ELC 5396 02 Final Project Report

Elizabeth Kooiman

December 5, 2023

Summary

The purpose of the project was to integrate an ADC and use it to obtain input information from a microphone to determine how close a 5 kHz sinusoidal sound source is to the microphone. This project used the XADC integrated in the Nexys4 DDR board, an electret microphone, and various op amps and resistors to perform filtering and amplification of the signal. Once the signal from the microphone was filtered and read, the seven segment display was used to display the voltage of the signal read and the LEDs were used to visually indicate the magnitude of the signal read. In addition to the microphone input, the project will display the current VCC of the FPGA as well as the temperature of the FPGA depending on which switch is on. The code for this project can be found in GitHub at https://github.com/elizabethkooiman/Kooiman_SoC_Project.

Results

To start, the hardware was designed and built to allow the microphone signal to be integrated with the ADC. Figure 1 shows the schematic of the hardware designed. First, the microphone signal is put through a low pass filter to get rid of the DC portion of the signal. Then, it is amplified using an operational amlifier and put through a high pass filter to attenuate frequencies away from the desired frequency of 5 kHz. The filter response is shown in Figure 2. The signal is run through a voltage follower before being amplified a final time. The output of the operational amplifier is put through a final low pass filter to get rid of the remaining DC offset. Finally, it is put through an envelope detection circuit. This uses a diode and a capacitor to essentially obtain an RMS value for the frequency which can describe relatively how loud the signal is. The closer to the microphone the 5 kHz source, the higher the amplitude of the signal, the greater the output of the envelope detection circuit. Figure 3 shows a picture of the circuit built. The output of the envelope detection circuit is what is passed into the XADC to be used by the processor.

Next, the XADC core was successfully integraed into the hardware system along with the seven segment display. The XADC allowed the output of the hardware circuit, the voltage of the FPGA, and the temperature of the FPGA to be read. Once these signals were available, code was used to develop the functionality of the system. The switches on the Nexys4 DDR board were used as input. If SW0 is on, then the system reads the voltage of the FPGA and displays it on the seven segment display. Figure 4 shows the output on the FPGA. If SW1 is on, then the system reads the temperature of the FPGA in celcius and displays it on the seven segment display. Figure 5 shows the output on the FPGA. If SW2 is on, then the system reads the input from the microphone cicuit and displays the voltage on the seven segment display. In addition to this, it lights up LEDs to visually indicate the magnitude of the signal. The closer the source to the microphone, the louder the signal, the more LEDs illuminated. Figure 6 shows the output on the FPGA.

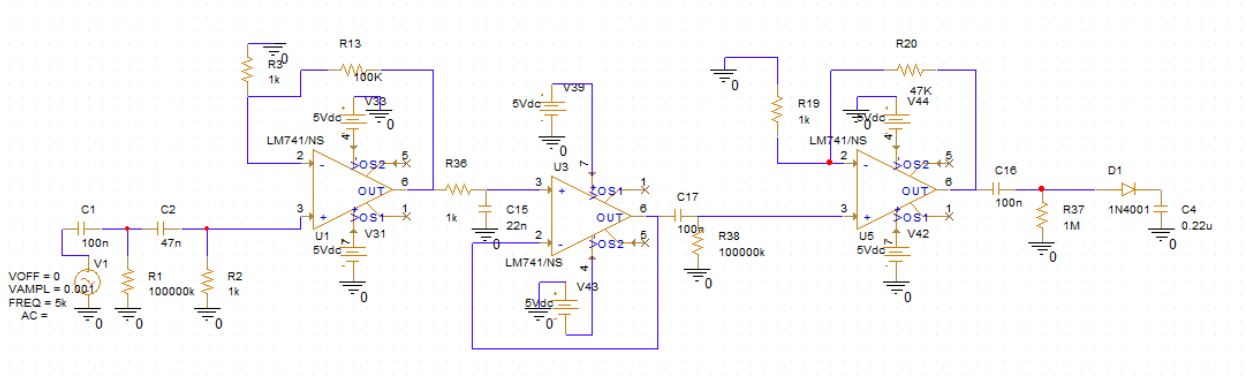

Figure 1: Microphone Circuit Schematic

A video of what the system does with the different switches is in the GitHub Repository linked in the summary. There is also a video demonstrating the system displaying the changing voltage from the circuit and illuminating LEDs to show the magnitude of the sound and how close it is to the microphone.

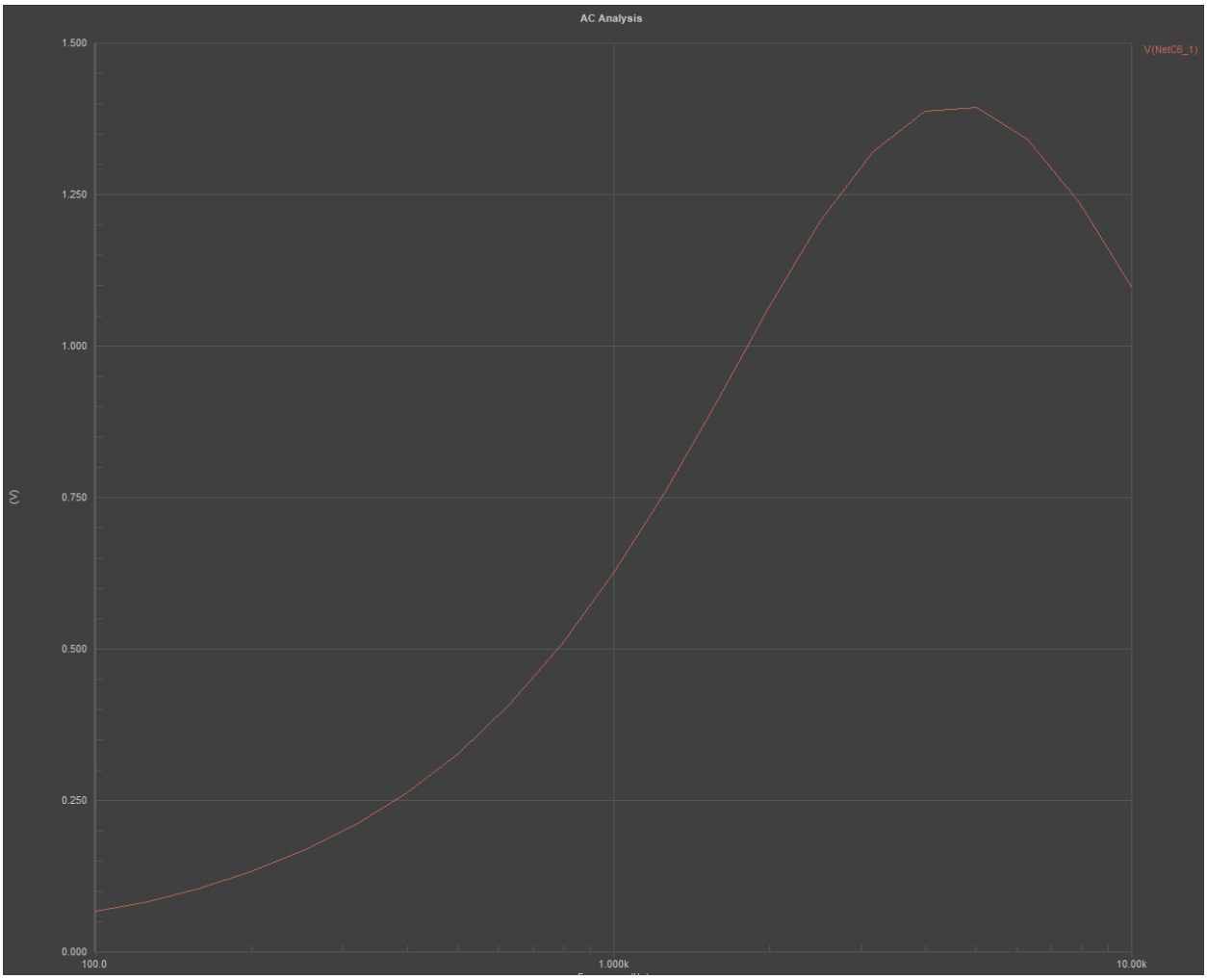

Figure 2: Filter Response

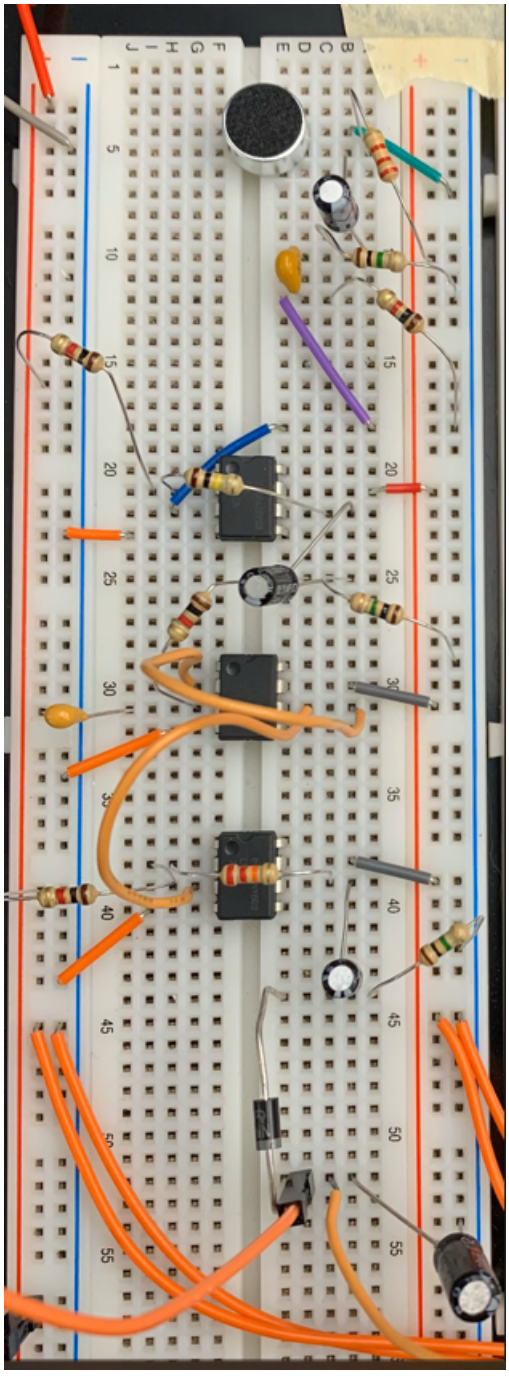

Figure 3: Built Circuit

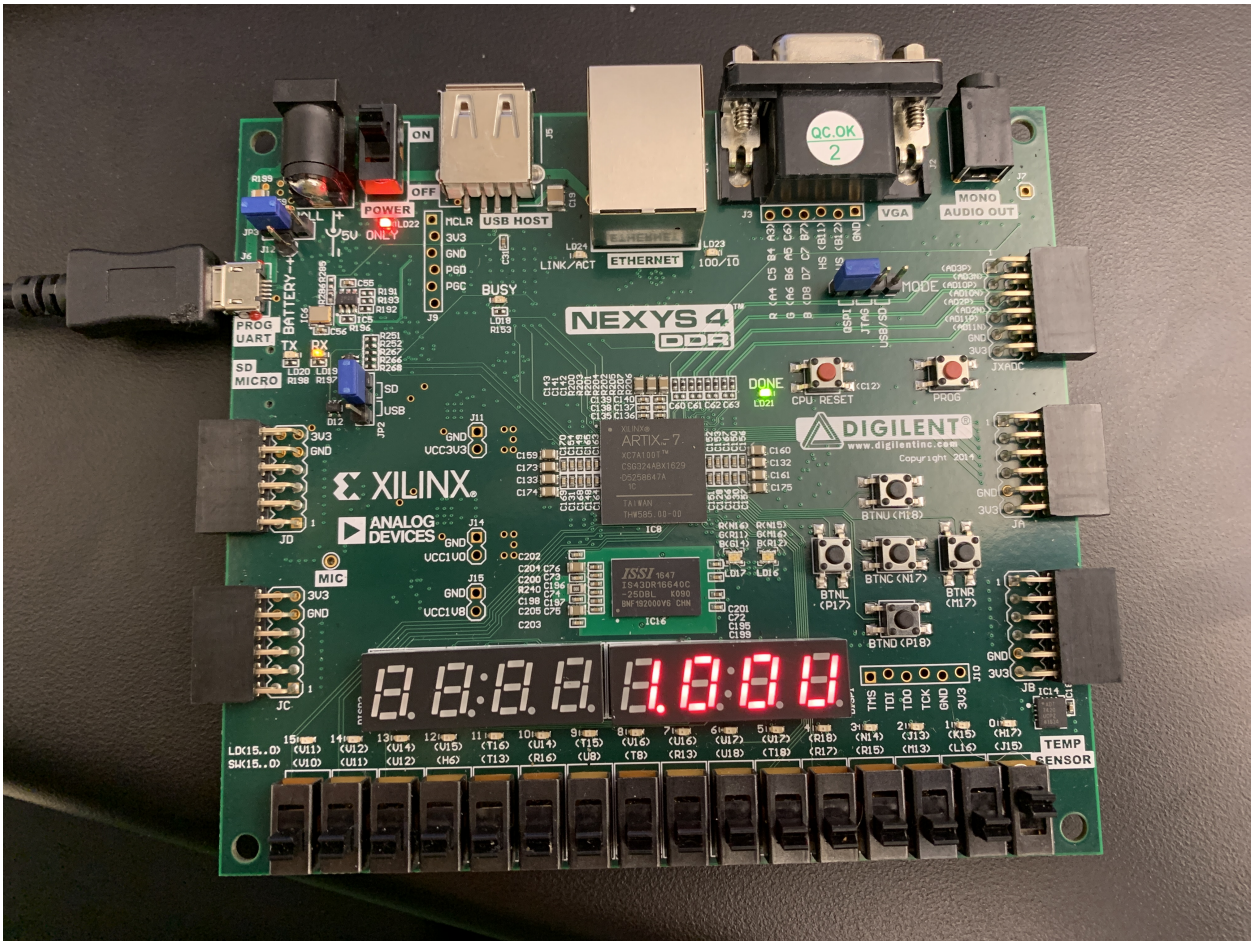

Figure 4: SW0 VCC Output

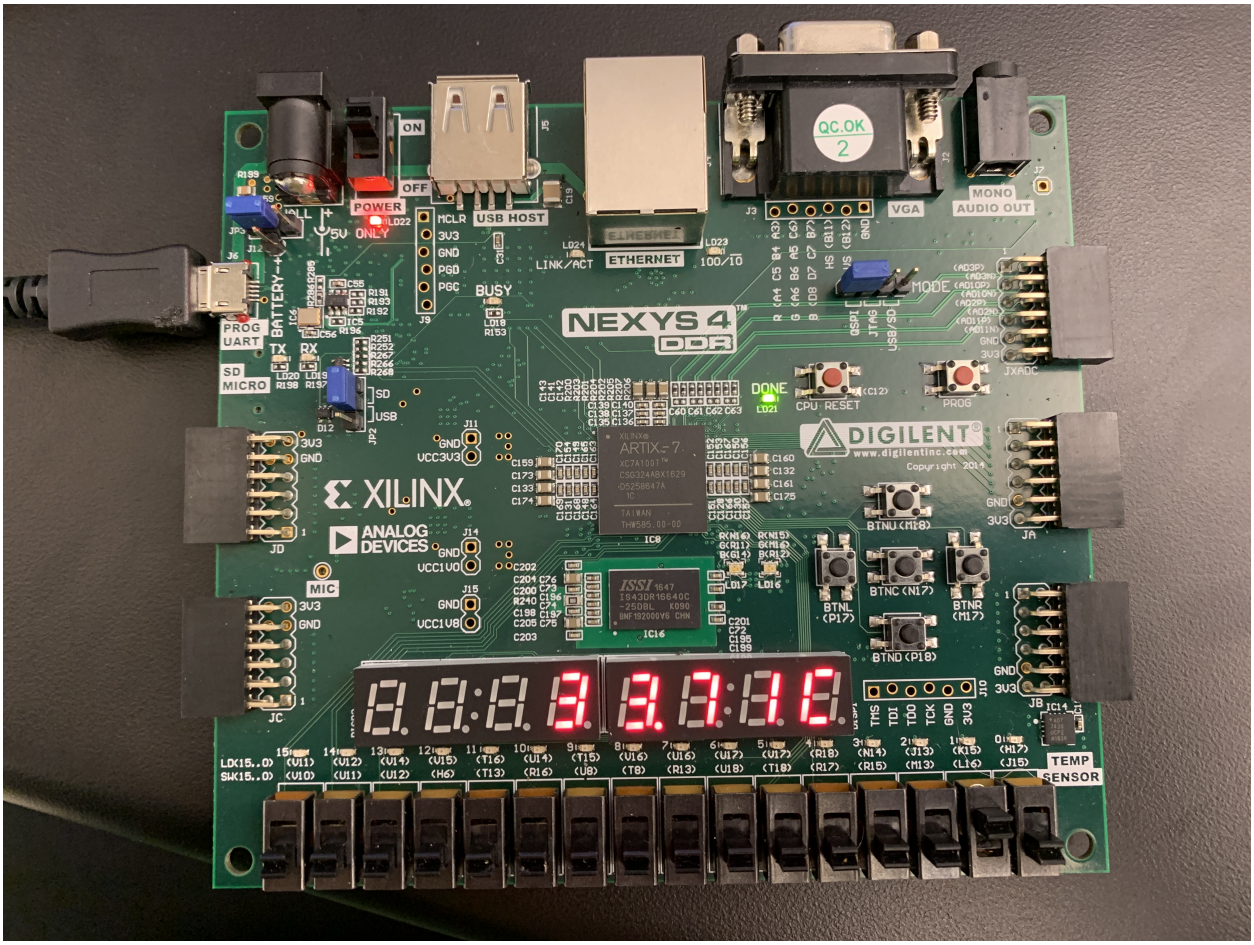

Figure 5: SW1 Temperature Output

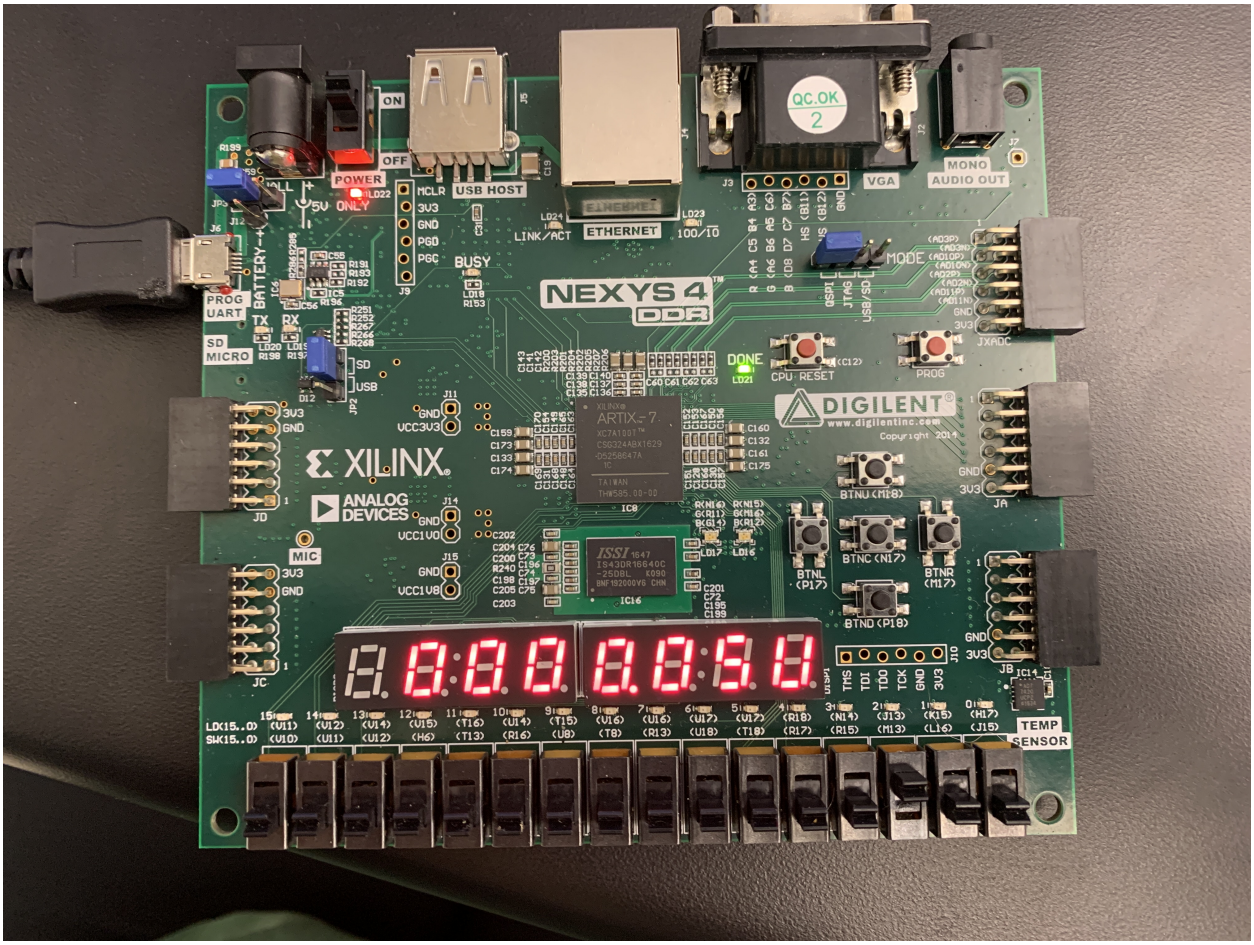

Figure 6: SW2 Microphone Output

Code

Listing 1: application function

```
void xadc_sseg(XadcCore* adc, SsegCore* sseg,GpoCore* led,int sw){
        double reading;
        int tens, ones, tenth, hundredth,hundreds, thousands;
        if(sw == 1){
                reading = adc->read_fpga_vcc();
                tens = int(reading/10.0);
                ones = int(reading-(tens*10.0));
                tenth = int((reading-(tens*10.0)-ones)*10.0);
                hundredth=int((reading-(tens*10.0)-ones-(tenth * 0.1))*100.0);
                sseg->write_1ptn(0xc1, 0); //V
                sseg->write_1ptn(int2sseg(hundredth), 1);
                sseg->write_1ptn(int2sseg(tenth), 2);
                sseg->write_1ptn(int2sseg(ones), 3);
                sseg->write_1ptn(0xff, 4);
            sseg->write_1ptn(0xff, 5);
                sseg->write_1ptn(0xff, 6);
                sseg->write_1ptn(0xff, 7);
                sseg->set_dp(8);
                led->write(0x0000);
        }
        else if(sw == 2){
                reading = adc->read_fpga_temp();
                tens = int(reading/10.0);
                ones = int(reading-(tens*10.0));
                tenth = int((reading-(tens*10.0)-ones)*10.0);
                hundredth=int((reading-(tens*10.0)-ones-(tenth * 0.1))*100.0);
                sseg->write_1ptn(0xc6 , 0);
                sseg->write_1ptn(int2sseg(hundredth), 1);
                sseg->write_1ptn(int2sseg(tenth), 2);
                sseg->write_1ptn(int2sseg(ones), 3);
                sseg->write_1ptn(int2sseg(tens), 4);
            sseg->write_1ptn(0xff, 5);
                sseg->write_1ptn(0xff, 6);
                sseg->write_1ptn(0xff, 7);
                sseg->set_dp(8);
                led->write(0x0000);
        }
        else if(sw == 3){
                reading = adc->read_adc_in(0);
                thousands = int(reading/1000.0);
                hundreds = int((reading-thousands*1000)/100.0);
                tens = int((reading-thousands*1000 - hundreds * 100)/10.0);
            ones = int(reading-(tens*10.0));
        tenth = int((reading-(tens*10.0)-ones)*10.0);
        hundredth=int((reading-(tens*10.0)-ones-(tenth * 0.1))*100.0);
```

```
            sseg->write_1ptn(0xc1, 0);
            sseg->write_1ptn(int2sseg(hundredth), 1);
            sseg->write_1ptn(int2sseg(tenth), 2);
            sseg->write_1ptn(int2sseg(ones), 3);
            sseg->write_1ptn(int2sseg(tens), 4);
            sseg->write_1ptn(int2sseg(hundreds), 5);
            sseg->write_1ptn(int2sseg(thousands), 6);
            sseg->write_1ptn(0xff, 7);
            sseg->set_dp(8);
            if(reading>0.9){
                    led->write(0xffff);
            }
            else if(reading>0.8){
                    led->write(0xffff);
            }
            else if(reading > 0.7){
                    led->write(0x3fff);
            }
            else if(reading > 0.6){
                    led->write(0x0fff);
            }
            else if(reading > 0.5){
                    led->write(0x03ff);
            }
            else if(reading > 0.4){
                    led->write(0x00ff);
            }
            else if(reading > 0.3){
                    led->write(0x003f);
            }
            else if (reading > 0.2){
                    led->write(0x000f);
            }
            else if (reading > 0.2){
                    led->write(0x0003);
            }
            else{
                    led->write(0x0000);
            }

    }
    else{
            sseg->write_1ptn(0xff, 0);
            sseg->write_1ptn(0xff, 1);
            sseg->write_1ptn(0xff, 2);
            sseg->write_1ptn(0xff, 3);
            sseg->write_1ptn(0xff, 4);
            sseg->write_1ptn(0xff, 5);
```

```
                sseg->write_1ptn(0xff, 6);
                sseg->write_1ptn(0xff, 7);
                sseg->set_dp(0);
        }
        sleep_ms(200);
}
```