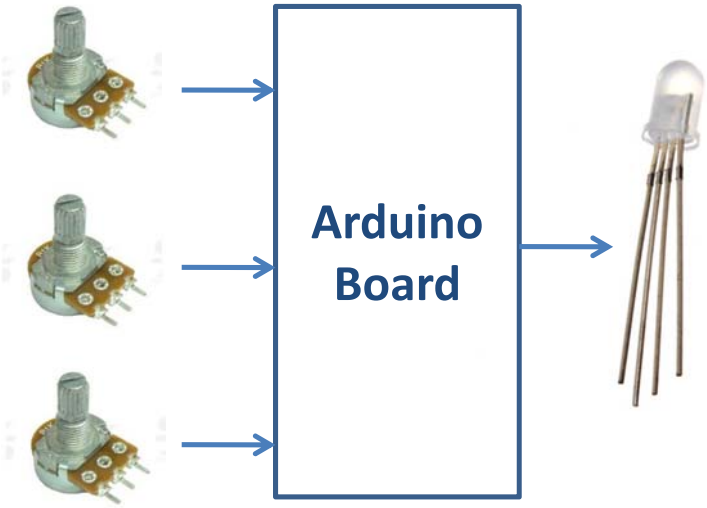
แบบฝึกหัดที่ 3.2 Tuning Color LED

จงเขียนโปรแกรมปรับสี color LED โดยรับค่าจาก ตัวต้านทานปรับค่าได้ จำนวน 3 ตัว เพื่อไปปรับค่าสี Red, Green, Blue เพื่อให้ปรับสีต่างๆ ได้ตามใจชอบ

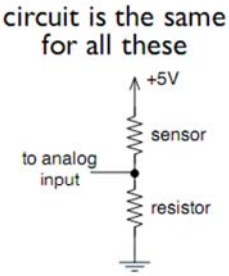


ตัวอย่าง สินค้า

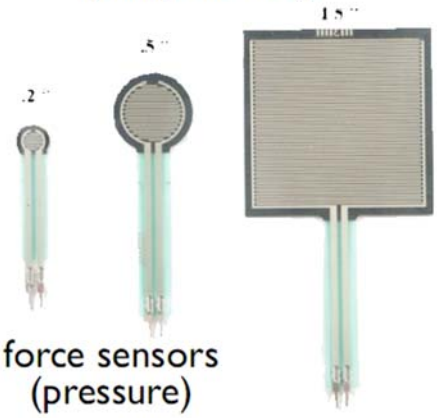
แนวทางการพัฒนา

- นำไปใช้อ่านค่าจาก Resistive Sensor ต่างๆ ได้ เช่น

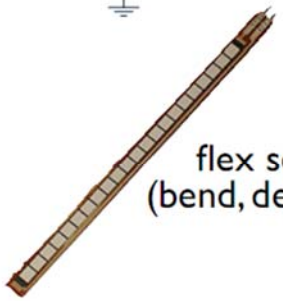
thermistor
(temperature)



photocell
(light)



force sensors
(pressure)



flex sensor
(bend, deflection)

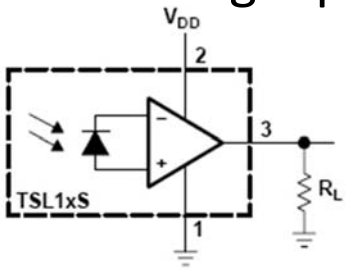
also air pressure
and others

แบบฝึกหัดที่ 3.3 Light Sensor

ต่อวงจรเซ็นเซอร์ ตรวจจับแสงดังรูป

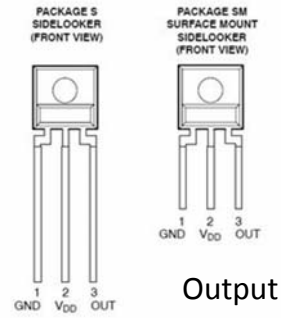
โดยให้ R_L = 10k , V_{DD} = 5V

และไปเข้าขา Analog input



ต่อเข้าขา
Analog input

รายละเอียดขา Sensor



จงเขียนโปรแกรม

Sensing the Dark

โดยที่ ให้ LED สว่าง ตามระดับความมืด

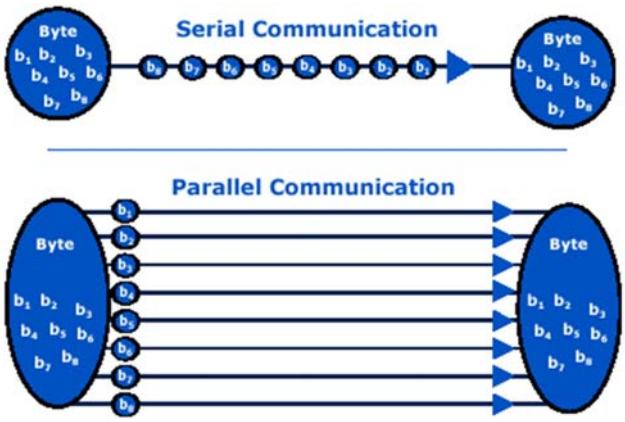
เช่น

ถ้าห้องสว่างมาก ให้ LED ดับ

ถ้าห้องมืดทึบ ให้ LED สว่างที่สุด

Output voltage is linear with light intensity

4. Serial Communication



4. Communicating with other

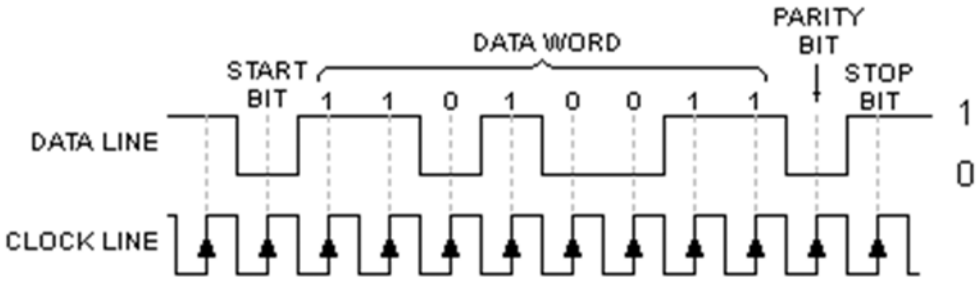
- Not just for computer-to-Arduino communications
- Many other devices speak serial
- Older keyboards & mice speak are serial (good for sensors!)
- Interface boards (graphic LCDs, servo drivers, RFID readers, Ethernet, Wi-Fi)

to Wi-Fi to Ethernet

to graphic LCD

4.1 Serial Communication

- การสื่อสารแบบอนุกรม หรือ Serial เป็นส่งข้อมูล โดยใช้เทคนิคการเลื่อนข้อมูล (Shift Bit) ส่งไปที่ละบิต บนสายสัญญาณเส้นเดียว โดยการส่งข้อมูลแบบ Serial จะไม่มีการ sync สัญญาณนาฬิการะหว่างตัวรับและตัวส่ง แต่จะอาศัยวิธี**ตั้งค่าความเร็วในการรับส่งสัญญาณให้เท่ากัน** หรือ เรียกว่าตั้งค่า **baud rate** และส่งสัญญาณ start และ stop เพื่อบอกว่า เป็นส่วนต้นของข้อมูล (start bit) หรือ ส่วนท้ายของข้อมูล (stop bit) ดังรูป



รูปแบบของ ข้อมูลจากที่ส่งผ่าน Serial จะมีการเพิ่ม Start bit และ Stop bit เข้าไปเพิ่มจากข้อมูลเดิม

- บิตเริ่มต้น (Start bit) จะมีขนาด 1 บิต จะเป็นลอจิก LOW
- บิตข้อมูล (Data bit) 8 บิต ข้อมูลที่จะส่ง
- บิตภาวะคู่หรือคี่ (Parity bit) มีขนาด 1 บิต ใช้ตรวจสอบข้อมูล ถ้าข้อมูลที่ได้รับไม่สมบูรณ์ นำเข้าค่ามา check กับ Parity bit จะได้ค่าไม่ตรงกัน
- บิตหยุด (Stop bit) เป็นการระบุถึงขอบเขตของการสิ้นสุดข้อมูล จะเป็นลอจิก HIGH

"Serial" because data is broken down into bits, each sent one-by-one on a single wire:

```
'H'
   =  0 1 0 0 1 0 0 0
   =  L H L L H L L L
   =
```

4.2 Arduino Communications

- Psst, Arduino doesn't really do USB
- It really is "serial", like old RS-232 serial
- All microcontrollers can do serial
- Not many can do USB
- Serial is easy, USB is hard

4.3 Serial command

- Talking to other uses the “Serial” command

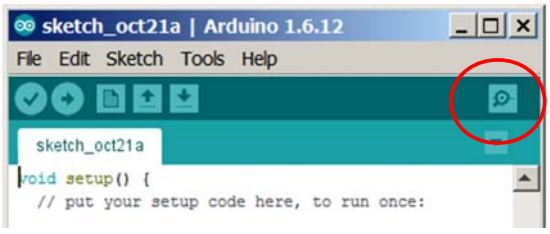
  - **`Serial.begin()`** : **prepare to use serial**
  - **`Serial.print()`** : **send data to serial port**
  - **`Serial.println()`** : **send data and newline to serial port**
  - **`Serial.read()`** : **read data from serial port**
  - **`Serial.available()`** : **ready to read**
  - **`Serial.flush()`** : **clear buffer at incoming serial data**
  - **`Serial.parseInt()`** : **returns the first valid (long) integer number from the serial buffer**

- Can talk to not just computers.
- Most things more complex than simple sensors/actuators speak serial.

การทดลองที่ 4.1 Serial Hello world !

ทดลองส่งค่าไปยังคอมพิวเตอร์

- Send “Hello World” to computer and Blink LED
- Click on “Serial Monitor” button to see output



- Watch LED at DigitalPin13

```
int ledPin =  13;
int i = 0;

void setup() {
   pinMode(ledPin, OUTPUT);
   Serial.begin(19200);
}
void loop(){
   Serial.print(i++);
   Serial.println(“Hello world”);
   digitalWrite(ledPin,HIGH);
   delay(500);
   digitalWrite(ledPin,LOW);
   delay(500);
}
```

Note!! : Serial.print()

- Prints data to the serial port as **human-readable ASCII text**. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is.

  For example:

  - Serial.print(78) gives "78"
  - Serial.print(1.23456) gives "1.23"
  - Serial.print(byte(78)) gives "N" (whose ASCII value is 78)
  - Serial.print('N') gives "N"
  - Serial.print("Hello world.") gives "Hello world."

  An optional second parameter specifies the base (format) to use

  - Serial.print(78, BIN) gives "1001110" BIN : เลขฐานสอง
  - Serial.print(78, OCT) gives "116" OCT : เลขฐานแปด
  - Serial.print(78, DEC) gives "78" DEC : เลขฐานสิบ
  - Serial.print(78, HEX) gives "4E" HEX : เลขฐานสิบหก

การทดลองที่ 4.2 Serial Read Basic

ทดลองรับค่าจากคอมพิวเตอร์

และส่งค่าออกกลับไปคอมพิวเตอร์

ในโปรแกรม “Serial Monitor”, ให้พิมพ์อะไร ก็ได้ จากนั้น กดปุ่ม Send


```
char inByte = 0;
void setup() {
    Serial.begin(19200);
   Serial.println("Hello,type something");
}

void loop() {
     if (Serial.available()) {
         inByte = Serial.read();
         Serial.print ("> ");
         Serial.println (inByte);
     }
}
```

Note !!!

- **`Serial.available()`** tells you if data present to read.
- Always check **`Serial.available()`** or if **`Serial.read() != -1`** to determine if there’s actual data to read.

การทดลองที่ 4.3 Controlling from computer

ทดลองรับค่าจากคอมพิวเตอร์

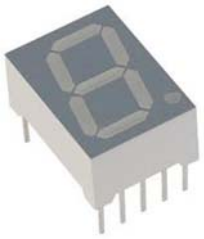
- In “Serial Monitor”, You type “H”, Press Send

- When you type “H”, LED Blink.

```
int ledPin =  13;
void setup() {
   pinMode(ledPin, OUTPUT);
   Serial.begin(19200);
}
void loop(){
if( Serial.available() ){
   int val = Serial.read();
   if (val == 'H') {
        digitalWrite(ledPin,HIGH);
        delay(500);
        digitalWrite(ledPin,LOW);
        delay(500);
        }
   }
}
```


แบบฝึกหัดที่ 4.1

- ส่งค่าจากคอมพิวเตอร์เป็นตัวเลข 0-9 แล้วแสดงผลออก 7-segment



แบบฝึกหัดที่ 4.2

- ส่งค่าจากคอมพิวเตอร์เป็นตัวเลข 0-255 ไปควบคุมความสว่างของ LED ที่ต่อที่ Digital pin 9. (ใช้คำสั่ง analogWrite ในการควบคุมความสว่าง)

แบบฝึกหัดที่ 5.3

จากการทดลอง Analog input (การทดลองที่ 3.2)

จงเขียนโปรแกรม อ่านค่าแรงดัน จากการปรับตัวต้านทานปรับค่าได้ แล้วไปแสดงผลที่จอคอมพิวเตอร์

แบบฝึกหัดที่ 5.4

จากการทดลอง Light Sensor (การทดลองที่ 3.2)

จงเขียนโปรแกรมอ่านค่าแรงดันที่ได้จาก Sensor และแสดงค่าออกจอคอมพิวเตอร์ว่า เมื่อมืดที่สุด (เอามือปิด Sensor) มีค่าเท่าใด และเมื่อสว่างสุด มีค่าเท่าใด

5. Liquid Crystal Display

5.1 Liquid Crystal Display : LCD

รายละเอียดขาต่าง ๆ

1. GND 2. VDD 3. Vb ทำหน้าที่ปรับความเข้มของจอ
4. RS (Register select) 5. R/W
6. E (Enable)
7-14 Data DB7-DB0

5.2 Function ที่ใช้ในการควบคุมจอ LCD

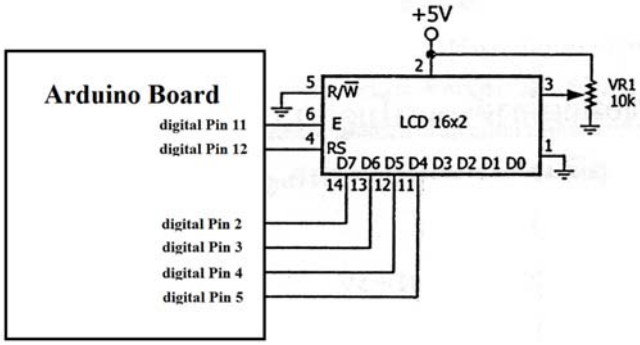
Arduino มี Library มาตรฐานสำหรับการเชื่อมต่อจอ LCD ที่ใช้ chip ของ Hitachi HD44780 (หรือ chip อื่นๆ ที่ compatible)
โดยมี function หลักๆ ที่อยู่ใน Library **`LiquidCrystal.h`** ดังนี้

- **LiquidCrystal()** ใช้ในการกำหนดขา **LCD** ที่ต่อ กับขา **ATMega**
  Syntax
  - LiquidCrystal(rs, enable, d4, d5, d6, d7)
  - LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
- **begin()** ใช้ในการกำหนดคอลัมน์ และ แถว ของ **LCD**
- **clear()** ใช้ในการ **clear** หน้าจอ **LCD** ทั้งหมด และ **cursor** มาอยู่ตำแหน่งเริ่มต้น ตรงแถวบนสุดซ้ายมือ

- **home()** ใช้ในการย้ายตำแหน่ง **cursor** มายังแถวบนสุดซ้ายมือ
- **setCursor()** ใช้ในการระบุตำแหน่งของ **cursor**
- **cursor() , noCursor()** ใช้กำหนดการแสดง **cursor**
- **blink(), noBlink()** ใช้กำหนดการกระพริบของ **cursor**
- **print()** ใช้ในการเขียน ข้อความ ลง **LCD**
- **write()** ใช้ในการเขียน ตัวอักษร ลง **LCD**
- **display() , noDisplay()** ใช้ในการควบคุมการปิด-เปิด หน้าจอ
- **scrollDisplayLeft()** ใช้ในการเลื่อนข้อความไปทางซ้าย
- **scrollDisplayRight()** ใช้ในการเลื่อนข้อความไปทางขวา
- **leftToRight()** ใช้ในการกำหนดทิศทางการเขียนข้อความ จากซ้ายไปขวา
- **rightToLeft()** ใช้ในการกำหนดทิศทางการเขียนข้อความ จากขวาไปซ้าย
- **createChar()** ใช้ในการสร้างตัวอักษรตัวใหม่

การทดลองที่ 5.1 Hello world LCD

- ต่อ LCD เข้ากับบอร์ดทดลองดังนี้

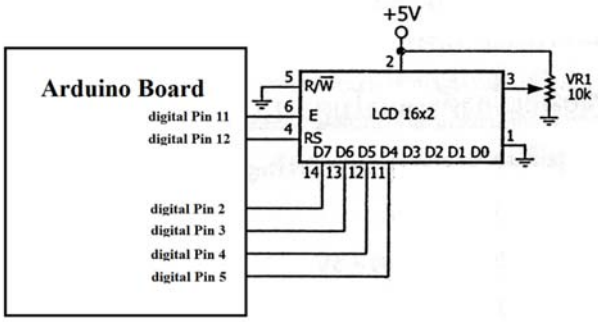



```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
      lcd.begin(16, 2);
      lcd.print("hello, world!");
   }
void loop() {
      lcd.setCursor(0, 1);
      lcd.print(millis()/1000);
   }
```

Note!!!
`lcd.setCursor(0, 1);`
set the cursor to column 0, line 1
line 1 is the second row, since counting begins with 0

การทดลองที่ 5.2 Text Direction

- ต่อ LCD เข้ากับบอร์ดทดลองดังนี้

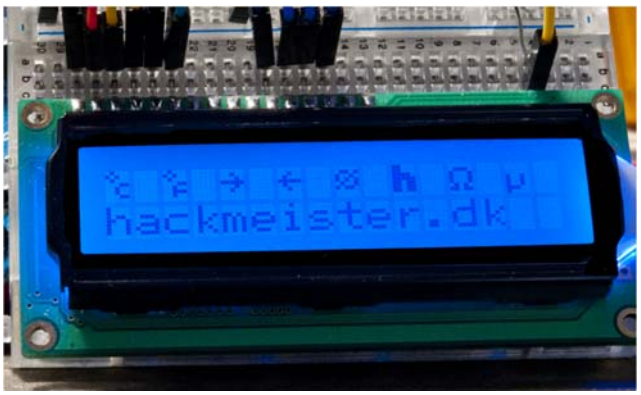


This program prints
a through l right to left,
then **m though r left to right**,
then **s through z right to left again**.

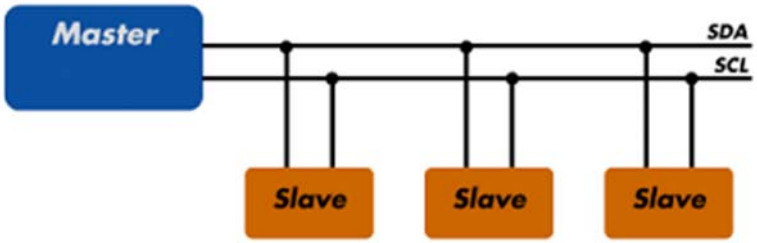
```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int thisChar = 'a';

void setup() {
      lcd.begin(16, 2);
      lcd.cursor();
    }
void loop() {

      if (thisChar == 'm') {
          lcd.rightToLeft();
      }
      if (thisChar == 's') {
          lcd.leftToRight();
      }
      if (thisChar > 'z') {
          lcd.home();
          thisChar = 'a';
      }
      lcd.write(thisChar);
      delay(1000);
      thisChar++;
    }
```


Create a custom character

- Create a custom character for use on the LCD. Up to eight characters of 5x8 pixels are supported (numbered 0 to 7).
- To display a custom character on the screen, write() its number.



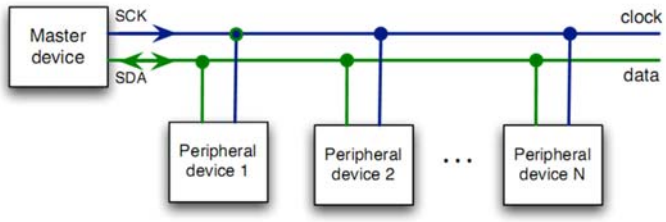
6. Inter Integrate Circuit Bus (I^2C)



I^2C , "Two Wire"

Synchronous serial bus with shared a data line

a little network for your gadgets



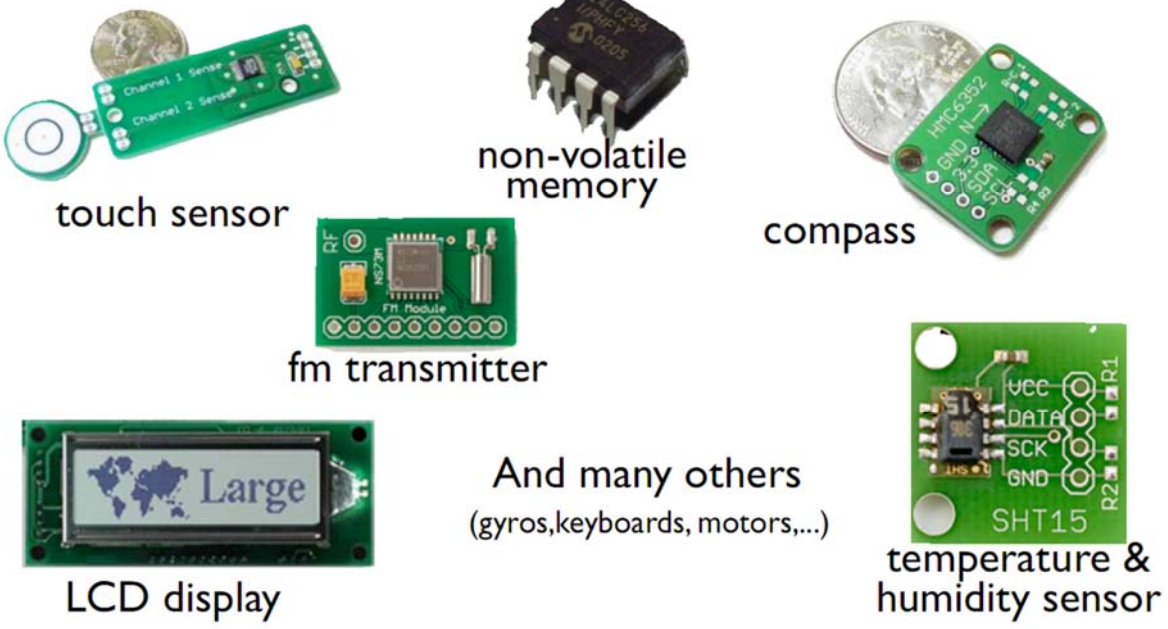
- *Up to 127 devices on one bus*
- *Up to 1Mbps data rate*
- *Really simple protocol (compared to USB,Ethernet,etc)*
- *Most microcontrollers have it built-in*

I²C , “Two Wire”

- I²C Bus ย่อมาจาก Inter Integrate Circuit Bus เป็นการสื่อสารอนุกรม แบบซิงโครนัส (Synchronous) เพื่อใช้ ติดต่อสื่อสาร ระหว่างไมโครคอนโทรลเลอร์ (MCU) กับอุปกรณ์ภายนอก

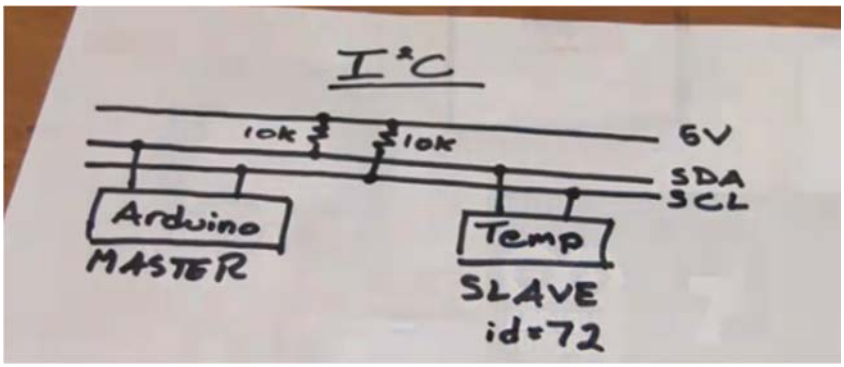
- ถูกพัฒนาขึ้นโดยบริษัท Philips Semiconductors โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้น คือ serial data (SDA) และสาย serial clock (SCL) ซึ่งสามารถ เชื่อมต่ออุปกรณ์ จำนวนหลายๆ ตัวเข้าด้วยกันได้ ทำให้ MCU ใช้พอร์ตเพียง 2 พอร์ตเท่านั้น

Many I²C devices



I²C on Arduino

- I²C built-in on Arduino's ATmega168 chip
- Use "Wire" library to access it
- **Analog In 4 is SDA signal**
- **Analog In 5 is SCK signal**



Wire library Functions

`Wire.begin()`	*join i²c bus (master)*
`Wire.begin(4)`	*join i²c bus with address #4 (Slave)*
`Wire.beginTransmission(112)`	*transmit to device #112*
`Wire.endTransmission()`	*stop transmitting*
`Wire.requestFrom(112, 2)`	*request 2 bytes from slave device #112*
`Wire.available()`	*Returns the number of bytes available for reading*
`Wire.read()`	*receive a byte*
`Wire.write("x is ")`	*sends five byte*
`Wire.write(x)`	*sends one byte*

Arduino “Wire Library”

Master Writing data to Slave

Load wire Library

Join i2c

Start sending

Send data

Stop sending

```
#include <Wire.h>
void setup()
{
  Wire.begin();                 // join i2c bus (master)
}

byte x = 0;

void loop()
{
  Wire.beginTransmission(4);    // transmit to device #4
  Wire.write("x is ");          // sends five bytes
  Wire.write(x);                // sends one byte
  Wire.endTransmission();       // stop transmitting

  x++;
  delay(500);
}
```


Arduino “Wire Library”

Master Reading data from Slave

Join i2c

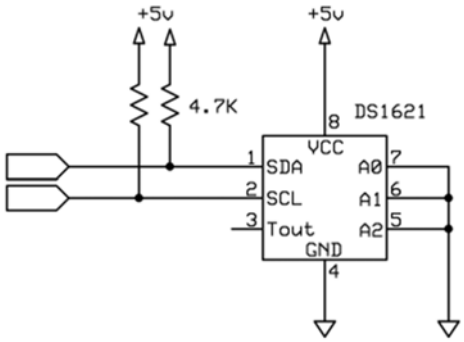
Request data

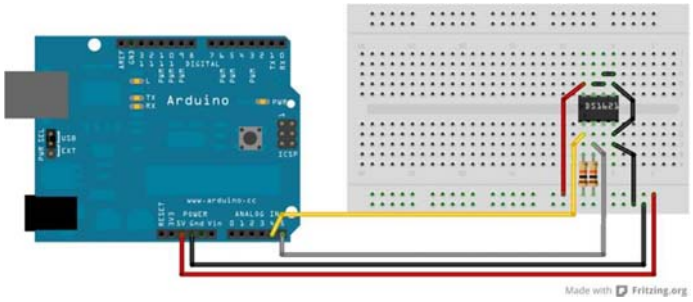
Get data

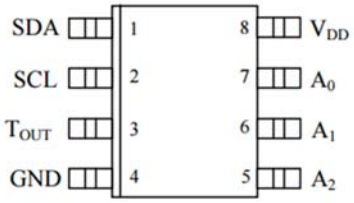
```
#include <Wire.h>

void setup()
{
 Wire.begin();                  // join i2c bus (master)
 Serial.begin(9600);            // start serial for output
}

void loop()
{
 Wire.requestFrom(2, 6);        //request 6 bytes from device #2
 while(Wire.available())        //slave may send less than requested
  {
    char c = Wire.read();       // receive a byte as character
    Serial.print(c);            // print the character
  }

 delay(500);
}
```

การทดลอง Temperature Sensor DS1621

- ต่อวงจรตามรูป







รายละเอียดขา

Address คือ $1001A_2A_1A_0$

หรือ 48H-4FH

รายละเอียด IC เบอร์ DS1621

ผู้ผลิตคือ Dallas Semiconductor เป็นไอซีวัดอุณหภูมิที่ใช้การติดต่อผ่านระบบบัส I^2C สำหรับคุณสมบัติที่สำคัญของ DS1621 มีดังนี้

-สามารถวัดอณุหภูมิ ได้ตั้งแ ต่ -55 เซลเซียส ถึง +125 เซลเซียส โดยมีความละเอียดในการวัด 0.5 เซลเซียส

- ความละเอียดของข้อมูลอุณหภูมิดิจิตอล 9 บิต

- ใช้ไฟเลี้ยงได้ตั้งแ ต่ 2.7-5.5 โวลท์

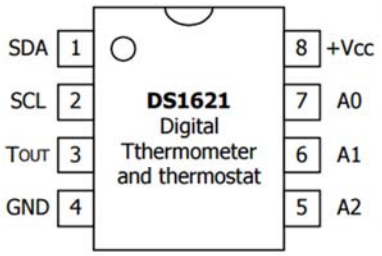
- ใช้เวลาในการวัดอุณหภูมิแล้วแปลงเป็นข้อมูลดิจิตอล 1 วินาที

- สามารถทำงานเป็นเทอร์โมสตัต (thermostat) ได้ พร้อมขาเอาต์พุต 1 ขา

- สามารถตั้งค่าอุณหภูมิที่ต้องการเมื่อทำงานเป็นเทอร์โมสตัตได้ โดยติดต่อผ่านระบบบัส I^2C และค่าที่กำหนดนี้จะคงอยู่ตลอดไป แม้ปลดไฟเลี้ยงแล้วก็ตาม สามารถเปลี่ยนแปลงได้โดยการ กำหนดทางซอฟต์แวร์เท่านั้น

-สามารถต่อพ่วงกันได้สูงสุด 8 ตัว

รายละเอียดขา และ การแปลงข้อมูล



ขาที่	ชื่อขา	หน้าที่/การทำงาน
1	SDA	ขาข้อมูลอนุกรมสำหรับเชื่อมต่อกับระบบบัส I^2C
2	SCL	ขาสัญญาณนาฬิกาสำหรับเชื่อมต่อกับระบบบัส I^2C
3	TOUT	ขาเอาต์พุตเทอร์โมสตัต กระแสซอร์สเอาต์พุต 1mA กระแสซิงก์เอาต์พุต 4mA "แอกทีฟ" เมื่ออุณหภูมิสูงถึงจุดกระตุ้นสูง (TH) "รีเซต" เมื่ออุณหภูมิลดลงต่ำกว่าจุดกระตุ้นต่ำ (TL)
4	GND	ขาต่อกราวด์ของวงจร
5-7	A2-A0	ขากำหนดแอดเดรสของ DS1621 หากมีตัวเดียวในระบบควรต่อลงกราวด์
8	+Vcc	ขาต่อไฟเลี้ยง +5V

อุณหภูมิ	ข้อมูลดิจิตอลเอาต์พุต (เลขฐานสอง)	ข้อมูลดิจิตอลเอาต์พุต (เลขฐานสิบหก)
+125°C	01111101 00000000	7B00H
+25°C	00011001 00000000	1900H
+0.5°C	00000001 00000000	0080H
0°C	00000000 00000000	0000H
-0.5°C	11111111 10000000	FF80H
-25°C	11100111 00000000	E700H
-55°C	11001001 00000000	C900H

รีจิสเตอร์ภายในไอซี DS1621

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
DONE	THF	TLF	NVB	1	0	POL	1SHOT

DONE (Conversion done – บิต 7) : บิตแสดงสถานะของการแปลงข้อมูลอุณหภูมิ

"0" – ยังอยู่ในระหว่างการแปลงข้อมูล

"1" – การแปลงข้อมูลเสร็จสมบูรณ์แล้ว

THF (Temperature High Flag – บิต 6) : บิตแฟลกแจ้งว่าอุณหภูมิที่วัดได้สูงกว่าจุดทริกอุณหภูมิสูงหรือ TH โดยเมื่อเกิดเหตุการณ์นี้ขึ้น บิตนี้จะเซตเป็น "1" และดำรงสถานะนี้ไว้จนกว่าจะมีการเขียนข้อมูล "0" มายังบิตนี้ด้วยกระบวนการทางซอฟต์แวร์ หรือเคลียร์ด้วยการปลดไฟเลี้ยง

TLF (Temperature Low Flag – บิต 5) : บิตแฟลกแจ้งว่าอุณหภูมิที่วัดได้เท่ากับหรือต่ำกว่าจุดทริกอุณหภูมิต่ำหรือ TL โดยเมื่อเกิดเหตุการณ์นี้ขึ้น บิตนี้จะเซตเป็น "1" และดำรงสถานะนี้ไว้จนกว่าจะมีการเขียนข้อมูล "0" มายังบิตนี้ด้วยกระบวนการทางซอฟต์แวร์ หรือเคลียร์ด้วยการปลดไฟเลี้ยงออกจาก DS1621

NVB (Nonvolatile Memory Busy Flag – บิต 4) : บิตแฟลกแสดงสถานะการเขียนข้อมูลลงในหน่วยความจำอีอีพรอมภายใน DS1621 เพื่อเก็บค่าพารามิเตอร์ที่จำเป็น ปกติจะใช้เวลาประมาณ 10 มิลลิวินาที

"0" – ยังอยู่ระหว่างการเขียนข้อมูล

"1" – การเขียนข้อมูลเสร็จสมบูรณ์

POL (Output Polarity Bit – บิต 1) : บิตเลือกสถานะเอาต์พุตของขา TOUT เมื่อทำงานในโหมดเทอร์โมสตัต เมื่อเลือกแล้วข้อมูลของบิตนี้จะดำรงอยู่ไปตลอดแม้ปลดไฟเลี้ยงก็ตาม หรือเรียกว่า นอนโวลาไทล์ (non-volatile)

"0" – แอกตีฟด้วยลอจิก "0"

"1" – แอกตีฟด้วยลอจิก "1"

1SHOT (One Shot Mode – บิต 0) : บิตเลือกวิธีการวัดและแปลงค่าอุณหภูมิของ DS1621 เมื่อเลือกแล้วข้อมูลของบิตนี้จะดำรงอยู่ไปตลอดแม้ปลดไฟเลี้ยงก็ตาม หรือเรียกว่า นอนโวลาไทล์ (non-volatile)

"0" – กำหนดให้ DS1621 ทำการวัดและแปลงค่าอย่างต่อเนื่อง

"1" – กำหนดให้ DS1621 เริ่มทำการแปลงค่าอุณหภูมิเมื่อได้รับสัญญาณเริ่มต้น

คำสั่งการทำงาน

คำสั่ง	ข้อมูลคำสั่ง	การทำงานหลังส่งคำสั่ง	หมายเหตุ
คำสั่งเกี่ยวกับการแปลงค่าอุณหภูมิ			
อ่านค่าอุณหภูมิ	AAH	อ่านข้อมูล 2 ไบต์	
อ่านค่าตัวนับ	A8H	อ่านข้อมูล 1 ไบต์	
อ่านค่าตัวนับสโลป	A9H	อ่านข้อมูล 1 ไบต์	
เริ่มต้นแปลงค่าอุณหภูมิ	EEH	หยุดและเตรียมพร้อมเริ่มต้น	1
หยุดแปลงค่าอุณหภูมิ	22H	หยุดและเตรียมพร้อมเริ่มต้น	1
คำสั่งในโหมดเทอร์โมสตัต			
เข้าถึง TH	A1H	เขียนข้อมูล 1 ไบต์	2
เข้าถึง TL	A2H	เขียนข้อมูล 1 ไบต์	2
เข้าถึงรีจิสเตอร์ Config.	ACH	เขียนข้อมูล 1 ไบต์	2

โปรแกรมอ่านค่าจาก DS1621

```
#include <Wire.h>


void setup()
{
    Serial.begin(9600);

    Wire.begin();
    Wire.beginTransmission(0x48);              // connect to DS1621 (#0)
    Wire.write(0xAC);                           // Access Config
    Wire.write(0x02);                        // set for continuous conversion
    Wire.beginTransmission(0x48);              // restart
    Wire.write(0xEE);                           // start conversions
    Wire.endTransmission();
}
```



```
void loop()
{
    int8_t firstByte;
    int8_t secondByte;
    float temp = 0;

    delay(1000);                          // give time for measurement

    Wire.beginTransmission(0x48);
    Wire.write(0xAA);                      // read temperature command
    Wire.endTransmission();
    Wire.requestFrom(0x48, 2);        // request two bytes from DS1621 (0.5 deg. resolution)

    firstByte = Wire.read();         // get first byte
    secondByte = Wire.read();         // get second byte

    temp = firstByte;

    if (secondByte)                       // if there is a 0.5 deg difference
        temp += 0.5;

    Serial.println(temp);

}
```