ (40 บิต)
4. SHT11 ผลิตโดยบริษัท
   - ก. Mitsubishi
   - ข. Samsung
   - ค. Sensirion
   - ง. Panasonic
5. SHT11 เป็นเซนเซอร์สำหรับวัดค่าอุณหภูมิและความชื้นสัมพัทธ์ เขียนโค้ดภาษาอะไรเพื่อสร้างเป็นไลบรารี่
   - ก. C
   - ข. basic
   - ค. C++
   - ง. Assembly
6. โมดูล HTU21D มีความละเอียดในการวัดความชื้นสัมพัทธ์ได้ถึง
   - ก. 12 บิต (ใช้เวลาในการวัดไม่เกิน 46 msec)
   - ข. 12 บิต (ใช้เวลาในการวัดไม่เกิน 36 msec)
   - ค. 12 บิต (ใช้เวลาในการวัดไม่เกิน 26 msec)
   - ง. 12 บิต (ใช้เวลาในการวัดไม่เกิน 16 msec)

7. โมดูล HTU21D มีความละเอียดในการวัดอุณหภูมิได้ถึง
   ก. 14 บิต (ใช้เวลาในการวัดไม่เกิน 20 msec)
   ข. 14 บิต (ใช้เวลาในการวัดไม่เกิน 30 msec)
   ค. 14 บิต (ใช้เวลาในการวัดไม่เกิน 40 msec)
   ง. 14 บิต (ใช้เวลาในการวัดไม่เกิน 50 msec)
8. โมดูล HTU21D มีความคลาดเคลื่อนเท่าใด
   ก. ±2%RH, ±0.4°C @25°C (20%RH to 80%RH)
   ข. ±3%RH, ±0.4°C @25°C (20%RH to 80%RH)
   ค. ±4%RH, ±0.4°C @25°C (20%RH to 80%RH)
   ง. ±5%RH, ±0.4°C @25°C (20%RH to 80%RH)
9. ตัวส่งคลื่นที่สร้างคลื่นเสียงออกไปในการวัดระยะแต่ละครั้งเรียกว่า
   ก. Bing
   ข. Ping
   ค. Sing
   ง. Ding
10. ความกว้างของมุมเมื่อคลื่นเสียงเดินทางออกไปจากตัวส่งเรียกว่า
   ก. absolute Beam Angle
   ข. Max Beam Angle
   ค. Beam Angle
   ง. Min Beam Angle

ปฏิบัติการทดลองหน่วยที่ 10

เรื่อง การใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต

**********คำ า

คำชี้แจง ให้ผู้เรียนทุกคนทำการทดลองตามปฏิบัติการทดลองหน่วยที่ 10 เรื่อง การใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต ใช้เวลา 180 นาที (20 คะแนน)

จุดประสงค์เชิงพฤติกรรม

1. สามารถใช้ฟังก์ชั่นได้ถูกต้อง
2. สามารถแก้ปัญหาในการทำงานของบอร์ด Arduino Uno R3 ได้
3. สามารถต่อใช้งานและอัพโหลดโปรแกรมให้กับบอร์ด Arduino Uno R3 ได้

อุปกรณ์การทดลอง

1. โปรแกรม Arduino IDE 1.6.9	1	โปรแกรม
2. สายโหลด USB Arduino Uno R3	1	เส้น
3. บอร์ด Arduino Uno R3	1	บอร์ด
4. สายต่อวงจร	1	ชุด
5. เครื่องคอมพิวเตอร์	1	เครื่อง
6. แผงต่อวงจร	1	ตัว

ข้อควรระวัง

1. ควรระวังไม่วางบอร์ด Arduino Uno R3 หรือชีลต่างๆ บนโต๊ะโลหะหรือที่วางที่เป็นโลหะเพราะอาจเกิดการลัดวงจรของภาคจ่ายไฟได้

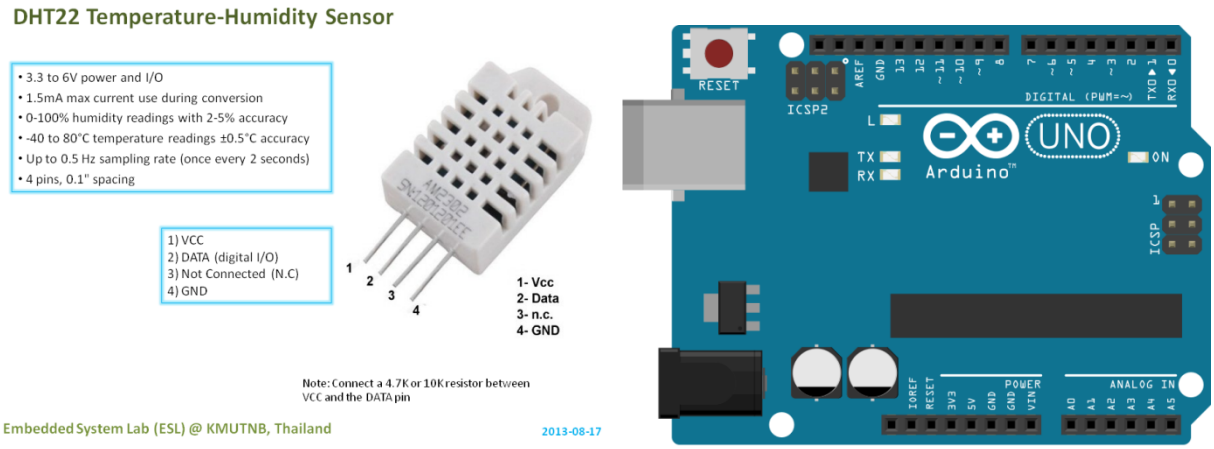
2. ไม่ควรต่อสายต่อวงจรในบอร์ด Arduino Uno R3 ทิ้งไว้ ควรถอดสายต่อวงจรออกให้หมด เพราะผลการทดลองอาจเกิดการผิดพลาดไม่เป็นไปตามทฤษฎีได้

3. ไม่ควรถอดสายสายโหลด USB เข้าออกตลอดเวลา เพราะอาจทำให้ภาคจ่ายไฟของบอร์ด Arduino Uno R3 เสียหายได้

การทดลองที่ 10.1 การอ่านข้อมูลจากโมดูลวัดอุณหภูมิและความชื้น DHT22 / AM2302

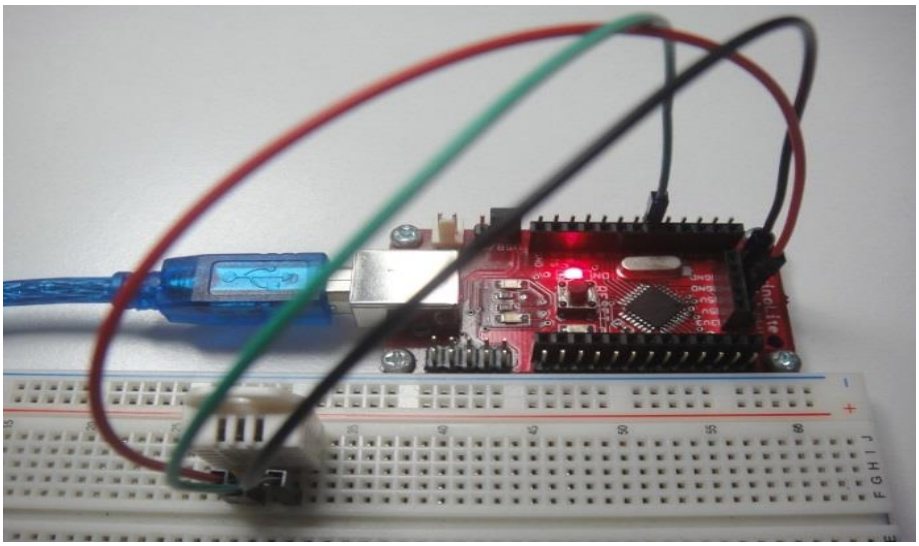
อุปกรณ์เซนเซอร์สำหรับวัดอุณหภูมิและความชื้นสัมพัทธ์ (Temperature & Relative Humidity Sensor) เป็นอุปกรณ์ที่สามารถนำมาประยุกต์ใช้งานทางด้านระบบสมองกลฝังตัวได้หลากหลาย การทดลองใช้งานโมดูล DHT22 / AM2302 ซึ่งมีราคาถูก ให้ค่าเป็นแบบดิจิทัล ใช้ขาสัญญาณดิจิทัลเพียงเส้นเดียวในการเชื่อมต่อแบบบิตอนุกรมสองทิศทาง (serial data, bi-directional) โดยนำมาเชื่อมต่อกับ Arduino เพื่ออ่านค่าจากเซนเซอร์

Hardware Required



รูปที่ 10.22 แสดงโมดูลวัดอุณหภูมิและความชื้น DHT22 / AM2302 + Arduino

Circuit / Schematics



รูปที่ 10.23 แสดงการต่อสายโมดูลวัดอุณหภูมิและความชื้น DHT22 / AM2302 + Arduino

CODE

```
const byte DATA_PIN = 2; // connected to the DATA pin of DHT22 (AM2302)
volatile boolean flag = false;
volatile uint8_t data[5];
volatile uint8_t bit_count = 0;
void eint_isr() { // ISR for EINT0
static uint32_t tH, tL = 0L;
if ( digitalRead( DATA_PIN ) ) { // HIGH
tH = micros();
if ( bit_count >= 42 ) {
flag = true;
bit_count = 0;
}
} else { // LOW
tL = micros();
uint8_t b = ((tL - tH) > 40) ? 1 : 0;
 if ( bit_count >= 2 ) { // skip the first two bits (start and response bits)
 uint8_t byte_index = (bit_count-2)/8;
 data[ byte_index ] <<= 1;
 data[ byte_index ] |= b;
}
bit_count++;
}
}
void setup() {
pinMode( DATA_PIN, INPUT );
digitalWrite( DATA_PIN, HIGH );          // enable internal pull-up
Serial.begin( 115200 );                  // use serial port (baudrate = 115200)
}
void dht22_send_start_bit() {
bit_count = 0;
flag = false;
```

```
        pinMode( DATA_PIN, OUTPUT );            // change direction to output
        digitalWrite( DATA_PIN, LOW );          // output low (send the start bit)
        delayMicroseconds( 1000 );
        digitalWrite( DATA_PIN, HIGH );         // output high
        delayMicroseconds( 40 );
        pinMode( DATA_PIN, INPUT );             // change direction to input
        digitalWrite( DATA_PIN, HIGH );         // enable internal pull-up
        attachInterrupt( 0, eint_isr, CHANGE ); }  // enable EINT0 interrupt
        boolean dht22_read_data( int16_t *humidity, int16_t *temperature ) {
 if (!flag) { return false; } // data not available
        flag = false;  // clear flag
        detachInterrupt( 0 ); // disable EINT on data pin
        uint8_t check_sum = 0x00;
for ( int x=0; x < 4; x++) { // calculate checksum
        check_sum += data[x];
        }
if ( check_sum == data[4] ) {
        *humidity    = (data[0] << 8) | data[1];
        *temperature = (data[2] << 8) | data[3];
        return true; // checksum OK
        }
        return false; // checksum error
        }
        char buf[20]; // used for sprintf()
void loop() {
        int16_t h, t;
        dht22_send_start_bit();
        while (!flag) { delay(10); }
        if ( dht22_read_data( &h, &t ) ) {
        sprintf( buf, "%d.%d%cRH, %d.%d C", h/10, h%10, '%', t/10, t%10 );
        Serial.println( buf );
 } else {
```

```
Serial.println( "DHT22 Checksum error!" );
}
delay(2500);
}
```

การทดสอบ

การอ่านค่าจากโมดูล DHT22 / AM2302 ด้วยบอร์ด Arduino แล้วนำค่าที่ได้แสดงผลผ่านทาง Serial Monitor ของ Arduino IDE (ตั้งค่า baudrate = 115200) แล้วทำขั้นตอนซ้ำ

ผลการทดลอง

.....

.....

.....

.....

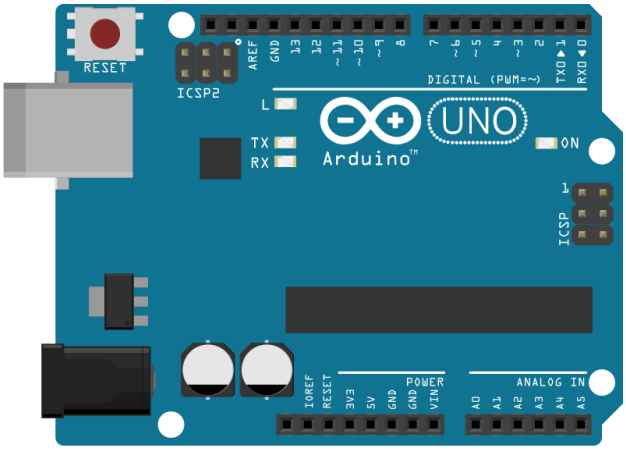
.....

.....

การทดลองที่ 10.2 การใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 หลายชุด

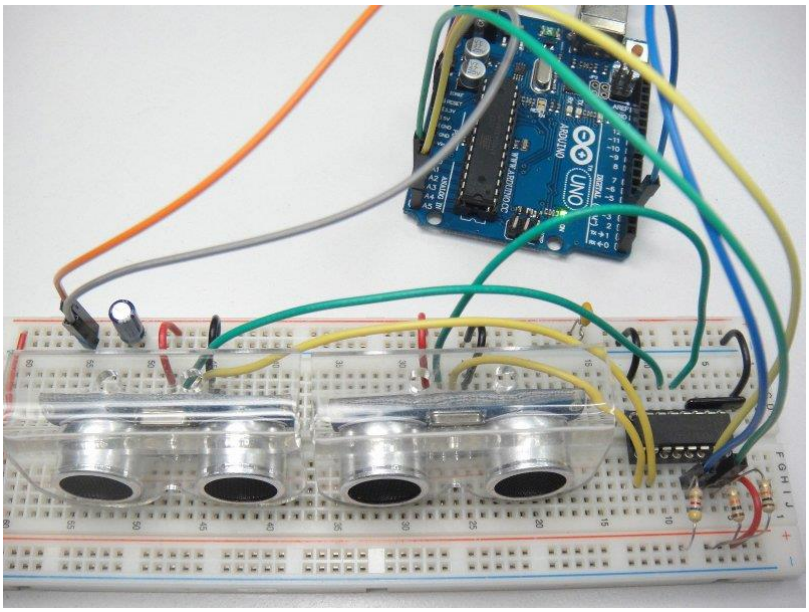
การทำงานของโมดูลอัลตร้าโซนิค จะอาศัยการส่งคลื่นเสียงความถี่สูงออกไป เช่น ประมาณ 40kHz และจับเวลาในการเดินทางของคลื่นเสียงเมื่อไปและกลับมาหลังจากสะท้อนวัตถุกีดขวาง โดยปรกติ จะใช้ไมโครคอนโทรลเลอร์อย่างเช่น บอร์ด Arduino สร้างสัญญาณแบบ Pulse ที่มีความกว้างอย่างน้อย 10 ไมโครวินาที ซึ่งต่อกับขา TRIG ของโมดูล และรอดูว่ามีสัญญาณแบบ Pulse ตอบกลับจากโมดูลที่ขา ECHO แล้วจึงวัดความกว้างของสัญญาณ Pulse ดังกล่าว นำค่าที่ได้มาคำนวณเป็นระยะห่างจากวัตถุ

Hardware Required



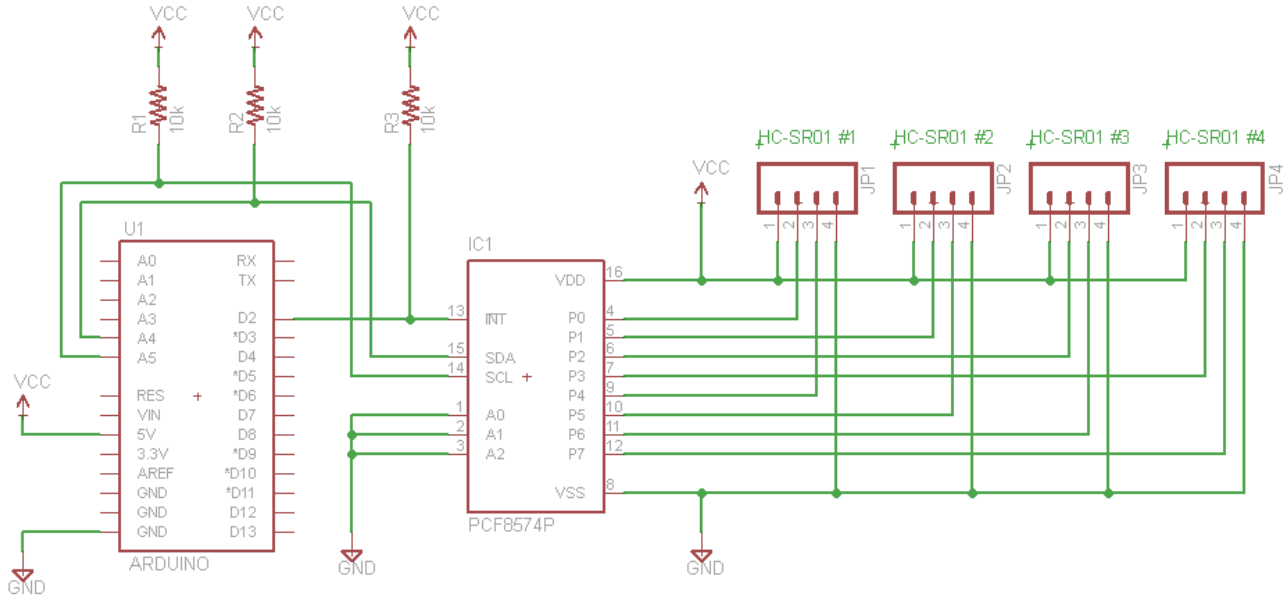
รูปที่ 10.24 Arduino UNO

Circuit



รูปที่ 10.25 การใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 หลายชุด

Schematics



Code

```
#include <Wire.h> // use the Wire library
// connect address pins: A0=0 (GND), A1=0 (GND), A2=0 (GND)
#define ADDR_BITS (0B000) // A0=0,A1=0,A2=0
#define I2C_SLAVE_ADDR ((0B0111000) | ADDR_BITS) // 7-bit address
#define MAX_DISTANCE_IN_MM      (4000)   // max. valid value for distance
#define DURATION_TO_DISTANCE(x)  ((17*(x))/100) // usec -> mm.
const int ECHO_PIN = 2; // D2 pin (External Interrupt 0)
const int TRIG_PIN = 4; // D4 pin
const int LED_PIN = 13; // D13 pin
volatile uint32_t tH, tL, pulse_width = 0;
uint16_t dist_mm;       // distance in mm.
char sbuf[32];          // used for sprintf()
void setup() {
    pinMode( LED_PIN, OUTPUT );
    Wire.begin();
    TWBR = 12; // for 400kHz
    writeOutput(0xF0);
```

```
        attachInterrupt( 0, eint_isr, CHANGE ); // D2 pin (EINT0)
        Serial.begin( 115200 );
        delay(1000);
        }
void writeOutput( byte value ) { // write one byte
        Wire.beginTransmission( I2C_SLAVE_ADDR );
        Wire.write( value );
        Wire.endTransmission();
        }
        byte readInput( void ) { // read one byte
        byte data = 0xff;
        Wire.requestFrom( I2C_SLAVE_ADDR, 1 );
        delayMicroseconds(4);
if ( Wire.available() ) {
        data = Wire.read();}
        return data;
        }
        uint16_t read_ultrasonic_sensor( int pin ) {
        digitalWrite( LED_PIN, HIGH);
        writeOutput( 0xF0 | (1 << pin) );
        writeOutput( 0xF0 );
        pulse_width = 0;
while ( pulse_width == 0 ) {}  // wait until pulse_width > 0
        digitalWrite( LED_PIN, LOW );
        return DURATION_TO_DISTANCE( pulse_width );
        }
void loop() {
        for (int i=0; i < 2; i++ ) {
        dist_mm = read_ultrasonic_sensor( i );
if ( dist_mm > MAX_DISTANCE_IN_MM ) {
        Serial.println( "Out of range." );
} else {
```

```
        sprintf( sbuf, "Sensor %d, %d.%1d cm", i, (dist_mm/10), (dist_mm%10) );
        Serial.println( sbuf );
        }
        delay(5);
        }
        delay( 250 );
        }
        volatile uint32_t timestamp;
void eint_isr() { // ISR for Ext. Interrupt
        timestamp = micros();  // read the timestamp (in microseconds)
if ( digitalRead( ECHO_PIN ) == LOW ) {
        tH = timestamp;
} else { // HIGH
        tL = timestamp;
        pulse_width = (tL-tH);}}
```

ผลการทดลอง

.....

.....

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ปัญหาอุปสรรคหรือข้อเสนอแนะ

.....

.....

ตารางการประเมินผลคะแนนภาคปฏิบัติ

หัวข้อการพิจารณาภาคปฏิบัติ	ระดับคะแนน
การทดลองที่ 10.1 การอ่านข้อมูลจากโมดูลวัดอุณหภูมิและความชื้น DHT22 / AM2302	10 คะแนน
การทดลองที่ 10.2 การใช้งานโมดูลตรวจจับสัญญาณอินพุต HC-SR04 หลายชุด	10 คะแนน
รวมคะแนนภาคปฏิบัติ	.....คะแนน

แบบทดสอบหลังเรียน หน่วยที่ 10

เรื่อง การใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต

เรื่อง การใช้งาน Arduino กับไอซีวัดอุณภูมิและโมดูลตรวจจับสัญญาณอินพุต **ใช้เวลา** 20 **นาที**

วิชา ไมโครคอนโทรลเลอร์เบื้องต้น **รหัสวิชา** (2127-2107)

ระดับชั้น ประกาศนียบัตรวิชาชีพ (ปวช.) **สาขาวิชา** เมคคาทรอนิกส์

****

คำชี้แจง 1. แบบทดสอบมีทั้งหมด 10 ข้อ (10 คะแนน)

2. ให้ผู้เรียนเลือกคำตอบที่ถูกที่สุดแล้วกาเครื่องหมายกากบาท (✕) ลงในกระดาษคำตอบ

1. โมดูล DHT21 (AM2301) เป็นเซนเซอร์วัดวัดอุณหภูมิและความชื้นสัมพัทธ์แบบใด
   - ก. แอนะล็อก
   - ข. ขนาน
   - ค. อนุกรม
   - ง. แบบดิจิตอล
2. DHT21 (AM2301) มีขาเชื่อมต่ออะไรบ้าง
   - ก. VCC, GND และ SLA
   - ข. VCC, GND และ SCA
   - ค. VCC, GND และ SDR
   - ง. VCC, GND และ SDA
3. DHT21 (AM2301) ต่อสายเชื่อมต่ออย่างไร
   - ก. สายสีแดงสำหรับ VCC สายสีดำคือ GND และสายสีเหลืองคือ SDA
   - ข. สายสีแดงสำหรับ GND สายสีดำคือ VCC และสายสีเหลืองคือ SDA
   - ค. สายสีแดงสำหรับ SDA สายสีดำคือ GND และสายสีเหลืองคือ VCC
   - ง. สายสีแดงสำหรับ VCC สายสีดำคือ SDA และสายสีเหลืองคือ GND
4. DHT22 / AM2302 ขาสัญญาณดิจิทัลเพียงเส้นเดียวในการเชื่อมต่อแบบ
   - ก. บิตอนุกรมหนึ่งทิศทาง (serial data, one-directional)
   - ข. บิตอนุกรมสองทิศทาง (serial data, bi-directional)
   - ค. บิตขนานสองทิศทาง (serial data, bi-directional)
   - ง. บิตผสมสองทิศทาง (serial data, bi-directional)

5. ค่า RH ที่อ่านได้คืออะไร

ก. ความชื้นสัมพัทธ์ (Relative Humidity - RH)

ข. ความชื้นสัมพันธ์ (Relative Humidity - RH)

ค. ความชื้นสัมผัส (Relative Humidity - RH)

ง. ความชื้นสัมประสิทธิ์ (Relative Humidity - RH)

6. โมดูล SHT11 ใช้แรงดันไฟเลี้ยงเท่าใด

ก. (Vcc): +0.5V .. +3.6V

ข. (Vcc): +1V .. +3.6V

ค. (Vcc): +1.5V .. +3.6V

ง. (Vcc): +2V .. +3.6V

7. โมดูล SHT11 ติดต่อสื่อสารแบบบัสแบบใด

ก. Uart

ข. I2C

ค. SPI

ง. AVI

8. โมดูล SHT11 ช่วงค่าความชื้นสัมพัทธ์ (Humidity Operating Range) คือ

ก. 0 ถึง 100 % RH

ข. -1 ถึง 100 % RH

ค. -5 ถึง 100 % RH

ง. 10 ถึง 100 % RH

9. โมดูลตรวจจับสัญญาณอินพุต HC-SR04 ใช้สำหรับทำอะไร

ก. วัดระยะทางด้วยคลื่นอัลตร้าไวโอเลต

ข. วัดระยะทางด้วยคลื่นอัลตร้าซูมเมอร์

ค. วัดระยะทางด้วยคลื่นอัลตร้าโซนิค

ง. วัดระยะทางด้วยคลื่นอัลตร้าบีม

10. โมดูล HC-SR04 ใช้คลื่นเสียงความถี่ประมาณ

ก. 10kHz

ข. 20kHz

ค. 30kHz

ง. 40kHz