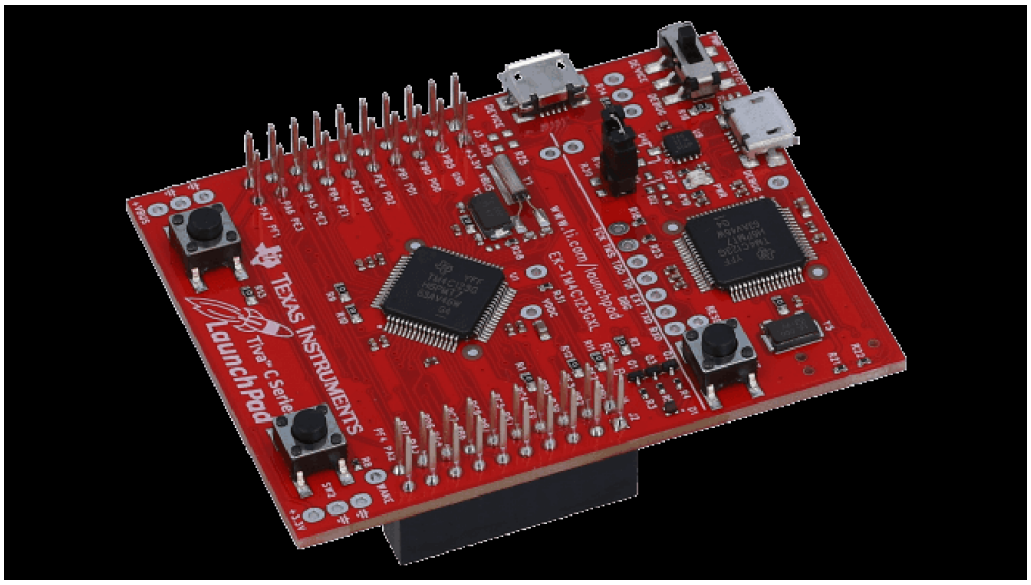
ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

EE447- INTRODUCTION TO MICROPROCESSORS LABORATORY PROJECT FINAL

LIGHT BASED LED DRIVER

1st Group Member: Barış GÜZEL - 2304764 - Screen & LED
2nd Group Member: Ahmet Fırat BEYDİLLİ - 2374619 - I2C & LED



INTRODUCTION & PROBLEM STATEMENT

In this project, we will be displaying the luminosity level measured by the light sensor on the LCD screen and light up the proper LEDs on the microcontroller board TM4C123GH6PM. For that purpose, there are two input devices and two output devices in this project. Inputs to the microcontroller board are luminosity sensor TSL2561 and potentiometer. Outputs are Nokia LCD screen, onboard RGB LEDs and output LED (external led). Among those devices, the first one to be inspected is the luminosity sensor.

Initial step in the usage of sensor is to establish the serial communication between the microcontroller and the luminosity sensor TSL2561. This operation will be done by using I2C. Sensor will sample the luminosity value in a 256 elements array and will send it to our microcontroller board. After attaining that input, we will be calculating the mean of the sent data and it will represent the average luminosity in every second. Required operations in the project will be done by using this average luminosity value. Second device to be inspected is the potentiometer. By using the potentiometer, the user will set two threshold values on the luminosity level. One of the thresholds is the lower threshold and the other is the higher threshold. In initial configuration, there will be some constant values (given by project members) and the user can later change those thresholds if he/she wants to. If the average luminosity value is below the lower threshold, the red LED in the microcontroller board will light up. If the value is between lower and higher threshold, the green LED in the microcontroller board will light up, and the last interval is above the higher threshold, in which the microcontroller board will light blue LED. Beyond all those, external output LED will light up based on the current luminosity value (independent of the on-board LEDs of the board). If luminosity is low, output LED will light up in a dim manner. As luminosity increases, LED brightness will increase as well.

Furthermore, the output LED will light up by using a transistor since GPIO pins of the microcontroller board will not produce sufficient current to feed the LED. Note also that threshold setting procedure can be also done by using 4x4 keypad. The last unit in this project is the Nokia LCD screen. In this screen, current luminosity value, the LED that is currently ON (in the microcontroller board) and the lower & higher threshold values will be displayed. Current luminosity value will be updated every second, even if other entities do not change in that 1 second interval. If other entities also change, those changes will also be illustrated in the LCD screen.

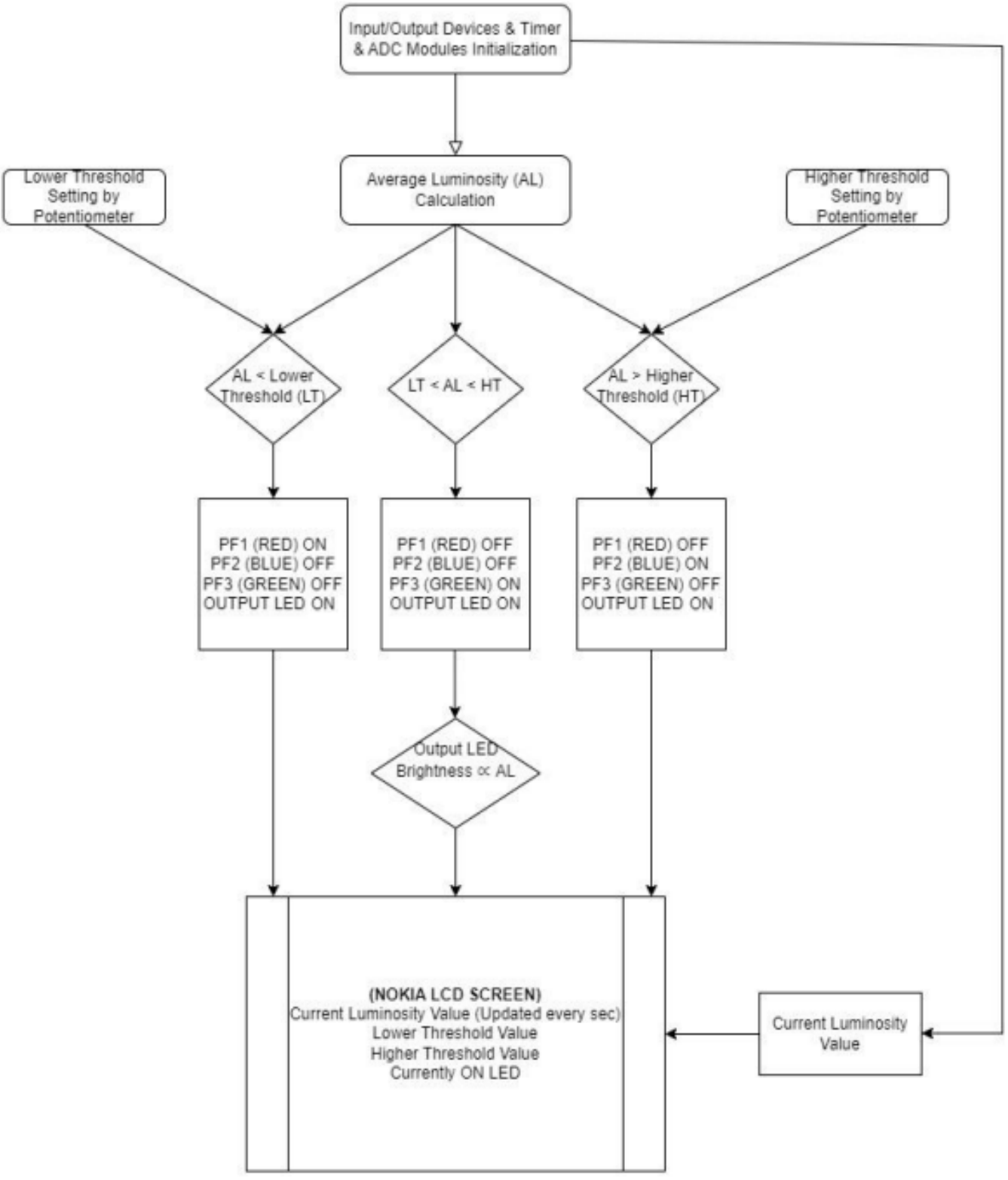


Figure 1: Approximate flow chart of the system.

SOLUTION PROCEDURE

Having talked about the problem statement, let us focus on the solution procedure. For that, let us inspect the units in the figure 2:

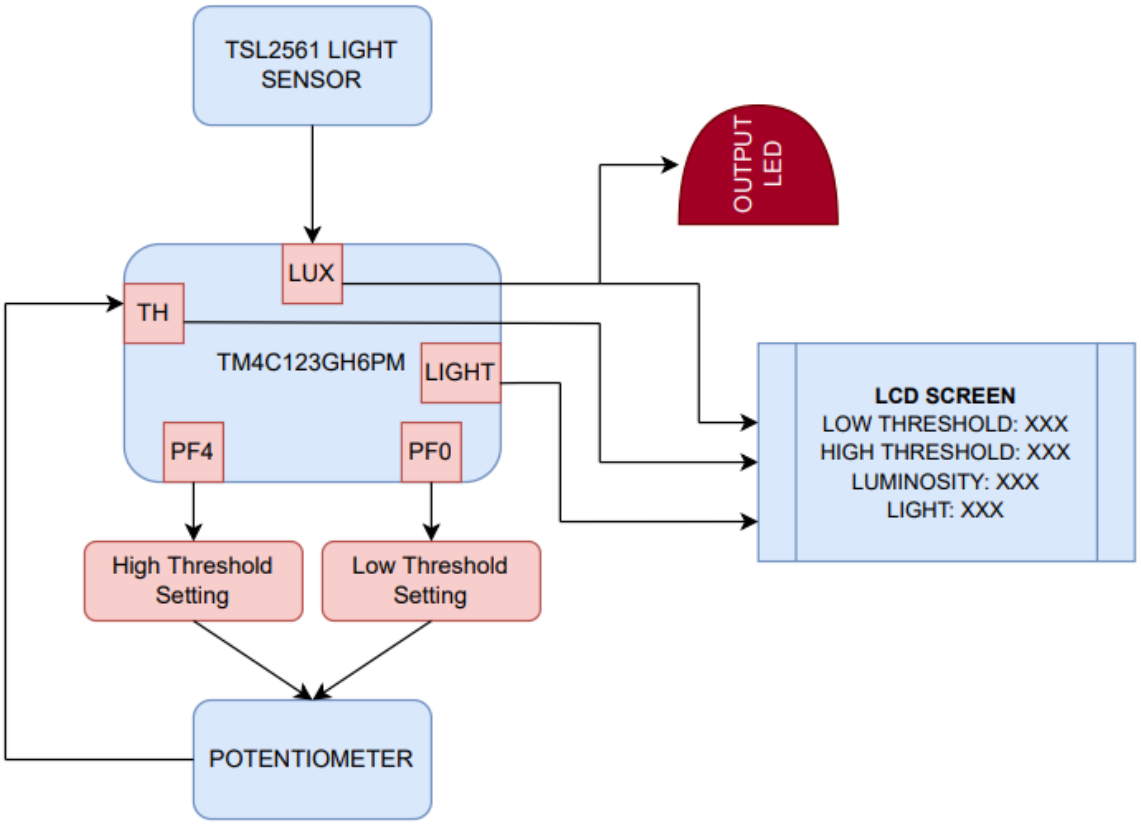


Figure 2: Solution Procedure Flowchart.

Our microprocessor TM4C123GH6PM has two inputs from light sensor and potentiometer. For the communication of light sensor and the board, I2C protocol is used (as mentioned before). To be able to do that, first, we have to configure the content of the light sensor TSL2561. Having read its datasheet, we have seen that register 0x0 and 0x1 of the sensor should be configured as follows:

Step 1:

$$Register\ 0\ =\ Control\ Register.\ Enables\ the\ powering\ -\ up\ of\ the\ device.$$

$$Write\ 0x03\ to\ Register\ 0\ to\ power\ -\ up\ the\ TSL2561.$$

Step 2:

$$Register\ 1\ =\ Timing\ register.\ Arrange\ the\ internal\ structure\ of\ the\ device.$$

$$Write\ 0x00\ to\ Register\ 1\ to\ obtain\ 13.7ms\ integration\ time.$$

Above steps are done by using the write function of I2C. One byte of information is written into the registers of the light sensor. After that, TSL2561 is ready to be used. 256 consecutive luminosity information is taken from the device by reading one byte at each time (I2C read function is used to read one byte information). The average of the 256 measurements is calculated afterwards. By doing so, we increase the reliability of the measurement. Note that the initial information taken from the device is raw. In other words, we have to perform compensation of the data to attain the real luminosity value in lux. To be able to do that, TSL2561 offers a built-in function to calculate the real value of the luminosity. That code segment is directly integrated into our device and the real lux value is obtained afterwards.

The other point to consider is the threshold value. As it is explained in the problem statement part, on-board LEDs of the device will be lit based on the interval that current luminosity stands with respect to low & high thresholds. Initially, it is decided that the low threshold value is 300 lux and high threshold value is 600 lux. However, users can change the thresholds by using the potentiometer provided. To be able to do that, the user has to first select which threshold that he/she is going to change. In order to ease the job, we used the on-board push buttons of the TM4C123GH6PM. As it can be seen from the figure 2, if the user presses the button PF0, low threshold value can be changed by using the potentiometer. If the user presses the button PF4, the high threshold value can be changed. This procedure is done by using the interrupt feature of the microcontroller. PF4 and PF0 buttons are pulled up initially (when a button is not pressed, their value is 1). If the user presses any of the buttons, he/she will cause falling edge transitions in the buttons (buttons will transition to 0) and it will create an interrupt which will be handled in the GPIOF_Handler function. Within that function, some flags will be set, and the potentiometer adjustment will change the low or high threshold values accordingly.

Having done that, we have completed the data transaction of two input units to our microcontroller. The other entity that we should save is “which on-board LED is ON at that moment”. After on-board LEDs are lit, information of the currently ON light is simply saved to a string and it will later be printed on the screen.

After all those, the information that should be conveyed to the screen is ready. As it is illustrated in the figure 2, the contents of the screen are:

- Low Threshold Value: If not changed by POT, default display is 300 lux. Updated at every change.
- High Threshold Value: If not changed by POT, default display is 600 lux. Updated at every change.
- Luminosity: Updated every second even if other entities do not change.
- Light: Currently ON light. Updated at every change.

Details of screen initialization is as follows:

To make the Nokia 5110 work, three steps are needed. First, the I/O pins on the GPIO must be set up to act as SPI pins. Next, the SPI module on the TM4C123 must be configured to work with the LCD. Finally, the Nokia 5110 display must be initialized to allow communication and to display data.

➔ GPIO and SPI Configuration

To configure the GPIO for the Nokia 5110, the clock for GPIO Port A has been enabled and the direction registers for PA6, and PA7 have been set as inputs. PA6 is used to configure the data formats between Data and Command, and PA7 is used for signal reset for operation of the LCD. Then we enabled alt functions for PA2, PA3, and PA5. PA2 is used as a pin clock (serial clock line), PA3 is used as a pin chip enable for controlling operation of pin controllers, and the last pin PA5 is used for pin data (serial data line). Note that for PCTL PA2, PA3, and PA5 are chosen as SSI whereas PA6, and PA7 are chosen as GPIO. After the GPIO initialization ended we started to set SPI configuration. The first step of this configuration started with clock initialization. We opened the SSI0 clock for our operations and we waited for the clock to stabilize. Then we continued the configuration with closing the SSI for setup setting then we chose master mode for our operation. After, we set the clock rate of the SPI Clock as follows.

$$Sysclk/(CPSDVSR * (1 + SCR)) = 80/(24 * (1 + 0)) = 3.3\,MHz$$

After the SPI clock setup we set SSI Serial Clock Rate, SSI Serial Clock Phase, SSI Serial Clock Polarity, SSI Frame Format Select, and SSI Data Size Select for 8 Bit data configuration respectively . As the final step for initialization we opened SSI after completing all setups according to the project manual given in the ODTUCLASS.

➔ NOKIA 5110 Configuration:

"To initialize the Nokia 5510 screen, we first toggled the Reset pin (PA7) by holding it low for 100ms and then setting it high. Then, we sent a series of commands to initialize the display. First, we set H=1 and V=0 to enable extended command mode and horizontal addressing, respectively. Next, we set the VOP to 0xBF, which was chosen from the range of 0x[B0-C0] because our screen was too dark. We set the temperature control value to 0x04, and the voltage bias value to 0x13. We then switched to basic command mode (H=0) and configured the display for normal display mode. Finally, we set the cursor to the start address using X=0 and Y=1 as the initial cursor position. Throughout this process, we checked the SSI busy bit after each step to ensure the commands were being properly received." Also It is important to note that we did not directly connect the backlight pin PA7 to 3.3V until the display is functioning correctly and displaying data.

Beyond all that, the final part of the project that we should consider is the output LED. This LED's brightness will depend on the current luminosity value. As luminosity increases, its brightness will also increase and vice versa. To be able to accomplish that, PWM module of the device is used. After a few tries, we decided that 1500 lux in the light sensor will create a 100% duty cycle for the LED, and the duty cycle will be scaled accordingly,

between [0,1500) lux values. For example, if 750 lux is measured at the moment, duty cycle is:

$$\frac{Present\ lux}{Maximum\ lux\ (decided\ by\ group\ members)} * 100\% = Duty\ Cycle$$

$$\frac{750\ lux}{1500\ lux} * 100\% = 50\%\ Duty\ Cycle$$

One of the GPIO pins is adjusted to provide this duty cycle according to the formula above. However, it was not sufficient to drive the LED by just using this PWM port. The reason behind it is that GPIO pins of the microcontroller board do not allow the flow of relatively high current. Because of that reason, we had to build up the following circuit:

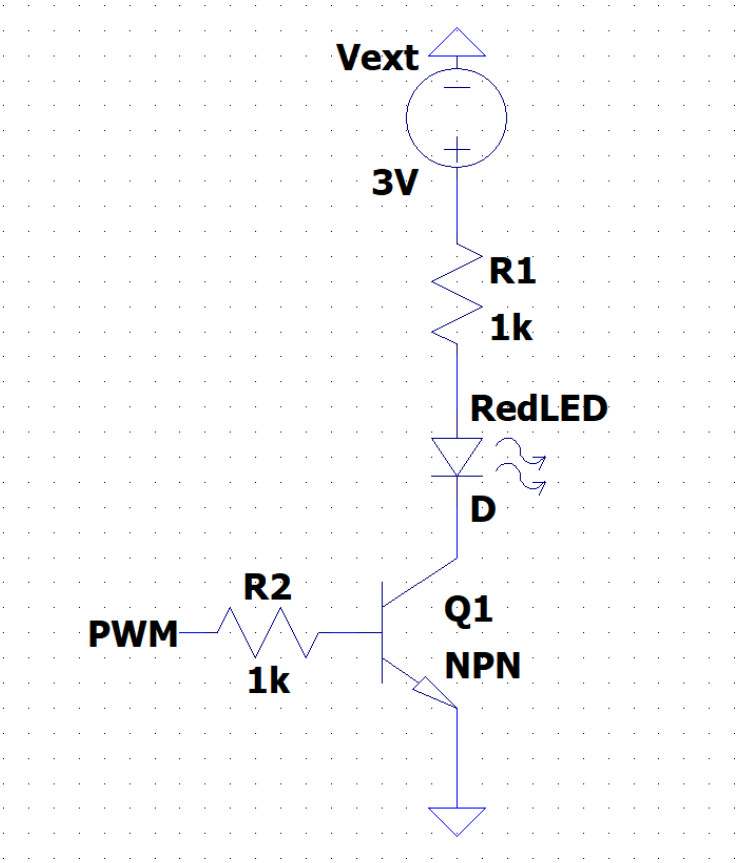


Figure 3: External circuit to drive the output led "RedLED".

Above circuit is a simple common emitter amplifier. Its collector is fed with an external supply, and it allows high current flows in our output LED, unlike the situation in GPIO pins. In our design, the PWM module coming from the GPIO pin is connected to the base of the npn transistor. As luminosity increases, the PWM duty cycle will increase and the voltage coming to the base will also increase (3.3V at 100% duty cycle, maximum). As base voltage increases, base current will also increase and will create even more current in the collector of the transistor. Since our LED resides at the collector, increasing current will cause

more brightness on our LED. Note that there are two 1kohm resistors in base and collector to prevent excessive current flow that can burn our output LED.

After all those steps, our design is ready to be used. The physical configuration of the device when it is working is as follows:

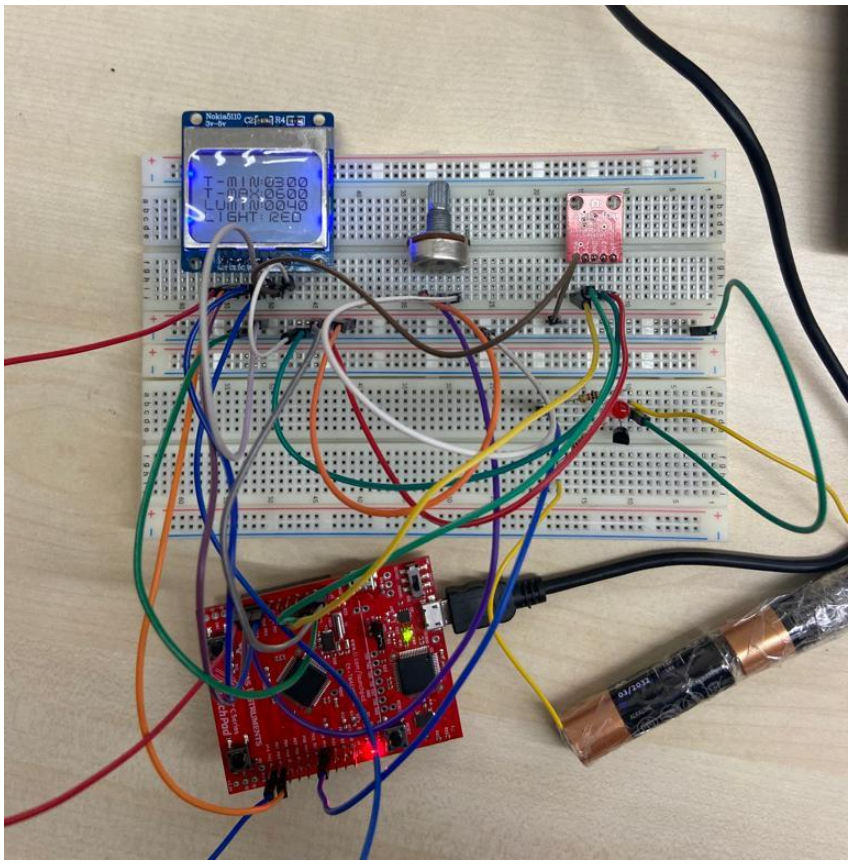


Figure 4: General configuration of the device.

DEVICE OPERATION IN DIFFERENT CASES

In this section, device operation will be shown for different case scenarios.

Scenario 1:

- Low Threshold = 300 (Default)
- High Threshold = 600 (Default)
- Luminosity = [0,300], lies below the low threshold
- Currently ON light = RED
- Output LED brightness = Not visible since luminosity is low.

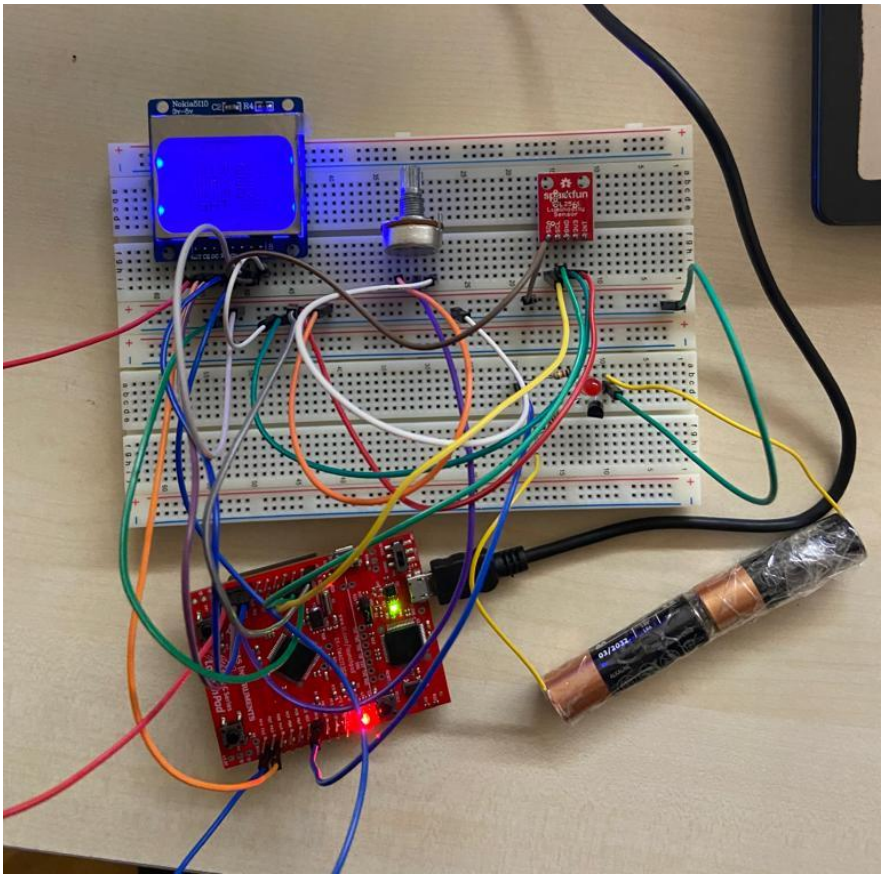


Figure 5: Device status for scenario 1.

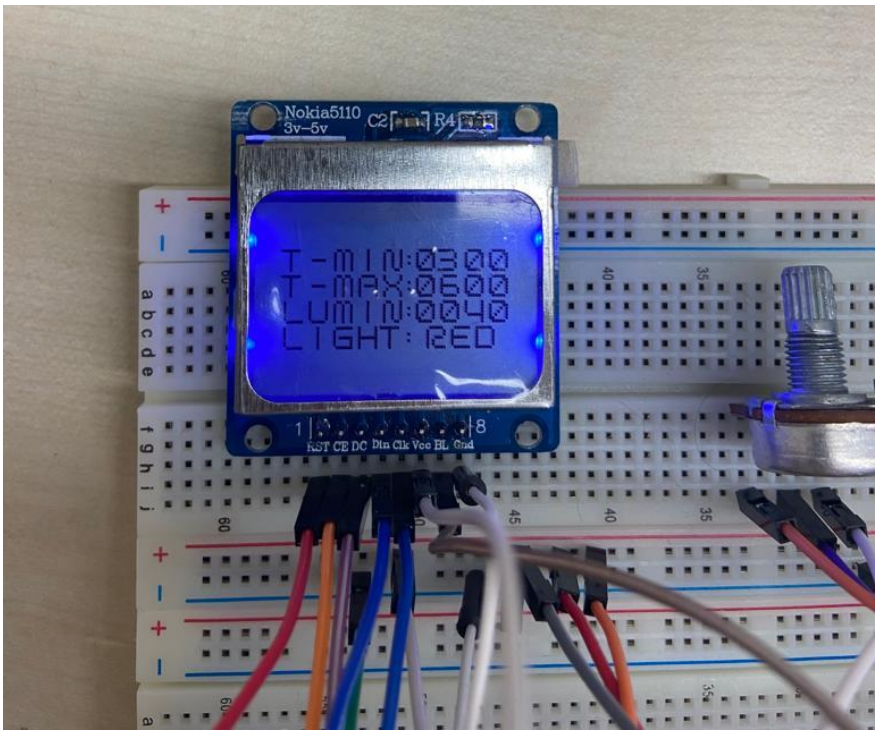


Figure 6: Screen status for scenario 1.

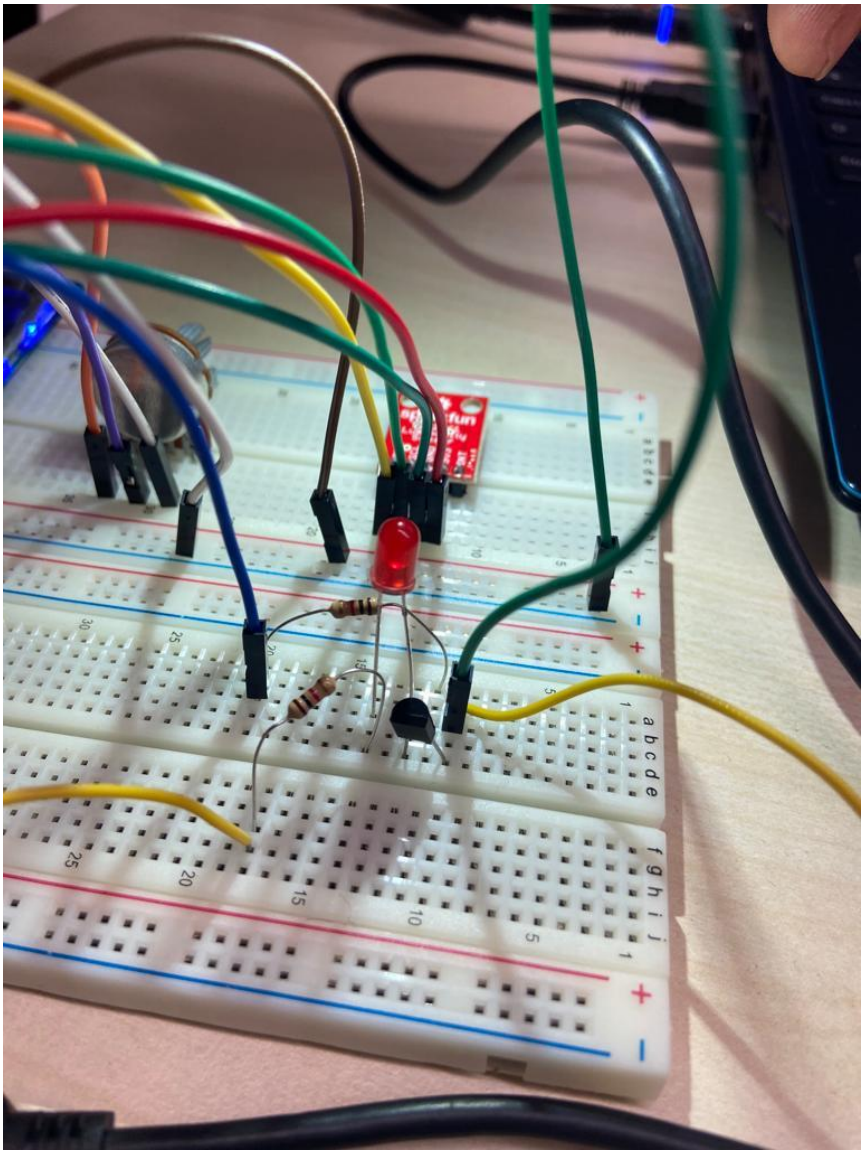


Figure 7: Output LED status for scenario 1.

Scenario 2:

- Low Threshold = 300 (Default)
- High Threshold = 600 (Default)
- Luminosity = [300,600], lies between low and high thresholds
- Currently ON light = GREEN
- Output LED brightness = Barely visible

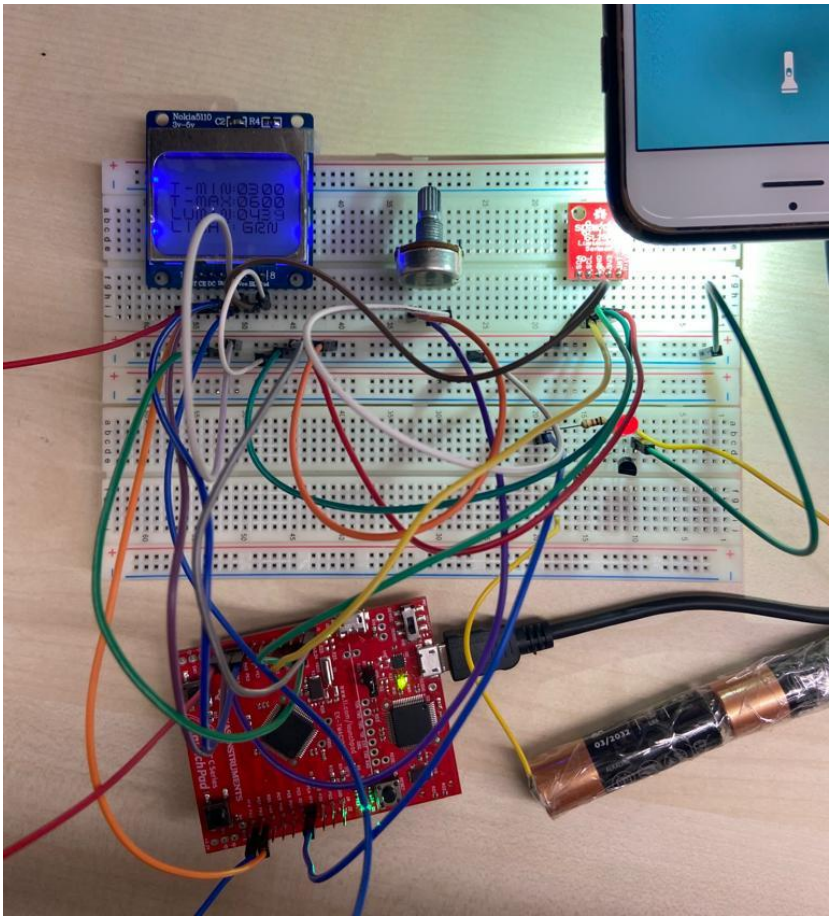


Figure 8: Device status for scenario 2.

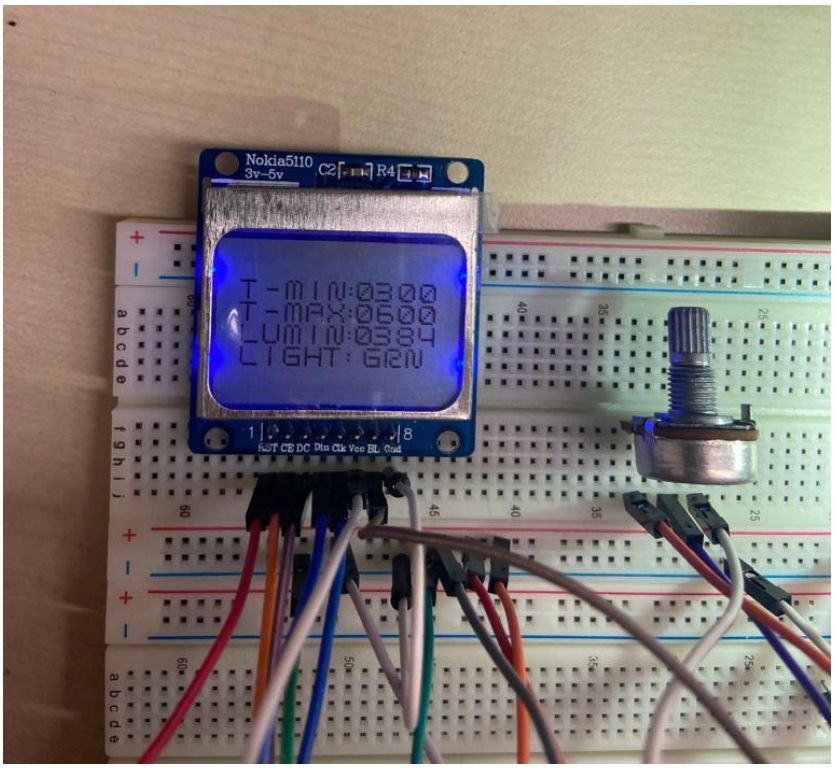


Figure 9: Screen status for scenario 2.

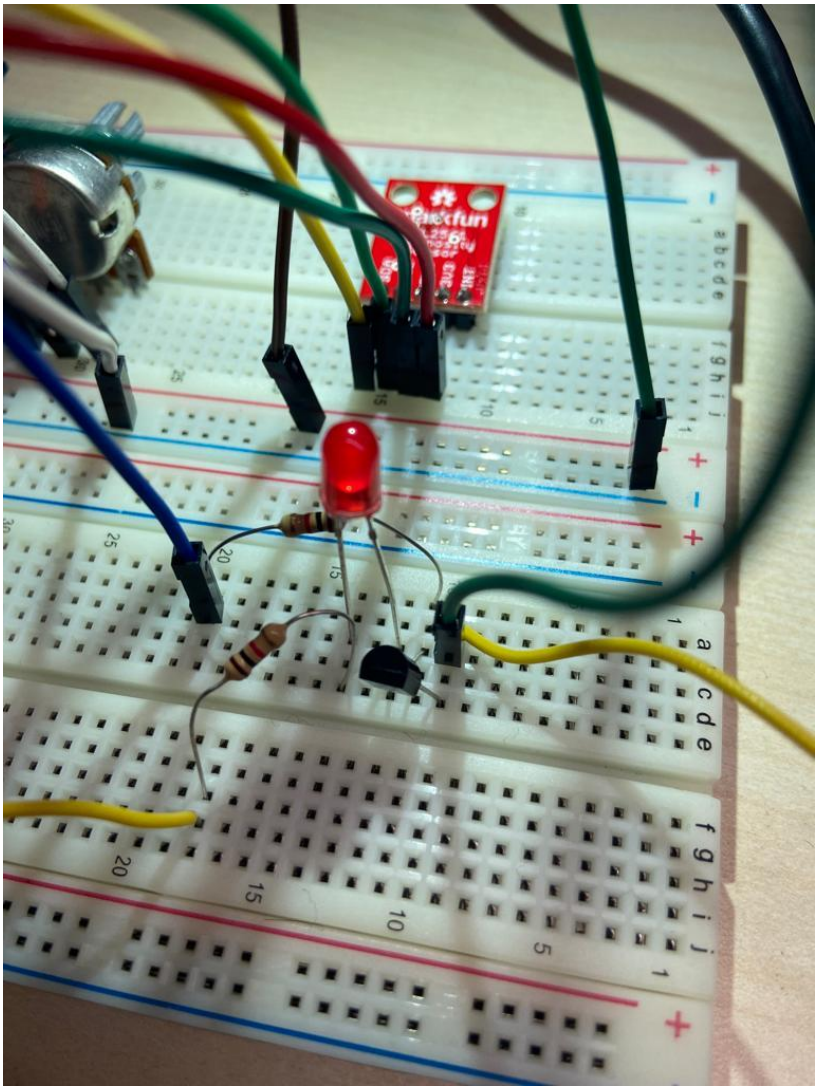


Figure 10: Output LED status for scenario 2.

Scenario 3:

- Low Threshold = 300 (Default)
- High Threshold = 600 (Default)
- Luminosity = 600+ , lies above high threshold
- Currently ON light = BLUE
- Output LED brightness = Visible & relatively high

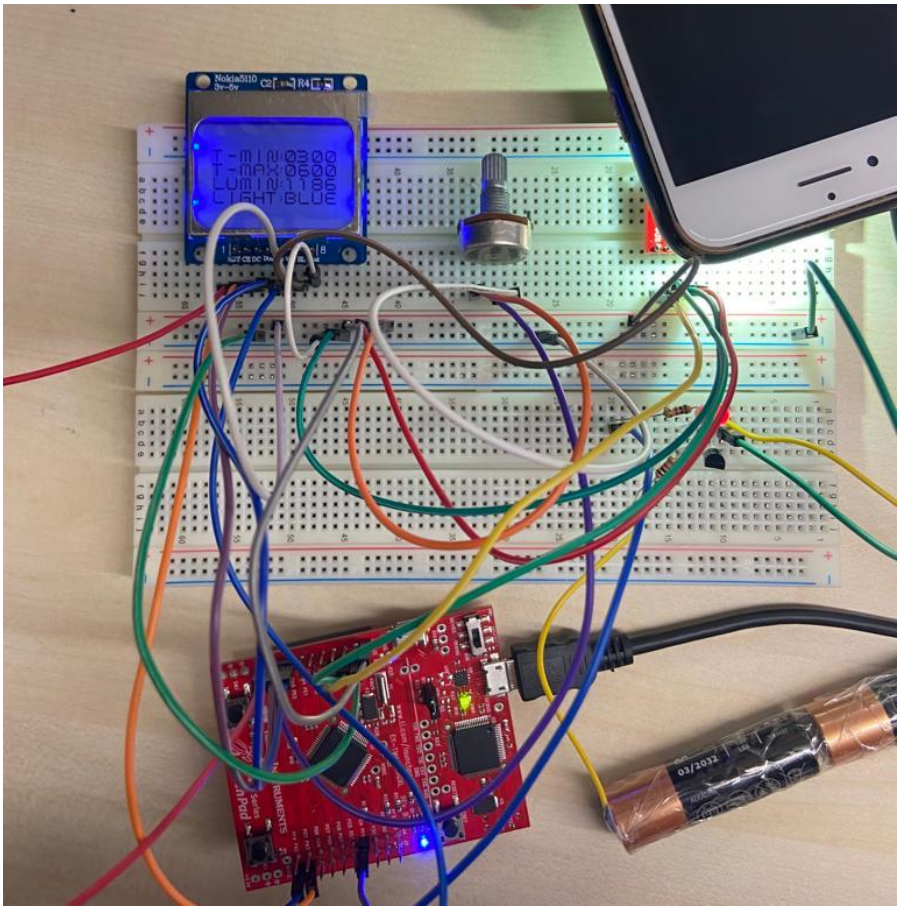


Figure 11: Device status for scenario 3.

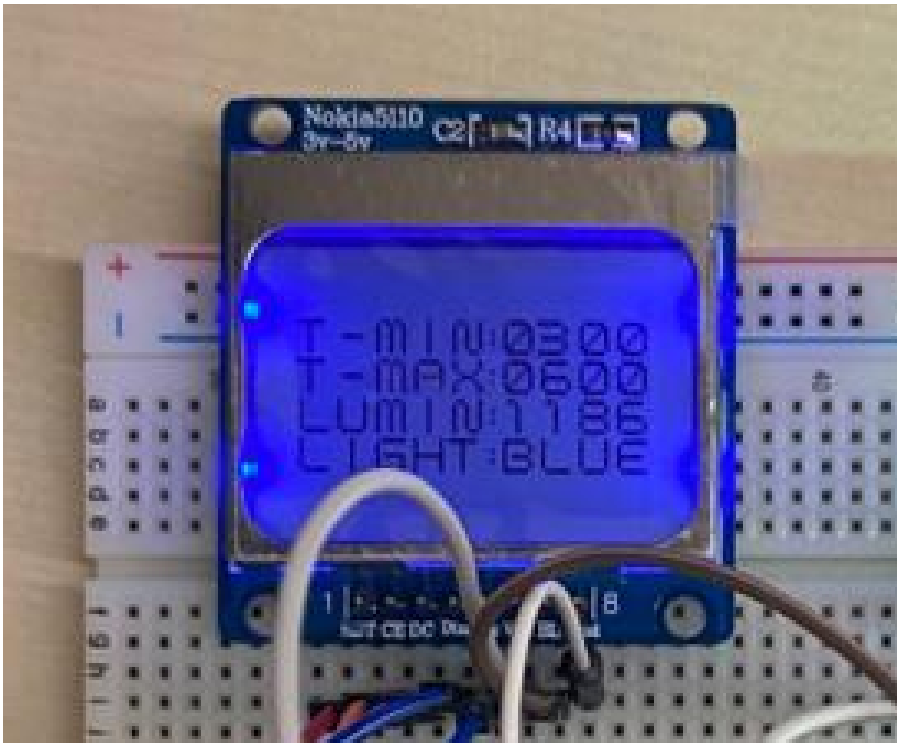


Figure 12: Screen status for scenario 3.

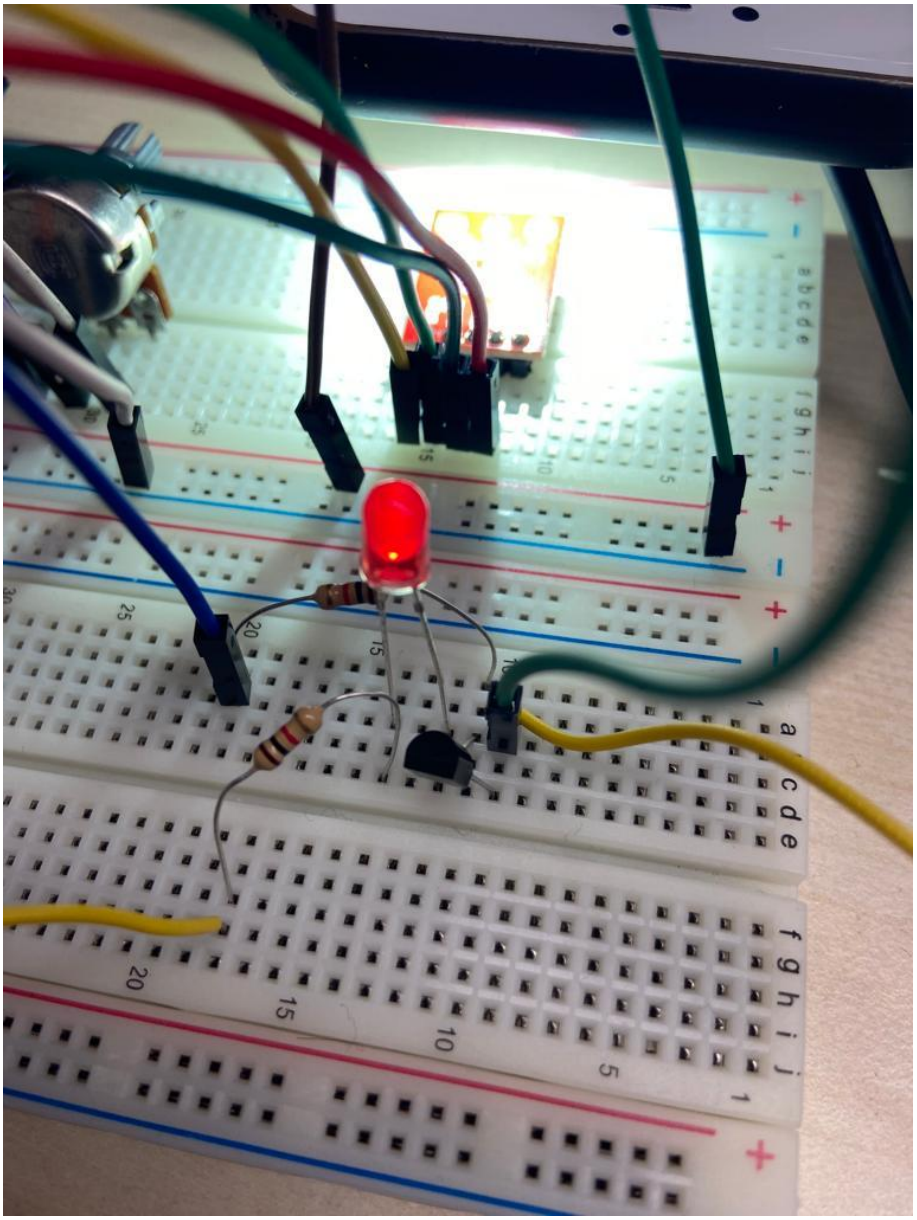


Figure 13: Output LED status for scenario 3.

Scenario 4:

- Low Threshold = 300 (Default)
- High Threshold = Changed by POT
- Luminosity = Can have any value
- Currently ON light = Can have any value

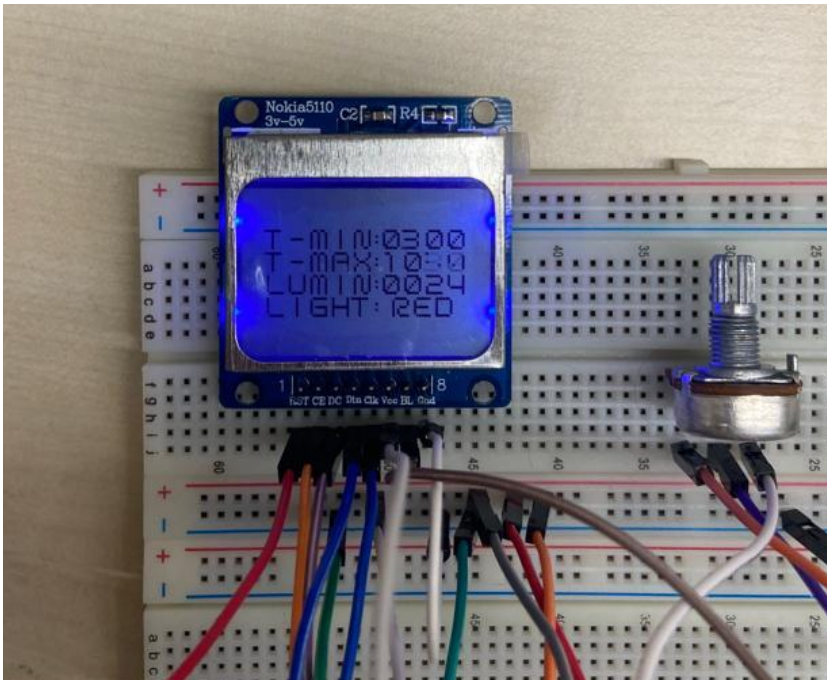


Figure 14: Screen status for scenario 4 (High threshold is changed).

Scenario 5:

- Low Threshold = Changed by POT
- High Threshold = Changed by POT
- Luminosity = Can have any value
- Currently ON light = Can have any value

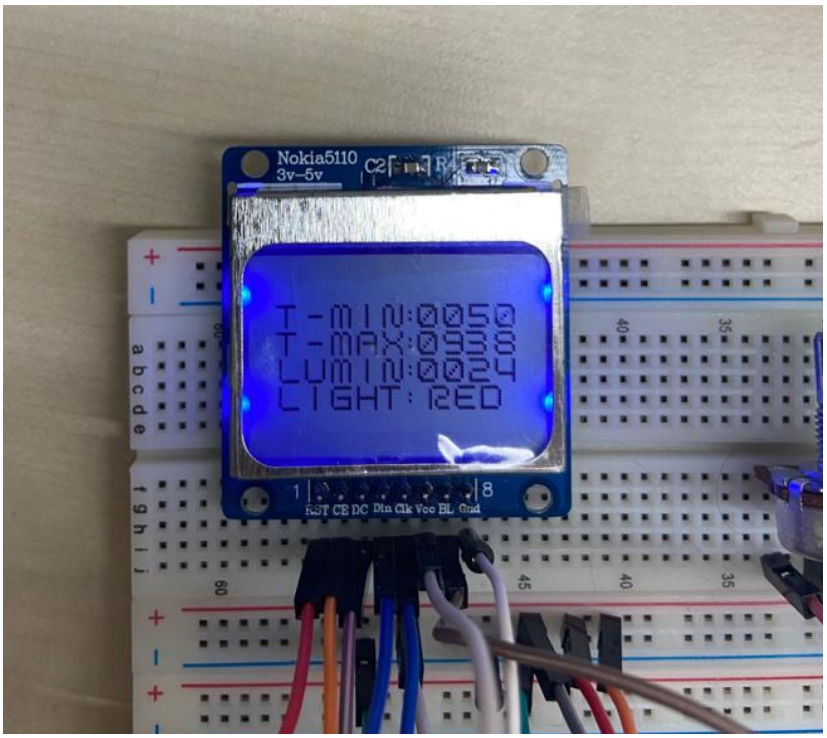


Figure 15: Screen status for scenario 5 (High & low thresholds are changed).

CONCLUSION

The Light Based LED Driver project was a comprehensive and challenging undertaking that required a strong foundation in electrical engineering and microcontroller programming. Through the use of a variety of techniques and concepts learned in EE447, including Polling, Interrupts, ADC, PWM, and SPI communication, we were able to effectively control the intensity of an LED based on ambient light levels using a sensor (TSL2561) and microcontroller with pulse width modulation. This system was able to provide a dynamic lighting experience that adapts to changing light conditions, enhancing its functionality and usefulness. In addition to the technical aspects of the project, we also gained a deeper understanding of the Nokia 5110 LCD and its configuration and use. This proved to be a valuable asset in the development process, as the LCD screen played a vital role in displaying relevant information to the user, including the current luminosity value, LED status, and threshold values. As with any project, we faced a number of challenges and had to employ various algorithms and approaches to overcome them. Also it is important to note the creation of large subroutines with the "LTORG" instruction. Through careful planning and execution, we were able to develop a system that functioned reliably and effectively. Overall, the Light Based LED Driver project has been a valuable learning experience that has enhanced our skills in electrical engineering and microcontroller programming. We are proud of the work we have done and are confident in the functionality and potential of our final design.

References

1. Tiva TM4C123GH6PM Microcontroller datasheet
2. Project manual of Light Based LED Driver
3. Datasheet - TSL2561.pdf
4. User_Manual_ET_LCD5110
5. Nokia5510 fonts library
6. https://microcontrollerslab.com
7. Github