ELECTRICAL & ELECTRONIC ENGINEERING
STELLENBOSCH UNIVERSITY

DESIGN (E) 314 Technical REPORT

Multi-Functional Light Source

Author:
[Sameer Shaboodien]

Student Number:
[25002783]

[22/05/23]

Plagiaatverklaring / Plagiarism Declaration

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.
2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.
3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.
4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Handtekening / *Signature*	Studentenommer / *Student number*
Sameer Shaboodien	25002783
Voorletters en van / *Initials and surname*	Datum / *Date*
	22/05/23

Abstract

This project follows the design and implementation of a complex lighting system to meet the module requirements of EDesign 314. The lighting system operates in a flashlight mode, emergency mode, and mood light mode. It serves as an introduction to low level integrated systems development. The report will include the design choices and system workings with respect to hardware and software and follow test methods that prove the systems functionality.

Contents

List of figures

List of Tables

List of Abbreviations

DAC – Digital to Analog convertor
V – Voltage
R – Resistance
I – current
KVL – Kirchoffs KVL - Kirchoffs Voltage Law
KCL – Kirchoffs
PWM – Pulse Width Modulator
MCU – Mictrocontroller unit – The STM32F303RE Nucleo board
GPIO _ General Purpose Input Output
TIC – Test Interface connection
DAC – Digital to analog converter
ADC – Analog to digital converter
LED – Light Emitting Diode
I2C - Inter-Integrated Circuit
HAL – Hardware Abstraction Layer
TIC – test interface connector
DMA – direct memory access
emf – electromotive force
RGB – Red Green BLue
UART – Universal Asynchronous receiver-transmitter

List of Symbols

m – milli

u - micro

Ω - Ohm

s – seconds

k – kilo

n – nano

V – volts

A - Amperes

1 Introduction

The Edesign 314 project required students to build a multi-input lighting system. The system was required to meet a set of user requirements (URs). The following is an outline of these requirements:

Table 1: User Requirements

UR1	The system shall generate its own regulated 5 V and 3.3 V supply voltage from a nominal 9-12 V battery or power supply.
UR2	During flashlight operation, the device shall turn the white LED on or off and change its intensity based on the input.
UR3	During emergency mode operation, the device shall transmit specific morse code or output a strobe light as well as maintain the functions of UR1. This shall require sub-modes and input.
UR4	During mood light operation, the device shall set the RGB light on or off and to specific intensities based on the input.
UR5	The device shall receive commands and shall transmit outputs via a UART connection. The device shall operate in an 8N2, even parity configuration at a data rate of 57600 baud.
UR6	The Azoteq trackpad shall be able to set either LED on or off and change the intensity and color of the white and RGB LED respectively.
UR7	The buttons shall be able to set either LED on or off as well as change between operational modes.
UR8	The device shall use four LEDs to indicate the active system

UR9	The white and RGB LEDs will be driven by 4x PWM signals and required circuitry. A DAC output must be included for monitoring

The system has 3 main modes

- Flashlight mode
- Emergency mode
- Mood light mode

2 System Description

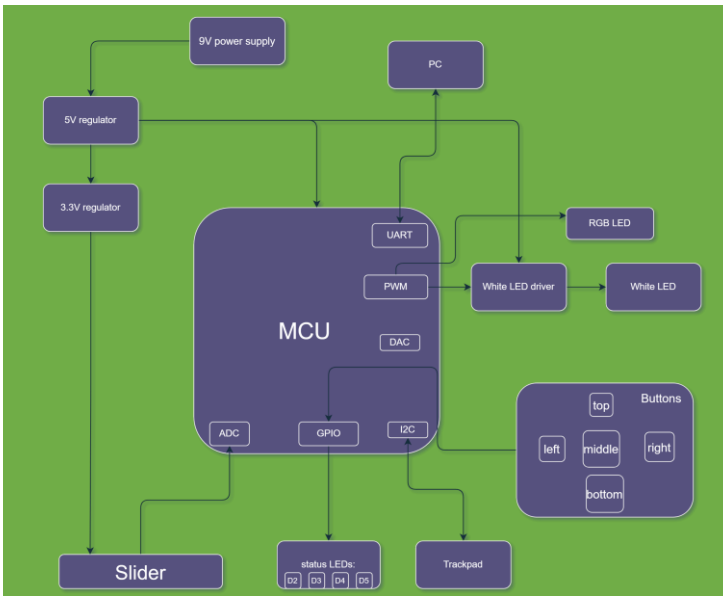

Figure 1: MCU topology

The logic diagram seen in figure 1 encompasses the topology of the hardware integration necessary to fulfill the user requirements of the lighting system. It consists of a main MCU to accomplish all the processing. This MCU alongside all other components are laid on a PCB board with copper traces to reduce soldering and clutter. The MCU has its GPIO, ADC, I²C, PWM, UART, and DAC peripherals configured. External to the MCU is a 9V supply, 5V regulator, 3.3V regulator, slider, status LEDs, push buttons, an Azoteq trackpad, white LED driver circuit, high power white LED, RGB LEDs, and a user-controlled PC used when debugging and running code. External to this diagram is a TIC where testing of system functionality occurs but it provides no added functionality to the system. This has connections to multiple components and would overcomplicate the diagram. The TIC connections are included per each component hardware description. The following hardware descriptions include the connections and hardware configurations of each sub-system.

3 Hardware Design Implementation

3.1 Power Supply:

The system is designed to input 9V from a power jack or battery. The system shall internally regulate this down to 5V and 3.3V. When using external power the MCU should be used in E5V mode. The outputs of these regulators are connected to specific PCB traces which allow easy access to the power rails at different points on the board.
The 9V supply was connected to the TIC connection J1 and J2.

5V regulator

The diagram specified in the PDD [1] is shown in figure 2. The capacitors act to improve transience and improve overall stability. The diode prevents any reverse emf. The LM7805CT is the core regulator and as per the datasheet [2] regulates an input of 9-12V down to 5V. LED D6 indicates the regulator is functioning. The trace connections made were J14, J16, and J25. Further, the 5V regulator was connected to TIC connection J4.

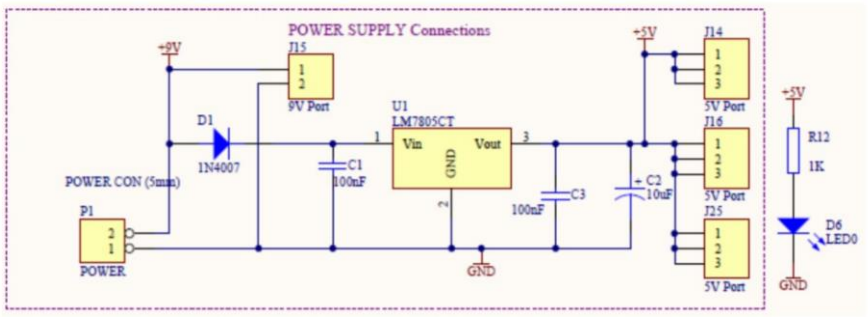

Figure 2: 5V regulator

3.3V regulator

The 3.3V regulator used the unidirectional current from the 5V regulator as its input, therefore no diodes were needed. The capacitors stabilize the voltages and improve the transience. The LM2950 regulator was used, with its datasheet in the reference section. The trace connections made were J17, J18, and J28. Further, the 3.3V regulator was connected to TIC connection J6. The circuitry of this regulator is visible in figure 3.

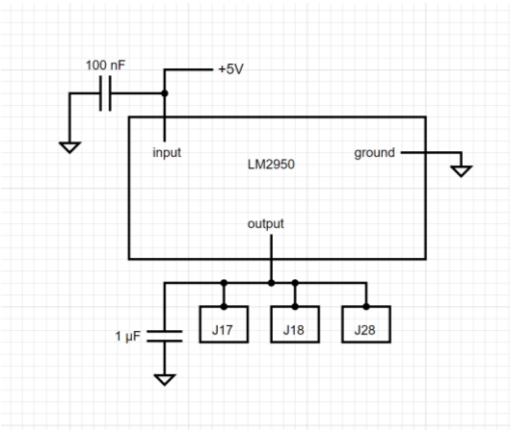

Figure 3: 3.3V regulator

TIC connections

J30 is shorted to allow the TIC to supply a 9V input.

3.2 Debug LEDs:

configuration

The 4 green debug LEDs were powered by pins on the MCU, configured as GPIO outputs. A resistor is put in series to the LED to prevent overloading the LEDs. The MCU pins were chosen based on availability and proximity to the debug LEDs. The pin connections and debug LED circuitry are in figure 4.

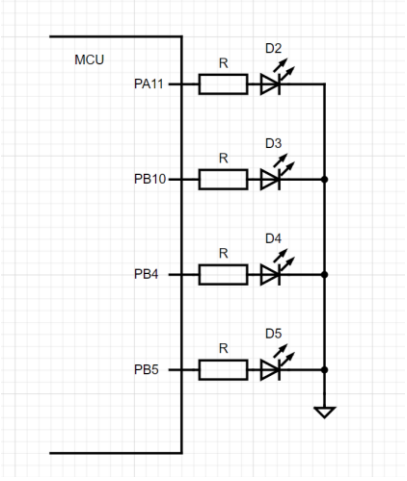

Figure 4: Debug LED circuitry

Each LED indicates a specific system state. The modes each LED indicate are visible in table 2 below.

Table 2: Debug LED modes

D2 on	Flash mode
D3 on and D5 off	Strobe mode (emergency mode)
D3 on and D5 on	Morse mode (emergency mode)
D4 on	Mood light mode

Resistor choice

The following calculations are based on any single LED – all 4 LED circuits are identical. According to the green LED datasheet, the LED has an optimal current of 20mA [3]. However, the GPIO pins on the STM board can output 8mA according to the STM32F303RE manual [4]. For general purposes, a current of 4mA will ensure safety to the board and sufficient brightness.
We can see from the forward voltage vs forward current graph of the green LED in figure 5, from its datasheet [3], that at 4mA, the LED has a forward voltage of approximately 1.9V.

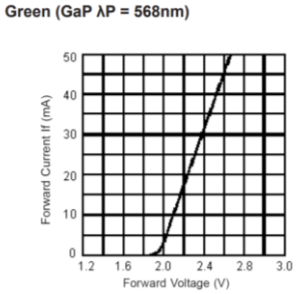

Figure 5: Green LED current vs voltage

Using KVL in the figure 4 circuit we find:
$-V_{MCU} + I*R + V_{LED} = 0$
→ $-3.3 + 4m*R + 1.9 = 0$
→ $R = 350\Omega$.

The closest standard resistor value of 330Ω was chosen which will result in a slightly higher, but still safe current of I = 4.2424mA by the same KVL equation.

TIC connections
LED D12 was connected to J9.

3.3 Push buttons

Configuration

The 5 push buttons are connected via GPIO pins configured in external interrupt mode with falling edge trigger detection and Direct Memory Access (DMA). The reason for this is software based.
When a button is unpressed, a pull-up or pull-down network is required to prevent the uncertain state of an open circuit being read by the pins. Hence, the 5 buttons are connected in an active-low, pull-up resistor network. Active low implies that when the button is pressed the pin voltage is pulled low (0V). Internal pull up resistors were implemented on the MCU to simplify circuitry and hardware debugging.
For simplicity, only the diagram of the middle button, connected to pin PA9, is shown in figure 6. Note that all the button circuits have the same configuration and resistor value, and the pins are simply different. The internal pull-up resistor is represented by R1 and the 3.3V source is internally supplied by the MCU.

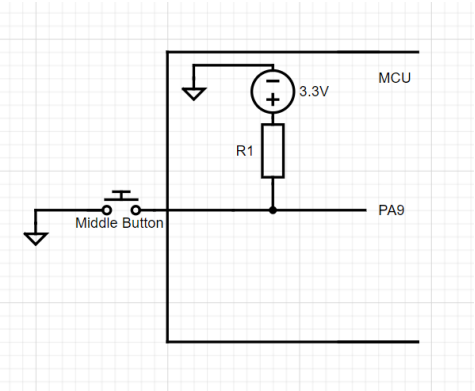

Figure 6: Button circuitry

Pull-up resistor choice

The MCU's internal pull up resistor has a minimum; typical; maximum possible value of 25;40;55 kΩ respectively. The GPIO pins can safely receive up to 8mA. Hence, a resistor value any margin greater than R1= V/I = 3.3V/(8mA) = 412.5 Ω is required to prevent overloading the pin. The above internal pull-up resistors are sufficient for this purpose.

The choice of pins was made based on availability and proximity from the buttons. Table 3 shows the pin and TIC connections for each button.

Table 3: Button Connections

Button	Pin	TIC connection
Middle	PA9	J13
Top	PA6	None
Bottom	PB9	None
Right	PA7	J14
Left	PB8	J11

3.4 ADC and slider

ADC Configuration

The ADC was connected to pin PA0 on the MCU and configured as ADC1_IN1 Single Ended with a 12-bit resolution. The input to this pin is the output voltage of the slider. The pin did not have an external current-limiting resistor. This is because the GPIO pins on the MCU have a 5kΩ internal resistor, according to the STM32 manual [4]. The ADC pin was configured with DMA to DMA1 Channel 1 with a half-word data width to ensure the ADC has enough size to store an accurate 12-bit value. Further, the DMA was configured to sample every 19.5 clock cycles to reduce noise susceptibility. The ADC was connected to TIC connection J8.

Slider

The slider is a potentiometer which has an output voltage dependent on the slider position. The output voltage is read by the ADC to ultimately control the output white LED brightness when changing the slider position. The slider schematic is shown in figure 7. 'Vin' represents the 3.3V input voltage. 'Vout' represents the output voltage which is connected to the ADC pin PA0 and the TIC connection J8. The maximum voltage occurs when the slider is at the left most position.

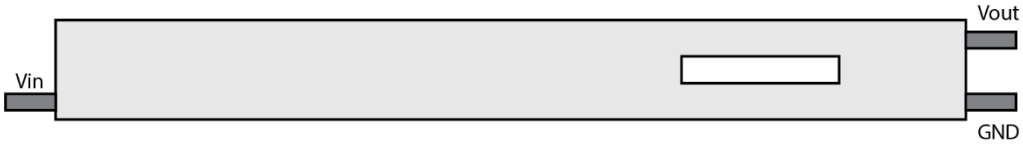

Figure 7: Slider Schematic

3.5 DAC and PWM

DAC

The DAC has 12-bits of resolution and therefore outputs a constant analog voltage from 0-3.3V based on an input digital value from 0-4095 specified in software. The DAC was outputted at MCU pin PA4, configured as DAC1_OUT1 with Output Buffer enabled. The DAC output was connected to the TIC at pin J3. The DAC was not used for any functionality in the system.

PWM

PWM outputs a series of varying width pulses averaging to a desired voltage. The PWM output requires a digital input value, PWM_VALUE, from 0-1000. Whatever this value is set to, expressed as a percentage of 1000, will be the percentage time that the pin will output 3.3V per period. For all other time in that period, it will output 0V. In other words, the on time is modulated instead of the amplitude which is constant. Since the period is so short, when this is outputted to the LED, our eyes cannot detect the flickering and hence we interpret the brightness as the mean voltage over the period. The mean voltage equals 3.3V * PWM_VALUE / 1000.

In the system there are 4 PWM output pins – each with the same configuration. The prescaler value is 720, the counter is in up mode, and the Auto Reload Register is 999. The Auto Reload Register was chosen to be 999 so that any value from 0 to 999+1 can be chosen as the amount of on time per period. This ensures a precise output. The prescaler was set to 720 due to the MCU clock frequency being 72MHz. The clock frequency is divided by this prescaler to ensure a less noise-susceptible clock frequency for PWM.
The first PWM pin is dedicated to the white LED output and is connected at pin PB11. It is configured as timer 2 channel 4. The white LED PWM pin was connected to TIC connection J10.
The other 3 PWM outputs are dedicated to the RGB led. In table 4 the configurations thereof are visible. They were not connected to any TIC connections.

Table 4: RGB LED configuration

RGB led color	MCU pin connection	Timer	Channel
Red	PC8	3	3
Green	PC6	3	1
Blue	PA12	4	2

All 4 pins were chosen first based on availability and proximity to the respective LED. Then, the timers and channels were chosen based on whichever timer and channel that pin could use. If the pin could not output the correct PWM channel and timer, the pin was rechosen based on the above criteria.

Motivation for configurations

Filters and an operational amplifier were not included in the system. These circuitry components allow for a low-noise, low flickering, high precision output. In the system we designed, the LEDs, being simple components, do not require such high-quality circuitry. It is sufficient to drive these systems via simply supplying them with the correct amount of power and controlling them with PWM.

3.6 Output LEDs

White LED

The high-power white LED is controlled by PWM and is configured as specified under PWM hardware. The PWM output and hence the LED may be influenced by the slider, buttons, UART, and the touchpad. The LED is driven by a current amplifier with a higher voltage capability to achieve its desired power.

Current Amplifier

The high-power white LED can input a maximum current of 35mA according to its datasheet [5]. The LED is therefore designed to operate at 30mA for a safety margin while maintaining its brightness capabilities. The forward voltage of the white LED at a forward current of 30mA is 3.6V as seen in figure 8 from its datasheet [5].

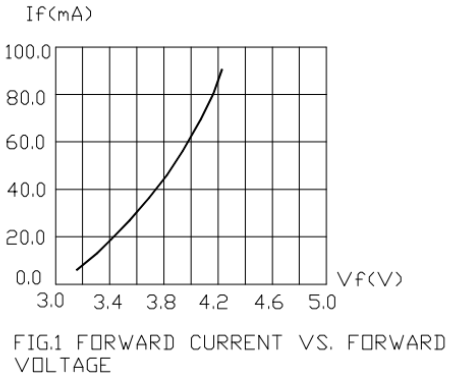

Figure 8: White LED current vs voltage

Since the GPIO pins on the MCU can output 3.3V at 8mA an external driving circuit for the white LED is required. The simplest transistor driving circuit was chosen and is visible in figure 9. The analysis to find the resistor values for the driving circuit is as follows:

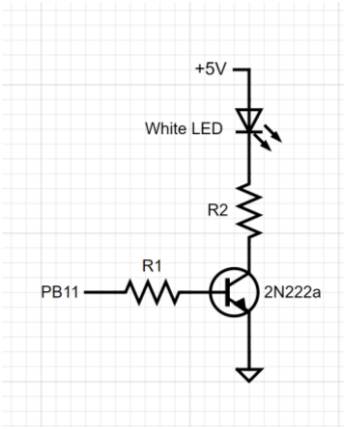

Figure 9: Current amplifier circuitry

The voltage at pin PB11 is either 0V or 3.3V as PWM is used.

According to the 2n222a datasheet [6], $V_{BE(ON)}$ = 0.7V and $V_{CE(SAT)}$= 0.3V

When PB11 = 0V:

V_{BE} = 0V → The transistor is off, and an open circuit exists in the place of the transistor. Thus, the LED is off.

When PB11 = 3.3V:

V_{BE} = $V_{BE(ON)}$ = 0.7V → The transistor is on.

The transistor is chosen to be driven in saturation as this is strategically done for amplifier circuits.

The LED current, and hence I_C was chosen to be 30mA

According to the 2N222a datasheet [6], B_{FORCED}, in saturation equals approximately 100.

KVL:

Equation 1 – base emitter loop:

$$-V_{\text{PB11}} + I_{\text{B}} \cdot R_1 + V_{\text{BE(ON)}} = 0, \quad I_{\text{B}} = \frac{I_{\text{C}}}{B_{\text{FORCED}}}$$

$$-3.3 + R_1 \cdot \frac{30m}{100} + 0.7 = 0$$

Equation 2 – collector emitter loop:

$$-5 + V_{\text{LED}} + I_{\text{C}} \cdot R_2 + V_{\text{CE(SAT)}} = 0$$

$$-5 + 3.5 + 30m \cdot R_2 + 0.2 = 0$$

Solving the equations, the resistors are found to be:

R1 = 8.67 kΩ

R2 = 43.33 Ω

These are the resistors required for the desired current through the white LED. The closest standard values of R1 = 8.2kΩ and R2 = 47 Ω such as to also deliver slightly less than 30mA instead of more which could cause an overcurrent.

RGB LED

The RGB leds are driven by the PWM pins and are configured as specified under PWM hardware. The RGB LEDs can receive a current larger than what the GPIO pins can supply. However, the LED is not designed to be high power and hence a maximum current equal to half of the max GPIO pin output current is sufficient. This value is 4mA. At 4mA the forward voltage of the red, green, blue LEDs are 1.7,2.85, 2.75V respectively according to figure 10 from the rgb LED datasheet [7]. Using Ohm's law, R= V/I, and the RGB circuit specified in figure 11 the following equation can be derived:

$$R = \frac{V_{\text{PWM\_MAX}} - V_{\text{LED}}}{I_{\text{LED\_MAX}}}$$

$$where\ V_{\text{PWM\_MAX}} = 3.3\text{V}\ and\ I_{\text{LED\_MAX}} = 4\text{mA}.$$

Note that some values are set to their maximum as this ensures the resistor is designed such that the maximum current is 4mA. Solving this equation for all 3 LEDs it is found that the resistances are 400;112.5; 137.5 ohms respectively for the R;G;B LEDs.

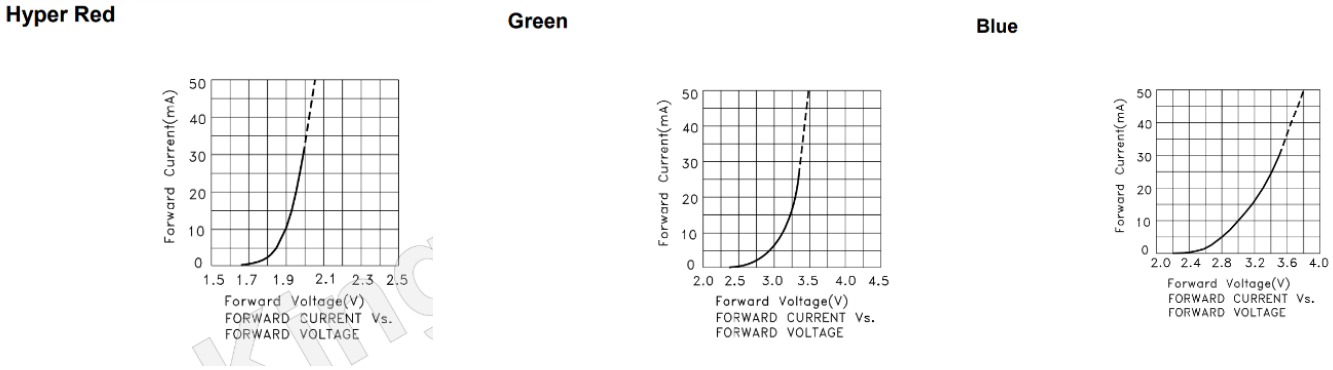

Figure 10: RGB LED current vs voltage

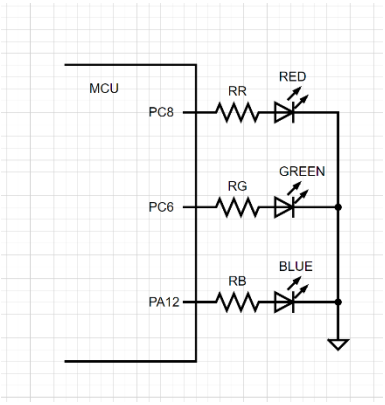

Figure 11: RGB LED circuitry

3.7 Azoteq Trackpad

The Azoteq Trackpad is a capacitive touch device which allows a range of gestures. The trackpad operates at a voltage of V_{DD} = 3.3V. The full resolution of the trackpad is 1792 x 768 in accordance to its datasheet [8]. The trackpad works via I^2C communication configured in fast mode with a speed frequency of 400kHz. DMA is configured to the I^2C receive channel to avoid slowing down the program when reading the touchpad. The following explanations can be aided by figure 12 and contain information pertaining to the Azoteq trackpad datasheet [8].

The 'RDY' or ready pin oscillates at 10Hz when the device is operational. This was connected to pin PB12 due to proximity and availability. PB12 was configured as an external interrupt with falling edge trigger detection to detect the oscillations.

The 'MCLR' or master clear is used to reset the trackpad at startup. It is pulled low and then back to its default high state to do this. This pin is connected to pin PB14 on the MCU due to proximity and availability. PB14 was configured as a GPIO output, with GPIO output level set as high.

The 'SDA' or serial data pin is connected to pin PB7. This was chosen because PB7 is a dedicated SDA pin.

The 'SCL' or serial clock pin is connected to pin PB6. This was chosen because PB6 is a dedicated SCL pin.

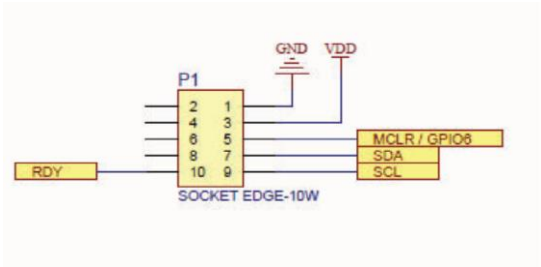

Figure 12: Trackpad schematic

The gestures for the trackpad include swiping, tapping and a press & hold. These are encompassed in the registers shown in Table 5 from the trackpad datasheet [8]. By reading from these registers, we can access the information on the user's action regarding the trackpad.

Table 5: trackpad gesture registers

Table A.14: Gesture Enable (0x80)

Gesture Enable														
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved										Swipe Y-	Swipe Y+	Swipe X+	Swipe X-	Press and Hold	Single Tap

4 Software Implementation

4.1 System state logic

The software implementation includes all the processing done to any peripheral inputs as well as any UART messages to achieve the required system functionality. The below software diagram, figure 13, encompasses the logical flow of the code on a higher level and its interactions with each peripheral.

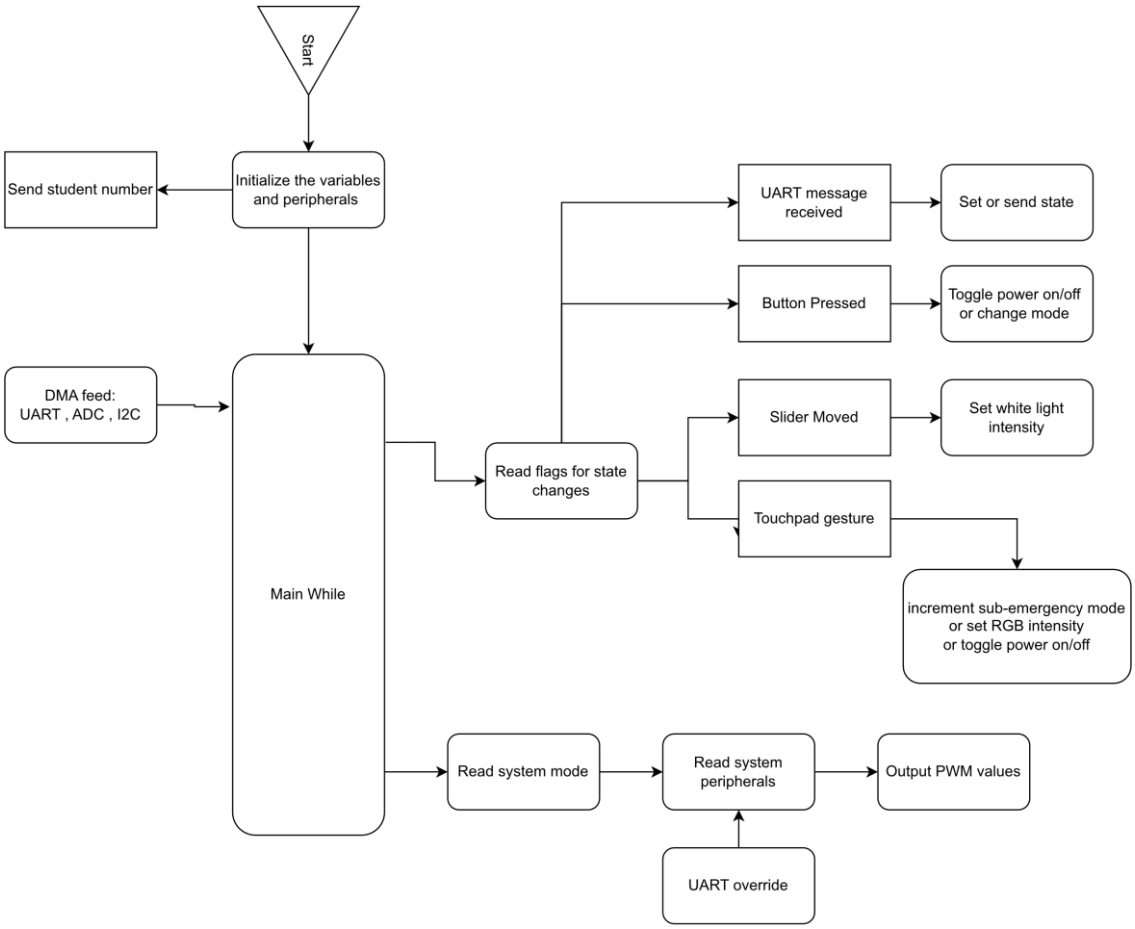

Figure 13: Software logic diagram

4.2 ADC and Slider

Calibration

Although our ADC is meant to read the output voltage of the slider perfectly, in reality a non-linear error exists between the slider output voltage (V_{S0}) and the average read in ADC digital value scaled to a 3.3V voltage (V_{RI}). Ideally, an equation should be present which takes V_{RI} and scales it to the actual slider output voltage. Mathematically an individual scaling factor equal to V_{S0} / V_{RI} can be used to set a single V_{RI} to its corresponding slider output voltage as follows:

$$V_{RI} \times \text{scaling factor} = V_{RI} \times \left(\frac{V_{S0}}{V_{RI}}\right) = V_{S0}$$

If this is done for many points, a scaling equation is produced which approximates the scaling factor for all possible voltages V_{RI}. The derivation thereof is visible under the ADC measurements section.

The scaling or calibration equation was found to be:

$$V_{ADC\_CALIBRATED} = V_{ADC\_READ} \times \left(1.19 + 0.0562 \times V_{ADC\_READ} + \frac{4.16 \times V_{ADC\_READ}^2}{1000}\right)$$

Where V_{ADC_READ} is the same as V_{RI}.

3.3V limiter

Due to the calibration equation, some values may exceed 3.3V. This will cause unexpected behavior. Hence, any value exceeding 3.3V is set to 3.3V. The logic of how the slider eventually leads to a change in the intensity of the white LED is summarized in figure 14.

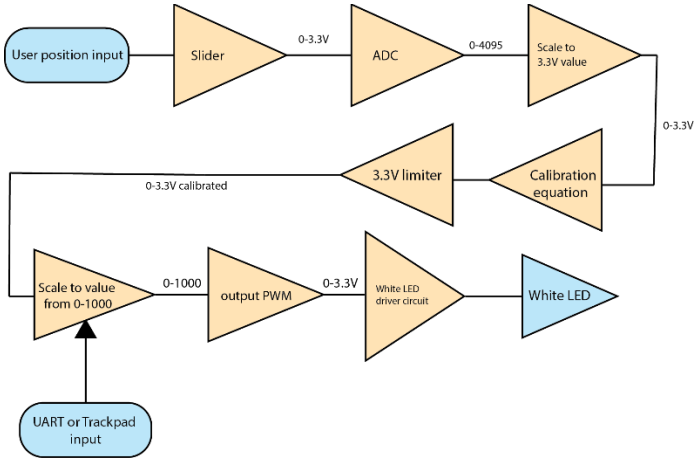

Figure 14: ADC input and output flow diagram

4.3 DAC

The DAC value is received from the ADC calibrated value and scaled to a value from 0-4095. This DAC value is then outputted and connected to the TIC but is not used for any other purposes. The DAC does not need any calibration as seen in measurements.

4.4 PWM

PWM receives a value from 0-1000 and sets the equivalent voltage to that value as a percentage of 1000 multiplied by 3.3V. There are 4 PWM channels where one controls the white LED and the other 3 control the individual RGB channels of the RGB LED. The white LED PWM channel is controlled by a combination of the UART input, the scaled ADC value which relates to the slider position, and any user trackpad slides as well as the buttons and trackpad press & hold gesture to control whether it is on or off. The RGB channels are influenced by UART, and trackpad taps as well as the buttons and trackpad press & hold gesture to control whether they are on or off. The inputs and outputs of the 4 PWM channels are summarized visually in figure 15. The PWM does not need any calibration as seen in measurements.

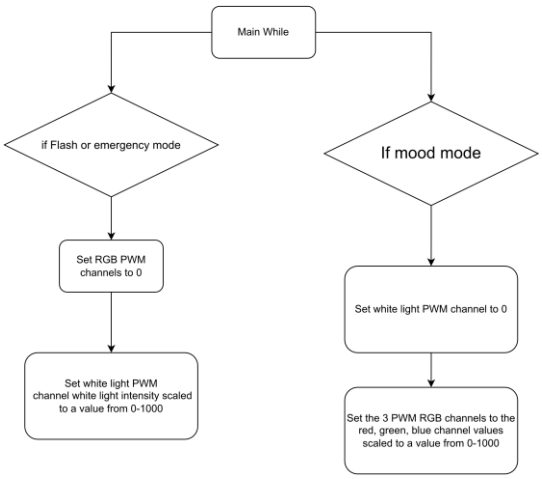

Figure 15: PWM logic diagram

4.5 Trackpad

I²C is used to communicate between the MCU and the trackpad.

Before any reading can be done from the trackpad it needs to be reset. This is done by pulling the master clear low for 20ms and then high again which will clear and reset the trackpad registers. Following this, the trackpad must be initialized for effective communication. This is done by implementing specific settings via the trackpad registers.

Reading is done from specific registers in the trackpad as specified in the hardware trackpad section. These store x, y information as well as tap, tap & hold, and swipe gestures. This allows us to set flags when a gesture occurs and simultaneously process the x and y information. Note that the x and y resolutions were set to 1692 x 674 when considering where the user pressed in mood light mode. This ensures the user can set the LED channels to the maximum values without having to struggle.

Because of the size of the reading and initializing code it has been put in a separate linked c file to keep the code modular. In the main loop the reset and initialization code is called and in the main while loop the trackpad is repeatedly read and flags are set if the trackpad is touched. These flags cause specific state changes to the system. The specific effect of each gesture is visually represented in figure 16. Note that if any of the conditions are untrue and their paths are unspecified, it implies that the trackpad read is complete. This was done for simplicity of the diagram.

Figure 16: Trackpad logic diagram

4.6 Push buttons debounce

The push buttons are spring operated and hence are prone to oscillating after a press. This causes extra false presses and needs to be suppressed. One way is hardware debouncing, however this is a risky and complex approach. It is simpler to implement software debouncing. The implementation causes a 20 millisecond wait, based on testing, after a press to ignore any post-press oscillations. This wait is short enough that the user will not notice a delay. This uses the MCU's internal timer, in this case the SysTick Timer on the MCU. Figure 17 shows the logic of the button debounce visually. Note that all the buttons have the same debounce code

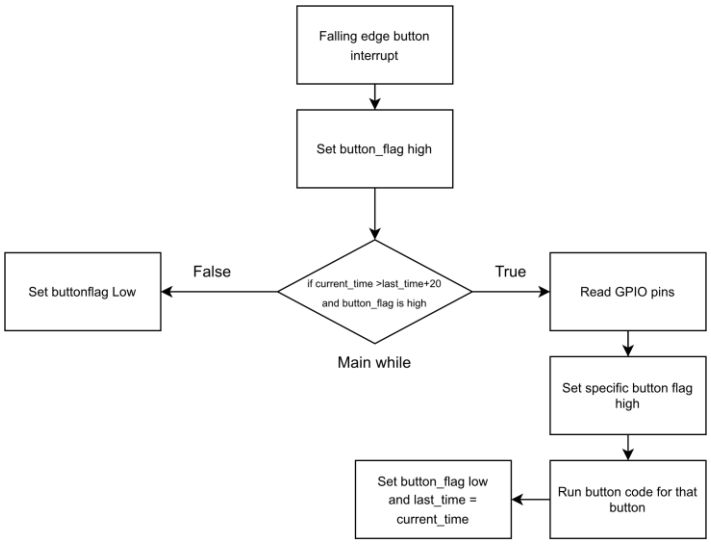

Figure 17: Push buttons logic diagram

4.7 UART

A requirement of the system was to enable communication via UART protocol. UART is a bidirectional communication system which allows digital communication between a user and the system. The UART was setup with 8 data bits, 2 stop bits, 1 start bit, and 1 even parity bit. Further, it was configured with a baud rate of 115200 bits/second. The system utilized DMA to not cause the code to be slowed down. Further, the DMA controller reads the input every 19.5 cycles instead of the default 1.5 to reduce noise applied to the system when receiving communication.

The UART communication is visually stipulated in figure 18, from ScienceDirect [9], showing the transmission of character 'A'.

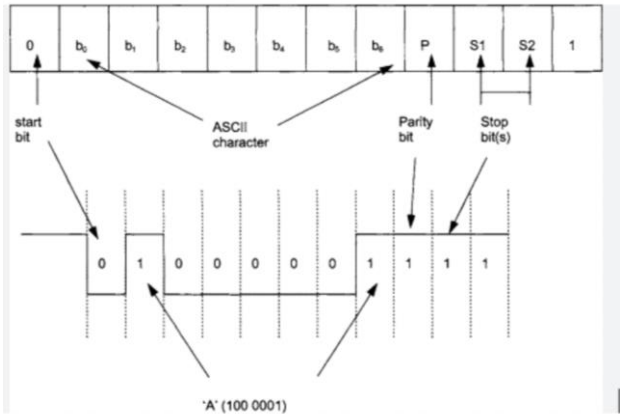

Figure 18: UART configuration example

There are two protocols specified for communication. One is where the user requests the system state and a second where the user sets the system state, or the board sends the state. Aside from the above communication protocols the MCU should transmit the student number of the developer upon startup. This should follow format:

#:STUDENTNUMBER:$/n

Where STUDENTNUMBER is replaced with an actual student number and /n represents the newline character.

The request state communication follows the following protocol:

Table 6: UART Request Protocol

#:	M	x	:$/n

In this protocol the exact characters should be sent by the user in that order and the x should be either F, E, M, depending on whether it is in flash, emergency, or mood mode respectively. This is always sent from the user to the board. When sending this 9-bit instruction the board shall respond with the system state that it is currently in as long as the x parameter corresponds to the current state.

The set state command (user → board) or send state command (board → user) follow identical parameters as follows:

Table 7: UART set or send state protocol

#:	MODE	:	STATE	:	PARAM1	:	PARAM2	:$\n

Table 8 shows the parameter combinations for each mode

Table 8: UART set or send state protocol parameters

	MODE		
MODE value	'MF' – Flashlight	'ME' – Emergency	'MM' – Mood
STATE value	White Light intensity: '000-512'	White Light intensity: '000-512'	Red Channel intensity: '0-512'
PARAM1 value	'000'	'000' in morse mode '001-512' in strobe mode = strobe period	Green Channel intensity: '000-512'
PARAM2 value	'000'	'000' – in strobe mode or in morse mode with default message 'SOS' 'XXX' in morse mode – 3 characters message	Blue Channel intensity: '000-512'

This 19-bit message protocol applies in two scenarios. When the user requests the state of the system, the board shall respond in this format. And when the user sends a message to the board in this format, the board shall set itself in this exact state.

5 Measurements

To prove the system meets the necessary functionality specifications one needs to measure and make sure that the design is correctly translated into practical. A secure testing method is essential to make sure that all possible issues, including not so obvious ones, are considered. In the below testing methods, a thorough navigation through each aspect of all components is done to ensure they are functioning as designed – or if not, to find the necessary data to fix this.

5.1 Power Supply

To evaluate the power supply is working as it should the following procedure is followed:

1. Power up the board with an external 9V supply
2. Measure with a multimeter that 5V is being supplied at the 5V rail
3. Measure with a multimeter that 3.3V is being supplied at the 3.3V rail

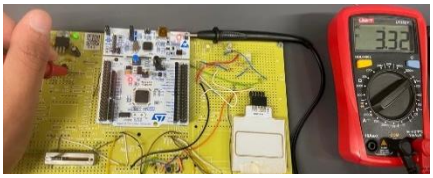

Figure 20: 3.3V regulator test

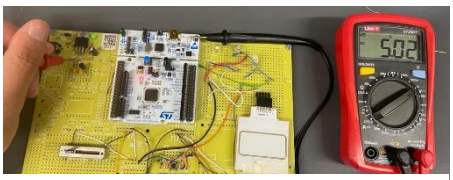

Figure 19: 5V regulator test

As seen in figure 20, and figure 19, the voltages at the rails are a mere 20mV off. This is well within 5% tolerance.

5.2 ADC

The analysis of the ADC is done by testing a set of slider output voltages, for different slider positions, versus the corresponding ADC read in value – scaled to a 3.3V signal and averaged. The slider output voltage is measured with a multimeter. The read in ADC value is averaged over 1 million iterations for a specific slider position to reduce noise and then scaled to a 3.3V voltage. Percentage deviation will be used to measure the error. If the error is sufficient, a set of scaling factors will be used to get a scaling equation to rectify any issues with the read in value. This is explained under software. The method is as follows:

1. Choose a slider position and record the slider output voltage with a multimeter.
2. Record the 3.3V scaled, averaged ADC value via software.
4. Record a table for 15 different slider positions and have a column of scaling factors and one of percentage deviations.
- The scaling factor = V_{SO} / V_{RI} . The percentage deviation equals (V_{SO}-V_{RI})/V_{SO} *100%.
5. Plot a graph of the slider voltage [V] vs ADC value [V] to visually understand any errors.
6. Plot a graph of the slider voltage [V] vs the scaling factor [V] to create a calibration equation.
7. Reapply this process to see that the required tolerance has been achieved.

Results

Table 9: ADC vs slider voltages

slider voltage [V]	ADC value [V]	scaling factor [V/V]	deviation from ideal [%]
3.16	2.94898343	1.071555699	6.677739557
3.32	3.23224616	1.027149492	2.643187952
3.27	3.08874774	1.058681471	5.542882569
3.01	2.7663188	1.088088618	8.09572093
2.63	2.40149379	1.095151697	8.688449049
2.35	2.13828301	1.099012614	9.009233617
2.07	1.87831271	1.102052916	9.260255556
1.8	1.62049234	1.110773532	9.972647778
1.52	1.36981249	1.109640926	9.880757237
1.39	1.24516439	1.116318465	10.41982806
1.07	0.943398476	1.134197295	11.83191813
0.93	0.818583131	1.136109412	11.98030849
0.78	0.684242666	1.139946453	12.27658128
0.59	0.497400671	1.186166474	15.69480153
0.41	0.351070493	1.167856622	14.37305049
2.59	2.507	1.0331073	3.204633205

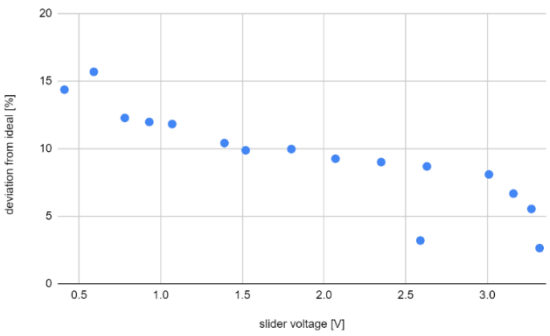

Figure 21: ADC deviation from slider

From table 9 it is seen that the deviations from ideal largely exceed the allowable 5% tolerance. This proves the need for scaling. Further, from figure 21 it's visible to see how the deviation percentages are not a constant value. This implies that we need a non-constant equation to scale the values. A non-linear equation would better approximate the deviations.

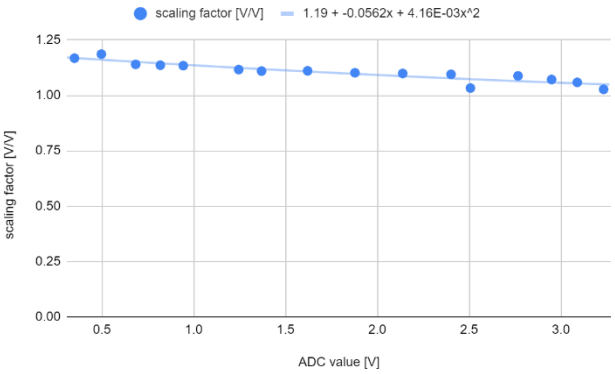

Figure 22: ADC scaling factor

In figure 22, the scaling factors are visible for a range of ADC values, in volts. Note the scaling factors are multiplied by the ADC read in value in volts to approximate the slider voltage. The approximation of these scaling factors for any value of the read in voltage is given by the equation at the top of the graph. This equation is done via non-linear regression analysis via google sheets. The equation is as follows:

$$V_{\text{ADC\_CALIBRATED}} = V_{\text{ADC\_READ}} \times \left(1.19 + 0.0562 \times V_{\text{ADC\_READ}} + \frac{4.16 \times V_{\text{ADC\_READ}}^2}{1000}\right)$$

After calibration with this equation, the following new data was found.

Table 10: Calibrated ADC vs slider voltage

slider voltage [V]	ADC value [V]	deviation from ideal [%]
3.32	3.29999999	0.6024099398
3.21	3.22125486	-0.3506186916
3.05	3.04325894	0.2210183607
2.86	2.85451965	0.1916206294
2.45	2.45685426	-0.2797657143
2.01	1.99954216	0.5202905473
1.87	1.87239841	-0.1282572193
1.45	1.44492365	0.3500931034
1.1	1.11139785	-1.036168182
0.78	0.78641298	-0.8221769231
0.51	0.51045687	-0.08958235294
0.98	0.97865415	0.1373316327
2.33	2.34578413	-0.6774304721
2.07	2.05	-0.2950926471
3.05	3.04	-0.6493016529
2.12	2.10985475	0.4785495283

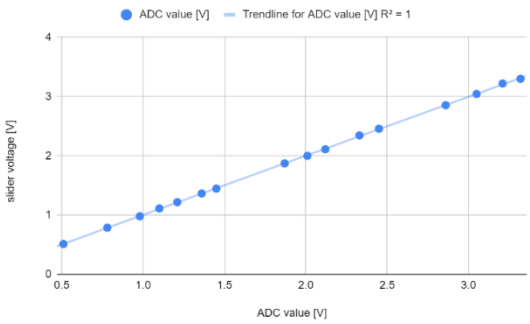

Figure 23: Calibrated ADC vs slider voltage

This deviation of the ADC value from the slider voltage shown in table 10 is all well within the 5% tolerance. The max deviation seen was 1.04%. The graph in figure 23 shows a linear, highly correlated relationship between the ADC read in value in volts and the slider

output voltage. The correlation is also approximately 1. No pictures are included in this section as it is out of the page scope of this report to add that many pictures.

5.3 DAC

The evaluation is done by testing a set of software set digital values sent to the DAC – scaled to a 3.3V voltage - versus the actual DAC output voltage. The DAC digital value is set by UART to ensure the value is not changing and is recorded as a scaled 3.3V voltage. The DAC output voltage is measured with a multimeter.

1. Set the DAC digital value with UART to ensure it does not change due to slider noise.
2. Scale this digital value to a 3.3V voltage and record it.
3. Measure the DAC output voltage with a multimeter
4. Record a table for 15 different voltage values and have a column of percentage deviation from the desired digital value as a voltage. Ensure the maximum deviation is below 5%.
5. Find DEV_MAX = (1+MAX_DAC_DEVIATION_RATIO) x (1+ MAX_ADC_DEVIATION_RATIO). Ensure this is less than 1.05 to ensure there is not a buildup of tolerance exceeding 5% when the ADC and DAC are connected together.
6. Plot a graph of the digital value [V] vs analog voltage [V] to visually see linearity and correlation

Results

Table 11: DAC voltage vs digital value

Digital output [V]	DAC output [V]	deviation from ideal [%]
3.32	3.26	1.807228916
2.32	2.28	1.724137931
1.77	1.74	1.694915254
0.91	0.9	1.098901099
0.41	0.41	0
2.49	2.44	2.008032129
2.02	1.98	1.98019802
2.62	2.59	1.145038168
2.87	2.82	1.742160279
1.46	1.43	2.054794521
0.55	0.54	1.818181818
1.07	1.05	1.869158879
1.61	1.59	1.242236025
2.07	2.05	0.9661835749
3.05	3.04	0.3278688525

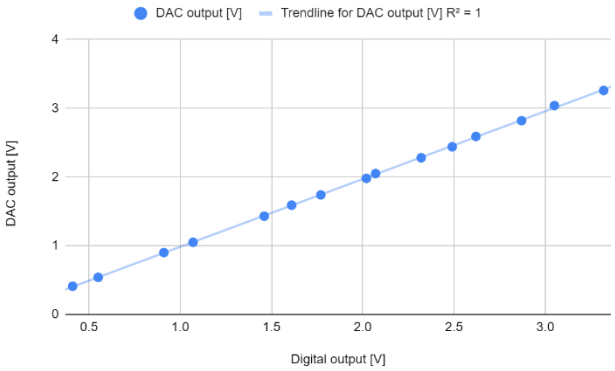

Figure 24: DAC voltage vs digital value

The max deviation from the ideal digital output value in volts is 2.05% as seen in table 11. This is within the specified tolerance. From figure 24 a strong linear trend is visible with an approximate correlation of 1.
DEV_MAX = 1.014 x 1.0205 = 1.0348. This is equivalent to 3.48% peak error when the ADC and DAC are connected and is still lower than 5% tolerance buildup issue will not occur.
No pictures are included in this section as it is out of the page scope of this report to add that many pictures.

5.4 PWM

The evaluation is done by testing a set of software set digital values sent to the PWM – scaled to a 3.3V voltage - versus the actual PWM output voltage. The PWM digital value is set by UART to ensure the value is not changing and is recorded as a scaled 3.3V voltage. The PWM output voltage is measured with a multimeter as it is the equivalent mean voltage that is of interest.

1. Set the PWM digital value with UART to ensure it does not change due to slider noise.
2. Scale this digital value to a 3.3V voltage and record it.
3. Measure the PWM output voltage with a multimeter
4. Record a table for 15 different voltage values and have a column of percentage deviation from the desired digital value as a voltage. Ensure the maximum deviation is below 5%.
5. Find DEV_MAX = (1+MAX_PWM_DEVIATION_RATIO) x (1+ MAX_ADC_DEVIATION_RATIO). Ensure this is less than 1.05 to ensure there is not a buildup of error exceeding 5% when the ADC and DAC are connected in series.
6. Plot a graph of the digital value [V] vs analog voltage [V] to visually see linearity and correlation
7. Verify correct modulation is occurring on the oscilloscope

Results

Table 12: PWM voltage vs digital value

digital output [V]	PWM output [V]	deviation from ideal [%]
3.32	3.299999	0.602439759
1.31	1.33	-1.526717557
0.51	0.52	-1.960784314
1.51	1.5	0.6622516556
0.32	0.32	0
1.24	1.25	-0.8064516129
2.35	2.34	0.4255319149
1.95	1.98	-1.538461538
0.81	0.83	-2.469135802
1.14	1.12	1.754385965
1.89	1.91	-1.058201058
3.01	3.02	-0.3322259136
2.28	2.29	-0.4366812227
2.54	2.53	0.395256917
3.15	3.14	0.3184713376

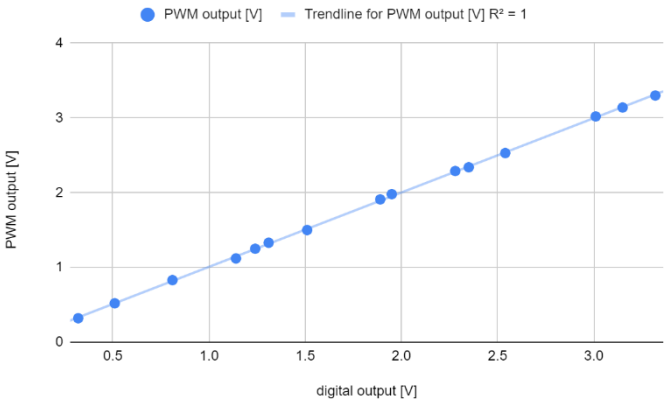

Figure 25: PWM voltage vs digital value

The max deviation for PWM from the ideal digital output value in volts is 2.47% as seen in table 12. This is within the specified tolerance. From figure 25 a strong linear trend is visible with an approximate correlation of 1.
DEV_MAX = 1.014 x 1.0247 = 1.039. This represents a maximum possible error of 3.9% when the ADC and PWM are connected and is still within the 5% tolerance. Pictures of each measurement are out of the page scope of this report. However, a picture showing the working modulation of PWM is included.

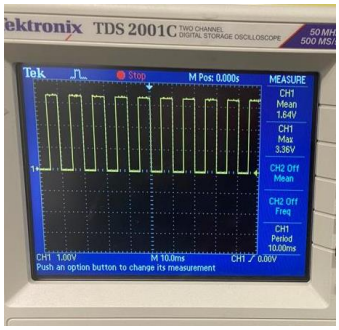

Figure 26: PWM modulation example

Figure 26 shows the PWM with an input digital value equivalent to 1.65V. It is visible that our amplitude remains constant – seen by the max voltage of 3.36V. Further, the mean of 1.64V tells us that exactly half of each period is on and half is off resulting in the correct mean voltage.

5.5 Current amplifier and white LED

The current amplifier was tested in the following way:
1) Connect the multimeter, in ammeter mode, in series between the ground pin of the LED and the boards ground. Note that the ground pin of the LED must first be desoldered.
2) See that the designed 30mA is achieved and that the LED turns on and is bright.

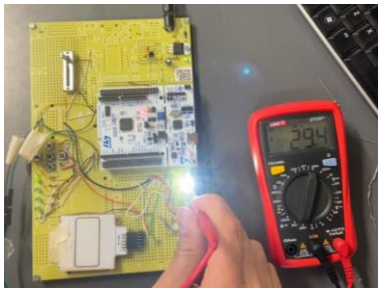

Figure 27: current in white LED

The results in figure 27 shows that the desired current was achieved. This is well within 5% tolerance of the design value. Further, the LED is extremely bright. Hence, the systems are functioning correctly.

5.6 UART

To test UART the following procedure is followed:
1) connect the oscilloscope to the RX pin of the MCU
2) Connect the usb connection from the developers PC to the MCU and make sure termite is open and connected
3) Power up the board
4) the student number should be displayed on termite
5) the student number should be displayed on the oscilloscope as per the UART configuration
6)Set the board to any state via termite
7) send a request state command via termite
8) The board should respond with the current state via termite
9) The oscilloscope should display the state as per the UART configuration

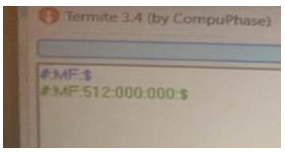

Figure 28: Termite state request

Figure 29: oscilloscope state request

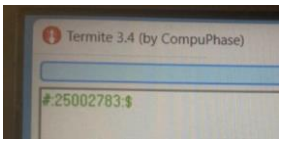

Figure 31: termite student number

Figure 30: oscilloscope student number

It is seen that the student number shows up correctly on termite (figure 31) and on the oscilloscope (figure 30). Similarly, after setting the state to max intensity flash mode, and then requesting the state, the UART responds correctly via termite (figure 28) and the state is displayed correctly on the oscilloscope (figure 29).

5.7 Trackpad

The trackpad will be tested in the following way:
1) power up the board
2) connect the oscilloscope to the ready line pin on the trackpad and make sure it is oscillating
3) connect the board to the developers PC and run the program in debug mode
4) watch the trackpads x and y coordinates in live expressions
5) touch the trackpad and see that the x and y values change correctly.
6) Put your finger at the top right of the trackpad and note down the actual resolution

Results

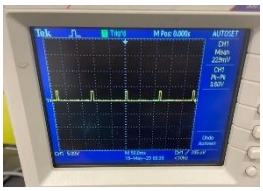

Figure 34: ready line oscilloscope

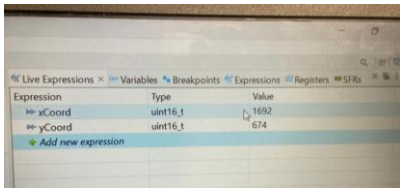

Figure 33: max resolution trackpad

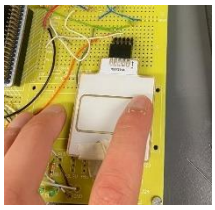

Figure 32: trackpad tap test

As seen the ready line is oscillating implying proper functioning of the trackpad. Further, the x and y coordinates correspond correctly to where I am placing my finger. It is noted that the actual resolution when I attempt to put my finger at the top right of the trackpad is 1692 x 674 and not the expected 1792 x 768. This is considered when setting up the trackpad to make sure the maximum value for the RGB LEDS is achievable.

5.8 RGB and Debug LEDS

The current through the LEDs was designed to be around 4mA to ensure MCU pin safety and sufficient brightness. Due to having to choose standard resistor values, the acceptable range becomes between 3 and 5mA. Further note that the effect of resistor tolerances plays a part in the current measurement. The functioning of the RGB and debug LEDs are measured with the following procedure:

1) Remove all ground connection from the 4 debug LEDS and the 3 RGB LEDS.
2) Connect the multimeter in series, in current mode, between each LEDs ground connection and the boards ground. Do this for all LEDs.
NOTE: Alternatively, measure the voltage over the resistors in series to the LEDs and divide it by the resistance to find the current. Measure resistance and do not simply use the standard value.
3) Make sure the LED turns on which entails the forward voltage is being correctly supplied.
4) The LEDs should have currents between 3 and 6mA to ensure sufficient brightness to the LEDs and safety to the GPIO pins.

Results

LED	current
Red	4.03mA
Blue	3.69mA
Green	4.7mA
D2	I=V/R = 1.28/325 = 3.94mA
D3	I=V/R = 1.29/319 = 4.04mA
D4	I=V/R = 1.26/326 = 3.87mA
D5	I=V/R = 1.26/324 = 3.89mA

Figure 35: Debug and RGB LED currents

Figure 35 shows from left to right the voltages over the resistors in series to the debug LEDs from D2-D5 and then the currents through the RGB LED from R to G to B.
These currents are all within the acceptable range for sufficient brightness and MCU pin safety.

5.9 Complete System

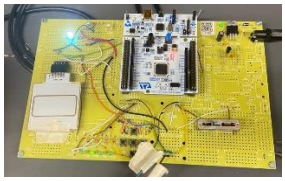

Figure 36: complete system

The complete system has passed all 4 demos with an average of 80%. The system functions almost exactly as specified in the user requirements. The demos include rigorous testing of all functionalities of the system and hence it is deduced via testing that the complete system is functional to a high degree. Alongside the demos, there were also tests done such as mentioned above to ensure the entire system works as it should.

5.10 Button debounce and current

The buttons were tested via the following way:
1) remove the ground connections of the button and connect the multimeter in series, in current mode, between the button ground pin and the boards ground.
2) measure the current and make sure it is less than 8mA
3) reconnect and press the button until you see bounce on the oscilloscope.
4) Measure the period of this bounce and repeat around 100 times to find a maximum bounce period.
5) Once software debounce is implemented, press the buttons 100 times and make sure the debug LEDs do not skip a mode.

Results

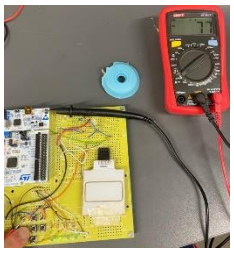

Figure 38: button to GPIO current

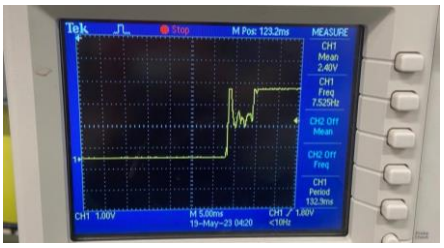

Figure 37: Bounce example oscilloscope

Figure 38 shows the current through the button flowing into the GPIO pins is 77uA. This is safe for the pins. Further, bounce is visible in figure 37 and this will cause false presses. This needs to be calibrated in software. Upon testing over 100 times, it was seen that the longest bounce period was 8ms. Hence the designed bounce period was 20ms as specified in the software.

Once calibrated, the button was pressed over 100 times and no bounce or mode skips were visible via the debug LEDs.

6 Conclusions

The multifunctional lighting system was designed and implemented successfully. It met the user requirements of the module and was rigorously tested. The modes have been demonstrated in accordance with their hardware specifications, the software logic, and testing methods.

Compliance with requirements

The system complied almost completely with all user requirements as seen by results in the demo. GIT, a versioning software, was used per the requirements to upload copies of the code prior to each demo. Noncompliance issues arose due to sending debug messages via UART during demos. Further noncompliance was due to button bouncing which occurred when the program became large causing unexpected behavior which was not solved.

shortcomings of design

The lighting system was not portable and extremely large. The long wires and large components cause a poor looking system. Ultimately although the system tries to act as a flashlight, emergency light, and mood light, its scale and design do not allow it to be a practical implementation of any of these devices.

Possible improvements

The system should focus on either being a flashlight and emergency light or a mood light. Trackpad gestures could be used to control all parameters instead of multiple inputs. The board could be entirely made in software and manufactured for a compact, reliable system. The flashlight and emergency light could be made into a torch form. The mood light could be connected to a set of individually addressable RGB strip lights allowing for a highly sophisticated RGB strip light controller.

7 References

[1] A. Barnard, "Policy Document," Stellenbosch University, Cape Town, 2023.

[2] JameCo Electronics, "Positive voltage regulators L7800 series," 2006.

[3] farnell multicomp, "LED YELLOW/GREEN," 2012.

[4] STMicroelectronics NV , "STM32F303xD STM32F303xE," Geneva, 2016.

[5] Cree, "P4 LED," Cree, Durham, 2014.

[6] Continental Device India Limited, "NPN SILICON PLANAR SWITCHING TRANSISTORS," Continental Device India Limited, New Delhi.

[7] Kingbright, "T-1 3/4 (5mm) FULL COLOR LED LAMP," Kingbright, City of Industry, 2013.

[8] Azoteq, "IQS7211A DATASHEET," Azoteq, Cape Town, 2022.

[9] ScienceDirect, "Parity Bit," PC Interfacing, 1998. [Online]. Available: https://www.sciencedirect.com/topics/engineering/parity-bit.

[10] HTC, "100mA LOW DROPOUT VOLTAGE REGULATORS," HTC, 2015.