Sheffield Hallam University

Department of Engineering

BEng (Hons) Computer Systems Engineering

BEng (Hons) Electrical and Electronic Engineering

Activity ID		Activity Title			Laboratory Room No.	Level
Lab 102		Advanced I/O			4302	6
Term	**Duration [hrs]**	**Group Size**	**Max Total Students**	**Date of approval/review**	**Lead Academic**	
1	4	2	25	09-18	Alex Shenfield	

Equipment (per student/group)

Number	Item
1	STM32F7 discovery board lab kit

Learning Outcomes

	Learning Outcome
2	Demonstrate an understanding of the various tools, technologies and protocols used in the development of embedded systems with network functionality
3	Design, implement and test embedded networked devices

Seeing the world, acquiring data and reacting to external conditions

Introduction

Computing is about more than the PC on your desktop! Embedded devices are everywhere – from wireless telecommunications infrastructure points to electronic point of sale terminals. One definition of an embedded system is:

> "An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints."
>
> (http://en.wikipedia.org/wiki/Embedded_system)

In the laboratory sessions for this module you are going to be introduced to the STM32F7 discovery board – a powerful ARM Cortex M7 based microcontroller platform capable of prototyping advanced embedded systems designs. The STM32F7 discovery board includes advanced functionality such as Ethernet connectivity, UART over the USB connection, an LCD screen, and a micro-SD slot. Appendix B shows the various pins that are broken out from the STM32F7 discovery board (onto the Arduino form factor header), and Appendix C provides a schematic showing how these map to the headers on the SHU base board.

This laboratory session will focus on advanced input and output features with the STM32F7 discovery board. As mentioned in the last lab sheet, any kind of embedded microcontroller is only really useful if it has some ability to see the world – whether as simple as taking input from switches and potentiometers or as complicated as reading and reacting to accelerometer data. To do this we will need to integrate sensors into our system.

As well as taking input from real world sensors, we will look in this lab sheets at the display of data via the LCD screen that forms part of our STM32F7-discovery evaluation kit. The display of data is an important feature in any embedded system.

Bibliography

There are no essential bibliographic resources for this laboratory session aside from this tutorial sheet. However the following websites and tutorials may be of help (especially if you haven't done much electronics previously or your digital logic and/or programming is a bit rusty):

- http://www.cs.indiana.edu/~geobrown/book.pdf[1]
- https://visualgdb.com/tutorials/arm/stm32/
- http://www.keil.com/appnotes/files/apnt_280.pdf
- https://developer.mbed.org/platforms/ST-Discovery-F746NG/

1 Note: this book is for a slightly different board – however, much of the material is relevant to the STM32F7 discovery

Methodology

Check that you have all the necessary equipment (see Figure 1)!

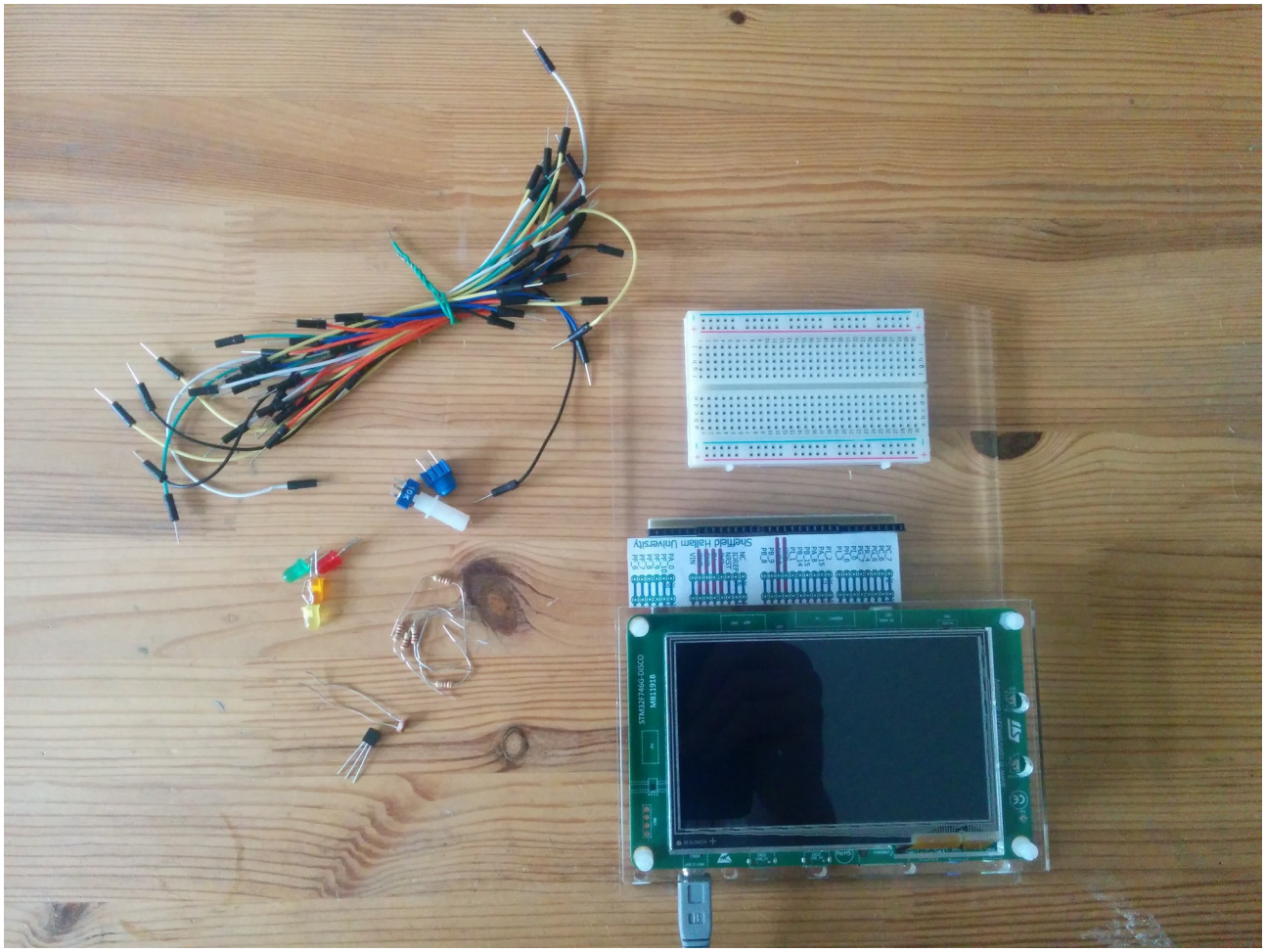


Figure 1 – The necessary equipment for this lab (don't worry if you have LEDs of a different colour)

Task 1

The STM32F7 discovery board lab kits on the benches have a 4.5” touch screen LCD built in to the evaluation board. This touch screen LCD allows us to draw shapes, display data as text, and even to render bitmaps!

Figure 1 shows the LCD display showing a simple “hello world” message and uptime counter (code for this application is provided in code snippet 1). This uptime counter is updated every second.

Run the code provided (the project can be downloaded from GitHub) and verify it works. Make sure you go through the code in detail to understand what it is doing.

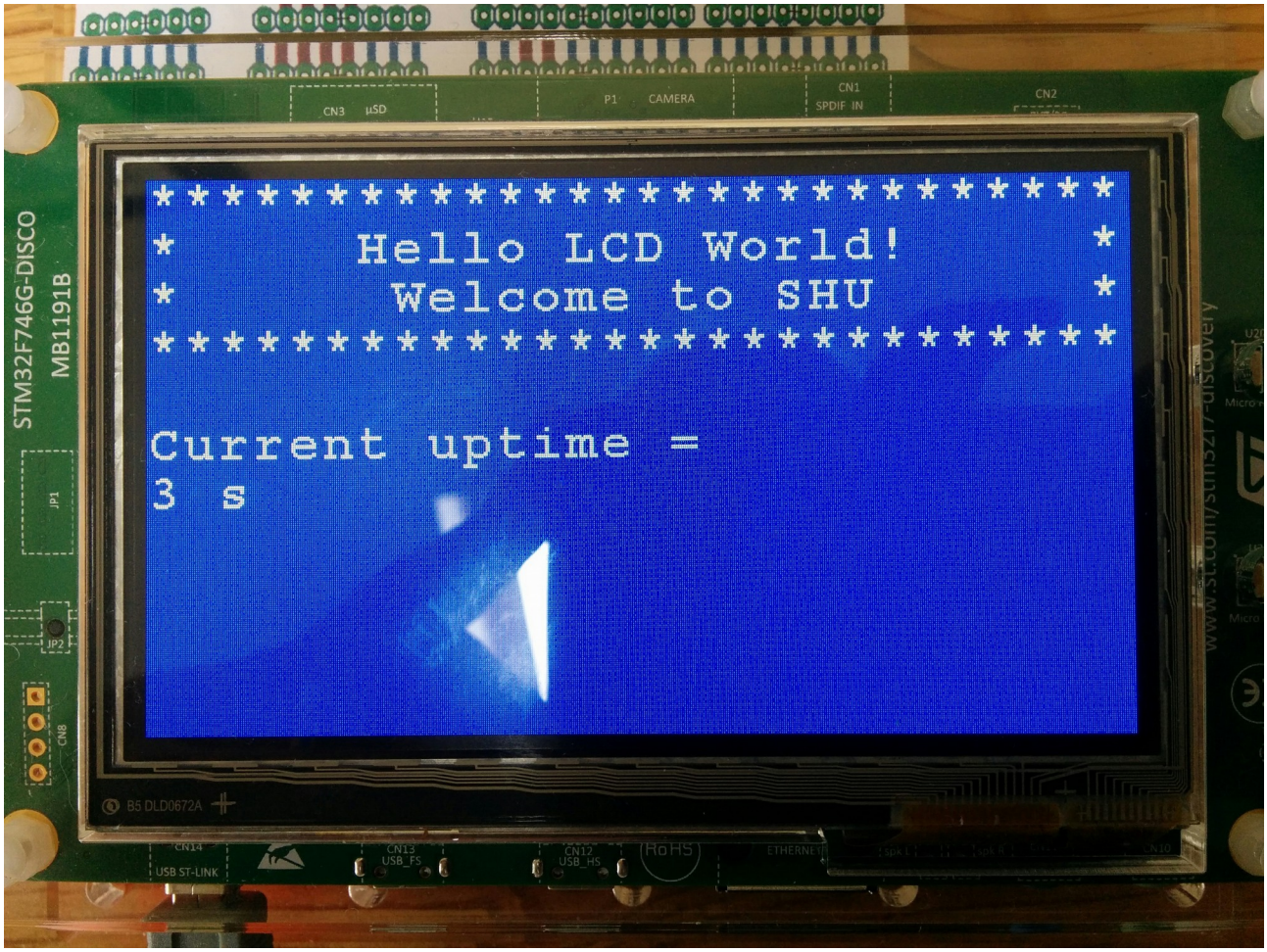


Figure 1 – The LCD “hello world” program in action

Code snippet 1:

```
/*
 * main.c
 *
 * this is the main lcd application
 *
 * author:    Dr. Alex Shenfield
 * date:      01/09/2017
 * purpose:   55-604481 embedded computer networks : lab 102
 */

// include the basic headers and hal drivers
#include "stm32f7xx_hal.h"

// include the shu bsp libraries for the stm32f7 discovery board
#include "pinmappings.h"
#include "clock.h"
#include "stm32746g_discovery_lcd.h"

// LCD DEFINES

// define a message boarder (note the lcd is 28 characters wide using Font24)
#define BOARDER     "****************************"

// specify a welcome message
const char * welcome_message[2] =
{
  "*     Hello LCD World!     *",
  "*      Welcome to SHU      *"
};
```

```
// CODE

// this is the main method
int main()
{
  // we need to initialise the hal library and set up the SystemCoreClock
  // properly
  HAL_Init();
  init_sysclk_216MHz();

  // initialise the lcd
  BSP_LCD_Init();
  BSP_LCD_LayerDefaultInit(LTDC_ACTIVE_LAYER, SDRAM_DEVICE_ADDR);
  BSP_LCD_SelectLayer(LTDC_ACTIVE_LAYER);

  // set the background colour to blue and clear the lcd
  BSP_LCD_SetBackColor(LCD_COLOR_BLUE);
  BSP_LCD_Clear(LCD_COLOR_BLUE);

  // set the font to use
  BSP_LCD_SetFont(&Font24);

  // print the welcome message ...
  BSP_LCD_SetTextColor(LCD_COLOR_WHITE);
  BSP_LCD_DisplayStringAtLine(0, (uint8_t *)BOARDER);
  BSP_LCD_DisplayStringAtLine(1, (uint8_t *)welcome_message[0]);
  BSP_LCD_DisplayStringAtLine(2, (uint8_t *)welcome_message[1]);
  BSP_LCD_DisplayStringAtLine(3, (uint8_t *)BOARDER);

  // delay a little ...
  HAL_Delay(5000);

  // display an "uptime" counter
  BSP_LCD_DisplayStringAtLine(5, (uint8_t *)"Current uptime =");
  int counter = 0;
  while(1)
  {
    // format a string based around the uptime counter
    char str[20];
    sprintf(str, "%d s", counter++);

    // print the message to the lcd
    BSP_LCD_ClearStringLine(6);
    BSP_LCD_DisplayStringAtLine(6, (uint8_t *)str);

    HAL_Delay(1000);
  }
}
```

Task 2

Previously we have read data from a potentiometer and used it to control the rate at which an LED flashed. Now you should add a potentiometer to the system and output the potentiometer value on the LCD display. You will have to use the `sprintf()`[2] function from the c standard io library to create the formatted string for writing to the LCD. Code snippet 2 (below) provides an example of how to use this function (the eagle eyed amongst you will notice that this is pretty much the same as the code for displaying the uptime counter in the last task).

Code snippet 2:

```
// include the stdio library
#include <stdio.h>

// <snip ...>

// main loop goes here ...

    // format a string based around the adc value and print to lcd
    char str[12];
    sprintf(str, "ADC = %4d", adc_val);
    BSP_LCD_DisplayStringAtLine(6, (uint8_t *)str);

// main loop ends here ...
```

Write your program!

2 https://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm

What is the maximum potentiometer value?

Why?

What is the minimum potentiometer value?

Now you are going to make some modifications to the program to add functionality:

1. Try and remap the potentiometer values to be within certain bounds (e.g. between 0 and 100). Use these ideas to make the LCD screen a percentage for the potentiometer value.

2. Now add some LEDs to your circuit and use the potentiometer to control when the LEDs come on. Output the status of the LEDs to the LCD screen.

3. Try outputting the potentiometer values to the LCD screen as a bar graph. You should look to scaling the potentiometer values and using them to create a rectangle[3]. I would **highly** recommend using pseudocode or other algorithm design methodologies before actually writing the code!

Figure 2 – An example bar graph display on the stm32f7 discovery board LCD

3 The *stm32746g_discovery_lcd.c* library provides a:

```
BSP_LCD_FillRect(uint16_t xpos, uint16_t ypos, uint16_t Width, uint16_t Height);
```

function that allows you to draw a rectangle of a specified width and height. In this function xpos is the horizontal alignment and ypos is the vertical alignment.

Task 3

You are now required to use the integrated circuit temperature sensors supplied to measure the temperature of the outside world. The sensors we will be using are Analog Device TMP36s (see Figure 3) and extracts from the data sheet are provided in Appendix A. Ensure that you insert these temperature sensors the correct way round (see data sheet for correct wiring order).

They get very hot otherwise!!

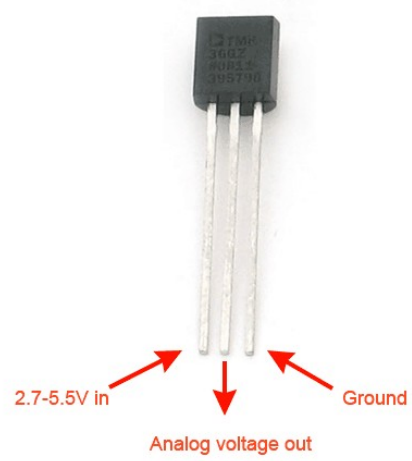


Figure 3 – A TMP36 temperature sensor

These sensors use a solid-state technique to determine the temperature. That is to say, they don't use mercury (like old thermometers), bimetalic strips (like in some home thermometers or stoves), nor do they use thermistors (temperature sensitive resistors). Instead, they use the fact as temperature increases, the voltage across a diode increases at a known rate. They amplify this voltage change by a precise amount to generate an analog signal that is directly proportional to the temperature.

Because these sensors have no moving parts, they are precise, never wear out, don't need calibration, work under many environmental conditions, and are consistent between sensors and readings. Moreover they are very inexpensive and quite easy to use.

As you can see from Figure 3 we get an analog voltage out (much like using a potentiometer). We will then have to convert that into a temperature using information from the data sheet (see Appendix A).

If you are unsure how to read the data sheet – ask a member of staff!!

Figure 4 shows a breadboard schematic for the circuit. Note, you can choose an input pin other than PA0 – but you should make sure that it works with the ADC[4]. The easiest way to do this is to stick to the pins that are broken out as A0-A5 on the Arduino form factor header (shown in Appendix B).

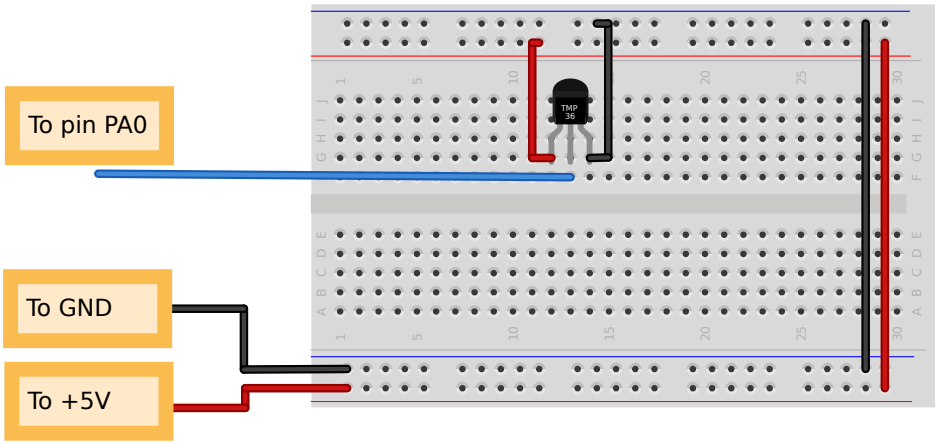


Figure 4 – The temperature sensing circuit

Now write a program to read this temperature sensor and display the actual temperature on the LCD screen!

4 To find out which pins work with the ADC we have to look at the stm32f7 data sheet - "DM00166116.pdf" in table 10 from pg 53 onwards - available on Blackboard

Task 4

In this task we are going to use light dependent resistors to obtain important information about the state of our environment (i.e. how dark it is!). Figure 5 shows one of the variants of light dependent resistor stocked by the university.

Figure 5 – An LDR

You are required to integrate this LDR into your circuit and display the results on the LCD display.

You might notice that the LDR only has two legs – this means that, like the pushbutton switch, you will need a pull up or pull down resistor in your circuit. Sketch your circuit below:

One common task when reading from analog sensors is calibrating those readings (i.e. tracking the highest and lowest values of a sensor during a start-up period and then using those values to remap the sensor range to be between another set of values). Code listing 3 provides some pseudo-code for this calibration process.

Code listing 3

```
Procedure CALIBRATE:

  while timer_tick < calibration_period:

    sensor_value = read_sensor()

    // check against maximum sensor value
    // found so far
    if sensor_value > sensor_max
        sensor_max = sensor_value
    end if

    // check against minimum sensor value
    // found so far
    if sensor_value < sensor_min
        sensor_min = sensor_value
    end if

  end while

end Procedure CALIBRATE
```

You should also rescale the values of your LDR to be a percentage. To do this you will need to know the values produced by the LDR when it is dark and when it is bright – write these in the box below:

LDR dark value: **LDR bright value:**

Now write your program!

You should now adapt your program from task 3 to display both temperature and light levels on the LCD screen. This is the basis for an environmental data acquisition and display unit!

Appendix A – Extracts from the TMP36 data sheet

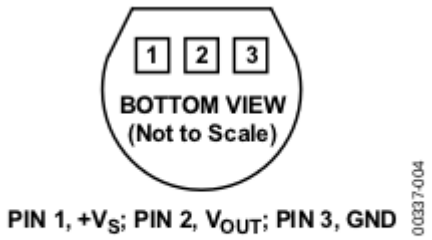


Figure 4. T-3 (TO-92)

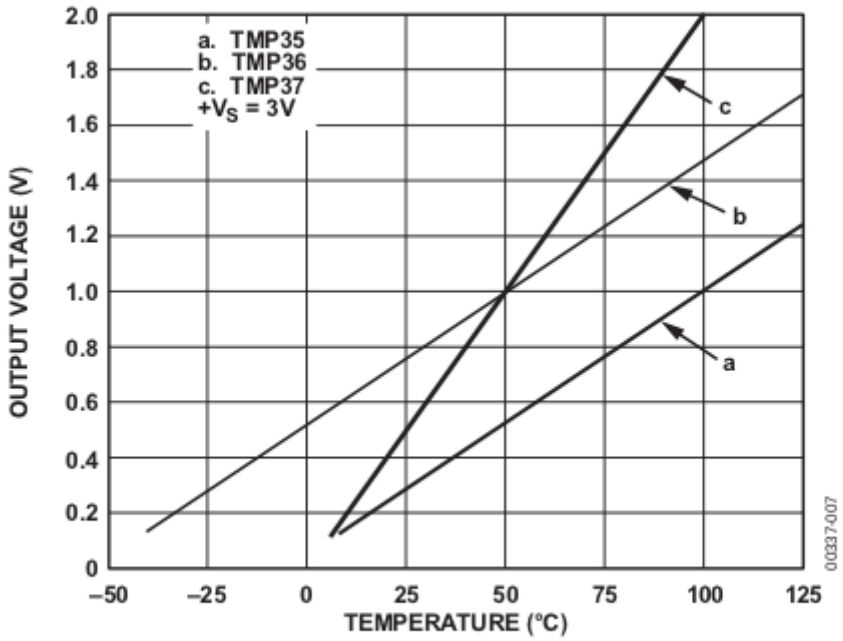
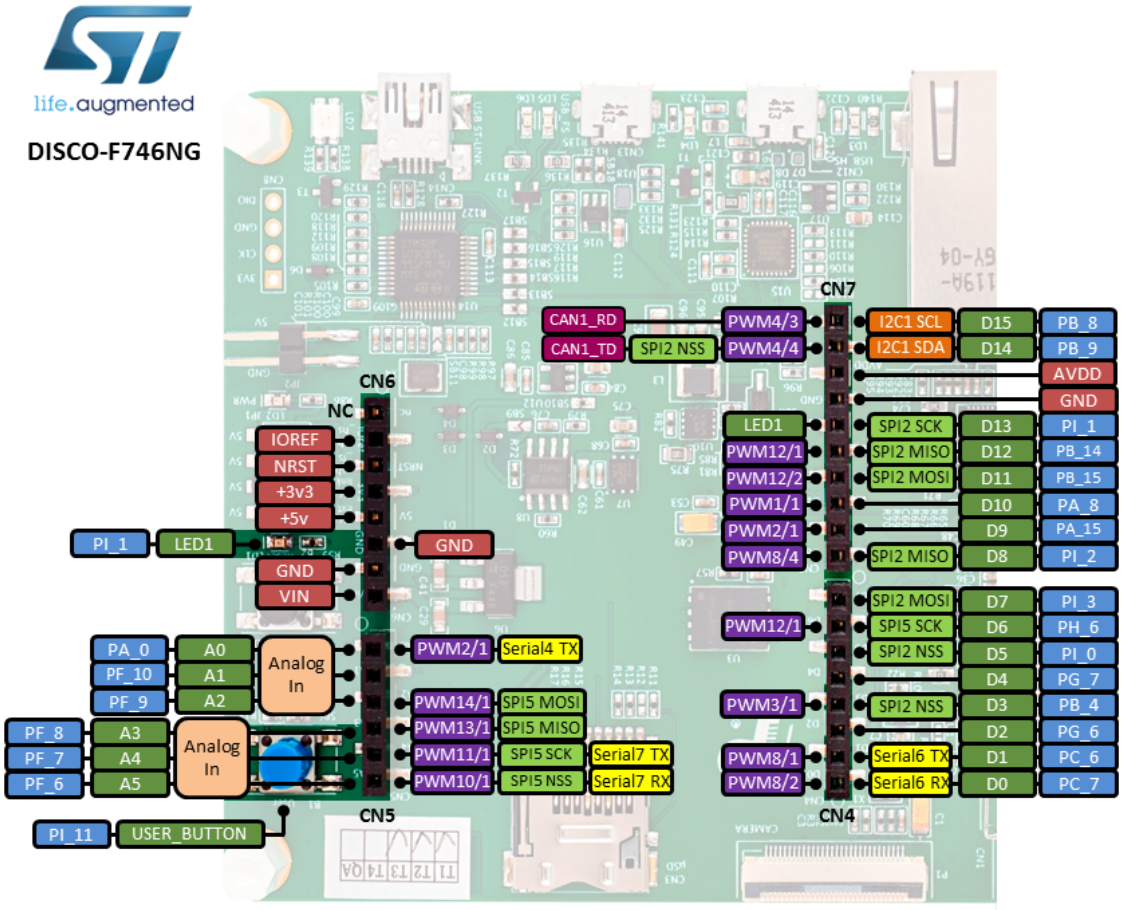


Figure 6. Output Voltage vs. Temperature

Table 4. TMP3x Output Characteristics

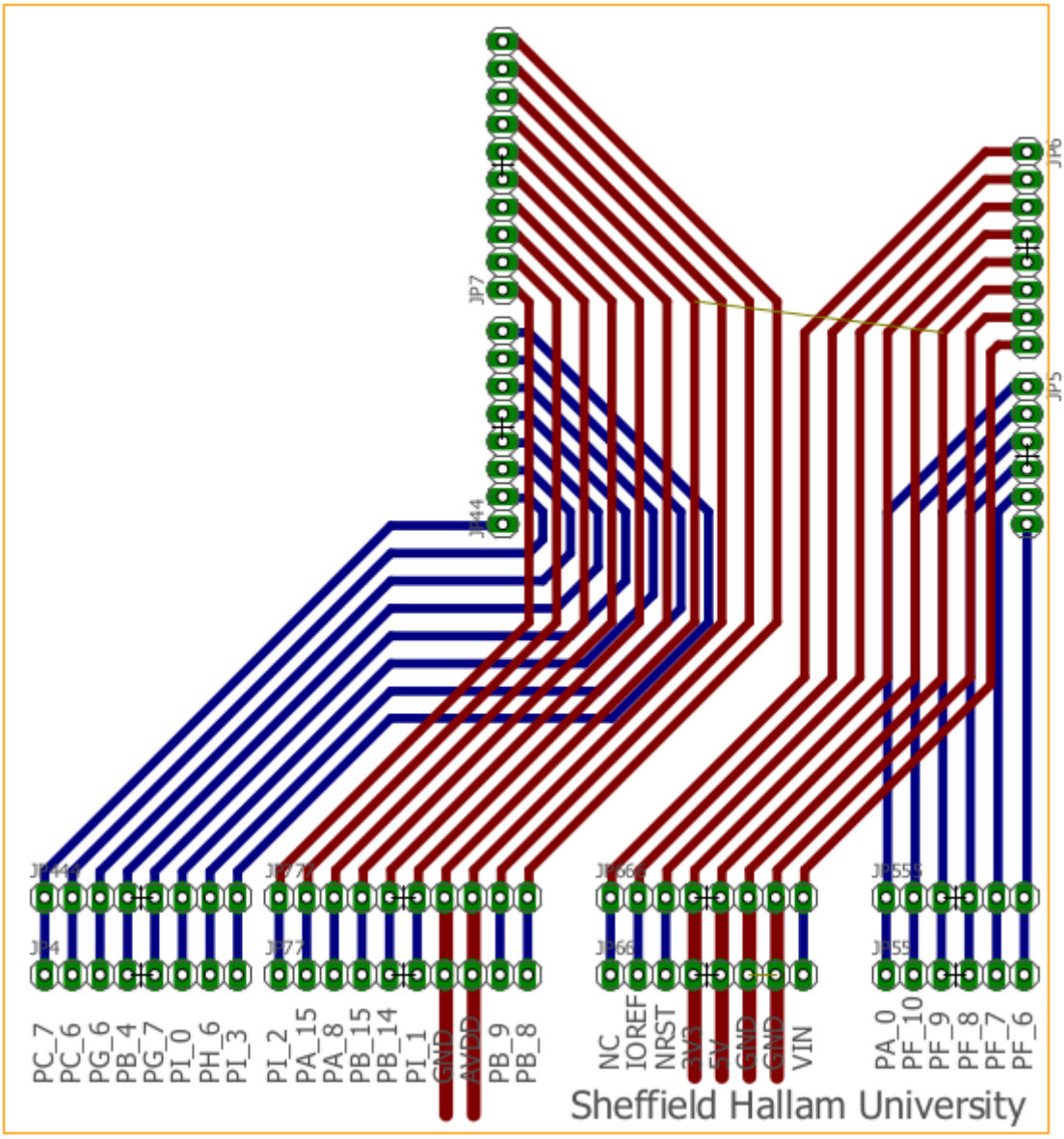
Sensor	Offset Voltage (V)	Output Voltage Scaling (mV/°C)	Output Voltage @ 25°C (mV)
TMP35	0	10	250
TMP36	0.5	10	750
TMP37	0	20	500

Appendix B – The STM32F7 discovery board schematic



STM32F7 discovery board pin outs

Appendix C – The STM32F7 discovery board SHU base board schematic



SHU breakout board schematic for the STM32F7 discovery