2018/12/17

1071 單晶片系統實作

作業 3：伺服馬達、搖桿、紅外線感測器、超音波感測器、溫溼度感測器及 LCD 之整合應用

中國文化大學 資訊工程學系 3B A5216047 政弘 王

目錄

圖目錄

一　題目

題目:設計一個單晶片系統具備以下功能：利用指撥開關切換 3 種模式：

(1) 模式 1：可以利用搖桿控制兩個伺服馬達的轉動角度，並將兩個伺服馬達之角度顯示在 LCD 上。(例如：X: 56　Y:90)

(2) 模式 2：利用紅外線感測器感測路徑(貼上一條黑膠帶)，並在 LCD 上顯示微控板之位置狀態：位置正常(Position：　Normal)、微偏右(Position：Slightly Right)、偏右(Position： Right)、微偏左(Position： Slightly Left)、偏左(Position： Left)、不正常(Position： Abnormal)。

(3) 模式 3：可利用超音波感測器感測距離，並使用溫溼度感測器感測溫度與濕度，然後將感測到的距離及溫溼度顯示在 LCD 上。

(例如:Distance: 30 cm　Temp:30 Humid:63)

作業報告內容：

(1) 封面

(2) 目錄(含報告目錄、圖目錄，要利用 Word 自動產生)

(3) 題目

(4) 程式碼(要有註解)

(5) 靜態展示(成果照片)

(6) 動態展示(拍成影片及配音說明)

(7) 學習心得

二　題目：單晶片系統

1. 說明

題目： 伺服馬達、搖桿、紅外線感測器、超音波感測器、溫溼度感測器及 LCD 之整合應用

接腳:

DIP switch:

Model 1:　接微控板的 2 腳

Model 2:　接微控板的 3 腳

Model 3:　接微控板的 4 腳

LCD(I2C 匯流排):

SDA : 接微控板的 A4 腳

SCL : 接微控板的 A5 腳

Servomotors:

Servomotor1 : 接微控板的 5 腳

Servomotor2 : 接微控板的 6 腳

Joystick :

Horizontal: 接微控板的 A0 腳

Vertical:　接微控板的 A1 腳

Infrared :

right:　接微控板的 11 腳

center: 接微控板的 10 腳

left:　接微控板的　9 腳

Ultrasonic:

trigger:　接微控板的 12 腳

echo:　接微控板的 13 腳

Temperature and humidity:

data:　接微控板的 A3 腳

功能:

(1) 模式 1：

可以利用搖桿控制兩個伺服馬達的轉動角度，並將兩個伺服馬達之角度顯示在 LCD 上。

(例如：X: 56 Y:90)

(2) 模式 2：

利用紅外線感測器感測路徑(貼上一條黑膠帶)，並在 LCD 上顯示微控板之位置狀態：

位置正常(Position： Normal)、

微偏右(Position： Slightly Right)、

偏右(Position： Right)、

微偏左(Position： Slightly Left)、

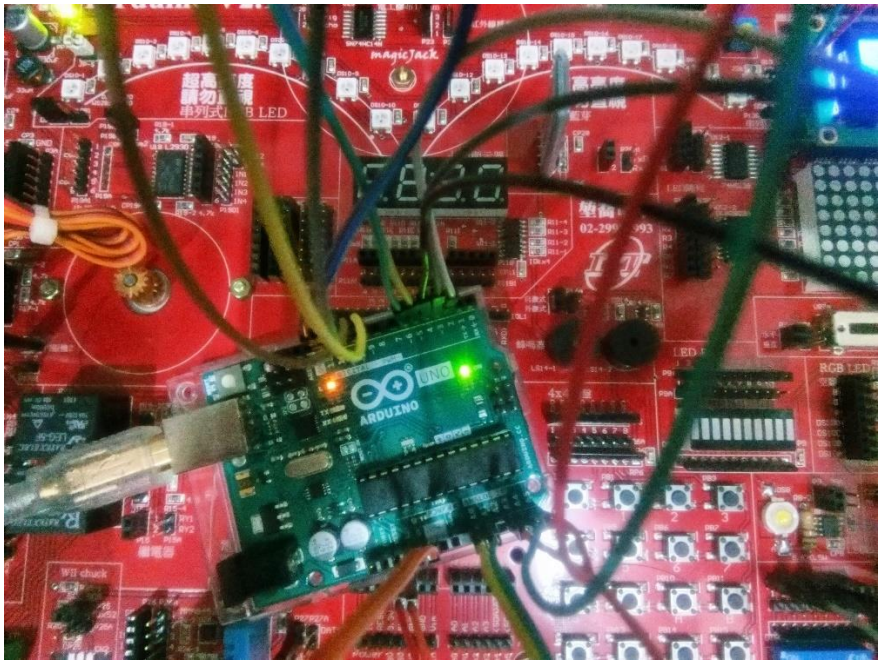
偏左(Position： Left)、

不正常(Position： Abnormal)

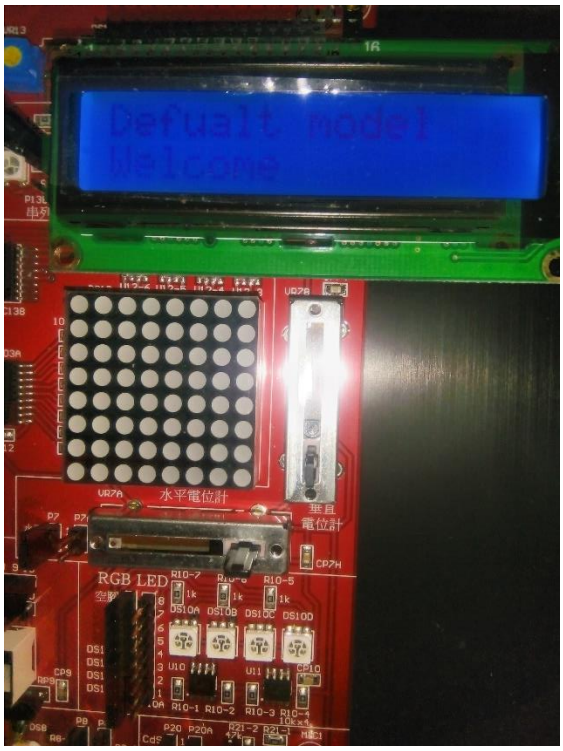
(3) 模式 3：

可利用超音波感測器感測距離，並使用溫溼度感測器感測溫度與濕度，然後將感測到的距離及溫溼度顯示在 LCD 上。(例如:Distance: 30 cm Temp:30 Humid:63)

2. 成品



圖表 1 系統成品-UNO 板



圖表 2 系統成品-LCD

3. Pseudocode

```
Required outside libraries:
    Adafruit_Sensor: https://github.com/adafruit/Adafruit_Sensor
    DHT-sensor-library: https://github.com/adafruit/DHT-sensor-library
    Arduino-LiquidCrystal-I2C-library: https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library
```

圖表 3 虛擬碼:使用到的外部函式庫

```
LCD Address{
    A0  A1  A2  Hex Address
    1   1   1   0x27
    0   1   1   0x26
    1   0   1   0x25
    0   0   1   0x24
    1   1   0   0x23
    0   1   0   0x22
    1   0   0   0x21
    0   0   0   0x20
}
```

圖表 4 虛擬碼:LCD Address 研究

```
Define pins{
    DIP switch{
        define pin of DIP switch 1
        define pin of DIP switch 2
        define pin of DIP switch 3
    }

    LCD(I2C LCD){
        define pin of SDA
        define pin of SCL
    }

    Servomotors 1{
        define pin of PWM
    }

    Servomotors 2{
        define pin of PWM
    }

    Joystick{
        define pin of horizontal
        define pin of vertical
    }

    Infrared sensor{
        define pin of right
        define pin of center
        define pin of left
    }

    Ultrasonic sensor{
        define pin of trigger
        define pin of echo
    }

    Temperature and humidity sensor{
        define pin of data
    }
}
```

圖表 5 虛擬碼:會用到的接腳

```
Initializing{
    Setup all pins
    Initialize system
}
```

圖表 6 虛擬碼:初始化系統

```
always{
    status = Get DIP switch status(Priority: 1>2>3, All close: 0)

    switch(status){
        case 0:
            Default status:
                Show default message
            break;
        case 1:
            Control two Servomotors to rotate.
            Show the angle on LCD.
            break;
        case 2:
            Infrared sensor walking correction.

            switch (Position) {
                case 1:
                    Normal
                    break;
                case 2:
                    Slightly Right
                    break;
                case 3:
                    Right
                    break;
                case 4:
                    Slightly Left
                    break;
                case 5:
                    Left
                    break;
                default:
                    Abnormal
                    break;
            }
```

圖表 7 虛擬碼:主程式-1

```
102
103             Show the status on LCD.
104             break;
105         case 3:
106             Sensing distance by ultrasonic sensor.
107             Sensing temperature and humidity by temperature and humidity sensor.
108             Show all data on LCD.
109             break;
110         default:
111             Show error message on LCD.
112             break;
113     }
114 }
```

圖表 8 虛擬碼:主程式-2

構思說明:

我先以構成此系統的感測器做參考引入所需的外部函式庫，接著思考 LCD 所要用的編碼問題。

因為這次的接線較為複雜，所以我在進入設計前先將所需的接腳列出。

而主程式我採用階層式的邏輯思考方式-先判斷是在什麼模式，更具所在模式進行詳細操作。

4. 程式碼

1. 主程式

```
55  // Include libraries:
56  #include "model_1.h" // 模式1的函式庫
57  #include "model_2.h" // 模式2的函式庫
58  #include "model_3.h" // 模式3的函式庫
59  #include "My_LCD.h"  // LCD library
60  //-----------------------------------------
```

圖表 9 主程式:引入函數庫

```
64  // Output pins:
65  // Servo motors:
66  int Servomotors1 = 5;
67  int Servomotors2 = 6;
68  // - - - - - - - - - - - - - - - - - - - - - -
69
70  // =========================================
71
72  // Input pins:
73  // DIP switch:
74  const int DIP_1 = 2;
75  const int DIP_2 = 3;
76  const int DIP_3 = 4;
77  // - - - - - - - - - - - - - - - - - - - - - -
78
79  // Joystick:
80  const int Horizontal = A0;
81  const int Vertical   = A1;
82  // - - - - - - - - - - - - - - - - - - - - - -
83
84  // Infrared:
85  const int left      =  9;
86  const int center    = 10;
87  const int right     = 11;
88  // - - - - - - - - - - - - - - - - - - - - - -
89
90  // Temperature and humidity:
91  int data = A3;
92  // - - - - - - - - - - - - - - - - - - - - - -
93
94  // =========================================
95
96  // Mixed pins:
97  // Ultrasonic:
98  const int trigger = 12;
99  const int echo    = 13;
100 // - - - - - - - - - - - - - - - - - - - - - -
101 // =========================================
```

圖表 10 主程式:接腳宣告

```
103   // Data:
104   const char* title     = ""; // title string
105   const char* message   = ""; // message string
106
107   Model_1* model_1;    // 模式1的物件
108   Model_2* model_2;    // 模式2的物件
109   Model_3* model_3;    // 模式3的物件
110   My_LCD* LCD;         // LCD object
111   // ========================================
```

圖表 11 主程式:物件及資料宣告

```
115 // Initializing
116 void setup(){
117     model_1 = new Model_1(Servomotors1, Servomotors2, Horizontal, Vertical);
118     model_2 = new Model_2(left, center, right);
119     model_3 = new Model_3(trigger,echo, data);
120     LCD = new My_LCD();
121     Serial.begin(9600);
122 }
123 //----------------------------------------
```

圖表 12 主程式:系統初始化及物件建立

```
125   // main program
126   void loop(){
127       if(!digitalRead(DIP_1)){          // Model 1
128           model_1->runWork();
129           title = model_1->getTitle();
130           message = model_1->getMessage();
131       }else if(!digitalRead(DIP_2)){    // Model 2
132           model_2->runWork();
133           title = model_2->getTitle();
134           message = model_2->getMessage();
135       }else if(!digitalRead(DIP_3)){    // Model 3
136           model_3->runWork();
137           title = model_3->getTitle();
138           message = model_3->getMessage();
139       }else{                            // Defualt model
140           title   = "Defualt model";
141           message = "Welcome";
142       }
143
144       LCD->set_words(title, message); // Display on LCD
145
146       // For Debug:
147       // Serial.println(title);
148       // Serial.println(message);
149       // ========================================
150   }
151   //----------------------------------------
```

圖表 13 主程式:主要運行程式

2. LCD 處理物件

Header:

```
16  // include libraries:
17  #include <Arduino.h>            // Ardruino base library
18  #include <Wire.h>               // Wire.h
19  #include "LiquidCrystal_I2C.h"  // Liquid Crystal(I2C) library
20  //-----------------------------------------
```

圖表 14 LCD 處理物件:引入函數庫

```
23  public:
24      // Constructors:
25
26      // Constructor: default
27      My_LCD(/* args */);
28
29      // Constructor: input(rows, column)
30      My_LCD(int rows, int column);
31      // ========================================
32
33      // Deconstructor
34      ~My_LCD();
35      // ========================================
36
37      // Functions:
38
39      // Setting Functions:
40
41      // Setting display words
42      void set_words(char* title, char* message);
43      void set_words(char* title, int title_size, char* message, int message_size);
44      // - - - - - - - - - - - - - - - - - - - -
45
46      // Getting Functions:
47
48      // Getting start cursor of words(Alignment: center)
49      int get_start_cursor(int words_size);
50      // - - - - - - - - - - - - - - - - - - - -
51
52      // ========================================
```

圖表 15 LCD 處理物件:公有函式宣告

```
53  private:
54      // Data:
55      LiquidCrystal_I2C* LCD;      // LiquidCrystal Object
56      const int Address = 0x27;    // I2C address
57      int row = 2;                 // How many rows (defualt: 2)
58      int column = 16;             // How may column in one row (defualt: 16)
59
60      // 自建字型編碼:
61      byte _left[8] = {
62          0b00010,
63          0b00110,
64          0b00110,
65          0b01110,
66          0b01110,
67          0b11110,
68          0b11110,
69          0b00000
70      };
71
72      byte _right[8] = {
73          0b01000,
74          0b01100,
75          0b01100,
76          0b01110,
77          0b01110,
78          0b01111,
79          0b01111,
80          0b00000
81      };
82      // - - - - - - - - - - - - - - - - - - - - - - -
```

圖表 16 LCD 處理物件:私有資料

Definer:

```
3   // Constructor: default
4   My_LCD::My_LCD(){
5       this->LCD = new LiquidCrystal_I2C(this->Address, this->column, this->row);    // Initialize LCD object
6       LCD->begin();               // LCD初始化
7       LCD->backlight();           // 開啟背光
8       LCD->createChar(0, _left);  // 載入自建字型編碼陣列
9       LCD->createChar(1, _right); // 載入自建字型編碼陣列
10  }
11  //-----------------------------------------
12
13  // Constructor: input(rows, column)
14  My_LCD::My_LCD(int rows, int column):row(row), column(column){
15      this->LCD = new LiquidCrystal_I2C(this->Address, this->column, this->row);    // Initialize LCD object
16      LCD->begin();               // LCD初始化
17      LCD->backlight();           // 開啟背光
18      LCD->createChar(0, _left);  // 載入自建字型編碼陣列
19      LCD->createChar(1, _right); // 載入自建字型編碼陣列
20  }
21  //-----------------------------------------
```

圖表 17 LCD 處理物件:建構子定義

```
23    // Deconstructor
24    My_LCD::~My_LCD(){
25
26    }
27    //--------------------------------------------
```

圖表 18 LCD 處理物件:解構子定義

```
29    // setting display words
30    void My_LCD::set_words(char* title, char* message){
31        LCD->setCursor(0, 0);    // 移至第0列的指標開始位置
32        LCD->print(title);       // display title
33
34        LCD->setCursor(0, 1);    // 移至第1列的指標開始位置
35        LCD->print(message);     // display message
36
37        delay(500);
38        LCD->clear();
39    }
40    //--------------------------------------------
```

圖表 19 LCD 處理物件:顯示文字(靠左對齊)

```
42  // setting display words
43  void My_LCD::set_words(char* title, int title_size, char* message, int message_size){
44
45      LCD->setCursor(get_start_cursor(title_size), 0);    // 移至第0列的指標開始位置
46      LCD->print(title);       // display title
47
48      LCD->setCursor(get_start_cursor(message_size), 1);  // 移至第1列的指標開始位置
49      LCD->print(message);     // display message
50
51      //delay(500);
52      LCD->clear();
53  }
54  //--------------------------------------------
```

圖表 20 LCD 處理物件:顯示文字(置中對齊)

```
56  // Getting start cursor of words(Alignment: center)
57  int My_LCD::get_start_cursor(int words_size){
58      return ( (this->column - words_size)>0?(this->column - words_size):0 );
59  }
60  //--------------------------------------------
61
```

圖表 21 LCD 處理物件:置中對齊 LCD 指標取得函式

3. 模式 1 物件

Header:

```
12   // include libraries:
13   #include <Arduino.h>    // Ardruino base library
14   #include <Servo.h>      // Servo working library
15   //-----------------------------------------
```

圖表 22 模式 1:引入函數庫

```
18   public:
19       // Constructor: input(pin of bottom servo, pin of top servo, pin of horizontal joystick, pin of vertical joystick)
20       Model_1(int servo_1, int servo_2, int joystick_H, int joystick_V);
21       // ========================================
22
23       // Deconstructor:
24       ~Model_1();
25       // ========================================
26
27       // Functions:
28
29       // 主要驅動程式
30       void runWork();
31       // - - - - - - - - - - - - - - - - - - - -
32
33       // 讀取信號並調整位置值
34       long mapping(int pin);
35       // - - - - - - - - - - - - - - - - - - - -
36
37       // Getting functions:
38       char* getTitle() const;         // Get title string
39       int getTitle_size() const;      // Get size of title
40
41       char* getMessage() const;       // Get message string
42       int getMessage_size() const;    // Get size of message
43       // - - - - - - - - - - - - - - - - - - - -
44
45       // ========================================
```

圖表 23 模式 1:公有函數宣告

```
46   private:
47       // Functions:
48       void updata();              // Updata all Data
49       void updataMessage();       // Updata message
50       void updataMessage_size();  // Updata size of message
51       // ========================================
52
53       // Data:
54       // About LCD:
55       static const char* const title = "Angle";         // title
56       int title_size = sizeof(title)/sizeof(title[0]);  // title size
57
58       const char* message = "X:%d Y:%d";                    // message
59       int message_size = sizeof(message)/sizeof(message[0]);  // message size
60       // - - - - - - - - - - - - - - - - - - - - - -
61
62       // Servo motors:
63       Servo* servo_1; // Bottom Servo Object
64       Servo* servo_2; // Top Servo Object
65       // - - - - - - - - - - - - - - - - - - - - - -
66
67       // Joystick:
68       int joystick_H = A0; // pin of joystick horizontal (Defualt: A0)
69       int joystick_V = A1; // pin of joystick vertical (Defualt: A1)
70       // - - - - - - - - - - - - - - - - - - - - - -
71
72       // ========================================
```

圖表 24 模式 1:私有函數及資料宣告

Definer:

```
// Constructor: input(pin of bottom servo, pin of top servo, pin of horizontal joystick, pin of vertical joystick)
Model_1::Model_1(int servo_1, int servo_2, int joystick_H, int joystick_V):joystick_H(joystick_H), joystick_V(joystick_V){
    // Initializing Servo Object:
    this->servo_1 = new Servo();
    this->servo_2 = new Servo();
    this->servo_1->attach(servo_1);
    this->servo_2->attach(servo_2);
    // ========================================

    // setting out pins:
    pinMode(this->joystick_H, INPUT);
    pinMode(this->joystick_V, INPUT);
    // ========================================
}
//-----------------------------------------
```

圖表 25 模式 1:建構子定義

```
// Deconstructor
Model_1::~Model_1(){

}
//-----------------------------------------------
```

圖表 26 模式 1: 解構子定義

```
// 讀取信號並調整位置值
long Model_1::mapping(int pin){
    return map(analogRead(pin), 0, 1023, 179, 0);
}
//-----------------------------------------
```

圖表 27 模式 1:信號讀取轉換函式

```
// 主要驅動程式
void Model_1::runWork(){
    servo_1->write( mapping(this->joystick_H) ); // 驅動水平伺服機
    servo_2->write( mapping(this->joystick_V) ); // 驅動垂直伺服機

    updata();
}
//-----------------------------------------
```

圖表 28 模式 1:主運行函式

```
// 讀取信號並調整位置值
long Model_1::mapping(int pin){
    return map(analogRead(pin), 0, 1023, 179, 0);
}
//--------------------------------------------

// Get title string
char* Model_1::getTitle() const{
    return (char*)title;
}
//--------------------------------------------

// Get size of title
int Model_1::getTitle_size() const{
    return title_size;
}
//--------------------------------------------

// Get message string
char* Model_1::getMessage() const{
    return (char*)message;
}
//--------------------------------------------

// Get size of message
int Model_1::getMessage_size() const{
    return message_size;
}
//--------------------------------------------
```

圖表 29 模式 1:Getting function

```
// Updata all Data
void Model_1::updata(){
    updataMessage();
    updataMessage_size();
}
//----------------------------------------

// Updata message
void Model_1::updataMessage(){
    // Updata message
    sprintf((char*)message,"X:%i Y:%i", (int)mapping(this->joystick_H), (int)mapping(this->joystick_V));
}
//----------------------------------------

// Updata size of message
void Model_1::updataMessage_size(){
    message_size = sizeof(message)/sizeof(message[0]); // Updata size of message
}
//----------------------------------------
```

圖表 30 模式 1:資料更新函式

4. 模式 2 物件

Header:

```
19  // include libraries:
20  #include <Arduino.h>    // Ardruino base library
21  //-----------------------------------------
```

圖表 31 模式 2:引入函式庫

```
24  public:
25      // Constructor:(input: pin of left, pin of center, pin of right)
26      Model_2(int left, int center, int right);
27      // =========================================
28
29      // Deconstructor:
30      ~Model_2();
31      // =========================================
32
33      // Functions:
34
35      // 主要驅動程式
36      void runWork();
37      // - - - - - - - - - - - - - - - - - - - - -
38
39      // Getting functions:
40      char* getTitle() const;         // Get title string
41      int getTitle_size() const;      // Get size of title
42
43      char* getMessage() const;       // Get message string
44      int getMessage_size() const;    // Get size of message
45      // - - - - - - - - - - - - - - - - - - - - -
46
47      // =========================================
```

圖表 32 模式 2:公有函數宣告

```
48  private:
49      // Functions:
50      void updata();              // Updata all data
51      void updataMessage();       // Updata message
52      void updataMessage_size();  // Updata size of message
53      // =========================================
54
55      // Data:
56      // About LCD:
57      static const char* const title = "Position";      // title
58      int title_size = sizeof(title)/sizeof(title[0]);  // title size
59
60      const char* message = "123";                            // message
61      int message_size = sizeof(message)/sizeof(message[0]);  // message size
62      // - - - - - - - - - - - - - - - - - - - - -
63
64      // Aoubt Infrared sensor:
65      int left    = 2;    // 紅外線感測器(左)接腳  (Defualt: 2)
66      int center  = 3;    // 紅外線感測器(中)接腳  (Defualt: 3)
67      int right   = 4;    // 紅外線感測器(右)接腳  (Defualt: 4)
68
69      int LEFT    = 0;    // 存放紅外線感測器(左)狀態
70      int CENTER  = 0;    // 存放紅外線感測器(中)狀態
71      int RIGHT   = 0;    // 存放紅外線感測器(右)狀態
72      // - - - - - - - - - - - - - - - - - - - - -
73
74      // =========================================
```

圖表 33 模式 2:私有函數及資料宣告

Definer:

```
// Constructor:(input: pin of left, pin of center, pin of right)
Model_2::Model_2(int left, int center, int right) : left(left), center(center), right(right)
{
    // 設定為輸入接腳:
    pinMode(this->left, INPUT);
    pinMode(this->center, INPUT);
    pinMode(this->right, INPUT);
    // ========================================
}
//-----------------------------------------
```

圖表 34 模式 2:建構子定義

```
// Deconstructor:
Model_2::~Model_2()
{
}
//-----------------------------------------------
```

圖表 35 模式 2:解構子定義

```
// 主要驅動程式
void Model_2::runWork()
{
    LEFT   = digitalRead(this->left);     // 讀取紅外線感測器(左)
    CENTER = digitalRead(this->center);   // 讀取紅外線感測器(中)
    RIGHT  = digitalRead(this->right);    // 讀取紅外線感測器(右)

    updata();
}
//-----------------------------------------
```

圖表 36 模式 2:主運行函式

```
// Get title string
char* Model_2::getTitle() const{
    return (char*)title;
}
//-----------------------------------------

// Get size of title
int Model_2::getTitle_size() const{
    return title_size;
}
//-----------------------------------------

// Get message string
char* Model_2::getMessage() const{
    return (char*)message;
}
//-----------------------------------------

// Get size of message
int Model_2::getMessage_size() const{
    return message_size;
}
//-----------------------------------------
```

圖表 37 模式 2:Getting function

```
// Updata all data
void Model_2::updata(){
    updataMessage();
    updataMessage_size();
}
//-----------------------------------------

// Updata message
void Model_2::updataMessage(){
    if ((LEFT || CENTER || RIGHT) == 0 || (LEFT && CENTER && RIGHT) == 1) // 不正常狀態
    {
        message = "Abnormal";
    }
    else if ((!LEFT && CENTER && !RIGHT) == 1)   // 位置正常狀態
    {
        message = "Normal";
    }
    else if ((!LEFT && !CENTER && RIGHT) == 1)  // 微偏右狀態
    {
        message = "Slightly Right";
    }
    else if ((!LEFT && CENTER && RIGHT) == 1)   // 偏右狀態
    {
        message = "Right";
    }
    else if ((LEFT && !CENTER && !RIGHT) == 1)  // 微偏左狀態
    {
        message = "Slightly Left";
    }
    else if ((LEFT && CENTER && !RIGHT) == 1)   // 偏左狀態
    {
        message = "Left";
    }
}
//-----------------------------------------

// Updata size of message
void Model_2::updataMessage_size(){
    message_size = sizeof(message)/sizeof(message[0]); // Updata size of message
}
//-----------------------------------------
```

圖表 38 模式 2: 資料更新函式

5. 模式 3 物件

Header:

```
16  // include libraries:
17  #include <Arduino.h>       // Ardruino base library
18  #include <DHT.h>          // Temperature and humidity sensor library
19  //-----------------------------------------
```

圖表 39 模式 3:引入函式庫

```
22  public:
23      // Constructor:(input: pin of trigger, pin of echo, pin of data)
24      Model_3(int trigger, int echo, int data);
25      // ========================================
26
27      // Deconstructor:
28      ~Model_3();
29      // ========================================
30
31      // Functions:
32
33      // 主要驅動程式
34      void runWork();
35      // - - - - - - - - - - - - - - - - - - - - -
36
37      // Getting functions:
38      char* getTitle() const;          // Get title string
39      int getTitle_size() const;       // Get size of title
40
41      char* getMessage() const;        // Get message string
42      int getMessage_size() const;     // Get size of message
43      // - - - - - - - - - - - - - - - - - - - - -
44
45      // ========================================
```

圖表 40 模式 3:公有函數宣告

```
46  private:
47      // Functions:
48      void updata();              // Updata all data
49      void updataTitle();         // Updata title
50      void updataTitle_size();    // Updata size of title
51      void updataMessage();       // Updata message
52      void updataMessage_size();  // Updata size of message
53      void updataTempHumi();      // Updata temperature and humidity
54      void updataDistance();      // Updata distance
55      // ========================================
56
57      // Data:
58      // About LCD:
59      const char* title = "Distance:%d cm";             // title
60      int title_size = sizeof(title)/sizeof(title[0]);  // title size
61
62      const char* message = "Temp:%d Humid:%d";               // message
63      int message_size = sizeof(message)/sizeof(message[0]);  // message size
64      // - - - - - - - - - - - - - - - - - - - - - -
65
66      // About temperature and humidity sensor:
67      DHT* dht11; // temperature and humidity sensor object
68
69      float temperature   = 0.0; // 存放濕度值的變數
70      float humidity      = 0.0; // 存放攝氏溫度值的變數
71      // - - - - - - - - - - - - - - - - - - - - - -
72
73      // About ultrasonic sensor:
74      const int samplePeriod = 50;    // 每50ms取樣一次(頻率為20Hz)
75      const int T25=29;               // 25度時之音速比例(1/0.0346)
76      int distance = 0;
77      int duration = 0;
78
79      int trigger = 2; // pin of trigger  (defualt: 3)
80      int echo    = 3; // pin of echo     (defualt: 2)
81      // - - - - - - - - - - - - - - - - - - - - - -
82
83      // ========================================
```

圖表 41 模式 3:私有函數及資料宣告

Definer:

```
3   // Constructor:(input: pin of trigger, pin of echo, pin of data)
4   Model_3::Model_3(int trigger, int echo, int data): trigger(trigger), echo(echo){
5       pinMode(trigger, OUTPUT);
6       pinMode(echo, INPUT);
7
8       dht11 = new DHT(data, DHTTYPE);
9   }
10  //-----------------------------------------
```

圖表 42 模式 3:建構子定義

```
12  // Deconstructor
13  Model_3::~Model_3(){
14
15  }
16  //-----------------------------------------
```

圖表 43 模式 3:解構子定義

```
18  // 主要驅動程式
19  void Model_3::runWork(){
20      updata();
21  }
22  //-----------------------------------------
```

圖表 44 模式 3:主運行函式

```
24  // Get title string
25  char* Model_3::getTitle() const{
26      return (char*)title;
27  }
28  //-----------------------------------------
29
30  // Get size of title
31  int Model_3::getTitle_size() const{
32      return title_size;
33  }
34  //-----------------------------------------
35
36  // Get message string
37  char* Model_3::getMessage() const{
38      return (char*)message;
39  }
40  //-----------------------------------------
41
42  // Get size of message
43  int Model_3::getMessage_size() const{
44      return message_size;
45  }
46  //-----------------------------------------
```

圖表 45 模式 3:Getting function

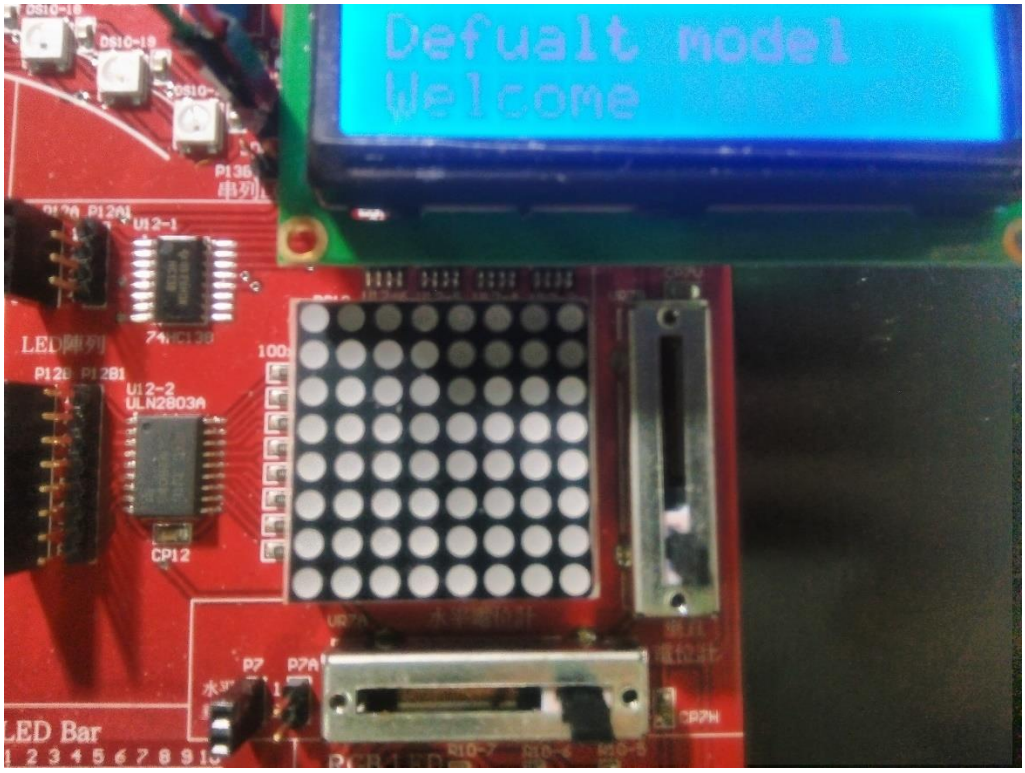
```
48  // Updata all data
49  void Model_3::updata(){
50      updataDistance();
51      updataTempHumi();
52  }
53  //-----------------------------------------
54
55  // Updata title
56  void Model_3::updataTitle(){
57      sprintf((char*)title,"Distance:%i cm", distance);
58      updataTitle_size();
59  }
60  //-----------------------------------------
61
62  // Updata size of title
63  void Model_3::updataTitle_size(){
64      title_size = sizeof(title)/sizeof(title[0]);
65  }
66  //-----------------------------------------
67
68  // Updata message
69  void Model_3::updataMessage(){
70      sprintf((char*)message,"Temp:%i Humid:%i", (int)this->temperature, (int)this->humidity);
71      updataMessage_size();
72  }
73  //-----------------------------------------
74
75  // Updata size of message
76  void Model_3::updataMessage_size(){
77      message_size = sizeof(message)/sizeof(message[0]);
78  }
79  //-----------------------------------------
```

圖表 46 模式 3: 資料更新函式-1

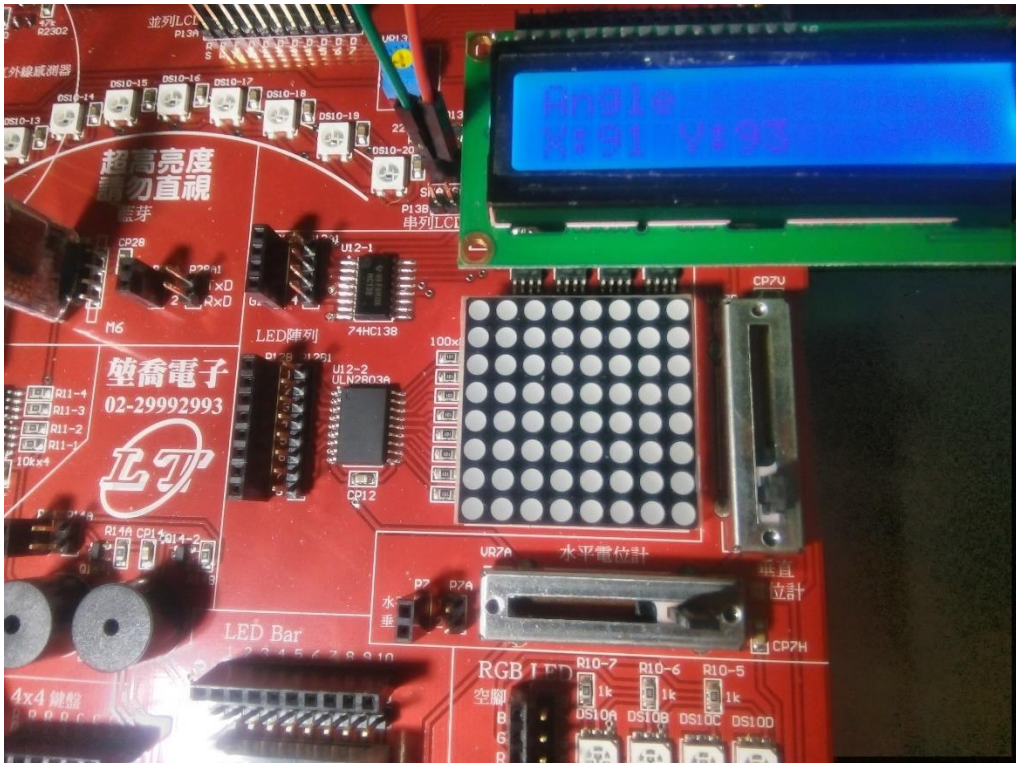
```
81   // Updata temperature and humidity
82   void Model_3::updataTempHumi(){
83       // 若讀取dht11無誤，則執行下列動作
84       this->temperature   = dht11->readTemperature(); // Read temperature as Celsius
85       this->humidity      = dht11->readHumidity();    // Read humidity
86       updataMessage();
87   }
88   //-----------------------------------------
89
90   // Updata distance
91   void Model_3::updataDistance(){
92       digitalWrite(trigger, HIGH);            // 持續送出超音波
93       delayMicroseconds(10);                  // 等待10微秒
94       digitalWrite(trigger, LOW);             // 停止送出超音波
95       duration = pulseIn(echo, HIGH)/2;       // 單趟時間
96       distance = duration/T25;                // 距離
97       updataTitle();
98   }
99   //-----------------------------------------
```

圖表 47 模式 3: 資料更新函式-2

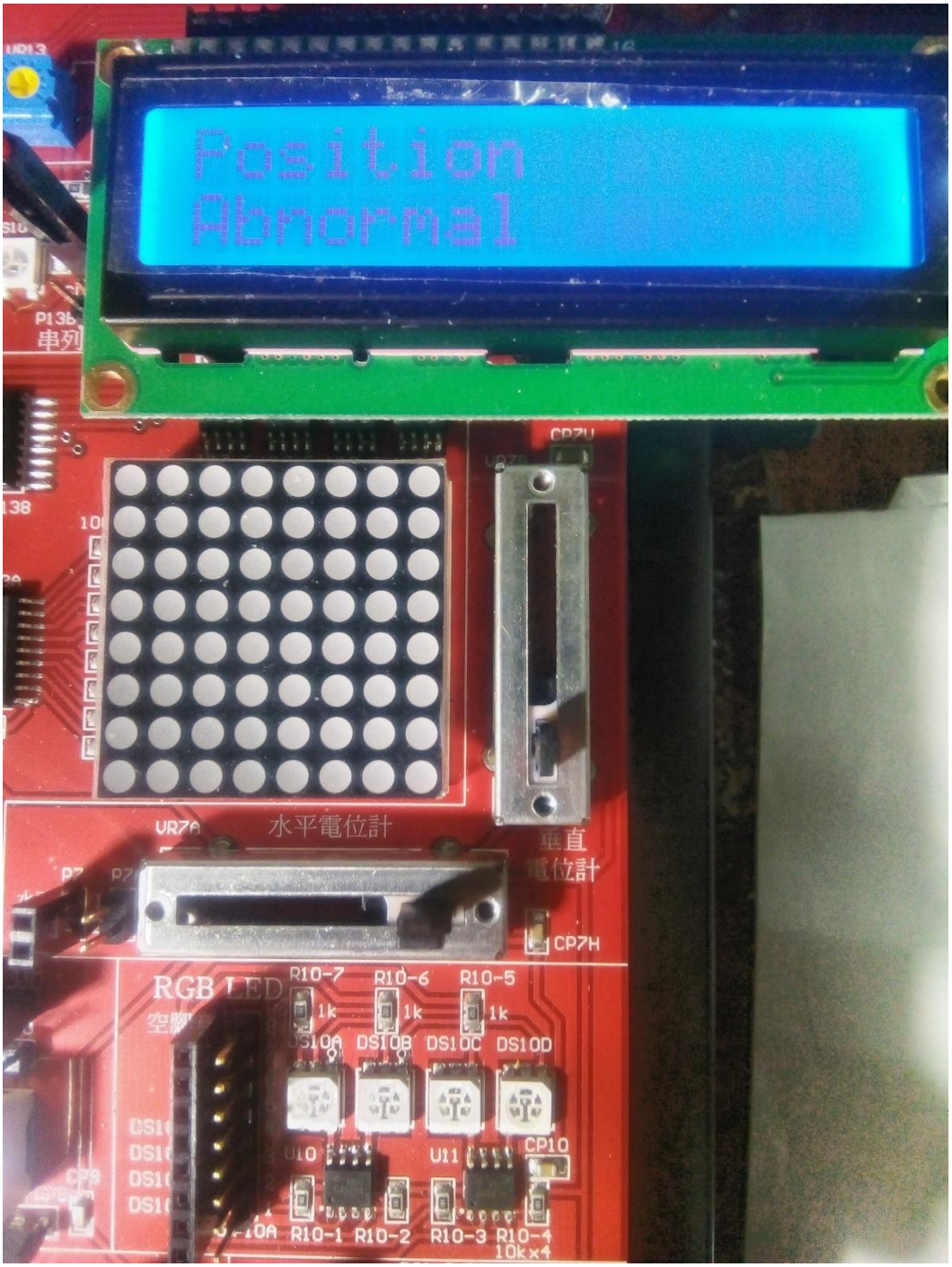
5. 靜態展示



圖表 48 靜態展示:預設模式



圖表 49 靜態展示:模式 1



圖表 50 靜態展示:模式 2

圖表 51 靜態展示:模式 3

6. 動態展示

YouTube

Google Cloud

三　學習心得

在這一整個系統的建立上，因為 Arduino 是以 C++作為主要建立的語言而使得在字串的傳輸及處理上變得較為不易，這部分讓我花了比較多的時間在處理，但也因為這樣使得我對這部分有更加深的了解。此外，在整體系統的優化上也使我花費了不少心力，因為對於 LCD 的 delay 及感測間隔問題，只能自己慢慢去嘗試才能理解設定值大概要設在多少範圍內會得到最好的結果。