ELECTRICAL & ELECTRONIC ENGINEERING
STELLENBOSCH UNIVERSITY

DESIGN (E) 314
TECHNICAL REPORT

Multi-Functional Light Source

Author:
Xari Mouchtouris

Student Number:
23795824

22/May/2023

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*
Voorletters en van / *Initials and surname*	Datum / *Date*

Abstract

This report describes the hardware design, software design and testing of a Multi-Functional Light Source. The system is implemented on a provided PCB board with software written for the STM32F303RE. The system implementation worked as a light source, being able to be used in all three main modes at full intensity range. Both the RGB and the white LED worked. The hardware and software implementation worked together to implement the Multi-Functional Light Source.

Contents

List of Figures

List of Figures

List of Abbreviations

TIC Test Interface Connector

ADC Analogue to Digital Converter

DAC Digital to Analogue Converter

I2C Inter-Integrated Circuit

GPIO General Purpose Input Output

SDA Serial Data Line

SCL Serial Clock Line

UART Universal Asynchronous Receiver-Transmitter

LED Light Emitting Diode

PCB Printed Circuit Board

ASCII American Standard Code for Information Interchange

NPN Negative Positive Negative

RDY Ready Line

PWM Pulse Width Modulation

STM STMicroelectronics

RGB Red, Green and Blue

List of Symbols

V Volts – Voltage

A Ampere – Current

k Kilo

Ω Ohm – Resistance
μ Micro
n Nano

m Milli

Hz Hertz – Frequency
s seconds - Time
F Farad - Capacitance

1 Introduction

The goal of the project is to design a multi-functional light source using a RGB LED and white LED. The system is built on a provided PCB and a STM32F303RE controls the software logic for the light source. Different peripherals control the input to the device. The trackpad, slider, buttons and UART commands can all change the state of the system. The first main state of the system is Flashlight mode, where the white LED can be switched on with button/trackpad hold input. The intensity of the white LED in this mode can be changed with slider, trackpad slide and UART input. The second main mode is Emergency mode, which further breaks down into three different sub-modes: Strobe mode, SOS mode and Custom-message mode. Emergency mode also uses the white LED as the light source, and the intensity or frequency can be changed by the trackpad or UART input. The third main mode is Mood mode, which uses a RGB LED as its light source. The colour channels of the RGB LED can be varied with trackpad taps or UART command input. The left buttons and right button are used to cycle through the different modes, where the middle button toggles the LEDs on or off. The first aspect discussed in the report is a system description of what is required in the system. The second aspect is the hardware design implementation of the system. Software design is the third aspect of the report, and the fourth aspect is to verify with measurements that the hardware and software implementation works. The final aspect of the report is a conclusion to identify if all requirements are met and discuss any shortcomings and non-compliances.

2 System description

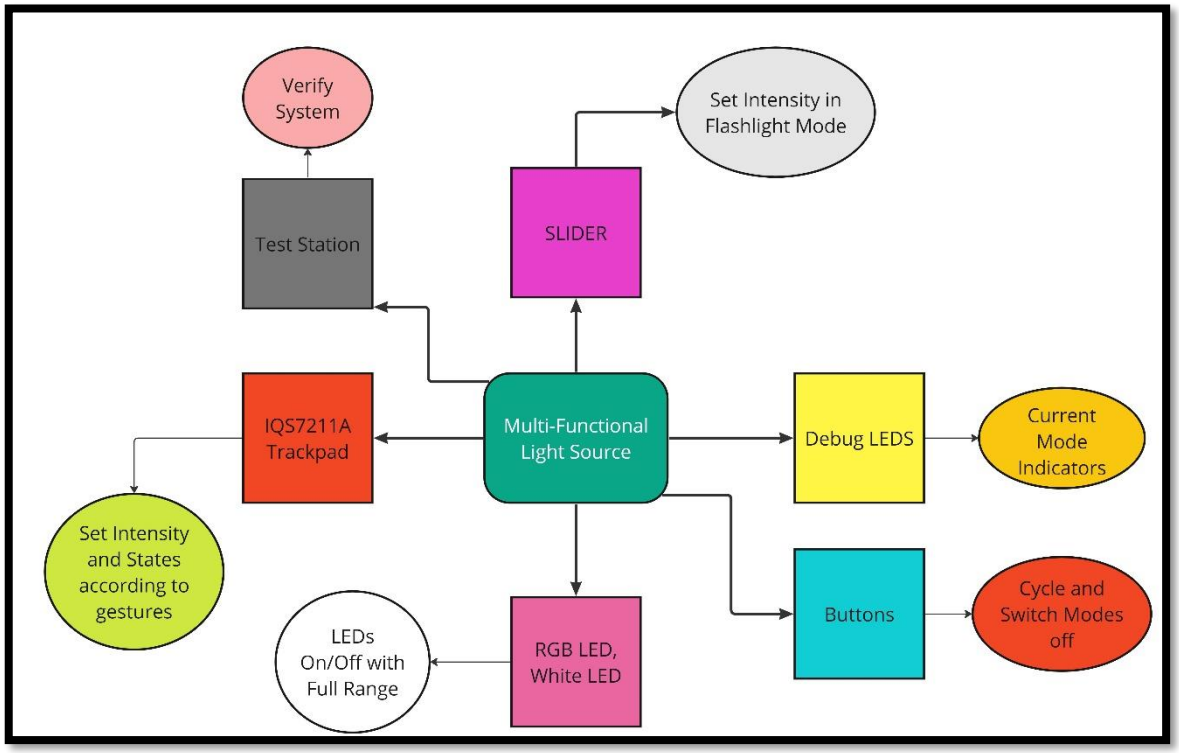


Figure 1: System Diagram

The main goal of the system is to drive a white LED and an RGB LED high in different mode states. The three main modes are Flashlight mode, Emergency mode, and Mood mode. Emergency mode consists of three sub modes: Strobe mode, SOS mode, and Custom message mode. When in Flashlight mode, the white LED emits a beam of light with adjustable intensity. Within Strobe mode, the white LED strobes with an adjustable intensity and frequency. In SOS mode, the white LED is utilized to transmit the Morse code message 'SOS' using its light output. In Custom Message mode, the white LED can be used to send a custom message over its light output using Morse code. Mood mode activates the RGB LED with adjustable intensity levels across its three distinct colour channels. The system can be controlled with different input mechanisms. The left button of the system changes between main modes. The right button changes between sub modes within Emergency mode. The trackpad has the same functionality as the right button and can also change sub modes. UART commands can also set a specific mode and state to the system. Overall, the system has three different modes and specific parameters set the function of two LEDs.

2.1 Summary of System Requirements

Table 1 below of user requirements were compiled and modified from the PDD [1, p.2].

Table 1: User Requirements modified from PDD

User Requirements #	User Requirement Description
UR1	The power supply should provide +5V and +3.3V to the rest of the system after being provided with +9V nominal
UR2	Flashlight Operation: Toggle White LED On/Off. Input Method: Trackpad Tap & Hold, Middle Button (Debounced) and UART Command Intensity Change: UART Command, Trackpad slide gesture and ADC Slider input
UR3	Emergency Mode: White LED strobe, morse with correct intensity and frequency. The duty cycle should always remain at 50% duty cycle. Input Methods: UART Command changes frequency and intensity. ADC Slider also sets the intensity of strobe and morse. The trackpad slide gesture sets the intensity as well. Trackpad tap and right button (debounced) pressed cycle through Emergency sub-modes.
UR4	Mood Mode: Toggle RGB LED with trackpad tap and hold gesture and middle button press (debounced). RGB LED colour intensity change with trackpad tap (x,y coordinates), UART Command Request and Button default values
UR5	UART Setup: 8E2 UART Configuration with Baud rate of 57600 bits/s Request command and parameters received through UART and current state reported back through UART
UR6	Trackpad (IQS7211A) Implementation: TAP, TAP & Hold and Slide Gestures
UR7	Three push buttons (left, middle and right) implemented. Middle toggles RGB and white LED. Left button cycles through main modes: Flashlight mode, Emergency Mode, and Mood mode. Right button cycle through sub-modes in emergency mode: Strobe, SOS and Custom-message mode.
UR8	Four Debug LEDs to be used as mode indicators
UR9	The RGB and White LED should be driven high by PWM Channels. Four PWM channels should be used. One channel should be used in the amplifier transistor circuit and the other PWM channels to drive each colour channel high for the RGB LED. DAC is only monitored on the TIC.

3 Hardware design and implementation

3.1 Hardware Block Diagram and Description of Interaction

The Hardware block Diagram below in Figure 2 shows the entire implementation of the E-Design 314 2023 project using a STM32F303RE microcontroller. The diagram indicates all the components used in the development of the system and what functions of the microcontroller were used to achieve a Multi-Functional Light Source. The system receives a 9V power input from a 9V power supply or a 9V battery. This voltage then gets stepped down using two voltage regulators (LM7805 and LM2940) to 5V and 3.3V. These two voltages then get distributed throughout the system where needed. The 5V powers the Nucleo-F303RE microcontroller and white LED. The 3.3V is used for all external components. The Slider, Buttons, Debug LED's and IQS7211A Trackpad all require a 3.3V power input. The slider is used to vary the intensity of the white LED in its different modes. The buttons are used to change between different modes and sub modes. The trackpad can be used to turn the LED's off/on and cycle between sub-modes. The trackpad can also increase the intensity of flashlight and emergency with a slide gesture. The debug LEDs are used to indicate the current mode state. The RGB LED is powered by three PWM channels to supply red, green and blue colour intensities. The white LED is driven high by a 2N2222A transistor and a PWM channel. The transistor implements a current amplifier circuit for the white LED. Throughout the system current limiting resistors are used to limit current flowing through the GPIO Pins. The test station (TIC) is important in the design of the system because it is used for verification purposes. Thus, most peripherals are connected to the TIC to verify that each component are behaving as designed during demonstrations.

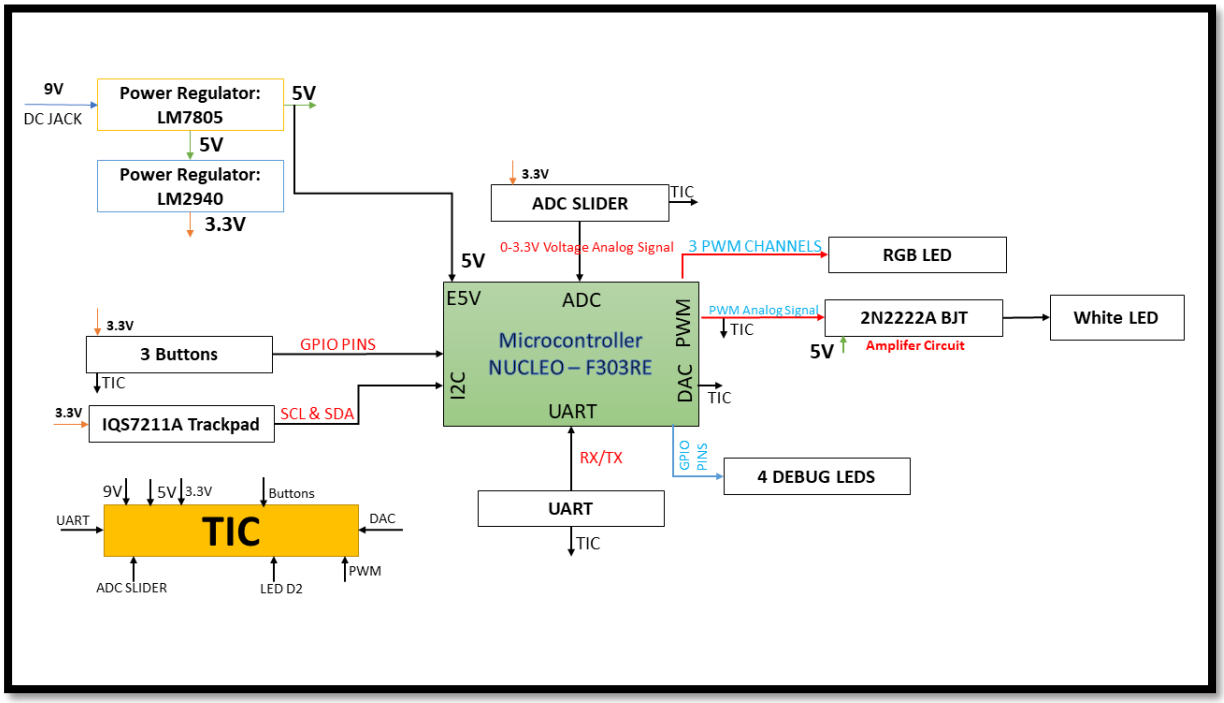


Figure 2:Block Diagram of System

3.2 Power Supply

The power supply circuit designs were decided by the specifications provided in the PDD, [1, p.7].

Specification 1:" The board must be supplied with a nominal 9 V Battery source (but you can use a bench power supply during development and testing, instead of a real battery). Your project will require you to implement both a 5 V and a 3.3 V power regulated supply".

The board uses an external 9 V DC power supply, provided by a bench power supply or a nominal 9 V battery. The battery or bench power supply connects to the PCB (Printed Circuit Board) using a barrel jack connector. The power supply circuit uses two voltage regulators, LM7805CV (U1) and LM2950 (U2). The 9V input voltage connects to the U1 regulator with a 1N4007 diode (D1). This D1 diode prevents any negative voltage feedback into the voltage regulator circuit. Thus, if the DC power cables polarity is switched it prevents any damage to the circuit. The LM7805 regulator is used to regulate the 9 V input voltage to a DC voltage of 5 V. The 5 V voltage regulator circuit uses three capacitors. The 100nF (C1) capacitor is connected to the input of the U1 regulator, and acts as a filter capacitor. This capacitor filters out any noise from the 9 V DC input. The output connects to a 10uF (C2) capacitor and a 100nF (C3) capacitor. These output capacitors also further filter the output dc voltage. The output of the U1 regulator has a voltage of 5V, which connects to the rest of the PCB and to pin E5V on the STM32F303RE microcontroller. The 5V regulator circuit also has a test LED (D6), to test if the regulator circuit has an output of 5V. This D6 LED is connected to a 1 KΩ series-resistor and if the circuit is working D6 will turn on. See below the 5V voltage regulator circuit design in Figure 3.

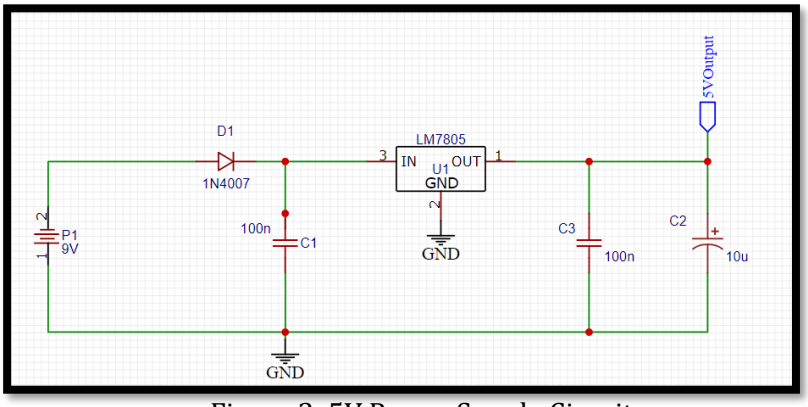


Figure 3: 5V Power Supply Circuit

The LM2950 (U2) regulator is used to regulate the 5V output of U1, to 3.3V. The output of U1 is connected to the input of the U2 regulator. Both the output and input of the U2 regulator have a capacitor, C4 and C5. These capacitors are decoupling capacitor that filters out high-frequency noise cause by the power supply. The output of the U2 regulator provide power for other components and ports on the PCB. See below the 3.3V voltage regulator circuit design in Figure 4.

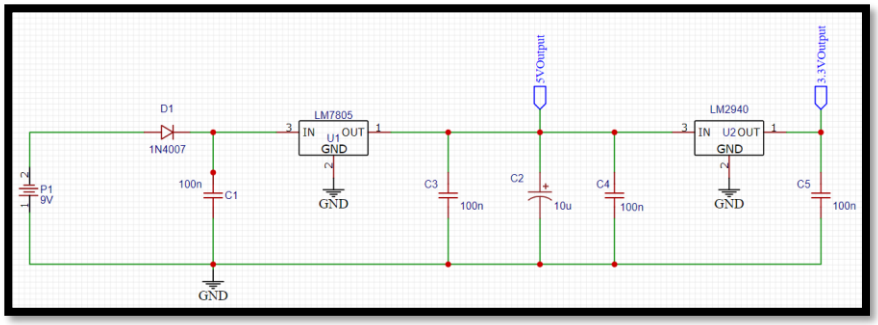


Figure 4: 3.3V Power Supply Circuit

3.3 LEDs (Debug)

The PCB uses four debug LEDs for display and debug purposes, according to the PDD [1, p.17]. The purpose of these four LEDS is to indicate a particular status or mode that the system is currently operating in. These LEDs are connected to the different GPIO PINs on the STM32F303RE board, by a current limiting series resistor. Each LED is used as a system state indicator, according to the PDD LED D2 is on when the system is in flashlight mode, while LED D3 is on when in emergency mode. LED D4 is on when the system is in mood light mode. LED D5 is on only during emergency mode when a morse message is being flashed. Figure 5 below shows the configuration of the four debug LEDs.

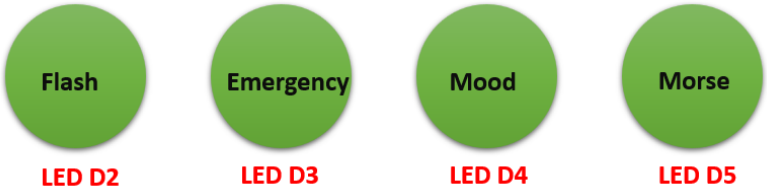


Figure 5: LED Configuration

The specification of the LEDs is a 3mm green LEDs that connect to different GPIO pins on the microcontroller. A series resistor has been used to limit the current through the LEDs, to not damage the LEDs or the GPIO pins. The LEDs receive an input voltage of 3.3V from each GPIO pin and the LEDs connect to a common ground. LED D2 is the only LED that connects to the TIC for verification purposes. Table 2 illustrates the different GPIO PINS for each LED.

Table 2: Debug LEDs GPIO Pins

DEBUG LEDS	GPIO PIN
LED D2	PA10
LED D3	PC7
LED D4	PA9
LED D5	PB5

Figure 6 below illustrates the schematic of the design of all four debug LEDs, it also indicates the GPIO Pin for each LED and the value of the current limiting series resistors.

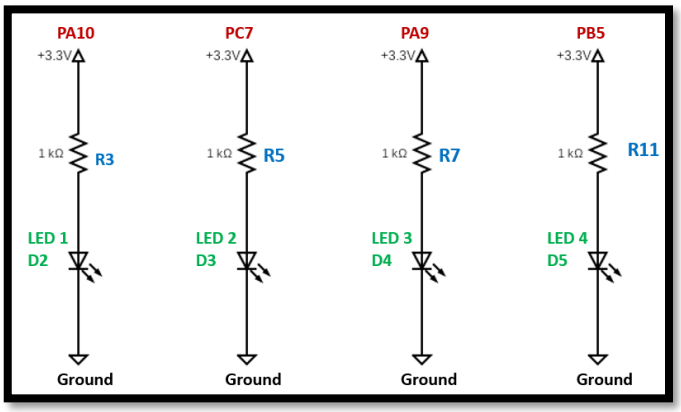


Figure 6: Diagram of Debug LEDs

The 3mm green LEDs supplied to us are sufficient to be used as debug LEDs. An assumption was made that the voltage drops over the LEDs is 2.1V. A typical forward voltage range of a 3mm LED, according to the datasheet [2, p.1] is 2V – 2.1V, thus my assumption is sufficient. The voltage drops over the resistors required should then be 1.2V, because the supplied voltage is 3.3V. The design value current chosen for the LEDs is 2mA, which is a safe current for both the LEDs and the GPIO pins. See below OHM's Law that was used to calculate the required resistor value for the circuit.

- $V = I*R$
- $R = \frac{1.2\ V}{0.002\ A} = 600\ \Omega$

As seen from above a 600-ohm resistor is sufficient for limiting the current, however for practical purposes the system used 10% value 1 kΩ resistors. These resistors still work in the circuit and the current is still high enough to turn on the LEDs. By using Ohm's law below can be seen that the current is still high enough for the LED to turn on.

- $I = \frac{1.2\ V}{1000\ \Omega} = 1.2\ mA$

3.4 Buttons

The PCB has space for five pushbuttons according to the PDD [1, p. 9], thus five pushbuttons were connected in active low configuration. Each pushbutton was connected to a GPIO Pin on the STM32F303RE board and to be used as an external interrupt in the IOC file. An external current limiting resistor has been used to limit the current flowing through the GPIO PIN. The hardware implementation of the pushbuttons can be seen below in figure 7.

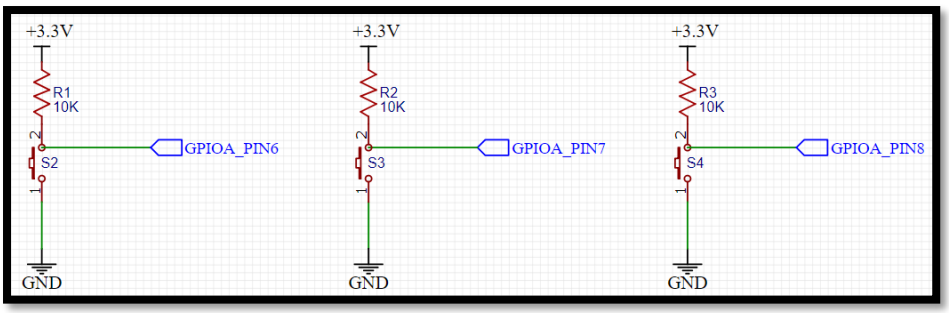


Figure 7: Diagram of Button Schematic

Table 3 below shows what GPIO Pin connects to what pushbutton.

Table 3:Pushbuttons GPIO Pins

Pushbutton	GPIO PIN
Pushbutton 1 (S2)	**PA6**
Pushbutton 2 {S3)	**PA7**
Pushbutton 3 (S4)	**PA8**

Figure 7 shows only three of the five pushbuttons, because according to the PDD [1] only three of the five pushbuttons have functionality. The left pushbutton (S2) is used to switch between main modes. The middle pushbutton (S3) is an off/on switch and the right pushbutton (S4) switch between sub modes. The pushbuttons use an external resistor. The external resistor should be designed to have the correct value to limit the current through the GPIO Pin sufficiently. The maximum allowable current sunk through a GPIO Pin on the STM32F303RE board is ±25 mA, according to the datasheet [3, p.71]. The system design choice is an external resistor with 6mA current flowing through it. The 6mA is below the specified max allowable current sunk of the GPIO PIN. The supplied voltage supplied to the pushbuttons is 3.3V. We can calculate what the resistor value should be using OHM's Law.

- $R = \left(\frac{VSupply}{Imax}\right)$
- $R = \frac{3.3}{0.006} = 550\ \Omega$

Thus, if we use a 550 Ω resistor that would be sufficient to limit the current through the GPIO PIN. The buttons use a standard 10% value for the resistor of 10 KΩ. This 10 Ω is sufficient and limits the current further. See below OHM's law for the current through a 10 KΩ resistor.

- $I = \frac{3.3}{10000} = 0.33\ mA$

Thus, the system uses a 10 KΩ resistor to limit the current sufficiently, because the current for the 10 KΩ resistor is 18 times smaller than the 550 Ω resistor. Thus, this resistor value will work in my design of the pushbuttons.

3.5 SLIDER/ADC

The project uses an analogue slider. The slider is a 10 KΩ potentiometer, which is wired up as a voltage divider which feeds into the ADC of the STM32F303RE microcontroller. The output of the slider outputs an analogue 0-3.3V signal into the microcontroller. The output voltage signal decreases/increases depending on which direction you slide the slider. See below the orientation of how the slider works.

- **LEFT**
- **Right**

The left side of the slider has two pins connected, while the right side has only one pin. The slider needs a 3.3V input voltage and the right side is connected to ground. The second pin on the left is the output voltage pin of the

slider which gets feed to the ADC of the microcontroller. The maximum voltage the ADC can receive is 3.3V on the far-left side and the minimum is 0.0V on the far-right side. The ADC receives the analogue voltage and converts it to a scaled value of 0 – 512. See below the schematic of the pin layout of the slider in figure 8. Table 4 indicates what GPIO Pin the ADC is using.

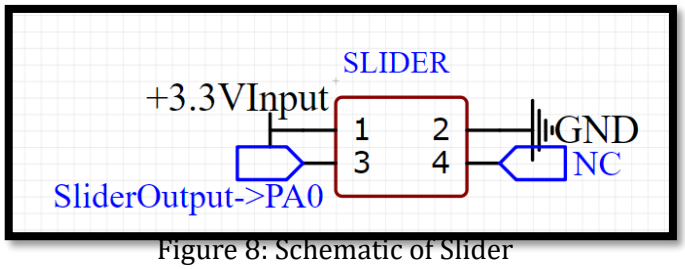


Figure 8: Schematic of Slider

Table 4: ADC GPIO Pins

ADC	GPIO PIN
ADC1_IN1	PA0

3.6 DAC/PWM filters

In a previous version of the PDD [1], the design requirements were to make use of a DAC/PWM summing Op Amp. DAC/PWM filters were going to be used to smooth out the signal received from the DAC and PWM outputs. This was necessary because the summing Op Amp would sum together two analogue signals and the filters would filter out high frequencies and any noise present in the system. The filtered signal would be a better representation of the desired signal to drive the White LED high. This implementation is not necessary, because the system used a current amplifier instead. Thus, because the PWM signal is only used to drive the White LED high, the implementation of a filter is not necessary.

3.7 Azoteq trackpad

The trackpad being used in the system is an Azoteq Rectangular Trackpad – IQS7211A. The trackpad communicates with the microcontroller using I2C. The trackpad has ten hardware pins, out of which six are utilized for connection to the PCB and microcontroller. The two I2C communication lines are SDA and SCL, according to the PDD [1, p.10]. The trackpad also consists of two digital lines MCLR and RDY. The trackpad can only operate on a V_{DD} of 3.3V and can't handle a voltage of 5V, according to the PDD [1, p.10]. The trackpad limits current through the pins with an internal 4.7 KΩ resistor, according to the datasheet [6]. Figure 9 below shows the schematic of the configuration of the trackpad and the GPIO pins used for the system.

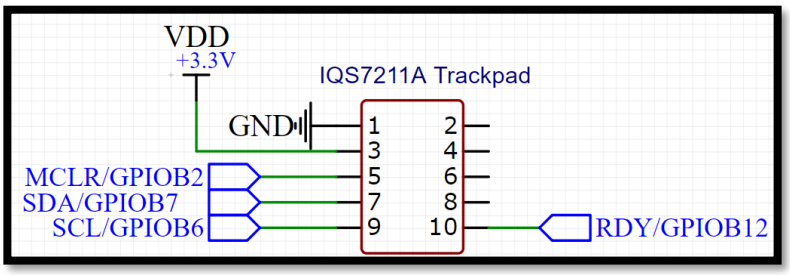


Figure 9: IQS7211A Schematic

Each pin of the trackpad has a different functionality. When the RDY pin is low it indicates that new data is available and that a communication window is open. The configuration of the RDY line is active-low, thus when the microcontroller is communicating with the trackpad the RDY line should be low. The MCLR line is used for a hardware reset at initialisation of the system. The SDA line is the I2C Data Pin and the SCL is the I2C Clock Pin. Figure 10 illustrates the connection between the microcontroller and the IQS7211A trackpad.

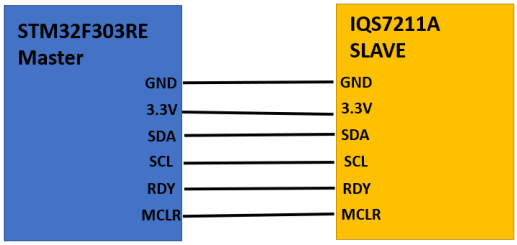


Figure 10: STM32 connection to IQS7211A Trackpad

3.8 RGB LED Circuit

The system uses a RGB led from Kingbright. This led is used in mood mode and is controlled with the buttons, UART and the trackpad. The RGB led is driven high by three PWM channels, according to the PDD [1]. Each PWM channel is used for one colour of the RGB LED. The PWM channel supply a 3.3V input to the led. Current limiting resistors are used to limit the current flowing through each channel of the RGB led. Each colour of the RGB led have a different forward voltage (V_{TN}), according to the datasheet [4]. Thus, three resistor values should be designed for the RGB led. The system was designed for 5mA forward current, which is below the current of both the GPIO Pin and RGB led, according to the datasheet [4]. The different forward current tables were used in the datasheet [4]. See below the calculations for each resistor value

$$\text{Equation: } V_{CC} - (Imax)*(R) - V_{TN} = 0$$

Red LED:	Blue LED:	Green LED:
$V_{TN} = \pm 1.85V$ @ 5mA	$V_{TN} = \pm 2.80V$ @ 5mA	$V_{TN} = \pm 2.9V$ @ 5mA
$3.3 - (5mA)*R_{red} - 1.85 = 0$ $R_{red} = 290\ \Omega$	$3.3 - (5mA)*R_{blue} - 2.80 = 0$ $R_{blue} = 100\ \Omega$	$3.3 - (5mA)*R_{green} - 2.9 = 0$ $R_{green} = 80\ \Omega$

The above resistors will limit the current sufficiently, but for practical purposes to get the colour of the RGB led correct 10% value resistors were used. The intensity of the RGB channels wasn't as expected, thus higher resistors were used to achieve the correct intensity. By using a variable resistor, higher resistor values were tested until the correct intensity were achieved. Table 5 illustrates the GPIO pins and PWM channels used for the RGB led. Figure 11 illustrates the schematic of the RGB LED circuit, and the higher 10% resistor values used.

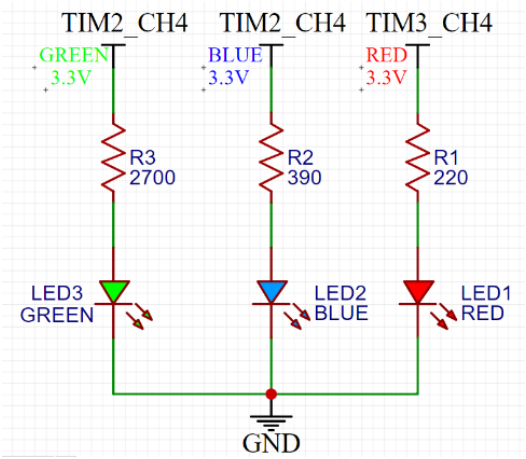


Figure 11: Schematic of RGB LED Circuit

Table 5:RGB LED GPIO Pins & PWM TIMERS

RGB LED	GPIO PINS
RED Channel:	PC8 -> TIM3_CH4
Green Channel:	PA12 -> TIM4_CH2

BLUE Channel:	PB11 -> TIM2_CH4
Ground	Ground

3.9 White LED Amplifier Circuit

The system implements a white LED amplifier circuit to be used in Flashlight mode and Emergency mode. A Cree P4 LED is being used as the white LED and a 2N2222A transistor to drive the amplifier circuit. The transistor is a NPN transistor, and the transistor circuit is a Common-Emitter configuration. The goal of the amplifier circuit is to amplify the current, high enough to turn on the high-power LED, according to the PDD [1, p.10]. The amplifier circuit requires a current limiting resistor thus, a collector resistor value and a base resistor value should be designed.

Design Requirements:

- β = ±140 (according to the datasheet [5])
- $I_{Forward}$ design for choice was 25mA
- Required input voltage is 5V, according to PDD [1]
- PWM max voltage is 3.3V
- $V_{forward}$ of LED is 3.6V of the datasheet [5]
- $V_{BE(sat)}$ = 0.7V according to the datasheet [5]

Collector Resistor Formula:

$V_{INPUT} = V_{LED} + R_C * I_c$

$5 = 3.6 + R_C * (25 \times 10^{-3})$

$R_C = 56\ \Omega$

Base Current Design:

$I_C = 0.5 * \beta * I_B$

$I_B = \frac{Ic}{0.5 * \beta}$

$I_B = \frac{25 * 10^{-3}}{0.5 * 140}$

$I_B = 3.571 * 10^{-4}$

Base Resistor Formula:

$V_{PWM} = V_{BEC(sat)} + R_B * I_B$

$3.3 = 0.7 + (3.571 \times 10^{-4}) * R_B$

$R_B = 7280.87\ \Omega$

For practical purposes 10% value resistors were used that were close enough to the theoretical values. See below the schematic in figure 12 of the white LED.

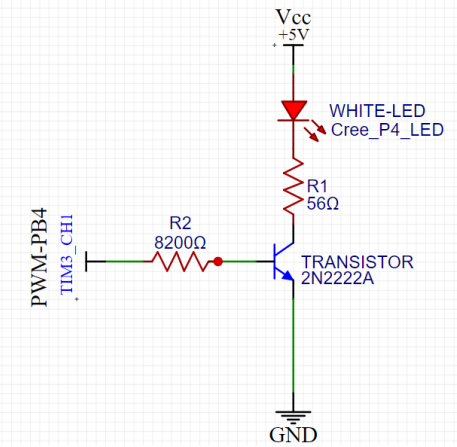


Figure 12: White LED Schematic

4 Software design and implementation

4.1 Software Block diagram and description of interaction

Figure 13 below shows a block diagram of the software state machine implemented for the Multi-Functional Light Source. The diagram indicates that at startup the system initializes all required peripherals. After initializing all

peripherals, the system sends the student number through a UART message to the TIC. The software diagram shows the input methods to communicate with the system. There are three main input methods, namely the buttons, UART commands and the trackpad. Each input has a flag which gets set in software when the input is triggered. The button flags get set when an external interrupt is detected for each button. The button cycles through the different Main modes and sub-modes. Without UART or any Trackpad input detected the intensity

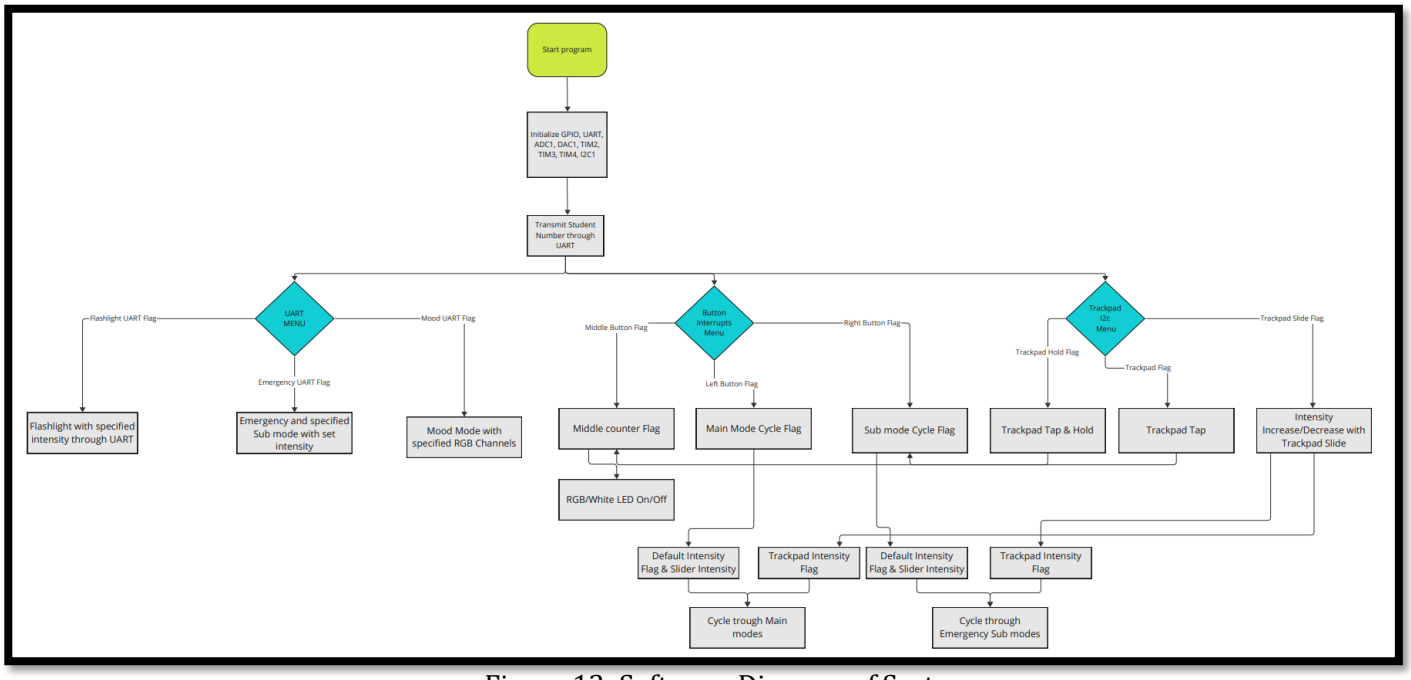


Figure 13: Software Diagram of System

of the White LED is by default the slider value and the RGB LED is set to default values specified within the software. If the trackpad input gets detected, the software checks what flag of the trackpad is set. The system updates according to the flag set, and either the intensity of Flashlight, Emergency or Mood mode changes. A UART request command flag sets a flag, and the current state to the request state. The intensity, frequency and message is set according to the specific request mode command send through UART.

4.2 Button bounce handling

Figure 14 shows how the button bounce handling was implemented for each button in the system. Each button makes use of the System Tick Timer interrupt for the button debounce. Each button is set to External Interrupt Mode on falling/rising trigger edge detection. When the software detects a button press the according flag for the correct button is set to high. For each button a global timer variable of size uint32_t is set to zero at startup. The variable is set equal to the System Tick Timer each time in the while loop of the specific button. The loop checks if the time elapsed from first pressing the button is greater than 50 milliseconds using an if statement. After the elapsed time the action of the button can be executed. This ensures that all bounce from the button is removed before performing the execution. The same button debounce code is used for the left button, middle button and right button and each button has its own global timer variable to ensure the correct timer is incremented when a button press is detected. Table 6 below shows the three different GPIO Pins set to External Interrupt for each button.

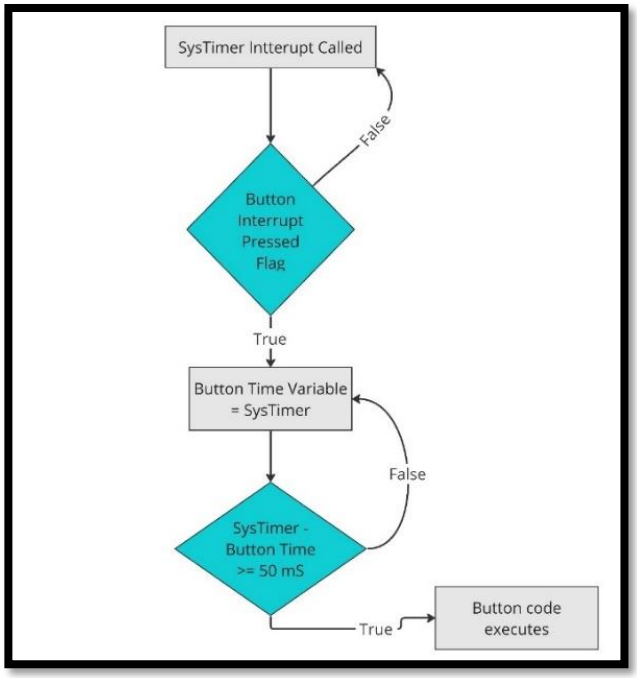


Figure 14: Button Debounce Software Diagram

Table 6: External Interrupt for each Button

External Interrupt with rising/falling:	GPIO PINS
Left Button (S2)	PA6
Middle Button (S3)	PA7
Right Button (S4)	PA8

4.3 UART communications (protocol and timing)

The main communication channel of the system happens through UART. The STM32F303RE board supports UART functionality, according to the datasheet [3, p.30]. UART is used for all communication to the student board and all communication back to the test station is also through UART. The student board should report on current operation modes, white LED intensity, on time of the strobe light, the current emergency message being flashed and the intensity of the RGB LED channels. UART communication is an easy way to communicate between the student board and test station. It also offers easy access to a virtual port to read and write data to. This helps with development of software using programs like Termite and Tera Term.

4.3.1 UART SETUP

1. The UART is configured as USART2 in asynchronous mode in the IOC file, with 57600 Bits/s.
2. The UART is setup to have 1 start bit, 8 data bits, one even parity bit and two stop bits using TTL levels (8E2), according to the PDD [1, p.17].
3. The UART messages use ASCII characters to send and receive messages to the system.
4. The global interrupt for USART2 is enabled in NVIC setting in the IOC file.

4.3.2 UART RECEIVE

Figure 15 below describes the Callback function of the method to receive commands over UART. The callback function reads one byte of data at a time and stores it in a uint8_t buffer of array size one. If the buffer detects the ASCII character '#' it starts storing each byte of data in a mode command change buffer of array size 19. A counter is incremented every time a byte of data is stored in the buffer. If byte seven is equal to the ASCII character '\n', then it is a mode request command being sent and not a mode command change request. A new mode request buffer of array size seven, is set equal to the command change buffer. The appropriate flag is set high to indicate that a mode request command was received. If byte seven is not equal to the ASCII character '\n' the command change buffer continues to store the received bytes within the command change buffer. If 19 bytes of data have been received, the callback function stops storing bytes of data. The appropriate flag is set high to indicate that a command change mode has been received.

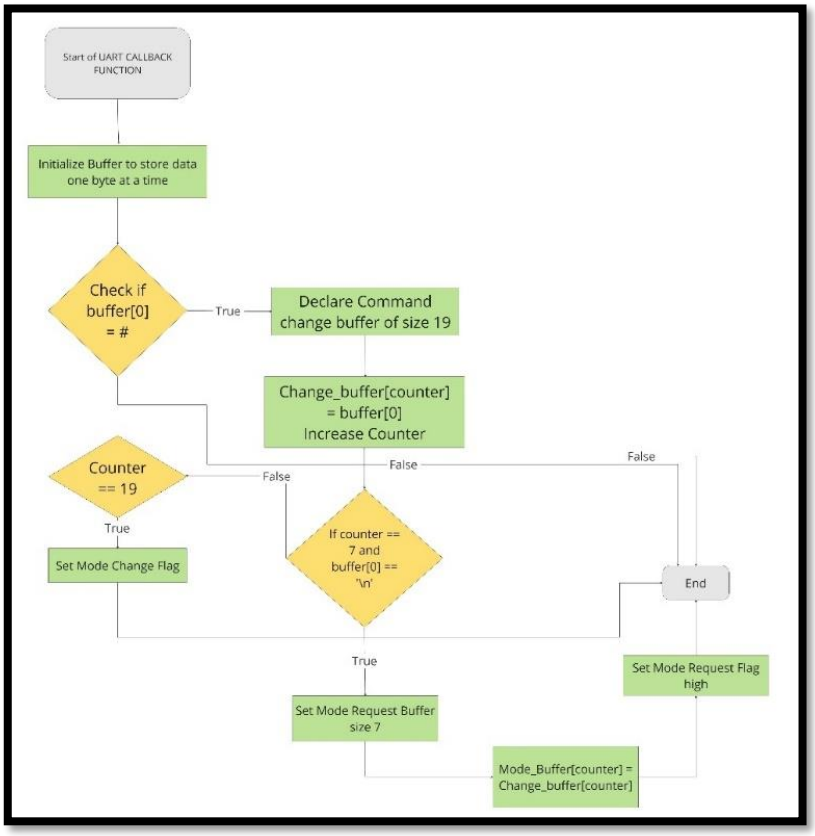


Figure 15: Receive Callback Function through UART

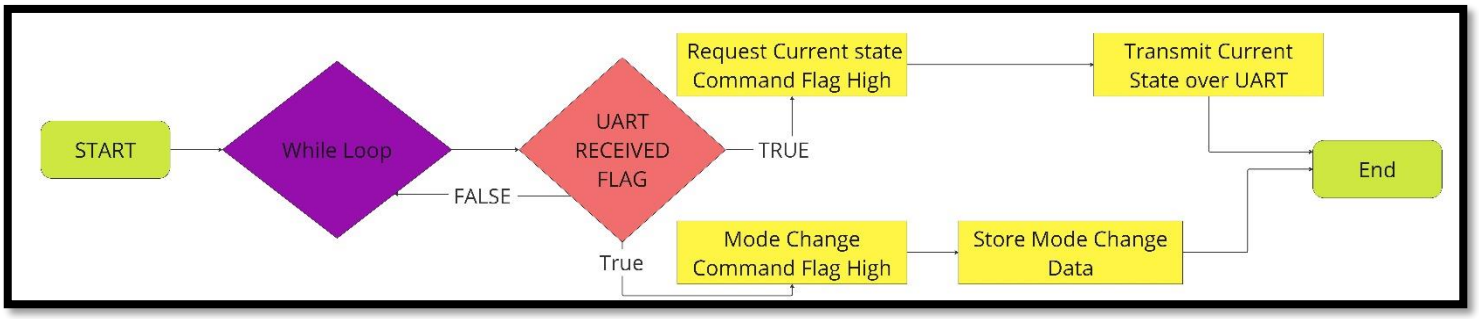


Figure 16: Sending UART Message to TIC

4.3.3 UART SENDING

Figure 16 above illustrates how the system responds to a mode request from a mode state request message. The UART menu function that transmits the current mode state is called within the main while loop. The function makes use of different variable flags to know when to transmit the current mode state over UART. An If statement checks if there is a mode change command flag detected. If the flag is high, the requested mode and parameters are set, and the data is stored within different buffer arrays for each mode and sub-mode. When a request current mode state is detected, the appropriate flag is set high. When the flag is high, the current mode buffer array is transmitted over UART.

4.4 ADC, Setup, Slider calibration and processing

The system uses an analogue slider with an input voltage of 3.3V. The function of the ADC is to control the brightness of the White LED using the input voltage of the analogue slider. The ADC converts the analogue voltage (0-3.3V) received to a digital value.

4.4.1 ADC SETUP

The ADC was configured in the IOC file within STM32CubeIDE. It was configured to convert the analogue voltage using the polling method. The ADC didn't use any DMA functionality. Table 7 below illustrates the configuration of the ADC in the IOC file.

Table 7: ADC1 IOC File Configuration

ADC1:	**IN1 Single-ended**
Resolution:	ADC 12-bit resolution
Mode:	Independent Mode
Sampling Time:	1.5 Cycles
GPIO Pin	PA4

4.4.2 Slider calibration

The digital converted value of the ADC is a value between 0 and 4095. If the slider voltage is at its maximum, the ADC maximum should be close to the value 4095. Because of the non-linearity at the start and end of the slider, the value won't be an accurate representation of the correct voltage. Calibration of the ADC is implemented using software to have an accurate digital value of the original analogue voltage. This calibrated digital voltage, also gets converted to a scaled value of maximum 512. The formulas below illustrate how calibration is done.

- Voltage = (ADC value/4095) * 512
- $Corrected\ Voltage = (voltage * voltage\ gain) + 1$

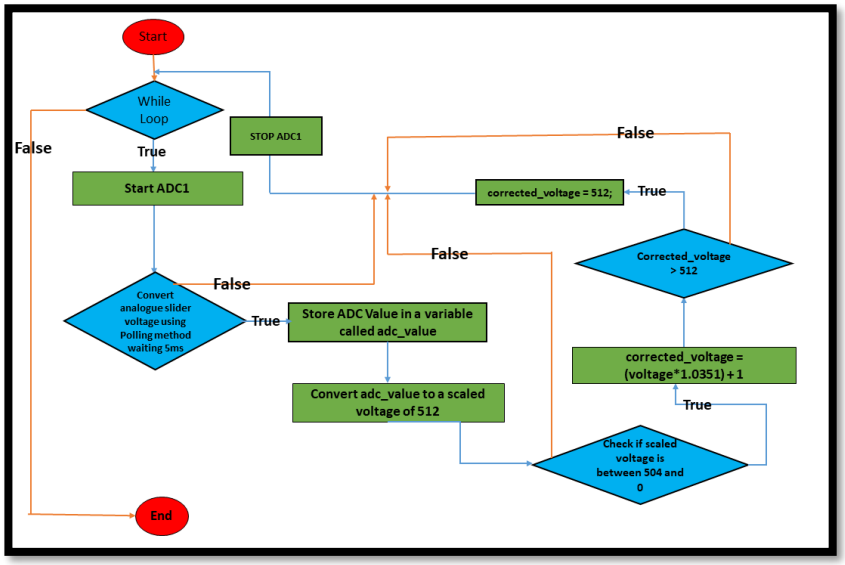


Figure 17: ADC Software Diagram

Figure 17 above illustrates the ADC software implementation to convert the analogue voltage to a scaled digital value, as well as calibration of the ADC. The figure shows that within the while loop the ADC is started. The ADC value of the slider get stored within a uint16_t variable after the ADC1 channel completed a conversion. This variable is then converted to a value between 0 and 512 and stored in a variable called voltage. An If statement checks whether the uncalibrated voltage variable is within a specific range. The ADC gets stopped at the end of each loop and gets restarted every time the ADC code loops again.

Expression	Type	Value
adc_value	uint16_t	4013
voltage	uint16_t	501
corrected_voltage	uint16_t	512
Add new expression		

Figure 18: Calibration of ADC STM32CubeIDE Debug

4.5 DAC, Setup, Calibration, and processing

The system uses DAC to convert digital values from the slider/UART commands and trackpad intensity changes to an analogue voltage. The DAC should convert a digital value to an analogue voltage in Flashlight mode, Strobe mode and Morse mode. The software of the system is developed to change the DAC output according to the current mode of the system.

4.5.1 DAC SETUP

Table 8 below shows the setup of DAC1 in the IOC file within STM32CubeIDE.

Table 8: DAC IOC Setup

DAC1	**DAC1_OUT1**
GPIO PIN	PA4
ALLIGNMENT	12-Bit Right-Aligned

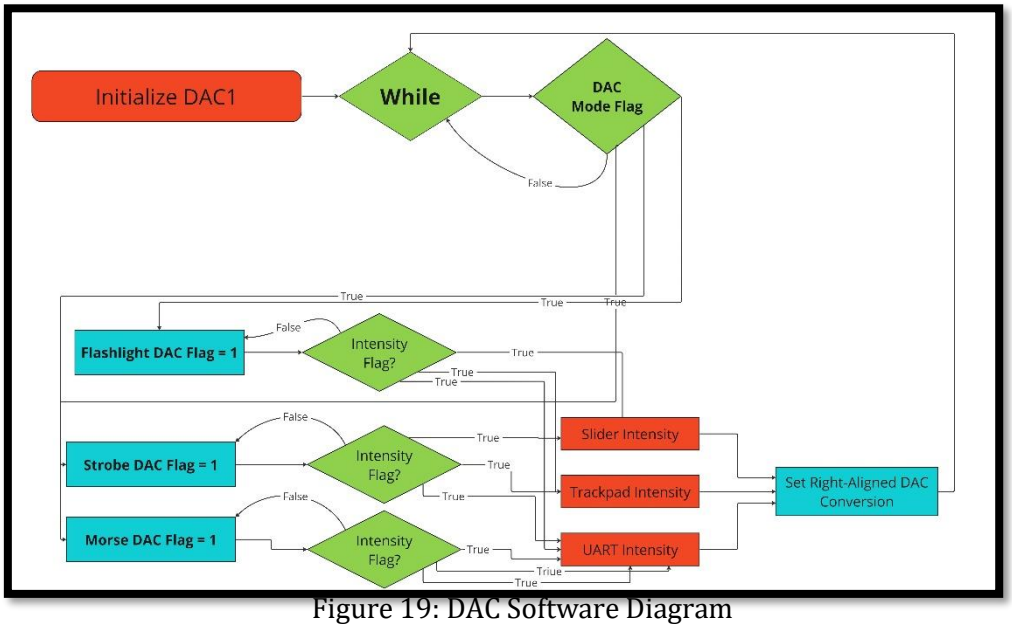


Figure 19: DAC Software Diagram

4.5.2 DAC Processing

Figure 19 illustrates the implementation of the DAC conversion with software. The DAC channel gets initialized before the while loop in the main function. An If statement checks if a DAC conversion flag has been set. Flashlight mode, Strobe sub-mode and Morse sub-mode requires DAC conversions within software. One flag is high at a time, because the DAC function can only perform one conversion at a time. The intensity parameter to be used by DAC for conversion is specified by either the ADC Slider, Trackpad, or a specified intensity command through UART. The specified intensity then gets Right Aligned Converted using the DAC Channel 1. The analogue voltage can then be measured on the GPIO PIN (PA0) and the according TIC pin.

4.5.3 DAC Calibration

The value that the DAC converts should be a value between 0 – 4095. The intensity from the different inputs is a scaled value of maximum 512. Thus, a formula is required to calibrate this to 4095. See the equation below to achieve this. The formula below subtracts one from the DAC value to ensure that the uint8_t variable does not overflow.

$$DAC\ Value = \left(\frac{Intensity}{512} * 4096\right) - 1$$

4.6 PWM, Setup, Calibration, and processing

The STM32F303RE microcontroller has the capability to generate six timers to be used as PWM channels, according to the datasheet [3, p.26]. Three PWM channels were configured within the IOC file, to be used by the three different RGB LED channels and the white LED. The PWM timer drive the white LED high using one PWM channel and the RGB LED high using three different PWM channels each.

4.6.1 PWM Setup

1. Configure Timer2, Timer3 and Timer4 in the IOC file.
2. Set the PWM Generation Channel for each PWM channel.
3. Set the PSC and ARR value for each timer.
4. Configure the clock frequency as 72MHz.

Table 9: RGB Channel PWM Configuration

PWM Timer and Channel	Type of LED	GPIO PIN
PWM TIM3 Channel 1	White LED	PB4
PWM TIM3 Channel 3	Red Intensity Channel	PC8
PWM TIM4 Channel 2	Green Intensity Channel	PA12
PWM TIM2 Channel 2	Blue Intensity Channel	PB11

4.6.2 PWM Calibration

The Prescaler value (PSC) and Auto reload Value (ARR) calculations are required, to make the PWM timer fast enough to avoid any flickering in the white LED. The PSC and ARR should be designed to meet the requirements of the white LED and RGB LED. The formulas below have been used to calculate the ARR and PSC values:

$$F_{Timer} = \frac{FClk}{PSC+1} \qquad Fpwm = Ftimer * \frac{1}{ARR+1} \qquad Duty\ Cycle = \frac{CCR}{ARR+1}$$

Table 10: PWM ARR & PSC Calculations

Clock Frequency	PWM Frequency	Choose timer Frequency	PSC (IOC FILE)	ARR (IOC FILE)	Duty Cycle
72MHz	2KHz	1MHz	71	499	0.50

The PSC and ARR values were set in the IOC file and used for Flashlight mode, Strobe and Morse mode. Thus, the frequency the PWM channel is operating on is 2KHz with a duty cycle of 50%. The frequency is fast enough to eliminate any visible flickering of the white LED.

4.6.3 PWM Processing

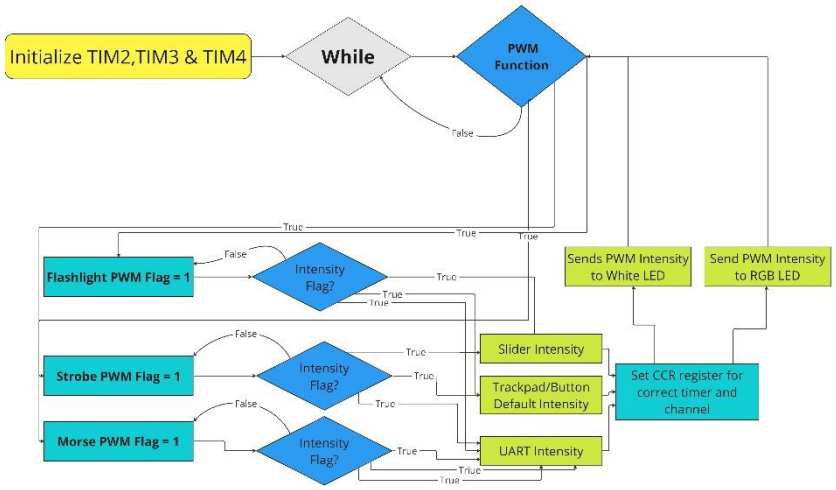


Figure 20: PWM Software Diagram

Figure 20 illustrates the software implementation for the PWM used in the system. The PWM works in a similar way to the DAC software. The function is called within the while loop and different If statements check which PWM flag should be set for each different state. The slider, trackpad and UART commands update the intensity with the most recent input for the white LED. The Capture Compare Register (CCR) for the current system state is set with the correct intensity. The white LED has one Timer and channel to set the intensity in Emergency sub-modes and the Flashlight main mode. The RGB channels have different flags and PWM channels for each intensity colour. The RGB software uses an execution handler to check what Timer and Channel to set for the trackpad and UART commands.

4.7 Trackpad

The main communication method between the STM32F303RE and the IQS7211A trackpad is I2C. The SDA and SCL lines are being used to transmit and receive data bit by bit. The trackpad has a hexadecimal address of 0x56, and this register should be used for writing or reading data. The datasheet [6] contains the different available registers allowed to be read/write for initialization purposes and reading gestures registers. The trackpad has an active-low Ready Line (RDY) window to inform the master (STM32F303RE) that updated data is available, according to the datasheet [6, p.33]. Figure 21 illustrates how the RDY windows works.

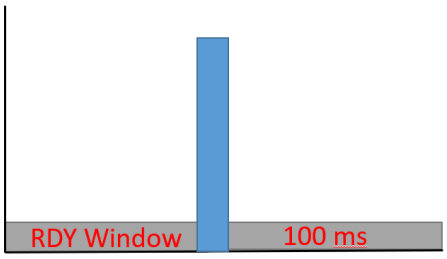


Figure 21: RDY Line

4.7.1 Trackpad Setup

IOC Configurations	SDA (PB7)	SCL (PB6)	MCLR (PB2)	RDY (PB12)
Configuration Type	I2C1_SDA	I2C1_SCL	GPIO Output	GPIO_EXTI12

The IQS7211A initialization functions have been provided by Azoteq in Arduino code format. The initialization functions have been translated into STM32 C to be used for the setup of the trackpad. The initialization process checks specific registers and either read or write to specific bits in the register. Figure 22 illustrates the setup of the IQS7211A trackpad. The figure was provided by an Azoteq slide [8] and modified for the report.

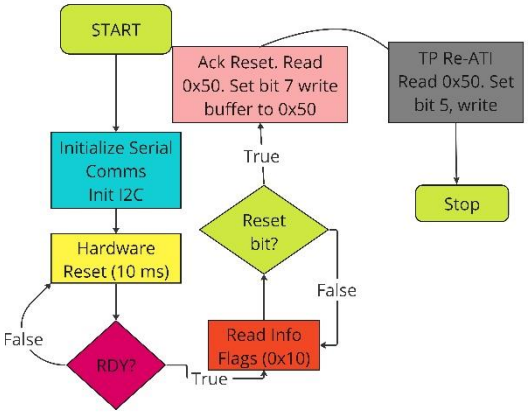


Figure 22: Setup Process of Trackpad

4.7.2 Trackpad Calibration

The only calibration done for the trackpad after the startup functions were to check if the ready line is low whenever data is read or written to the trackpad. The ready line was checked with an If statement, that if the RDY line is high, to make the RDY line low. This ensures that all gestures inputs are received on the master device.

4.7.3 Trackpad Interface

Figure 23 below illustrates the software implementation to update the intensity of the white LED, RGB LED and switching off/on the LEDs using gestures from the trackpad. The software was implemented in a function inside my while loop. The function reads the address 0x11 and checks if different bits are set for each gesture. Slide, tap, tap and hold gestures have its own flag, to be used. Different If statements check whether a gesture is detected

and updates the intensity or switch the LEDs off/on accordingly. The different states update depending on which current state the system is in. Table 11 indicates the bits check for each gesture used in the system.

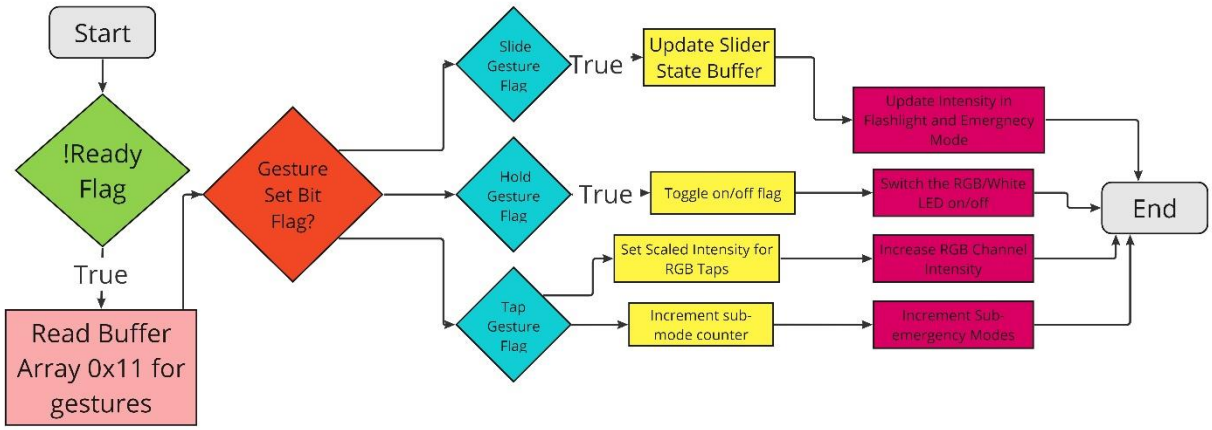


Figure 23: Trackpad Software Interface

Table 11: Check Trackpad Gesture Bits

Gesture	Binary Value
Hold	0000 0010 (Check Bit 2)
Tap	0000 0001 (Check Bit 1)
Slide Negative X Direction	0000 0100 (Check Bit 3)
Slide Positive X Direction	0000 1000 (Check Bit 4)

4.8 Debug + System State Indication Logic

Debug LEDs are the main visible state indicators used in the system to indicate the current system state. Four debug LEDs indicate different states in the system. LED1 (D2) turns on to indicate when the system is in Flashlight mode. LED2 (D3) goes high when the system changes to Emergency mode and LED4 (D5) is only high when the current state is Morse/Custom sub-mode. LED3 (D4) is only being used when the current mode is mood mode.

4.8.1 Development of Debug LEDS & State Logic

1. Configure each LED in the IOC file as GPIO Output
2. Set a flag for each state and the according LED being used.
3. Use a software LED/State Handler to switch between different logic states.
4. Use HAL Libraries to set the correct LED high after each new state change.

Figure 24 below illustrates how the software was developed for the Debug LEDs and state logic.

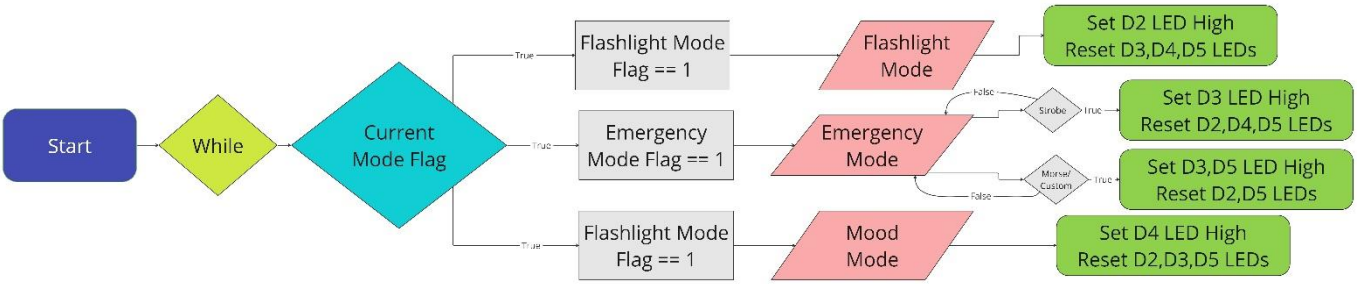


Figure 24: Debug LED and State Logic Diagram

5 Measurements and Results

5.1 Power Supply

The power supply was tested and measured using an oscilloscope and multimeter to verify that the 5V power supply and 3.3V power supply were working as specified in section 3.1. The input voltage was supplied by a lab power supply and a range of input voltages were tested. The 5V LM7805 regulator was measured and tested first. See below in table 12 what the measurements were.

Table 12: 5V Power Supply Measurement

VInput (DC JACK)	LM7805 Oscilloscope	LM7805 Multimeter
9V	5.12V	5.01V
7V	4.80V	4.80V
6V	3.84V	3.84V

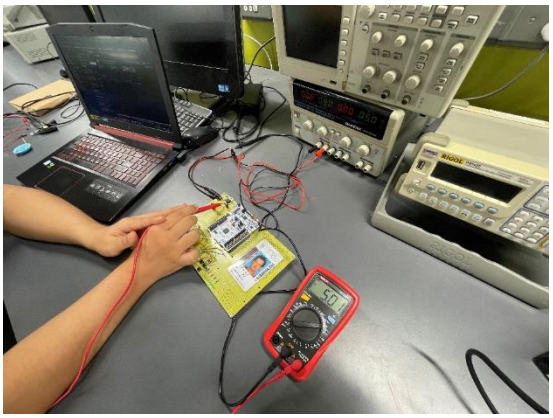


Figure 25: Multimeter measurement of 5V

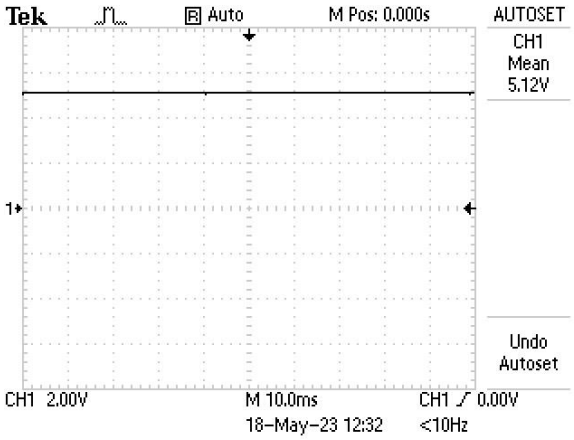


Figure 26: Oscilloscope measurement of 5V

As can be seen at 9V the power regulator regulates the voltage to 5V, but from 7V the voltage starts to drop and continues dropping till it reaches zero. This is as expected, and the measurements show that the power supply works with the 9V input. The power regulator can regulate any voltage between 7V and 35V, according to the datasheet [7, p.5]. See figure 27 for a graph of how the voltage drops if the input voltage reaches 7V or lower.

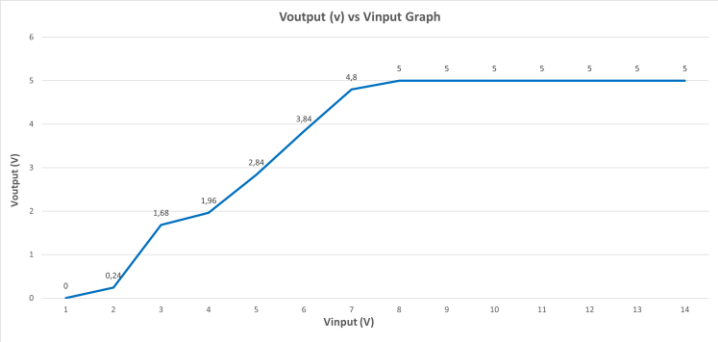


Figure 27: Graph of effectiveness of Voltage Regulator

The 3.3V LM2940 was tested the same way as the 5V voltage regulator. The output was tested to verify if the LM2940 stepped the voltage down to 3.3V. Table 13 below shows the input voltage supplied and what the output voltage of the LM2940 regulator is.

Table 13: 3.3V Power Supply Measurement

V_{Input} (LM7805 Output)	V_{Output} (LM2940 Output) Multimeter	V_{Output} (LM2940 Output) Oscilloscope
5V	3.3V	3.3V

As can be seen from the table above, the voltage regulator works as intended and the output voltage is close the desired value of 3.3V. The oscilloscope measurement is only higher, but close to 3.3V. See below in figure 28 and figure 29 the measurements.

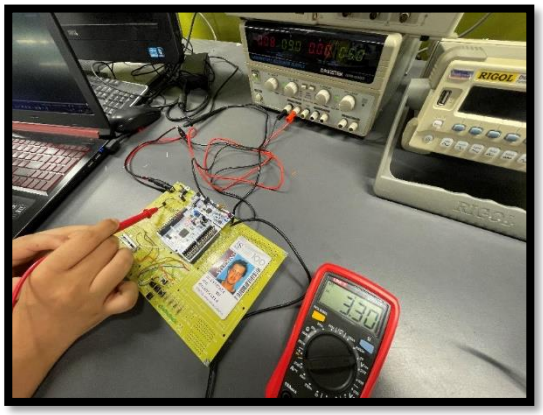


Figure 28: Multimeter Measurement of 3.3V Regulator

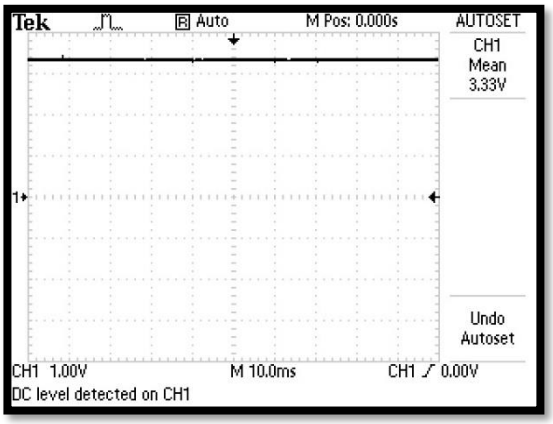


Figure 29: Oscilloscope Measurement of 3.3V Regulator

5.2 UART communications

UART communication was measured to verify if the software implementation of UART is effective. The UART was tested by measuring the startup student number message. An Oscilloscope probe was connected to the Rx Pin of the nucleo-F303RE board and to a common ground. The student number message should be "#:23795824:$\n" on the oscilloscope. The oscilloscope measurement transmit the student number with the least significant bit first, thus UART uses Little Endian Configuration. Figure 31 shows the decoding of the first two bytes of the student number message. Table 14 illustrates the analysis of the bytes using an ASCII table.

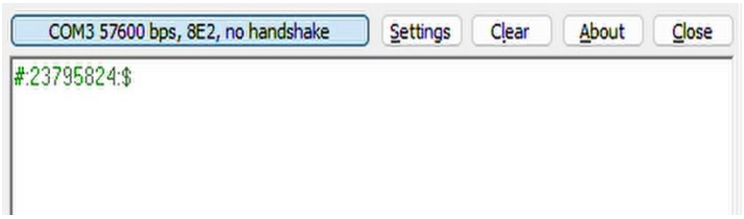


Figure 30: Termite Transmission of Student Number

Table 14: Decoded UART Transmission

Byte 1: (8E2)	**011000100111**
Data:	001000110
ASCII:	'#'
Byte 2: (8E2)	001011100011
Data:	00111010
ASCII:	':'

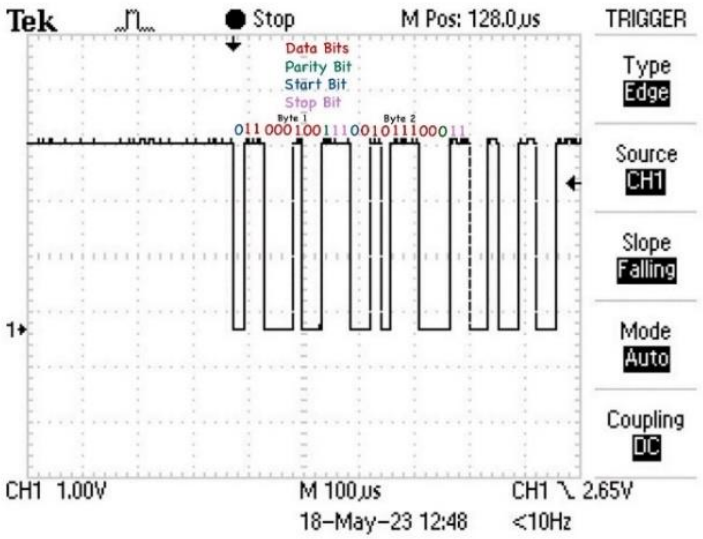


Figure 31: Student Number Measurement

5.3 Buttons

The buttons were implemented as discussed in section 3.3. It was tested to confirm if it's configured as active-low and that the button debounce software implementation works.

5.3.1 Setup and Measurements

- Measure the voltage over the external resistor when a button is pressed with a multimeter.
- Measure the voltage over the external resistor using an oscilloscope.

Table 15: Button Measurements

Theoretical Button Calculations	Practical measure values
Voltage over resistor: 3.3V	**Voltage over resistor:** 3.29V
Max current: 0.33 mA	**Max current:** 0.329 mA

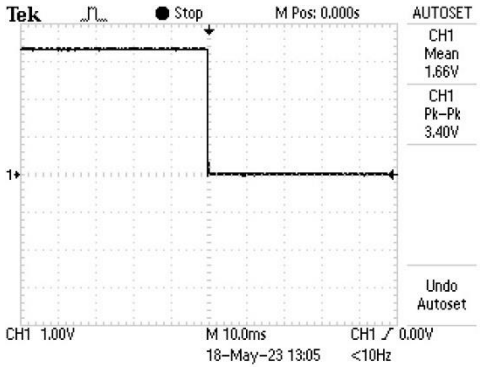


Figure 33: Active Low Oscilloscope Measurement

Figure 32: External Button Resistor Measurement

Figure 33 illustrates the active-low configuration. When the button is not pressed, the voltage is 3.3V. However, upon pressing the button, the voltage decreases to 0V. This agrees with the design choice specified in section 3.3. The measured current flowing through the GPIO Pin matches the theoretical calculated current. The difference between the measured value and theoretical value is negatable.

5.3.2 Button Bounce

The push buttons are mechanical switches that have button bounce, which causes the buttons to bounce multiple times. The buttons were tested for button bounce, by connecting a 1kΩ series resistor to the middle button (S3) and building it on a breadboard. An oscilloscope measurement was used to measure the amount of button bounce in seconds. Figure 34 illustrates the circuit, and the measurement of the bounce is illustrated in figure 35.

Figure 34: Button Bounce Circuit

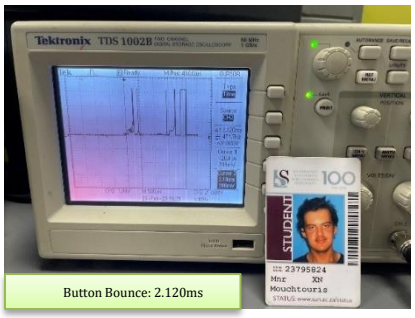


Figure 35: Bounce Measurement

The above bounce measurement was measured with a scale of M 500 µs and the total bounce measurement was ± 2.120 ms. This button bounce measurement is relatively low, compared to the button debounce elapse time used in software. Trial and error in software was used to eliminate all bounce, but the measured button bounce was used as a reference point.

5.4 Slider/ADC

The slider in the system was measured to verify that all requirements specified were achieved. The analogue voltage was measured at the output pin of the slider at three different positions using a multimeter and oscilloscope. The voltage was measured at a minimum position, middle position, and maximum position, to test if the slider works as a voltage divider. Table 16 below indicates the position relative to the voltage measured and verifies that the slider works as a voltage divider.

Table 16: SLIDER Measurements

SLIDER POSITION (0%)	Multimeter Voltage	Oscilloscope Voltage
0%	0.00V	832µV
50%	1.60V	1.64V
100%	3.29V	3.33V

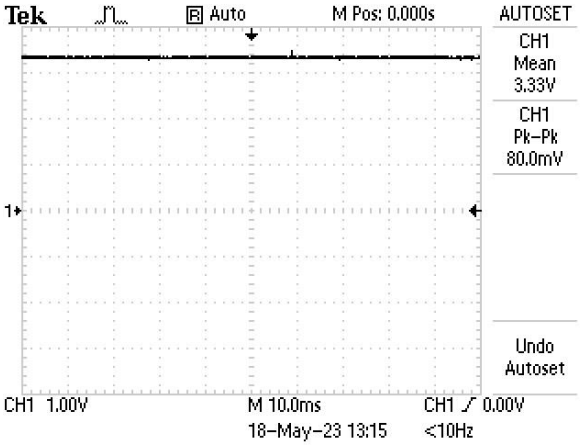


Figure 37: Maximum Slider Output Voltage

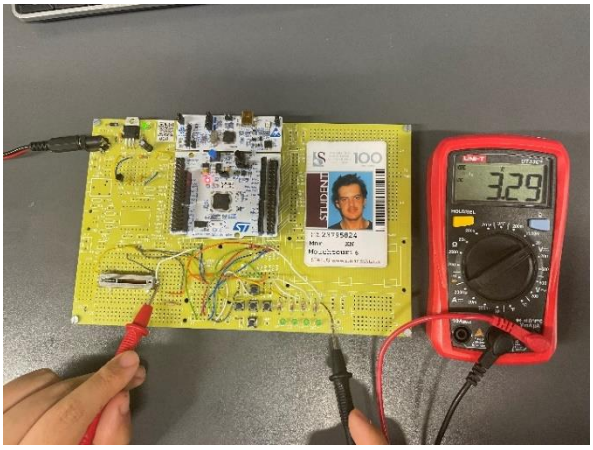


Figure 36: Maximum Slider Output with Multimeter

The calibration of the ADC was measured, to verify that the software calibration worked as discussed in section 4.4.2. The ADC input pin (PA0) was measured using a multimeter to compare the maximum, minimum and middle values. The measured values were then compared to the ADC Calibrated digital value read in "Live Expressions" in STM32CubeIDE. Calibration is sufficient when the analogue voltage matches the digital scaled value. Table 17 below indicates that calibration worked and that the digital scaled value a faithful representative of the original signal is.

Table 17: ADC Calibration Measurements

SLIDER MEASURED	SLIDER DEBUG VALUE	ADC INPUT MEASURED	ADC INPUT DEBUG
MIN: 0.0 V	**MIN:** 0.0 V	**MIN:** 0	**MIN:** 1
MAX: 3.29 V	**MAX:** 3.289 V	**MAX:** 4095	**MAX:** 4095
MIDDLE: 1.60 V	**MIDDLE: 1.5984 V**	**MIDDLE: 1985**	**MIDDLE: 1983**

5.5 DAC

The DAC GPIO Pin (PA4) was measured using an Oscilloscope Probe. The output analogue voltage of the DAC was measured and compared with the input digital value. The measurements are being used to verify if the DAC according to changes being made to the intensity, by the slider, trackpad and UART commands.

5.5.1 Setup and Measurement

- A termite command requesting the system to change to Flashlight Mode with a scaled intensity value of 123.
- The DAC was measured to verify that the scaled digital '123' voltage matches the equivalent analogue voltage.

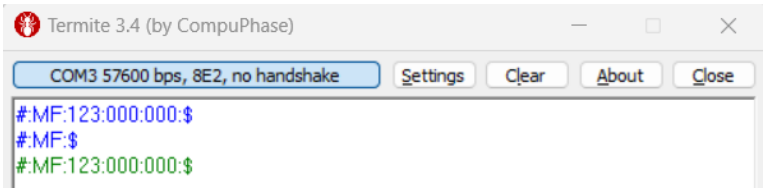


Figure 38: Termite Requested Intensity

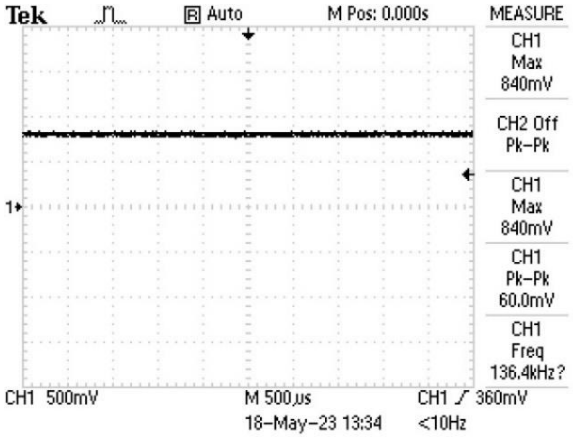


Figure 39: DAC Termite Output Measurement

The analogue voltage measured in figure 38 is close enough to the requested intensity. The requested voltage should have a theoretical analogue voltage of 793mV. The measured value of 840mV is almost identical to the calculated voltage, thus the DAC and calibration works sufficient.

5.6 PWM

The PWM timers and channels were measured to verify that it worked as designed in section 4.6.3. Because PWM is a changing output single with a frequency of 2KHz the output signal fluctuates a lot. Thus, measuring the PWM GPIO Pins with a multimeter wouldn't work because the multimeter isn't fast enough to calculate the PWM signal accurately. An Oscilloscope probe with an Oscilloscope was used to measure the PWM signal.

5.6.1 Setup and Measurements Steps

1. Connect the Oscilloscope Probe to Tim3 Channel 1 (PB4)
2. Send a termite command request for Flashlight mode with an intensity of '300'.
3. Send a termite command request for Strobe sub-mode with a frequency of '300' ms and '512' intensity.

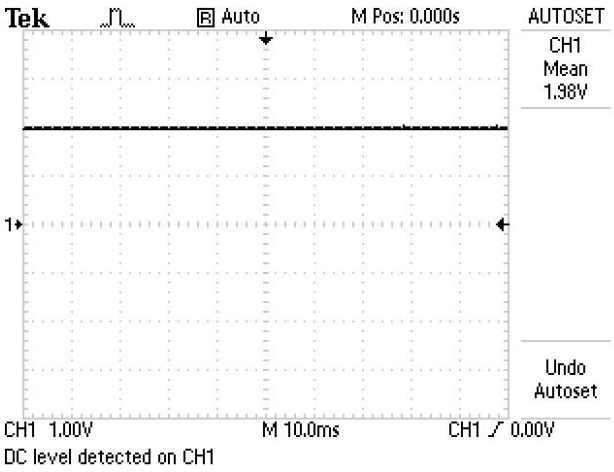


Figure 41: Flashlight Mode with intensity of '300'

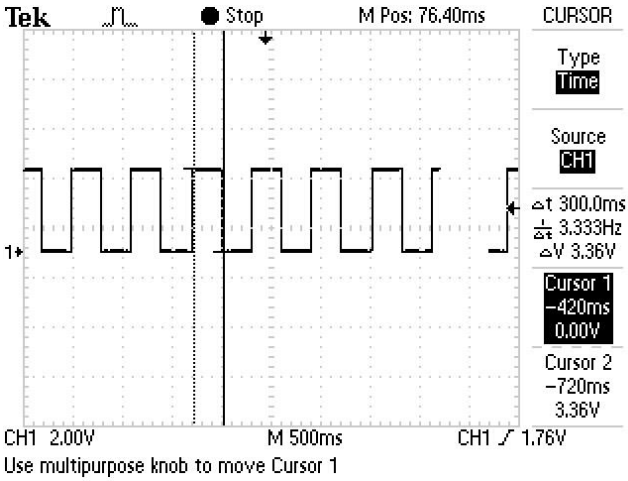


Figure 40: Strobe Mode with 300ms frequency

Figure 40 and Figure 41 above illustrate that the PWM works. When in Flashlight mode like in Figure 41 the PWM should be a constant Flashlight intensity at the specified intensity. The theoretical calculated intensity for an intensity of '300' is 1.934 V. This value is close enough to the 1.98V displayed on the oscilloscope. This means that the logic for flashlight mode and the PWM works. Figure 40 illustrates a termite command for Strobe sub-mode with an intensity of '512' (full-range) and a frequency of '300'. The oscilloscope displays that the on time and off time is 300ms which matches the specified frequency. Thus, the duty cycle for the PWM is still at 50% and strobe mode is changing to the correct frequency using the PWM channel.

5.7 Current Amplifier

The current amplifier transistor circuit discussed in section 3.8, was tested to verify that the current amplifier works as intended. The voltage drop over the base and collector were measured and the current for each were calculated.

5.7.1 Setup of Measurement

1. Turn the White LED on with a tap and hold or middle button press.
2. Measure the voltage drop over the collector resistor with a multimeter and calculate the current through the white LED.
3. Measure the voltage drop over the base resistor with a multimeter and calculate the current.

5.7.2 Measurements of Amplifier Circuit

Table 18:Amplifier Circuit Current Measurements

White LED	Collector Voltage	Base Voltage	Collector Current	Base Current
Measurements	1.66V	2.56V	29.64 mA	0.3122 mA

The above current calculations in table 18, have been calculated using the formulas discussed in section 3.8. The current amplifier works as designed, because the measured current and voltages is close to the theoretical calculated current and voltage. The measured collector current is higher than the design value but does not impact the effectiveness of the amplifier circuit. The base current is lower than the calculated value, because a higher 10% value resistor of 8.2 KΩ has been used. The base current is lower than the collector current, because the base current is used to turn the transistor on. This current then gets amplified to turn the White LED on. The measurements prove the white LED works as intended. See the measurement below in figure 42 and figure 42.

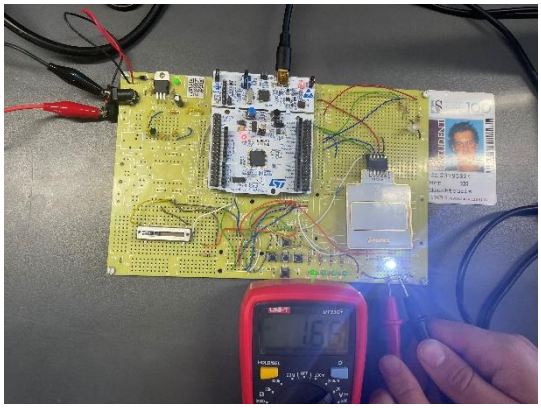


Figure 43: Collector Voltage Drop: 1.66V

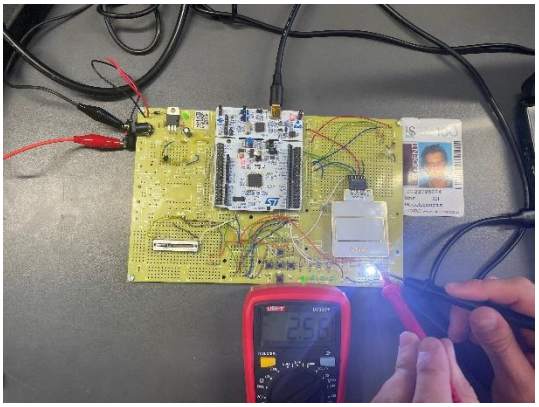


Figure 42: Base Voltage Drop: 2.56V

5.8 Azoteq Trackpad

The Azoteq Trackpad was tested to verify that communication between the STM32F303RE and the trackpad is taking place. An oscilloscope probe was used to measure different pin connections between the trackpad and the STM32F303RE. The Ready Line (RDY) was tested to see if the ready line window is ± 100ms. The I2C transmission of Serial Data (SDA) and Serial Clock (SCL) was measured for a tap and hold gesture on the trackpad.

5.8.1 Trackpad Setup:

1. Connect the oscilloscope to the Ready Line (RDY) pin (PB12).
2. Connect two oscilloscope probes to the SDA (PB7) and SCL (PB6) to measure a tap and hold I2C transmission.

Figure 45 illustrates the ready line going low after a ready line window. The period of the ready line is 102.2 ms, which matches the 100 ms ready line window requirement. A tap and hold I2C transmission are illustrated with a SDA and SCL oscilloscope measurement in figure 44.

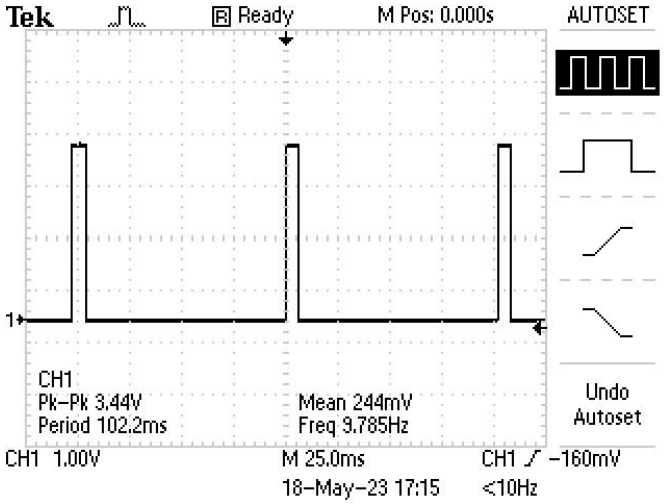


Figure 45: Ready Line (RDY) Measurement

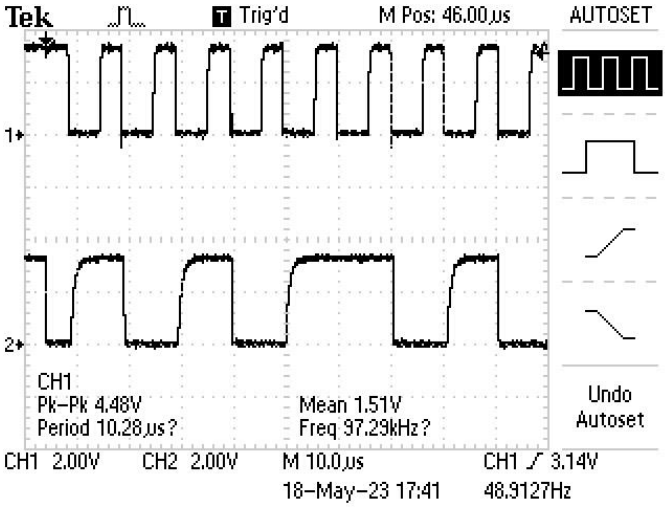


Figure 44: I2C Tap & Hold Transmission

5.9 Output + Debug LEDs

5.9.1 Debug LEDs

The Debug LEDs were implemented as discussed in section 3.2. The theoretical and measured values for the debug LEDS are listed in table 19 below:

Setup:

- Measure the voltage over the LED/Resistor using a Multimeter.

Table 19: Debug LED Meaurements

LEDs	Theoretical Values	Measurement Values
Voltage Drop Over resistor	1.2 V	1.37V
Voltage Drop Over LED	2.1 V	1.90V
Current through resistor	1.2 mA	1.37 mA

The voltage drops over the resistors is a bit higher than the theoretical calculations and the voltage drop is lower over the LEDs. The current is 1.37 mA which is also higher than calculated. These values are not a problem, because it much lower than the maximum rated current for the GPIO Pin. Thus, the 1000 Ω resistor works sufficiently to limit the current.

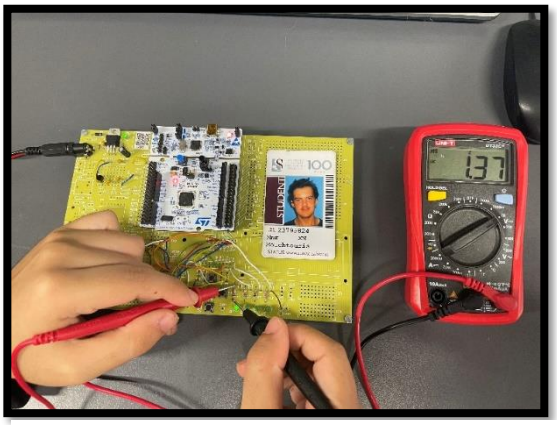


Figure 47: Voltage drop over 1kΩ resistor

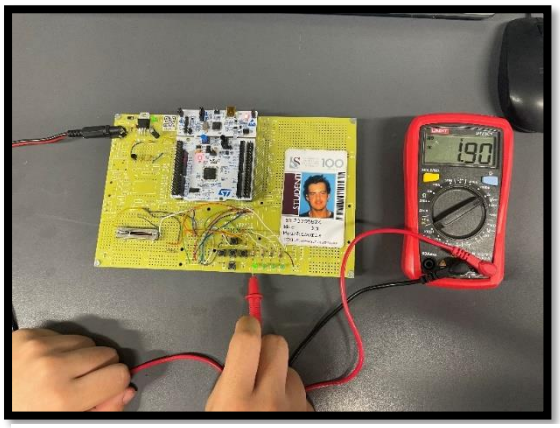


Figure 46: Voltage drop over Debug LED.

5.9.2 Output LEDs (RGB and White LED)

The Output LEDs were measured using a multimeter to validate the theoretical voltage and current calculations. As discussed in section 5.7.2 the white LED worked and the current and voltage measurements were similar to the theoretical calculations. The RGB LED was measured in the same way, using a multimeter to measure every colour intensity of the RGB LED and verifying it with theoretical calculations.

Table 20: RGB LED Measurements

RGB LED	Voltage over Resistor [V]	Current [mA]
RED Channel	1.33V	6.045 mA
GREEN Channel	0.99V	0.367 mA
BLUE Channel	0.52V	1.33 mA

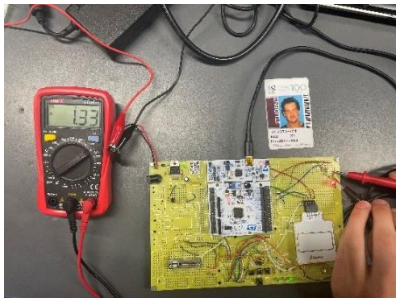


Figure 49: Voltage for RED Channel

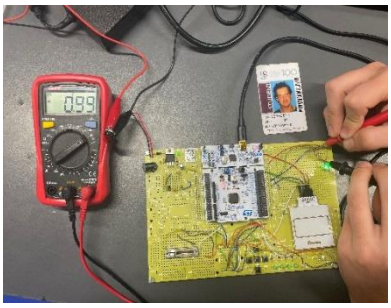


Figure 50: Voltage for Green

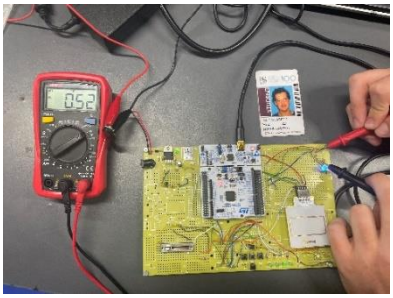


Figure 48: Voltage for Blue

Table 20 above indicates the voltage values for the different channels in relation to each channel current. The voltage and current values differ a bit from the theoretical calculations because the original calculated resistor values were not used. Higher resistor was used to achieve the correct intensity for each channel colour. The green channel used the highest resistor value of 2.7 KΩ and thus the current is the lowest at 0.367 mA. The red channel is the closest to the original current and voltage value, because the resistor value used did not change a lot. The blue channel used a higher resistor value than calculated and thus the lower current. Overall, the measurements are as expected with the higher resistor values used, and above in figure 48, figure 49 and figure 50 can be seen that the intensity of the RGB LEDs is as expected.

5.10 Complete System

The complete system measurement was a test to verify that when a termite command is set, that the system changes to the desired state and turn the Output LED on.

5.10.1 Setup

1. A termite command requested a mode change to Flashlight mode with Intensity of 512.

Table 21 below illustrates the requirements set and if the system achieved the requirements set.

Table 21: Complete System Test

System Requirement	Pass/Failed
Flashlight Mode	Pass
Debug LED D2 On	Pass
Full Range Intensity for White LED	Pass
Non-Flickering	Pass
RGB LED Off	Pass

See below figure 51 for the requested mode and the white LED switched on with full range intensity.

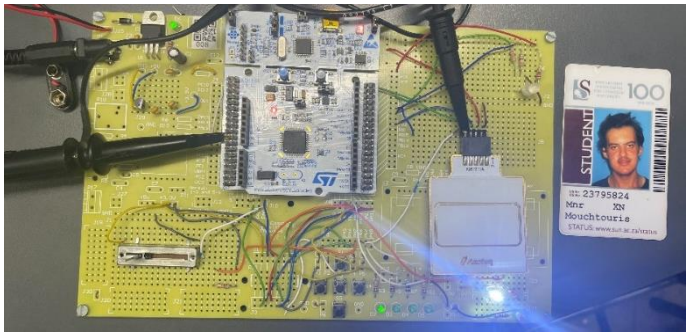


Figure 51: Flashlight Mode with Full Intensity

6 Conclusions

The system was overall a success, and all the hardware design choices were implemented in the final design of the multi-functional light source. The software design discussed in section 4 were followed as well and most software design choices were achieved as discusses. The system worked well, with minor bugs, but there are not major problems in the system.

6.1 Non-Compliance

Although the software design implementation of the slider gesture for the trackpad were discussed in section 4.7.3., the final system did not implement this feature. The software to implement this feature were written, but the code was commented out before demo 4, because there were problems with the slide gesture causing bugs with the UART receive and transmit menu. The slide gesture was very sensitive and gestures would be detected without any slide input from a user. This problem would cause system state command changes to be ignored, because a random slide would be detected. For this reason, the system did not implement any slide code in the final version of the software.

6.2 Design Shortcomings

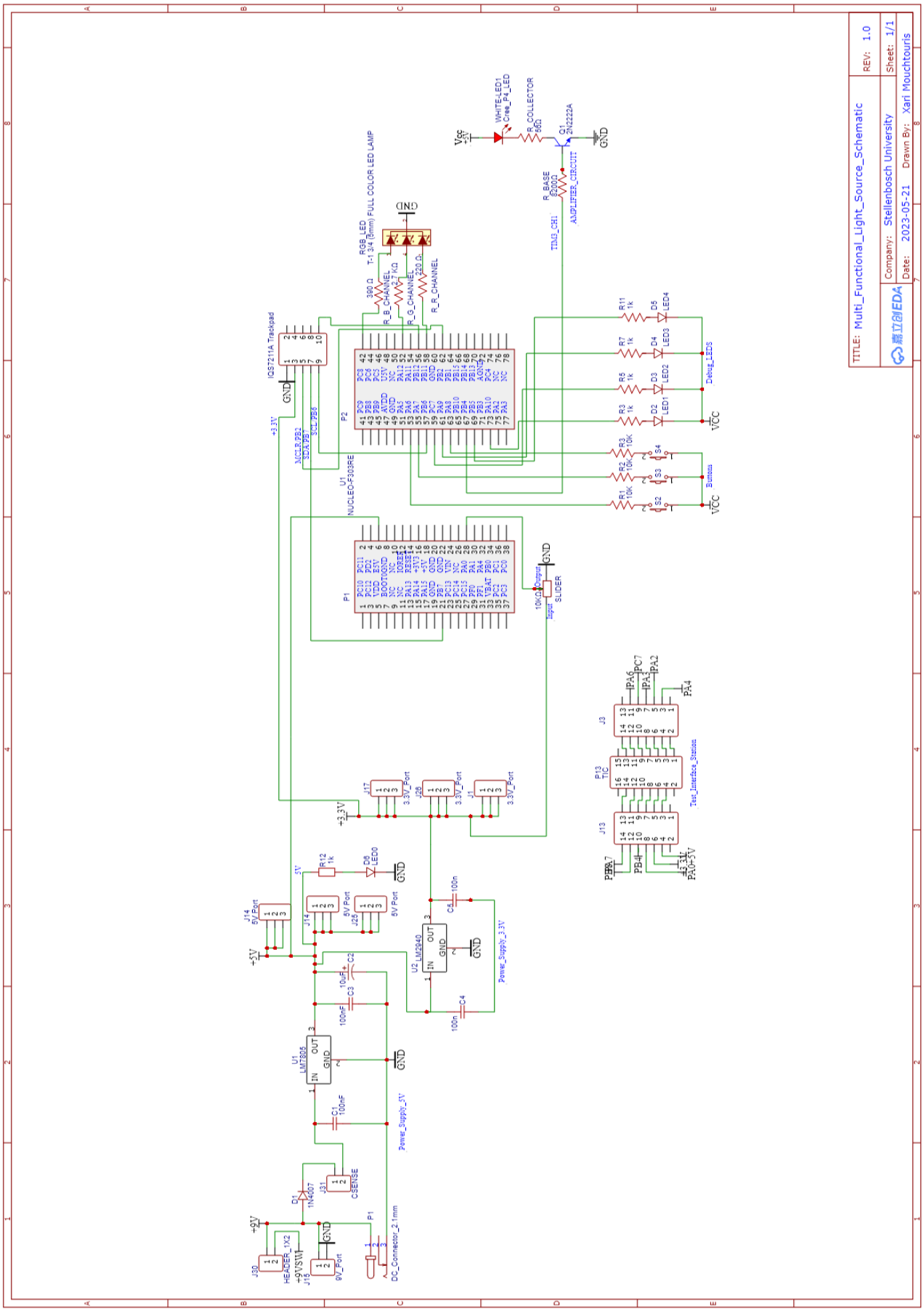
The hardware design choices were all implemented as discussed in section 3 of the report. All the discussed hardware and components worked. The only design shortcomings were in the software design. As discussed in section 6.1. the slide gesture of the trackpad was not used. This shortcoming causes the system not to be able to set the intensity through the slide of the trackpad. Although DAC and PWM was implemented in software, there still were problems with the timing of both. The measurements of both matched the requirements set by the PDD [1], on the Oscilloscope, but the timing was wrong on the TIC. Flashlight mode for both DAC and PWM worked, but Strobe and Morse had the wrong timing. This is a shortcoming, because the software implemented did not worked when it needed to on the TIC. The system also did not pass the UART stress test, which means the UART design menu is to slow, for fast rapid commands.

6.3 Possible Improvements

The system met most of the requirements, but improvements can be made for future implementations. The software should be written in a more modular way, because currently it is very limited. The slide gesture for the trackpad should be implemented in future implementations of the system. The UART menu should be changed to be able to handle fast rapid UART commands. These are some of the elements that can be changed to improve the system in the future.

7 Appendixes

7.1 Complete Schematic of Multi-Functional Light Source



7.2 STM32F303RE GPIO PIN CONFIGURATION

GPIO PIN:	FUNCTION:	USE CASE:
PA0	**ADC1_IN1**	**SLIDER**
PA2	**USART_TX**	**UART**
PA3	**USART_RX**	**UART**
PA4	**DAC1_OUT1**	**TIC Verification**
PA6	**GPIO_EXTI6**	**Left Button**
PA7	**GPIO_EXTI7**	**Middle Button**
PA8	**GPIO_EXTI8**	**Right Button**
PA9	**GPIO_Output**	**Debug LED D4**
PA10	**GPIO_Output**	**Debug LED D2**
PA12	**TIM_CH2**	**Green RGB PWM Channel**
PB2	**GPIO Output**	**MCLR Line**
PB4	**TIM3_CH1**	**White LED PWM Channel**
PB5	**GPIO_Output**	**Debug LED D5**
PB6	**I2C1_SCL**	**I2C Trackpad**
PB7	**I2C1_SDL**	**I2C Trackpad**
PB11	**TIM2_CH4**	**Blue RGB PWM Channel**
PB12	**GPIO_EXTI12**	**Ready Line (RDY)**
PC7	**GPIO_Output**	**Debug LED D3**
PC8	**TIM3_CH3**	**Red RGB PWM Channel**

8 References

[1] Barnard, A. and Grootboom, I., "DesignE314_2023_PDD_v0.4", [Online], 14 Feb. 2023 [Revised 9 Mar 2023]. Available: https://learn.sun.ac.za/pluginfile.php/4065498/mod_resource/content/0/DesignE314_2023_PDD_v0.4.pdf

[2] Multicomp. (2012, October 10). LED Orange/Green, 3mm [online]. Available: https://www.farnell.com/datasheets/1671514.pdf

[3] ST. (2016, October). STM32F303xD STM32F303xE [online]. Available: https://www.st.com/resource/en/datasheet/stm32f303re.pdf

[4] Kingbright. (2013, May, 20]. T-1 3/4 (5mm) FULL COLOR LED LAMP [online]. Available: https://learn.sun.ac.za/pluginfile.php/4089415/mod_resource/content/0/0900766b8139d70e_RGB.pdf

[5] CDIL. NPN SILICON PLANAR SWITCHING TRANSISTORS [online]. Available: https://learn.sun.ac.za/pluginfile.php/4089414/mod_resource/content/0/2n2221a_22a.pdf

[6] Azoteq. (2022, February). IQS7211A Datasheet [online]. Available: https://learn.sun.ac.za/pluginfile.php/4077755/mod_resource/content/0/IQS7211A_Datasheet.pdf

[7] ST. (2006, August). L7800 series [online]. Available: https://learn.sun.ac.za/pluginfile.php/3752811/mod_resource/content/2/STMicroelectronics-L7805CV-datasheet.pdf

[8] Azoteq. [online]. Available: https://learn.sun.ac.za/pluginfile.php/4081250/mod_resource/content/0/Azoteq%20introduction_Stellenbosch_20230418.pdf

[9] HTC. (2015, December). LM2950 [online]. Available: https://learn.sun.ac.za/pluginfile.php/3752812/mod_resource/content/3/LM2950.pdf