ELECTRICAL & ELECTRONIC ENGINEERING
STELLENBOSCH UNIVERSITY

Design (E) 314
Technical Report

Digital Multimeter and Signal Generator

Author:
Kamogelo Malatsi

Student Number:
21809453

May 27, 2022

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.

MALATSI.KD	21809453
Handtekening / *Signature*	Studentenommer / *Student number*
KD MALATSI	27/05/2022
Voorletters en van / *Initials and surname*	Datum / *Date*

Contents

List of Figures

List of Tables

List of Abbreviations

LED - Light Emitting Diode
PCB - Prototyping Circuit Board
HAL - Hardware Abstraction Libraries
GPIO - General purpose input/output
MCU - Microcontroller unit
TIC - Testing Interface Connector
TTL - Transistor - Transistor Logic
UART - Universal Asynchronous Receiver and Transmitter
TS - Test Station

1 Introduction

We must create a digital multimeter and signal generator as part of the Design(E) 314 module project. An STM32F303RE - Nucleo64 microcontroller will be used to measure DC/AC voltages and currents, as well as to generate signals using peripherals such as Timers, DAC, ADC, and I2C. Design considerations for software and hardware implementation will be made in order to create a correct and fully functional system. The following are the report's objectives:

- To configure buttons to be active low
- Implement buttons to be active low
- Implement UART protocol to receive command messeges and respond accordinly.
- Perform Signal generation using DAC with a sensible filter.
- Perform voltage measurement using ADC
- Implement middle push button to go in and out of menu state
- Implement LCD with good contrast and backlight.

2 System description

The system diagram is shown in Figure 1.

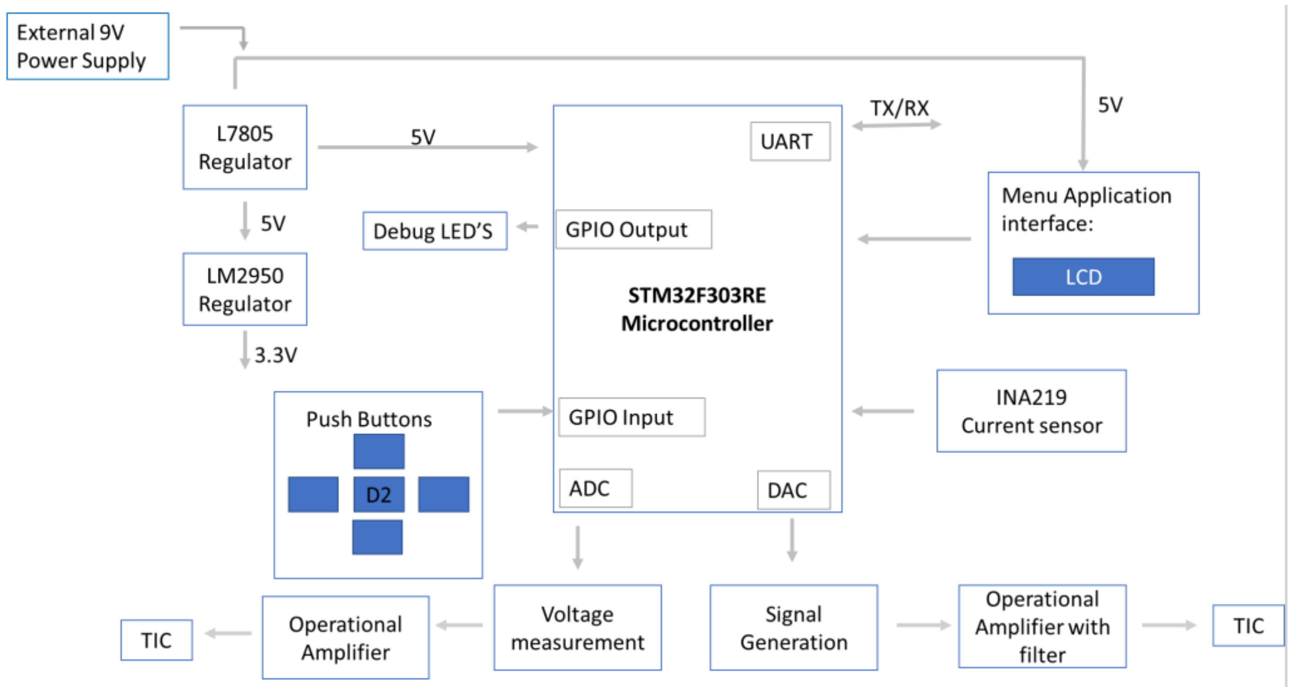


Figure 1: Block diagram of the system.

Power Supply: According to the design specifications, the board requires a nominal 9 V power source. Power will be regulated to 5V using a standard 7805 regulator and regulated to 3.3 V using the LM2950 regulator, to power various components on the baseboard. An LED (D6) circuit is provided to demonstrate the presence of a voltage on the 5 V line.

Microcontroller: A STM32F303RE microcontroller is used in this project. It has a NUCLEO-64 modular design, which allows it to stack via the Arduino headers or the Morpho connectors. The STM board is stacked on top of the prototyping baseboard. The microcontroller has numerous peripherals that will be used in this project, including ADC, DAC, UART and GPIO pins, among others. The microcontroller is powered by a 5V supply.

Menu Application Interface: The user interface controls the system's operation and displays information to the user on an LCD screen. The user interface is controlled by five push buttons. The buttons are configured to be active low. As the system's modes change and the user interacts with the system via the buttons and menu, the LCD data will change.

Debug LED's: For display and/or debug purposes, four debug LEDs and series resistors are used. The debug LEDs will be used to indicate the state of the system.

Voltage measurement: The system measures the voltage of a signal applied to the system's input port. Using the ADC input, the system measures both a DC and an AC voltage. Where voltage range at the ADC input is between 0 V and 3.3 V.

Current measurement: The system measures the current of a signal applied to the system's input port. Using the INA219 current sensor, the system measures both DC and AC current. The system will measure the current with the supplied current sensor module and send the result to the microcontroller via I2C.

Signal generation: The system generates a DC or AC voltage signal and sends it to the system's output port. The output signal can be set to be DC voltage, sinusoidal, or pulse waveform. The operational amplifier's design is determined by the output signal range between 0.1 V and 3.2 V.

3 Hardware design and implementation

The following sections describes the design motivations, calculations and implementation,using equations where applicable. The complete hardware block diagram is shown in Figure 2

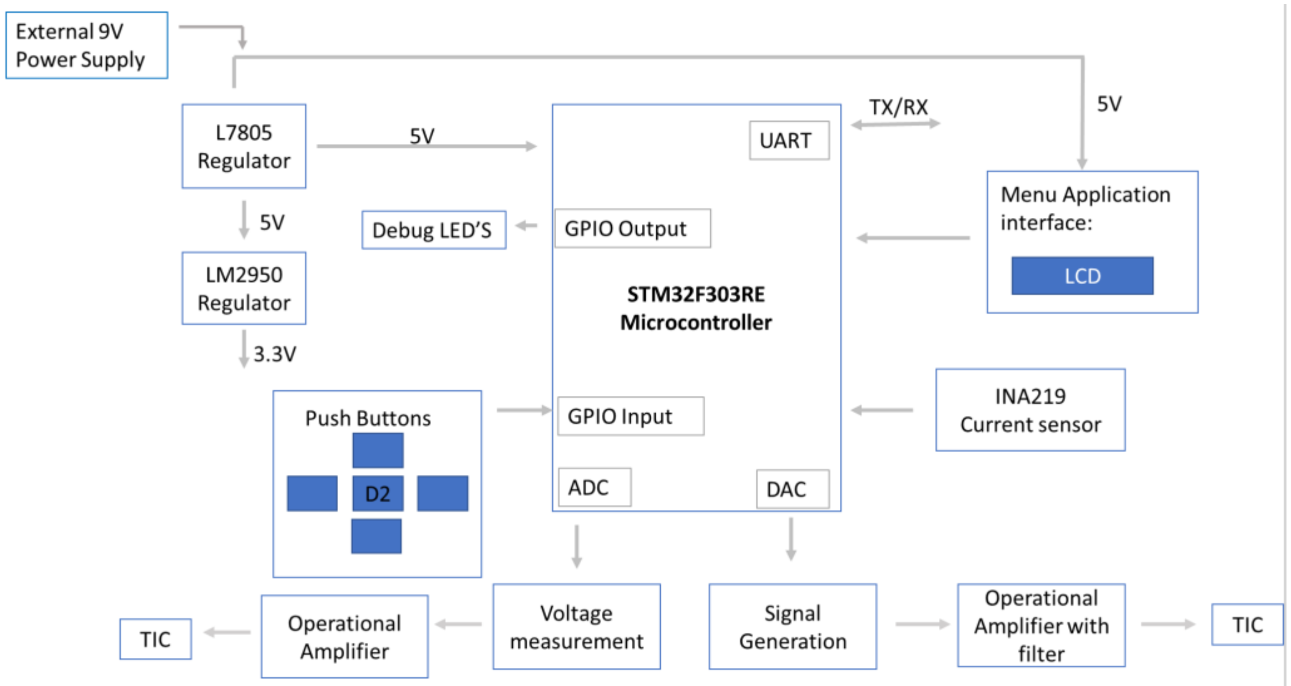


Figure 2: Block diagram of the hardware.

3.1 Power supply

The power supply circuit is described in this section and given in Figure 3 . A 9V external power source is supplied to the PCB. To power various components on the baseboard, the voltage is regulated to 5V using a standard L7805 regulator and 3.3V using an LM2950 regulator.

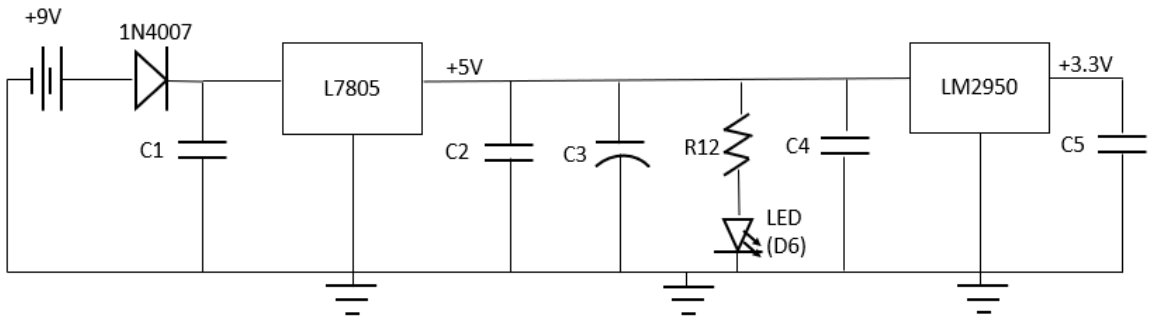


Figure 3: Schematic diagram of power supply.

Capacitors C2 and C5 in Figure 3 enhance the regulator's stability and transient response. Capacitor C3 stores 5V in case the input voltage changes abruptly. The 1N4007 diode prevents reverse current from entering the voltage regulators. An LED (D6) circuit is provided to indicate the presence of a voltage on the 5 V line; if it is on, the circuit is connected and distributing the required voltage. A current limiting resistor connects LED (D6) to the L7805 voltage regulator output.

The data sheets for the respective voltage regulators recommended the value of the capacitors and resistor R12.A Table of component values is given in Table 1:

Table 1: Power supply Capacitor and Resistor Values

Component	Value
C1	100 nF
C2	100 nF
C3	10 µF
C4	100 nF
C5	100 nF
R12	1 kΩ

3.2 Push Button's

There are 5 push button's for this project. The following requirements must be met for each button circuit, firstly the current flowing through the button must be limited by a series resistor to protect the system's voltage regulator and secondly The button must be connected to the STM32F303RE microcontroller as an external Interrupt. Thirdly the buttons must be configured to be active low.An active low button means that the button will perform its function when it is at a logical 0 and inactive at a logical 1.
To satisfy the requirements we could either solder a 10k resistor on board or have an internal resistor that can be activated in software. The circuit was made active low internally in the microcontroller as there is limited space and to avoid too many wires and resistors on the baseboard PCB which could lead to confusion when trying to debug. According to the data sheet the internal resistor is about 40k regardless whether its active low or high.
The button's are read as inputs by the GPIO pins of the microcontroller and receive a 3.3V input. Any pin besides the pins which are already in use can be selected for the five push button's. The

GPIO Pin's were selected because they were closer to the button connections on the PCB which made it easier to solder and made it look neat.Table 2 shows the selected GPIO Pin's.

Table 2: Final GPIO Pins for Push Button's

Button	GPIO Pin
Middle	PC3
Left	PC12
Right	PC10
UP	PC11
Down	PA15

Figure 4 illustrates the schematic diagram for the middle push button, the other button's are connected the same way.

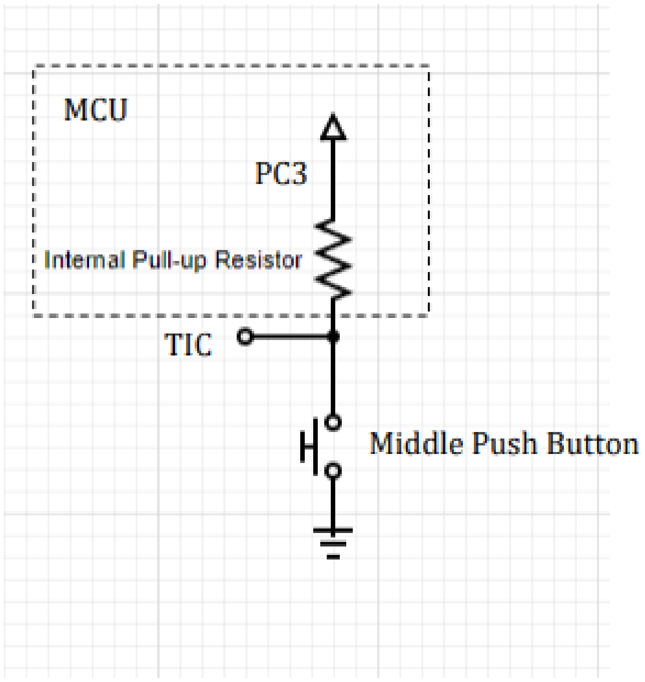


Figure 4: LED(D2) circuit diagram

3.3 Debug LED's

To design the LED circuit we need to take into consideration the value of the resistance needed. We can calculate the value of the required resistance by taking the value of the supplied voltage(3.3V) and subtracting it by the value of the voltage over LED. The voltage over the LED is measured to be 1.9 V. This difference in voltage can then be divided by a desired current of 5mA as stated in the data sheets. The calculated resistance will determine the current flowing through the LED and the brightness of the LED.

$$\begin{aligned} R_{led} &= \frac{V_{sup} - V_{led}}{I_{desired}} \\ R_{lim} &= \frac{3.3 - 1.9}{0.01} \\ R_{lim} &= 280\,\Omega \end{aligned} \tag{1}$$

This results in a resistance value of 280 ohm's. The resistor value was chosen to be substantially larger because this will limit the current in the LED. As a result, a final resistor value of 330 ohms was chosen for all of the LED's; by using this resistor value, we were able to achieve the desired brightness while limiting the current.

Any pin besides the pins which are already in use can be selected for the four debug LED's. The GPIO Pin's were selected because they were closer to the LED connections on the PCB which made it easier to solder and made it look neat. Table 3 shows the selected GPIO Pin's.

Table 3: Final GPIO Pins for Debug LED's

LED	GPIO Pin
D2	PB8
D3	PB3
D4	PB5
D5	PB4

Figure 5 illustrates the schematic diagram for LED(D2), the other LED's are connected the same way.

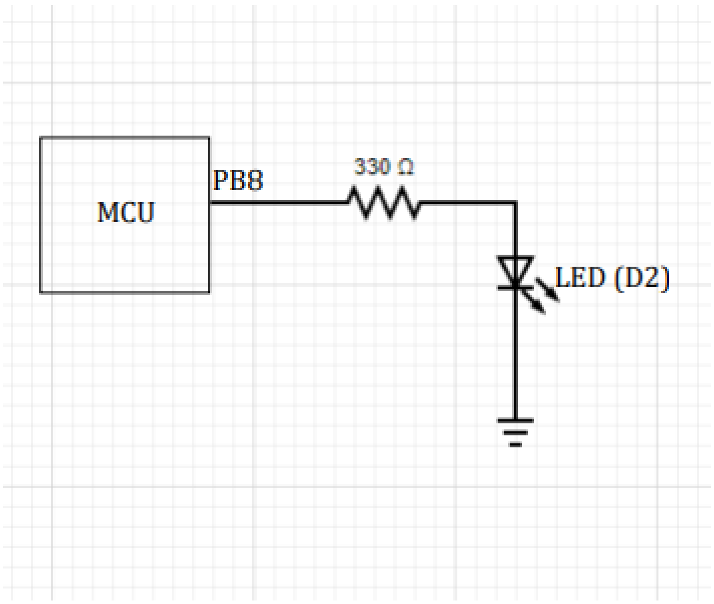


Figure 5: LED(D2) circuit diagram

The four debug LED's are used to indicate the system state.

- LED(D2) - When ON, indicates that the system is in menu state
- LED(D3) - When ON, indicates that the system is in measurement and output display state
- LED(D4) - When ON, indicates that the system is in current measurement mode
- LED(D5) - When ON,Output signal is active , when OFF, Output signal is inactive

3.4 LCD display

We used a 16x2 character LCD display (1602A) as the display device for this project. The LCD is powered by a 5V supply and operates in 4-bit (nibble) interface mode. The LCD also has a contrast adjustment pin, with two resistors R8 and R9 designed to provide good display contrast. Resistor R8 was recommended to be between 18k and 22k and R9 was recommended to be greater than 600 ohm's. We were able to find resistor values that provided good contrast through trial and error.

To keep the current below 10 mA, the LCD requires a current limiting resistor for the backlight, which must be connected in series because an LCD is still an LED. The following equation was used to calculate the value of the limiting resistor.

$$\begin{aligned} R_{lim} &= \frac{V_{in}-V_{led}}{I} \\ R_{lim} &= \frac{5-1.9}{0.01} \\ R_{lim} &= 310\,\Omega \end{aligned} \tag{2}$$

This means that any resitor value above 310 ohm's will be able to deliver a current below 10mA. Resistor values for good contrast and backlight are provided in Table 4:

Table 4: Final resistor values for LCD display

Component	Value
R8	22 kΩ
R9	3.9 kΩ
Rlim	330 Ω

Hardware pin configurations for the LCD are given in Figure 6.

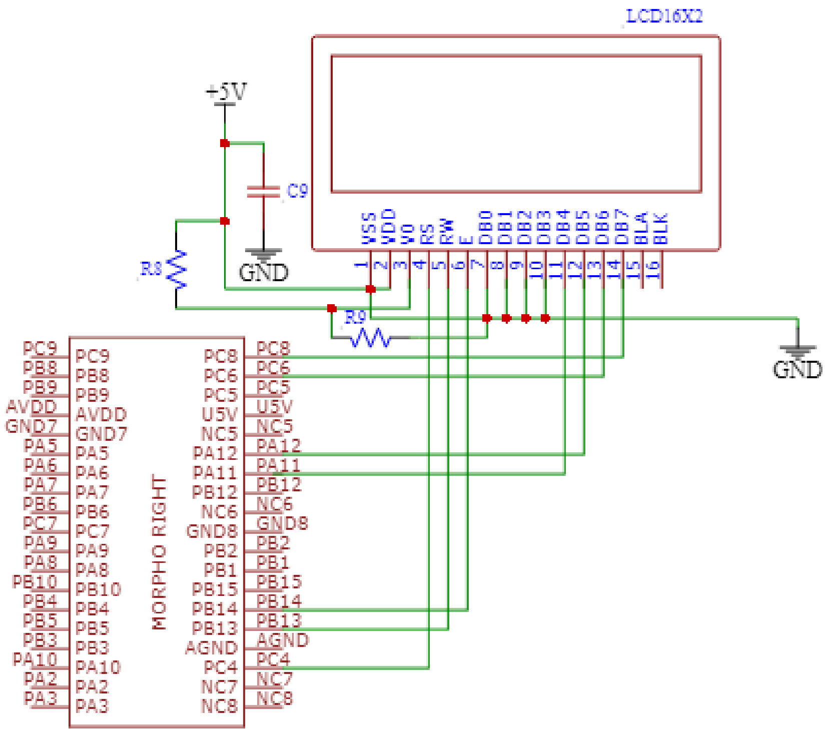

LCD16X2
+5V
C9
GND
R8
R9
GND
VSS
VDD
V0
RS
RW
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
BLA
BLK
MORPHO RIGHT
PC9 PB8 PB9 AVDD GND7 PA5 PA6 PA7 PB6 PC7 PA9 PA8 PB10 PB4 PB5 PB3 PA10 PA2 PA3
PC8 PC6 PC5 U5V NC5 PA12 PA11 PB12 NC6 GND8 PB2 PB1 PB15 PB14 PB13 AGND PC4 NC7 NC8

Figure 6: LCD 4 bit mode schematic diagram

3.5 ADC (input stage)

An Analog to Digital Converter (ADC) input is used in the system to read DC and AC voltage from analogue sensors. The ADC clock drives each conversion step, which is one bit from result to output. The microcontroller has a 12-bit resolution and an input voltage range of 0V to 3.3 V. 12-bit refers to the ADC's resolution and is equals to 2 to the power of 12, or 4096. Because this is the number of sample steps for our ADC, our ADC values will range from 0 to 4095.

To protect the microprocessor, an op-amp buffer circuit based on the MCP602 operational amplifier device is implemented. By clipping at high voltages and delivering unity gain at low voltages, the operational amplifier buffer circuit prevents damage to the microcontroller. The TS supplies the analogue voltage signal via the TIC as the input to the operational amplifier, which acts as a reference voltage with unity gain. The operational amplifier's output is connected to GPIO pin PA0; Figure 7 below depicts the ADC's hardware connections.

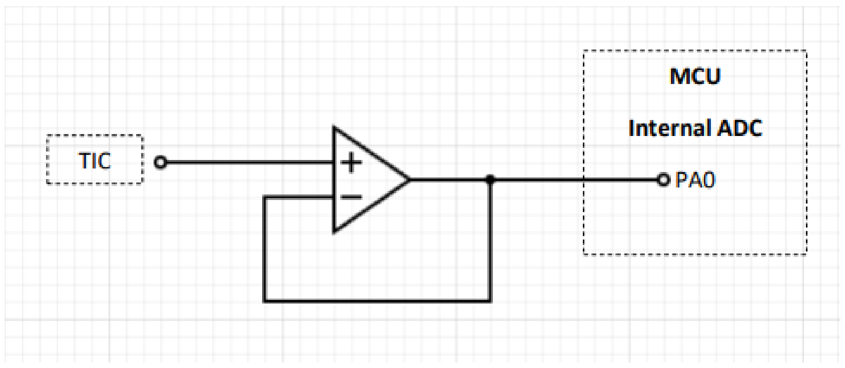


Figure 7: ADC op amp hardware.

3.6 DAC (output stage)

The system produces a pulse, AC, and DC voltage using the microcontroller's internal DAC, where a stream of digital values are converted to an analogue signal. The DAC hardware uses an MCP602 operational amplifier. The MCP602 is a dual op-amp device, and one side of the op-amp is the input buffer, and the second side of op-amp as the output filter circuit of the. The analogue signal is generated at the microcontroller's output pin, GPIO Pin (PA4), then it passes through a designed filter before reaching the TIC.

The system employs a non-inverting amplifier, which provides a high input impedance and a positive voltage gain. The operational amplifier was designed o have unity gain at first, but because it clipped at voltages above 2.5V, the design was modified to have a voltage gain of 2 to cover the required output voltage range using the equation $A_V = 1 + \frac{R_f}{R_1}$. To construct an active low pass filter, resistor values of 1 kΩ and capacitor value of 1 nF were chosen. Figure 8 shows the DAC hardware connections

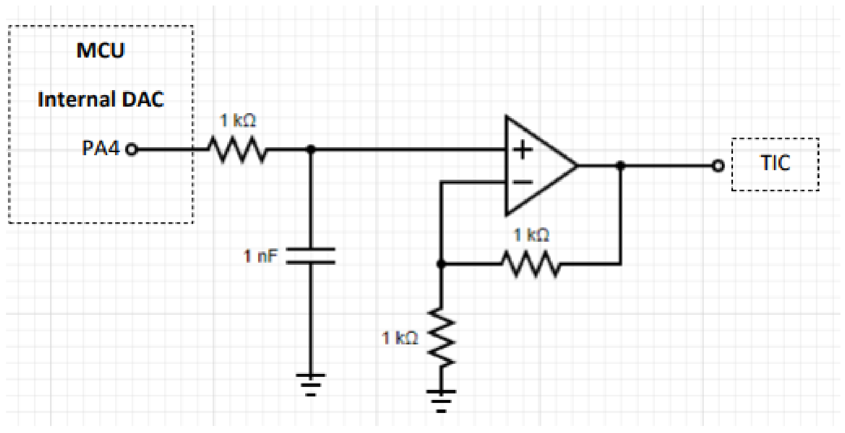


Figure 8: DAC Hardware

3.7 Current sensor

The current of an input signal will be measured for the project using an INA219B device mounted on a small breakout PCB with a sense resistor and support electronics. The sensor is powered by a 3.3V input supply and is linked via I2C interface wires[1]. The current sensor hardware is implemented as shown in Figure 9

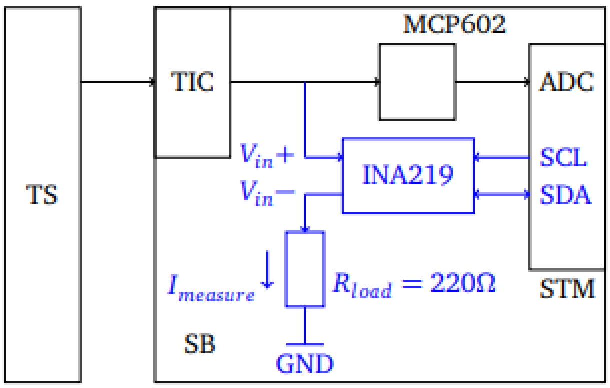
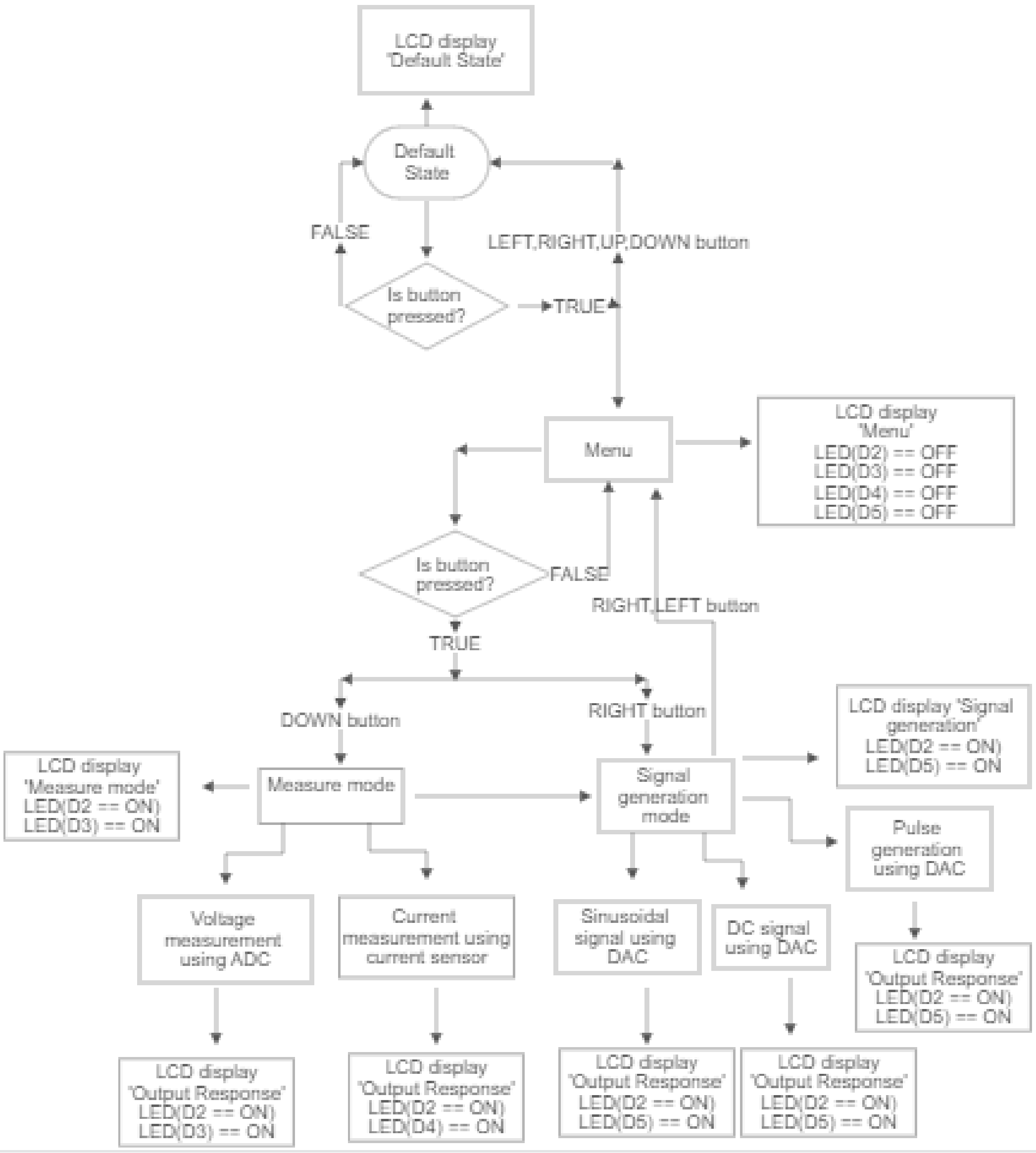


Figure 9: Current sensor Hardware [1]

Due to limited time the hardware considerations for current sensor could not be completed.

4 Software design and implementation

The following section discusses top-level software design and implementation, using design tools, like flow diagrams where needed. The complete software block diagram is given in Figure 10.



LCD display 'Default State'
Default State
FALSE
LEFT,RIGHT,UP,DOWN button
Is button pressed?
TRUE
Menu
LCD display 'Menu' LED(D2) == OFF LED(D3) == OFF LED(D4) == OFF LED(D5) == OFF
Is button pressed?
FALSE
RIGHT,LEFT button
TRUE
DOWN button
RIGHT button
LCD display 'Measure mode' LED(D2 == ON) LED(D3) == ON
Measure mode
Signal generation mode
LCD display 'Signal generation' LED(D2 == ON) LED(D5) == ON
Pulse generation using DAC
Voltage measurement using ADC
Current measurement using current sensor
Sinusoidal signal using DAC
DC signal using DAC
LCD display 'Output Response' LED(D2 == ON) LED(D5) == ON
LCD display 'Output Response' LED(D2 == ON) LED(D3) == ON
LCD display 'Output Response' LED(D2 == ON) LED(D4) == ON
LCD display 'Output Response' LED(D2 == ON) LED(D5) == ON
LCD display 'Output Response' LED(D2 == ON) LED(D5) == ON

Figure 10: Complete software diagram.

4.1 Button debouncing

We are required to implement the push button debouncing to be able to accurately use the button's state in logic statements. The LED's will be used to monitor for correct display changes. Software design considerations to implement button de-bouncing made use of the HAL libraries.

We want to record the state and time of the button's activation and deactivation. We can use an oscilloscope to see how long it takes for all the bounces to complete. It takes about 2.3ms from the time we press the button to the time we release it. A time of 100ms can be used to improve button efficiency. We can use the `HAL_GetTick(..)` function to get the last time the button was pressed. The number of milliseconds that the system has been running is returned by `HAL_GetTick(..)`. This function increments by one every millisecond. To receive the button state, we can use the `HAL_GPIO_ReadPin(..)` function. Using these two HAL libraries we can store the button state and time.

We want to move the LED forward at a rising edge (the transition from low to high). When a rising edge occurs, we can set a flag that is read by our main.c function. When the flag is set to one in the main function, we will move the LED forward and immediately reset the flag to stop the LED from moving. This flag is set in an external interrupt and occurs only on rising edges. The rising edge occurs when previous button state == 0 current button state == 1.

We want to do all of this when the current button time – button last time ¿ 100ms, where the current time is where the interrupts in the middle occur, and the button last time is where the first interrupt occurred.
The flow diagram for the button debouncing is given in Figure 11

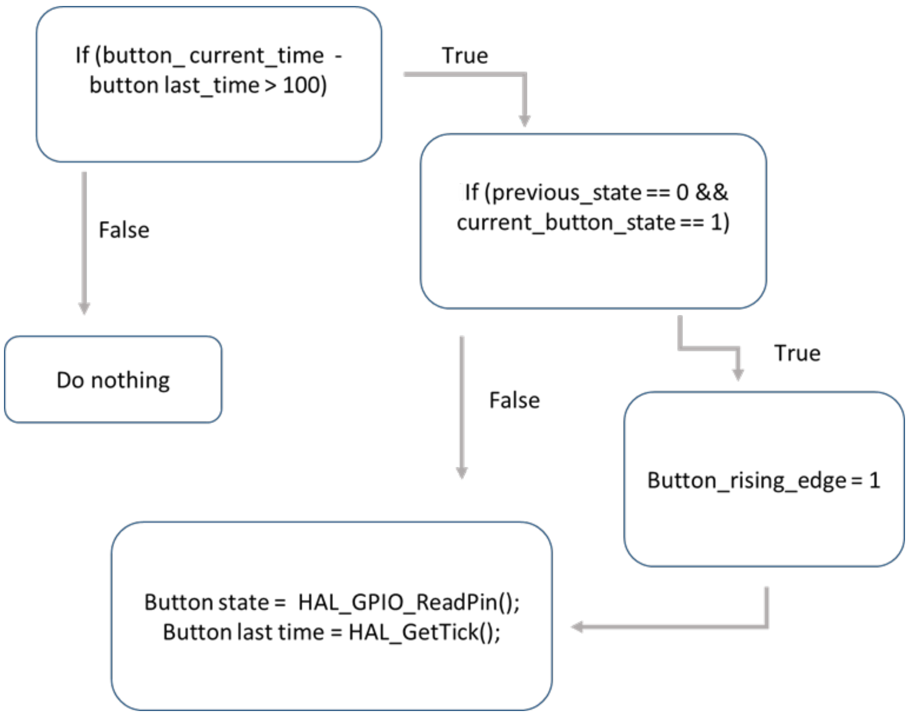

If (button_current_time - button last_time > 100)
True
False
If (previous_state == 0 && current_button_state == 1)
Do nothing
True
False
Button_rising_edge = 1
Button state = HAL_GPIO_ReadPin();
Button last time = HAL_GetTick();

Figure 11: Button debouncing Pseudocode.

4.2 UART Communication's

Figure 12 illustrates the UART data flow process.

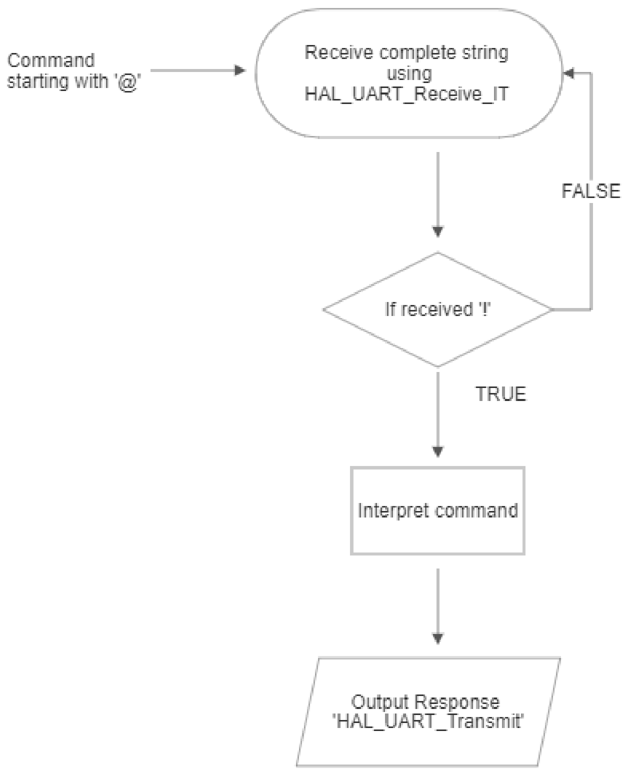

Command starting with '@'
Receive complete string using HAL_UART_Receive_IT
FALSE
If received '!'
TRUE
Interpret command
Output Response 'HAL_UART_Transmit'

Figure 12: UART dataflow process

UART communication is asynchronous (no clock is transmitted), uses TTL signal levels, and has 8 data bits, 1 stop bit, and no parity checking. This system uses the least significant bit first and is bi-directional at moderate speeds, with a baudrate of 115200 bp. In both transmit and receive modes, the device employs a UART connection.

The system employs HAL libraries to transmit (GPIO Pin PA2) and receive commands (GPIO Pin PA3). The `HAL_UART_Transmit` function is used to send data to the LCD screen, while the `HAL_UART_RxCpltCallback` function receives messages being sent from the TS. Each message begins with a '@' and ends with a '!' followed by a newline character. A ',' separates each field in the message. All of the messages sent are in response to a message request from the TS. In the absence of a test station, UART messages use ASCII printable characters to make it easy to interface with the PCB debug connector using simple terminal programs such as Teraterm and Realterm.

A UART protocol's structure includes a clock generator to allow sampling in a bit period, as well as input and output shift registers. It includes transmit and receive control, read and write logic, as well as transmit and receive buffers and a DMA controller.

Data bits are used to send data in serial communication. Each data bit consists of a starting bit, 8 bits, no parity checking, and a stop bit. Before sending the data byte, a start bit is sent to notify the receiver device to begin logging the data. This bit is active low, which means it goes low when it needs to notify the receiver. The data byte is sent after the starting bit. The time it takes to send 8 consecutive bits depends on the baud rate, for this system it takes 115200 bits per second.

4.3 ADC data flow

Figure 13 shows the data flow process for ADC.

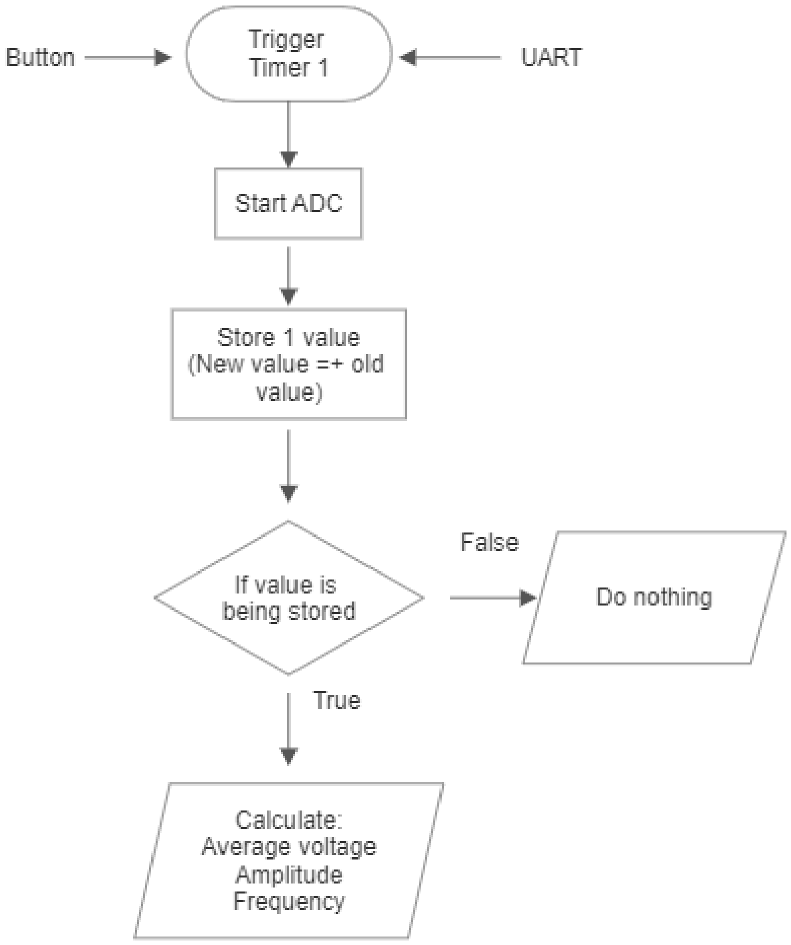

Button
Trigger Timer 1
UART
Start ADC
Store 1 value (New value =+ old value)
If value is being stored
False
Do nothing
True
Calculate: Average voltage Amplitude Frequency

Figure 13: ADC flow process

The measure mode system employs an ADC to convert analogue signals to digital signals. The ADC was configured to be triggered by a Timer 1 interrupt. The ADC has a 12-bit resolution, which means that the maximum digital value stored iS 4095, which corresponds to the maximum sampled voltage of 3.3V. The ADC makes use of the HAL libraries to Start the ADC, get values and to wait until ADC has completed.
Analog signals, such as sensor output in volts, and must be converted into digital values for processing, and the conversion must be precise. When a voltage is applied to the microcontroller's GPIO PIN PA0, the analog value is interpreted and stored in an integer variable. The stored analog value in the rage 0 to 3.3V is converted to an integer value between 0 and 4096 using the following formula:

$$V_{input} = ADC_{value} \times \frac{3.3}{4096} \tag{3}$$

4.4 DAC data flow

Figure 14 shows the data flow process for DAC. The system used the DAC triggered by timer 2 to produce an analogue signal from the digital values obtained by the microcontroller. The DAC was configured in software using the trigger from timer 2 as a global interrupt and enabling DMA in circular mode. DMA was used to free up the CPU for other tasks[2].
It is stated by Nyquist's theorem that a periodic signal must be sampled at more than twice its highest frequency component. As a result, we must sample our given maximum frequency of 5 kHz by at least 10 kHz. The sample rate was calculated using the formula below.

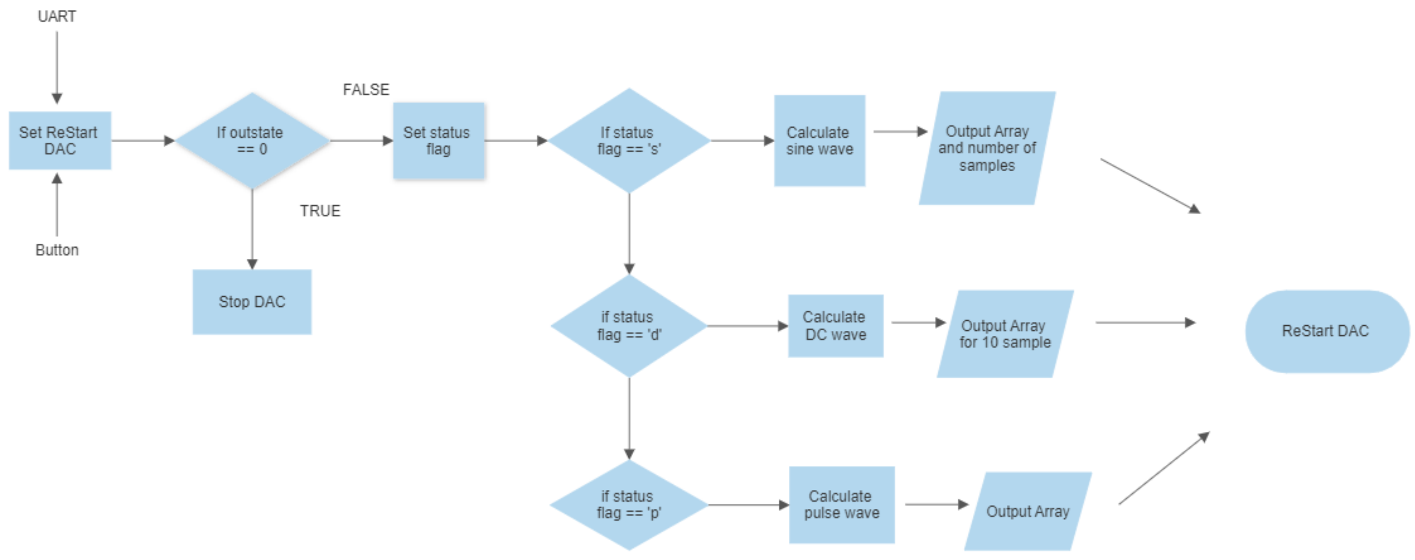


Figure 14: DAC flow process

$$f_{timer} = \frac{f_{clock}}{(ARR-1)(PSC-1)} \tag{4}$$

Because the timer 2 clock is 72 MHz, we used a prescaler (PSC) value of 72. The array value (ARR) is set to 20, which divides the clock further for a sampling frequency of 50 kHz, which is more than 10 times the maximum frequency, resulting in better accuracy and speed. HAL libraries were used to start and stop the DAC, Timer, and DMA.
The frequency of the generated wave was controlled by varying the number of samples. This was calculated using the formula.

$$n_{sample} = \frac{f_{timer}}{f_{generated}} \tag{5}$$

Because the microcontroller has a 12-bit resolution, the maximum digital value produced is 4095, which results in maximum output voltage of 3.3V at the selected GPIO Pin PA4. We used a sine function to generate digital values and included the math.h library. The ST document for DAC provided the formula used to generate AC voltage signal[3].

$$y_{sine} = [amplitude \times \sin(x\frac{2\pi}{n_s}) + offset]\,[\frac{4095+1}{3.3}] \tag{6}$$

The following equation was used to calculate the DC voltage signal.

$$y_{dc} = amplitude \times \frac{4095+1}{3.3} \tag{7}$$

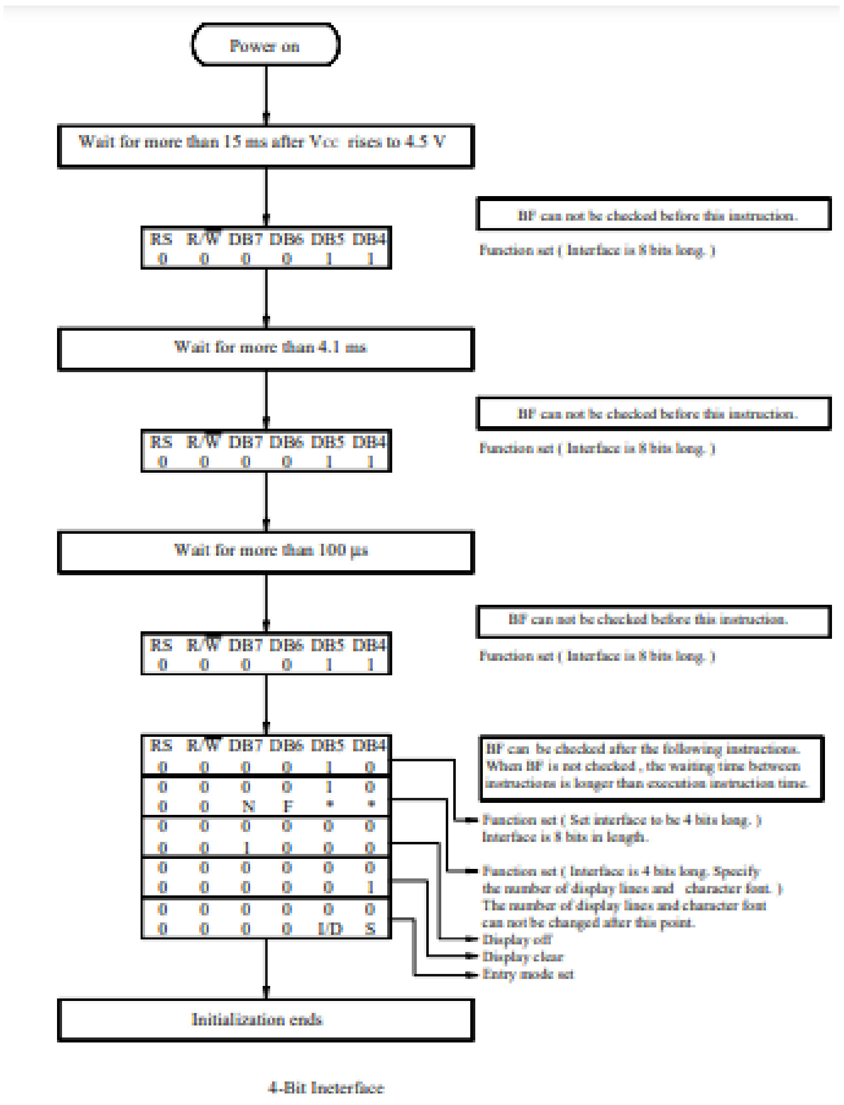
To account for the designed filter voltage, gain at the DAC circuit's output, the above equations are divided by 2. The pulse conversion equation is similar to the DC equation, but the output varies depending on frequency and duty cycle.

4.5 LCD interface and timing

This project makes use of a 16x2 character LCD display. This means that we can display a character in two rows and 16 columns, for a total of 32 characters. The LCD is powered by a 5V supply and runs in 4-bit (nibble) interface mode, which means we only use data pins D4-D7. Characters are displayed on the LCD in two pulse signals in 4-bit mode. 4 data bits must be ready 60 ns before the Enable's falling edge. Because two enable signals are required to display a single character, the bit rate in 4-bit mode is long.

The LCD includes two registers: a Data register and a Command register. Any data that must be displayed on the LCD must be written to the LCD's data register. A command can be issued to the LCD by writing it to the LCD's Command register. This pulse is used to distinguish the data/commands received by the LCD. If the RS signal is LOW, the LCD interprets the 4-bit information as Command, writes it to the Command register, and then executes the command. When the RS signal is HIGH, the LCD recognizes the 4-bit information as data and writes it to the data register. Following that, the LCD decrypts the data to produce the 5x7 pattern, which is then displayed on the LCD[4].

To initialize the lcd in 4 bit mode, we send the value 0x20 to the lcd's command register, this value indicates to the LCD controller that we want to interact in 4-bit mode. 0x20 corresponds to one row, and we also want the character shape to be displayed in a 5x7 matrix. If our character has two rows, we will send 0x28 rather than 0x20. It informs the LCD controller that we want 4 bit communication and that the character size should be between 5x7 dot matrix[5].The concept in Figure 15, was adopted for the initialisation of the lcd and timers.



Power on
Wait for more than 15 ms after Vcc rises to 4.5 V
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
BF can not be checked before this instruction.
Function set (Interface is 8 bits long.)
Wait for more than 4.1 ms
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
BF can not be checked before this instruction.
Function set (Interface is 8 bits long.)
Wait for more than 100 μs
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
BF can not be checked before this instruction.
Function set (Interface is 8 bits long.)
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 0
0 0 0 0 1 0
0 0 N F * *
0 0 0 0 0 0
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 1
0 0 0 0 0 0
0 0 0 0 I/D S
BF can be checked after the following instructions. When BF is not checked , the waiting time between instructions is longer than execution instruction time.
Function set (Set interface to be 4 bits long.) Interface is 8 bits in length.
Function set (Interface is 4 bits long. Specify the number of display lines and character font.) The number of display lines and character font can not be changed after this point.
Display off
Display clear
Entry mode set
Initialization ends
4-Bit Ineterface

Figure 15: LCD initialisation.

4.6 I2C interface and current sensor

The software considerations for the I2C interface and current sensor could not be implemented due to time constraints. The student focused on other sections of the project.

5 Measurements and Results

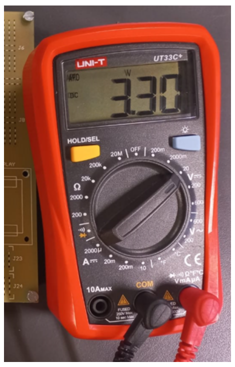
5.1 Power Supply

The board receives a nominal 9 V power source which is regulated to 5V and 3.3V. The method used to test the power supply was measuring the voltage across the two regulators, using a multimeter. The voltage was measured 3 times for each regulator to determine if our power supply is working efficiently and within the required 10% accuracy. The voltages were measurements and the averages are given in Table 5:

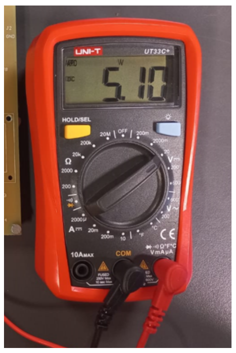
Table 5: Power supply voltages

Measurement	L7805	LM2950
1	5.08 V	3.30 V
2	5.1 V	3.31 V
3	5.1 V	3.29 V
Average	5.09 V	3.30 V

Figure 16 shows one of the measurements taken for the voltage regulators.



(a) LM2950 measurement

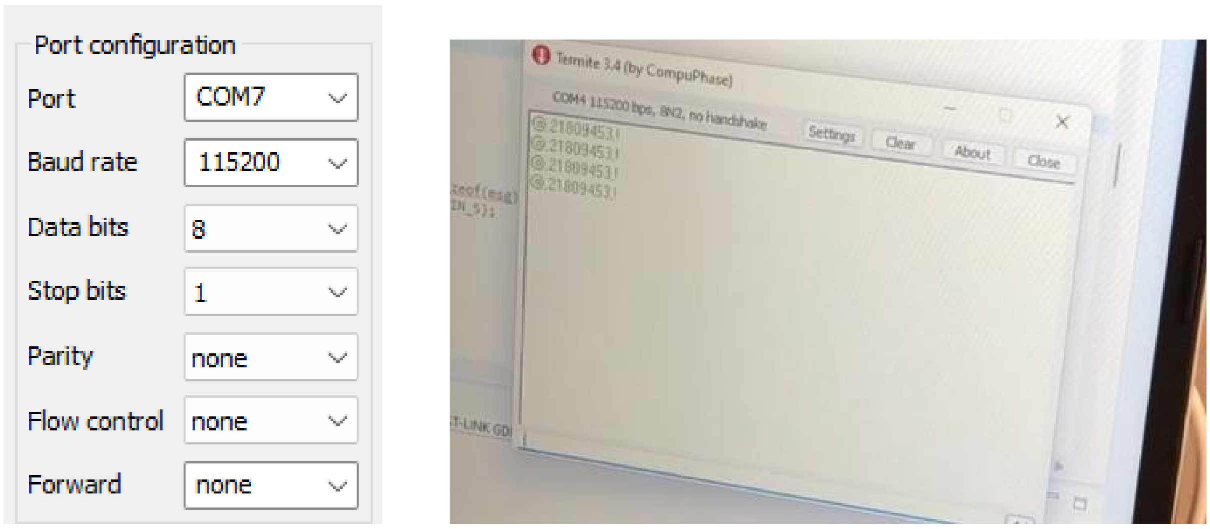


(b) L7805measurement

Figure 16: Voltage regulator measurements

5.2 UART communications

The UART communication is configured in an 8N1 configuration The device at a rate of 115200 baud. We can ensure that it is working at this buadrate by setting this information in Termite as show in Figure 17 and checking that information is being transmitted and received correctly in Termite. Figure 17 shows the received student number format in Termite.



Port configuration

Port	COM7
Baud rate	115200
Data bits	8
Stop bits	1
Parity	none
Flow control	none
Forward	none

```
Termite 3.4 (by CompuPhase)
COM4 115200 bps, 8N2, no handshake
@21809453!
@21809453!
@21809453!
@21809453!
```

(a) Termite settings (b) Student number ouput

Figure 17: Voltage regulator measurements

5.3 Buttons

The method used to test the button circuit was done via the oscilloscope, which was connected to a button with crocodile clips. After pressing the button, the oscilloscope was used to determine how long the button bounces. The oscilloscope has a feature that allows to freeze the measurement frame, this was used read the button bounce time, it was found that it takes 2.3ms which can be seen in figure Figure 18 . The procedure was repeated several times to ensure that the bounce delay of 100 ms was adequate. The same procedure was followed for all the buttons.

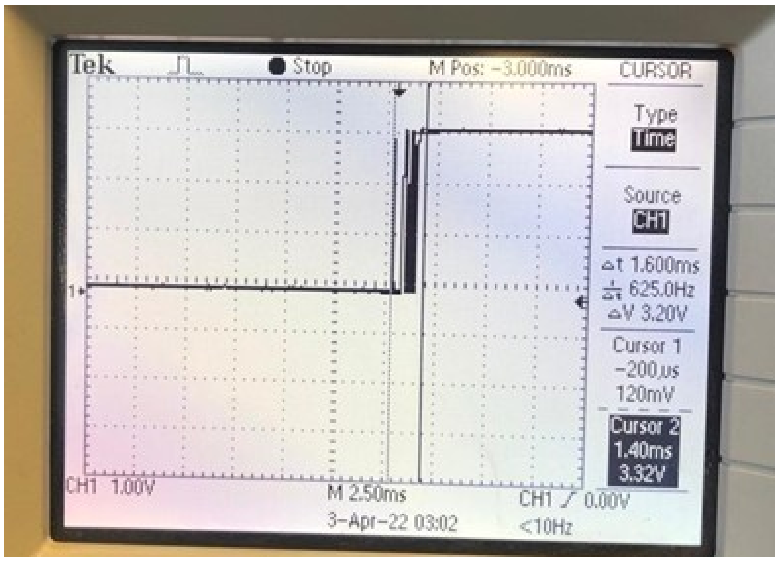


Figure 18: Button Bounce on Oscilloscope

To ensure that the button debounce was correctly implemented, the button circuit was connected to the oscilloscope using crocodile clips. The button was pressed and the resulting picture is shown in Figure 19

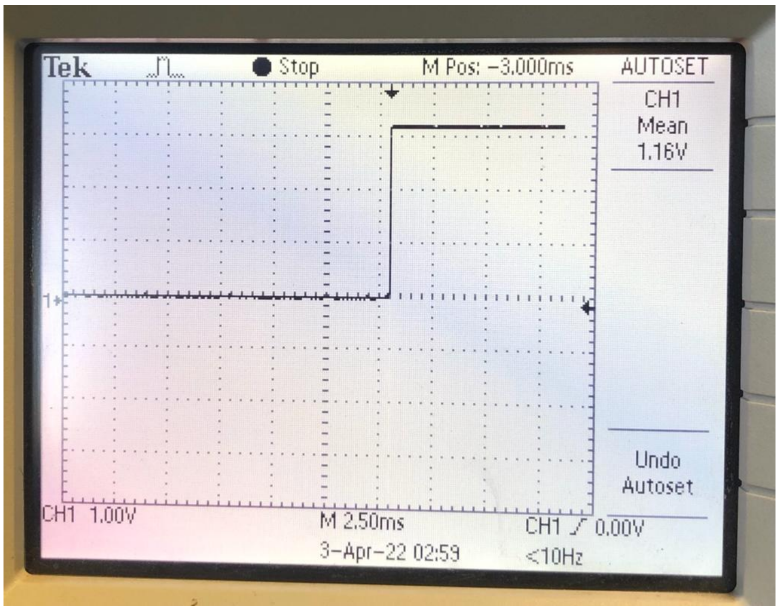
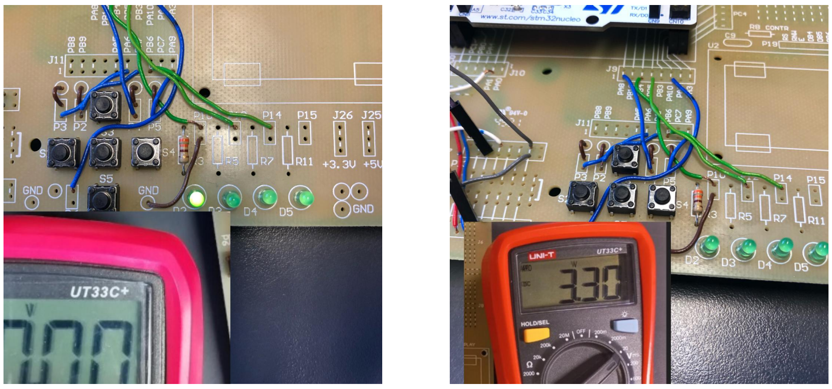


Figure 19: Button debounce on Oscilloscope

To determine if our requirement for an active low button circuit is successful we can measure the voltage across the buttons when the PCB is powered. We can expected a voltage of 3.3V when the button is not pressed and a voltage of 0V when the button is pressed.



(a) Button pressed (b) Button not pressed

Figure 20: Push Button measurements

5.4 LEDs (debug)

Using a multi-meter the voltage over the 330 resistor of the LED (D2) was measured. The reading on the multimeter gave a voltage of $V_{meas} = 1.29\,\text{V}$, this test method and value is shown in Figure 21. This value is very close to our earlier calculated value of the resistor $V_{supplied} - V_{led} = 3.3\text{–}1.9 = 1.4\,\text{V}$. Using this measured value, the current through our LED (D2) is: $I_{led} = \frac{V_{meas}}{R} = 3.9\,\text{mA}$ Calculating we get a value of Iled = 3.9 mA. The current over the LED is less than 5mA, this is expected. The achieved current value is very good compared to the maximum current through an LED which is typically 20mA. This indicates that the chosen value of the resistance was correct as we want to choose a resistance that requires the minimum current to be drawn but provide maximum brightness to the LED. The current achieved will not strain the LED. This measurement procedure was done for all the LED's.

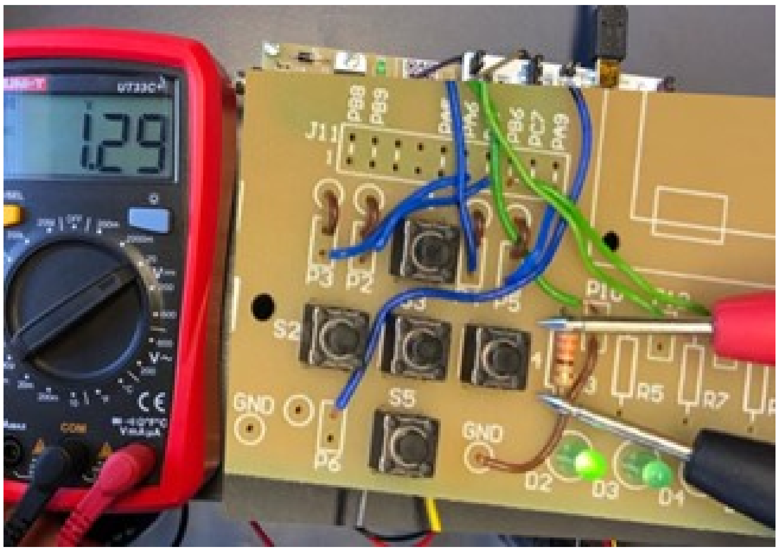


Figure 21: LED current measurement

To test the debug LED’s, the system state was changed by pressing the button’s and the LED’s were checked for change according to their functionality. Initial state of system Figure 23:

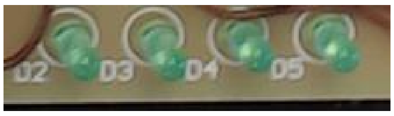


Figure 22: Initial state

Test for the system in Menu State Figure 23:

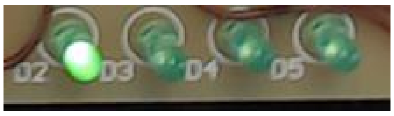


Figure 23: Test 1

Test for the system in Menu state, measurement mode and measuring the current Figure 24:

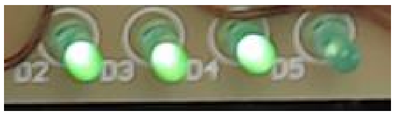


Figure 24: Test 2

Test for the system in Menu state, measurement mode and measuring the current with output signal generated active Figure 25:

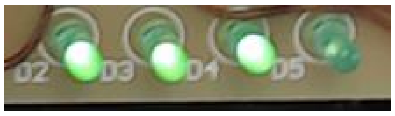
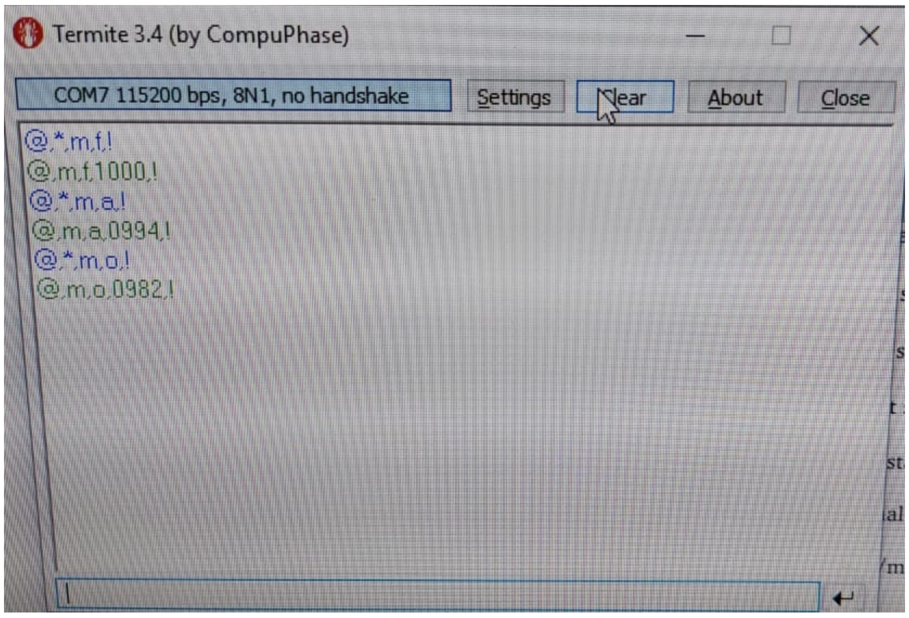


Figure 25: Test 3

In all the cases our LED's switch on as prompted indicating successful implementation.

5.5 ADC

The method used to test for ADC included connecting the input buffer of the operational amplifier. Commands to start the ADC were sent to termite using UART.Figure 26 shows the UART commands as well the output response.



COM7 115200 bps, 8N1, no handshake

```
@,*,m,f,!
@,m,f,1000,!
@,*,m,a,!
@,m,a,0994,!
@,*,m,o,!
@,m,o,0982,!
```

Figure 26: ADC output on Termite

Figure 26 illustrates successful implementation of ADC

5.6 DAC

The method used to test for DAC, involved using the oscilloscope, which was connected to the DAC output using crocodile clips. Termite was used to send commands in order to generate outputs for signal generation.

Figure 27 shows the result for sinusoidal signal

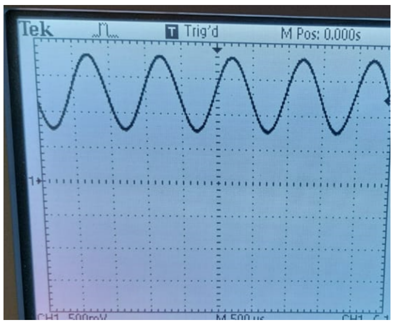


Figure 27: LED debug

Figure 28 shows the result for dc signal.

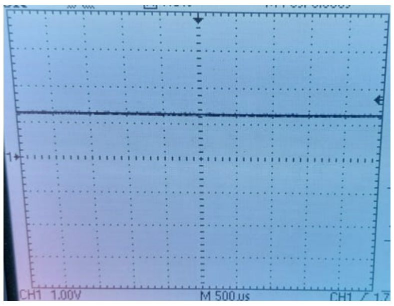


Figure 28: LED debug

Figure 29 shows the result for the pulse signal.

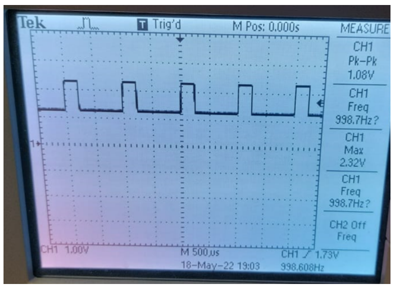


Figure 29: LED debug

5.7 LCD

The method used to test the current over the LCD was done using a multimeter, as the LCD was designed for a current below 10 mA. A voltage drop of 1.13 V was measured across the limiting resistor.The value of the limiting resistor used is 330 Ω, therefore the current over the LCD IS $I_{LCD} = \frac{1.13}{330} = 3.42mA$. This value is well below 10 mA indicating correct application of LCD hardware. To test for good contrast, characters were sent to the LCD and observed to see if the contrast met the requirements and that the chosen values for resistors R8 and R9 were correct. Figure 32 indicates good contrast and backlight of the LCD when in menu mode. The LCD was designed to change display as the user interacts with the system,Figure 32 and Figure 32 shows that the LCD changes states as the user interacts with the system.

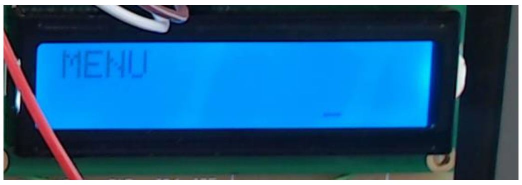


Figure 30: LCD Display.

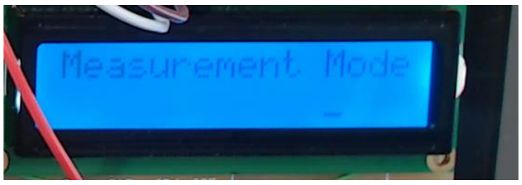


Figure 31: LCD Display in Measurement mode.

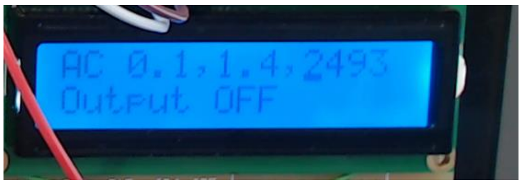


Figure 32: LCD Display in AC voltage measurement.

5.8 Current sensor

There are no measurements for this section, as it was not implemented.

5.9 Complete system

When the middle push button is pressed, the system enters menu state, the LED (D2) switches ON to indicate that the system is in menu mode, and the LCD displays 'MENU'. On each release of the push buttons, the system should be able to toggle the appropriate LED. The user can then select their preferred measurement mode. The DAC can be used for signal generation, the ADC for voltage measurement, and the current sensor to measure current. The Debug LED's will then indicate the current state of the system, and the LCD will change as the user interacts with the system, and display the output response based on the user command. The entire system on the PCB is shown in Figure 33

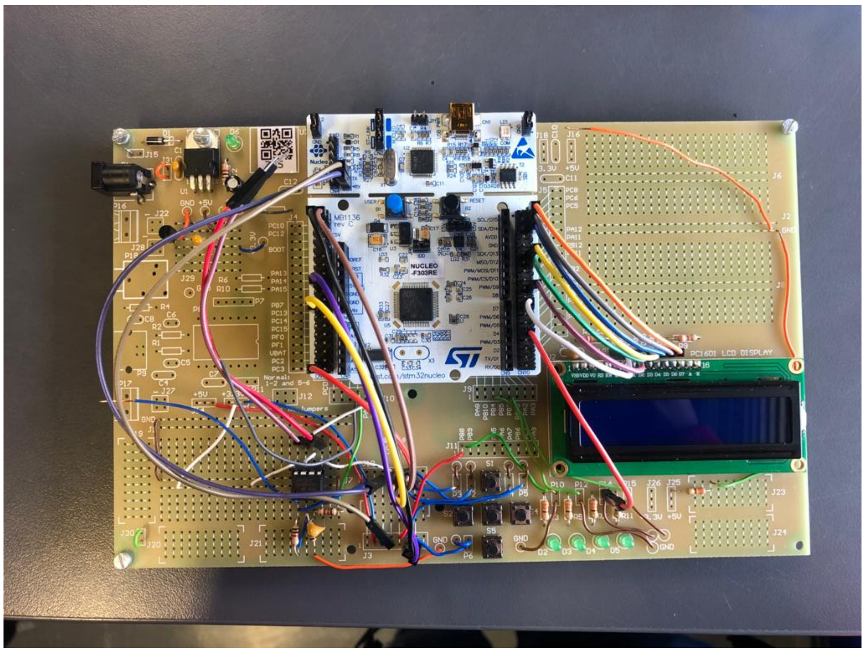


Figure 33: PCB with complete system.

6 Conclusions

To summarize this project, the entire system was successfully implemented, all the subsystems interacted fairly well and efficiently, and we were able to meet the majority of our objectives. The software and hardware considerations used were correct and functioned perfectly. Due to time constraints, the current sensor portion of the project could not be completed. Other aspects of the project were given preference more than current sensor software due to the limited time. The DAC's inaccuracy at certain voltages is one of the system's flaws. All measurements are allowed a 10% tolerance for error, but signal generation does not always meet this standard. Future recommendations and possible improvements include prioritizing project objectives to ensure project completion and success.

Use experimental results, design limitations and system performance, explain your conclusions drawn.

References

[1] A. Barnard, L. Grootboom, U. Louw, and D. Klink, “Design (e) 314 project definition,” 2022, last accessed 23 May 2022. [Online]. Available: https://learn.sun.ac.za/pluginfile.php/3452176/mod$_{r}esource/content/0/DesignE314_{2}022_{P}DD_{v}0.7.pdf$

[2] C. Tech, “Dac in stm32,” 2022, last accessed 25 May 2022. [Online]. Available: https://controllerstech.com/dac-in-stm32/

[3] *NUCLEO-F303RE*, ST Microelectronics, 2016. [Online]. Available: https://www.st.com/en/microcontrollersmicroprocessors/stm32f303re.html

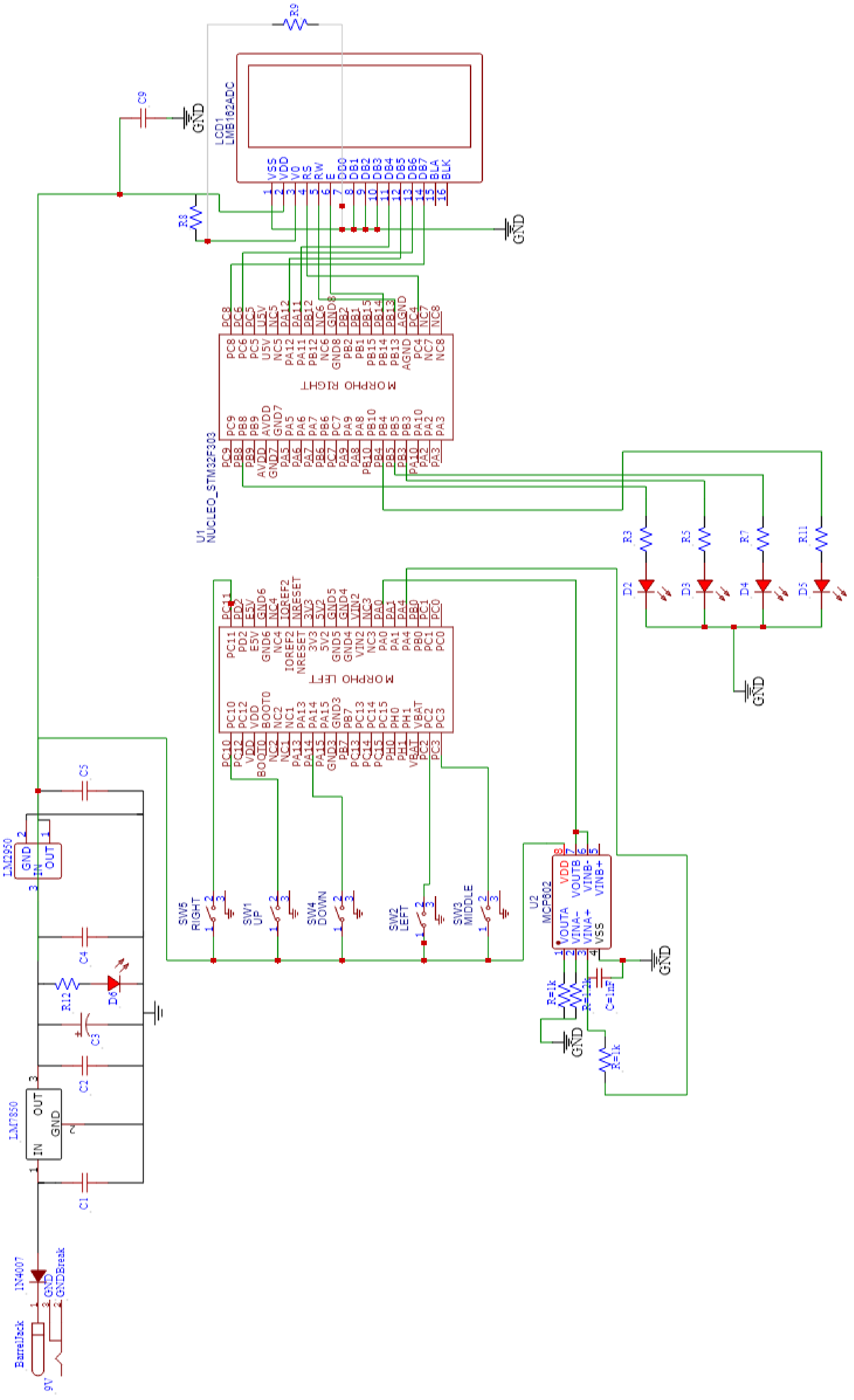
[4] *LCD-1602A Datasheet*, Crystalfontz America, Inc., 2016. [Online]. Available: https://datasheet4u.com/datasheet-pdf/CA/LCD-1602A/pdf.php?id=519148

[5] O. V. den Eede, “Lcd 1602 parallel display with nucleo stm32f446re using stm32cubeide,” 2018, last accessed 25 May 2022. [Online]. Available: https://www.micropeta.com/video60

7 Appendix

Table 6: STM32 pins used in their configuration

Connection	GPIO Pin
LD2 [Green LED]	PA5
LED(D2)	PB8
LED(D3)	PB3
LED(D4)	PB5
LED(D5)	PB4
Middle	PC3
Left	PC2
Right	PC11
UP	PC10
Down	PA15
RS	PC4
RW	PB13
E	PB14
DB4	PA11
DB5	PA12
DB6	PC6
DB7	PC8
`ADC1_IN1`	PA0
`DAC_OUT1`	PA4
`USART_TX`	PA2
`USART_RX`	PA3


BarrelJack
9V
1N4007
GND
GNDBreak
LM7850
IN
OUT
GND
C1
C2
C3
C4
R12
D6
LM2950
GND
IN
OUT
C5
SW5
RIGHT
SW1
UP
SW4
DOWN
SW2
LEFT
SW3
MIDDLE
U2
MCP602
VDD
VOUTA
VINA-
VINA+
VSS
VOUTB
VINB-
VINB+
R=1k
R=1.2k
C=1nF
R=1k
GND
U1
NUCLEO_STM32F303
MORPHO LEFT
MORPHO RIGHT
R3
R5
R7
R11
D2
D3
D4
D5
LCD1
LMB162ADC
VSS
VDD
V0
RS
RW
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
BLA
BLK
R8
R9
C9