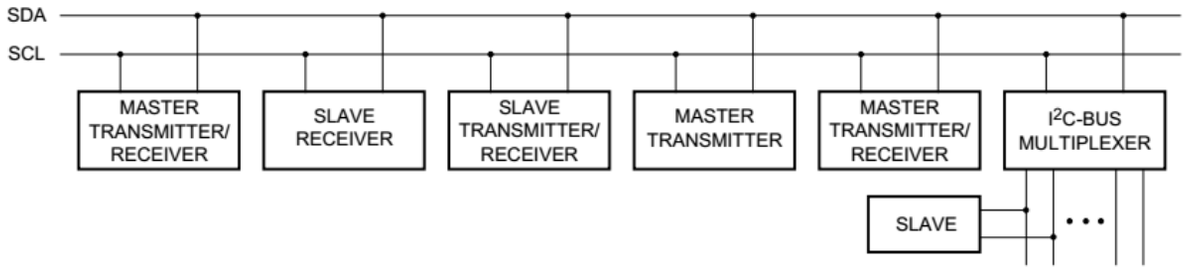
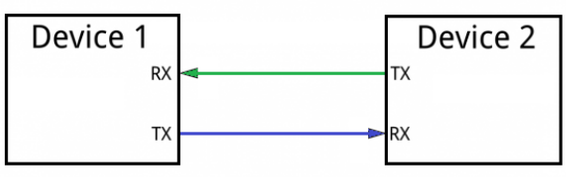
I^2C (Inter-Intergrated Circuit) เป็นโปรโตคอลมาตรฐานที่ใช้สำหรับการติดต่อสื่อสารระหว่างไมโครโปรเซสเซอร์กับอุปกรณ์ภายนอก ด้วยสัญญาณอนุกรมแบบซิงโครนัส (Synchronous) โดยใช้สายสัญญาณเพียง 2 เส้นเท่านั้นคือ SDA (Serial Line Data) และ SCL (Serial Clock) บางทีอาจจะเรียกว่า TWI (Two Wire Interface) สามารถเชื่อมต่ออุปกรณ์หรือไอซีจำนวนหลายๆตัวเข้าด้วยกันได้ แต่จะสามารถส่งข้อมูลระหว่างกันได้ครั้งละคู่เท่านั้น โดยมีตัว Master ควบคุมออกคำสั่งว่าต้องการติดต่อสื่อสารกับอุปกรณ์อะไรได้ครั้งละตัว และมีตัว Slave จะถูกสั่งให้รับข้อมูลหรือส่งข้อมูลกลับ สามารถมี Slave และ Master รวมกันได้ถึง 128 ถูกพัฒนาขึ้นโดยบริษัท Phillips Semiconductor ตั้งแต่ปี คศ.1982 ซึ่งในปัจจุบันคือบริษัท NXP Semiconductor โปรโตคอลแบบ I2C มีโหมดที่ใช้ความเร็วในการรับส่งข้อมูลตั้งแต่ 100 Kbps ไปจนถึง 5 Mbit/s

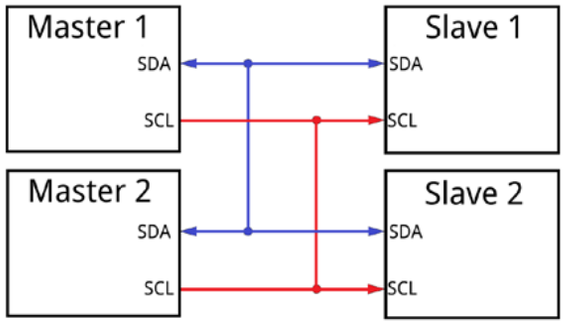


System configuration

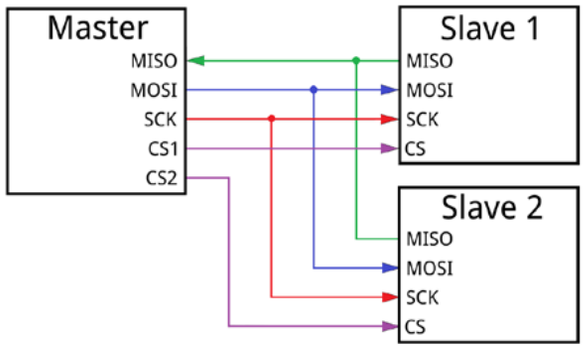
ส่วนอีกโปรโตคอลซึ่งเป็นที่นิยมใช้เช่นกันคือ SPI (Serial Peripheral Interface) ความแตกต่างระหว่าง I2C และ SPI คือจำนวนสายที่ใช้ในการติดต่อ I2C ใช้ 2 เส้น ส่วน SPI ใช้สายสัญญาณ 4 เส้นและจะเพิ่มจำนวนของสายสัญญาณตามจำนวนอุปกรณ์ที่มากขึ้น การสื่อสารของ I2C จะใช้สายที่น้อยกว่าเพราะเป็นแบบ Half Duplex และความเร็วที่ใช้ในการส่งของ I2C จะช้ากว่าแบบ SPI ที่เป็นการสื่อสารแบบ Full Duplex ซึ่งมีสายสัญญาณข้อมูลแยกกันเป็นการรับ (MISO) และส่ง (MOSI) อย่างละ 1 เส้น แต่ก็เพียงพอสำหรับการงานทั่วไป ตัวอย่าง Interface ที่ใช้ในการสื่อสารแต่ละแบบแสดงดังรูป



UART Interface diagram

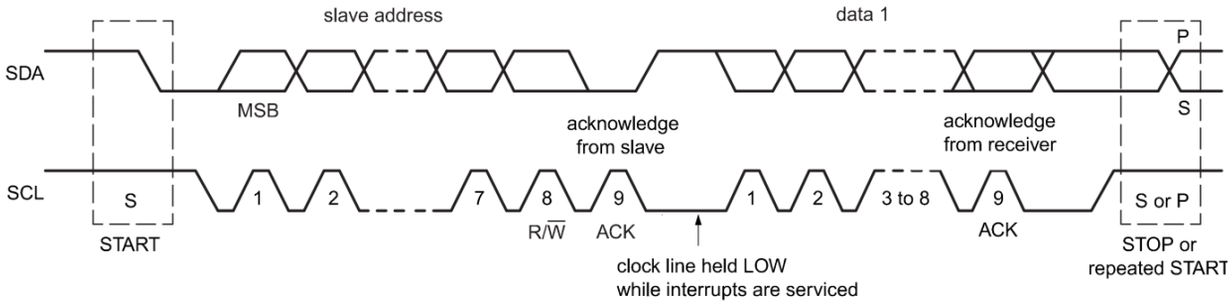


I2C interface diagram



SPI interface diagram

ขั้นตอนการส่งข้อมูลด้วยโปรโตคอล I2C ต้องต่อสายตามที่แสดงในรูปคือ VCC , GND , SDA และ SCL โดยขาสัญญาณ SDA และ SCL จะเป็นขาสัญญาณเอาต์พุทที่มีลักษณะเป็นแบบ Open-drain หรือ Open-collector ซึ่งจะต้องใช้ไฟเลี้ยงจากแหล่งจ่ายไฟภายนอกไอซี จึงต้องต่อกับตัวต้านทานแบบ Pull up ไว้ เพื่อจะให้เอาต์พุตสามารถเชื่อมต่อกันได้หลายตัวพร้อมกัน การทำงานตัว Master จะเป็นตัวเรียกไปที่ Slave โดยส่ง Address ไปที่สายสัญญาณข้อมูล SDA ซึ่ง Master จะเป็นตัวควบคุม Slave ให้ทำงานเข้าจังหวะกับ Clock ซึ่งจะถูกส่งโดยใช้สาย SCL ตัว Slave ที่มี Address ตรงกับที่ถูกเรียกจะส่งสัญญาณ Acknowledge ตอบกลับไปยังสายสัญญาณ SDA ด้วยวิธีการ Pull down สายให้แรงดันเป็น 0 V ลำดับขั้นตอนการทำงาน (Timing Diagram) เป็นดังนี้



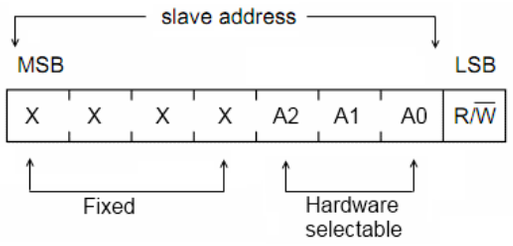
Data transfer on the I^2C-bus

1. ในสภาวะปกติที่ยังไม่มีการใช้งานจะเป็นสภาวะบัสว่าง โดยสายสัญญาณ SDA และ SCL จะมีแรงดันเป็น High ทั้งคู่

2. เมื่อมาสเตอร์ (Master) ซึ่งอุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I2C ต้องการรับ-ส่งข้อมูล จะส่งสภาวะเริ่มต้น (START) เพื่อแสดงการขอใช้บัส โดย Master จะสั่งให้มีการเปลี่ยนสัญญาณของสาย SDA จาก High ไปเป็น Low ในขณะที่ SCL จะยังคงเป็น High อยู่

3. ต่อจากนั้นตัว Master จะส่งให้สัญญาณของสาย SCL ลงไปเป็น Low และ Master จะเริ่มส่งข้อมูลค่า Address บิทแรกไปที่สาย SDA ตามด้วยสัญญาณ Clock ไปที่สาย SCL และสเลฟ(Slave) ซึ่งเป็นอุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I2C จะเริ่มอ่านค่าในจังหวะที่ SCL เป็น High โดยอุปกรณ์ที่ต่ออยู่บนบัสอาจจะมีอุปกรณ์ต่อพร้อมกันหลายตัวก็ได้ โดยค่า Address นี้จะเป็นรหัสควบคุม (Control Byte) ที่ประกอบไปด้วย 3 ส่วน คือ

- รหัสประจำตัวของอุปกรณ์ (Device ID) มีขนาด 4 บิท ถูกกำหนดโดยผู้ผลิตไอซีจะเปลี่ยนแปลงหรือแก้ไขไม่ได้
- Device Address มีขนาด 3 บิท สามารถกำหนดเองได้จากการต่อขาสัญญาณลอจิกให้กับไอซี
- Mode เป็นบิทควบคุมการอ่านหรือเขียนข้อมูลกับอุปกรณ์ภายนอก



สภาวะของช่วงเวลาการรับ-ส่งข้อมูลใน I2C bus จะกระทำในขณะที่ขา SCL เป็น High ในระหว่างการถ่ายทอดข้อมูลสายข้อมูล (SDA) ต้องรักษาข้อมูลไว้ ไม่ให้เกิดการเปลี่ยนแปลงเด็ดขาด และสภาวะการเปลี่ยนแปลงข้อมูลจะกระทำในขณะที่ขา SCL เป็น Low เท่านั้น

4. ตัว Master จะส่งค่า Address ของอุปกรณ์ที่ต้องการจะติดต่อด้วยไปเรื่อยๆรวมทั้งหมด 7 บิท แล้วตามด้วยบิทที่ 8 ซึ่งจะเป็นคำสั่ง Mode คือเป็นการระบุว่าสั่งให้ Slave ทำอะไร ถ้าให้เขียน (Write) หรือ Output mode คือ ตัว Master ต้องการส่งข้อมูลไปที่ Slave จะให้สาย SDA เป็น Low และถ้าให้อ่าน (Read) หรือ Input mode คือให้ตัว Master รับค่าที่ส่งมาจาก Slave จะต้องให้สาย SDA เป็น High

5. ส่วนในบิทที่ 9 เป็นการติดต่อด้วยโดยจะต้องส่งสถานะรับรู้ Acknowledge (ACK) เป็นบิทที่ใช้บอกว่า Slave มีการตอบสนองต่อคำสั่งที่ได้รับมาแล้วหรือไม่ คือจะเป็นการตอบกลับจาก Slave ที่มี Address ตรงกับที่ Master ส่งไป ซึ่งถ้ามี Slave ที่ Address ตรงกับที่ระบุไว้ ตัว Slave จะตอบรับ (Acknowledge) โดยการดึงสัญญาณสาย SDA ลงเป็น Low คือพร้อมจะสื่อสารด้วย แต่ถ้าไม่มี Slave ที่ Address ตรงกับที่ Master ต้องการสื่อสารด้วย สายสัญญาณ SDA จะยังค้างอยู่ที่ High คือไม่มีการตอบรับ (Not Acknowledge)

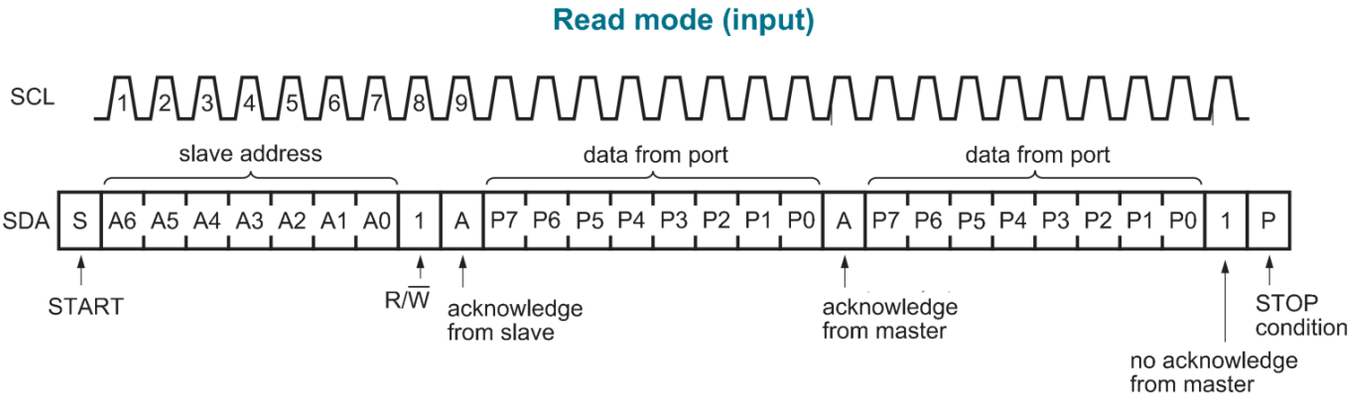
6. เมื่อ Slave มีสัญญาณตอบกลับมาเรียบร้อย คือ Acknowledge แล้ว จะเป็นช่วงเวลาที่สาย SCL ถูกดึงสัญญาณลง Low เป็นเวลาสั้นๆ และสาย SDA จะถูกปล่อยว่าง ก่อนที่ Slave ที่ติดต่อด้วยจะเริ่มส่งค่าข้อมูลบิทแรกมาที่สายสัญญาณ SDA

7. จากนั้นตัว Master จะส่งสัญญาณจากสาย SCL ออกไป เพื่ออ่านค่าข้อมูลเข้ามาจากสาย SDA ในช่วงจังหวะที่สาย SCL เป็น High และจะรับค่าข้อมูล (Data) ไปเรื่อยๆ จนครบข้อมูลชุดแรก 8 บิท ตามจังหวะของสายสัญญาณ SCL โดยในช่วงที่เกิดการอ่านข้อมูลค่าสภาวะของลอจิกที่เกิดขึ้นบนสาย SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสภาวะเป็น High ไม่เช่นนั้นแล้วข้อมูลที่ทำการถ่ายทอดจะเกิดความผิดเพี้ยนได้ และข้อมูลจะมีการมเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็น Low เท่านั้น

8. เมื่อรับข้อมูลครบ 8 บิทแล้ว ในกรณีข้อมูลที่ Slave ต้องการส่งยังมีข้อมูลชุดอื่นอีก Master จะให้สัญญาณ Acknowledge โดยดึงสัญญาณสาย SDA เป็น Low เพื่อบอกให้ Slave รู้ว่า Master จะรอข้อมูลอีก 8 บิทชุดต่อไป และจะเริ่มกลับไปทำงานต่ออีกเหมือนในข้อ 6

9. แต่ถ้ารับข้อมูลจนครบ 8 บิท และ Slave ไม่มีข้อมูลชุดอื่นที่จะส่งมาอีกแล้ว เมื่อถึงจังหวะการ Acknowledge ในบิทที่ 9 สายสัญญาณ SDA จะถูกปล่อยให้เป็น High

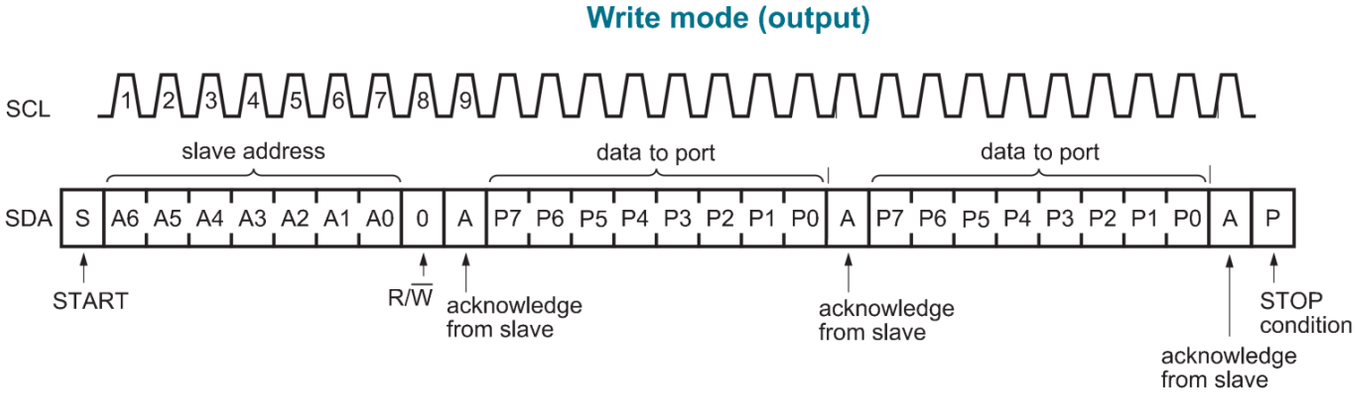
10. เมื่อรับข้อมูลจนครบหมดแล้ว Master ก็จะต้องสั่งให้หยุด (Stop) เป็นสถานะที่บอกให้อุปกรณ์รู้ว่าสิ้นสุดการใช้บัสในการรับส่งข้อมูลแล้ว โดยการส่งสัญญาณให้สาย SDA เปลี่ยนจาก Low เป็น High ในขณะที่สัญญาณสาย SCL เป็น High อยู่ ซึ่งจะได้ Timing Diagram ของการอ่านข้อมูลเข้ามาของ I2C แสดงดังรูป



ตัวอย่าง รหัสข้อมูลที่ใช้ในโหมดการอ่าน โดยข้อมูลที่ถูกส่งจาก Slave จะเป็นอักษรตัวหนา

<S> <slave address + read> ***<ACK>*** ***<data in>*** <ACK> ***<data in>*** <ACK> ***<data in>*** <NACK> <P>

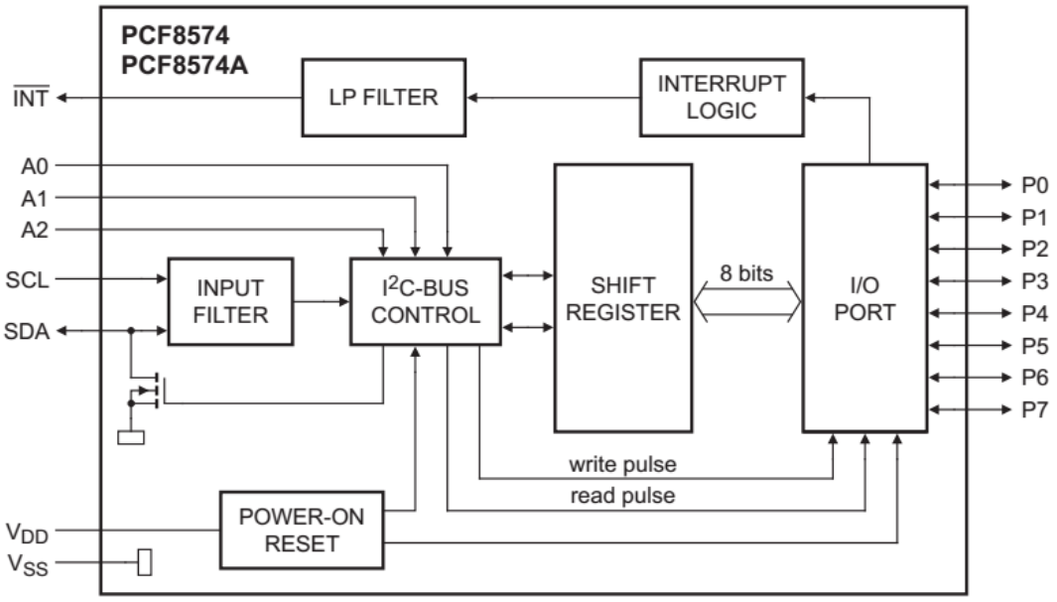
ส่วน Timing Diagram ของ I2C ในการเขียนข้อมูล ซึ่งจะส่งออกจาก Master ไปที่ Slave แสดงได้ดังรูป



ตัวอย่าง รหัสข้อมูลที่ใช้ในโหมดการเขียน โดยข้อมูลที่ถูกส่งจาก Slave จะเป็นอักษรตัวหนา

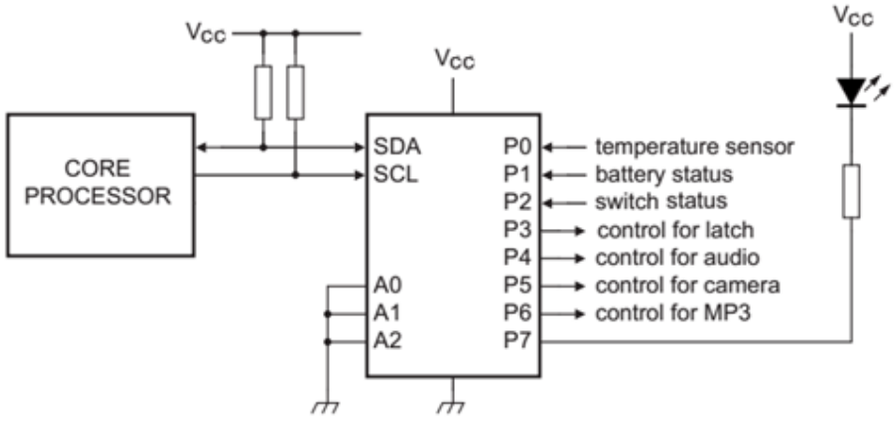
<S> <slave address + write> ***<ACK>*** <data out> ***<ACK>*** <data out> ***<ACK>*** <data out> ***<ACK>*** <P>

การทดลองนี้จะใช้งานโมดูล I2C ของ LCD โดยนำเอามาประยุกต์ใช้งานในการติดต่อกับอุปกรณ์ต่างๆ เพื่อเรียนรู้การทำงานของ I2C จากสายสัญญาณ SDA และ SCL ของโมดูลรับข้อมูลเข้ามาแบบอนุกรม ภายในบอร์ดจะใช้ Chip PCF8574 ซึ่งทำหน้าที่รับสัญญาณจาก I2C มาผ่าน Shift Register เพื่อแปลงข้อมูลที่เข้าออกจากอนุกรมให้เป็นข้อมูลแบบขนานออกที่พอร์ต โดยมีจำนวนขาสัญญาณที่สามารถใช้เป็นอินพุทหรือเอาท์พุท 8 ช่อง คือขา P0 ถึง P7 ขนาดเท่ากับ 1 byte โดยมี Block diagram ของไอซีดังรูป รายละเอียดเพิ่มเติมให้เปิดดูได้จากเอกสาของไอซี PCF8574 DataSheet และ I2C-bus specification and user manual



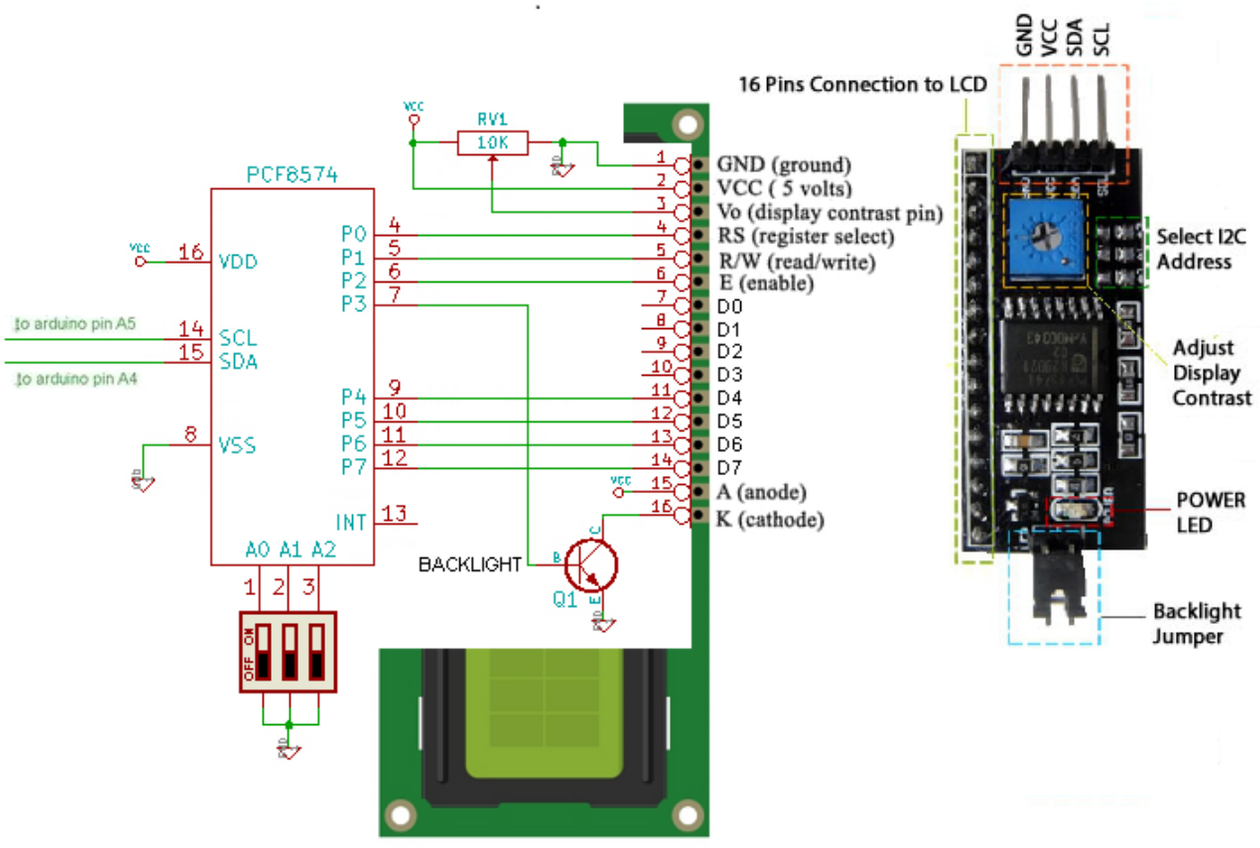
PCF8574 diagram

เราสามารถนำไอซี PCF8574 ไปประยุกต์ใช้งานต่างๆได้หลายแบบดังรูป



Bidirectional I/O expander application

ตัวโมดูล LCD มีขาอยู่ทั้งหมด 16 ขา ใช้งานข้อมูลเป็นโหมด 4 บิท อยู่ที่ขา P4 ถึง P7 กับสายสัญญาณ RS, RW, E และการควบคุม Back Light จำนวน 4 บิท อยู่ที่ขา P0 ถึง P3 โมดูลนี้จะมีจุดต่อดังรูปด้านล่าง จะเห็นมีตำแหน่งเขียนว่า A0 A1 A2 จะใช้ในกรณีที่ต้องการควบคุมอุปกรณ์นี้หลายตัวพร้อมกัน โดยจะต้องเปลี่ยนค่า Address ของอุปกรณ์แต่ละตัวไม่ให้ซ้ำกัน สามารถทำได้โดยการบัดกรีเพื่อเชื่อมต่อขั้วให้เข้ากัน ทำให้สามารถเลือก Address 0010 0XXX ใน 3 บิทสุดท้าย A0 A1 และ A2 ได้เป็นค่าตั้งแต่ 0 ถึง 7 ถ้าไม่บัดกรีทั้ง 3 บิทสุดท้ายจะมีค่าเป็น 111 การทดลองจะใช้ I2C Board Module For LCD มาทำหน้าที่รับส่งข้อมูลต่างๆ โดยใช้ขา P0 ถึง P2 ของไอซี PCF8574 มาทำหน้าที่เป็น Input เมื่อต่ออยู่ในบอร์ดจะเป็นขา 4 ถึงขา 5 และใช้ขา P4 ถึง P7 ของไอซี PCF8574 มาทำหน้าที่เป็น Output เมื่อต่ออยู่ในบอร์ดจะเป็นขา 11 ถึงขา 14



การเขียนโปรแกรมเพื่อควบคุมสัญญาณแต่ละเส้นจากบอร์ด Arduino จะใช้คำสั่งที่ได้จาก Library ที่ชื่อว่า Wire เพื่อนำการสื่อสารแบบ I2C มาใช้ โดยเลือก Address ของอุปกรณ์ให้ตรงกับฮาร์ดแวร์ แล้วใช้คำสั่งใน Wire.h จัดการรับส่งค่าให้ เริ่มจากการหา Address ที่ใช้สำหรับโมดูลนั้น ถ้าไม่รู้ค่า Address เราสามารถเขียนโปรแกรมเพื่อทดสอบหา Address ได้ดังนี้

```
// Write data to I2C slave device
#include <Wire.h>

void setup()
{
    Wire.begin();                                   // Start I2C bus
    Serial.begin(115200);                           // Setup serial for debug
}

void loop()
{
   byte address,data,device;

    for(address = 1; address < 127; address++ )     // sets the value (range from 1 to 127)
    {
        Wire.beginTransmission(address);            // transmit to address
        if (Wire.endTransmission() == 0)            // I2C devices found
        {
            device = address;
            Serial.print("\n I2C Device Address: "); // Print Device Address
            Serial.println(address, HEX);           // print as an ASCII-encoded hexa);
        }
    }
}
```

1. การทดลองให้ต่อ I2C Board Module For LCD เข้ากับบอร์ด Arduino ผ่านการเชื่อมต่อทางพอร์ต I2C

2. ให้ทดลองป้อนโปรแกรมเพื่อตรวจหาอุปกรณ์ต่างๆ ที่ต่ออยู่กับพอร์ต I2C และบันทึกค่าที่ได้

I²C Device Adress : 21

3. ให้ต่อวงจรโดยใช้ LED ที่เป็นแม่สี 3 สีและตัวต้านทาน 220 โอห์ม เข้าทีขา P4 P5 P6 ของ I2C Board
4. ให้แก้ไขโปรแกรมข้อ 2 โดยเพิ่มโปรแกรมการส่งค่าไปที่พอร์ต I2C โดยค่าที่ส่งไปมีค่าตั้งแต่ 0 ถึง 255 และพิมพ์ผลค่าที่ส่งออกไปทาง Serial Monitor ดังนี้

```
for (data = 0 ; data <= 255; data++)                // sets the value (range from 0 to 255)
{
        Wire.beginTransmission(device);             // transmit to device
        Wire.write(data);                           // sends one byte
        Wire.endTransmission();                     // stop transmitting

        Serial.print("pin state  :   Out =  ");     // Print pin state
        Serial.print(data, BIN);                    // print as an ASCII-encoded binary);
        delay(100);                                 // wait for 100 milliseconds
}
```

5. ให้อธิบายผลลัพธ์ที่ได้ของ LED จากการส่งข้อมูลผ่านทางพอร์ต I2C ส่งผล 0-255 = 8 bit

LED จะเปลี่ยนสีตามค่า data ที่เพิ่มขึ้น

6. ให้แก้ไขโปรแกรมข้อ 4 โดยเพิ่มโปรแกรมการรับค่าจากพอร์ต I2C หลังจากส่งข้อมูลไปในแต่ละค่า และพิมพ์ผลลัพธ์ของค่าที่รับมาไปทาง Serial Monitor ดังนี้

```
Wire.requestFrom(device, 1);                // receive 1 bytes from slave device
x = Wire.read();                            // Read pin state
Serial.print("\t,  In =  ");                 // Print pin state
Serial.println(x, BIN);                     // print as an ASCII-encoded binary);
```

7. ให้อธิบายผลลัพธ์ที่ได้จากค่าที่รับมาทาง Serial Monitor ผ่านทางพอร์ต I2C

0000 0000 → 1111 1111

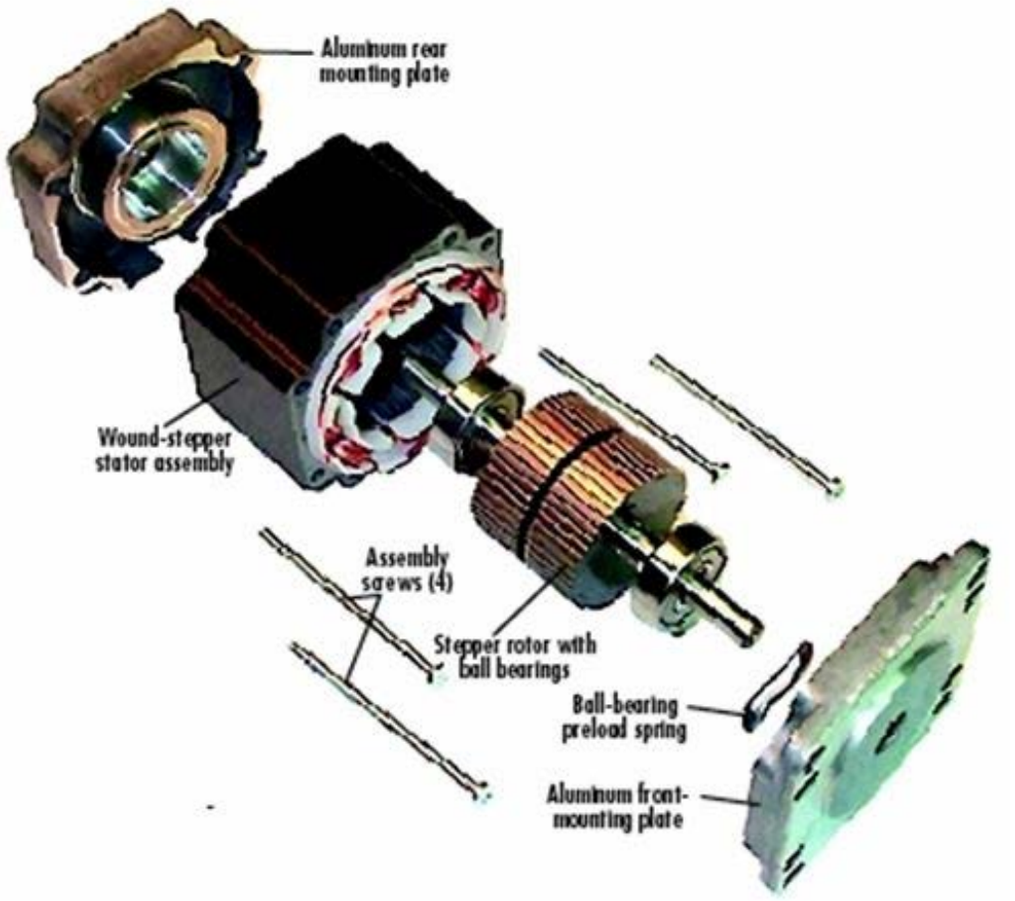
เพิ่มทีละ 1 ไปจนสุด

8. ให้ต่อตัวความต้านทาน 10 KΩ เข้ากับขา Vcc ของ I2C Board และนำขาอีกข้างต่ออนุกรมกับสวิทช์แล้วลงกราวด์ โดยให้ขา P2 ต่อเข้ากับจุดต่อร่วมระหว่างตัวความต้านทานกับสวิทช์
9. จากโปรแกรมในข้อ 2 ให้เพิ่มโปรแกรมทดสอบการกดสวิทช์ กำหนดเงื่อนไขไว้ว่าเมื่อมีการกดสวิทช์ให้ส่งค่าที่ได้กลับไปยังบอร์ด Arduino ผ่านทางพอร์ต I2C แล้วให้ LED ที่บอร์ด Arduino ขา 13 สว่าง

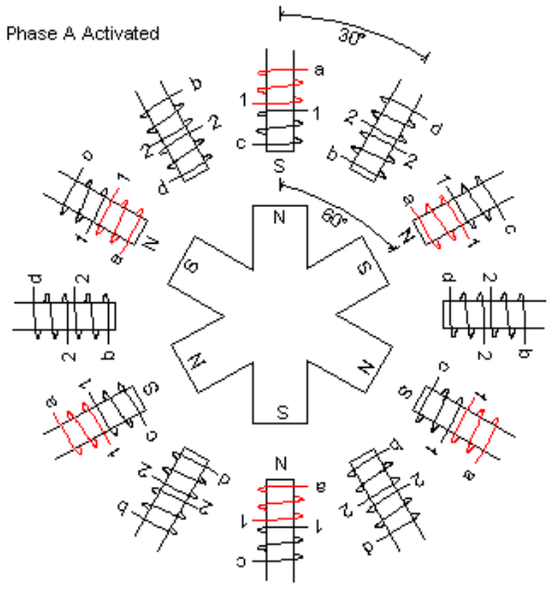
```
Wire.requestFrom(device, 1);                // recive 1 bytes from slave device
x = Wire.read();                            // Read pin state

Serial.print("\t pin state  :  In =  ");     // Print pin state
Serial.print(x, BIN);                       // print as an ASCII-encoded binary);

x = x & 0x0f;
if ((x & 4) != 4)
      digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on
else
      digitalWrite(LED_BUILTIN, LOW);       // turn the LED off
```

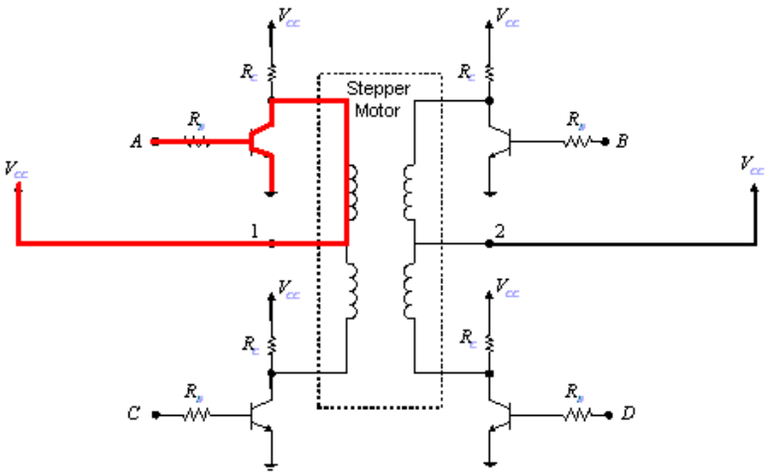
10. ให้แก้ไขโปรแกรมโดยเพิ่มโปรแกรมจากข้อ 9 กำหนดเงื่อนไขไว้ว่าเมื่อมีการกดสวิทช์ให้ส่งค่าที่ได้กลับไปยังบอร์ด Arduino ผ่านทางพอร์ต I2C แล้วสั่งงานให้ LED ที่บอร์ด I2C ขา P4 สว่าง

Stepper Motor หรือ Stepping Motor คือ มอเตอร์ที่มีการหมุนเป็นขั้นๆ (Step) เมื่อมีสัญญาณไฟฟ้าที่เป็นพัลส์มาป้อนเข้าที่สเต็ปเปอร์มอเตอร์จะทำให้แกนกลางของมอตอร์หมุนเป็นมุมคงที่ คือมีค่าเป็นองศาต่อสเต็ป ทำให้สามารถควบคุมตำแหน่งของการหมุนได้อย่างแม่นยำ โดยไม่ต้องใช้การควบคุมแบบป้อนกลับ (Feedback Control) หรืออาศัยตัวตรวจจับการหมุนมาควบคุมตำแหน่ง การควบคุมสเต็ปเปอร์มอเตอร์จะใช้ไมโครคอนโทรลเลอร์ส่งสัญญาณดิจิตอลมาควบคุมบังคับทิศทาง และความเร็วในการหมุนของแกนสเต็ปเปอร์มอเตอร์ได้โดยตรง โดยไม่ต้องอาศัยแปรงถ่านจึงไม่เกิดการสึกหรอและไม่มีสัญญาณรบกวนที่เกิดจากหน้าสัมผัสของแปรงถ่าน จึงเป็นที่นิยมใช้ในอุปกรณ์ที่ต้องการควบคุมตำแหน่งและมุมได้อย่างแม่นยำ เช่น ปริ้นเตอร์ สแกนเนอร์ ฮาร์ดดิสก์ กล้องวงจรปิด เครื่องปรับอากาศ และอุปกรณ์ของรถยนต์ เป็นต้น โดยที่ Stepping Motor แต่ละตัวจะมีความละเอียดของมุมหมุนที่แตกต่างกันขึ้นอยู่กับโครงสร้างการผลิต

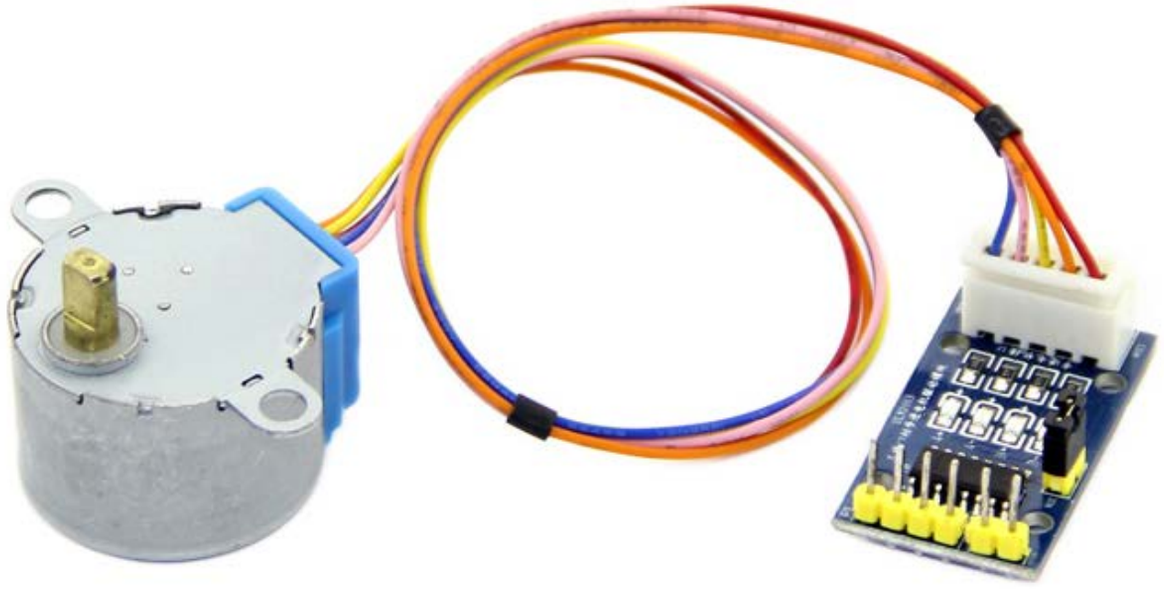


การทำงานของ Stepper Motor จะต้องป้อนแรงดันไฟฟ้าไปที่ขดลวดที่ติดตั้งบนสเตเตอร์ ให้ถูกต้องตามจังหวะเพื่อไปบังคับให้แม่เหล็กถาวร (Permanent Magnet) บนแกนโรเตอร์หมุนไปตามทิศทางที่กำหนด โดยทั่วไปสเต็ปเปอร์มอเตอร์แบ่งได้ 2 แบบ ตามลักษณะของโครงสร้างการต่อขดลวดภายในมอเตอร์ คือ Unipolar และ Bipolar ซึ่งหลักในการขับของ Stepper Motor ทั้งสองแบบทำงานจะทำงานคล้ายกัน คือการป้อนพัลส์เป็นช่วงๆเข้าไปยังขดลวดต่างๆ เพื่อให้ Stepper Motor หมุนไปตามองศาที่ต้องการ โดยปกติแบบ Bipolar จะมีสายไฟต่อ 4 เส้น การต่อวงจรอาจจะต้องใช้วงจร H-Bridge เข้ามาช่วยเพื่อกลับทิศทางของสนามแม่เหล็กภายใน ส่วนแบบ Unipolar จะมีสายไฟต่อ 5 เส้น แต่การทำวงจรควบคุมจะทำได้ง่ายกว่า เนื่องจากไม่ต้องกลับทิศของกระแสไฟที่ป้อนเข้าไปที่ขดลวด

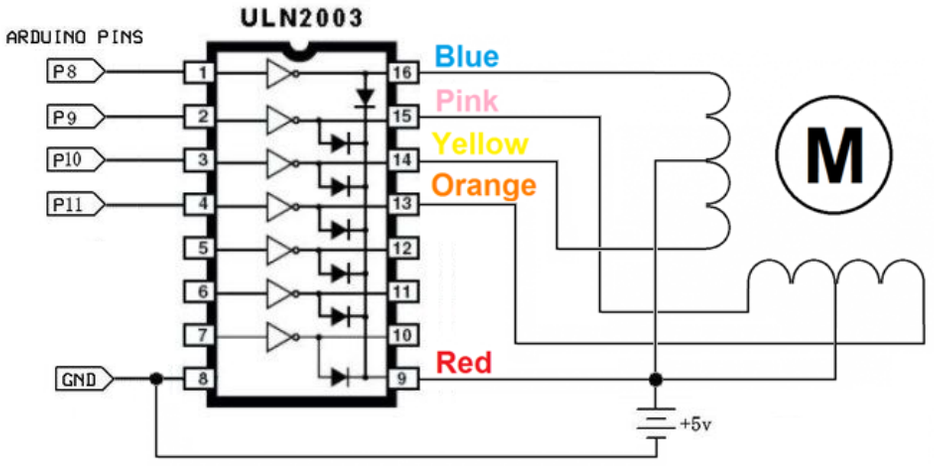




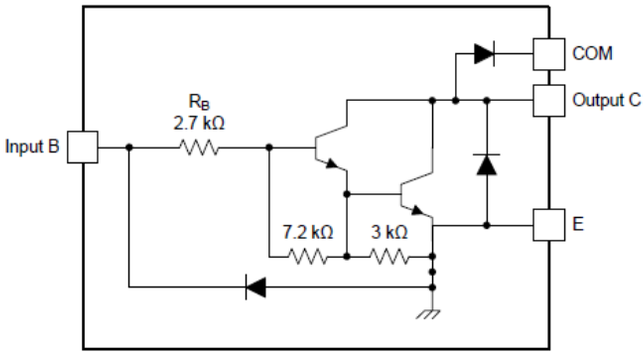
การทำงานของ Stepper Motor นั้นต้องใช้กระแสสูงเพื่อป้อนเข้าขดลวดให้ทำงาน ดังนั้น Microcontroller จะไม่สามารถเชื่อมต่อโดยตรงได้ จึงต้องมีชุดขับกระแสใช้ไอซีเบอร์ ULN2003 การทดลองจะใช้วงจรควบคุมที่เป็นโมดูล 4 Phase Stepper Motor Driver ตามในรูป



การทำงานของวงจรจะจ่ายแรงดันไฟฟ้า 5V เข้าที่จุดต่อร่วม Common Anode ไปยัง Stepping Motor และป้อนสัญญาณจาก Microcontroller เป็นลอจิกต่างๆจำนวน 4 บิท เข้าไปยัง IC 2003 ที่ทำหน้าที่เป็น Driver เพื่อขับเฟสให้กับ Stepping Motor ทั้ง 4 เส้นเพื่อให้ลงกราด์ตามจังหวะของสัญญาณที่ป้อน ซึ่งสามารถดูการทำงานทั้ง 4 เฟสได้จาก LED ที่ต่ออยู่บนบอร์ด โดยมีวงจรดังรูป



โดยที่วงจรภายในแต่ละขาจะใช้ทรานซิสเตอร์ต่อเป็นวงจรดาร์ลิงตัน (Darlington) เพื่อขับกระแสทำให้สามารถใช้กระแสได้ถึง 500 mA และมีช็อทกี้ไดโอด (Schottky Diode) ซึ่งเป็นไดโอดที่มีค่าแรงดันตกคร่อมขณะนำกระแสต่ำและทำงานได้ดีที่ความถี่สูง มาทำหน้าที่ ป้องกันแรงดันไฟย้อนกลับ (Negative Undershoot) ที่เกิดจากการทำงานของมอเตอร์ ซึ่งจะเป็นอันตรายทำให้วงจรควบคุมเสียหายได้



ULN2003A Block Diagram

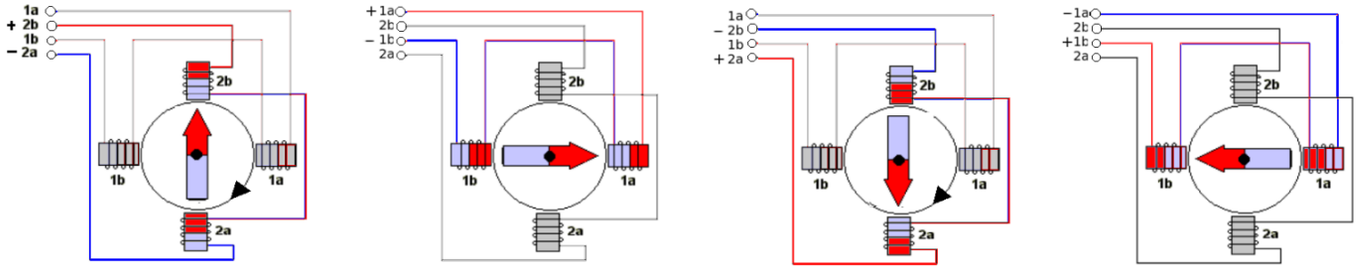
11. ให้เชื่อมต่อ Steper Motor กับวงจร 4 Phase Stepper Motor Driver Module เข้ากับบอร์ด Arduino ทางขา 8 ถึง 11 และต่อแหล่งจ่ายไฟฟ้า
12. ป้อนโปรแกรมเพื่อกำหนดการทำงานของ Stepping Motor ดังนี้

```
//declare variables for the motor pins
int motorPin1 = 8;                    // Blue
int motorPin2 = 9;                    // Pink
int motorPin3 = 10;                   // Yellow
int motorPin4 = 11;                   // Orange
                                      // Red

int motorSpeed = 100;                 //variable to set stepper speed
int stepCount = 0;                    // number of steps the motor has taken

void setup()
 {
        pinMode(motorPin1, OUTPUT);   //declare the motor pins as outputs
        pinMode(motorPin2, OUTPUT);
        pinMode(motorPin3, OUTPUT);
        pinMode(motorPin4, OUTPUT);
        Serial.begin(115200);         // initialize the serial port:
}
```

```
void loop()
{
        wavedrive();                                    // จ่ายไฟให้ทำงานครั้งละ 1 ขด
        fullstep();                                     // จ่ายไฟให้ทำงานครั้งละ 2 ขด
        halfstep();                                     // ใช้ทั้งสองแบบรวมกันทำให้หมุนได้ครั้งละครึ่งเสตป

        Serial.print("steps:" );
        Serial.println(stepCount);
        stepCount++;
}
```

การควบคุมการหมุนของสเต็ปเปอร์มอเตอร์ สามารถทำได้โดยการป้อนแรงดันไฟฟ้าคงที่เข้าไปที่ขั้วของขดลวดที่ควบคุมการหมุน เพื่อบังคับให้แม่เหล็กถาวรบนแกนโรเตอร์หมุนไปตามทิศการบังคับของขดลวดที่ติดตั้งบนสเตเตอร์ โดยจะต้องป้อนแรงดันให้ถูกต้องตามจังหวะเพื่อให้แกนโรเตอร์หมุนดังรูป

ตามตัวอย่างขดลวดแต่ละขดห่างกัน 90 องศา การหมุนแบบง่ายที่สุดทำได้โดยการจ่ายกระแสไฟเข้าไปกระตุ้นทีละขดลวดในแต่ละเฟสตามลำดับ 1a 2a 1b 2b ถ้าหากต้องการให้กระแสไหลในเฟสใดก็จะทำให้สถานะของเฟสนั้นเป็น High ซึ่งจะทำให้เกิดสนามแม่เหล็ก เพื่อไปดูดแม่เหล็กถาวรที่อยู่บนโรเตอร์ให้เคลื่อนที่ โดยมีทิศทางการหมุนตามลำดับการจ่ายกระแสไฟเข้าที่ขดลวดอยู่ 4 จังหวะต่อการหมุน 1 รอบ



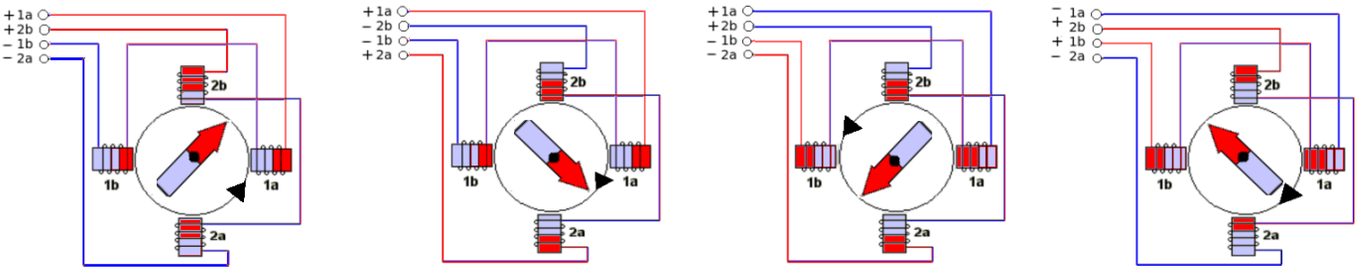
การควบคุมการหมุนแบบ Wave Drive จะเป็นการป้อนกระแสไฟให้กับขดลวดของสเต็ปเปอร์มอเตอร์ทีละขด โดยจะป้อนกระแสเรียงตามลำดับกันไป ดังนั้นกระแสที่ไหลในขดลวดจะไหลในทิศทางเดียวกันทุกขด การควบคุมแบบนี้ทำได้ง่ายแต่แรงขับของสเต็ปปิ้งมอเตอร์ที่ได้มีน้อย ความเร็วที่ได้จากการหมุนของสเต็ปเปอร์มอเตอร์จะขึ้นอยู่กับการหน่วงเวลา (Time Delay) ของการป้อนกระแสไฟฟ้าให้กับขดลวดในแต่ละครั้งตามลำดับ ถ้า Time Delay มีค่ามากมอเตอร์จะหมุนช้า และถ้า Time Delay มีค่าน้อยมอเตอร์จะหมุนเร็ว แต่ถ้าน้อยมากๆก็อาจจะไม่เสถียรได้

13. การทดลองการทำงาน จากโปรแกรมในข้อ 12 ให้เพิ่มโปรแกรมการส่งข้อมูลไปที่ Stepping Motor โดยใช้การควบคุมแบบ Wavedrive เพื่อจ่ายไฟให้ทำงานครั้งละ 1 ขด ซึ่งก็คือให้ทำงานครั้งละ 1 เฟส ทดลองการทำงานของโปรแกรมและให้บันทึกผลที่ได้

```
void wavedrive()
{
        // 1
        digitalWrite(motorPin4, HIGH);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, LOW);
        delay(motorSpeed);
        // 2
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, LOW);
        delay(motorSpeed);
```

```
        // 3
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin1, LOW);
        delay(motorSpeed);
        // 4
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, HIGH);
        delay(motorSpeed);
}
```

14. Stepping Motor ทำงานมีค่ามุมหมุนต่อ Step เท่ากับ ..... 0.175 ..... องศา
15. ถ้าต้องการให้หมุน 1 รอบจะต้องใช้ทั้งหมดเท่ากับ ..... 2060 ..... Step
16. การหมุนของแต่ละ Step ในโปรแกรมเป็นการหมุนตามเข็มหรือทวนเข็มนาฬิกา ..... ทวนเข็ม .....
17. ถ้าต้องการให้หมุนในทิศทางตรงกันข้ามกันต้องแก้ไขโปรแกรมในส่วนไหน .....
..... ทำการสลับค่าใน waveDrive() จาก 1000 → 0001 .....
..... 0100 → 0010 .....
..... 0010 → 0100 .....
..... 0001 → 1000 .....
18. ให้ทดลองแก้ไขค่าใน Delay ให้น้อยลงและมากขึ้นและอธิบายผลลัพธ์ที่ได้ .....
..... 3 < Delay ≤ 100 → 350 มอเตอร์จะไม่หมุน แต่ยังมีเสียงดังอยู่ .....
..... 0 ≤ 100 มอเตอร์หมุนช้าจนไม่มีเสียงดังหมุน .....
.....



การควบคุมการหมุนแบบ Full Step จะเป็นการป้อนกระแสไฟให้กับขดลวดของสเต็ปเปอร์มอเตอร์แบบทีละ 2 เฟสพร้อมกัน โดยต้องป้อนกระแสเรียงตามลำดับกันไปครั้งละ 2 ขด ดังนั้นจึงมีกระแสไหลในขดลวดของมอเตอร์มากขึ้น ซึ่งทำให้มอเตอร์มีแรงบิดในการหมุนมากขึ้นตามไปด้วย

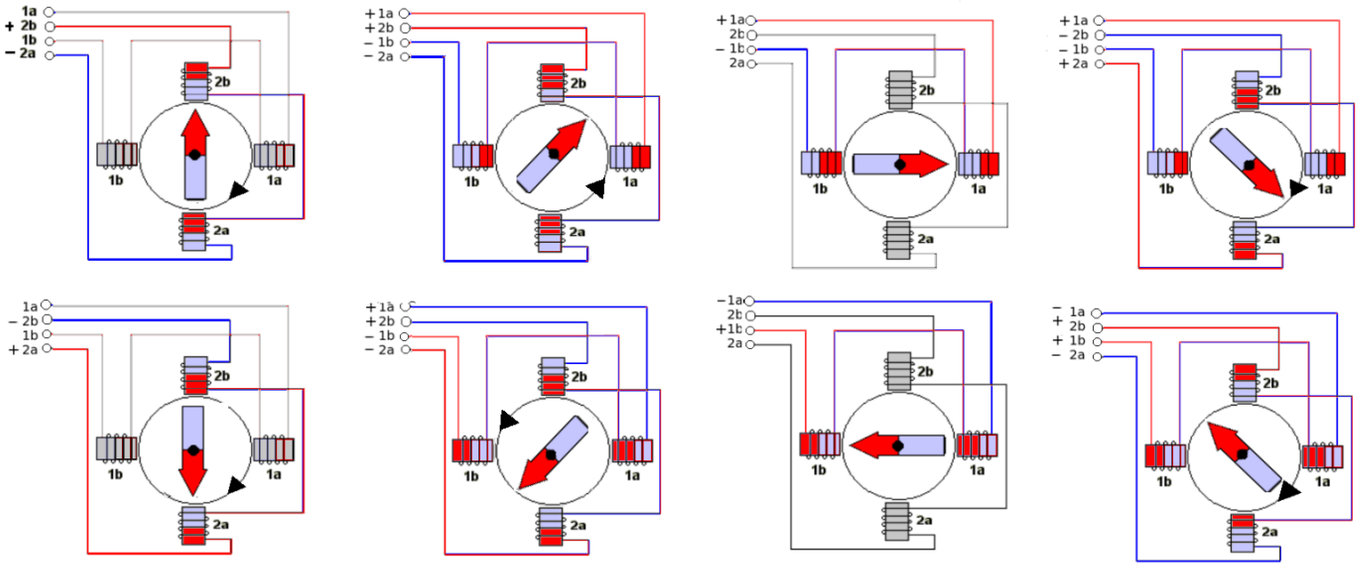
19. จากโปรแกรมในข้อ 13 ให้เพิ่มโปรแกรมการส่งข้อมูลไปที่ Stepping Motor โดยใช้การควบคุมแบบ Fullstep ให้ทดลองการทำงานของโปรแกรมและบันทึกผลที่ได้

```
void fullstep()
{
        // 1
        digitalWrite(motorPin4, HIGH);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, LOW);
        delay (motorSpeed);
```

```
        // 2
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin1, LOW);
        delay(motorSpeed);
        // 3
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin1, HIGH);
        delay (motorSpeed);
        // 4
        digitalWrite(motorPin4, HIGH);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, HIGH);
        delay(motorSpeed);
}
```

20. Stepping Motor ทำงานแบบ Full Step มีมุมหมุนต่อ Step เท่ากับแบบ Wavedrive หรือไม่ มีองศาเท่ากัน

21. การทำงานในแต่ละ Step ของแบบ Full Step กับแบบ Wavedrive อยู่ในตำแหน่งองศาเดียวกันหรือไม่ มี องศาเดียวกัน



การควบคุมการหมุนแบบ Half Step เป็นการป้อนกระแสไฟให้กับขดลวดของสเต็ปเปอร์มอเตอร์ครั้งละ 1 เฟส และ 2 เฟส สลับกันไป ทำให้สเต็ปเปอร์มอเตอร์มีความละเอียดของตำแหน่งในการหมุนเพิ่มขึ้น 2 เท่า ซึ่งจะหมุนได้ครั้งละครึ่งสเต็ป โดยที่ไม่ต้องปรับเปลี่ยนฮาร์ดแวร์เพียงแต่แก้ไขโปรแกรมวิธีการจ่ายกระแสไฟเข้าขดลวดให้เพิ่มมากขึ้น

22. จากโปรแกรมในข้อ 19 ให้เพิ่มโปรแกรมการส่งข้อมูลไปที่ Stepping Motor ด้วยวิธีการควบคุมแบบ Half Step โดยนำข้อมูลของทั้งสองแบบมารวมกัน ให้ทดลองการทำงานของโปรแกรมและบันทึกผลที่ได้

```
void halfstep()
{
        // 1
        digitalWrite(motorPin4, HIGH);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, LOW);
        delay(motorSpeed);
```

```
        // 2
        digitalWrite(motorPin4, HIGH);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, LOW);
        delay (motorSpeed);
        // 3
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, LOW);
        delay(motorSpeed);
        // 4
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin1, LOW);
        delay(motorSpeed);
        // 5
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin1, LOW);
        delay(motorSpeed);
        // 6
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin1, HIGH);
        delay (motorSpeed);
        // 7
        digitalWrite(motorPin4, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, HIGH);
        delay(motorSpeed);
        // 8
        digitalWrite(motorPin4, HIGH);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin1, HIGH);
        delay(motorSpeed);
}
```

23. Stepping Motor ทำงานแบบ Half Step มีมุมหมุนต่อStep เท่ากับ ..... 0.088 ..... องศา
24. ถ้าต้องการให้แบบ Half Step หมุน 1 รอบจะต้องใช้ทั้งหมดเท่ากับ ..... 4096 ..... Step
25. ให้เชื่อมต่อ Steper Motor กับวงจร 4 Phase Stepper Motor Driver Module เข้ากับ P4 ถึง P7 (ขา 11 ถึง 14) ของ I2C Board Module แล้วไปเชื่อมต่อเข้ากับบอร์ด Arduino ผ่านทางพอร์ต I2C
26. จากโปรแกรมในข้อ 2 ให้เพิ่มโปรแกรมเพื่อให้ควบคุม Stepping Motor แบบ Wavedrive ผ่านทาง I2C แล้วพิมพ์ผลค่าที่ส่งออกและรับเข้ามาจากพอร์ต I2C ออกไปทาง Serial Monitor ดังนี้

```
data = 0x80;
for (i = 1 ; i <= 4; i++)                    // sets the value (range from 1 to 8)
{
  Wire.beginTransmission(device);            // transmit to device
  Wire.write(data);                          // sends one byte
  Wire.endTransmission();                    // stop transmitting

  Serial.print("pin state  :  Out =  ");     // Print pin state
  Serial.print(data, BIN);                   // print as an ASCII-encoded binary);
  delay(5);                                  // wait for stepper speed
  data = data >>  1;

  Wire.requestFrom(device, 1);               // receive 1 bytes from slave device
  x = Wire.read();                           // Read pin state
  Serial.print("\t,  In =  ");               // Print pin state
```

```
  Serial.println(x, BIN);                    // print as an ASCII-encoded binary);
}
```

27. ให้ต่อตัวความต้านทาน 10 KΩ เข้ากับขา Vcc ของ I2C Board และนำขาอีกข้างต่ออนุกรมกับสวิทช์แล้วลงกราวด์ เหมือนในข้อ 8 จำนวน 2 ชุด โดยจุดต่อร่วมระหว่างตัวความต้านทานกับสวิทช์แต่ละชุดต่อเข้ากับขา P0 และ P1 ตามลำดับ
28. จากโปรแกรมในข้อ 26 ให้แก้ไขโปรแกรมเพิ่ม โดยกำหนดว่าเมื่อกดสวิทช์ที่ตำแหน่งขา P1 ให้ Stepping Motor หมุน และถ้าปล่อยสวิทช์ให้ Stepping Motor หยุดหมุน ตามตัวอย่างของโปรแกรมดังนี้

```
Wire.requestFrom(device, 1);               // recive 1 bytes from slave device
x = Wire.read();                           // Read pin state

Serial.print("\t pin state  :  In =  ");    // Print pin state
Serial.print(x, BIN);                      // print as an ASCII-encoded binary);
x = x & 0x0f;

if ((x & 2) == 2)
{
    data = 0x80 | x;
    for (i = 1 ; i <= 4; i++)              // sets the value (range from 1 to 4)
    {
            Wire.beginTransmission(device);// transmit to device
            Wire.write(data);              // sends one byte
            Wire.endTransmission();        // stop transmitting
            delay(5);                      // wait for stepper speed

            Serial.print("\n Out =  ");     // Print pin state
            Serial.print(data, BIN);       // print as an ASCII-encoded binary);

            data = data >> 1;
            data = data | x;
    }
}
```

29. จากโปรแกรมในข้อ 28 ให้แก้ไขโปรแกรมเพิ่ม โดยกำหนดว่าเมื่อมีการกดสวิทช์ที่ตำแหน่งขา P0 ให้ Stepping Motor หมุนตามเข็มนาฬิกา และถ้าปล่อยสวิทช์ให้ Stepping Motor หมุนทวนเข็มนาฬิกา
30. จากโปรแกรมในข้อ 29 ให้แก้ไขโปรแกรมที่ใช้ในการควบคุม Stepping Motor จากแบบ Wavedrive ไปเป็นแบบ Half Step