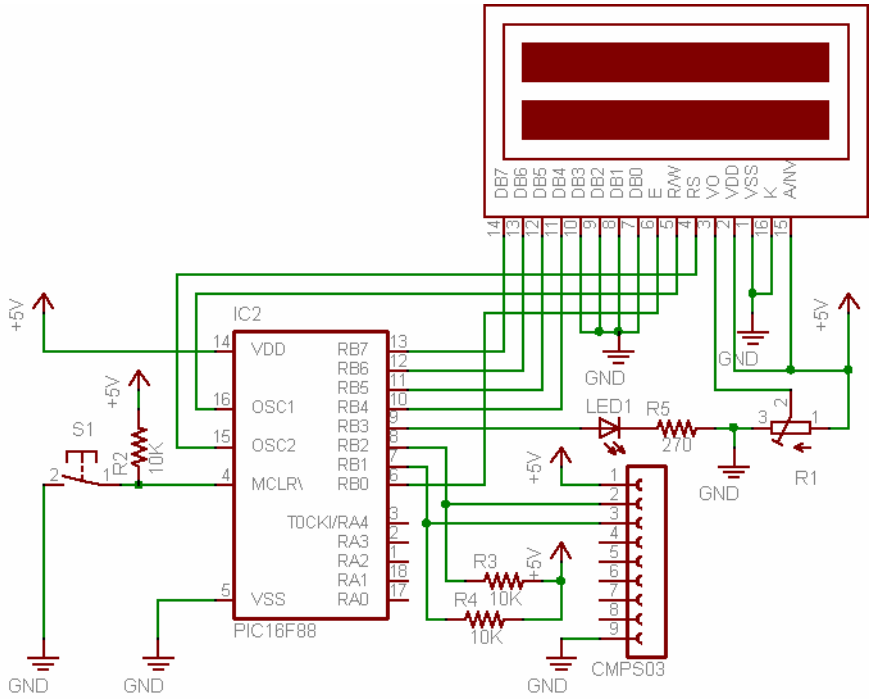
Project 10 I²C Master

โจทย์

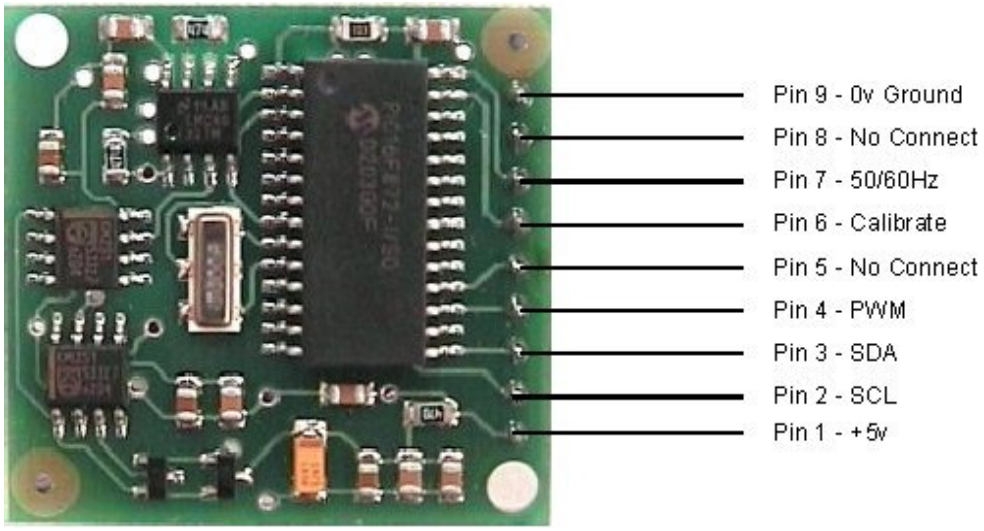
สร้างวงจรและเขียนโปรแกรมสำหรับ PIC16F88 เพื่อสื่อสารกับอุปกรณ์เข็มทิศดิจิตอลผ่านการสื่อสารแบบ I²C ทำการอ่านค่าองศาปัจจุบันแล้วนำมาแสดงผลออกทาง LCD Display

ขั้นตอนการปฏิบัติ

1) การออกแบบวงจร

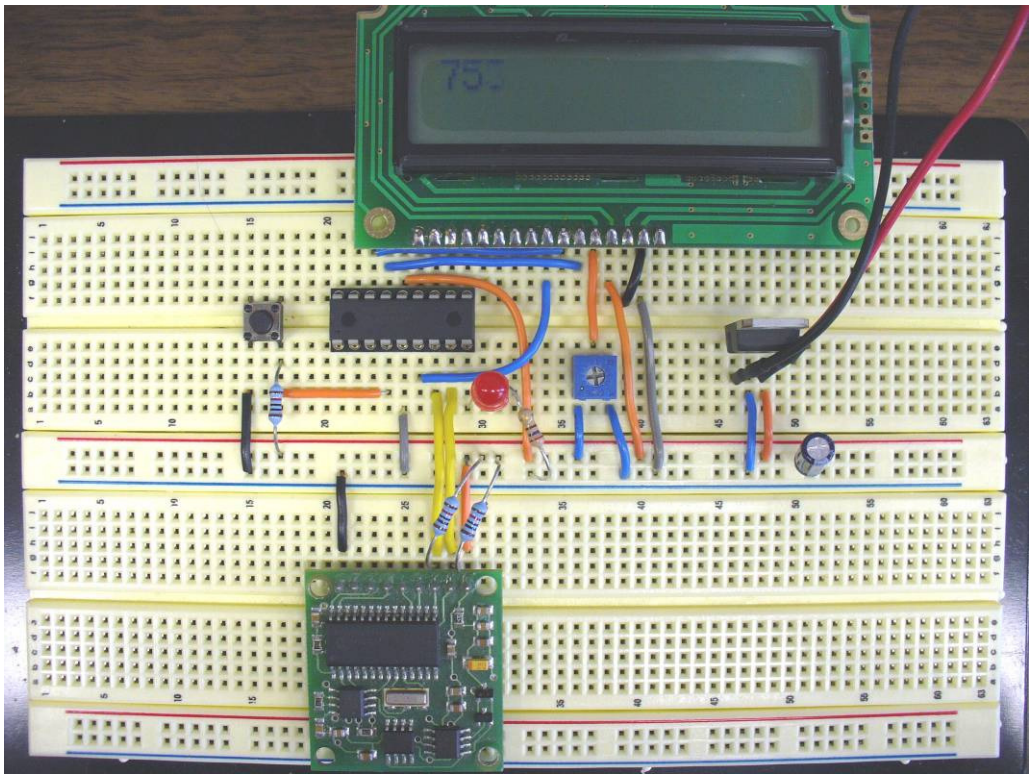


รูปที่ 1 แผนผังวงจรสำหรับการสื่อสารแบบ I²C กับ CMPS03 และการแสดงผลผ่าน LCD Display



รูปที่ 2 รูปวงจรสำเร็จรูปเข็มทิศดิจิตอล CMPS03

2) การต่อวงจร



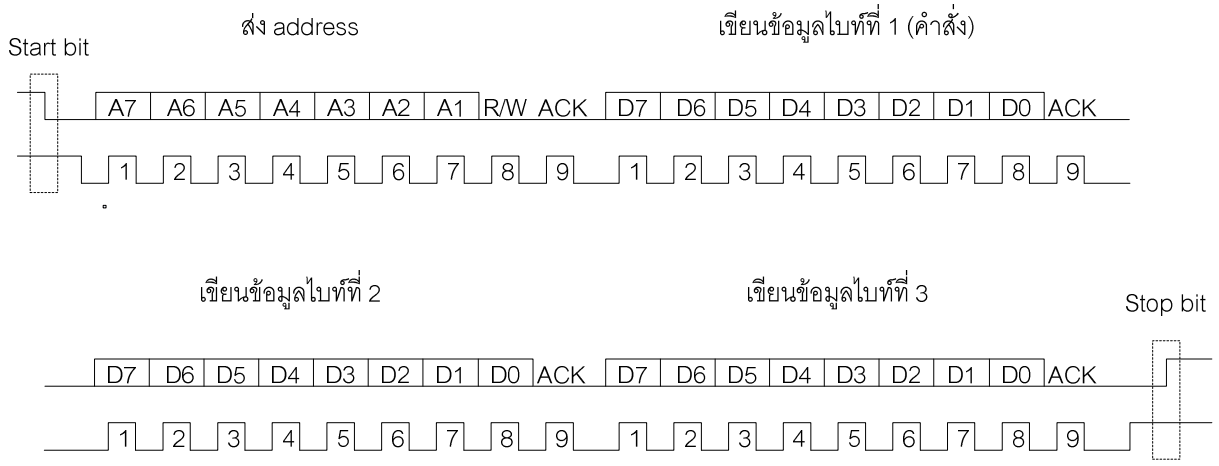
รูปที่ 3 รูปแสดงการต่อวงจรสำหรับการสื่อสารแบบ I^{2}C กับ CMPS03

3) การเขียนโปรแกรม

<u>การสื่อสารแบบ I^{2}C</u>

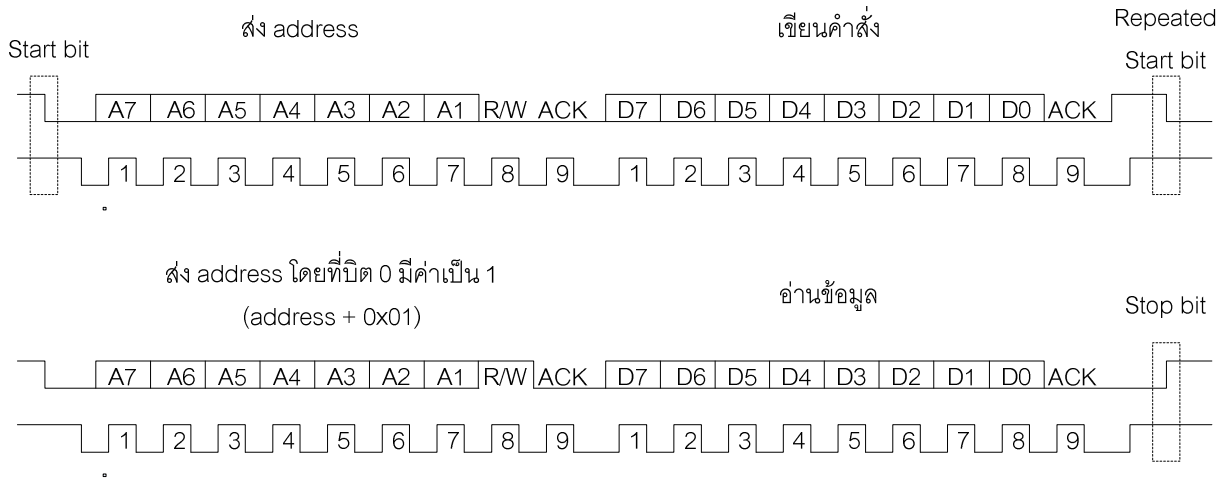
การสื่อสารแบบ I^{2}C ใช้สัญญาณสองเส้นได้แก่ สัญญาณนาฬิกา (SCL) และสัญญาณข้อมูล (SDA) โดย master จะเป็นตัวสร้างสัญญาณนาฬิกา และควบคุมการอ่านและเขียนข้อมูลทั้งหมด โดยที่ slave จะต้องปฏิบัติตาม โปรโตคอลการสื่อสาร I^{2}C ที่ใช้สำหรับ CMPS03 จะเหมือนกับที่ใช้กับอุปกรณ์ EEPROM เช่น 24C04 ซึ่งมีรูปแบบการรับส่งข้อมูลดังต่อไปนี้

ในกรณีที่ master ต้องการส่งหรือเขียนข้อมูลมาให้กับ slave ก็จะเริ่มด้วยการส่ง start bit และส่ง address ของอุปกรณ์ที่ต้องการสื่อสารด้วย Address สำหรับ I^{2}C จะใช้เพียง 7 บิต (MSB) เท่านั้น ดังนั้น address จึงต้องเป็นเลขคู่เสมอ (เช่น 0x00, 0x02 เป็นต้น) ส่วนอีกบิตหนึ่ง (LSB) จะใช้เป็นบิตที่บอกให้ slave รู้ว่า master กำลังต้องการอ่านหรือเขียนจาก slave ในกรณีที่เป็นการเขียนข้อมูลบิตนี้จะมีค่าเป็น 0 หลังจากส่ง address แล้วก็จะเริ่มการส่งข้อมูลซึ่งจะมีจำนวนหลายไบท์ก็ได้ เมื่อ master ไม่ต้องการส่งข้อมูลแล้วก็จะส่ง stop bit เพื่อปิดการสื่อสารรูปที่ 4 แสดงโปรโตคอลการสื่อสารแบบ I^{2}C ในการเขียนข้อมูลของ master



รูปที่ 4 I²C Communication Protocol (Master write)

ในกรณีที่ master ต้องการรับหรืออ่านข้อมูลจาก slave ก็จะเริ่มต้นเหมือนการส่งข้อมูลให้กับ slave ก่อนเพื่อบอก slave ว่าต้องการให้ส่งข้อมูลอะไรมาให้ โดยเริ่มจากการส่ง start bit เช่นเดิม และตามด้วยการส่ง address ของอุปกรณ์ที่ต้องการสื่อสารด้วย จากนั้น master จะส่งคำสั่ง 1 ไบท์ไปให้กับ slave ตามด้วย start bit อันที่สอง (repeated start bit) เพื่อบอกให้ slave รู้ว่าจะทำการส่ง address อีกครั้ง โดยครั้งนี้จะส่ง address ซึ่งมีบิต LSB มีค่าเป็น 1 เพื่อบอกให้ slave รู้ว่าจะเริ่มทำการอ่านข้อมูลจาก slave แล้วจึงเริ่มทำการอ่านข้อมูลจาก slave ซึ่งอาจจะมีจำนวนหลายไบท์ก็ได้ และเมื่อ master ไม่ต้องการรับข้อมูลแล้วก็จะส่ง stop bit เพื่อปิดการสื่อสาร รูปที่ 5 แสดงโปรโตคอลการสื่อสารแบบ I²C ในการอ่านข้อมูลของ master



รูปที่ 5 I²C Communication Protocol (Master read)

ใน Hi-Tech PIC มี library สำหรับการสื่อสารแบบ I²C อยู่แล้วคือ i2c.h และ i2c.c เราสามารถคัดลอกไฟล์มาไว้ในโปรเจ็กแล้วทำการแก้ไขเพียงเล็กน้อยก็สามารถใช้งานได้ นอกจากนี้ยังต้องทำการ include ไฟล์ที่จำเป็นซึ่งต้องคัดลอกมาไว้ในโปรเจ็กอีกคือ delay.h, delay.c และ lcd4bit.h

สำหรับการใช้งานแบบ master mode ให้ทำการแก้ไขไฟล์ i2c.h ในตำแหน่งดังนี้คือ

- ให้ทำการ comment ข้อความที่ถูก define ไว้ (บรรทัดที่ 17) ดังนี้

  ```
  /* Uncomment the next line to use the PIC's SSP Module*/
  #define I2C_MODULE 1
  ```

  เป็น

  ```
  /* Uncomment the next line to use the PIC's SSP Module*/
  // #define I2C_MODULE 1
  ```

- ทำการเปลี่ยนพอร์ต SCL และ SDA ให้เป็นพอร์ตที่เราต้องการใช้งาน (บรรทัดที่ 31-36) ดังนี้

  ```
  /* Change port as required - defaults to port b */
  #define SCL          RB2      /* clock on port B bit 2 */
  #define SCL_DIR      TRISB2

  #define SDA          RB1      /* data on port B bit 1 */
  #define SDA_DIR      TRISB1
  #define I2CTRIS TRISB
  ```

เมื่อกำหนดค่าใน i2c.h แล้วเราก็จะสามารถใช้งานฟังก์ชันต่างๆ ที่ใช้ควบคุมสัญญาณ SCL และ SDA ได้อย่างถูกต้อง

สำหรับการเขียนข้อมูลไปยังอุปกรณ์ ต้องเรียกใช้งานอุปกรณ์ตัวนั้นด้วยการบอก address ของอุปกรณ์ที่ต้องการจะเขียนข้อมูลก่อนด้วยฟังก์ชัน i2c_WriteTo(device address) จากนั้นส่ง address ของหน่วยความจำหรือรีจีสเตอร์ และข้อมูลที่ต้องการเขียนตามลำดับ (ทีละ 1 ไบต์) ด้วยฟังก์ชัน i2c_PutByte(memory address) และ i2c_PutByte(data) ซึ่งจะคืนค่ากลับเป็นผลการส่งสัญญาณว่าผิดพลาดหรือไม่

สำหรับการอ่านข้อมูลจากอุปกรณ์ ต้องเรียกใช้งานอุปกรณ์ตัวนั้นด้วยการส่ง address ของอุปกรณ์ที่ต้องการจะอ่านข้อมูลก่อนด้วยฟังก์ชัน i2c_WriteTo(device address) จากนั้นส่ง address ของหน่วยความจำหรือรีจีสเตอร์ที่เก็บข้อมูลที่ต้องการไว้ด้วยฟังก์ชัน i2c_PutByte(data) ให้ส่ง address ของอุปกรณ์ที่ต้องการจะอ่านข้อมูลอีกครั้งด้วยฟังก์ชัน i2c_ReadFrom(device address) แล้วจึงทำการอ่านค่าที่อุปกรณ์ส่งกลับมาทีละไบต์ด้วยฟังก์ชัน i2c_GetByte(I2C_LAST)

สำหรับในโปรเจ็กนี้จะมีการเรียกใช้งานฟังก์ชันที่ใช้ในการอ่านข้อมูลจากอุปกรณ์เท่านั้น สำหรับตัวอย่างการเขียนข้อมูลในอุปกรณ์จะได้เขียนอธิบายในโปรเจ็กต่อไป

<u>โปรแกรม project10.c</u>

```
#include<pic.h>
#include<stdio.h>
#include "delay.h"
#include "i2c.h"
#include "lcd4bit.h"
__CONFIG(UNPROTECT & LVPDIS & BOREN & MCLREN & PWRTDIS & WDTDIS & INTIO );

#define DEVADDR 0xC0              // Address of I2C device

// Flash a LED on RB3 when operation fail.
void flashled(void)
{
```

```
   while(1)
   {
      RB3 = 1;
      DelayMs(200);
      RB3 = 0;
      DelayMs(200);
   }
}
// Write data "byte" to I2C device at address or regester NO. "addr"
void WriteByte(unsigned char addr, unsigned char byte)
{
   i2c_WriteTo(DEVADDR);                 // Call device address "DEVADDR".
   if (i2c_PutByte(addr)==I2C_ERROR)     // Send address "addr" to keep data.
      flashled();                        // Flash LED if communication errer.
   if (i2c_PutByte(byte)==I2C_ERROR)     // Send data "byte" for writing.
      flashled();
}

// Read data from I2C device at address or regester NO. "addr"
int ReadByte(unsigned char addr) {
   i2c_WriteTo(DEVADDR);                 // Call device address "DEVADDR".
   if (i2c_PutByte(addr)==I2C_ERROR)     // Send address "addr" to read data.
      flashled();                        // Flash LED if communication errer.
    i2c_ReadFrom(DEVADDR);               // Send read command.
   return i2c_GetByte(I2C_LAST);         // Get data then return its value.
}

void main (void)
{
   unsigned int temp;
   unsigned char tempH;
   unsigned char tempL;
   char word[17];
   // Setup oscillator
   IRCF2   =  1;
   IRCF1 =  1;
   IRCF0 =  0;              // IRCF<2:0> = 111: internal RC oscillator frequency = 8 MHz
   SCS1  =  0;
   SCS0  =  0;              // SCS<1:0> = 00: Oscilator mode defined by FOSC<2:0>
   while(!IOFS);            // Wait for the frequency to become stable

   // Setup LCD ports
   ANSEL = 0x00;            // Select every bit of port A as digital I/O
   lcd_init(1);             // Initialise LCD use hi bite of port

   TRISB3 = 0;              // Config port RB3 for indicate errer on LED.
   RB3 = 0;

   while(1)
   {
      tempH = ReadByte(0x02);            // Read first byte from register NO. 2
      tempL = ReadByte(0x03);            // Read first byte from register NO. 3
      temp = ((unsigned int)tempH << 8)|(unsigned int)tempL;       // Combine two byte.
      sprintf(word,"%d",temp);           // Convert integer value to string.
      lcd_clear();                       // Clear LCD first.
      lcd_cursor(1,1);                   // Move cursor to origin.
      lcd_display(word);                 // Display current value in 0.1 degree unit.
   }
}
```

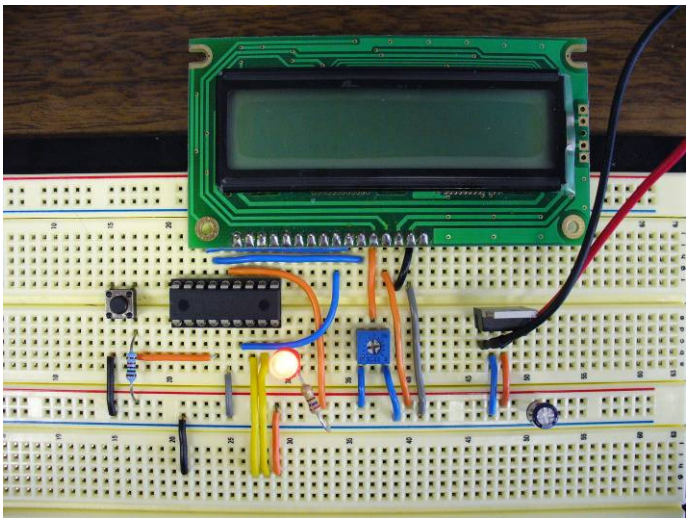
คำอธิบายโปรแกรม

โปรแกรม project10.c ถูกดัดแปลงมาจากโปรแกรม i2c_demo.c ซึ่งเป็นโปรแกรมตัวอย่างจาก Hi-Tech PIC มีฟังก์ชันที่สำคัญคือ ฟังก์ชันสำหรับการเขียนข้อมูลไปยังอุปกรณ์ I²C ฟังก์ชันสำหรับการอ่านข้อมูลจากอุปกรณ์ I²C ทั้งสองฟังก์ชันนี้รวมและลำดับการทำงานของฟังก์ชันจาก i2c.h เพื่อการเขียนหรืออ่านข้อมูลแบบ I²C ผู้ใช้เพียงกำหนด device address ของอุปกรณ์ในตอนต้นของโปรแกรม (ในโปรเจ็กนี้คือ #define DEVADDR 0xC0) หลังจากนั้นก็สามารถเรียกฟังก์ชัน WriteByte(addr, byte) โดยที่ addr คือ address ที่ต้องการเขียน และ byte คือข้อมูลที่ต้องการเขียน เพื่อทำการเขียนข้อมูลได้ทันที หรือเรียกฟังก์ชัน ReadByte(addr) แล้วรับค่าที่ได้จากการอ่านจากฟังก์ชันได้ทันทีเช่นเดียวกัน ฟังก์ชัน main() ในโปรแกรมนี้ทำหน้าที่วนลูปรับค่าองศาของเข็มทิศจากคำสั่ง tempH = ReadByte(0x02); และ tempL = ReadByte(0x03); (0x02 และ 0x03 เป็นรีจิสเตอร์ที่เก็บ

ผลลัพธ์ไว้) แล้วจึงรวมข้อมูลทั้งสองไบต์แปลงเป็นข้อมูลตัวอักษรเพื่อแสดงออกทาง LCD เท่านั้น ส่วนฟังก์ชัน flashled() สำหรับแสดงความผิดพลาดในการสื่อสารผ่าน I^2C ด้วย LED ซึ่งจะทำให้ LED ที่ต่อไว้ที่พอร์ต RB3 กระพริบตลอดเมื่อเกิดความผิดพลาด

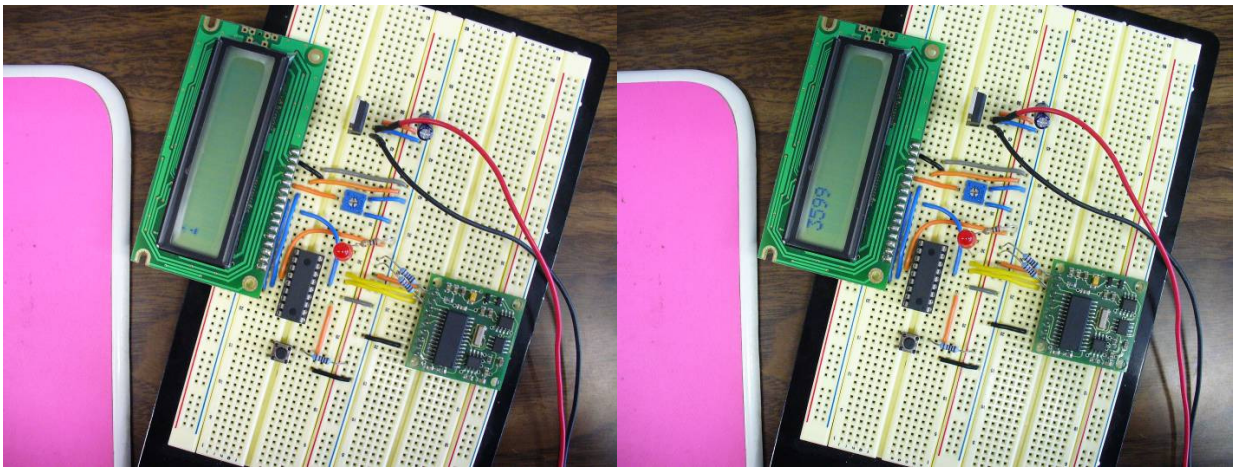
4) การทดสอบการทำงาน

ในกรณีที่ไม่มีการต่ออุปกรณ์ I^2C ไว้ และไม่มี $R_{pull\ up}$ ไฟ LED ที่ต่ออยู่กับ port RB3 จะกระพริบดังรูปที่ 6



รูปที่ 6 รูปแสดงผลการทำงานของ LED เมื่อการสื่อสารผิดพลาด

รูปที่ 7 LCD แสดงผลการอ่านค่าจาก CMPS03 ได้อย่างถูกต้อง สังเกตจากผลขององศาอยู่ในช่วง 0-3599 เท่านั้นซึ่งมีหน่วยเป็น 10^{-1} ° หากหมุนเกิน 3599 ก็จะวนกลับมาที่ 0 อีกครั้ง ตามที่ datasheet อธิบาย



รูปที่ 7 รูปแสดงผลการทดลองต่อวงจรเพื่ออ่านข้อมูลจาก CMPS03