ELECTRONIC DESIGN PROJECT 2

Heart beat measurement device

Group F

2211603 – Samuel Chua
2169252 – Joseph Neto
2211604 – Alvin Chan
2200408 – Varun Nadar

MARCH 24, 2016
UNIVERSITY OF GLASGOW

Contents

1. Introduction

The purpose of this project is to create a heart rate monitor that can show similar trace to an oscilloscope and have a separate LED which flashes each time a pulse is detected by the instrument. The heart rate monitor for this project need to be able to operate using an external battery power source and does not require support from any other computer. The last requirement for this heart rate monitor is to be smaller than 6” x 6” in area and smaller area is preferable. This report will outline the overall design and software for the heart rate monitor. The result section will show the working product which can show the outcome as required by the requirements as well as discussion on glitches observed when using this product.

2. Overall Design

2.1 Principle of operation

The heart rate monitor works with user inserting finger into the probe which consists of a red light emitting diodes (LED) and a phototransistor as shown in figure 1. When the light source travels from the red LEDs to the phototransistor via the finger, the light will be absorbed by the tissue and blood hence changing the wavelength of the light source. The light detected by the phototransistor will be converted into a signal and sent to the amplifier circuit for amplification and filtration. Finally, the signal will be processed by the embedded computer with reference to the signal of red LED and output the result on the LED display.

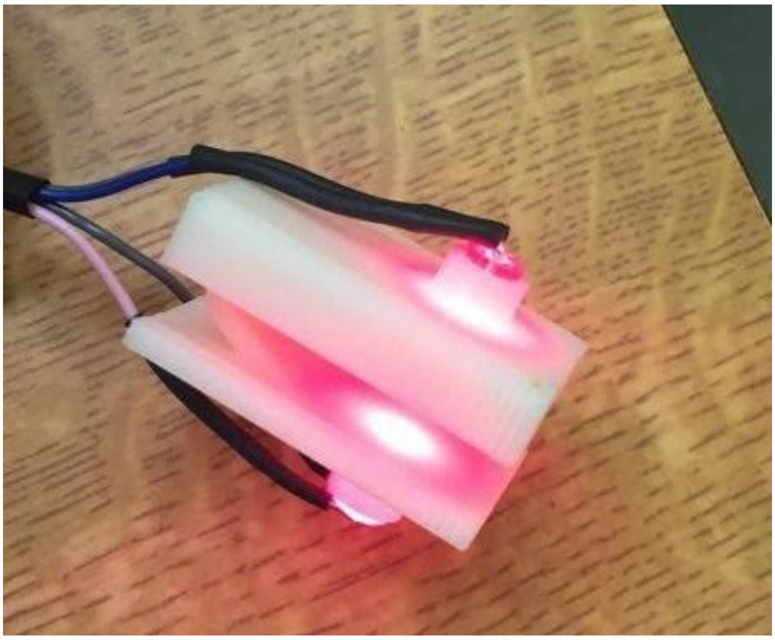

Figure 1: Sensor probe; LED (Bottom), Phototransistor (Top)

2.2 Overall Diagram

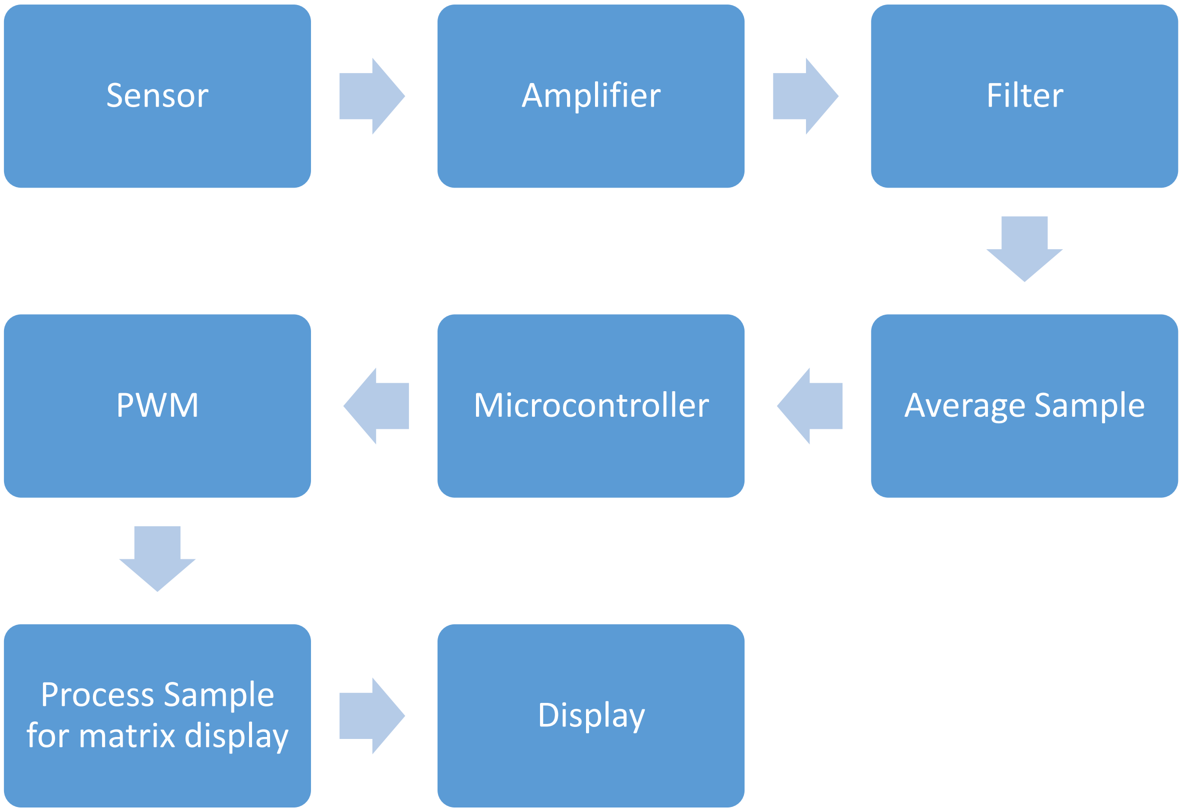

Figure 2: Overall diagram

The concept of this project is shown in figure 2. First, the information received by the sensor were converted into a signal and the signal was sent to the amplifier for amplification and filter for filtration. The amplified and filtered signal will be sent to the microcontroller to process for matrix display. Finally, the result will be displayed on an 8x8 LED matrix.

2.3 Sensor

The sensor consists of a red LED and a phototransistor as discussed earlier. The BPV 11 silicon NPN phototransistor was used for this project as it is capable of detecting a wide range of wavelength (450nm to 1080nm), operating in extreme temperatures (-40˚C to 100˚C) as well as not interfered by mobile frequencies due to the cut-off frequency of 110kHz. [1] Red LED was used for this project as its wavelength of around 800nm to 900nm [2] is most suitable for the phototransistor used.

Different combinations of resistor were tested to achieve quality results and it was concluded that 75 ohms' resistor is the most suitable candidate as it supplies 24mA of current to the LEDs. Despite having a low current supply, the light is still capable of penetrating through the skin. However, it was observed that the light have difficulty penetrating through darker skin tone.

2.4 Amplifier

Figure 4 below shows the amplifier schematic created in OrCad.

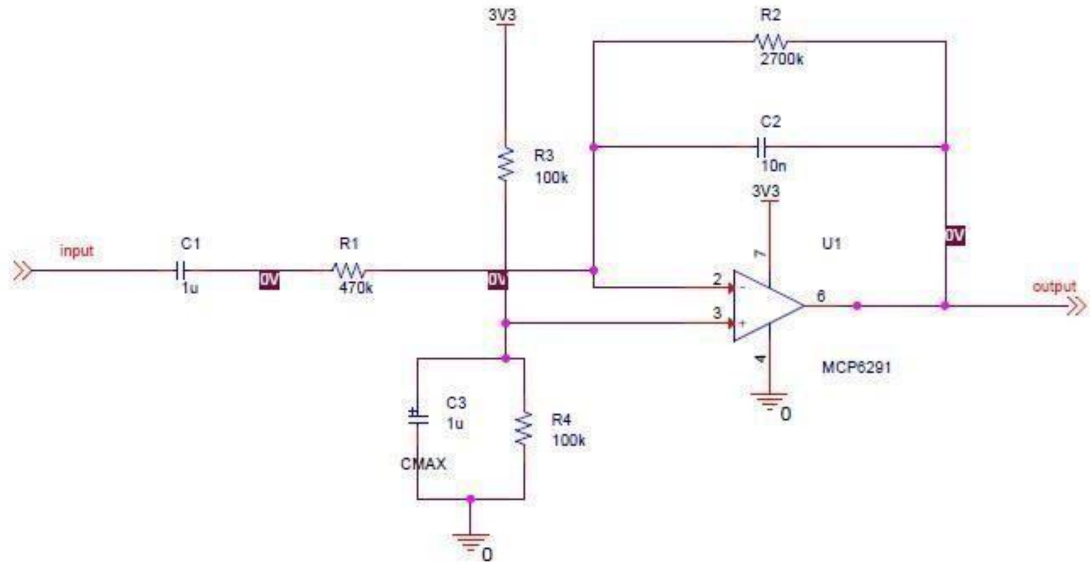

Figure 3: Amplifier schematic

In this amplifier schematic, C1 refers to the DC blocking capacitor which is responsible for blocking the DC offset.

This amplifier was designed as a band pass filter. The bandwidth of this amplifier was set to be between the lower cut-off frequency of 1Hz (approximately 60bpm for human heartbeat) and upper cut-off frequency of 3Hz (approximately 180bpm for human heartbeat). This is designed for the human heart as the heart beats between 60bpm and 110bpm for a healthy person at rest. [3]

During testing, it was observed the 50Hz signal was not filtered hence a low pass filter was installed at the output of the amplifier as noise removal. The outcome of the result is as seen in figure 5.

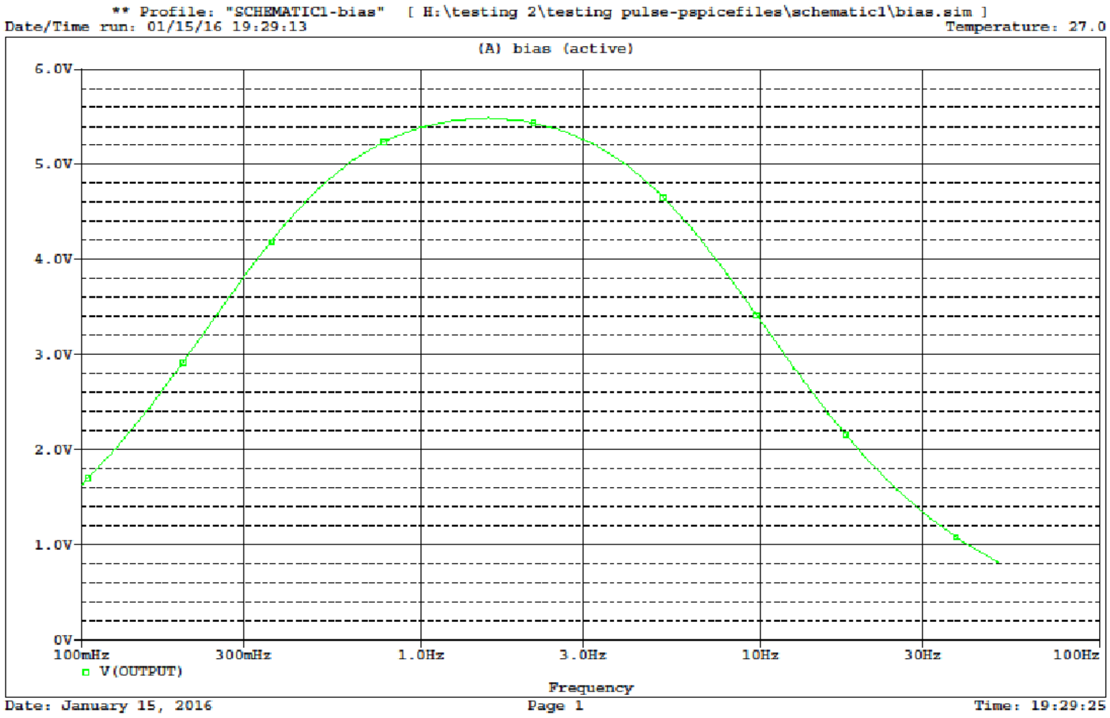

Figure 4: Output simulation of amplifier circuit without low pass filter (OrCad)

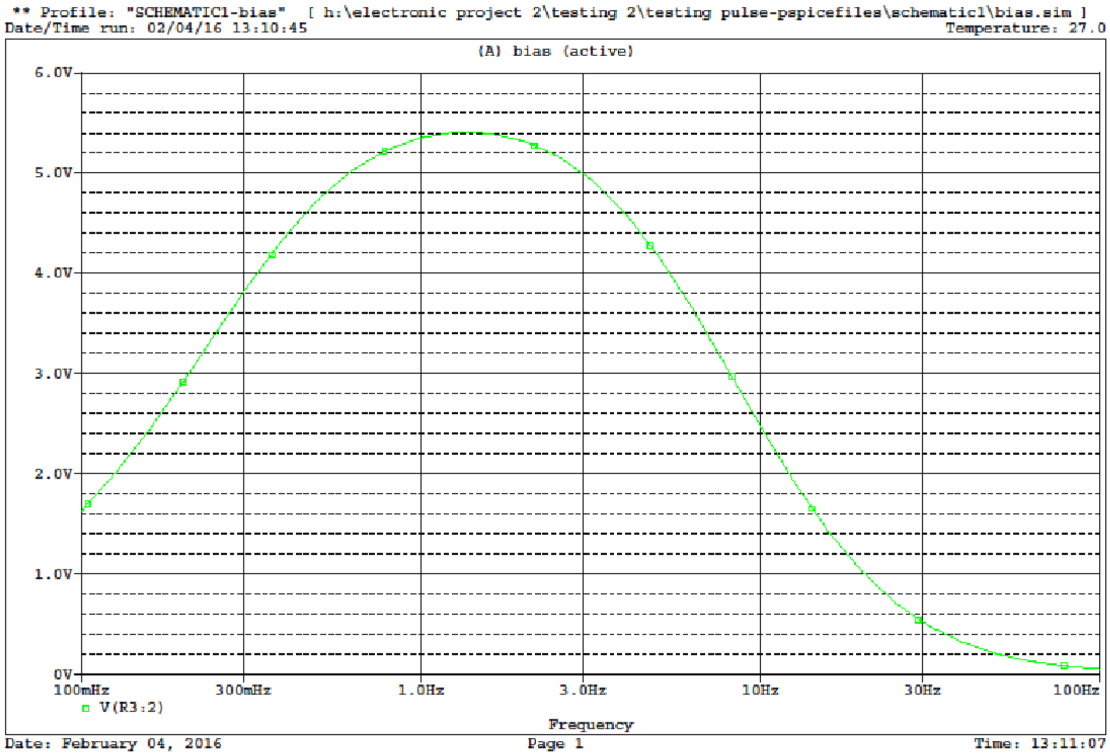

Figure 5: Output simulation of amplifier circuit with low pass filter (OrCad)

2.5 Microcontroller

The microcontroller provided is the FRDM – KL25Z embedded processor. This microcontroller can communicate with the MAX7221 display driver via Serial Peripheral Interface (SPI). This microcontroller has multiple general purpose input/output (GPIO) pins which allow the signal to be processed by the built-in processor via the ADC port and the processed signal will be outputted to the oscilloscope via the digital to analogue converter (DAC). This microcontroller has an RGB (Red, Green, Blue) LED which interacts with the output pin. This LED will flash whenever a pulse is detected.

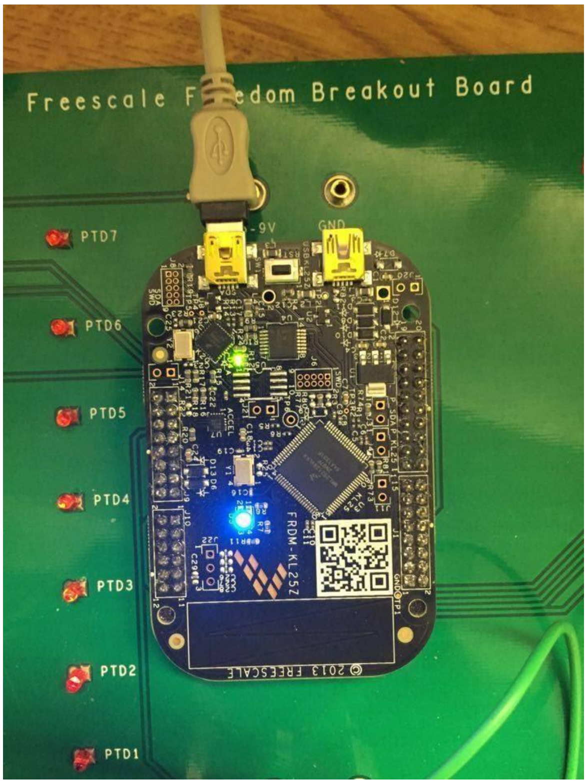

Figure 6: FDRM - KL25Z microcontroller

2.6 Display

The display chip and display driver used for this project shown in figure 7 are a HX-12088BEG 8x8 LED matrix and MAX7221 display driver. The LED driver was programmed to display waveform as displayed on the oscilloscope. However, the waveform needs to be shrank to fit the 8x8 LED matrix.

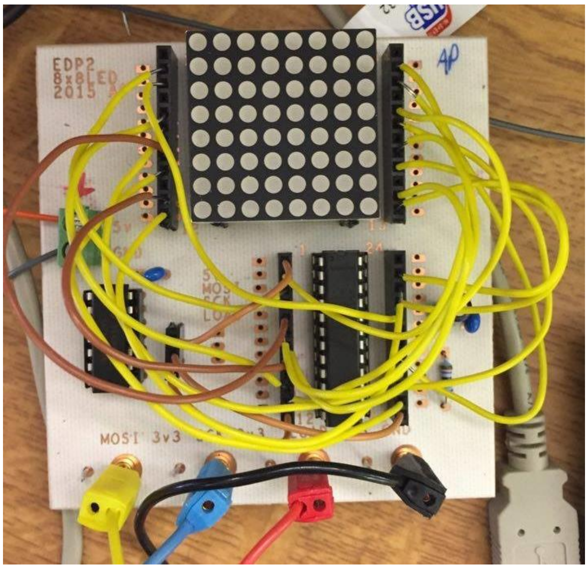

Figure 7: 8x8 LED Matrix (Top); MAX7221 Display driver (Bottom right)

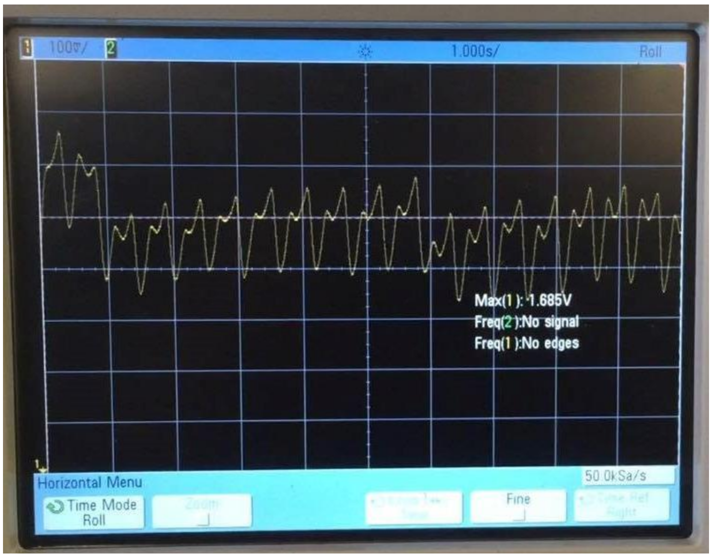

Figure 8: Output signal after amplification on oscilloscope

2.7 Power Supply Unit – PSU

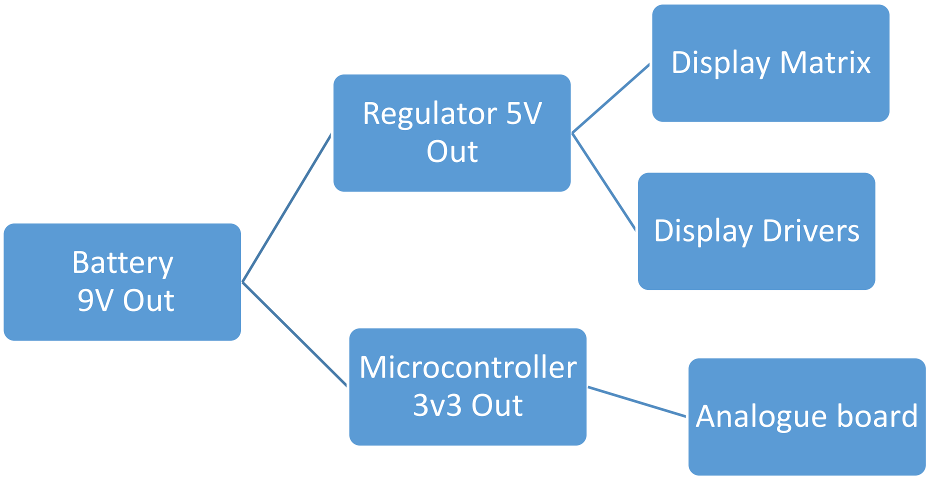

Figure 9: Power supply unit wiring plan

The source of this power supply unit is from a PP3 9V battery. This battery will supply 9V to the microcontroller and to the LM2938 regulator to regulate the voltage to a fixed 5V DC output. The regulated 5V DC is being supplied to the display matrix and drivers whereas the microcontroller's built-in regulator outputs 3v3 DC to the analogue board.

The regulator was used in the standard configuration as given in the datasheet. [4]

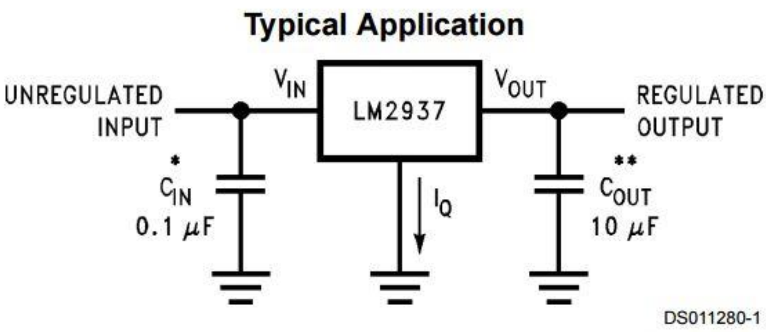

Figure 10: Voltage regulator

2.7.1 Heatsink

Using the equations as given in the datasheet [4] and the expected values for current. Calculations for the circuit was done in Appendix A. The result suggests that a heatsink is not required for the regulator.

3. Software Design

3.1 Program flow

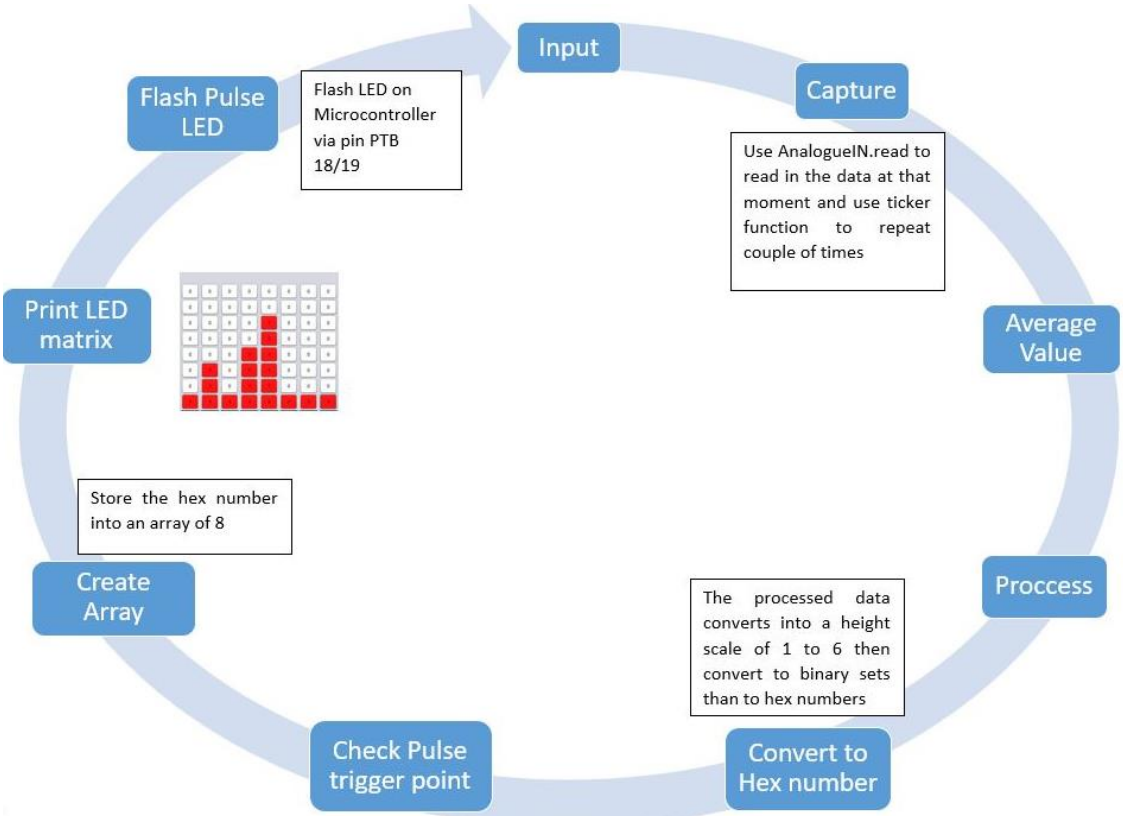

Figure 11: Program flow

Diagram 3 shows the flow of the program. The program was designed such that 100 samples is captured and averaged so that the values can be processed.

3.2 Pulse measuring algorithm

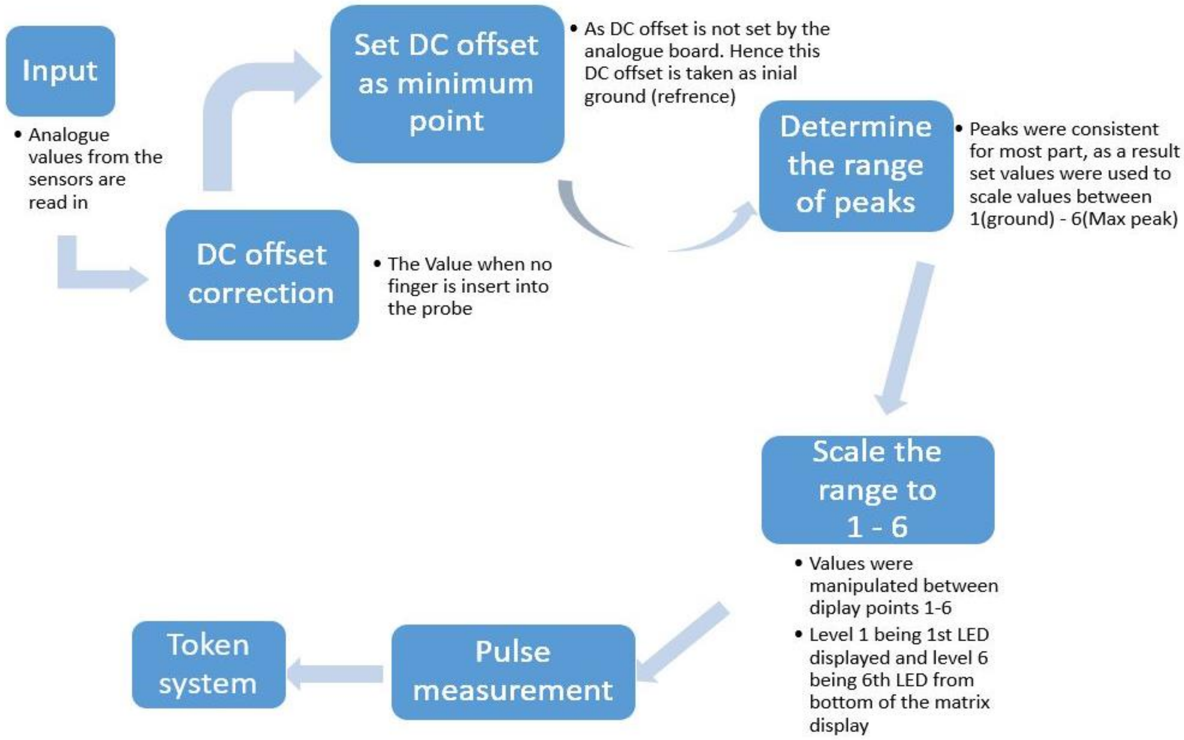

Figure 12: Pulse measuring algorithm

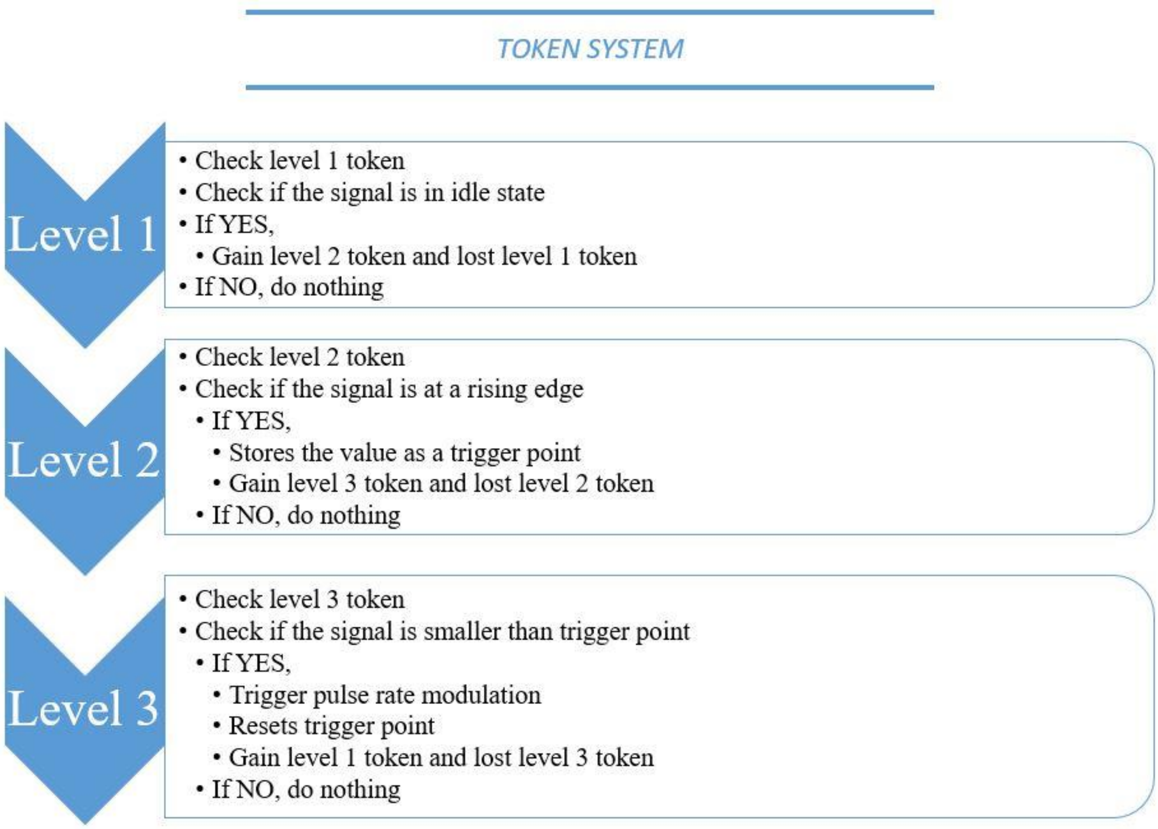

Figure 13: Token system

Figure 12 outlines the process section of the program flow and figure 13 explains how the token system works.

Firstly, the analogue value input from the sensor was sent for DC offset correction as the reference signal. This is done by recording the signal from the probe with finger absent. Next, the peak value was determined by recording the value from the probe with finger present. Subsequently, the range of values was scaled down to level 1 as the minima and level 6 as the maxima. Consequently, these values will be displayed on the LED matrix as pulse wave. Finally, the token system utilizes the pulse wave to detect the heartbeat. The LED will flash whenever a heartbeat is detected.

3.3 Display driver

The software is programmed such that the input signal is converted into integer range from 1 to 6. Each integer is triggered by a different upper and lower values. The integer range will determine the row of LEDs lit up in each column.

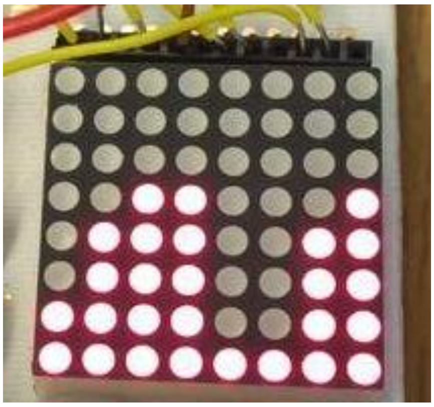

Figure 14: 8x8 LED display matrix

4. Results

The working PCB as shown below is functional but some glitches were found. The accuracy of the pulse rate was found to be similar to the traditional nurse trained method when the user is not moving. However, the RGB led that detects heartbeat is not stable when the user's finger is moving. Other than that, surrounding lighting conditions also affect the reliability of the sensor resulting in inconsistent results.

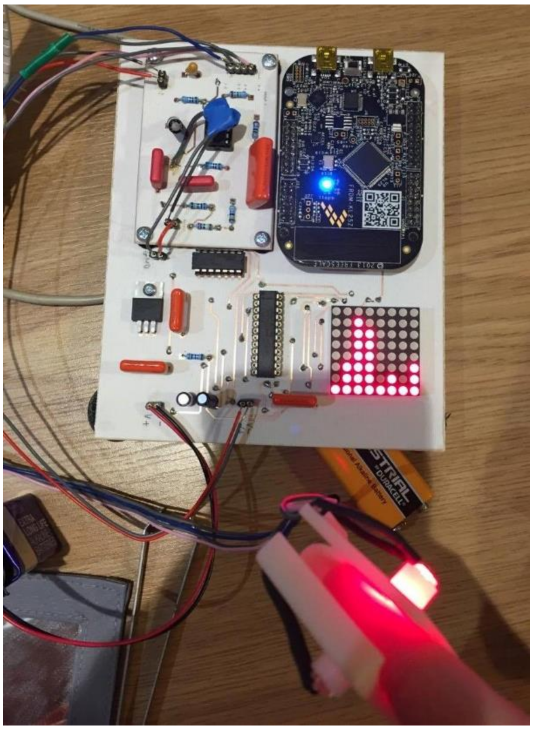

Figure 15: Functional display matrix

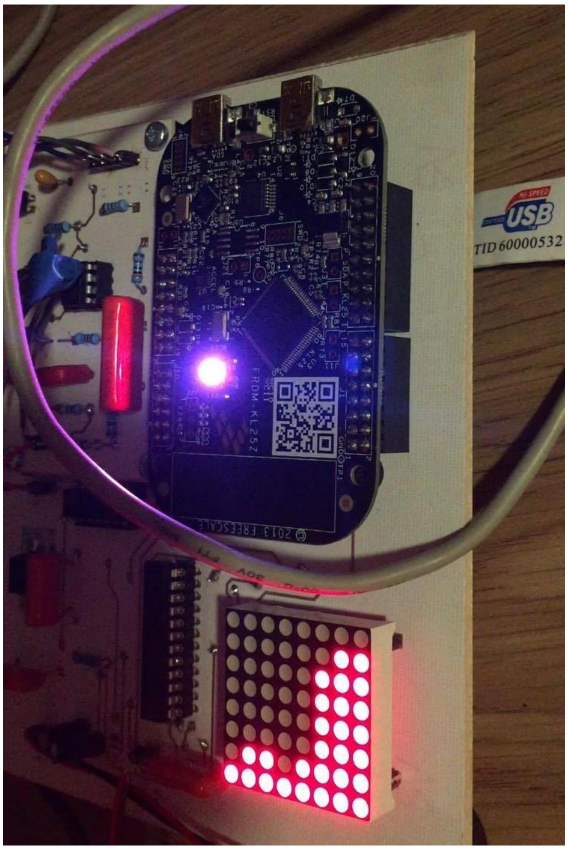

Figure 16: RGB LED flashes when each pulse is detected

5. Conclusion

The heart rate monitor created for this project has fulfil the requirement by able to achieve the task as shown below:

1. Show on the LED display a trace, similar to an oscilloscope trace, which shows the pulse of the patient as it happens.
2. Have a separate LED which flashes each time a pulse is detected to show that pulses are being detected by the instrument.
3. Be able to work from battery power and also not need the support of any other computer.
4. Be no larger than 6” x 6” in area.

However, there are some glitches that has caused inconsistency in the result. The glitches are shown as below:

1. Dark skin tone reduce the effectiveness of the sensor.
2. Bright surrounding lighting condition will affect the data recorded by the phototransistor.
3. Finger motion will result in unreadable data.

6. References

[1] Vishay Semiconductors, "Silicon NPN Phototransistor," 03 March 2013. [Online]. Available: http://moodle2.gla.ac.uk/pluginfile.php/736659/mod_resource/content/1/bpv11.pdf.

[2] Kingbright, "T-1 3/4 (5mm) SUPER BRIGHT LED LAMPS," 23 September 2001. [Online]. Available: http://moodle2.gla.ac.uk/pluginfile.php/739867/mod_resource/content/1/72-8982.pdf.

[3] American Heart Association, "Target Heart Rates," 13 January 2016. [Online]. Available: http://www.heart.org/HEARTORG/HealthyLiving/PhysicalActivity/Target-Heart-Rates_UCM_434341_Article.jsp.

[4] National Semiconductor Corporation, "LM 2937 550 mA Low Dropout Regulator," July 2000. [Online]. Available: http://moodle2.gla.ac.uk/pluginfile.php/395128/mod_resource/content/1/LM2937-LDO-5V%20regulator.pdf.

7. Appendix

7.1 Appendix A

Parameters:

- TR (max): is the maximum allowable temperature rise.
- TJ (max): is the maximum allowable junction temperature. This is taken as 125℃ for commercial part.
- TA (max): is the maximum ambient temperature. Taken according to the location 11℃.
- Θ (max): the maximum allowable value for the junction-to-ambient thermal resistance.
- PD: Power dissipated
- **Finding PD:**

$$P_D = (Vin - Vout) \times I_L + (Vin) \times I_G$$

$$P_{D=}\ 4 * 421.0410^{-3} + 9 * 20.10^{-3}$$

$$P_{D=} 1.86416\ W$$

- **Finding TRmax**

$$T_R(max) = T_J(max) - T_A(max)$$

$$T_R(max) = 125\ \mathring{C} - 25\ \mathring{C} = 100\ \mathring{C}$$

- **Finding Θ (max):**

$$\theta max = \frac{T_R(\max)}{P_D} = \frac{100}{1.86416} = 53.64\ {}^{\circ}C/_{W}$$

- **The maximum temperature T_A**

$$53\ {}^{\circ}C/_{W} = \frac{125-X}{1.86416}$$

X=26.19˚C

the circuits is designed to work at temperature under 26.19˚C, for a higher temperature the device will not work properly.

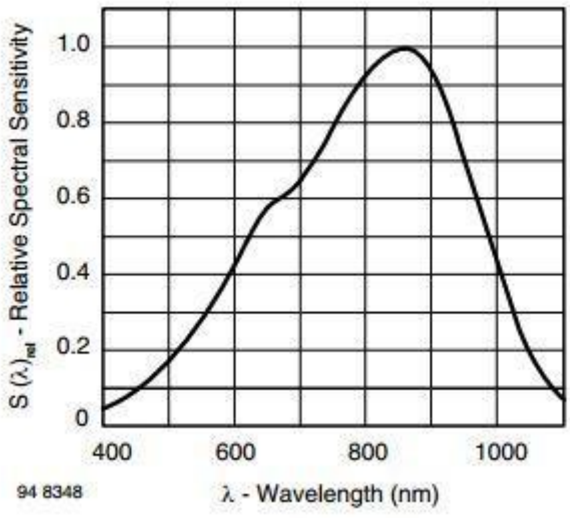

Figure 17: Sensitivity of phototransistor BPV11

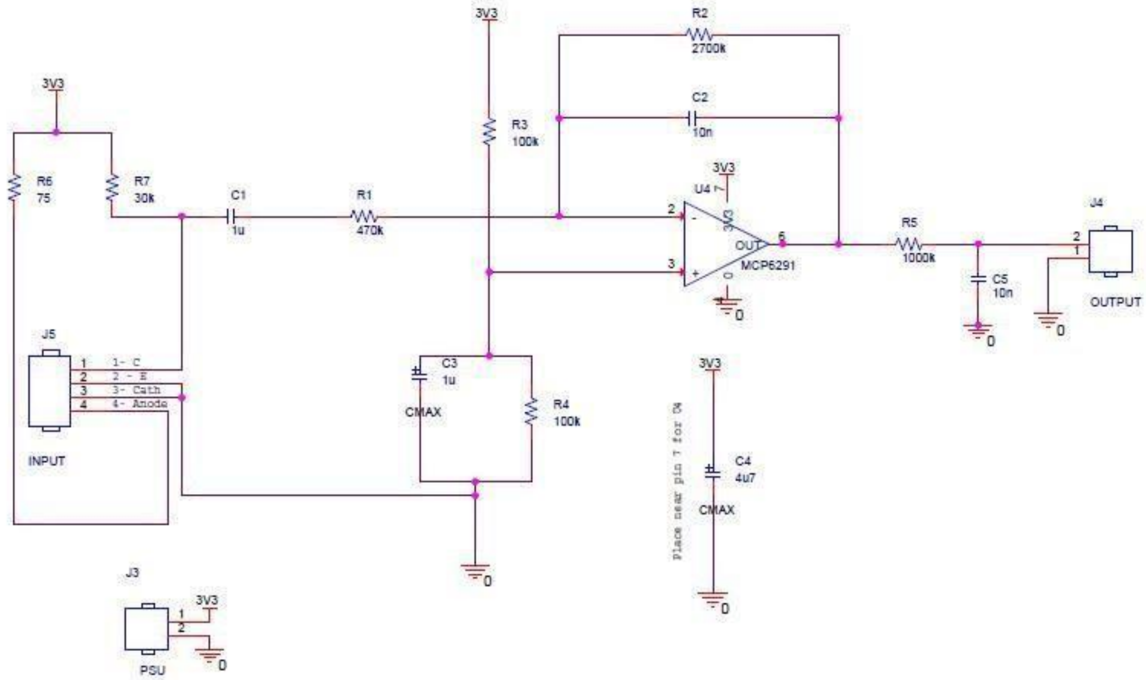

Figure 18: Schematic of the amplifier circuit (OrCad)

RGB LED	KL25Z128
Red Cathode	PTB18
Green Cathode	PTB19
Blue Cathode	PTD1 [1]

Table 1: LED pins

Supply Source	Valid Range	OpenSDA Operational?	Regulated on-board?
OpenSDA USB (J7)	5V	Yes	Yes
KL25Z USB (J5)	5V	No	Yes
V_{IN} Pin	4.3-9V	No	Yes
3.3V Pin	1.71-3.6V	No	No
Coin Cell Battery	1.71-3.6V	No	No

Table 2: Power supply source for FRDM - KL25Z (Datasheet)

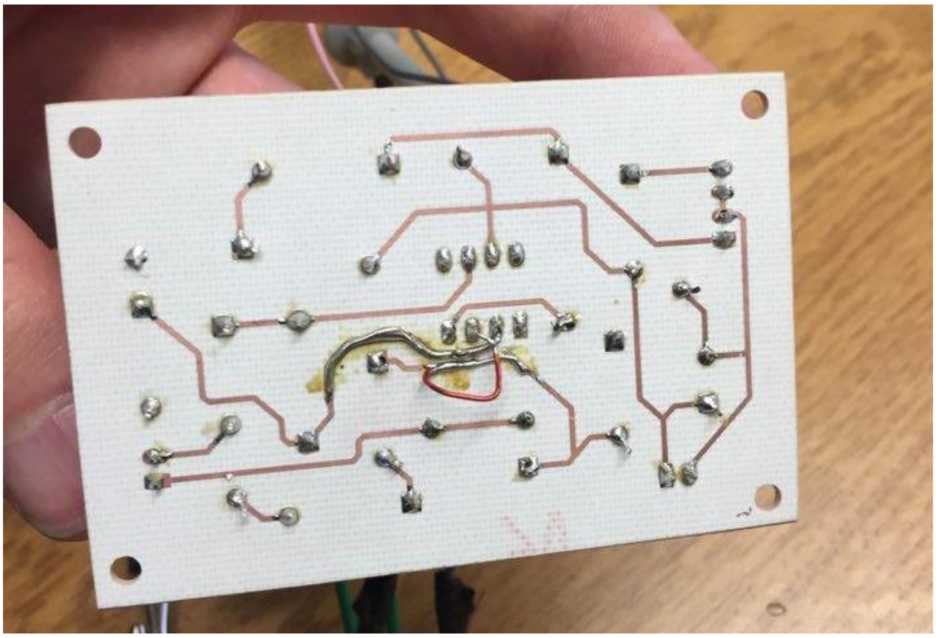

Figure 19: Rewired analogue board

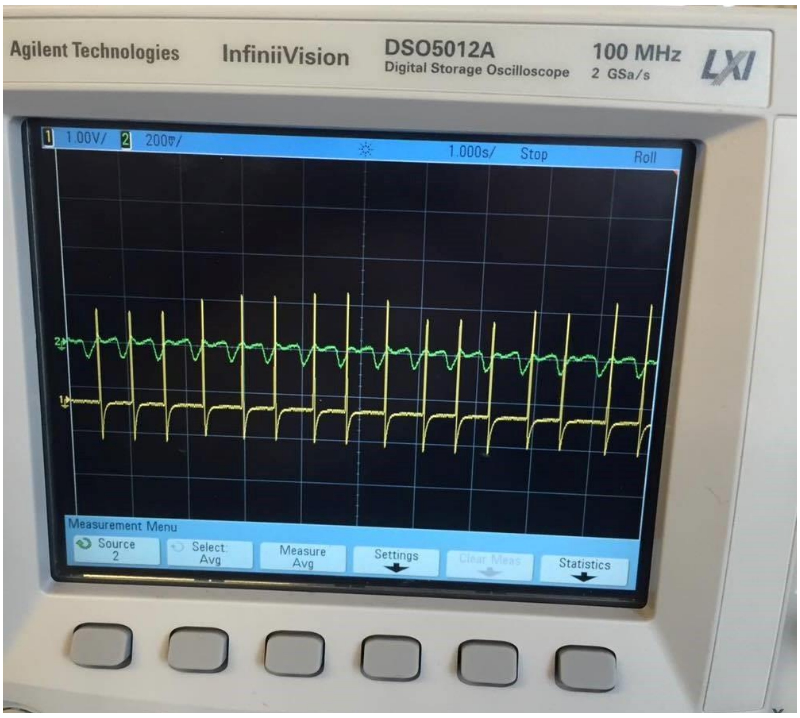

Figure 20: Scope trace in steady state (Green signal); Indication when 1 pulse is detected (Yellow signal)

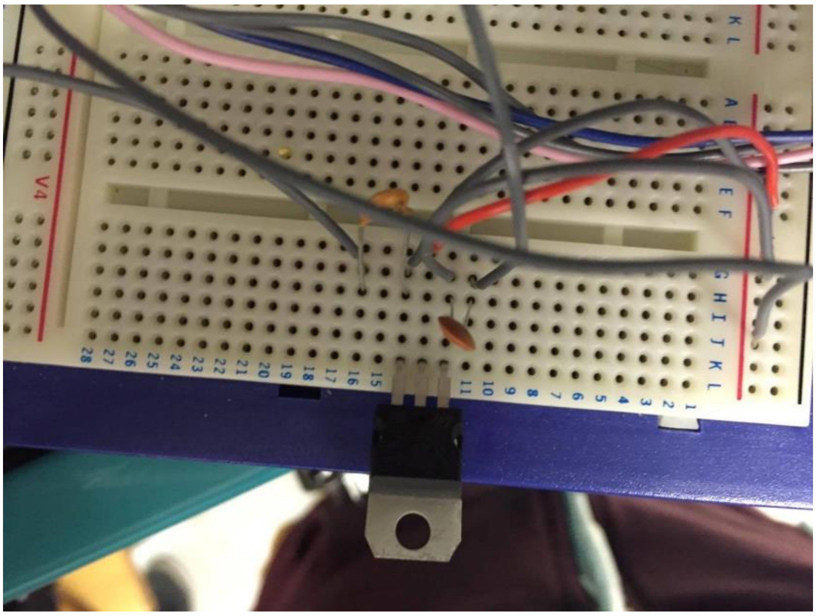

Figure 21: LM2937 regulator

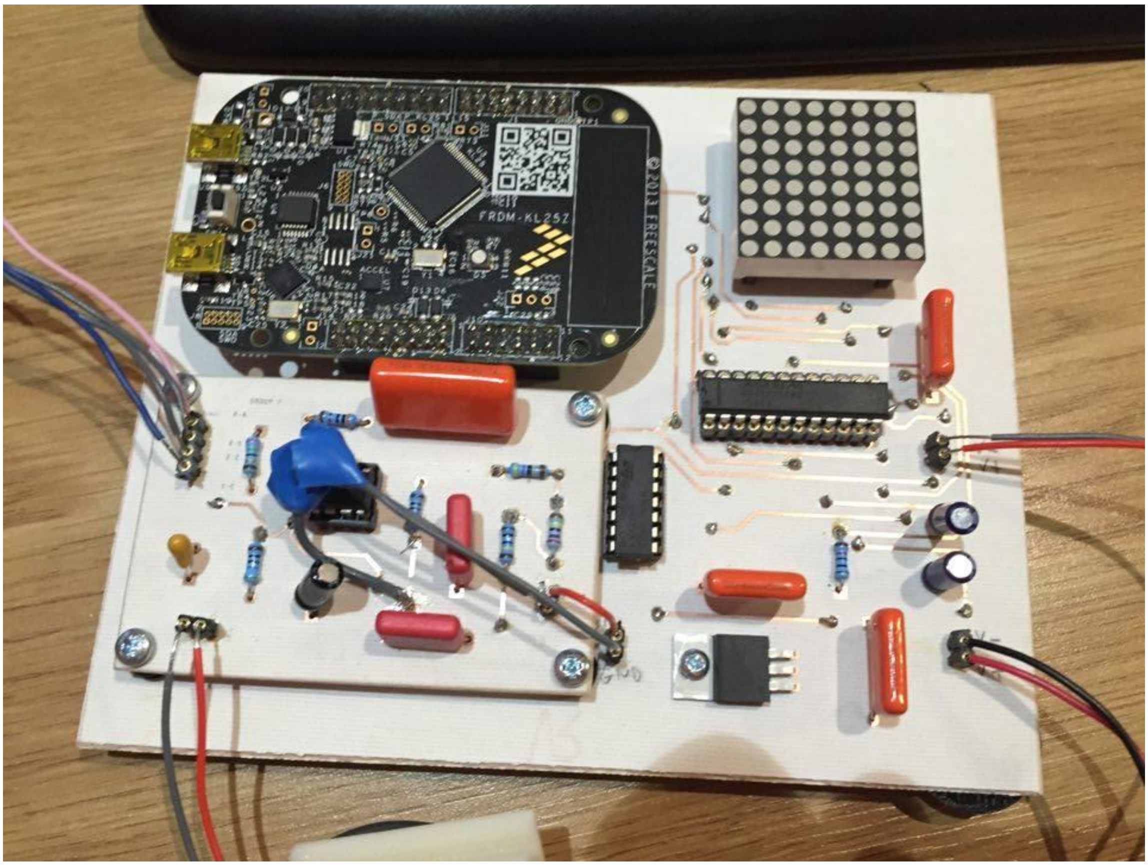

Figure 22: Full PCB

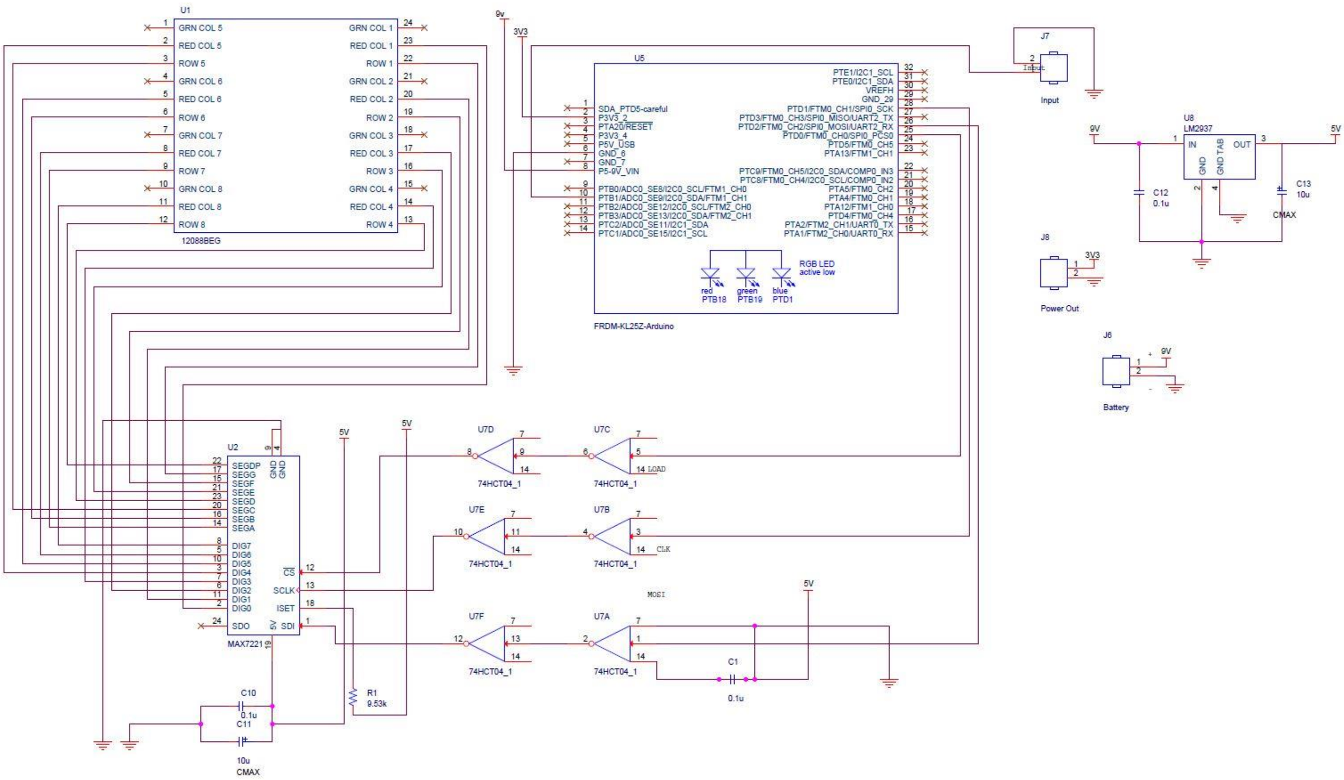

Figure 23: Full schematic

7.2 Appendix B

```
#include "mbed.h"

#define max7219_reg_noop         0x00

#define max7219_reg_digit0       0x01

#define max7219_reg_digit1       0x02

#define max7219_reg_digit2       0x03

#define max7219_reg_digit3       0x04

#define max7219_reg_digit4       0x05

#define max7219_reg_digit5       0x06

#define max7219_reg_digit6       0x07

#define max7219_reg_digit7       0x08

#define max7219_reg_decodeMode   0x09

#define max7219_reg_intensity    0x0a

#define max7219_reg_scanLimit    0x0b

#define max7219_reg_shutdown     0x0c

#define max7219_reg_displayTest  0x0f

#define LOW 0

#define HIGH 1

SPI max72_spi(PTD2, NC, PTD1);

DigitalOut pulse_trig(LED1);

DigitalOut load(PTD0);

AnalogIn Ain(PTB1);

Ticker TimerInt;

int temp1 = 0, temp2 = 0, max_val=0, min_val=6000;

int peak1= 0, peak2 = 0, count_avg = 1, prog_run = 0;

volatile int i_timer,j_timer;

double value =0;

char data[8] = {0,0,0,0,0,0,0,0};

void write_to_max(int row,int col){

    load=LOW;

    max72_spi.write(row);

    max72_spi.write(col);
```

```
    load=HIGH;
}
void pattern_to_display(char *testdata){
    int cdata;
    for(int idx = 0; idx <= 7; idx++) {
        cdata = testdata[idx];
        write_to_max(idx+1,cdata);
    }
}
int convert(double input){
    double x;
    int i, j, k;
    i = 0;
    x = input*10000;
    if (x > max_val){
        max_val = x;
    }
    if (x< min_val){
       min_val = x;
    }
    x = x - 4000;
    if (x >= 900 && x <= 925)
        i= 1;
    else if(x < 900)
        i= 1;
    else if(x >= 900 && x <=950)
        i=2;
    else if(x > 950 && x <= 1021)
        i= 3;
    else if(x > 1021 && x <= 1020)
        i= 4;
    else if(x > 1020 && x <= 1050)
```

```
        i= 5;
    else if(x > 1050 && x <= 1096)
        i = 6;
    else if (x > 1096)
        i = 7;
    if (x >= 910 && x <= 925)
        i = 1;
    if(temp1 == 0 && prog_run == 0 && i != 1){
        temp1 = x;
        prog_run = 1;
    }
    if ( x > temp1 && (prog_run == 1 || peak1 != 0)){
        temp1 = x;
        prog_run = 2;
    }
    if(x < temp1 && prog_run == 2){
        temp1 = 0;
        peak1 = x;
        temp1 = 0;
        peak1 = x;
        pulse_trig = 0;
        prog_run = 0;
        peak1 = 0;
        peak2 = 0;
        temp2 = 0;
    }
     j=0;
    for (k=1; k<=i; k++){
    j=j<<1;
    j=j++;
    }
    return j;
```

```
}
void output(float input){
    int x,i;
    x=convert(input);
    ::data[0]=x;
    pattern_to_display(::data);
    for (i=7;i>-1;i--){
        ::data[i]= ::data[i-1];
    }
}
 void clear(){
     for (int e=1; e<=8; e++) {
        write_to_max(e,0);
    }
}
void average_data(){
    double average;

    if (::count_avg >= 100){
        average = ::value / ::count_avg;
        ::value = 0;
        ::count_avg = 1;
        output(average);
    }
}
void flip() {
    i_timer=!i_timer;
    load=i_timer;
    if(::count_avg<=100){
        ::value = ::value + Ain.read();
        ::count_avg++;
    }
```

```
    average_data();

    pulse_trig = 1;

}

void setup_dot_matrix (){

    max72_spi.format(8, 0);

    max72_spi.frequency(100000);

    write_to_max(max7219_reg_scanLimit, 0x07);

    write_to_max(max7219_reg_decodeMode, 0x00);

    write_to_max(max7219_reg_shutdown, 0x01);

    write_to_max(max7219_reg_displayTest, 0x00);

    for (int e=1; e<=8; e++) {

        write_to_max(e,0);

    }

    write_to_max(max7219_reg_intensity,  0x08);

}

int main() {

    setup_dot_matrix ();

    TimerInt.attach(&flip, 0.00125);


    ::i_timer=LOW;

    ::j_timer=LOW;


    while (true) {

    }

}
```