ใบปฏิบัติการการเขียนโปรแกรมภาษาซี

การทดลองที่ 12

การเขียนโปรแกรมควบคุมหุ่นยนต์

วัตถุประสงค์

1. เพื่อให้ผู้อ่านสามารถเขียนโปรแกรมควบคุมหุ่นยนต์ได้
2. เพื่อให้ผู้อ่านมีทักษะในการเขียนโปรแกรมในการควบคุมอุปกรณ์ที่หลากหลายชนิดร่วมทั้งการติดต่อกับเซนเซอร์ได้
3. เพื่อให้ผู้อ่านสามารถเขียนโปรแกรมการโดยนำทฤษฎีมาประยุกต์ใช้ในการเขียนโปรแกรมได้

ความรู้พื้นฐาน

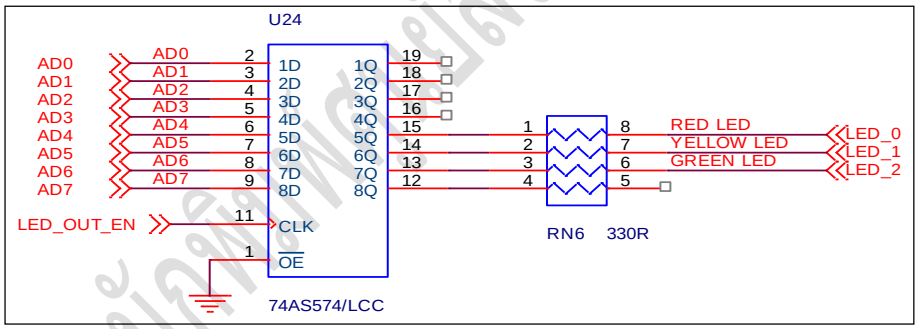
1. เรียกโปรแกรม AVR Studio 4 และโปรแกรมที่เกี่ยวข้องมาใช้งาน
2. เขียนโปรแกรมตามที่กำหนดลงบน AVR Studio 4
3. สั่งทำการ แปลภาษา (Compiler) ที่เขียนและตรวจความถูกต้องและทำการแก้ไขข้อผิดพลาดตามการแสดง Error
4. สั่งทำการให้โปรแกรมทำการเชื่อมต่อกับหุ่นยนต์
5. สังเกตการณ์ทำงานของหุ่นยนต์
6. เขียนสรุปการทดลอง

การทดลองการเขียนโปรแกรม ด้าน LED Control (ED cooperation, Online)

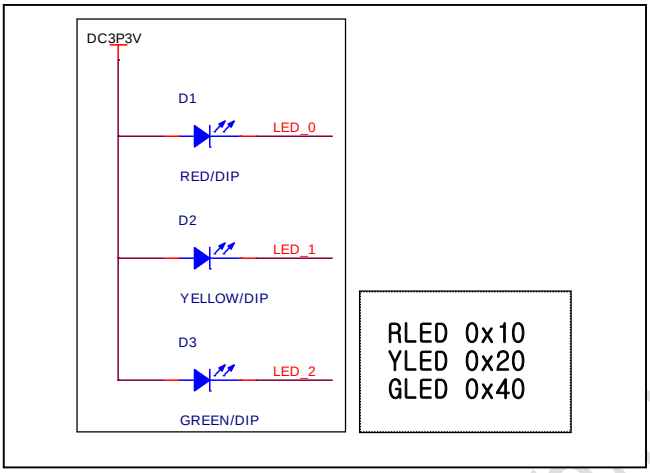
วัตถุประสงค์การทดลอง

1. ตรวจสอบการควบคุม LED บนหุ่นยนต์
2. เพื่อเรียนรู้ลักษณะการออกแบบวงจรของ LED
3. เพื่อเรียนรู้ลักษณะการเขียนโปรแกรมควบคุม LED
4. เพื่อเรียนรู้การทำงานของหุ่นยนต์

1. ศึกษาวงจรการทดลอง LED ที่ใช้ในการทดลองที่ประกอบอยู่ในหุ่นยนต์ของ ED-7275 แสดงดังรูปที่ Lab12.1 และรูปที่ Lab12.2



รูปที่ Lab12.1 แสดงวงจร LED ของ ED-7275



รูปที่ Lab12.2 แสดงวงจร LED ของ ED-7275

2. เขียนโปรแกรมการควบคุมแสดงผล LED ดังนี้

```
#include "LCD.h"
void LcdCmd(unsigned char cmd)          {
        LCD_CMD = 0x00;
        LCD_DATA = cmd;
        LCD_CMD = 0x04;
        asm volatile("NOP");
        asm volatile("NOP");
        LCD_CMD = 0x00;
        _delay_us(50);
}
```

```
void LcdData(unsigned char data)          {
        LCD_CMD = 0x01;
        LCD_DATA = data;
        LCD_CMD = 0x05;
        asm volatile("NOP");
        asm volatile("NOP");
        LCD_CMD = 0x01;
        _delay_us(50);
}
void LcdClear(void)                       {
        LcdCmd(0x01);
        _delay_ms(2);
        LcdCmd(0x80);
        _delay_us(50);
}
void LcdInit(void)                        {
        LCD_CMD = 0x00;
        _delay_ms(2);
        LcdCmd(0x38);
        LcdCmd(0x0f);
        LcdCmd(0x01);
        _delay_ms(2);
}
```

```
void LcdString(unsigned char pos, unsigned char *str)
{
        LcdCmd(pos);

        int size = strlen(str);
        int i = 0;

        for(i = 0; i < size; i++)
        {
                LcdData(str[i]);
        }
}
```

3. เมื่อทำการดาวน์โหลดโปรแกรมเสร็จแล้วให้สังเกตที่ LED บนด้านหน้าของหุ่นยนต์จากนั้นให้ทำการแก้ไขโปรแกรมให้ LED ติดเป็นเวลา 4 วินาทีแล้วบันทึก Source Code ที่ทำการแก้ไข

4. จงอธิบายขั้นตอนการส่งค่าข้อมูลไปให้กับ LED สีแดง เพื่อที่จะให้ LED สีแดงแสดงผล

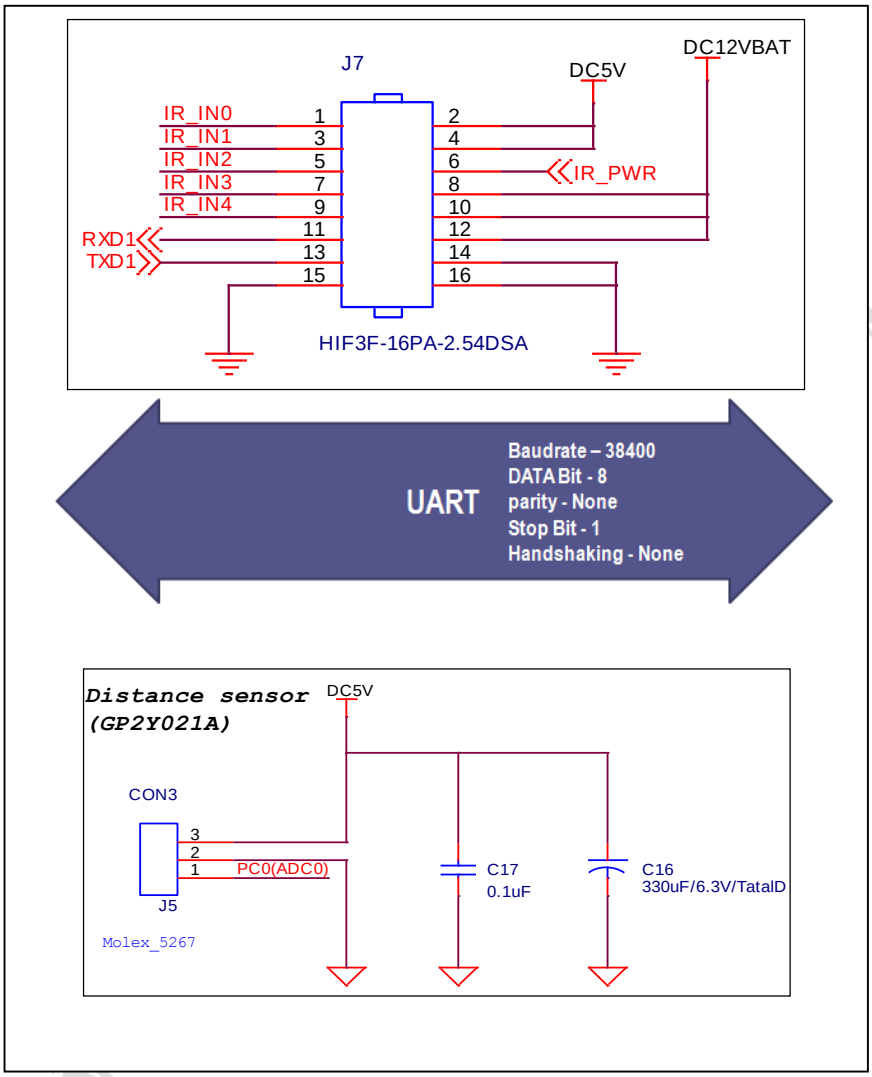
5. สรุปผลการทดลอง

การทดลองการเขียนโปรแกรมภาษาซี ด้าน PSD Sensor Module Control

วัตถุประสงค์การทดลอง

1. เพื่อเรียนรู้หลักการและขั้นตอนการรับค่ามาจากโมดูลเซนเซอร์ผ่านการสื่อสารแบบ UART
2. เพื่อเรียนรู้ลักษณะการเขียนโปรแกรมการรับ-ส่งข้อมูลจากโมดูลเซนเซอร์
3. เพื่อเรียนรู้ลักษณะวัดระยะทางของวัตถุโดยใช้ PSD Sensor Module

1. ศึกษาวงจรการทดลอง PSD Sensor Module ที่ใช้ในการทดลองที่ประกอบอยู่ในหุ่นยนต์ของ ED-7275 แสดงดังรูปที่ Lab12.3



รูปที่ Lab12.3 แสดงวงจร PSD Sensor Module

2. เขียนโปรแกรมการควบคุม PSD Sensor Module ดังโปรแกรมด้านล่าง

```
// ED-7275 PSD Sensor Operation Example
// psd.c
#include <stdio.h>                              // Include header file by
using sprintf
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "lcd.h"
volatile unsigned int psd[3]; // PSD sensor variable
// Initialize PSD sensor communication.
void psd_init(void) {
// Set a communication speed to 38400.
UBRR1H = 0;                 // BAUD Rate Hi
UBRR1L = 12;   // BAUD Rate Lo ((F_CPU / (16UL * 38400)) - 1)
// RX Interrupt Enable, Rx / Tx Enable
UCSR1B = (1<<RXEN1) | (1<<TXEN1) | (1<<RXCIE1);
UCSR1C = (1<<UCSZ1) | (1<<UCSZ0);  // no parity, 1 stop, 8 data
psd[0] = 0;
psd[1] = 0;
psd[2] = 0;
}
```

```
// Out of data sent from PSD sensor module
// Interrupt function called whenever
receiving 1byte
ISR(USART1_RX_vect) {
        static unsigned char chk=0, rxcnt=0;
        static unsigned char rxdata[7];
        unsigned char rx, status;
        status = UCSR1A;
        rx = UDR1;
        if (status & ((1<<FE)|(1<<DOR)))
        {               // If a received value has an error
        rxcnt = 0;
        }else{          // Received ISR
                if(rxcnt == 0)
        {       // Check if it is the first value.
        if(rx == 0xCC)
        {       // Save the first value and set rxcnt to 1.
                rxdata[rxcnt] = rx;
                rxcnt++;
        chk = rx;       }
        }else{  // Save a received value.
                rxdata[rxcnt] = rx;
                // Skip the check sum part.
                if(rxcnt != 5)          chk += rx;
                rxcnt++;
```

```
 if(rxcnt == 7)
{           // Check if check sum parts correspond.
            if(chk == rxdata[5])
{ // If check sum is correct, save a sensor value.
            psd[rxdata[1] - 0x11] = (unsigned int)rxdata[3] << 8 |
(unsigned int)rxdata[4];   }
            // Finish receiving 7 byte
            rxcnt = 0;
                                    }          }          }          }
// In case of newly reading the PSD sensor, clear a psd value to 0.
void psd_clear(void)        {
            psd[0] = 0;
            psd[1] = 0;
            psd[2] = 0;         }
// Read a PSD sensor value. ( id = 1, 2, 3 ) // RETURN = 0 (not
received) ,  1 (received)
unsigned char psd_read(void)       {
            unsigned char id;
            unsigned int timeout, i;
            unsigned char data[7];
            data[0] = 0xAA;                    // Start data transmission.
            data[2] = 0x07;                    // (PSD_CMD = 0x07)
            data[3] = 0;
            data[4] = 0;
            data[6] = 0x55;                    // Finish data transmission.
```

```
 for( id = 0; id<3; id++)
{
// PSD sensor no.1 0x11, no.2 0x12, no.3 0x13
data[1] = 0x11 + id;
data[5] = data[0] + data[1] + data[2] + data[3] + data[4] + data[6];
// Send a command.
for (i=0; i<7; i++)
{
        timeout = 4000;  // Approx. 250us or more
        // USART Data Register Empty
        while (!(UCSR1A & (1<<UDRE)) && timeout--);
        if(timeout == 0) break;
        // It is a transmission error.
                UDR1 = data[i]; }          }
        if(psd[0] > 0 && psd[1] > 0 && psd[2] > 0)
        {
                return 1;
        // If a return value is 1, a new sensor  value is obtained.
        }
        return 0;           // A new sensor value is not obtained yet.
}
// End of PSD sensor-related function
// test psd
```

```
int main(void)
{
        char str[21];
        unsigned char result;

  // Initialize an external memory.
  MCUCR = 0x80;

  // enable external memory and I/O
  XMCRA = 0x44;

  // 0x1100-0x7FFF=1 wait, 0x8000-0xFFFF=0 wait
  XMCRB = 0x80;

  // enable bus keeper, use PC0-PC7 as address
        lcd_init();

        // Initialize psd sensor.
        psd_init();
        SREG = 0x80;
        // Start interrupt.
        lcd_string(LCD_LINE1, "PSD SENSOR");

 // In case of getting a new PSD sensor value, clear the psd value.
        psd_clear();
```

```
while(1)
{       // Obtain a PSD sensor value in the psd[] variable.
        result = psd_read();
        if(result == 1)
        // In case of getting a new PSD sensor value, its result is 1.
          {
                // C basic function used for entering a character
                //string in str[21] array

        // sprintf (include stdio.h in the upper part)
        sprintf(str, "[3]%3d [1]%3d [2]%3d", psd[2], psd[0], psd[1]);
        lcd_string(LCD_LINE2, str);
          }
        _delay_ms(100);
        }
}
```

3. เมื่อทำการดาวน์โหลดโปรแกรมเสร็จแล้วให้สังเกตสถานะของ PSD Sensor แสดงบนจอ LCD จากนั้นทดลองนำวัตถุเพื่อตรวจสอบการตรวจจับวัสดุสีต่างๆ ที่โมดูลเซนเซอร์บนหน้าของหุ่นยนต์ดังนี้

- วัสดุสีดำ
- วัสดุสีขาว
- วัสดุที่เป็นโลหะ

4. สังเกตการตรวจจับของวัสดุและระยะทางบนหน้าจอ LCD แล้วบันทึกผล

5. ให้อธิบายขั้นตอนการอ่านค่าจากโมดูลเซนเซอร์ PSD มาแสดงผลบน LCD

6. สรุปผลการทดลอง

อุปกรณ์

1. เครื่องคอมพิวเตอร์
2. หุ่นยนต์ ED-7275
3. ตัวโปรแกรมภาษาซี หรือ
4. ตัวโปรแกรม CodeBlock
5. โปรแกรม AVR Studio
6. โปรแกรมWinAVR
7. หน่วยความจำเคลื่อนที่ (Handy Drive, External Drive)

วิธีการทดลอง

1. ให้ผู้อ่านเรียกโปรแกรมที่ใช้ในการเขียนโปรแกรมภาษาซี
2. โปรแกรมภาษาซีจาก Turbo C
3. โปรแกรม CodeBlock
4. ให้ผู้อ่านฝึกการโปรแกรมที่ผู้อ่านได้เลือกทำการติดตั้งในเครื่องคอมพิวเตอร์
5. ทดสอบการเขียนโค๊ด
6. ทดสอบการสั่งให้โปรแกรมทำการแปลคำสั่ง (Compiler)
7. ทดลองทำการสั่งให้โปรแกรมเริ่มทำงาน (Run)
8. ให้ผู้อ่านทดสอบหาจุดผิดพลาดของโปรแกรมและทำการแก้ไขจุดผิดพลาดโดยยึดทฤษฎีมาเป็นหลักในการแก้ปัญหา
9. ให้ผู้อ่านได้ทดลองทำการป้อนตามที่เนื้อหาได้กำหนด ทั้งส่วนทฤษฎีและการทดลอง ให้ผลการทำงานโปรแกรม ตรงตามที่เนื้อหากำหนด
10. ให้ผู้อ่านได้ทำการเขียนผลการทดลองตามกำหนด
