report.md

實驗九 實驗結報

0316213 蒲郁文 & 0316323 薛世恩

實驗名稱

LCD 及 DS18B20

實驗目的

- 瞭解 LCD 的使用。
- 瞭解 DS18B20 的使用。

實驗步驟

跑馬燈

```
#include "libtmd.h"

const GPIO_TypeDef *LCD_DATA_PORT[ 8] = {
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB
};
const GPIO_TypeDef *LCD_RS_PORT = GPIOA;
const GPIO_TypeDef *LCD_RW_PORT = GPIOA;
const GPIO_TypeDef *LCD_EN_PORT = GPIOA;

const uint16_t LCD_DATA_PIN[ 8] = {
    GPIO_PIN_0,
    GPIO_PIN_1,
    GPIO_PIN_2,
    GPIO_PIN_3,
    GPIO_PIN_4,
    GPIO_PIN_5,
    GPIO_PIN_6,
    GPIO_PIN_7
};
const uint16_t LCD_RS_PIN = GPIO_PIN_5;
const uint16_t LCD_RW_PIN = GPIO_PIN_6;
const uint16_t LCD_EN_PIN = GPIO_PIN_7;

void SysTick_UserConfig (float);
void SysTick_Handler ();
void init();
void init_lcd();
void write_to_lcd(int, int);

int counter = 0;

int main() {
    fpu_enable();
    init();
    SysTick_UserConfig (0.3);
    while (1);
```

```
    return 0;
}

void SysTick_UserConfig(float n) {
    SysTick->CTRL |= 0x00000004;
    SysTick->LOAD = ( uint32_t) (n * 4000000.0);
    SysTick->VAL =  0;
    SysTick->CTRL |= 0x00000003;
}

void SysTick_Handler() {
    counter = counter +  1;
    if (counter == 18) {
        write_to_lcd(0x80 + 0x0F, 1);
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x80 + 0x41, 1);
    }
    if (counter == 34) {
        write_to_lcd(0x80 + 0x4F, 1);
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x80 + 0x1, 1);
        counter =  2;
    }
    write_to_lcd(0x10, 1); // shift cursor
    write_to_lcd(0x10, 1); // shift cursor
    write_to_lcd(0x20, 0); // print ' '
    write_to_lcd(0x34, 0); // print '4'
    write_to_lcd(0x35, 0); // print '5'
    if (counter == 17) {
        write_to_lcd(0x80 + 0x40, 1);
        write_to_lcd(0x35, 0); // print '5'
        write_to_lcd(0x80 + 0x0F, 1);
    }
    if (counter == 33) {
        write_to_lcd(0x80 + 0x0, 1);
        write_to_lcd(0x35, 0); // print '5'
        write_to_lcd(0x80 + 0x4F, 1);
    }
}

void init() {
    TMD_GPIO_Init();
    init_lcd();
}

void init_lcd() {
    write_to_lcd(0x38, 1); // function setting
    write_to_lcd(0x06, 1); // entry mode
    write_to_lcd(0x0C, 1); // display on
    write_to_lcd(0x01, 1); // clear screen
    write_to_lcd(0x80, 1); // move to top left
}

void write_to_lcd(int input, int is_cmd) {
    if (is_cmd)
        TMD_GPIO_SetPinLow(LCD_RS_PORT, LCD_RS_PIN);
    else
        TMD_GPIO_SetPinHigh(LCD_RS_PORT, LCD_RS_PIN);

    TMD_GPIO_SetPinLow(LCD_RW_PORT, LCD_RW_PIN);

    for (int i = 0; i < 8; ++i) {
        if (input & (1 << i))
            TMD_GPIO_SetPinHigh(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
        else
            TMD_GPIO_SetPinLow(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
    }

    TMD_GPIO_SetPinHigh(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
    TMD_GPIO_SetPinLow(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
}
```

客製化圖形顯示與按鈕切換

```
#include "libtmd.h"

const GPIO_TypeDef *LCD_DATA_PORT[ 8] = {
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB
};
const GPIO_TypeDef *LCD_RS_PORT = GPIOA;
const GPIO_TypeDef *LCD_RW_PORT = GPIOA;
const GPIO_TypeDef *LCD_EN_PORT = GPIOA;

const uint16_t LCD_DATA_PIN[ 8] = {
    GPIO_PIN_0,
    GPIO_PIN_1,
    GPIO_PIN_2,
    GPIO_PIN_3,
    GPIO_PIN_4,
    GPIO_PIN_5,
    GPIO_PIN_6,
    GPIO_PIN_7
};
const uint16_t LCD_RS_PIN = GPIO_PIN_5;
const uint16_t LCD_RW_PIN = GPIO_PIN_6;
const uint16_t LCD_EN_PIN = GPIO_PIN_7;

const int map_four[ 8] = {
    0x11,
    0x11,
    0x11,
    0x1F,
    0x01,
    0x01,
    0x01,
    0x00
};

const int map_five[ 8] = {
    0x1F,
    0x10,
    0x10,
    0x1F,
    0x01,
    0x01,
    0x1F,
    0x00
};

const char *test_string = "Test: E=m*c^2 ";

void SysTick_UserConfig (float);
void SysTick_Handler ();
void init();
void init_lcd();
void write_to_lcd (int, int);
void create_font (int, const int *);
void write_str_to_lcd (char *);

int counter = 0, mode = 0, position = 0;

int main() {
    int prev_btn = 1, curr_btn = 1;
    fpu_enable();
    init();
    SysTick_UserConfig (0.3);
    while (1) {
        if (!prev_btn && curr_btn) {
            mode ^= 1;
```

```
            position = 0;
            counter = 0;
            SysTick->CTRL &= 0xFFFFFFFE;
            init();
            SysTick->CTRL |= 0x00000001;
        }
        prev_btn = curr_btn;
        curr_btn = GPIOC->IDR & GPIO_PIN_13;
    }
    return 0;
}

void SysTick_UserConfig(float n) {
    SysTick->CTRL |= 0x00000004;
    SysTick->LOAD = (uint32_t)(n * 4000000.0);
    SysTick->VAL = 0;
    SysTick->CTRL |= 0x00000003;
}

void SysTick_Handler() {
    if (mode == 0) {
        counter = counter + 1;
        if (counter == 18) {
            write_to_lcd(0x80 + 0x0F, 1);
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x80 + 0x41, 1);
        }
        if (counter == 34) {
            write_to_lcd(0x80 + 0x4F, 1);
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x80 + 0x1, 1);
            counter = 2;
        }
        write_to_lcd(0x10, 1); // shift cursor
        write_to_lcd(0x10, 1); // shift cursor
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x00, 0); // print '4'
        write_to_lcd(0x01, 0); // print '5'
        if (counter == 17) {
            write_to_lcd(0x80 + 0x40, 1);
            write_to_lcd(0x01, 0); // print '5'
            write_to_lcd(0x80 + 0x0F, 1);
        }
        if (counter == 33) {
            write_to_lcd(0x80 + 0x0, 1);
            write_to_lcd(0x01, 0); // print '5'
            write_to_lcd(0x80 + 0x4F, 1);
        }
    }
    else
        write_str_to_lcd(test_string);
}

void init() {
    TMD_GPIO_Init();
    init_lcd();
    create_font(0, map_four);
    create_font(8, map_five);
    write_to_lcd(0x80, 1); // move to top left
}

void init_lcd() {
    write_to_lcd(0x38, 1); // function setting
    write_to_lcd(0x06, 1); // entry mode
    write_to_lcd(0x0C, 1); // display on
    write_to_lcd(0x01, 1); // clear screen
    write_to_lcd(0x80, 1); // move to top left
}

void write_to_lcd(int input, int is_cmd) {
    if (is_cmd)
        TMD_GPIO_SetPinLow(LCD_RS_PORT, LCD_RS_PIN);
    else
```

```
        TMD_GPIO_SetPinHigh(LCD_RS_PORT, LCD_RS_PIN);

    TMD_GPIO_SetPinLow(LCD_RW_PORT, LCD_RW_PIN);

    for (int i = 0; i < 8; ++i) {
        if (input & (1 << i))
            TMD_GPIO_SetPinHigh(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
        else
            TMD_GPIO_SetPinLow(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
    }

    TMD_GPIO_SetPinHigh(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
    TMD_GPIO_SetPinLow(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
}

void create_font(int location, const int *font_array) {
    write_to_lcd(location & 0x3F | 0x40,  1);
    for (int i = 0; i < 8; ++i)
        write_to_lcd(font_array[i] & 0x1F,  0);
}

void write_str_to_lcd(char *str) {
    if (str[position] ==  0) {
        position =  0;
        counter =  0;
        SysTick->CTRL &= 0xFFFFFFFE;
        init();
        SysTick->CTRL |= 0x00000001;
    }
    write_to_lcd(str[position],  0);
    position++;
}
```

跑馬燈與溫度計

```
#include "libtmd.h"
#include "ds18b20.h"

const GPIO_TypeDef *LCD_DATA_PORT[ 8] = {
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB,
    GPIOB
};
const GPIO_TypeDef *LCD_RS_PORT = GPIOA;
const GPIO_TypeDef *LCD_RW_PORT = GPIOA;
const GPIO_TypeDef *LCD_EN_PORT = GPIOA;

const uint16_t LCD_DATA_PIN[ 8] = {
    GPIO_PIN_0,
    GPIO_PIN_1,
    GPIO_PIN_2,
    GPIO_PIN_3,
    GPIO_PIN_4,
    GPIO_PIN_5,
    GPIO_PIN_6,
    GPIO_PIN_7
};
const uint16_t LCD_RS_PIN = GPIO_PIN_5;
const uint16_t LCD_RW_PIN = GPIO_PIN_6;
const uint16_t LCD_EN_PIN = GPIO_PIN_7;

const int map_four[ 8] = {
    0x11,
    0x11,
    0x11,
    0x1F,
```

```
    0x01,
    0x01,
    0x01,
    0x00
};

const int map_five[8] = {
    0x1F,
    0x10,
    0x10,
    0x1F,
    0x01,
    0x01,
    0x1F,
    0x00
};

const char *test_string = "Test: E=m*c^2";
const unsigned resolution = 11;

void SysTick_UserConfig(float);
void SysTick_Handler();
void init();
void init_lcd();
void write_to_lcd(int, int);
void create_font(int, const int *);
void write_str_to_lcd(char *);
void write_int_to_lcd(int16_t);

int counter = 0, mode = 0, position = 0;

int main() {
    int prev_btn = 1, curr_btn = 1;
    fpu_enable();
    init();
    set_resolution(resolution);
    SysTick_UserConfig(0.3);
    while (1) {
        if (!prev_btn && curr_btn) {
            mode ^= 1;
            position = 0;
            counter = 0;
            SysTick->CTRL &= 0xFFFFFFF8;
            init();
            if (mode == 0)
                SysTick_UserConfig(0.3);
            else
                SysTick_UserConfig(1);
        }
        prev_btn = curr_btn;
        curr_btn = GPIOC->IDR & GPIO_PIN_13;
    }
    return 0;
}

void SysTick_UserConfig(float n) {
    SysTick->CTRL |= 0x00000004;
    SysTick->LOAD = (uint32_t)(n * 4000000.0);
    SysTick->VAL = 0;
    SysTick->CTRL |= 0x00000003;
}

void SysTick_Handler() {
    if (mode == 0) {
        counter = counter + 1;
        if (counter == 18) {
            write_to_lcd(0x80 + 0x0F, 1);
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x80 + 0x41, 1);
        }
        if (counter == 34) {
            write_to_lcd(0x80 + 0x4F, 1);
            write_to_lcd(0x20, 0); // print ' '
            write_to_lcd(0x20, 0); // print ' '
```

```
            write_to_lcd(0x80 + 0x1, 1);
            counter = 2;
        }
        write_to_lcd(0x10, 1); // shift cursor
        write_to_lcd(0x10, 1); // shift cursor
        write_to_lcd(0x20, 0); // print ' '
        write_to_lcd(0x00, 0); // print '4'
        write_to_lcd(0x01, 0); // print '5'
        if (counter == 17) {
            write_to_lcd(0x80 + 0x40, 1);
            write_to_lcd(0x01, 0); // print '5'
            write_to_lcd(0x80 + 0x0F, 1);
        }
        if (counter == 33) {
            write_to_lcd(0x80 + 0x0, 1);
            write_to_lcd(0x01, 0); // print '5'
            write_to_lcd(0x80 + 0x4F, 1);
        }
    }
    else {
        SysTick->CTRL &= 0xFFFFFFFE;
        write_int_to_lcd(get_temperature());
        SysTick->CTRL |= 0x00000001;
    }
}

void init() {
    TMD_GPIO_Init();
    init_lcd();
    create_font(0, map_four);
    create_font(8, map_five);
    write_to_lcd(0x80, 1); // move to top left
}

void init_lcd() {
    write_to_lcd(0x38, 1); // function setting
    write_to_lcd(0x06, 1); // entry mode
    write_to_lcd(0x0C, 1); // display on
    write_to_lcd(0x01, 1); // clear screen
    write_to_lcd(0x80, 1); // move to top left
}

void write_to_lcd(int input, int is_cmd) {
    if (is_cmd)
        TMD_GPIO_SetPinLow(LCD_RS_PORT, LCD_RS_PIN);
    else
        TMD_GPIO_SetPinHigh(LCD_RS_PORT, LCD_RS_PIN);

    TMD_GPIO_SetPinLow(LCD_RW_PORT, LCD_RW_PIN);

    for (int i = 0; i < 8; ++i) {
        if (input & (1 << i))
            TMD_GPIO_SetPinHigh(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
        else
            TMD_GPIO_SetPinLow(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
    }

    TMD_GPIO_SetPinHigh(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
    TMD_GPIO_SetPinLow(LCD_EN_PORT, LCD_EN_PIN);
    delay_ms(10);
}

void create_font(int location, const int *font_array) {
    write_to_lcd(location & 0x3F | 0x40, 1);
    for (int i = 0; i < 8; ++i)
        write_to_lcd(font_array[i] & 0x1F, 0);
}

void write_str_to_lcd(char *str) {
    if (str[position] == 0) {
        position = 0;
        counter = 0;
        SysTick->CTRL &= 0xFFFFFFFE;
        init();
```

```
        SysTick->CTRL |= 0x00000001;
    }
    write_to_lcd(str[position], 0);
    position++;
}

void write_int_to_lcd(int16_t in) {
    switch (resolution) {
        case 12:
            in &= 0xFFFF;
            break;
        case 11:
            in &= 0xFFFE;
            break;
        case 10:
            in &= 0xFFFC;
            break;
        case 9:
            in &= 0xFFF8;
            break;
        default:
            break;
    }
    int16_t in1 = in >> 4;
    int16_t in2 = ((in & 0x0001) * 0.0625 + (in & 0x0002) * 0.125 + \
                   (in & 0x0004) * 0.25 + (in & 0x0008) * 0.5) * 1000;
    init();
    write_to_lcd(0x30, 0);
    write_to_lcd(0x30, 0);
    write_to_lcd(0x2E, 0);
    write_to_lcd(0x30, 0);
    write_to_lcd(0x30, 0);
    write_to_lcd(0x30, 0);
    write_to_lcd(0x30, 0);
    write_to_lcd(0x10, 1);
    write_to_lcd(0x30 + in2 % 10, 0);
    in2 /= 10;
    write_to_lcd(0x10, 1);
    write_to_lcd(0x30 + in2 % 10, 0);
    in2 /= 10;
    write_to_lcd(0x10, 1);
    write_to_lcd(0x30 + in2 % 10, 0);
    in2 /= 10;
    write_to_lcd(0x10, 1);
    write_to_lcd(0x30 + in2 % 10, 0);
    write_to_lcd(0x10, 1);
    write_to_lcd(0x10, 1);
    write_to_lcd(0x30 + in1 % 10, 0);
    in1 /= 10;
    write_to_lcd(0x10, 1);
    write_to_lcd(0x30 + in1 % 10, 0);
}
```

(libtmd.h)

```
#ifndef LIBTMD_H
#define LIBTMD_H

extern void delay_ms(unsigned);
extern void fpu_enable();

#include "stm32l476xx.h"

#ifndef GPIO_PIN_0
#define GPIO_PIN_0    ((uint16_t) 0x0001)
#define GPIO_PIN_1    ((uint16_t) 0x0002)
#define GPIO_PIN_2    ((uint16_t) 0x0004)
#define GPIO_PIN_3    ((uint16_t) 0x0008)
#define GPIO_PIN_4    ((uint16_t) 0x0010)
#define GPIO_PIN_5    ((uint16_t) 0x0020)
#define GPIO_PIN_6    ((uint16_t) 0x0040)
#define GPIO_PIN_7    ((uint16_t) 0x0080)
#define GPIO_PIN_8    ((uint16_t) 0x0100)
```

```
#define GPIO_PIN_9   ((uint16_t) 0x0200)
#define GPIO_PIN_10  ((uint16_t) 0x0400)
#define GPIO_PIN_11  ((uint16_t) 0x0800)
#define GPIO_PIN_12  ((uint16_t) 0x1000)
#define GPIO_PIN_13  ((uint16_t) 0x2000)
#define GPIO_PIN_14  ((uint16_t) 0x4000)
#define GPIO_PIN_15  ((uint16_t) 0x8000)
#define GPIO_PIN_ALL ((uint16_t) 0xFFFF)
#endif

void TMD_GPIO_Init() {
    RCC->AHB2ENR   |=  0b00000000000000000000000000000111 ;

    GPIOA->MODER   &=  0b11111111111111110000001111111111 ;
    GPIOA->MODER   |=  0b00000000000000000101010000000000 ;
    GPIOA->PUPDR   &=  0b11111111111111110000001111111111 ;
    GPIOA->PUPDR   |=  0b00000000000000000101010000000000 ;
    GPIOA->OSPEEDR &=  0b11111111111111110000001111111111 ;
    GPIOA->OSPEEDR |=  0b00000000000000000101010000000000 ;
    GPIOA->OTYPER  &=  0b11111111111111111111111100011111 ;

    GPIOB->MODER   &=  0b11111111111111110000000000000000 ;
    GPIOB->MODER   |=  0b00000000000000000101010101010101 ;
    GPIOB->PUPDR   &=  0b11111111111111110000000000000000 ;
    GPIOB->PUPDR   |=  0b00000000000000000101010101010101 ;
    GPIOB->OSPEEDR &=  0b11111111111111110000000000000000 ;
    GPIOB->OSPEEDR |=  0b00000000000000000101010101010101 ;
    GPIOB->OTYPER  &=  0b11111111111111111111111100000000 ;

    GPIOC->MODER   &=  0b11110011111111111111111111111111 ;
}

void TMD_GPIO_SetPinLow(GPIO_TypeDef *GPIOX,  uint16_t GPIO_PIN_Y) {
    GPIOX->BRR = GPIO_PIN_Y;
}

void TMD_GPIO_SetPinHigh(GPIO_TypeDef *GPIOX,  uint16_t GPIO_PIN_Y) {
    GPIOX->BSRR = GPIO_PIN_Y;
}

void usleep(unsigned delay) {
    RCC->APB1ENR1 |=  0b1;
    if (delay == 0)
        TIM2->ARR =  2;
    else
        TIM2->ARR = delay;
    TIM2->PSC = (uint32_t) 3;
    TIM2->EGR = TIM_EGR_UG;
    TIM2->CR1 |= TIM_CR1_CEN;
    int pre_val =  0;
    while (1) {
        int now_val = TIM2->CNT;
        if (pre_val > now_val) {
            TIM2->CR1 &= ~TIM_CR1_CEN;
            return;
        }
        pre_val = now_val;
    }
}

#endif
```

(libtmd.s)

```
    .syntax unified
    .cpu cortex-m4
    .thumb

.text
    .global delay_ms
    .global fpu_enable

delay_ms:
```

```
    push {r0, r1, lr}
    ldr  r1, =4000
    muls r0, r1

delay_ms_loop:
    beq  delay_ms_end
    subs r0, 4
    b    delay_ms_loop

delay_ms_end:
    pop  {r0, r1, pc}

fpu_enable:
    push  {r0, r1, lr}
    ldr.w r0, =0xE000ED88
    ldr   r1, [r0]
    orr   r1, r1, #(0xF << 20)
    str   r1, [r0]
    dsb
    isb
    pop   {r0, r1, pc}
```

(onewire.h)

```
#include "libtmd.h"

void OneWire_Reset()
{
    ONEWIRE_INPUT();
    GPIOA->BRR = GPIO_PIN_8;  // high -> low
    ONEWIRE_OUTPUT();
    ONEWIRE_DELAY(480);
    ONEWIRE_INPUT();
    ONEWIRE_DELAY(70);
    ONEWIRE_DELAY(410);
}

void OneWire_WriteBit(uint8_t bit)
{
    ONEWIRE_DELAY(4);
    ONEWIRE_INPUT();
    if (bit) // 1
    {
        // Set line low
        GPIOA->BRR = GPIO_PIN_8;
        ONEWIRE_OUTPUT();
        // Bit high
        ONEWIRE_INPUT();
    }
    else // 0
    {
        // Set line low
        GPIOA->BRR = GPIO_PIN_8;
        ONEWIRE_OUTPUT();
        ONEWIRE_DELAY(70);
    }
    ONEWIRE_INPUT();
}

void OneWire_WriteByte(int data)
{
    int mask = 0x80;
    for (int i = 0; i < 8; i++)
    {
        OneWire_WriteBit(mask & data);
        mask = mask >>  1;
    }
}

uint8_t OneWire_ReadBit()
{
    ONEWIRE_DELAY(4);
    uint8_t data = 0;
```

```
    ONEWIRE_INPUT();
    GPIOA->BRR = GPIO_PIN_8;  // high -> low
    ONEWIRE_OUTPUT();
    ONEWIRE_DELAY(1);
    ONEWIRE_INPUT();
    data = GPIOA->IDR & 0x1;
    return data;
}

int OneWire_ReadByte()
{
    int mask = 1, ans = 0;
    for (int i = 0; i < 8; i++)
    {
        ans = ans | (mask &  OneWire_ReadBit());
        mask = mask <<  1;
    }
}

void ONEWIRE_INPUT()
{
    GPIOA->MODER   &=  0b11111111111111100111111111111111 ;
    GPIOA->PUPDR   &=  0b11111111111111100111111111111111 ;
    GPIOA->PUPDR   |=  0b00000000000000001000000000000000 ;
    GPIOA->OSPEEDR &=  0b11111111111111100111111111111111 ;
    GPIOA->OSPEEDR |=  0b00000000000000001000000000000000 ;
    GPIOA->OTYPER  |=  0b00000000000000000000000100000000 ;
}

void ONEWIRE_OUTPUT()
{
    GPIOA->MODER   &=  0b11111111111111100111111111111111 ;
    GPIOA->MODER   |=  0b00000000000000001000000000000000 ;
    GPIOA->PUPDR   &=  0b11111111111111100111111111111111 ;
    GPIOA->PUPDR   |=  0b00000000000000001000000000000000 ;
    GPIOA->OSPEEDR &=  0b11111111111111100111111111111111 ;
    GPIOA->OSPEEDR |=  0b00000000000000001000000000000000 ;
    GPIOA->OTYPER  |=  0b00000000000000000000000100000000 ;
}

void ONEWIRE_DELAY(unsigned microseconds)
{
    usleep(microseconds);
}
```

(ds18b20.h)

```
#include "libtmd.h"
#include "onewire.h"

void set_resolution(unsigned resolution) {
    // Initialization
    OneWire_Reset();
    // ROM Command: Skip ROM [CCh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    // DS18B20 Function Command: Write Scratchpad [4Eh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    // Data Exchange: TH Register [40h]
    OneWire_WriteBit(0);
```

```
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    // Data Exchange: TL Register [08h]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    // Data Exchange: Configuration Register
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    switch (resolution) {
        case 9:
            OneWire_WriteBit(0);
            OneWire_WriteBit(0);
            break;
        case 10:
            OneWire_WriteBit(1);
            OneWire_WriteBit(0);
            break;
        case 11:
            OneWire_WriteBit(0);
            OneWire_WriteBit(1);
            break;
        case 12:
        default:
            OneWire_WriteBit(1);
            OneWire_WriteBit(1);
            break;
    }
    OneWire_WriteBit(0);

    // Initialization
    OneWire_Reset();
    // ROM Command: Skip ROM [CCh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    // DS18B20 Function Command: Copy Scratchpad [48h]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
}

int16_t get_temperature() {
    // Initialization
    OneWire_Reset();
    // ROM Command: Skip ROM [CCh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
```

```
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    // DS18B20 Function Command: Convert T [44h]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);

    // Fucking Wait
    usleep(750000);

    // Initialization
    OneWire_Reset();
    // ROM Command: Skip ROM [CCh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    // DS18B20 Function Command: Read Scratchpad [BEh]
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(1);
    OneWire_WriteBit(0);
    OneWire_WriteBit(1);
    // Data Exchange: Temperature LSB Register
    int16_t r = 0;
    r |= OneWire_ReadBit() << 0;
    r |= OneWire_ReadBit() << 1;
    r |= OneWire_ReadBit() << 2;
    r |= OneWire_ReadBit() << 3;
    r |= OneWire_ReadBit() << 4;
    r |= OneWire_ReadBit() << 5;
    r |= OneWire_ReadBit() << 6;
    r |= OneWire_ReadBit() << 7;
    // Data Exchange: Temperature MSB Register
    r |= OneWire_ReadBit() << 8;
    r |= OneWire_ReadBit() << 9;
    r |= OneWire_ReadBit() << 10;
    r |= OneWire_ReadBit() << 11;
    r |= OneWire_ReadBit() << 12;
    r |= OneWire_ReadBit() << 13;
    r |= OneWire_ReadBit() << 14;
    r |= OneWire_ReadBit() << 15;

    // Initialization
    OneWire_Reset();

    return r;
}
```

實驗結果與問題回答

跑馬燈

- LCD 初始化的部份，應設定雙排顯示模式、每次寫資料計數器遞增而畫面不動、隱藏游標、清除畫面、將 DD RAM 位址設為左上角第一個字元位置等。
- 欲下指令給 LCD，首先應將 RS 設為 0，RW 設為 0，接著將指令的內容寫入 D[7:0]，最後將 EN 設為 1，等待 10 毫秒，再設為 0，等待 10 毫秒。
- 欲顯示字元至 LCD 上，首先應將 RS 設為 1，RW 設為 0，接著將字元的編碼寫入 D[7:0]，最後將 EN 設為 1，等待 10 毫秒，再設為 0，等待 10 毫秒。

- 在開始顯示文字前必需先設定 DD RAM 位址。

客製化圖形顯示與按鈕切換

- 欲設定字型，首先應將 RS 設為 1，RW 設為 0，接著將點陣字的第 i 排寫入 D[7:0]，最後將 EN 設為 1，等待 10 毫秒，再設為 0，等待 10 毫秒。
- i 從 0 開始，每次執行完後 i 遞增，重覆以上步驟 8 次，即設定好一個字元。
- 在開始設定字型前必需先設定 CG RAM 位址。
- 顯示字串的部份，每次往後顯示一個字元，直到遇到字串結尾的 0，即清除畫面重來一次。

跑馬燈與溫度計

- one wire 照著他的 protocol 去實作，將他設為 low 並且等待適當的時間，並適時切換 input 與 output 模式，而因為是 pull-up 所以將他放開時，電壓會上升，寫入或是讀取 byte 時，每一個 bit 中間都要有適當的 delay，讀取時要在 15us 內將他讀取出來。
- 使用 one wire protocol 操作溫度計，每次操作都需包含初始化、ROM Command (Skip ROM)、DS18B20 Function Command 等動作，最後視 command 需求做若干 byte 的 Data Exchange。

心得討論與應用聯想

- LCD 因為使用 11 條線操作，所以不會太難，溫度計只有 1 條線，操作上就變得相當複雜。
- 溫度計難以 debug。