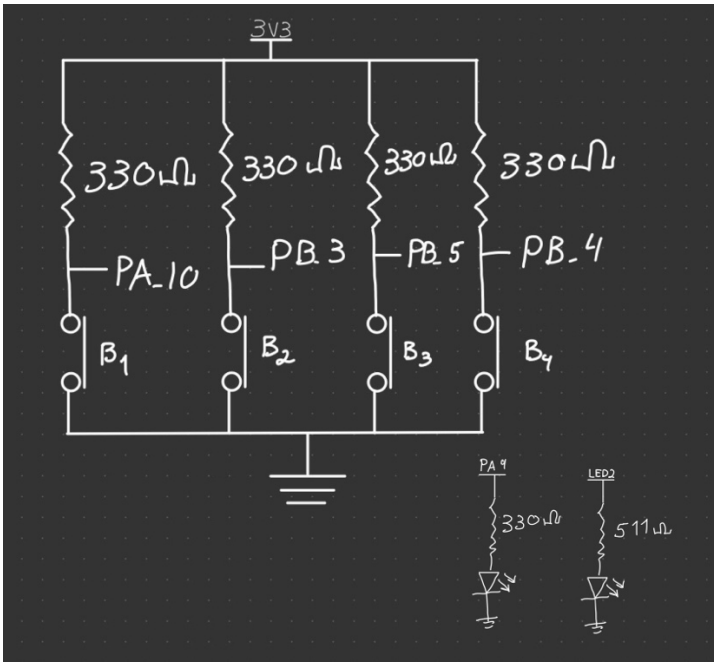
Michael Starks & Sovann Chak

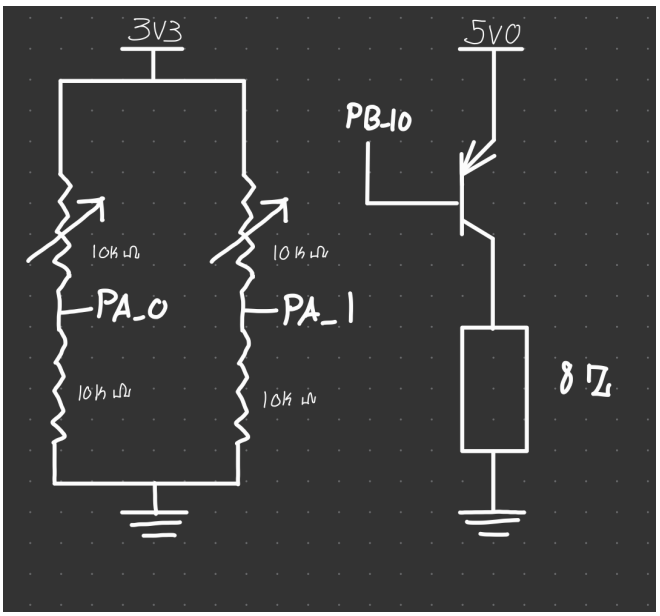
Project 1 Module 2

ECEN 5803-002

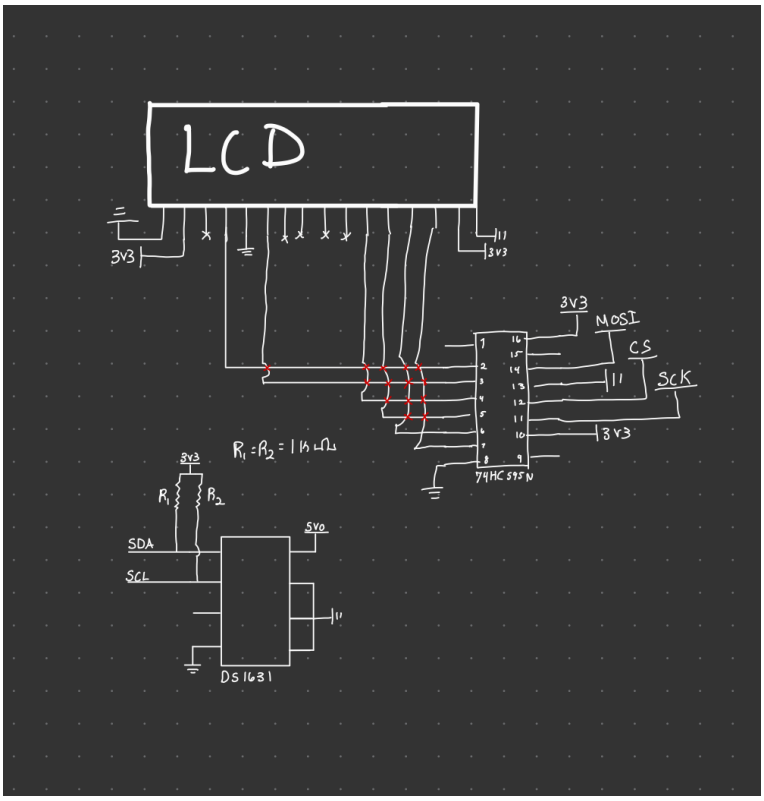
1. a. Wiring diagram for exercise 2_8.1 and exercise 2_8.2



c. Wiring diagram for exercise 2_9



d. Wiring diagram for exercise 2_11



2. a. *Exercise 2_8: Try to issue an interrupt on different signal edges (rising edge or falling edge). What changes?*

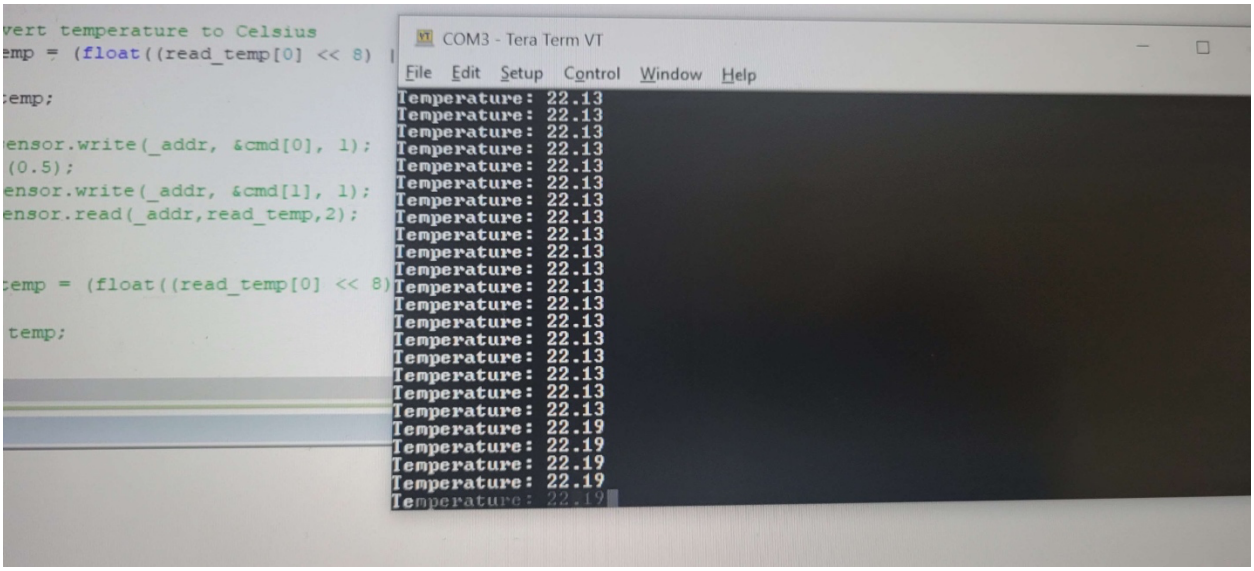
When the interrupt is triggered on a falling edge, the LED state is changed when the button is released instead of when the button is pressed.

b. *Exercise 2_9: What changes when you adjust the amount by which variable i is incremented/decremented?*

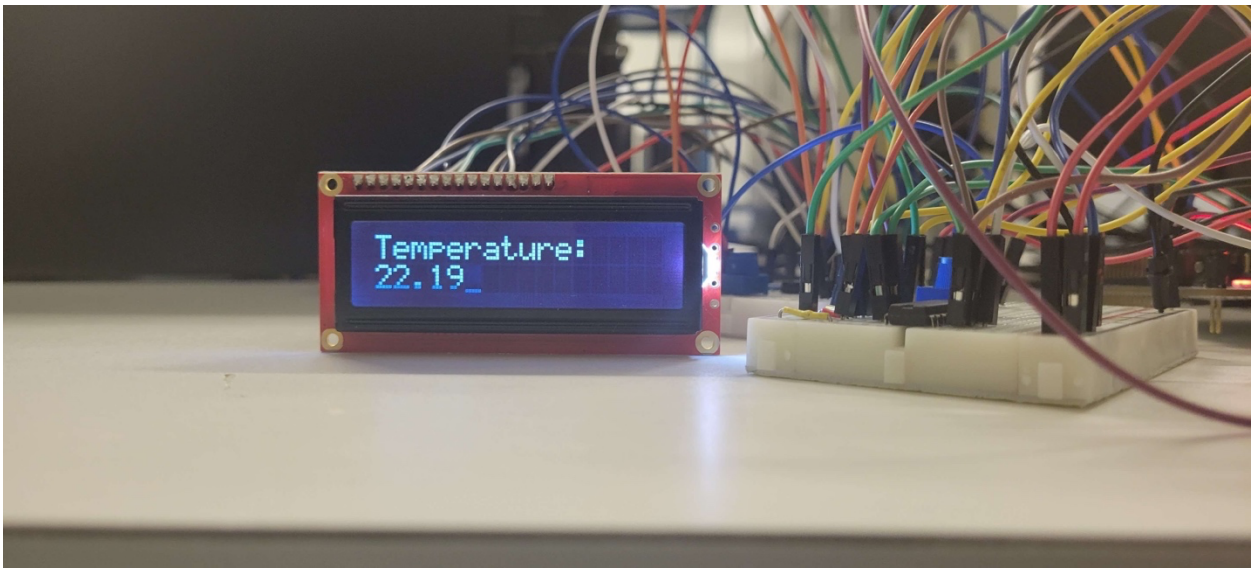
When i is changed, the volume increases at a faster rate.

c. *Exercise 2_11: What temperature is displayed on your PC terminal window? What is displayed on the LCD?*

PC Terminal Window:



LCD Display:



Appendix.

2_8.1 Source Code:

```
/*----------------------------------------------------------------------------
LAB EXERCISE 8.1 - DIGITAL INPUTS AND OUTPUTS
PROGRAMMING USING MBED API
 ----------------------------------------
 In this exercise you need to use the mbed API functions to:

        1) Define BusIn, BusOut interfaces for inputs and outputs
        2) The LED is controlled by the button:
                        + USER_BUTTON - LED1

        GOOD LUCK!
 *----------------------------------------------------------------------------*/

#include "mbed.h"

//Define input bus
        //Write your code here
        BusIn buttons(D2,D3,D4,D5); //1 - PA_10/D2, 2 - PB_3/D3, 4 - PB_5/D4, 8 - PB_4/D5

//Define output bus for the  LED
        //Write your code here
        BusOut leds(LED2,D8); //1 - nucleo led , 2 - breadboard led

/*----------------------------------------------------------------------------
 MAIN function
 *----------------------------------------------------------------------------*/

int main(){

        while(1){

                //Check which button was pressed and light up the corresponding LED
                //Write your code here
                // This will control the led on the nucleo board
                if ((buttons & 0x2) && not (buttons & 0x1)) {
                        leds = leds | 0x1;
                } else if ((buttons & 0x1) && not (buttons & 0x2)){
                        leds = leds & 0x2;
                }

    if ((buttons & 0x8) && not (buttons & 0x4)) {
                        leds = leds | 0x2;
                } else if ((buttons & 0x4) && not (buttons & 0x8)){
                        leds = leds & 0x1;
                }
                __wfi(); // sleep
        }
}

// ********************************ARM University Program Copyright (c) ARM Ltd
2014*************************************
```

2_8.2 Source Code:

```
/*----------------------------------------------------------------------------
LAB EXERCISE 8.2 - INTERRUPT IN/OUT
PROGRAMMING USING MBED API
 ----------------------------------------
 In this exercise you need to use the mbed API functions to:

        1) Define InterruptIn and ISR for the button press
        2) In the interrupt service routine, the LED is used to indicate when a
        button was pressed:
                        + USER_BUTTON - LED1
```

```
        3) Put the processor into sleep mode upon exiting from the ISR

        GOOD LUCK!
 *-------------------------------------------------------------------------------*/

#include "mbed.h"

//Define outputs

        //Write your code here
        BusOut leds(LED2,D8); //1 - nucleo led , 2 - breadboard led


//Define interrupt inputs

        //Write your code here
        InterruptIn button1(D2);
        InterruptIn button2(D3);
        InterruptIn button3(D4);
        InterruptIn button4(D5);


//Define ISRs for the interrupts
void button_1_handler(){

        //Write your code here
        leds = leds | 0x1;

}

void button_2_handler(){

        ////Write your code here
        leds = leds & 0x2;

}

void button_3_handler(){

        ////Write your code here
        leds = leds | 0x2;

}

void button_4_handler(){

        ////Write your code here
        leds = leds & 0x1;

}

/*-------------------------------------------------------------------------------
 MAIN function
 *-------------------------------------------------------------------------------*/

int main(){

        __enable_irq();                  //enable interrupts

        //initially turn off  LED
        //Write your code here
        leds = 0x0;

        //Interrupt handlers
        //Attach the address of the ISR to the rising edge
```

```
                //Write your code here
        button1.rise(&button_1_handler);
        button2.rise(&button_2_handler);
        button3.rise(&button_3_handler);
        button4.rise(&button_4_handler);

        //Sleep on exit
        while(1){

                //Write your code here
                __wfe();

        }
}

// ********************************ARM University Program Copyright (c) ARM Ltd 2014*************************************
```

2_9 Source Code:

```
/*-----------------------------------------------------------------------------
LAB EXERCISE 9 - Analog input and PWM
 ---------------------------------------
        Use two potentiometers to adjust the volume and pitch of the output sound wave.

        Inputs: Virtual potentiometers 1 and 2
        Output: Virtual speaker,  Real PC

        GOOD LUCK!
 *-----------------------------------------------------------------------------*/

#include "mbed.h"
#include "pindef.h"

/*
Define the PWM speaker output
Define analog inputs
Define serial output
*/

        //Write your code here
  PwmOut speaker(D6); // Speaker Output
        AnalogIn pitchControl(A0); // Picth Control
        AnalogIn volumeControl(A1); // Volume Control


//Define variables
float i;

/*-----------------------------------------------------------------------------
 MAIN function
 *-----------------------------------------------------------------------------*/

int main(){
        while(1){
                /*
                Print values of the potentiometers to the PC serial terminal
                Create a saw-tooth sound wave
                Make the period and volume adjustable using the potentiometers
                */
                float freq = (7680.0f * ((pitchControl.read() - 0.49f)/0.51f) + 320.0f); // will need to do some math to map the pitch
                float volume = (volumeControl.read()-0.49f)/0.51f; // maps volume analog input to duty cycle
                volume = volume < 0.01 ? 0 : volume;
```

```
                printf("Frequency: %f\r\n",freq);
                printf("Volume: %0.2f\r\n",volume);

                speaker.period(1.0f/freq);

                for(i=0; i<1; i+=0.1) {
                        speaker = i * volume;
                        // will need to wait here to make the entire loop one period
                        wait_ms(20);
                }


        }
}

// ********************************ARM University Program Copyright © ARM Ltd
2014*************************************
```

2_11 Source Code:

DS1631.cpp

```
/*-----------------------------------------------------------------------------
 MAXIM DS1631 2-wire temperature sensor library
 *-----------------------------------------------------------------------------*/

#include "mbed.h"
#include "DS1631.h"

int ret = 0;
char reset_cmd[1] = {0x54};
char cmd[] = {0xAA};
char read_temp[2];
char stop_reading = 0x22;
char start_convert = 0x51;

DS1631::DS1631(PinName sda, PinName scl, int addr) : temp_sensor(sda, scl), _addr(addr){

}

float DS1631::read(){

        temp_sensor.write(_addr,&start_convert,1,false);

                /*
                Write the Start Convert T command to the sensor
                Write the Read Temperature command to the sensor
                Read the 16-bit temperature data
                */

                //Write your code here

                do {
                        ret = temp_sensor.write(_addr,cmd,1);
                } while (ret != 0);


                ret = temp_sensor.read(_addr,read_temp,2);

                //Convert temperature to Celsius
        float temp = (float((read_temp[0] << 8) | read_temp[1]) / 256);

        return temp;
```

```
//      temp_sensor.write(_addr, &cmd[0], 1);
//      //wait(0.5);
//      temp_sensor.write(_addr, &cmd[1], 1);
//      temp_sensor.read(_addr,read_temp,2);
//
//
//      float temp = (float((read_temp[0] << 8) | read_temp[1]) / 256);
//
//      return temp;


}
// ********************************ARM University Program Copyright (c) ARM Ltd
2014*************************************
```

main.cpp

```
/*-----------------------------------------------------------------------------
LAB EXERCISE 11.4- SPI and I2C interface
SERIAL COMMUNICATION
 ---------------------------------------
 Display the temperature from the virtual DS1631 temperature sensor on the
      virtual LCD

 Input: virtual DS1631 temperature sensor
 Output: virtual LCD display

        GOOD LUCK!
 *-----------------------------------------------------------------------------*/

#include "NHD_0216HZ.h"
#include "DS1631.h"
#include "pindef.h"

//Define the LCD and the temperature sensor

//Write your code here

DS1631 temp_sensor(I2C_SDA,I2C_SCL,0x90);
NHD_0216HZ display(SPI_CS,SPI_MOSI,SPI_SCLK);

//Define a variable to store temperature measurement
float temp;

/*-----------------------------------------------------------------------------
 MAIN function
 *-----------------------------------------------------------------------------*/

int main() {
        //Initialise the LCD

        //Write your code here
        wait(5);
        display.init_lcd();
        display.printf("Hello World!");
        wait_ms(50);



        while(1){
                /*
                Read the temperature from the DS1631
                Update the LCD with new temperature measurement
                */

                //Write your code here
        display.clr_lcd();
        display.set_cursor(0,0);
```

```
        temp = temp_sensor.read();
        display.printf("Temperature:");
        display.set_cursor(0,1);
        display.printf("%0.2f",temp);
        printf("Temperature: %0.2f\r\n", temp);

        }
}

// ********************************ARM University Program Copyright (c) ARM Ltd
2014*********************************
```

NHD_0216HZ.cpp

```
/*-----------------------------------------------------------------------------
 Newhaven NHD0216HZ LCD library
 *-----------------------------------------------------------------------------*/

#include "mbed.h"
#include <stdarg.h>
#include "NHD_0216HZ.h"

NHD_0216HZ::NHD_0216HZ(PinName SPI_CS, PinName SPI_MOSI, PinName SPI_SCLK) : _SPI_CS(SPI_CS),
_SPI_MOSI(SPI_MOSI), _SPI_SCLK(SPI_SCLK){
                _SPI_CS = 0;
                //printf("Display constructor\n\r");
}

void NHD_0216HZ::shift_out(int data) {
    _SPI_CS = 0;
    for(int i=0; i<8; i++){
        _SPI_MOSI = (data & (0x80 >> i));
        _SPI_SCLK = 0;
        wait_us(1);
        _SPI_SCLK = 1;
    }
    _SPI_MOSI = 0;
    _SPI_CS = 1;
}

void NHD_0216HZ::init_lcd(void) {
//              //printf("Running\n\r");
    wait_ms(50);
    write_cmd(0x30);    //function set 8-bit
    wait_us(37);
    write_cmd(0x20);    //function set
    wait_us(37);
    write_cmd(0x20);    //function set
    wait_us(37);
    write_cmd(0x0C);    //display ON/OFF
    wait_us(37);
    write_cmd(0x01);    //display clear
    wait_us(1520);
    write_cmd(0x06);    //entry-mode set
    wait_us(37);
                write_cmd(0x28);
                wait_us(37);
    set_cursor(0,0);

}

void NHD_0216HZ::write_4bit(int data, int mode) {
    int hi_n;
    int lo_n;

    hi_n = hi_n = (data & 0xF0);
```

```
    lo_n = ((data << 4) &0xF0);

    shift_out(hi_n | ENABLE | mode);
    wait_us(1);
    shift_out(hi_n & ~ENABLE);
    shift_out(lo_n | ENABLE | mode);
    wait_us(1);
    shift_out(lo_n & ~ENABLE);
}

void NHD_0216HZ::write_cmd(int cmd) {
    write_4bit(cmd, COMMAND_MODE);
}

void NHD_0216HZ::write_data(char c) {
    write_4bit(c, DATA_MODE);
}

void NHD_0216HZ::printf(const char *format, ...) {
                char arr[16];
                sprintf(arr, "Int: %d, float %f", 4, 6.4);
    va_list v;
    char buffer[16];
    va_start(v, format);
    vsprintf(buffer, format, v);
    char *b = buffer;
    for(int i=0; i<16 && *b; i++) {
        write_data(*b++);
    }
}

void NHD_0216HZ::set_cursor(int column, int row) {
    int addr;

    addr = (row * LINE_LENGTH) + column;
    addr |= TOT_LENGTH;
    write_cmd(addr);
}

void NHD_0216HZ::clr_lcd(void) {
    write_cmd(0x01);    //display clear
    wait_us(1520);
}

// ******************************ARM University Program Copyright (c) ARM Ltd
2014*************************************
```