## Team 6

# Final Report

Relay Analyzer

By: Matthew Hengeveld (8534331) and Kevin MacIntosh (7885551)
Conestoga College

Capstone Project 1 (EECE74125)

Spring 2021

# Table of Contents

# Table of Figures

# List of Terms

| | |
|---|---|
| USB | Universal Serial Bus |
| PC | Personal Computer |
| GUI | Graphical User Interface |
| PCB | Printed Circuit Board |
| GPIO | General Purpose In/Out |
| ADC | Analog to Digital Converter |
| DMA | Direct Memory Access |
| DAC | Digital to Analog Converter |
| FreeRTOS | Free Real Time Operating System |
| PGA | Programmable Gain Amplifier |
| VCP | Virtual Com Port |
| MVP | Model-View-Presenter |
| RSS | Resting Steady State |

# Abstract

Relays have measurable characteristics that can change over the life of the relay. Factors such as coil voltage, switching frequency, temperature, and cycles completed all have an effect. The Relay Analyzer is a tool for automatically determining and tracking realy characteristics over the life of the relay. Using a high-speed USB connection to a PC, test results and settings can be changed and viewed on the GUI.

# Introduction

## Purpose of Document

The purpose of this document is to review the Relay Analyzer project in detail. This document will go through the project in order of execution, in a logical order, so each section builds on the previous, starting with the project definition, and finishing with results and conclusion.

## Project Definition

The Relay Analyzer will automatically test relays for specific characteristics, over the lifetime of the relay. The Relay Analyzer will be controlled by a PC, where a GUI will set test settings and compute and display results.

## Background

Thorough research was done into existing solutions for testing and characterising of electromechanical relays. Very few solutions were discovered, and at great cost. Many other electronic components have associated test equipment dedicated to their characterization, such as transistors and motors. However, relays do not, despite being used in high-performance telecommunications and aerospace applications [1].

Many relay manufacturers will state specific relay characteristics on their datasheets. These characteristics are typically limited to a few characteristics, and always in a brand-new state. This lack of detailed information from manufacturers can lead to relay failure or product failure prematurely, where a relay's characteristics have changed significantly enough to cause a product to cease to function properly. Best-case,

## Proposed Solution

Our proposed solution is to develop a fully-automatic, PC-connected system to characterize a relay over its life, as well as catalog test relays. This information is then displayed on the GUI. The relay-under-test is tested at a specified frequency throughout a test. Several

different tests can be set to be performed at each cycle, or every n cycles. The one test performed each cycle is the activation time and deactivation time.

Very few commercial solutions exist. The KoCos Artes 460 & 600 and the products from Applied Relay Testing are the only solutions found. These solutions focus on a greater range of relays - safety, timing, as well as standard power, signal and automotive types - and therefore come with much greater complexity and cost. Several “dumb” relay testers also exist, however, these lack any automated tests, and typically only test a relay to make sure the contacts open and close upon coil excitation.

Our solution will focus on providing critical characteristic results for commonly used relays in telecommunications, automotive, and consumer electronics, while doing so at a much-reduced price compared to existing solutions.

Our solution would allow an electrical design engineer to make more informed choices about which relay to use for a specific application by giving them information not normally found in datasheets. This information would ensure a product would not fail prematurely due to unknown or unexpected degradation in relay performance or characteristics.

# Literature Review

It is possible to find some outlines on how to perform relay testing, but the instructions are often lacking detail and require multiple pieces of equipment and a manual setup. These setups can be clunky and usually only measure one one parameter at a time [2]. The proposed design will solve this by providing an easy and safe to use method of life testing and analyzing various relay characteristics.

In a previous work term, one of the projects worked on was a simple relay testing device for in-house use. This both inspired designing a relay life testing device and shows that there is demand for this device in industry.

The document “Relay Technical Information” from Matsushita Electric Works, outlines terminology surrounding relays, including basic components, characteristics and packages. Methods of determining specifications are also covered which, along with the section on applications of relays has been used in determining what should be analyzed and how. The general applications guideline goes into detail on standard power, signal and temperature operations of signal relays which will help guide the design of the circuits. In addition, there is a check list of cautions for use that provides many good considerations for the final design. Reliability is discussed in terms of a Bathtub curve, Weibull analysis and their correlation that can be referenced when determining graphics for displaying information [3].

Pickering offers a look at how life testing is done by manufacturers including load conditions, time, cycles [4]. This can be used for reference when designing our proposed solution.

There's currently only one major manufacturer of purpose built relay analyzing devices, Applied Relay Testing (ART). They offer a range of devices for particular tests, as a sample the Low voltage (Reflex 10), life/chatter (Reflex 51) and connector (Reflex 950) test devices will be compared to the proposed solution.

The low voltage Reflex 10 tests various characteristics like coil resistance, adaptor continuity, contact resistance and VRamp. It tests one relay at a time, and focuses on fast test times [5]. This contrasts the proposed solution which focuses on life tests of vital characteristics.

ART's Reflex 51 offers a modular life testing system for contact timing and pull-in/drop-out characteristics [6]. While this device does support life testing, it looks at different characteristics than the proposed solution and does not appear to be as simple to use.

The Reflex 950 is designed for hi-pot, high voltage AC, DC and DC biased capacitance testing with up to 160 four wire contacts [7]. Again, this device examines different characteristics from the proposed design and does not perform life testing which is the key aspect of the proposed design.

# Project Description

The Relay Analyzer will characterize a relay under test over the lifetime of the relay, a specific number of cycles, or a specified amount of time. The Relay Analyzer will automatically perform tests in order to characterize the relay under test. The Relay Analyzer will test 4 main parts of a relay:

1. Relay coil
2. Relay contacts
3. Relay contact bounce
4. Extended use (time/cycles)

Specifically, the Relay Analyzer will characterize a relay's coil voltage, drop-out voltage, minimum turn-on voltage, maximum turn-off voltage, coil current, coil resistance, activation and deactivation times, and contact bounce.

# Project Requirements

**MoSCoW Table**

| Requirements | Must | Should | Could | Wouldn't | Notes |
|---|---|---|---|---|---|
| Measure the activation[*] and deactivation[**] times of a relay | X | | | | Times will vary between 1ms to 30ms |
| Measure and characterize the switch-on and switch-off bounce[***] | X | | | | |
| Store waveform data of relay bounce | X | | | | Store waveform data from before trigger to several milliseconds after relay contacts have settled |
| Adjust coil voltage for different types of relays | X | | | | Adjust coil voltage from 2V to 15V with an accuracy of 100mV, in order to work with 5V, 9V, and 12V relays, as well as relays that fall within or near that specification |
| Measure the current consumption of the relay coil | X | | | | Measure from 1mA to 450mA with an accuracy of 1mA |
| Measure the impedance of a coil | X | | | | Measure from 20**Ω** to 500**Ω** |
| Accurately measure time in order to perform timed tests | X | | | | Measure time with an accuracy of 100ms |
| Perform tests at specified intervals | X | | | | Perform tests from 500mHz to 10Hz |
| | X | | | | In order to work with relays up to 12V and 450mA, as well as power the rest of the |

| | | | | | |
|---|---|---|---|---|---|
| Use an off-the-shelf power AC/DC supply | | | | | circuits, the power supply must have greater than 12V and 450mA (plus what is required by rest of the system capability) |
| Interface with a PC | X | | | | |
| Send collected data to a PC | X | | | | |
| Receive commands from a PC | X | | | | |
| Archive all data to a database | X | | | | |
| Provide search tools for the database | X | | | | |
| Provide comparison tools for the database | X | | | | |
| Provide sorting for each relay characteristic | X | | | | |
| Set a voltage threshold for activation and deactivation time | | X | | | Voltage threshold from 50% to 90% of applied voltage |
| The PC should automatically detect that the relay analyzer is connected | | X | | | |
| The PC software should be able to compare up to three relays at once | | X | | | |
| Measure multiple relays at once | | | X | | |
| Adapt tests to non-mechanical relays, such as reed relays and solid-state relays | | | X | | |
| Adjust coil voltage to 25V, in order to be able to test relays with coil voltages | | | X | | This would also require a different system power supply |

| | | | | | |
|---|---|---|---|---|---|
| from 15V to 24V | | | | | |
| Measure specific characteristics about relay contacts | | | | X | Max contact current and max contact voltage |

[*] Activation time will be defined as the time from the microcontroller switching the output pin, to the *first time* the relay-under-tests' contacts reach the programmed threshold voltage (from 10%-90%).

[**] Deactivation time will be defined as the time from the microcontroller switching the output pin, to the *first time* the relay-under-tests' contacts reach the programmed threshold voltage (from 10%-90%).

[***] Bounce will be defined as the voltage changes between the end of [1] or [2], and when the voltage settles within a few percent of the nominal voltage

**SMART Tables**

| **Task/Goal Title** | Measurement of Relay Characteristics |
|---|---|
| **Specific** | Measure the activation and deactivation times of a relay, measure and characterize the switch-on and switch-off bounce, measure the current consumption of the relay coil, measure the impedance of a coil.<br><br>This will require assembly and testing of the required circuits using multimeters and oscilloscopes. As well as an understanding of relay functionality to test each characteristic properly. |
| **Measurable** | If the analyzer can measure the characteristics listed in the specific section, then this task has been successful. |
| **Attainable** | The project was chosen, and scheduled so that there the members have the knowledge and skill necessary to complete this task. Additionally, the schedule has been designed to allow adequate time to complete. |
| **Relevant** | This is the main goal of the entire project. Without this, all other functions are without purpose. |

| | |
|---|---|
| **Time-Bound** | This task is dependent on the hardware and embedded software being completed and must be completed before final testing can begin. |

| **Task/Goal Title** | Store/Loads data |
|---|---|
| **Specific** | Store all data collected during a test to a database and be able to recall it.<br><br>This will require an IDE, most likely for C, and knowledge of databases. |
| **Measurable** | This task will be successful if the computer properly stores and loads the test data |
| **Attainable** | Considering the group members previous experience with databases and programming, this task should be accomplishable. The only issue may come from the format of the files. |
| **Relevant** | This is vital for comparisons and further analysis by the user. |
| **Time-Bound** | Since this can be tested independently of other functions, it is not reliant on any of them. It does need to be completed before the end of the project. |

| **Task/Goal Title** | Connects to computer |
|---|---|
| **Specific** | The device and computer will be able to send and receive data via USB in order to save/ display data on the PC and send test instructions to the analyzer.<br><br>This will require the STM microcontroller and the assembly and testing of the communication circuit, and IDE for C programming and multimeters/oscilloscope. Knowledge of serial communication and C will |

| | |
|---|---|
| | be needed. |
| **Measurable** | This task will be successful if the analyzer and computer can send data back and forth reliably. |
| **Attainable** | Given the members past experience with serial communications and STM boards, this task will be obtainable with few to no issues. |
| **Relevant** | This function allowed the data to be stored and displayed in a way that is meaningful to the user. |
| **Time-Bound** | This will need to be finished to fully test the UI functionality. |

| | |
|---|---|
| **Task/Goal Title** | Displays Information (UI) |
| **Specific** | The computer UI will display the data from the analyzer and the database, as well as test settings that can be set.<br><br>This will require design and coding knowledge of UIs, likely in C and an IDE for coding in C. Additionally, a multimeter and oscilloscope will be needed. |
| **Measurable** | By sending known data to the UI, it can be determined if the information is displayed properly. Additionally, the voltage, current, sigals, etc. on the analyzer can be measured to determine that the test settings displayed are accurate. |
| **Attainable** | Given the tools available and the members previous UI design and programming experience, this task should be attainable. |
| **Relevant** | It is vital that the information is displayed so the user can interact with the device and properly read the data collected. |
| **Time-Bound** | Since the outputs of the analyzer can be tested using other tools, it is not necessary to have the UI finished until the end of the project. |

| **Task/Goal Title** | Accepts Targeted Relay Types |
|---|---|
| **Specific** | The relay analyzer will be able to test several types of electromechanical relays. The target relay types are signal, power, RF, and automotive. |
| **Measurable** | This will be successful if the relay analyzer can get consistent, accurate results from each type of relay. |
| **Attainable** | Every type of relay that the relay analyzer will be able to test are electromechanical, and therefore have very similar characteristics. Successfully testing one type of relay ensures the task is attainable for all types of target relays. |
| **Relevant** | The ability to test several types of relays makes the project more versatile and capable for a broader range of engineers. |
| **Time-Bound** | Must be completed before measurements of relays can be taken |

# System Design

The system is divided into three parts: the hardware, the microcontroller firmware, and the software. The hardware is shown in the following block diagram:

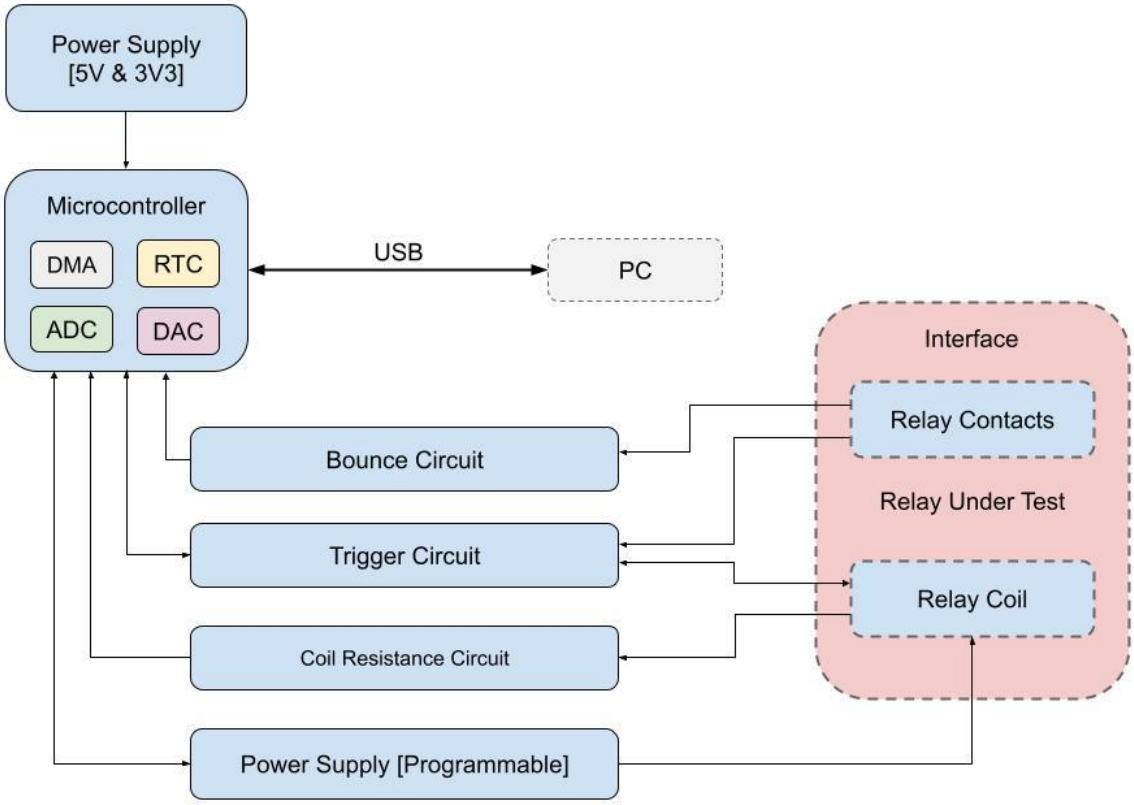



*Figure 1: System hardware block diagram*

The central part of the system is the microcontroller. The microcontroller uses its onboard peripherals and gpio to interface with the other test PCBs. The various test PCBs each have a single function, except the variable power supply, which also provides 3V3 and 5V power to the test PCBs. The test PCBs interface with the relay under test via a fifth PCB. Lastly, the microcontroller communicates with the PC using USB.

The firmware runs FreeRTOS and utilizes a state machine to perform each test at the appropriate time. Each test has its own code which controls any associated peripherals and/or GPIO.

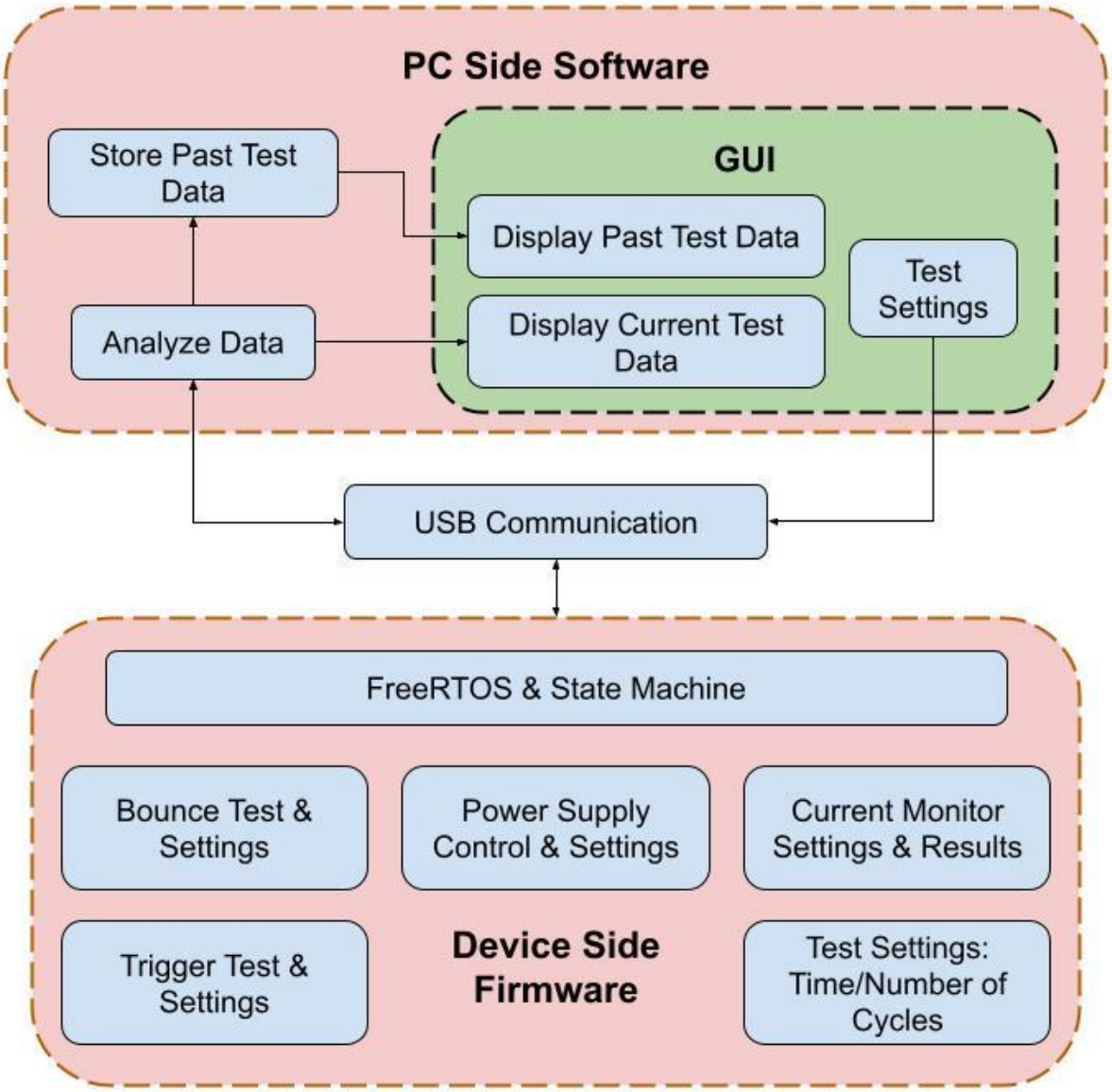



*Figure 2: System software block diagram*

The PC software has three main functions. First, the software converts all received data to a form that is more human-readable. This is critical to the second function, which is to display the data using the GUI. Data will be displayed using number displays, as well as graphs. The third function of the software is to manipulate test settings, and send them to the microcontroller.

# Design of Functional Units

This section outlines the purpose, lists major components, an explanation of the principle of operation of each functional unit, and a simulation.

## Bounce Circuit

Purpose:

Measure the bounce characteristics of the tested relay.

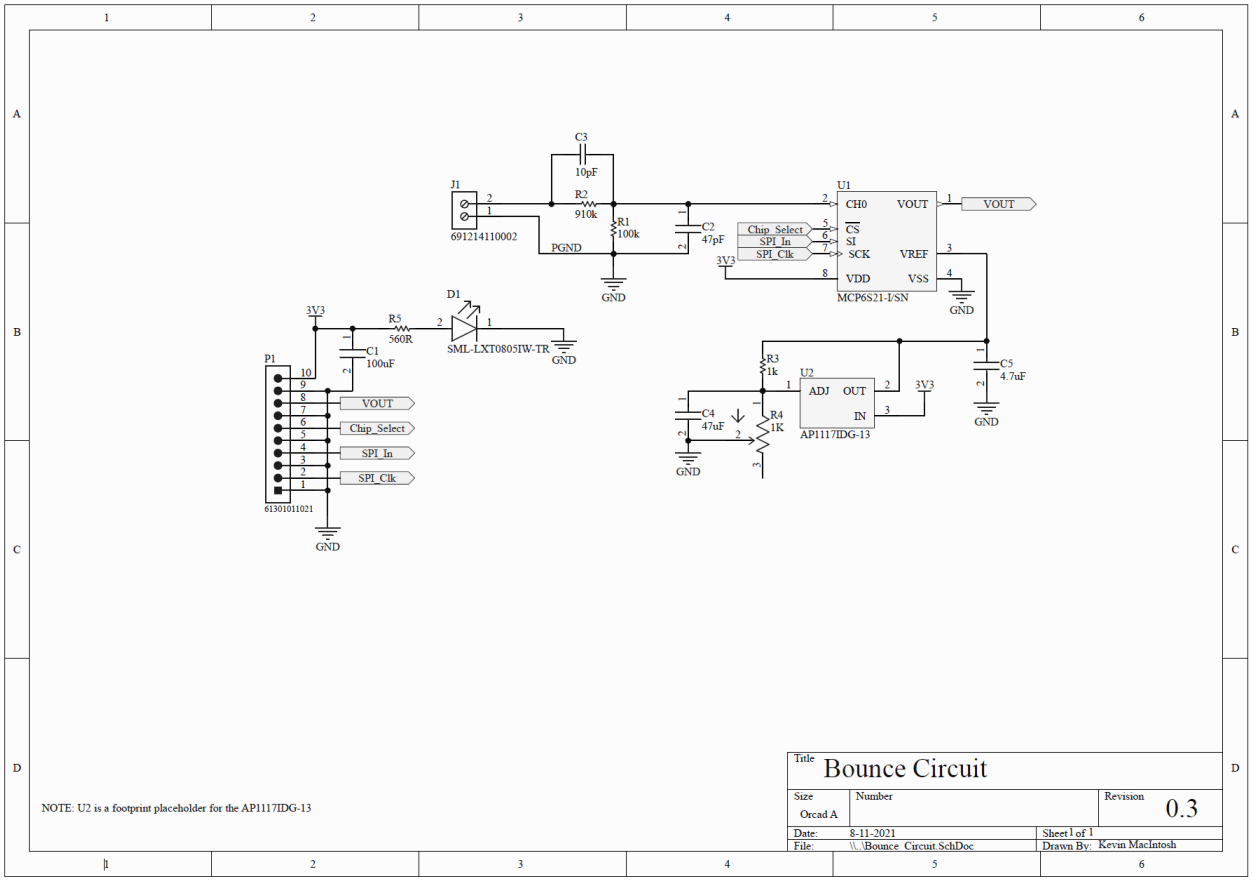



*Figure 3: Bounce Circuit Diagram*

Major Components

- MCP6S21 Programmable Gain Amplifier
- AP1117 linear voltage regulator

Principle of operation:

Upon the relay being activated, the microcontroller will capture the waveform data from the bounce circuit. The circuit monitors the voltage output of the relay, which is sent to the amplifier chip MCP6S21-I/SN, which uses a 0.5V reference voltage to drive the A/D conversion. The reference voltage is provided by the AP1117IDG-13 IC and associated circuit, trimming the 1kOhm pot until a voltage of 0.5V is achieved (achieves +/-5.5V range). The MCP6S21-I/SN has the ability to use multiple channels, which are not in use in the current design, but may be used should future designs need to test more than one relay at once. The range is calculated by:

$$Range = V_R \times \frac{(R_1 + R_2)}{R_1}$$

Simulation:

A DC source of 5V was used to simulate input and DC source was used in replacement of the reference voltage sub-circuit due to the chip not existing in the simulation.

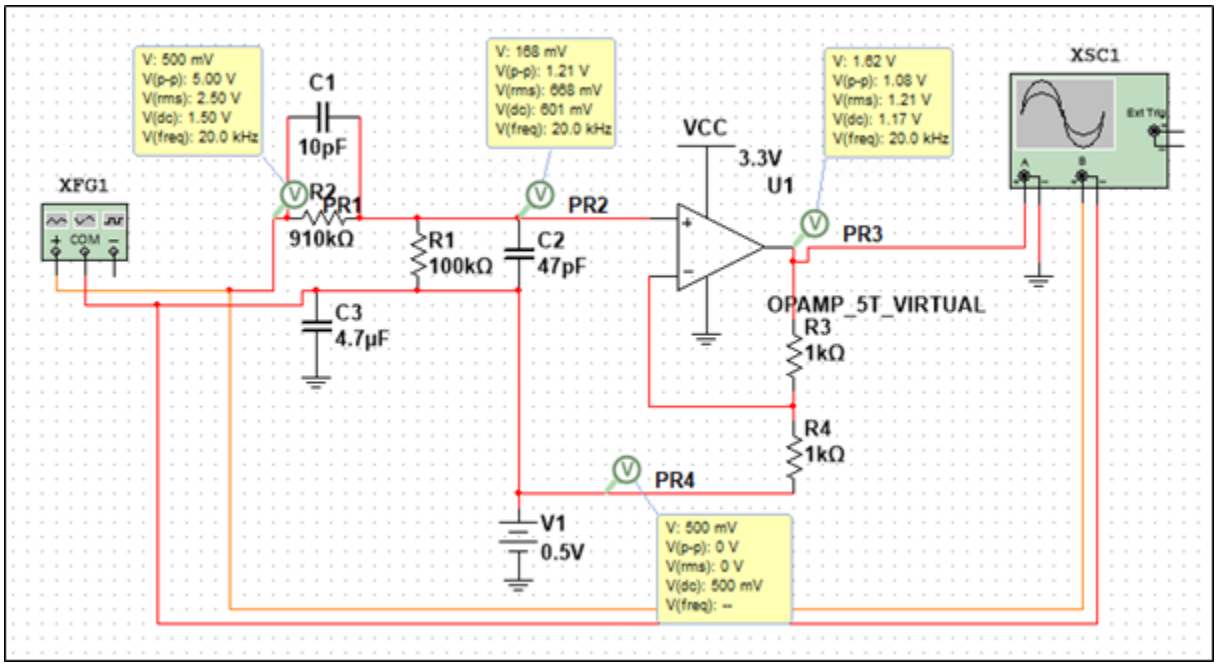



*Figure 4: Bounce Circuit Multisim Simulation*

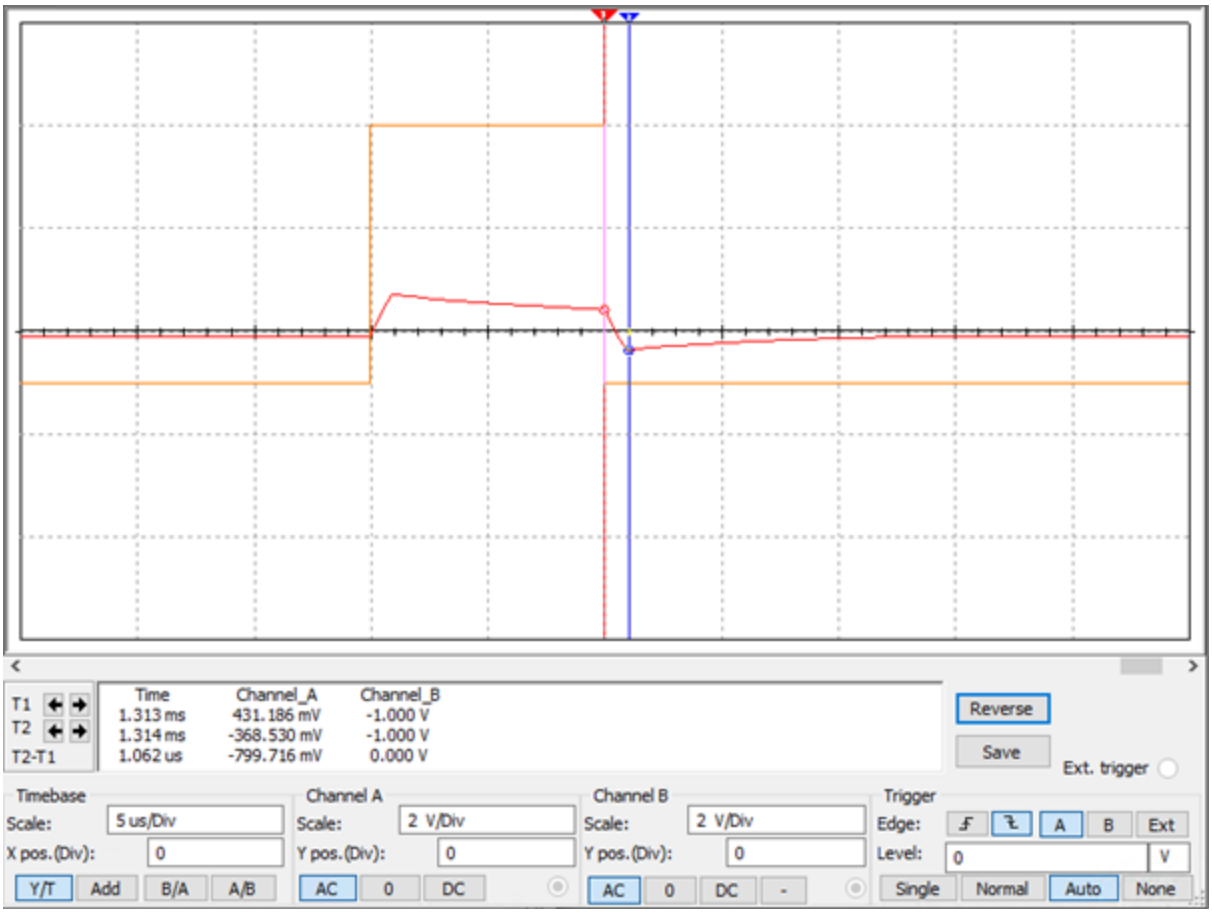



*Figure 5: Simulated bounce circuit output signal (red) compared with the input signal (orange) showing the delay on the edges to be about 1µs.*

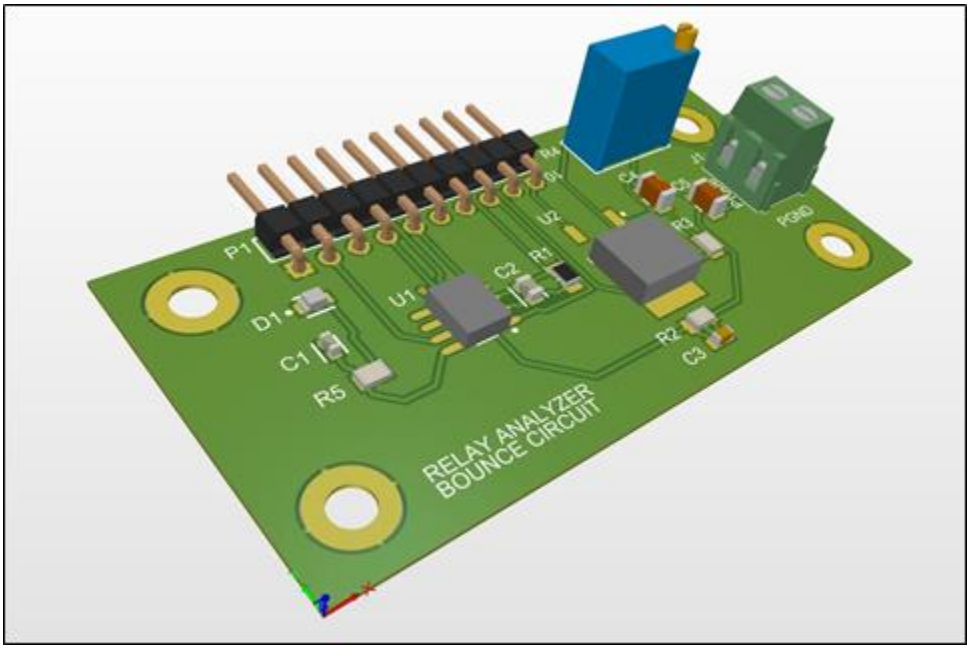



*Figure 6: Bounce Circuit Altium Render.*

# Trigger Circuit

Purpose:

Measure the time from microcontroller activation/deactivation signal to switching of the relay contacts

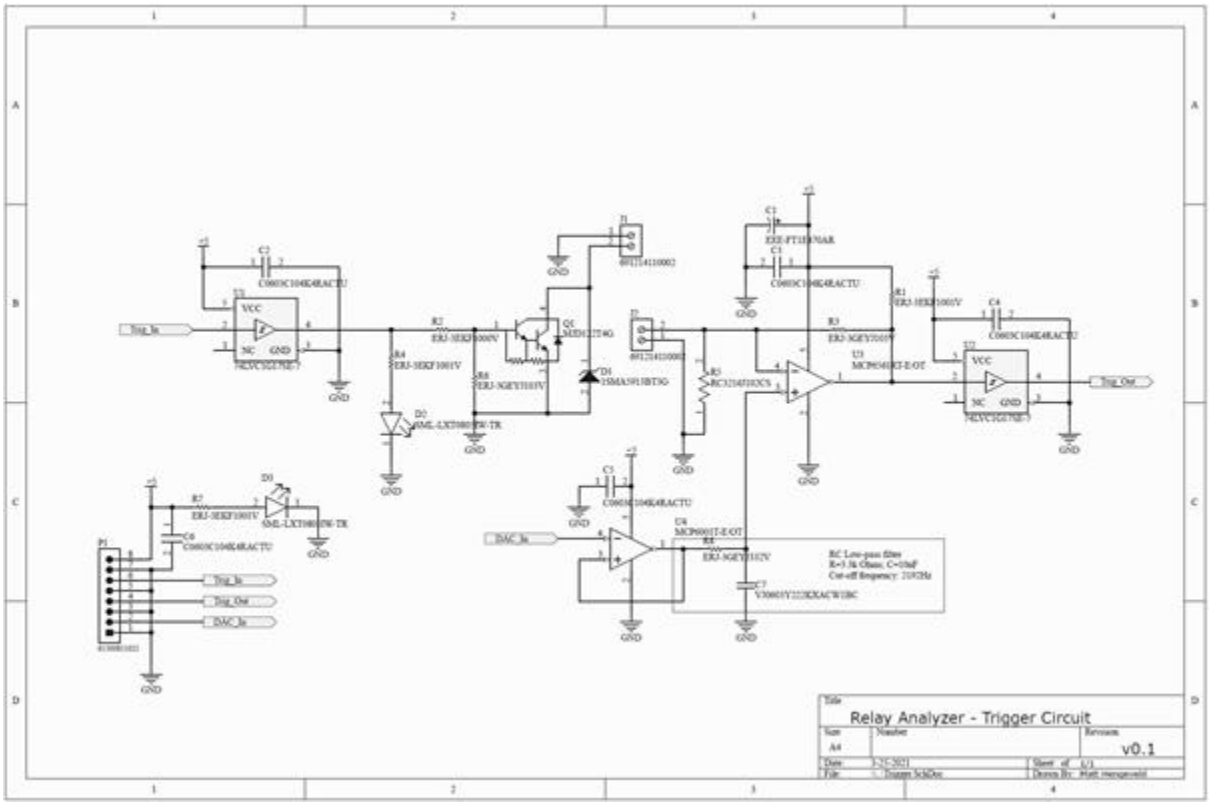



*Figure 7: Trigger Circuit Diagram.*

Major Components:

- 74LVC1G17 Schmitt triggers
- MJD122T4G Darlington transistor
- 18V Zener diode
- MCP6561 comparator
- MCP6001 rail-to-rail opamp

Principle of operation:

The trigger circuit activates the relay coil using a PNP Darlington transistor and a signal from the microcontroller. The Darlington transistor ensures the fastest possible switch time. The signal is buffered using a non-inverting Schmitt trigger buffer, so the signal has a fast transition.

The PNP transistor is protected from the relay coil back EMF by a Zener diode in reverse direction, and across the emitter-collector pins. The Zener diode shunts reverse back EMF voltage to ground. When the relay contacts switch, a voltage on one contact appears on the other, between it and a small load resistor. This voltage is sensed by a high-speed comparator, with threshold voltage controlled by a DAC on the microcontroller. The output signal is then once again buffered by a non-inverting Schmitt trigger in order to drive long output wires. The DAC input is buffered by a rail-to-rail op amp with an output low-pass filter.

Simulation:

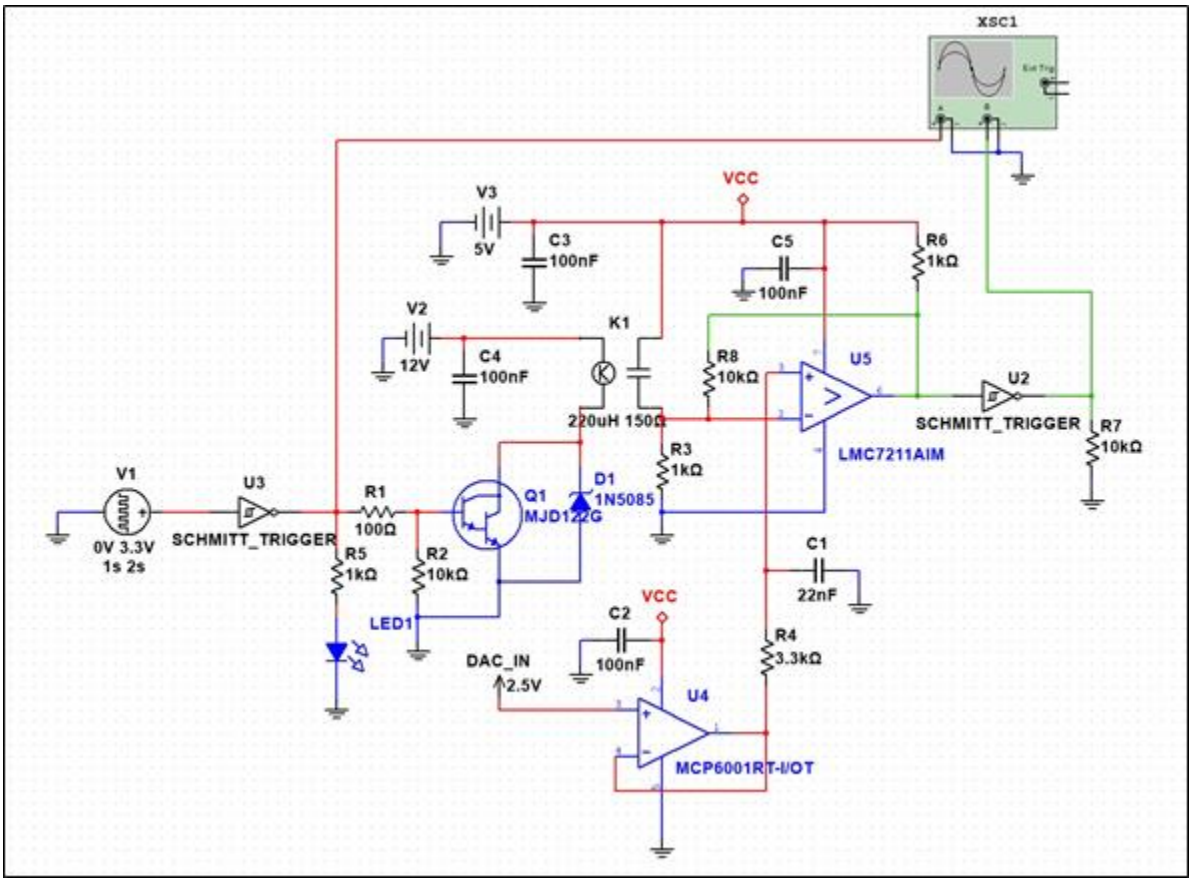



*Figure 8: Trigger Circuit Multisim Simulation.*

The simulated circuit is very close to final circuit design, and shares the same principle of operation. There are two parts where the simulation fails to accurately represent real life:

- The relay coil inductance
- The delay between relay coil activation and relay contact switching

Increasing the relay coil inductance to values from real-world relays will crash the simulation. This appears to be an effect of time consuming and complex calculations that exceed the simulation boundaries.

The relay models included with NI Multisim do not include a delay between the activation of the relay coil and the switch of the relay contacts. The delay in the signal trace seen below is propagation delay through the simulated non-inverting Schmitt triggers and comparator. Except for the two issues above, the circuit simulates exactly as predicted.

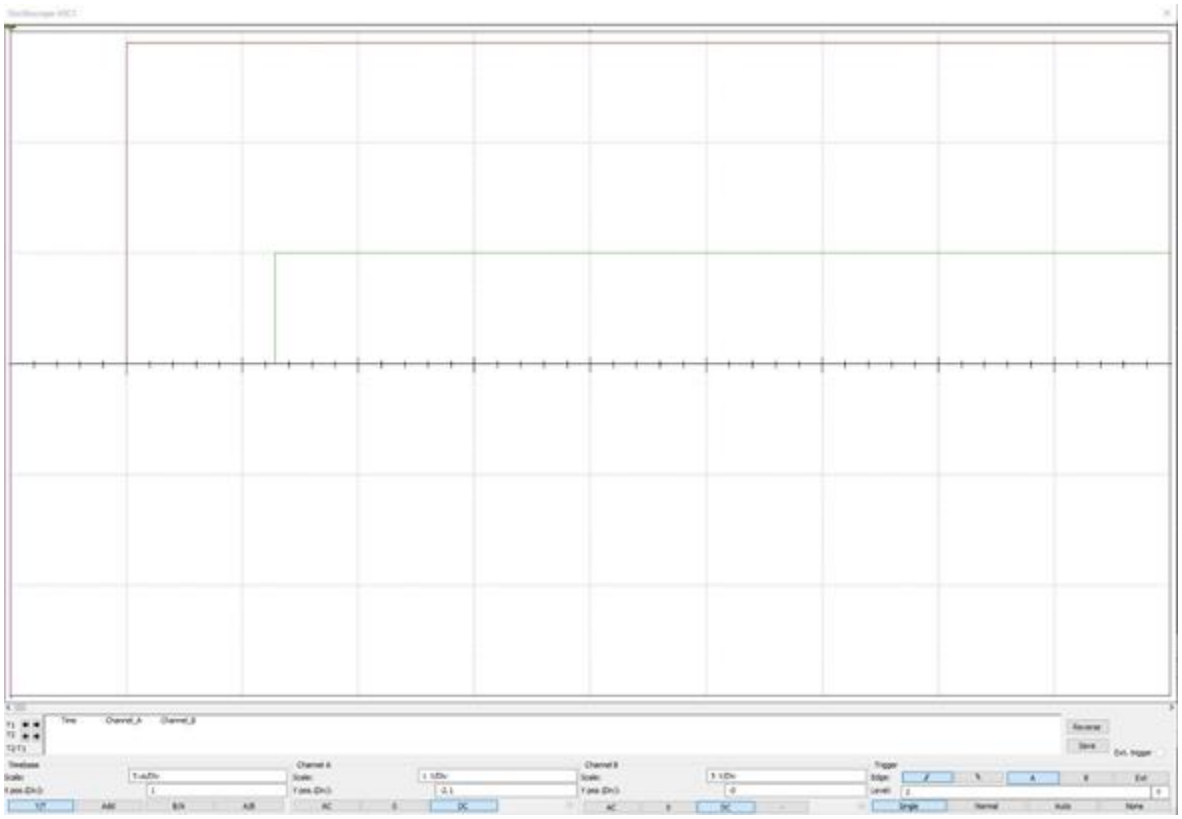


*Figure 9: Input signal (red) and output signal (green) showing and in-phase output signal, and the propagation delay between the two signals (approx. 7us).*

The above oscilloscope trace shows the input signal and output signal in phase with each other. This is assuming the relay contacts are connected in a Normally Open configuration.

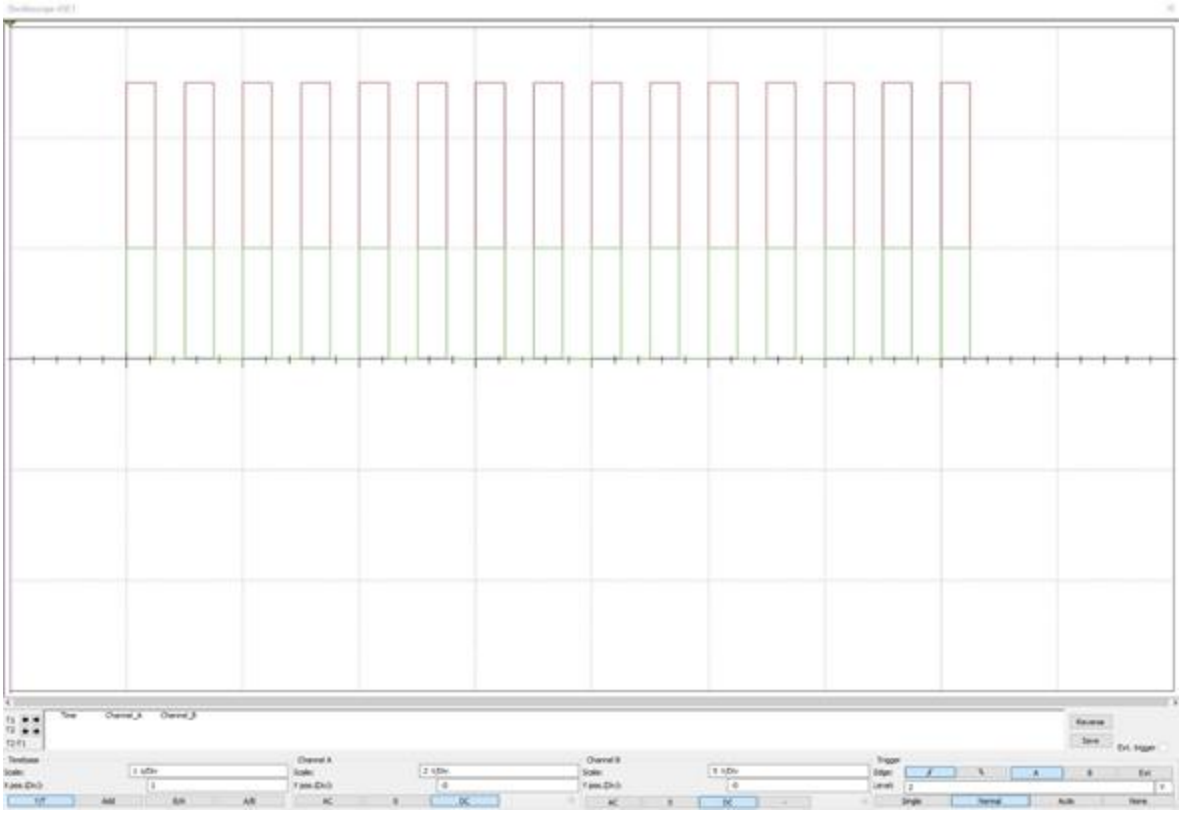


*Figure 10: Input signal (red) and output signal (green) showing repeated cycles.*

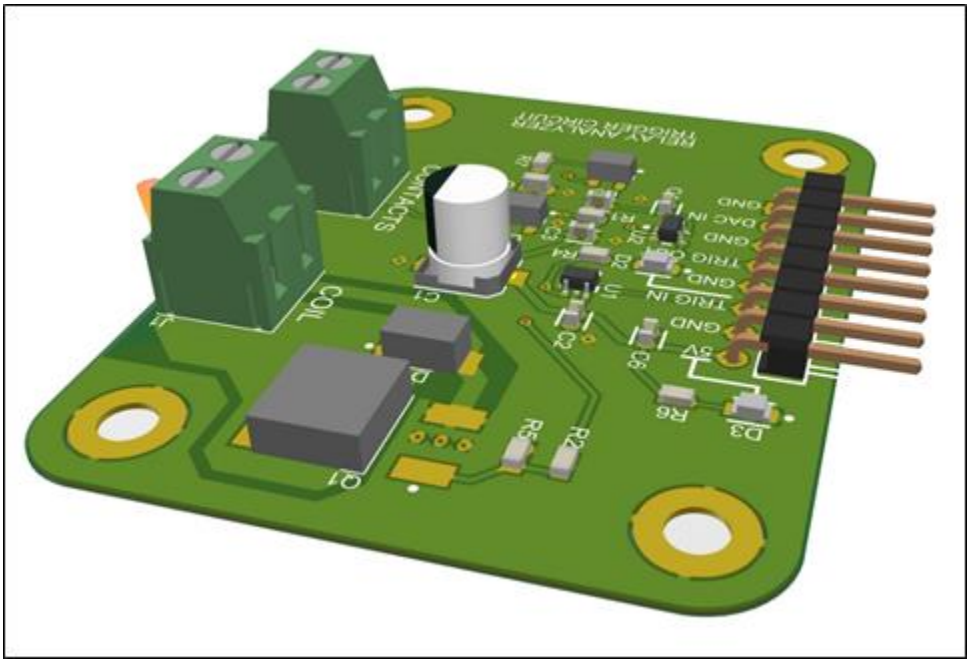



*Figure 11: Trigger Circuit Altium Render.*

# Interface Circuit

Purpose:

To switch each test circuit in with the relay under test, and to provide each test circuit with a connection to the relay under test.

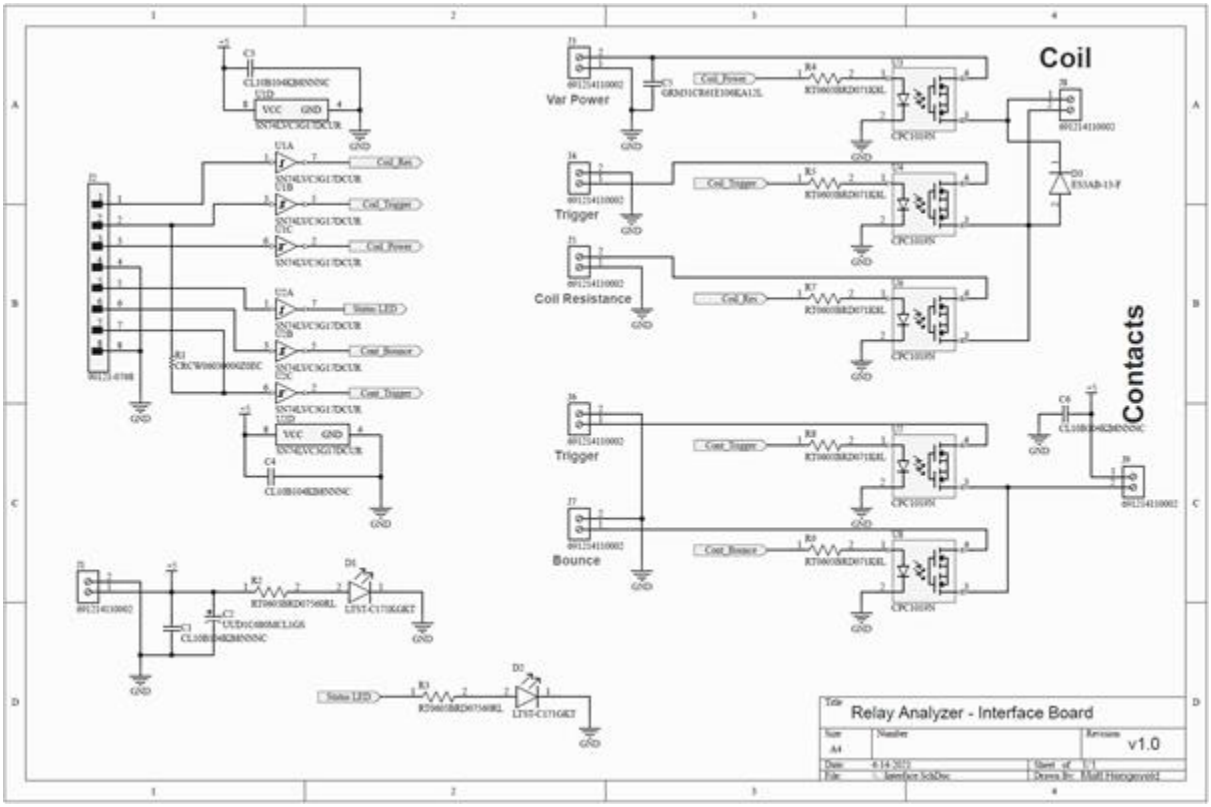



*Figure 12: Interface Board Circuit Diagram.*

Major Components:

- SN74LVC3G17 non-inverting Schmitt triggers
- CPC1019N Solid State Relays

Principle of Operation:

The interface circuit switches in and switches out test circuits using solid-state relays (SSRs) and signals from the microcontroller. The signals are buffered by non-inverting Schmitt triggers, which can drive the SSRs with the required current to switch.

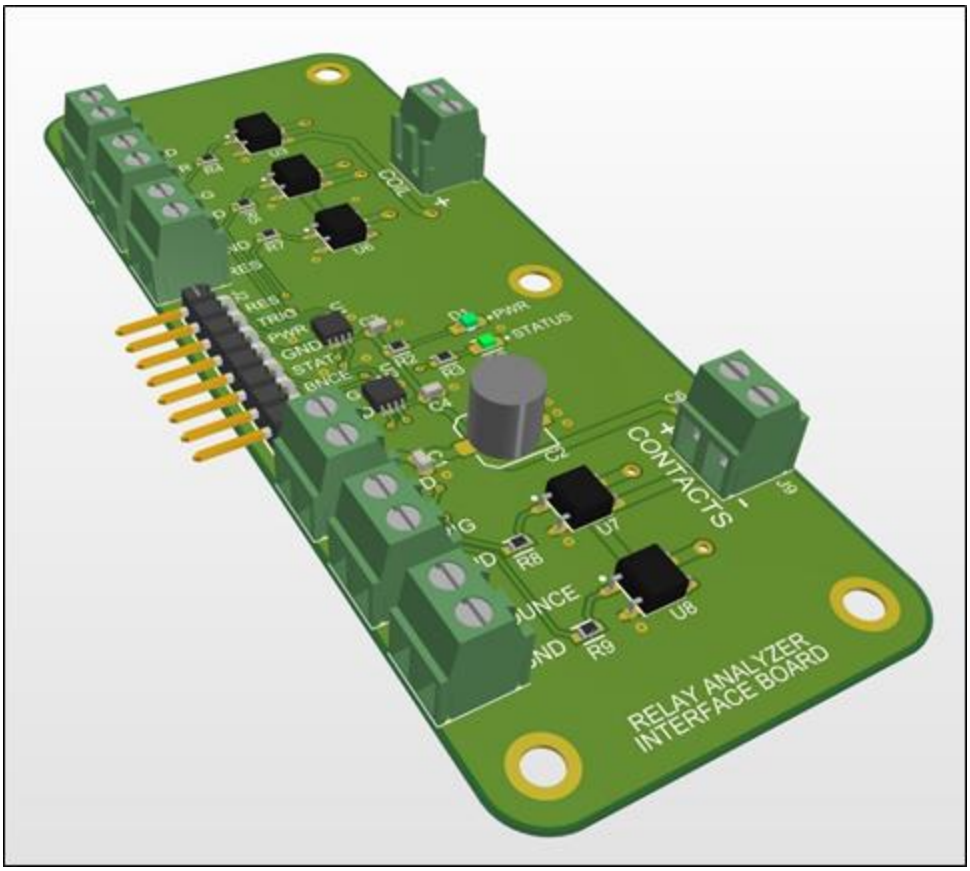



*Figure 13: Interface Board Altium Render.*

# Power Supply & Programmable Power Supply

Purpose:

To generate the needed voltages from the input voltage. The programmable power supply will generate a programmable voltage between 3.3V and 12V, and the other supplies will generate 5V and 3.3V for the test circuits. A power monitor measures the programmable power supply output voltage and current.

Power Budget:

The following power budget calculates the needed current capability of the 5V and 3V3 voltage rails, based on all the components in the system, and their worst-case maximum current draw in their respective circuits.

| 3V3 Power Budget | | |
|---|---|---|
| **Component** | **Quantity x Current [mA]** | **Total Current [mA]** |
| **MCP6S21** | 1 x .0 | 1.0 |
| **AP1117** | 1 x 6.0 | 6.0 |
| **LEDs** | 2 x 2.2 | 4.4 |
| **Total** | | **11.4** |
| **Total w/ 25% safety margin** | | **14.3** |

*Table 1: 3V3 power budget*

| 5V Power Budget | | |
|---|---|---|
| **Component** | **Quantity x Current [mA]** | **Total Current [mA]** |
| **SN74LVC1G17** | 2 x 5.0 | 10.0 |
| **SN74LVC2G17** | 1 x 10.0 | 10.0 |
| **SN74LVC3G17** | 1 x 0.5 | 0.5 |
| **MCP6561** | 1 x 4.0 | 4.0 |
| **MCP6001** | 1 x 4.0 | 4.0 |
| **CP1019N** | 6 x 3.0 | 18.0 |
| **MCP6241** | 1 x 4.0 | 4.0 |
| **INA226A** | 1 x 2.0 | 2.0 |
| **TPS54240** | 1 x 2.0 | 2.0 |
| **LMR16020** | 1 x 1.0 | 1.0 |
| **TPS73633** | 1 x 0.5 | 0.5 |
| **NUCLEO-F767ZI** | 1 x 500.0 | 500.0 |
| **LEDs** | 5 x 2.0 | 10.0 |
| **5V to 3V3 (η = 50%)** | | 18.8 |
| **Total** | | **584.8** |
| **Total w/ 25% safety margin** | | **754.5** |

*Table 2: 5V power budget*

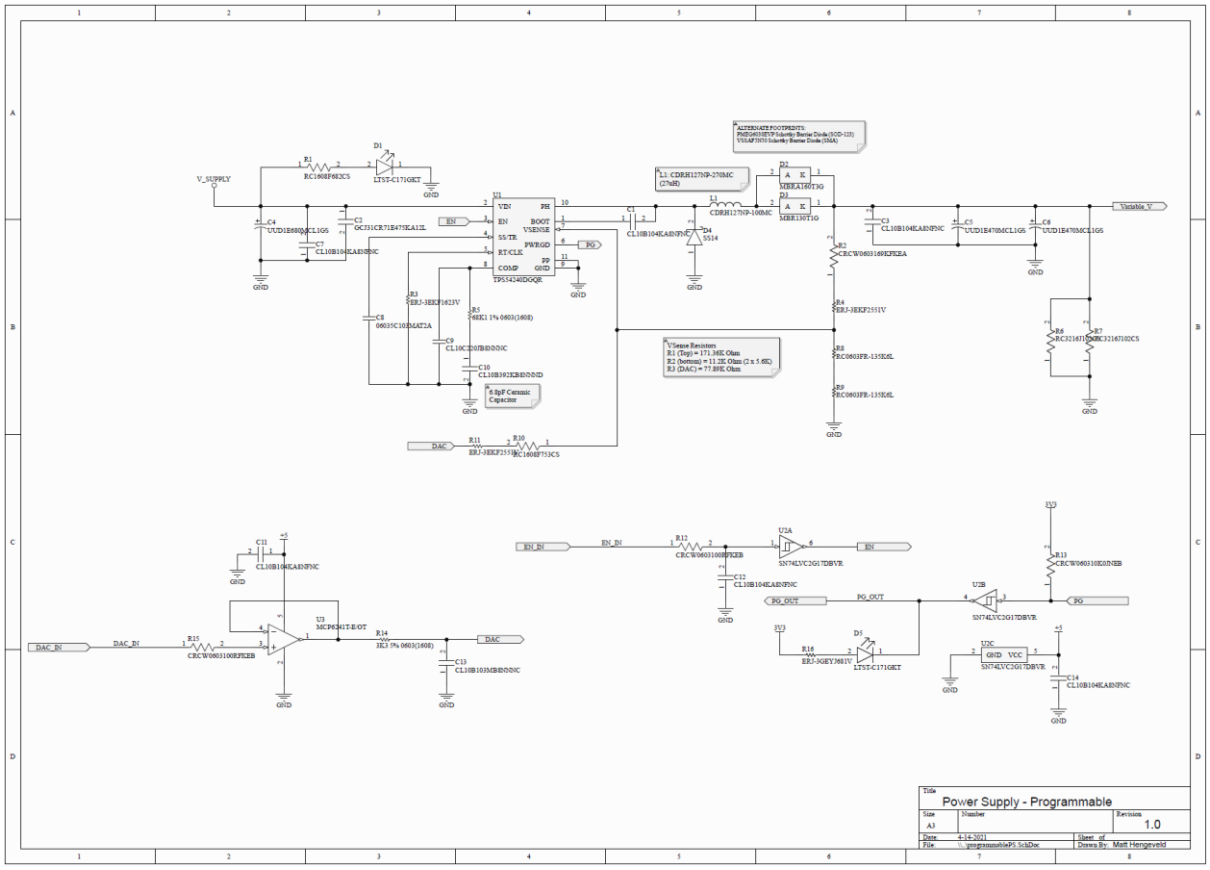



*Figure 14: Programmable Power Supply Circuit Diagram.*

Major Components:

1. TPS54240 buck converter
2. SS14 Schottky diode
3. CDRH127NP-220MC 22uH shielded inductor
4. PMEG6030EVP or VSSAF5N50 Schottky diode
5. MCP6241T rail-to rail op amp
6. SN74LVC2G17 non-inverting Schmitt triggers

Principle of Operation (Programmable Power Supply):

The programmable power supply is a switching buck converter, with an additional signal on the feedback pin. This signal is driven by a DAC on the microcontroller, and is buffered by a rail-to-rail op amp.

The Enable (EN) signal activates the switching controller, or puts it into a low-power state, and has a threshold of 2.5V. The Power Good (PG) signal is an open-drain output, and indicates to

the microcontroller when the output voltage is within 3% of nominal, based on the voltage on the feedback pin. Both signals are buffered by non-inverting Schmitt trigger buffers.
To protect against reverse voltage from the relay coil, a Schottky diode is included in series with the output. Because it is included within the feedback loop, the forward voltage of the diode does not affect the output voltage. A primary, as well as an alternate footprint has been included in order to increase the number of available choices of diodes.

The output voltage is smoothed by two 47uF electrolytic capacitors in parallel. These capacitors are low ESR, and combine in parallel to decrease ESR further. Two load resistors ensure a constant minimum load, so that changes in output voltage appear quickly, and are not held up by the output capacitance.

The switching controller compares the voltage on the feedback pin to the internal 0.8V reference. The voltage on the feedback pin is derived from the output voltage, through a voltage divider. An error signal is generated, and this error signal changes the PWM duty cycle, resulting in a change of output voltage. [3]

The DAC changes the voltage on the feedback pin, and therefore changes the output voltage. In this circuit, an increase in feedback pin voltage will result in a decrease in duty cycle, and a decrease in output voltage. The output voltage can be calculated using the following:

$$V_{OUT}=\left[I_{R2}-\left(\frac{V_{DAC}}{R3}\right)\right]\times R1+V_{REF}$$

Where R1 is the resistor between the output voltage and the feedback pin, and is determined by:

$$R_1=\frac{V_{min}}{I_{R2}-\frac{\left(V_{DAC\_max}-V_{REF}-V_f\right)}{R_3}}$$

R2 is the resistor from the feedback pin to ground and is determined using recommendations from the datasheet. R3 is the resistor from the DAC to the feedback pin, and is determined by:

$$R_3=\frac{V_{max}\left(D_{DAC\_max}-V_{REF}\right)-V_{min}\left(V_{DAC\_min}-V_{REF}\right)}{I_{R2}\left(V_{max}-V_{min}\right)}$$

Using the maximum and minimum output voltages from the DAC, maximum and minimum required output voltages, and the switching regulators reference voltage, all resistor values can be calculated.

For a DAC input of 0-5V, a reference voltage of 0.8V, a maximum output voltage of 13V, a minimum voltage of 2V, and R2 = 11.2KΩ, R1 and R3 are calculated to be 171.4KΩ and 78KΩ, respectively.

The output voltage vs DAC input voltage relationship can be seen in the graph below. The graph shows that an *increase* in DAC voltage results in a *decrease* in output voltage. Maximum output voltage is achieved when DAC voltage is at the minimum. The relationship is also linear.

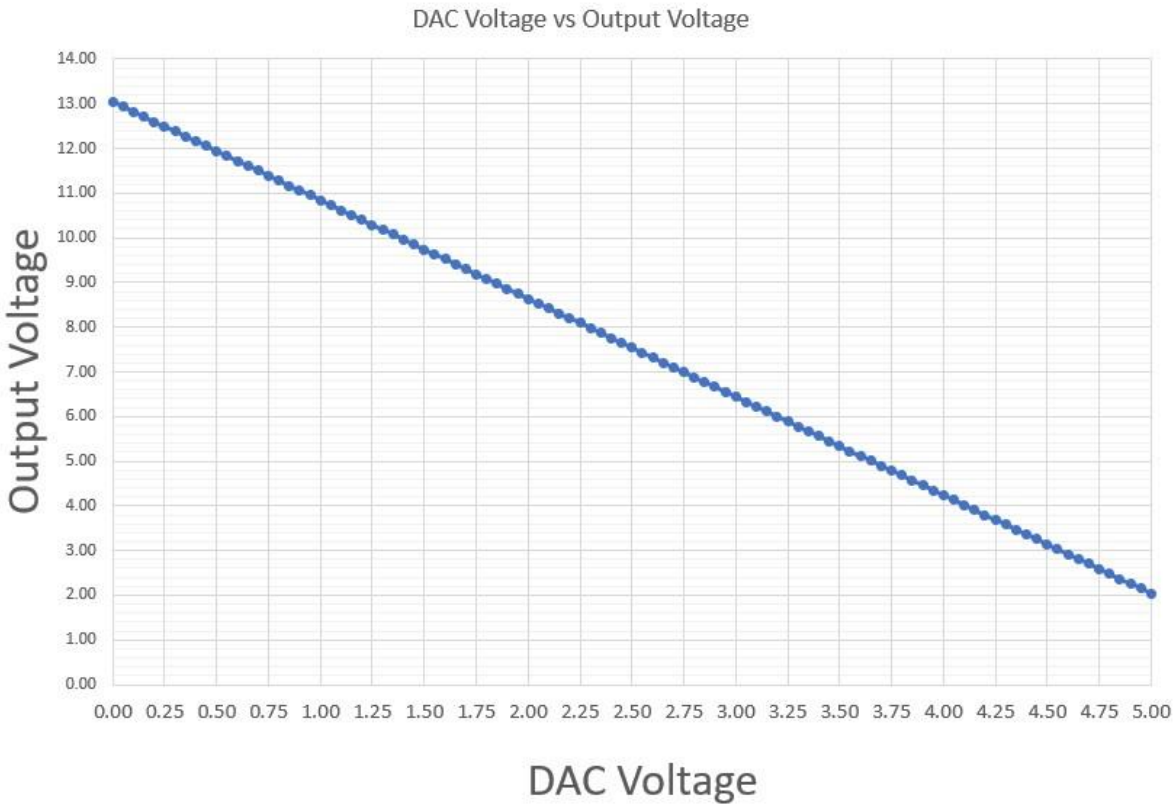



*Figure 15: Relation between microcontroller DAC voltage and output voltage of the power supply.*

Simulation:

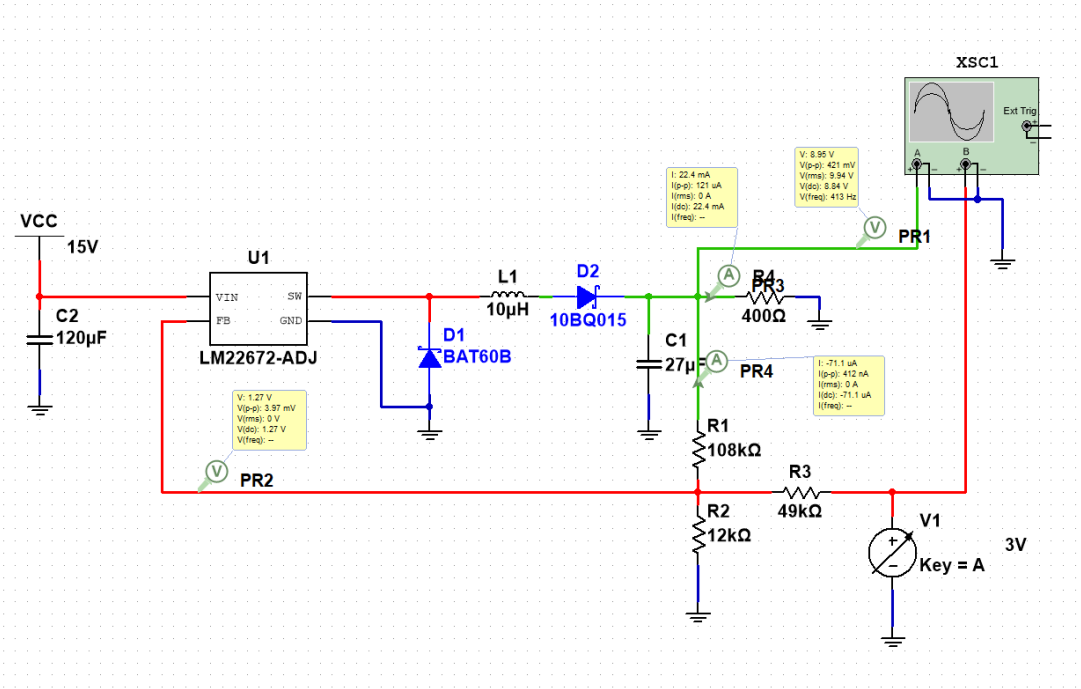



*Figure 16: Programmable power supply Multisim simulation circuit.*

The programmable power supply was simulated using NI Multisim software. The switching controller is *similar* to the component used, with one key difference: the reference voltage is 1.25V vs 0.8V. As a result, R1, R2 and R3 have been calculated for a 1.25V reference voltage.

The simulation ran extremely slow, and therefore less than 10ms of simulation time is shown in the screenshots below. A side-effect of this is the relatively long looking under and overshoot on the output voltage. In reality, this common effect of switching power supplies will not affect the test circuits, as the microcontroller will ensure accurate output voltage before applying to the relay under test, and the voltage will not change throughout the test.

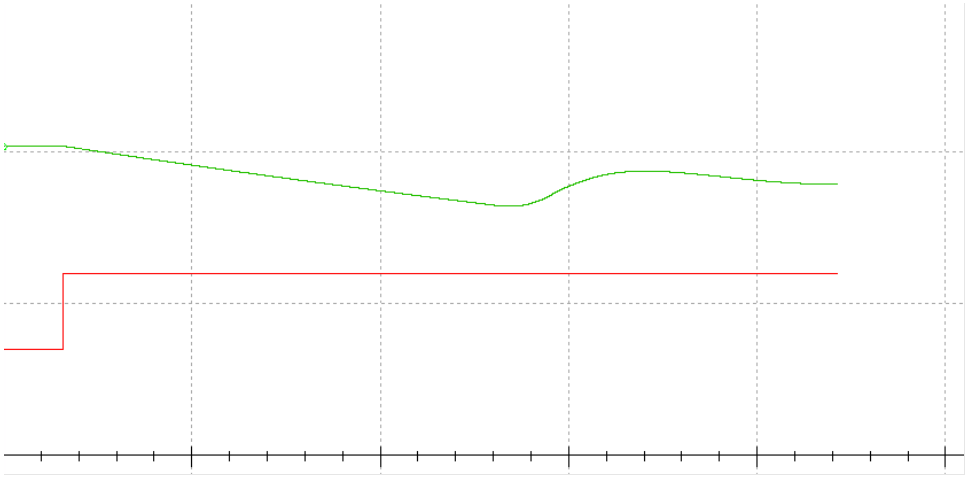


*Figure 17: Oscilloscope traces showing the DAC voltage (red) increasing, and the output voltage (green) decreasing. Shown simulation time is ~8ms, change in DAC voltage is 0.5V.*

In *Figure 17*, the output voltage can be seen decreasing, with undershoot and overshoot, in response to an increase in DAC voltage. Because the output voltage will only be adjusted when the load (the relay under test) is disconnected, there are two load resistors on the output, before the current monitor. These ensure a minimum load, and allows the output voltage to respond quickly, without being 'held up' by the output capacitor.

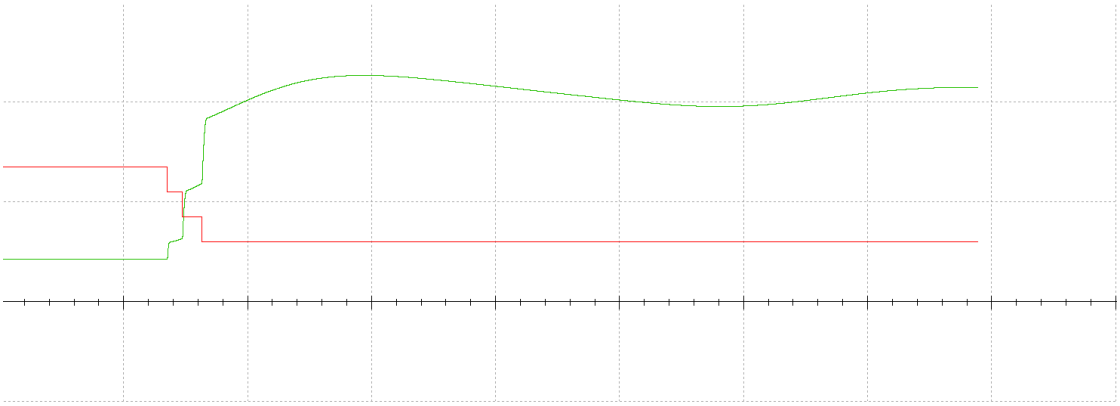


*Figure 18: Oscilloscope traces showing the DAC voltage (red) decreasing, and the output voltage (green) increasing. Shown simulation time is ~8ms, change in DAC voltage is -0.75V total.*

In *Figure 18*, the output voltage increases in response to the decreasing DAC voltage. The response is much quicker, as the output capacitor acts as the load in this case, charging up to the new output voltage. There is still overshoot and undershoot.

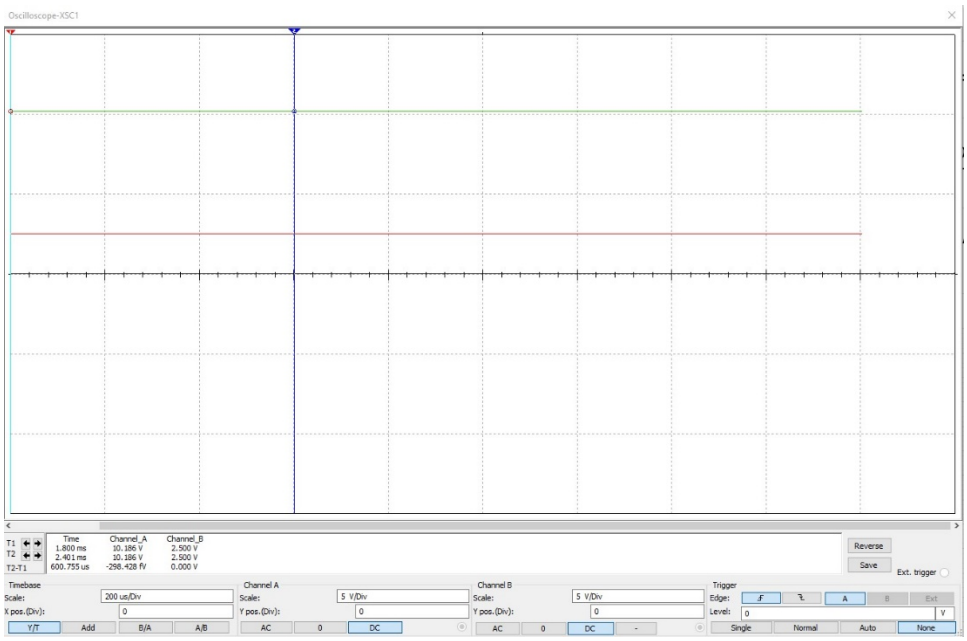



*Figure 19: Steady state output voltage (green) from a steady state DAC voltage (red).*

Using the above Vout equation, the simulation circuit, and the DAC value (2.5V) from the oscilloscope trace above, Vout is calculated to be 10.38V. The simulation gives an output of 10.186V - a difference of 1.9%.

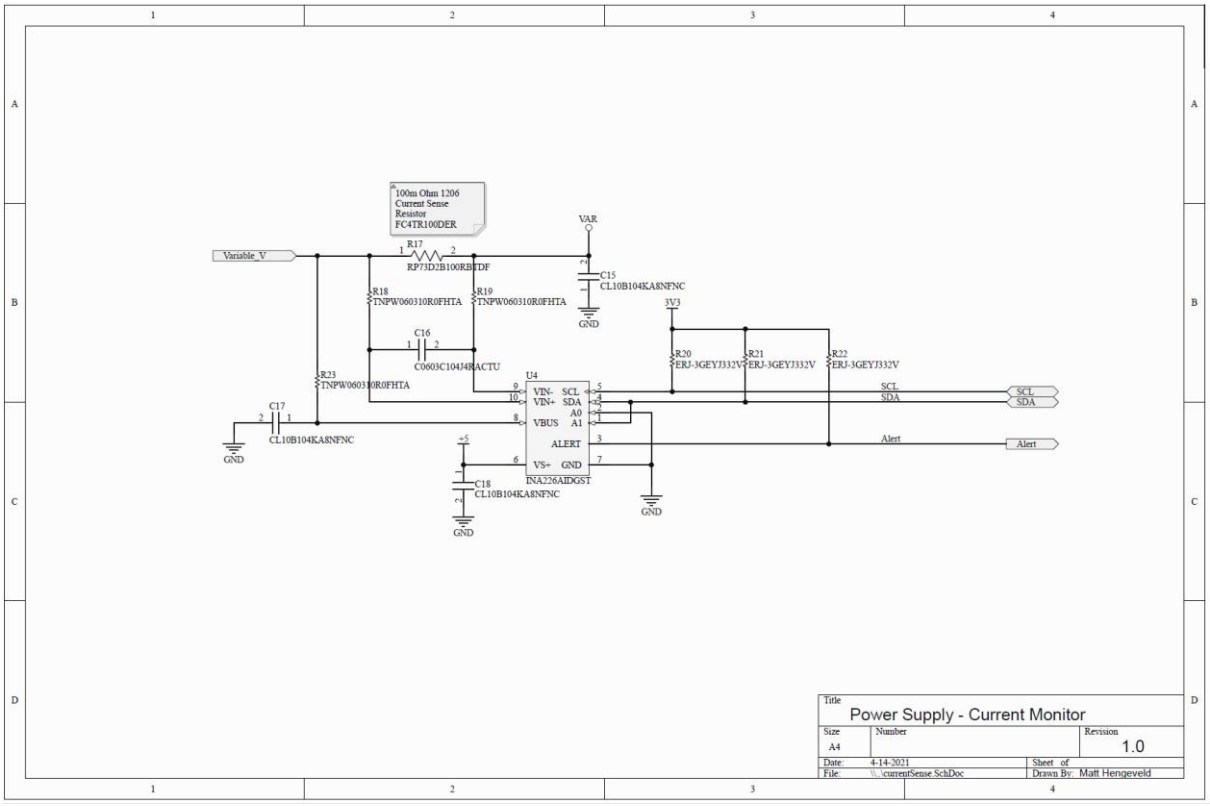



*Figure 20: Power Supply Current Monitor Circuit Diagram.*

Major Components:

- INA226AID power monitor
- FC4TR100DER 100mΩ, 0.5%, 50ppm/°C current sense resistor

Principle of Operation (Current Monitor):

The power monitor circuit uses a high-side current sense resistor to detect the output current of the programmable power supply. It also detects output voltage. The current sense resistor is connected to the power monitor using a Kelvin connection, and via 10Ω series resistors with a 100nF filter capacitor.
Current and voltage readings are sent to the microcontroller using an $I^2C$ bus.

The error due to the INA226 can be seen in the following graph from TI:

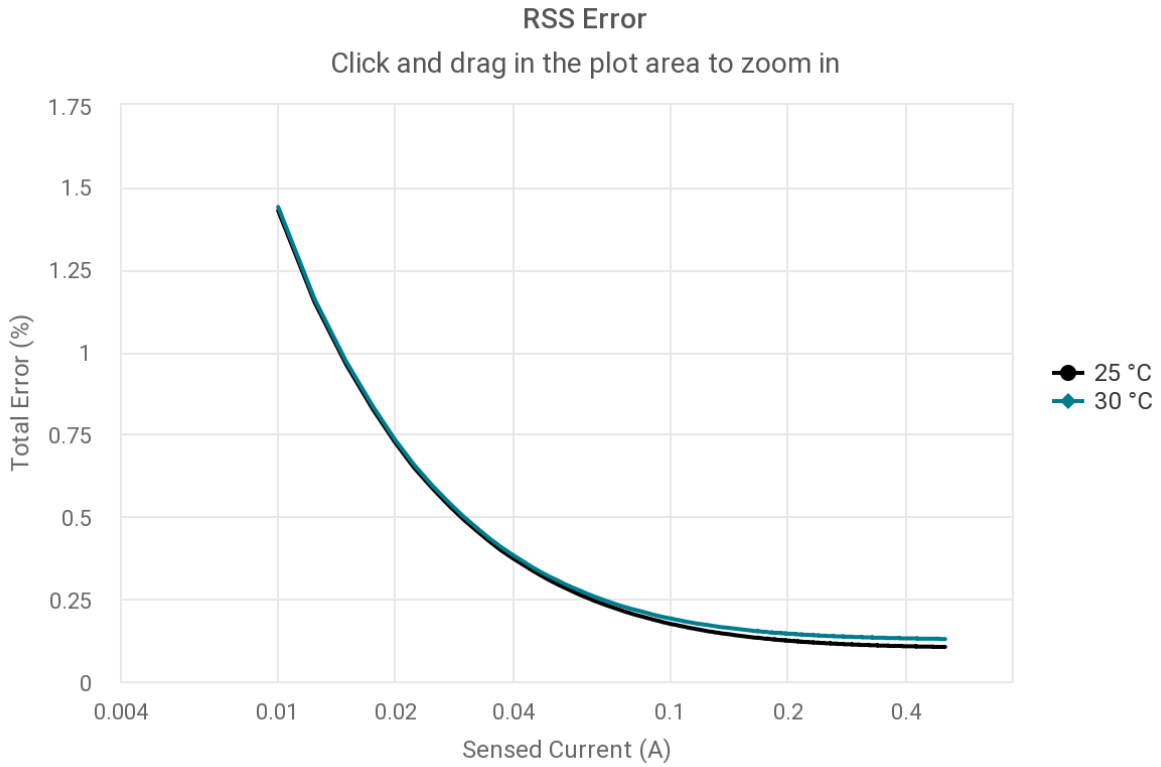



*Figure 21: The Resting Steady State (RSS) error of the INA226AID current monitor over 10mA to 500mA. This does not include error from the current sense resistor tolerance.*

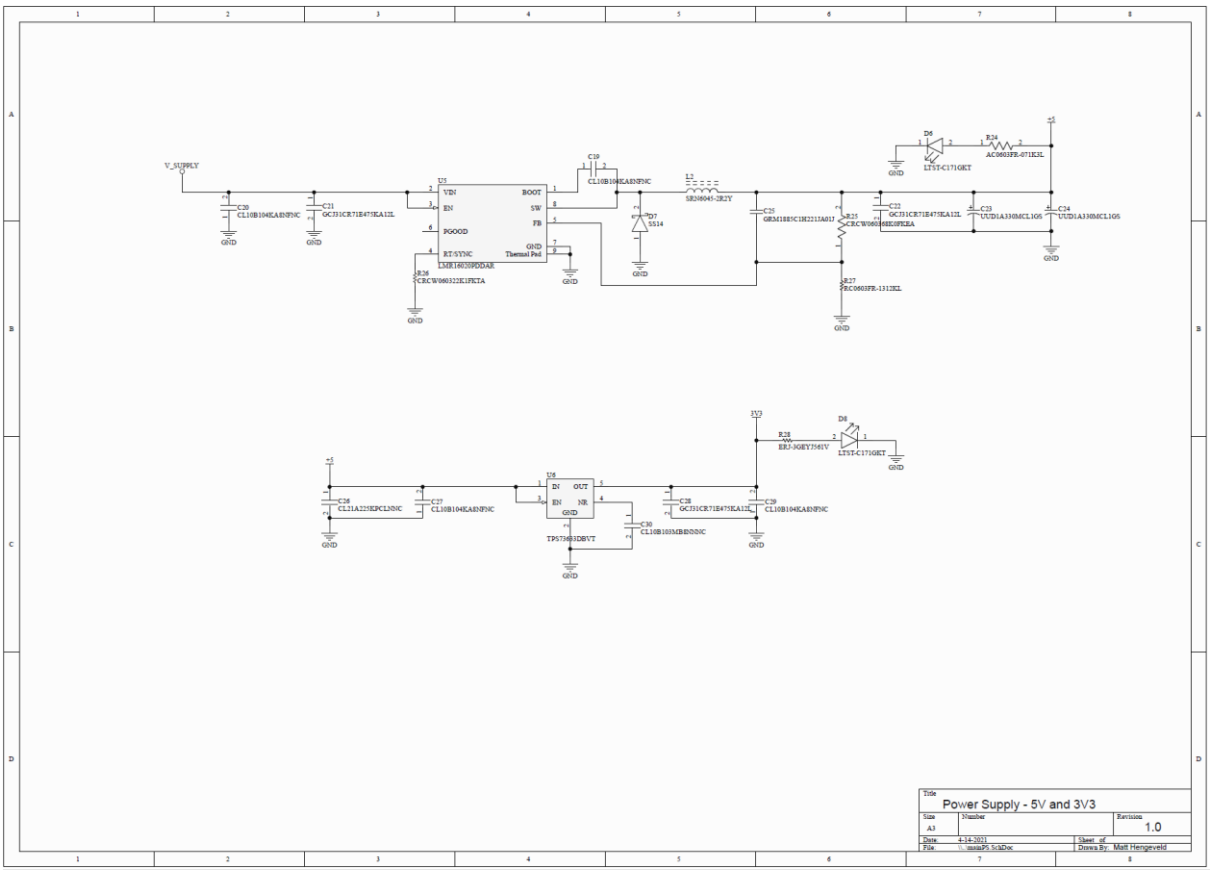



*Figure 22: 5V (top) and 3V3 (bottom) Power Supplies.*

Major Components:

- LMR16020 switching controller
- SS14 Schottky diode
- SRN6045 6.8uH inductor
- TPS73633 3V3 linear voltage regulator

Principle of Operation (5V and 3V3 rails):

The 5V rail is supplied by a switching buck regulator. The components can handle up to 1A output current.

The 3V3 rail is supplied by a linear regulator, for lowest ripple possible. The TPS73633 is a low-dropout, low-noise and high accuracy regulator with 400mA output current capability. The Noise Reduction (NR) pin allows further noise reduction in a custom frequency range by using an external capacitor. The selected 10nF NR capacitor is most effective in the 50KHz to 2MHz range, which includes the switching frequencies of both switching buck converters.  A low-ESR ceramic output capacitor further decreases output noise.

Simulation:

The following simulations are from TI Webench.

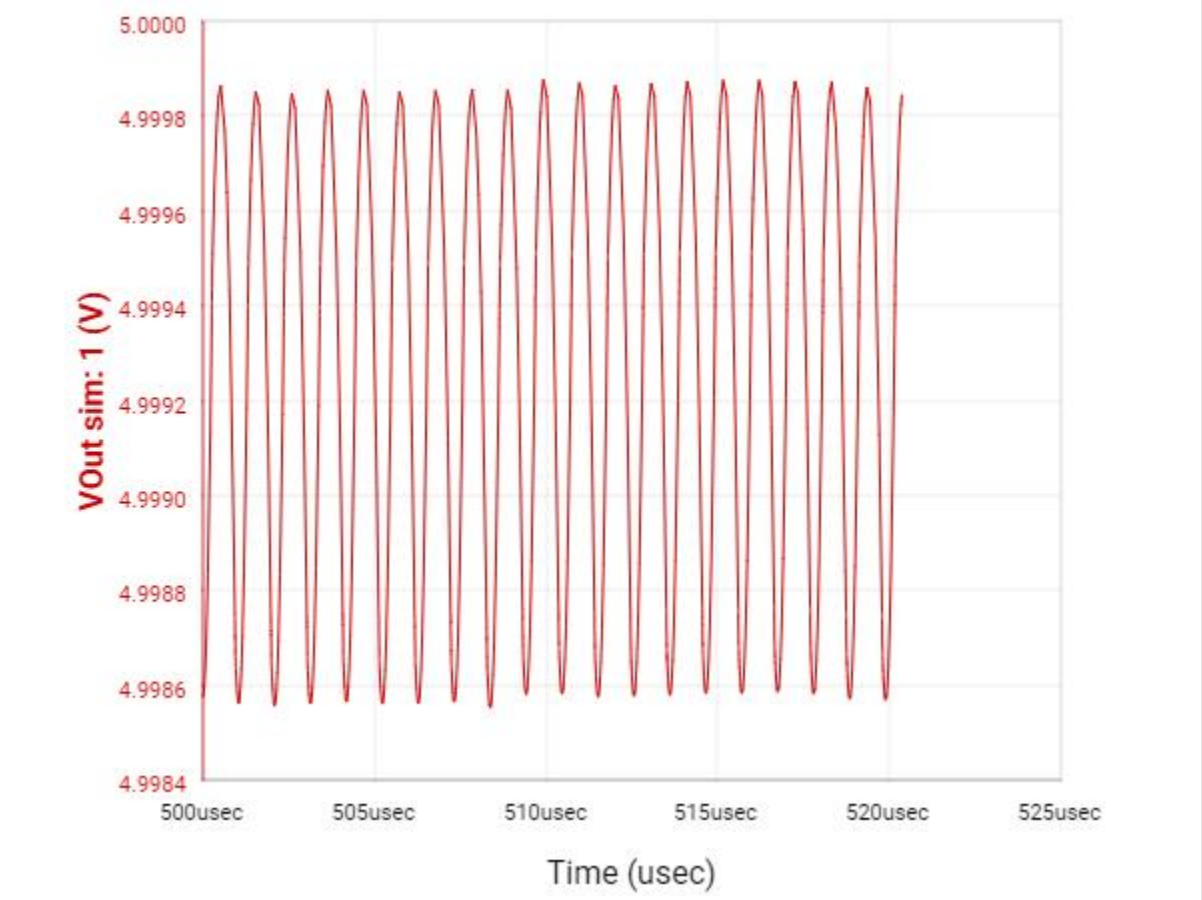



*Figure 23: TI Webench simulation results of the 5V power supply voltage output over ~521usec.*

From *Figure 23*, it can be seen that the output ripple is less than 2mVp-p and accuracy is better than 1%.

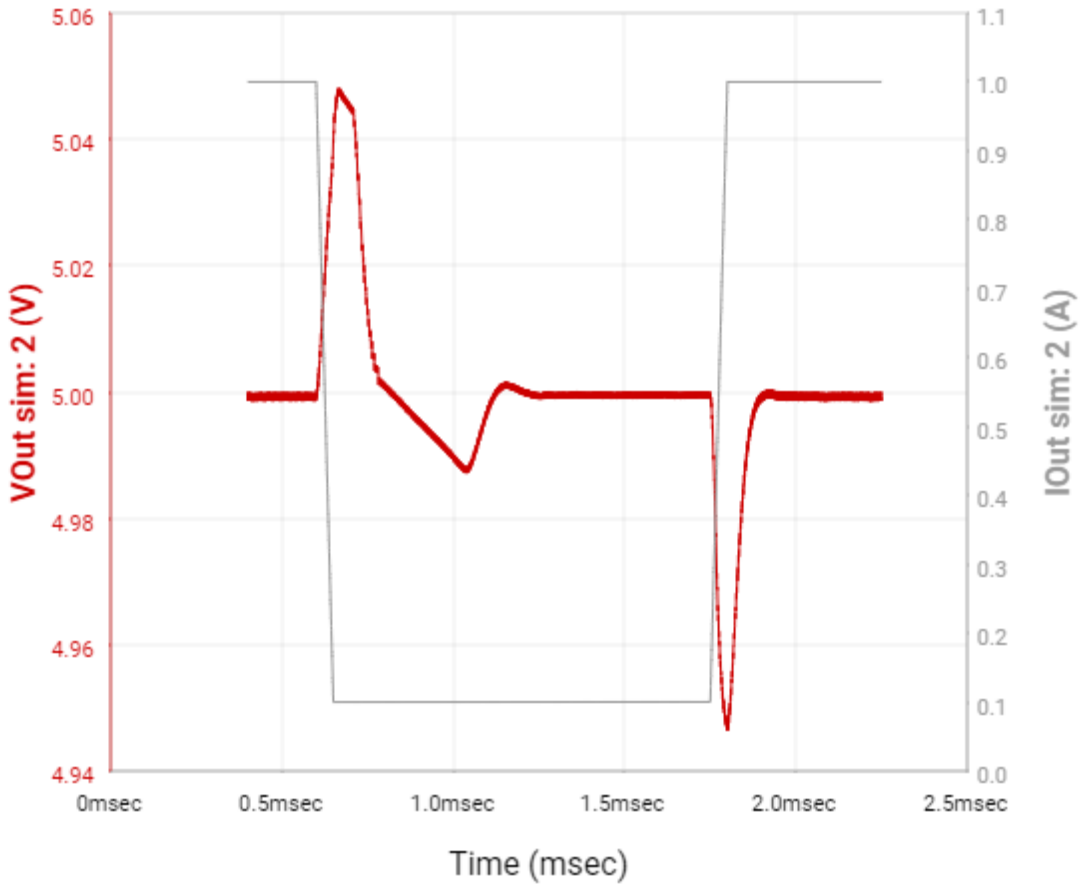



*Figure 24: TI Webench simulation results of the 5V power supply voltage (red) and current (grey) over ~2ms.*

From *Figure 24*, it can be seen that overshoot is less than 1%, and undershoot is less than or equal to 1% when the load current change, $\Delta I = 0.9A$.

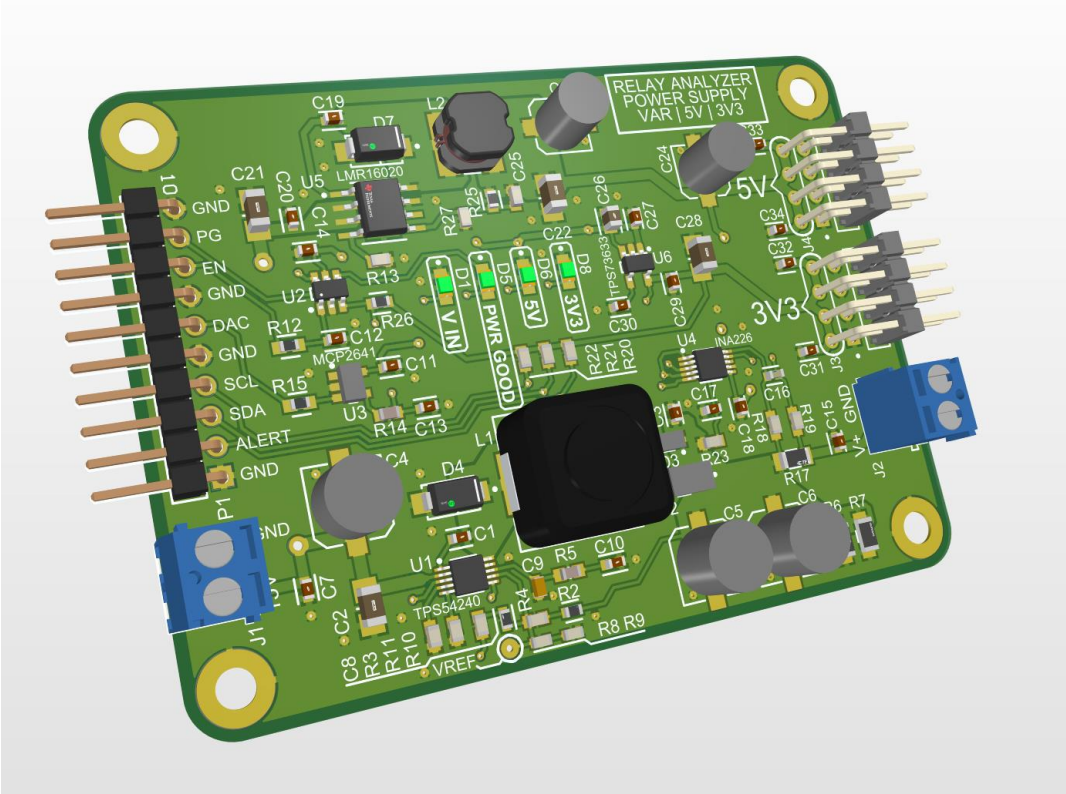



*Figure 25: Programmable, 5V, 3.3V Power Supply Altium render.*

# Coil Resistance Circuit

Purpose:

To measure the resistance of the relay coil in the on-state.

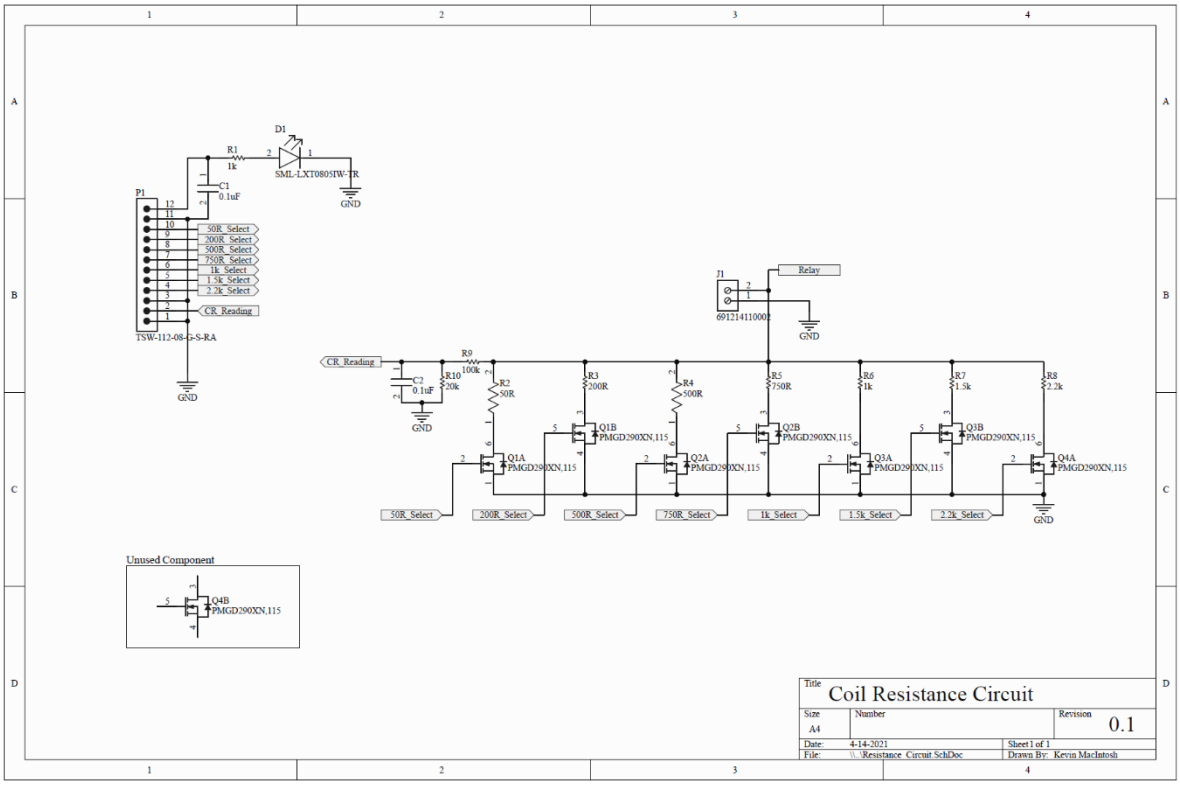



*Figure 26: Coil Resistance Circuit Diagram.*

Simulation:

A coil resistance of 1kΩ was used for the simulations.

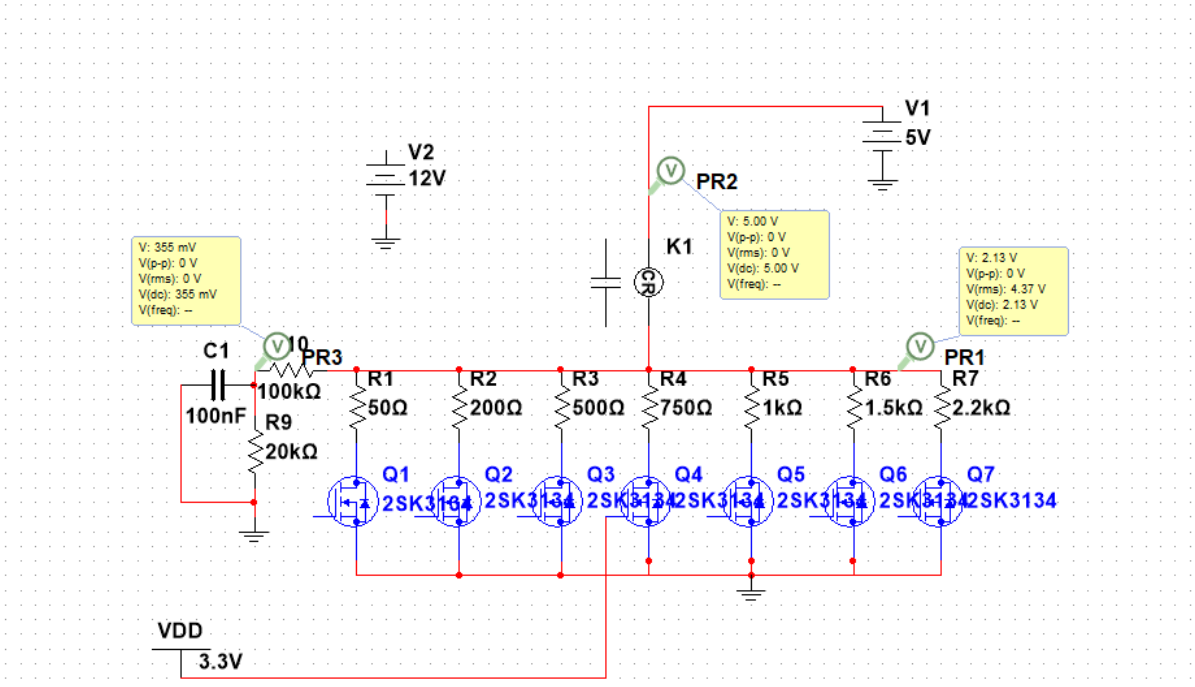



*Figure 27: Coil Resistance Multisim simulation checking reference resistance below coil resistance.*

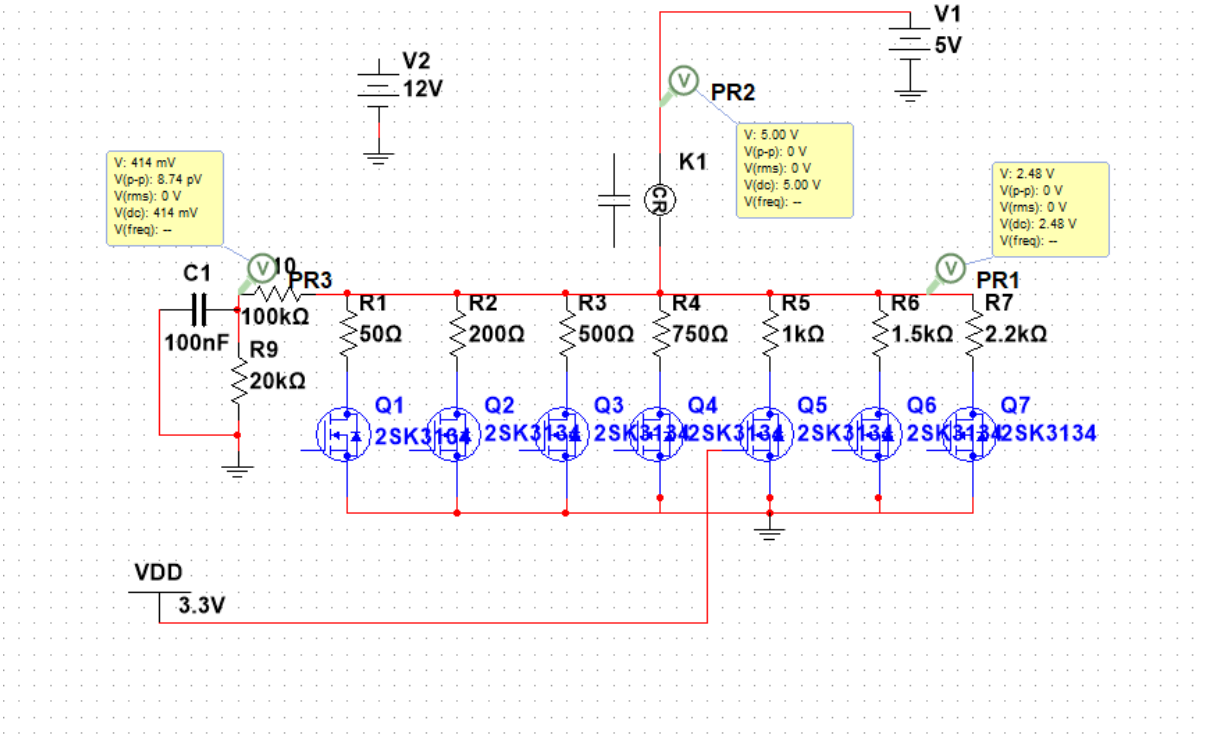



*Figure 28: Coil Resistance Multisim simulation checking reference resistance equal to coil resistance.*

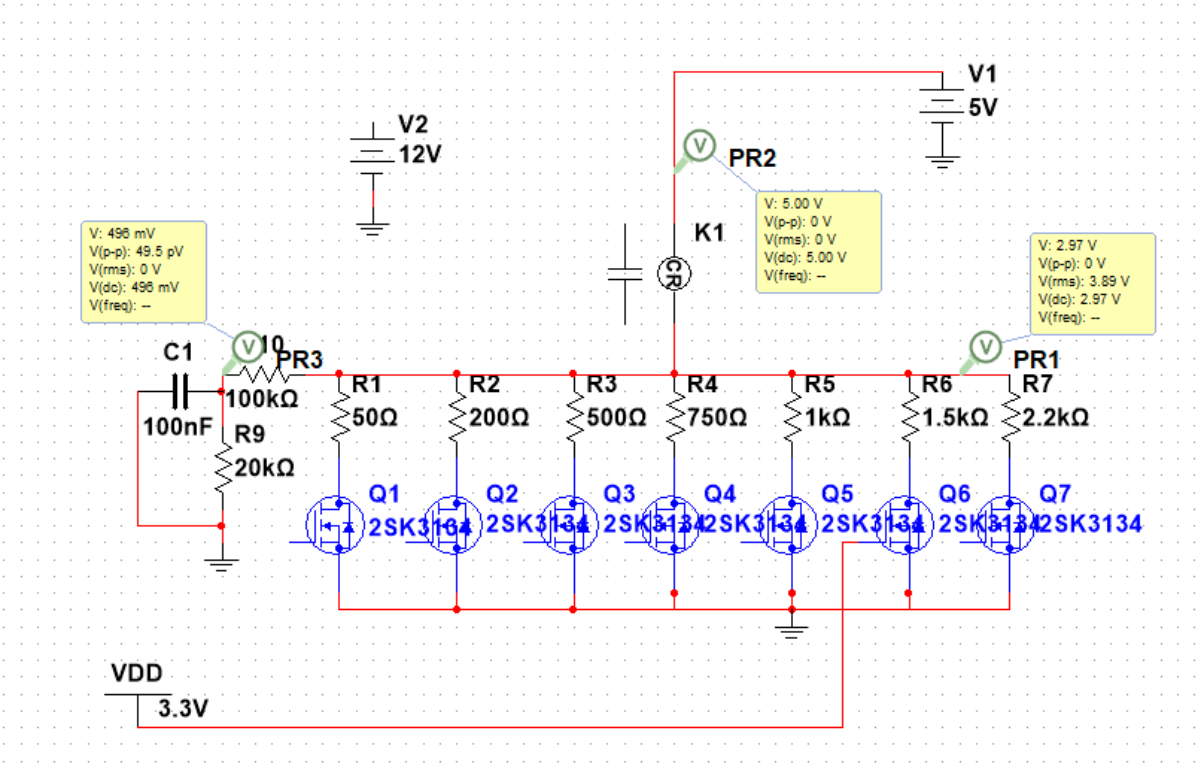



*Figure 29: Coil Resistance Multisim simulation checking reference resistance above coil resistance.*

Major Component:

- PMGD290XN Dual N-channel MOSFET (x4)

Principle of Operation:

The coil resistance circuit uses several reference resistors to create a voltage divider with the relay coil. The relay is activated using the programmable power supply. When in the voltage divider configuration, the output voltage is read by the microcontroller ADC. Because the programmable power supply can output up to 12V, a second voltage divider is used to reduce the highest possible voltage on the ADC input to a maximum of 5V. All resistor values are known, and through firmware, the relay coil resistance can therefore be calculated.

There are seven different reference resistors. The seven resistor values were determined by looking at over 300 different relays, from 5V to 12V, and picking out the most common coil values. The microcontroller switches in a reference resistor one at a time, using the N-Channel MOSFET, and checking for the resistor that provides a voltage reading closest to half the programmable supply voltage. This means that the reference resistor is close to the same value as the coil, thus providing the ADC with a voltage within its voltage range. The further the reference resistor value is from the coil resistance, the further the voltage is from half the programmable supply voltage. The coil resistance is calculated by:

$$R_C = \frac{V_{out}}{V_S - (V_{out} \times N)} \times \frac{1}{R_{Ref}}$$

Where N is the number of ADC bits equivalent to the output voltage. The calculation will be adjusted for error after prototype testing.

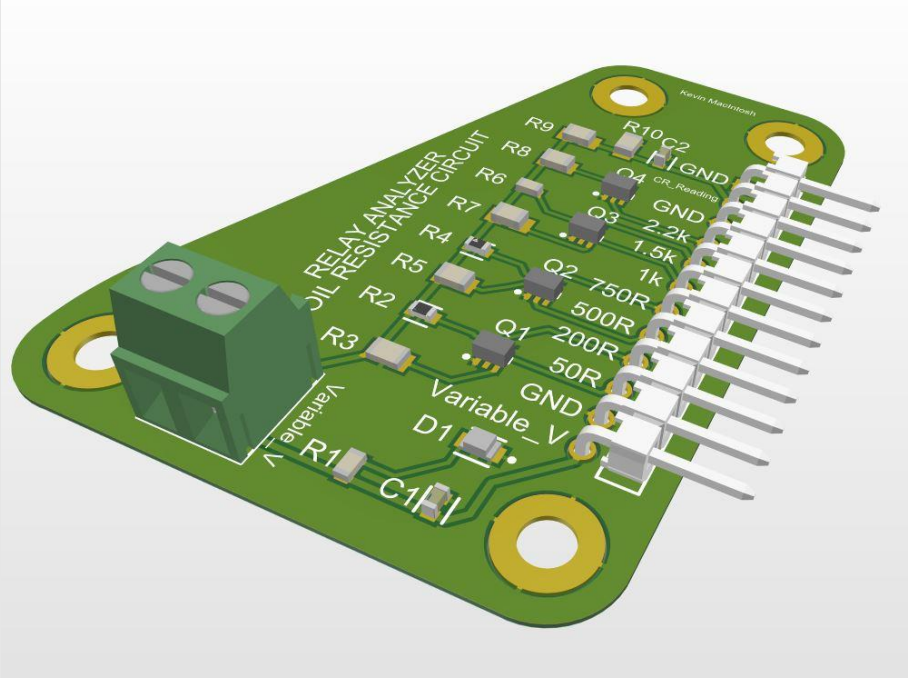



*Figure 30: Coil Resistance Circuit Altium render.*

## Software Design

The software GUI was designed during the software development phase - after the hardware prototyping and testing. The design was done in Qt designer, and therefore follows the MVP design pattern.

There are two main screens in the GUI. The first is the Settings screen. This allows the user to input test parameters - test period, number of cycles, etc. A button will send all settings to the STM32 at once.

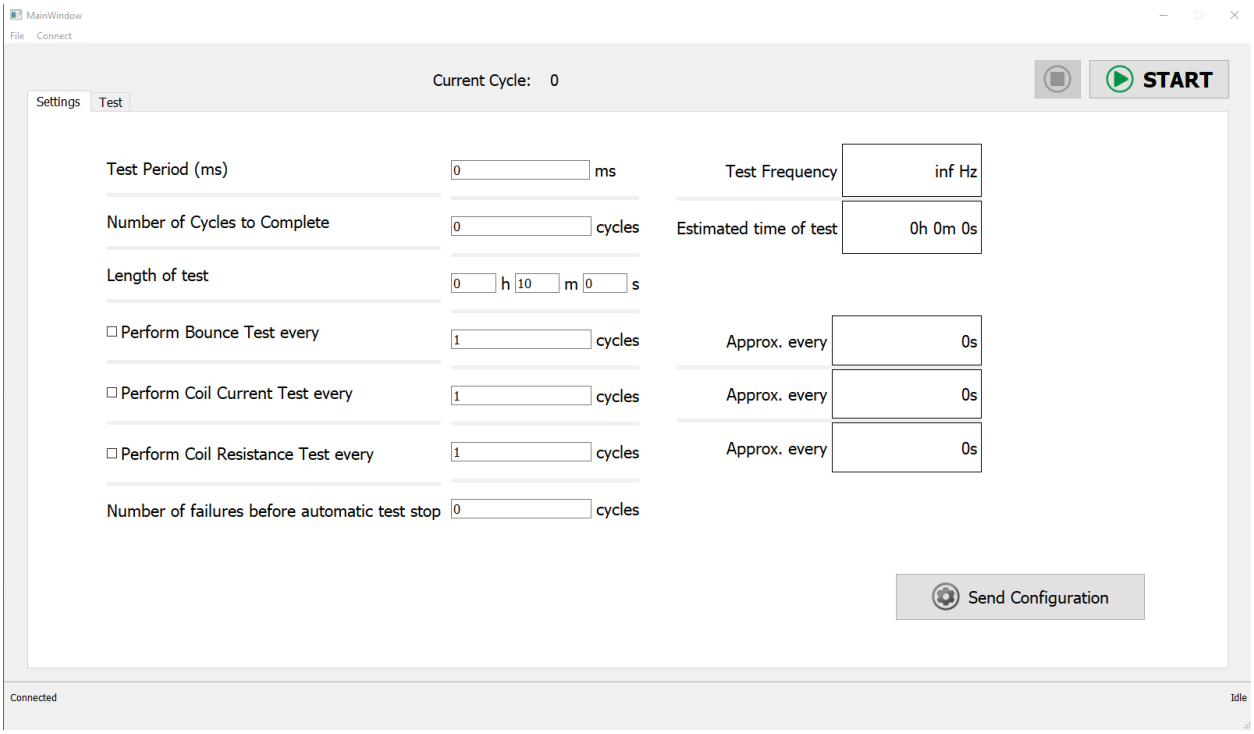



*Figure 31: Settings screen GUI design*

The second screen will show test results. Coil voltage, coil current, coil power, activation and deactivation times will show numbered results. Bounce data will be displayed in a graph, so the user can see the waveform. Results are logged in a database. Graphs of result history can be displayed so that changes in characteristics can be more easily seen. To start a test, there is a start button. A stop button will stop the test.

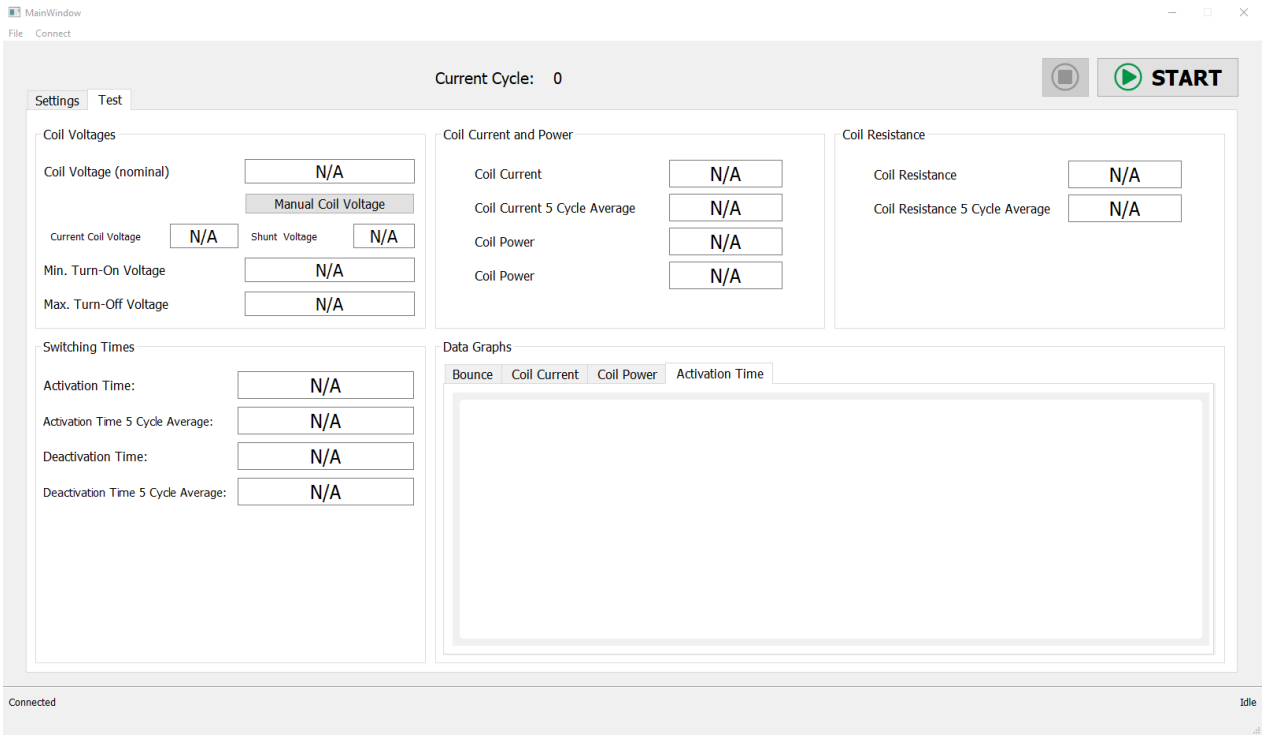



*Figure 32: Test screen GUI design*

Upon start of the program, the software automatically detects if the Relay Analyzer is connected. The connection status is displayed at the bottom.

# USB

*The USB part of the system was developed in the Advanced Elective course. However, it is a critical part of the project, so the relevant information is repeated here for completeness.*

There are three parts to USB that can be broken down into smaller parts. First, the microcontroller. The USB device firmware consists of several parts itself. These include the device descriptors - device, configuration, interface and endpoint - handling of control endpoint messages and commands, and transfer of data packets.

The USB driver is a third-party solution. This is not unlike what would be done in a project for a real product, as there are many tools that streamline development of drivers, such as the Windows Driver Kit (WDK). The WDK, or any other tool, was not used due to the compounded problem of time constraints and cost associated with Windows signed drivers.

Lastly, the software that communicates with the USB device. Virtually all software for USB uses a pre-built USB library, such as libusb-win32, or WinUSB.

## Driver

In order for the PC Operating System to know what kind of USB device has been connected, and what it's capabilities are, a driver is needed. Due to how Windows works, and it's security features, developing drivers can be a long, complicated task. Therefore, for this project, we used a USB driver filter, along with a generic USB driver.

The generic USB driver and USB filter work together to circumvent Windows' automatic take-over of USB devices, and its automatic assigning of generic USB drivers. These two OS processes will lock a USB device into an unusable error state if the OS can't find a suitable, signed, driver.

In order to develop applications for a USB device on Windows, a USB filter and generic USB driver is employed.

The USB filter filters USB devices by VID and PID, and prevents Windows from taking control of the device. This allows the filter to freely associate the USB device with the generic USB driver. After the USB device is associated with the generic driver, any application is free to open and claim the USB interface.

Normally, the driver would include the device, interface and endpoint descriptors. However, because of the generic USB driver, the OS gets this information from querying the USB device using the control endpoint. The full descriptor for the Relay Analyzer can be found in the "STM" section of Project Development.

## USB Library (Software)

Once the USB device has been filtered, and associated with the generic USB driver, applications are free to open and claim the USB interface. Our application - a Graphical User Interface (GUI) for the Relay Analyzer - uses a USB library for Windows.

This USB library is libusb-win32. It is a wrapper for Windows' lower-level USB functions. Through this library, the application:

1. Opens a USB device. In our case, we open a specific USB device based on the VID and PID of the device.
2. Claims an interface. USB devices can have more than one interface, and therefore can be claimed by more than one application at a time. However, the Relay Analyzer only has a single interface. Typically, the OS kernel claims all interfaces when the device is first connected, and the interface must be released by the OS before an application can claim it. In our case, because of the USB filter, the OS does not claim the interface first, and therefore it does not need to be released.
3. Manipulate endpoints. Through the USB interface descriptor, the OS, and therefore the application already knows how many endpoints there are, what type they are, and what

direction they are. With this information, an application can send and receive data through the endpoints.

Below is the application code for the above:

Opening a USB device:

```
usb_init(); /* initialize the library */

usb_find_busses(); /* find all busses */

usb_find_devices(); /* find all connected devices */

handle = usb_open(0x0843, 0x002A);  /* open USB device with VID, PID */
```

Claiming an interface:

```
usb_claim_interface(handle, 1); /* claim interface 1 */
```

Sending and receiving through bulk endpoints:

```
/* read data from a bulk endpoint */
/* usb_bulk_read(USB_Handle, uint8_t address, int bytes, int timeout); */
usb_bulk_read(handle, 0x81, data, 4, USB_TIMEOUT);

/* write data to bulk endpoint */
/* usb_bulk_write(USB_Handle, uint8_t address, int bytes, int timeout); */
usb_bulk_write(handle, 0x01, data, 4, USB_TIMEOUT);
```

## Data format

In order for the PC to control the Relay Analyzer, a set of one byte commands have been created. Additionally, a set of fixed-length data formats have been designed to be sent and received by the PC. The most recent versions of the data formats are outlined below. Note that these may not be the final versions, as they may change due to ongoing development of the Relay Analyzer capstone project.

Get update (0xF1)

- Received update format (from STM32):
  - Timestamp (4 bytes)
  - Current cycle (4 bytes)
  - Activation time in timer counts (4 bytes)
  - Deactivation time in timer counts (4 bytes)
  - INA226 raw bus voltage value (2 bytes)
  - INA226 raw shunt voltage value (2 bytes)
  - INA226 raw bus current value (2 bytes)
  - INA226 raw bus power value (2 bytes)
  - Coil resistance value (2 bytes)
  - **Total: 26 bits + struct**[1]

Set config (0xF2)

- Transmitted configuration format (to STM32):
  - Coil voltage (4 bytes)
  - Period of test (4 bytes)
  - Maximum number of cycles (4 bytes)
  - Timestamp (4 bytes)
  - Period of bounce test in cycles (2 bytes)
  - Period of current test in cycles (2 bytes)
  - Period of resistance test in cycles (2 bytes)
  - Number of failures (2 bytes)
  - **Total: 24 bytes + struct**

Start test (0xF5)

Stop test (0xF7)

Timestamp (0xF9)

- Received timestamp format:
  - Hours (1 byte), minutes (1 byte), seconds (1 byte), subseconds (1 byte)
  - Subseconds must be converted from 0-255 to proper time scale before being displayed
  - **Total: 4 bytes + struct**

Check connection (0xFE)

- Requests a status OK (1byte) from STM. If there is no response, there is a connection issue, if there is a NOT OK response (1 byte), there is an STM error
- Total: 1 byte

STM32_OK (0xAA)

STM_NOT_OK (0xBB)

[1] C compilers may add padding to structs, so their size is not known until build time

# System Development, Prototyping and Testing

The following are the results of the test plans outlined in the previous section, for each functional unit. *Note that some test plans have been updated. All updates are described in this section.*

## General Test Plan

Before each PCB is tested according to the test plans outlined above, some general testing will be performed. This will ensure each PCB has no defects, no shorted power pins, etc. The general testing step will be:

- Visual inspection of PCB
- Multimeter continuity test on power rails and sensitive pins
- Application of current-limited power to supply rails on first power-up
- Observation of designed visual debugging tools, such as LEDs
- Application of full-current power
- Scope of known signals & supply rails

After these steps are performed, the test plan for each circuit will be performed.

### Results

| Test | Bounce | Trigger | Interface | Power Supply | Coil Resistance |
|---|---|---|---|---|---|
| Visual Inspection | Pass | Pass | Pass | Pass | Pass |
| Continuity Test | Pass | Pass | Pass | Pass | Pass |
| First Power-Up | Fail | Fail | Pass | Pass | Pass |
| Debug Tools | Fail | Fail | Pass | Pass | Pass |
| Full Power | N/A | Fail | Pass | Fail | Pass |
| Scope Signals | N/A | Pass | Pass | Pass | Pass |

*Table 3: First pass of general test plan*

Following the first pass of the general test plan, the PCBs that experienced a *fail* were diagnosed, and either repaired, or had design changes made, and PCBs remade.

# Bounce Circuit

## Test Plan

A power supply will provide 3.3V to the bounce circuit. To test the waveform accuracy of the circuit, a function generator will be connected to the terminal block in place of a relay. It will run a signal that alternates between 0 and 5V, at 20KHz [1]. An oscilloscope connected to $V_{out}$ will measure the output signal.

Test Equipment:

- Function generator
- Oscilloscope
- Power supply

Signals:

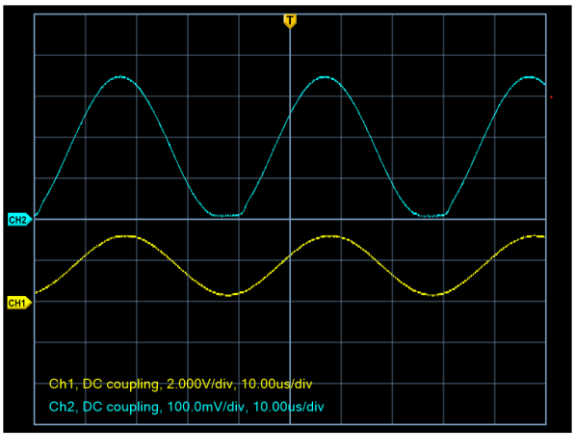
- Terminal block: Function Generator
- $V_{out}$: Oscilloscope probe
- 3V3 plane: 3.3V Power supply

Pass Condition:

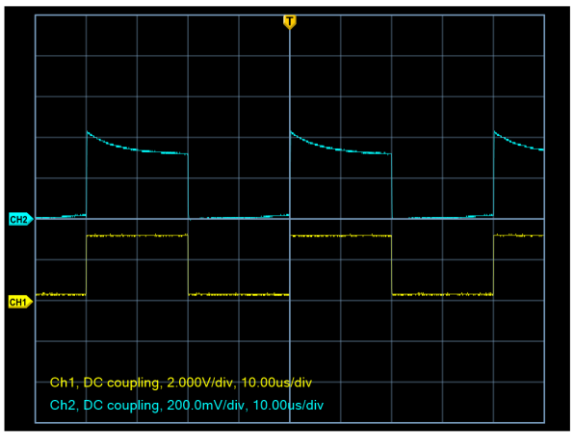
The output waveform must have timing on edges accurate to 2µs. This is based off the sampling rate of the microcontroller being 2.4 million samples per second.

## Results

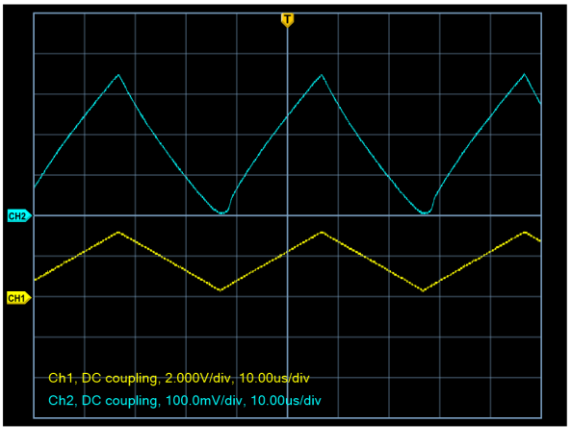
In the following four oscilloscope captures, a 25KHz signal from a function generator was fed into the bounce circuit input pin. The input and output signals were connected to an oscilloscope. The yellow trace is the input, and the blue trace is the output. Note the different vertical scales.



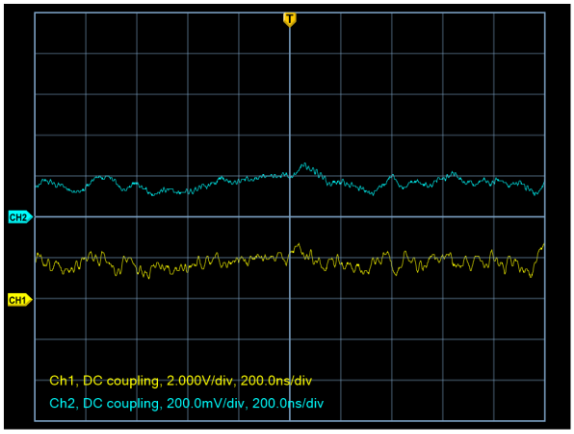

Input (Yellow), Output (Blue)




Input (Yellow), Output (Blue)




Input (Yellow), Output (Blue)




Input (Yellow), Output (Blue)

The input to the MCP6S21 Programmable Gain Amplifier has a voltage divider. Therefore, the output voltage is less than the input. This was to protect the PGA from accidental wrong connection of the relay, in which 12V could be applied to the input.
All output signals exhibit some distortion, specifically when the waveform is near 0V. Square waves appear with overshoot, and noise has noticeably less high-frequency components. All signals exhibit around 20µs delay.

# Trigger Circuit

## Test Plan

To test the trigger circuit, a function generator will be connected to the TRIG IN pin. A 5V relay will be connected to the coil and contacts connections, and the relay will be powered by 5V. The contacts will be wired in Normally Open (NO) configuration. The circuit will be powered by

5V, and a second power supply will provide 2.5V on the DAC IN pin. An oscilloscope will be connected to the TRIG IN and TRIG OUT pins.
The input signal will be a square wave, between 50mHz and 2Hz, 0 to 3.3V, and 50% duty cycle.

Test Equipment:

- Function generator
- Oscilloscope
- (2) Power supplies
- Multimeter [optional]

Signals:

- 5V: Power supply set to 5V
- TRIG IN: Function generator
- TRIG OUT: Oscilloscope, Multimeter
- DAC IN: Power supply set to 2.5V

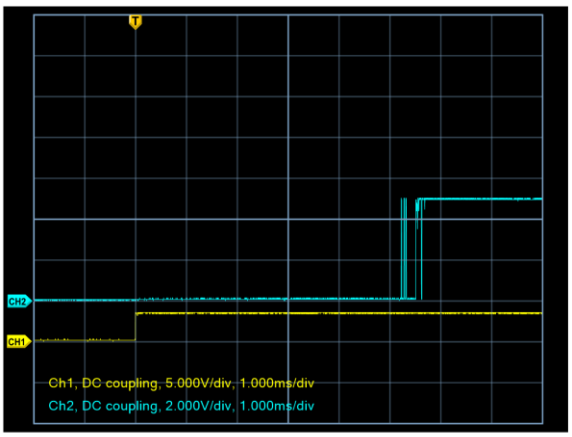
Pass Condition:

The output signal must be in phase with the input trigger signal. The output signal must be within 0 and 5V, with rise and fall times of less than 10ns. The delay between input signal rising edge and output signal rising edge should not be less than 500us, and no more than 30ms.

## Results

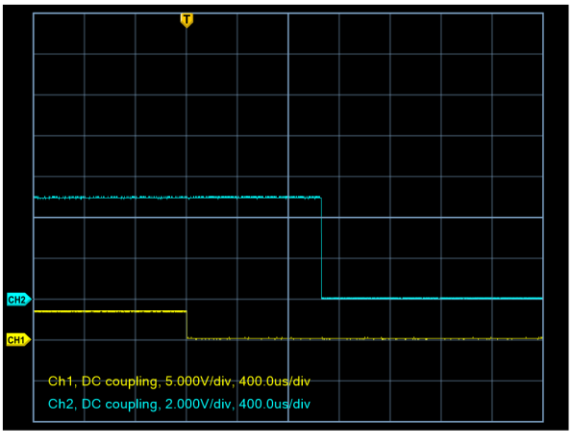
The Trigger PCB initially fails some of the general tests outlined in the General Test Plan. Symptoms included excessively high current draw and excessive heat generated by U3 - a MCP6561. The problem was that the part has two pinouts - one with the VCC and VSS pins reversed compared to the other. The part that was ordered had the reversed power supply pins, but the part used in the schematic did not have the power supply pins reversed. After the part was switched, the Trigger circuit passed all the tests outlined in the General Test Plan.

For each of the following oscilloscope screen captures, the relay under test was connected using the NO (Normally Open) pin. This has the effect of closing the relay when the trigger signal is HIGH, and open when the trigger signal is LOW.



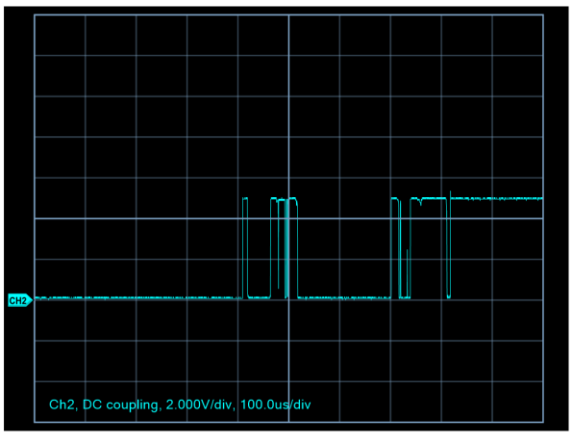

Test Relay - Trigger ON

*Figure 33*



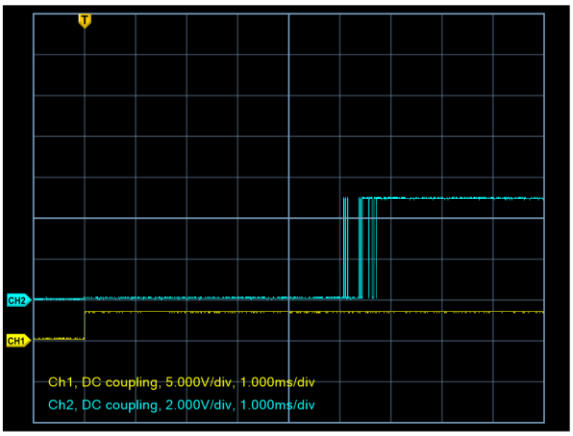

Test Relay - Trigger OFF

*Figure 34*



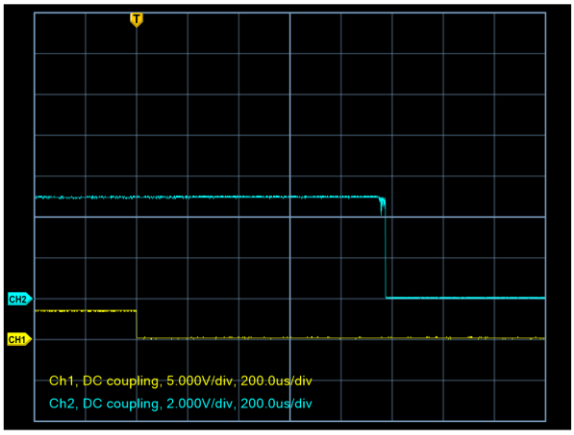

Test Relay - Bounce

*Figure 35*



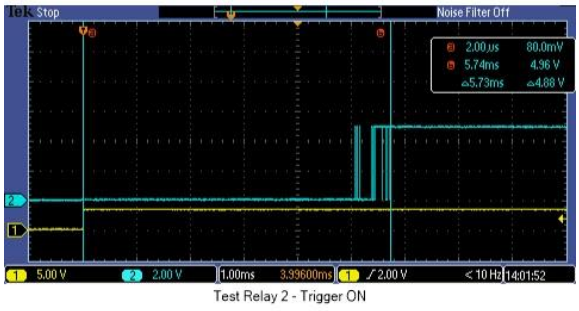

Test Relay 2 - Trigger ON

*Figure 36*




Relay 2 - Trigger OFF

*Figure 37*

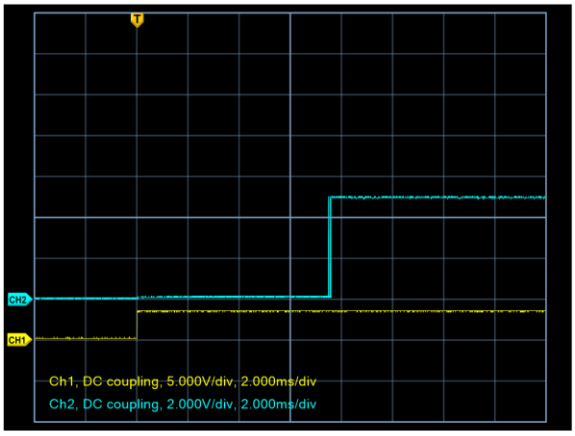



Test Relay 2 - Trigger ON

*Figure 38*

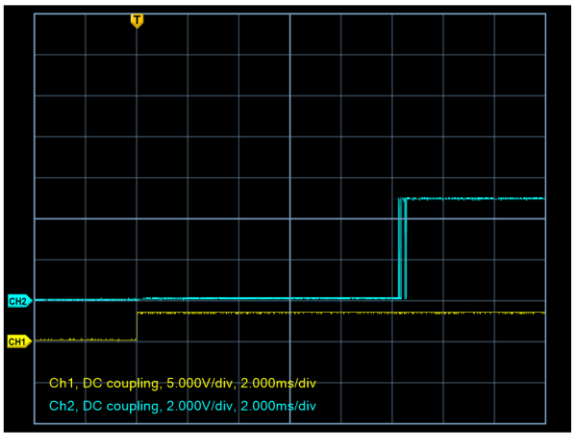

Test Relay 2 - Trigger OFF

*Figure 39*



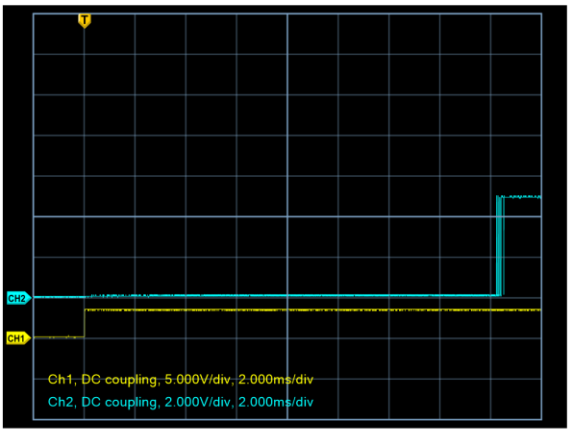

Test Relay 3 - Trigger ON 12V

*Figure 40*




Test Relay 3 - Trigger ON 10V

*Figure 41*




Test Relay 3 - Trigger ON 8.5V

*Figure 42*

Figures 33 through 42 show oscilloscope screen captures of the trigger circuit activating one of three relays under test. Relay 1 is a 5V, 70mA coil current SPDT relay. Relay 2 is a 12V, 100mA coil current relay. Lastly, relay 3 is a 12V, 150mA coil current automotive relay.

Figure *33* and *Figure 34* show that there is a significant delay between the trigger input signal (yellow trace) and the contacts switching (blue trace). Activation (*Figure 33*) has more bounce than deactivation. *Figure 35* shows a close-up of this relay's switch bounce, which lasts over 450us.

*Figure 36* and *Figure 37* show the second test relay. The bounce characteristics are similar to the first. *Figure 38* and *Figure 39* show the oscilloscope cursors, which measure the time from the input trigger signal edge, to the last edge of the relay contact signal. In this case, activation took 5.73ms, and deactivation took 371ms, in *Figure 38* and *Figure 39*, respectively.

For the next relay, only the activation was recorded in oscilloscope captures. However, the coil voltage was varied in order to see the effect on the activation times. *Figure 40* is the baseline, and shows the activation at 12V: between 7 and 8ms. *Figure 41* shows the relay activation at 10V: between 10 and 11ms. Lastly, *Figure 42* shows the relay activation at 8.5V. This was the lowest voltage that the relay would reliably operate at. It took between 16 and 17ms to activate.

These figures validate that the trigger circuit passed the test plan.

# Interface Circuit

## Test Plan

To test the interface circuit, a power supply will be connected to each test circuit input (trigger, bounce, coil resistance, etc) in turn. The circuit will be powered by 5V. Each SSR will be deactivated and activated using 5V and ground. When in each state, the resistance between the two switched terminals will be measured using a DMM, and the values recorded*.

*The original test used an LED to indicate when the SSR was in the activated state. This was deemed to not be a sufficient test, as the MOSFETs in the SSR have ON-state resistance, which could have a significant effect on the operation of the circuit, and connected circuits.

Test Equipment:

- Power supply
- Multimeter

Signals:

- 5V: Power supply set to 5V
- RES, TRIG, PWR, STAT, BNCE: 5V
- COIL+, CONTACTS+: 5V
- COIL-, CONTACTS-: LED

Pass Condition:

When the corresponding SSR trigger pin is activated, the resistance between the two terminals of the SSR should have less than 600mΩ in both directions (The datasheet specifies a typical on-state resistance of 350mΩ). When deactivated, there should be more than 1MΩ of resistance between the terminals.

### Results

| SSR | Off Resistance | Resistance (Forward) | Resistance (Reverse) |
|---|---|---|---|
| Power | >1MΩ | 220mΩ | 215mΩ |
| Trigger (Coil) | >10MΩ | 190mΩ | 204mΩ |
| Coil Resistance | >10MΩ | 195mΩ | 207mΩ |
| Bounce | >10MΩ | 230mΩ | 213mΩ |
| Trigger (Contacts) | >10MΩ | 225mΩ | 218mΩ |

*Table 4: Interface PCB test results*

*Table 2* shows the resistance measurements of the different SSRs in activated and deactivated states. The forward and reverse directions are measures, since it is critical to some circuits proper operation. All of the measured values are well within the values specified by the datasheet, and in the case of the on-state resistances, exceed the specified datasheet value.

## Programmable Power Supply

### Test Plan

To test the programmable power supply, the circuit will be powered by a lab bench power supply at 15V. A second power supply – a programmable lab bench power supply capable of 10mV increments – will be connected to the DAC IN pin, and set to 2.5V. An electronic load will be connected to the output, and set at 200mA constant current*. The programmable power supply enable pin (EN) will be activated using a 3V3 signal, and the output voltage will be observed using an oscilloscope.

While the power supply is active, the voltage on the DAC IN pin will be changed from 0-3.3V, in 500mV increments**.

With the same setup as above, and the output connected to an oscilloscope, oscilloscope captures of steady state, load steps, and DAC steps will be recorded***.

*The original test used a 220Ω resistor on the output. This was switched to an electronic load, so as to draw a constant 200mA over all output voltages.

**The original test specified 250mV steps. This was changed to 500mV steps to expedite testing.

***These tests were added, as they are standard power supply tests.

Test Equipment:

- Power supply
- Programmable power supply
- 220Ω load resistor
- Oscilloscope
- Electronic load
- Multimeter [optional]

Signals:

- 15V: Power supply set to 15V
- EN: 3V3
- DAC IN: 0-5V
- V+: Oscilloscope, multimeter, electronic load

Pass Condition:

The output of the programmable power supply will be stable, with less than 100mVp-p ripple, while the DAC IN voltage is constant. Secondly, the output voltage will *increase* as the DAC IN voltage is *decreased*, and vice versa.
In the second test, the output voltage will be stable, with less than 100mVp-p ripple for a constant DAC IN voltage.

## Results

The following results include oscilloscope screen shots, as well as data obtained by multimeter measurements and internal sensors on the PCB.

| DAC Voltage | Output Voltage (No Load) | Output Voltage (200mA) |
|---|---|---|
| 0.20 | 14.06 | 14.13 |
| 0.30 | 13.68 | 13.75 |
| 0.40 | 13.30 | 13.36 |
| 0.50 | 12.92 | 12.98 |
| 0.60 | 12.54 | 12.60 |
| 0.70 | 12.15 | 12.21 |
| 0.80 | 11.77 | 11.82 |
| 0.90 | 11.39 | 11.44 |
| 1.00 | 11.00 | 11.05 |
| 1.10 | 10.62 | 10.67 |
| 1.20 | 10.24 | 10.28 |
| 1.30 | 9.86 | 9.90 |
| 1.40 | 9.47 | 9.51 |
| 1.50 | 9.09 | 9.13 |
| 1.60 | 8.71 | 8.74 |
| 1.70 | 8.32 | 8.36 |
| 1.80 | 7.94 | 7.97 |
| 1.90 | 7.56 | 7.59 |
| 2.00 | 7.18 | 7.20 |
| 2.10 | 6.80 | 6.82 |
| 2.20 | 6.41 | 6.43 |
| 2.30 | 6.03 | 6.05 |
| 2.40 | 5.65 | 5.66 |
| 2.50 | 5.26 | 5.28 |
| 2.60 | 4.88 | 4.89 |
| 2.70 | 4.50 | 4.50 |
| 2.80 | 4.12 | 4.12 |
| 2.90 | 3.73 | 3.73 |
| 3.00 | 3.35 | 3.35 |
| 3.10 | 2.97 | 2.97 |

| | | |
|---|---|---|
| 3.20 | 2.59 | 2.58 |
| 3.30 | 2.20 | 2.20 |

*Table 5: Variable power supply test results*

*Table 1* is a table of DAC input voltages, and the resulting output voltage at no load, and at 200mA output. The circuit performs better than expected - at input DAC voltages below 2.70V, the output voltage is actually higher under load than it is with no load. With DAC input voltages at or above 2.70V, the output voltage is the same at no load and under load.

The data from *Table 1* can be charted, as seen below:

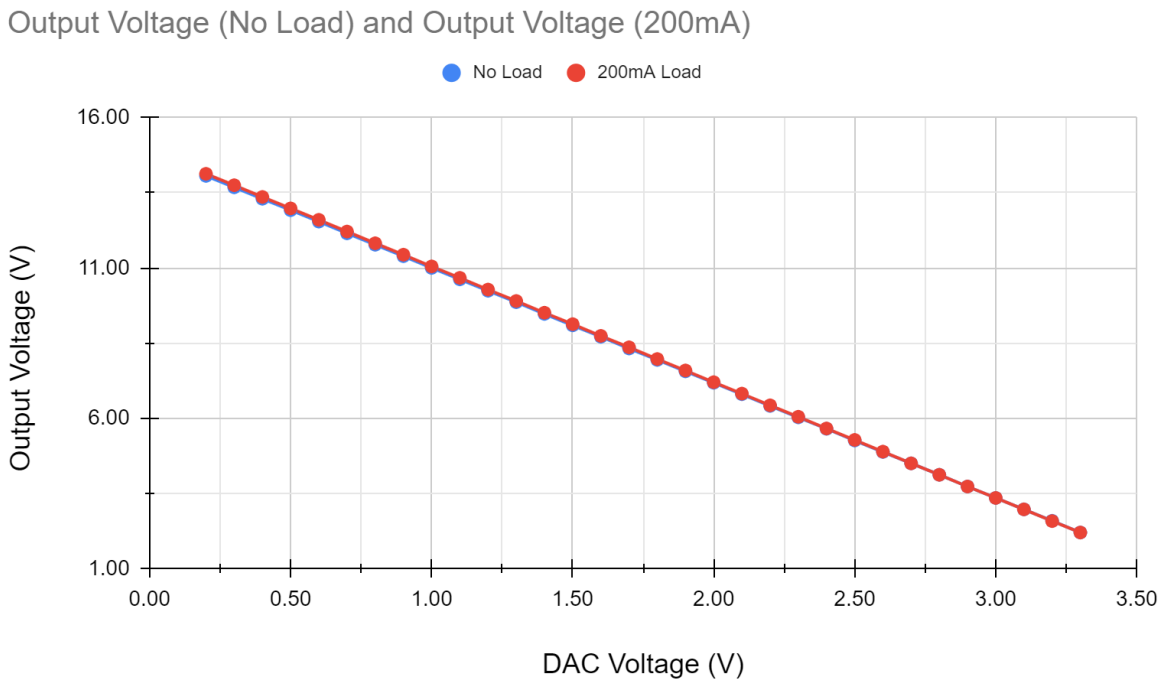
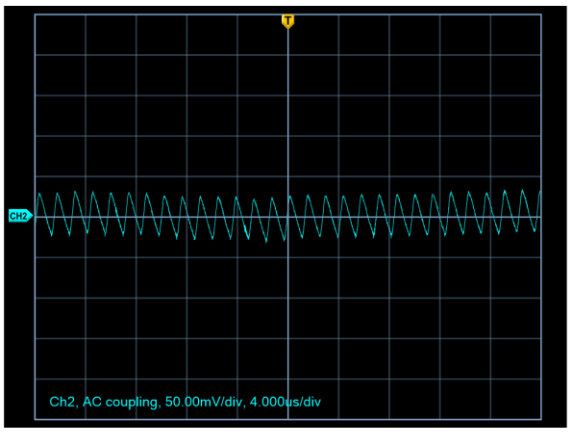


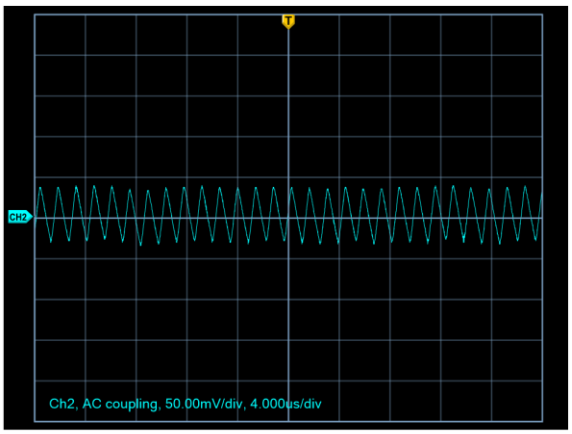

*Figure 43: Variable voltage supply output*

*Graph 1* shows that the relationship between DAC voltage and output voltage is linear.



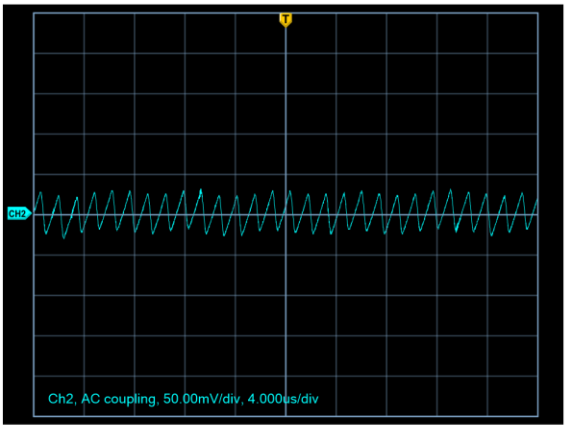

3V3 Output - steady state

*Figure 44*



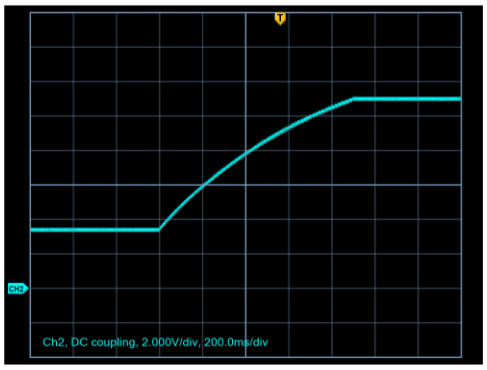

5V Output - steady state

*Figure 45*



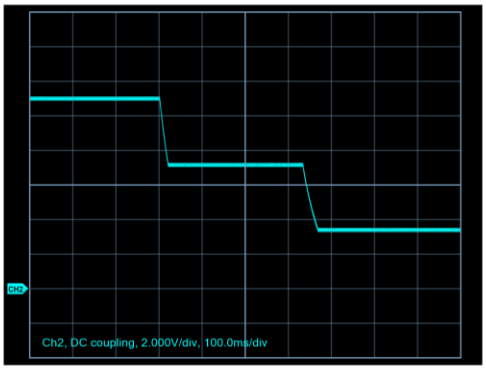

12V output - steady state

*Figure 46*



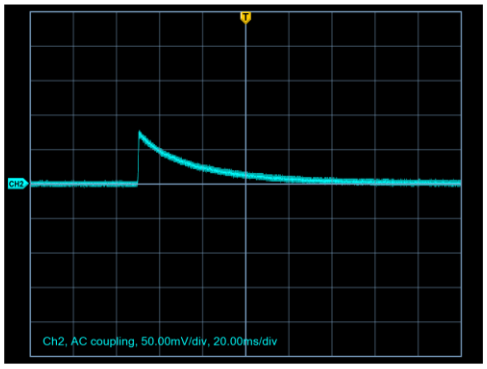

Prog. PS - VDAC: 3V - 1V step

*Figure 47*



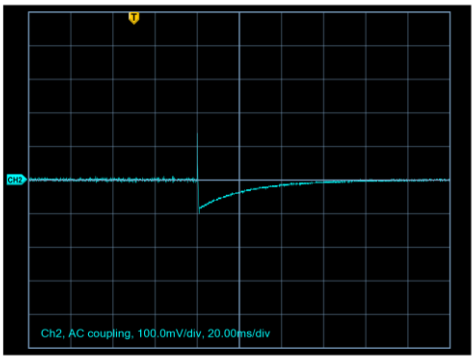

Prog. PS - VDAC: 1V to 2V & 2V to 3V

*Figure 48*




Prog. PS - Load: 0mA to 500mA step

*Figure 49*




Prog. PS - Load: 500mA to 0mA step

*Figure 50*

From *Figure 44, 45 and 46*, the steady state of the variable power supply, at 3V, 5V, and 12V, has ripple voltages of 60mV, 80mV and 60mV, respectively. These are well within acceptable levels, especially when powering relay coils, which do not contain sensitive parts, and therefore are not affected by ripple.

The next two figures - *Figure 47 and 48* - show the response of the output voltage to a step in DAC input voltage. In *Figure 47*, the DAC voltage was changed from 3V to 1V, which results in an *increase* of output voltage. In *Figure 48*, the DAC voltage was changed from 1V to 3V (in two steps, due to a power supply limitation), which results in a *decrease* of output voltage. The time for the output voltage to respond to the change in DAC voltage in *Figure 47 and 48* is 900ms and 100ms, respectively. As the DAC voltage will typically be changed without a load present, the output bulk capacitance results in a longer hold-up time.

The step response of the output voltage to a change in load current can be seen in *Figure 49 and 50*. Both load steps were performed at an output voltage of 5V. *Figure 49* shows a load step from 0mA to 500mA output. There is approx. 75mV of overshoot on the output voltage. *Figure 50* shows a load step from 500mA to 0mA output. There is a very quick transient of approx. 130mV, as well as approx. 100mV of undershoot on the output voltage. These figures correspond to load regulation of 2.5%.

# Programmable Power Supply - Current Monitor

## Test Plan

To test the current monitor, an Arduino microcontroller, with known good firmware for interfacing with the INA226, will be connected to the $I^2C$ bus. The circuit will be powered by 3V3. While testing the variable power supply in the previous test plan, the current and voltage

reported by the INA226 will be recorded. The INA226 was set to an average of 64 samples, with a sample time of 4.2ms*.

*The original test only recorded INA226 reported current and voltage at three different voltages, and three different currents.

Test Equipment:

- Power supply
- Arduino
- Computer
- Oscilloscope
- Electronic load
- Multimeter [optional]

Signals:

- 15V: Power supply set to 15V
- SDA, SCL: Arduino
- EN: 3V3
- DAC IN: 0-5V
- V+: Oscilloscope, multimeter, electronic load

Pass Condition:

The current measurement from the Arduino microcontroller will be observed, and compared to the set output current of the programmable power supply. The current measurement will be within 10% of the current measure by the electronic load. Secondarily, if required, a bench multimeter will be connected in series with the output, in order to verify the output current measurements. The reported voltage will be within 5% of the voltage measured using a multimeter.

## Results

| Output Voltage (200mA) | INA226 Reported Voltage | INA226 Reported Current (mA) |
|---|---|---|
| 14.13 | 14.17 | 204 |
| 13.75 | 13.78 | 204 |
| 13.36 | 13.39 | 204 |
| 12.98 | 13.01 | 204 |
| 12.60 | 12.62 | 204 |
| 12.21 | 12.24 | 204 |
| 11.82 | 11.85 | 204 |
| 11.44 | 11.47 | 204 |
| 11.05 | 11.08 | 204 |
| 10.67 | 10.70 | 204 |
| 10.28 | 10.31 | 204 |
| 9.90 | 9.93 | 204 |
| 9.51 | 9.54 | 204 |
| 9.13 | 9.15 | 204 |
| 8.74 | 8.77 | 204 |
| 8.36 | 8.39 | 204 |
| 7.97 | 8.00 | 204 |
| 7.59 | 7.62 | 204 |
| 7.20 | 7.23 | 204 |
| 6.82 | 6.84 | 204 |
| 6.43 | 6.46 | 204 |
| 6.05 | 6.07 | 204 |
| 5.66 | 5.69 | 204 |
| 5.28 | 5.30 | 204 |
| 4.89 | 4.92 | 204 |
| 4.50 | 4.53 | 204 |
| 4.12 | 4.15 | 204 |
| 3.73 | 3.76 | 204 |

| | | |
|---|---|---|
| 3.35 | 3.38 | 204 |
| 2.97 | 3.00 | 204 |
| 2.58 | 2.61 | 204 |
| 2.20 | 2.22 | 204 |

*Table 6: INA226 test results*

*Table 4* gives the output voltage at 200mA load, and the reported current and voltage from the INA226. The INA226 reports a voltage with less than 1.5% error compared to the multimeter connected to the output. The current error is a constant 1.96%. Both reported voltage and reported current are well within the 5% and 10% pass conditions, respectively.

# Programmable Power Supply - 3.3V and 5V Supplies

## Test Plan

To test the 5V power supply, the circuit will be powered by a lab bench power supply at 15V. The output will be connected to an oscilloscope, as well as a programmable electronic load. The output voltage will be observed while the load current varied by 100mA, from 0mA to 1A*.

To test the 3V3 power supply, the 5V power supply will be powered by 15V, which will power the 3V3 linear regulator. The output will be connected to an oscilloscope, as well as a programmable electronic load. The output voltage will be observed while the load current is varied by 50mA, from 0mA to 200mA*.

*The original test plan had three currents.

Test Equipment:

- Power supply
- Oscilloscope
- Electronic load
- Multimeter [optional]

Signals:

- 15V: Power supply set to 15V
- 5V, 3V3: Oscilloscope, multimeter, electronic load

Pass Condition:

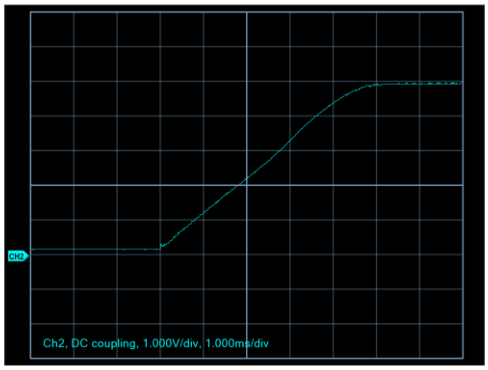
The output of the 5V power supply will be stable, with less than 50mVp-p ripple. The load regulation should be within 5% from 0mA to 1A output current.
The output of the 3V3 power supply will be stable, with less than 15mVp-p ripple. The load regulation should be within 5% from 0mA to 200mA output current.

## Results

| 5V Supply | |
|---|---|
| **Load** | **Output Voltage** |
| 0mA | 5.00 |
| 100mA | 4.98 |
| 200mA | 5.02 |
| 300mA | 5.05 |
| 400mA | 5.06 |
| 500mA | 5.03 |
| 600mA | 5.02 |
| 700mA | 4.98 |
| 800mA | 4.97 |
| 900mA | 4.97 |
| 1A | 4.96 |

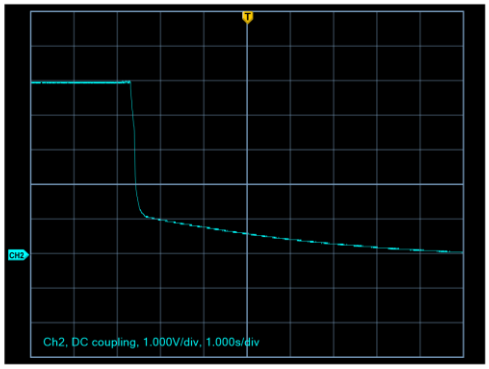
*Table 7: 5V power supply test results*

*Table 5* shows that the output voltage is less than 1.2% from 0mA to 1A output current.



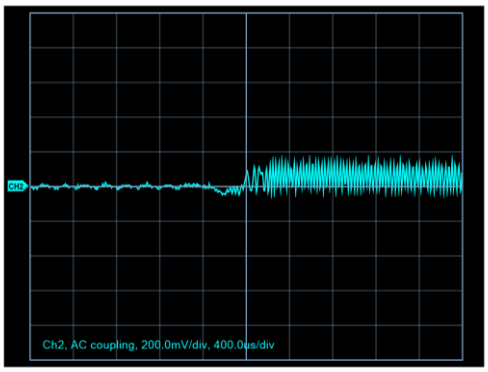

5V Turn On

*Figure 51*



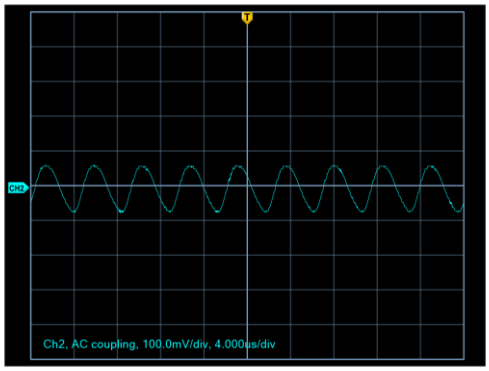

5V Turn Off

*Figure 52*




5V Load Transient - 0mA to 500mA

*Figure 53*




5V Output - 500mA load [140mV pk-pk]

*Figure 54*

*Figure 51* and *Figure 52* show the turn on and turn off characteristics of the 5V supply. Note, this is with no load. *Figure 53* shows the response to a load step. *Figure 54* shows the steady state at 500mA load.

| | |
|---|---|
| **Input Current** | 278mA |
| **Output Current** | 750mA |
| **Input Voltage** | 15.0V |
| **Output Voltage** | 5.03V |
| **Input Power** | 4.17W |
| **Output Power** | 3.69W |
| **Efficiency** | **88.50%** |

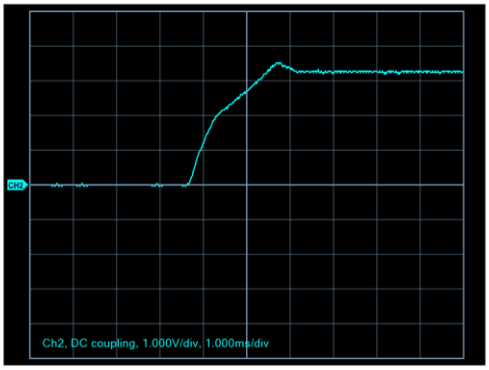
*Table 8: 5V efficiency calculation*

*Table 6* calculates the efficiency of the 5V switching regulator at 75% output current, or 750mA.The calculation results in an efficiency of 88.5%, which is very close to the 89.1% that TI Webench - the tool used to help design the power supply - calculated.

| 3V3 Supply | |
|---|---|
| **Load** | **Output Voltage** |
| 0mA | 3.29 |
| 25mA | 3.28 |
| 50mA | 3.28 |
| 100mA | 3.28 |
| 150mA | 3.28 |
| 200mA | 3.28 |

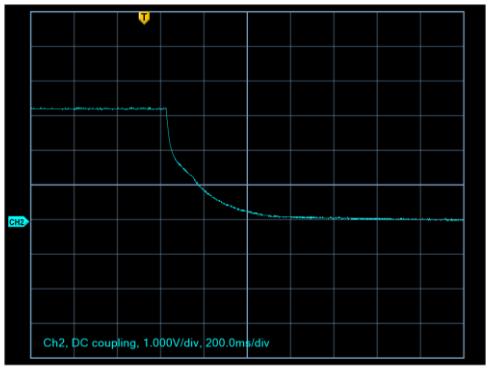
*Table 9: 3V3 power supply test results*

*Table 7* shows that the output voltage has an error of 0.6% or less for all tested output currents.



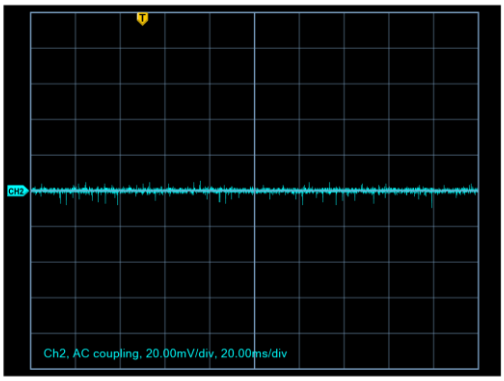

3V3 Turn On

*Figure 55*




3V3 Turn Off

*Figure 56*




3V3 Output - 200mA load

*Figure 57*

*Figure 55* and *Figure 56* show the turn on and turn off characteristics of the 3V3 supply. Note, this is with no load. *Figure 57* shows the steady state at 200mA load.

# Coil Resistance Circuit

## Test Plan

To test the coil resistance circuit, several relays, with known coil resistances, and corresponding to the seven reference resistors, will be connected. The circuit will be powered by 5V and the programmable power supply will activate the relay coil. The output will be connected to an oscilloscope. In turn, each MOSFET will be activated using a 3V3 signal on the gate pin. The output voltage will be observed.

Test Equipment:

- (2) Power supplies
- Oscilloscope

Signals:

- Variable_V: Programmable power supply
- Output: Oscilloscope
- 2.2K, 1.5K, 1K, 750R, 500R, 200R, 50R: 3V3

Pass Condition:

The output will be within 15% of half the Variable_V voltage when the reference resistor value and the coil resistance value are the same (or similar). The 15% condition may change if the coil resistance and the reference resistor values are significantly different. Using the formula above, the coil resistance will be calculated, and compared to the datasheet.

## Results

Using the equation below, the measured value of the reference resistor (R2), the voltage at the output of the voltage divider (Vmid), the source voltage (Vsource), the value of R1 (the resistance of the relay coil) was calculated.

$$R1 = \frac{(R2 \times Vsource)}{(Vmid - R2)}$$

| R2 (Reference) | Vmid | Vsource | R1 (measured) | **R1 (calculated)** | Error |
|---|---|---|---|---|---|
| 998 | 2.46 | 4.99 | 992 | **1026** | 3.3% |
| 1.486K | 2.48 | 4.99 | 1.49K | **1.504K** | 0.9% |
| 2.18K | 2.48 | 4.99 | 2.17K | **2.20K** | 1.3% |
| 749 | 2.49 | 4.99 | 746 | **752** | 0.8% |
| 499 | 2.49 | 4.99 | 495 | **501** | 1.2% |
| 199 | 2.49 | 4.99 | 197.5 | **199.8** | 1.2% |
| 50 | 2.46 | 4.99 | 49.7 | **51.4** | 3.3% |
| 499 | 2.99 | 4.99 | 329 | **333** | 1.2% |
| 499 | 5.90 | 9.97 | 329 | **344** | 4.4% |
| 197.5 | 1.84 | 4.99 | 329 | **338** | 2.6% |
| 1.5K | 4.53 | 9.97 | 1.77K | **1.8K** | 1.7% |
| 499 | 2.12 | 4.99 | 670 | **675** | 0.7% |

*Table 10: Coil resistance PCB test results*

*Table 6* is a table of tests for the coil resistance circuit. 10 different test resistances (R1) were connected to the coil resistance circuit, in the same way that a coil of a relay would be. Using the equation above, as well as several measurements, a value was calculated for R1. The error - calculated value vs measured value - was also calculated. Each reference resistor was tested against a test resistance that closely matched R1. Several test resistances were tested against reference resistors that didn't closely match R1. One R1 resistor - 329Ω was tested at different source voltages.

Only a single relay has been tested with the coil resistance circuit. While the test results are good, this does not validate the coil resistance circuit. The results from the single relay are below:

| R2 (Ref) | Vmid | Vsource | R1 (measured) | **R1 (calc)** | Error |
|---|---|---|---|---|---|
| 50 | 2.75 | 6.81 | 75.9 | **73.76** | 2.9% |

*Table 11: Coil resistance PCB relay test result*

*After further testing, the reference resistor values were deemed to be too high. For any relay with a coil voltage less than 12V, none of the reference resistors allowed the coil to activate. Therefore, all the reference resistors have been replaced with values between 0.5Ω and 50Ω. Testing of the coil resistance circuit with new reference resistors has yet to be completed.*

***Due to time constraints, the coil resistance PCB, despite working, was never extensively tested. The limited testing completed shows that there was too great an error compared to the results from the INA226 current monitor, and therefore, the coil resistance PCB was not used in the final system.***

## Hardware Interconnection

Now that all individual PCBs have been tested, the next tests involve multiple PCBs connected and working together. The microcontroller development board is the central part of the Relay Analyzer, and is therefore the logical starting point when testing multiple PCBs together. The following diagram shows how all the PCBs are connected together, including the type of connections (GPIO, SPI, $I^2C$, or analog).

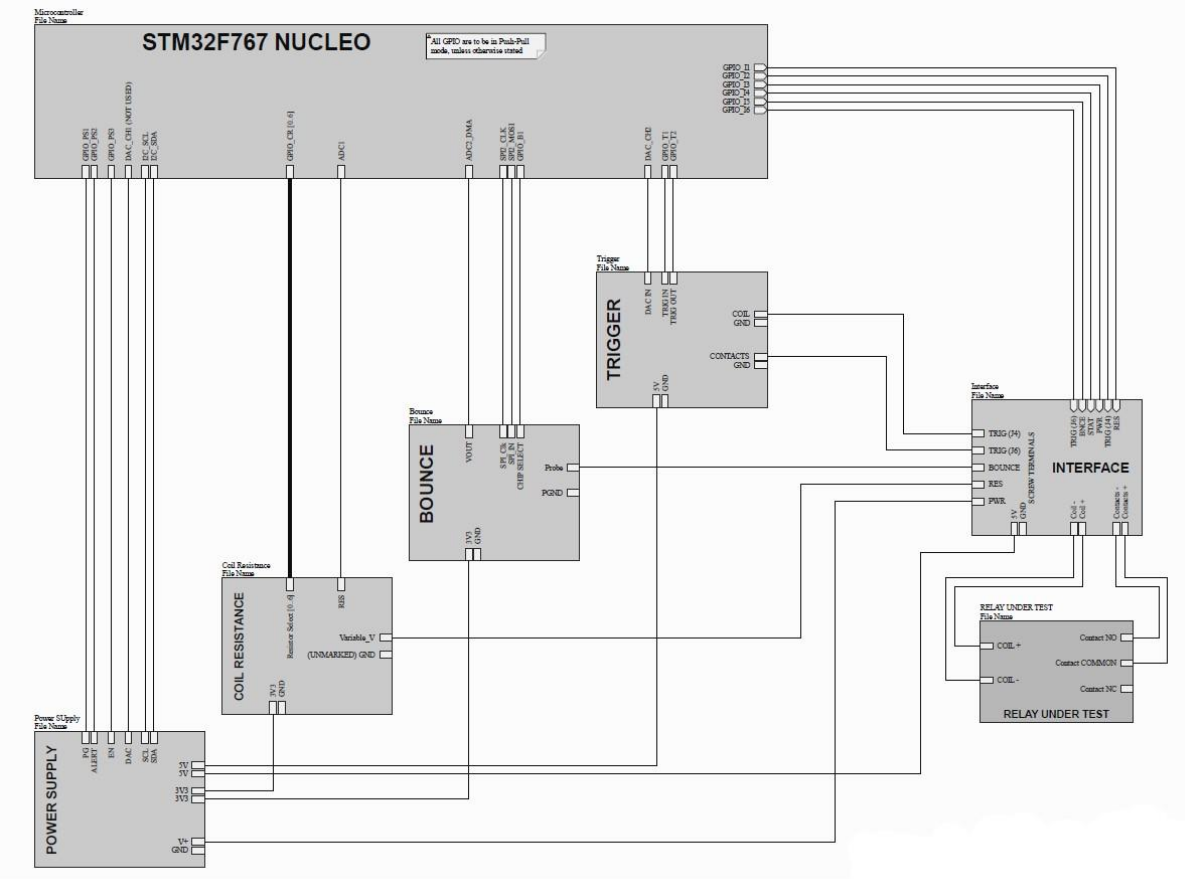



*Figure 58: System PCB connection diagram*

# Microcontroller & Power Supply

## Test Plan

Using GPIO and DAC, the microcontroller will first enable the variable power supply, and then set it to each of the following voltages: 3V, 3.3V, 4.5V, 5V, 6V, 9V, and 12V.

The microcontroller will also use $I^2C$ to interface with the INA226 on the power supply. It will write to the configuration register and the calibration register, and then read the Bus Voltage, Shunt Voltage, Bus Current, and Bus Power registers. Using simple conversion formulas, the values read from the registers should be close to those read on various lab equipment.

The following table shows the calculated DAC codes for each output voltage.

| Output Voltage | DAC Voltage | DAC Code (calculated) |
|---|---|---|

| | | |
|---|---|---|
| 3.00 | 3.093 | **3842** |
| 3.30 | 3.014 | **3744** |
| 4.50 | 2.700 | **3354** |
| 5.00 | 2.569 | **3191** |
| 6.00 | 2.308 | **2867** |
| 9.00 | 1.524 | **1893** |
| 12.00 | 0.740 | **919** |

*Table 12: Calculated output voltage DAC codes*

The formula for the code is:

$$LSB\ Voltage\ = \frac{Vref}{4096} = \frac{3.3}{4096} = 0.805mV$$

$$DAC\ Code\ = \frac{DAC\ Voltage}{LSB\ Voltage}$$

Pass Conditions:

The Microcontroller must be able to enable the variable power supply, and command it using a DAC, to predetermined voltages.

The microcontroller must also be able to write to, and read from, the INA226 current monitor on the power supply, and get accurate values for shunt voltage, bus voltage, bus current, and bus power.

## Results

The microcontroller was successfully able to enable and set the output voltage of the variable power supply. Using the codes calculated above, the output voltage was slightly offset, and therefore the following DAC codes were found experimentally:

| Output Voltage | DAC Voltage | DAC Code (experimental) |
|---|---|---|
| 3.00 | 3.093 | **3805** |
| 3.30 | 3.014 | **3709** |

| | | |
|---|---|---|
| 4.50 | 2.700 | **3321** |
| 5.00 | 2.569 | **3160** |
| 6.00 | 2.308 | **2837** |
| 9.00 | 1.524 | **1870** |
| 12.00 | 0.740 | **902** |

*Table 13: Experimental output voltage DAC codes*

The following table shows the output of the INA226 current monitor, the calculated value, and the actual value, as measured using lab equipment.

| Metric | Returned uint16_t | Calculated Value | Measured Value |
|---|---|---|---|
| Shunt Voltage | 733 | 1.8325mV | 1.8mV |
| Bus Voltage | 1730 | 2.16V | 2.2V |
| Bus Current | 1686 | 37.1mA | 36.6mA |
| Bus Power | 163 | 77.7mW | 80.5mW |

*Table 14: INA226 results vs measured results*

The calibration value, as well as the calculated values were found using the following formulas:

$$Current\ LSB\ = \frac{0.75A}{2^{15}} = 0.000022$$

$$CAL\ = \frac{0.00512}{Current\ LSB\ \times\ 0.05\Omega} = 4474$$

$$Shunt\ Voltage\ =\ Returned\ Value\ \times 0.0000025$$
$$Bus\ Voltage\ =\ Returned\ Value\ \times 0.00125$$
$$Bus\ Power\ =\ Returned\ Value\ \div\ 2097$$
$$Bus\ Current\ =\ Returned\ Value\ \times 0.022$$

# Microcontroller & Bounce

## Test Plan

The microcontroller will use SPI to increase the gain of the PGA on the bounce PCB. A function generator will provide the bounce circuit with an input signal. The microcontroller will then sample the bounce circuit output signal as fast as possible, using DMA.

Pass Conditions:

The Microcontroller must be able to change the PGA gain setting using SPI.
The microcontroller must also be able to capture the waveform from the output pin on the bounce circuit, using an ADC and DMA.

## Results

The following results were obtained from the microcontroller, and not through the GUI. The bounce signal was fed through the bounce circuit with a gain setting of 10. Because of the -20dB attenuation of the input voltage divider on the bounce circuit, the PGA at 1x gain would output a maximum voltage of 0.33V. The results show that the output signal has a max voltage of 3.3V, and therefore the PGA gain setting of 10 must be working.

The output of the bounce circuit was then fed into the microcontroller ADC, using a resolution of 8 bits. Approximately 4000 samples were taken over 4ms, and the data was then output to the PC via the STM32 debug VCP to the console. The data was then copied to a spreadsheet, where the results could be graphed:

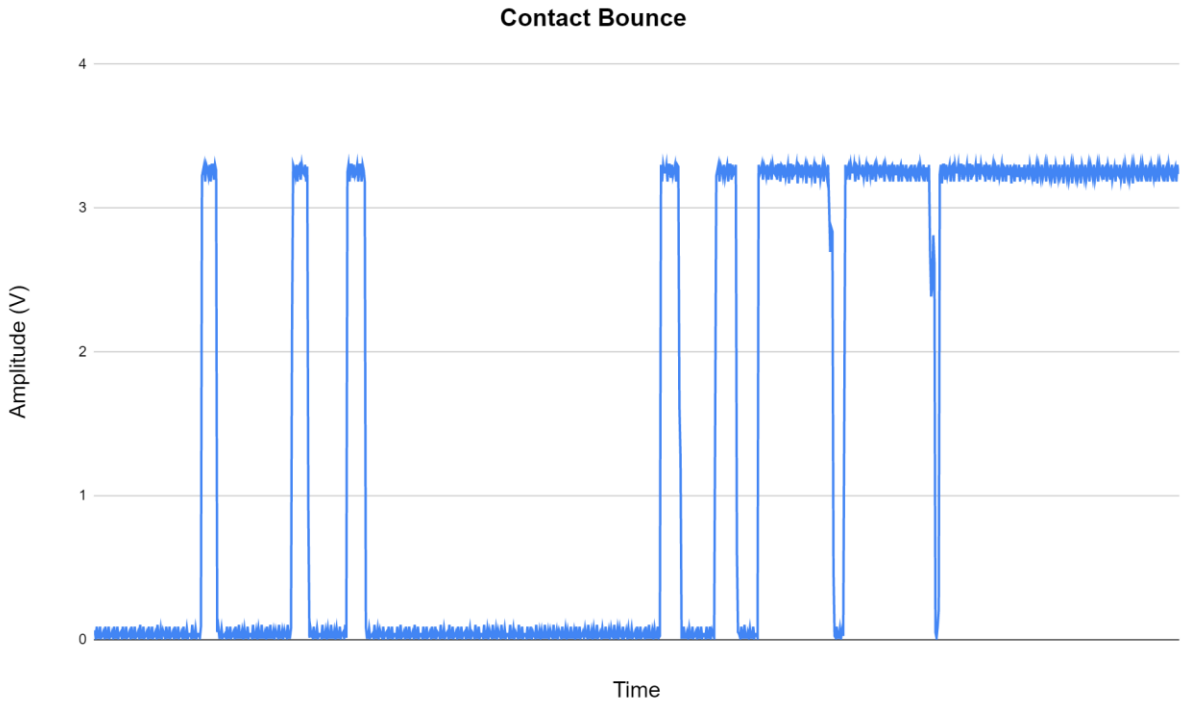



*Figure 59: Bounce data*

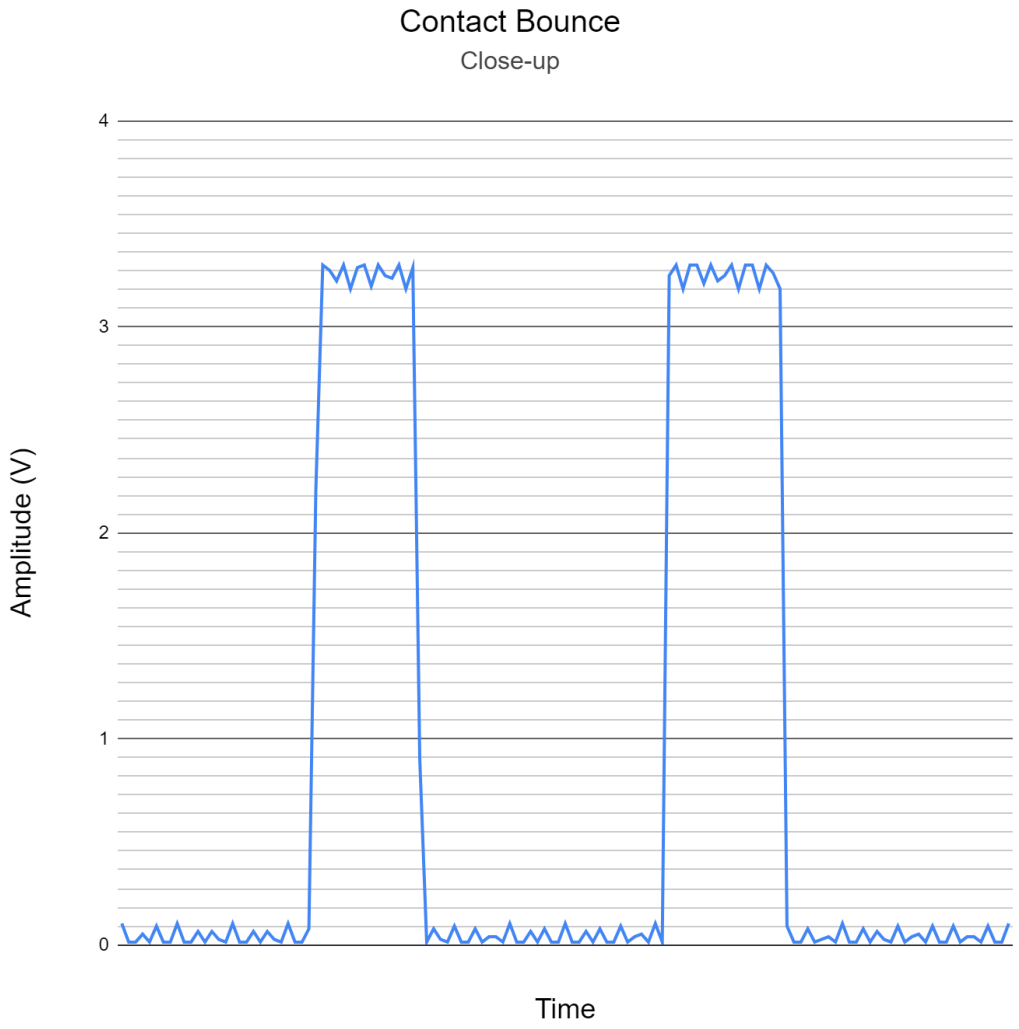



*Figure 60: Bounce data close-up*

From *Figure 59*, we can see that the ADC captures a sufficient amount of data in order to see the major features of the contact bounce. This is perfectly acceptable, as the exact voltage and time are not as important as seeing the number of transitions between high and low, as well as the approx duration of the bouncing.

*Figure 60*, is a close-up of the first graph, specifically the second and third low-high-low transitions. From this graph, we can see that the smallest ΔV is less than 50mV. This is confirmed by calculating the minimum voltage step of the ADC:

$$\Delta V_{MIN} = 3.3V \div 2^8 \cong 0.0129V$$

**Due to time constraints, the tests of the microcontroller with the remaining test PCBs was done during the development of software GUI and the final firmware code.**

# Firmware

**General Test Equipment**

- STM32F767ZI Nucleo board
- PC running Windows or Linux with STM32CubeIDE/MX installed
- Micro-B to Type A USB cable

**General Test Setup:**

For all firmware tests, the STM32F767ZI Nucleo board is connected to the PC via the ST-Link.

**General Code:**

The following should be placed in USER CODE Includes in main.c

```
25 /* USER CODE BEGIN Includes */
26 #include <string.h>
27 #include <stdio.h>
```

**NOTE:** All arrays can be monitored in debug mode with STM32CubeIDE by hovering over them with the cursor while the debugger is paused.

## ADC with DMA

**Additional Test Equipment:**

- Potentiometer of any value
- 3 jumper wires
- breadboard

**Test setup:**

Place the potentiometer on the breadboard with high side connected to Nucleo 3.3V, low side to Nucleo ground and wiper connected to the ADC pin (D32) on the STM32F767ZI using jumper wires. Using STM32CubeMX, configure the ADC1 settings to IN0, PCLK2 divided by 6, 8 bit resolution, right alignment, Continuous conversion mode enabled. Configure the DMA2 settings for a DMA request on ADC1 in circular mode. Configure ADC1 to have DMA continuous requests enabled. Leave all other settings at default, as seen in *figures 61a and 61b*. Fill in the provided code.

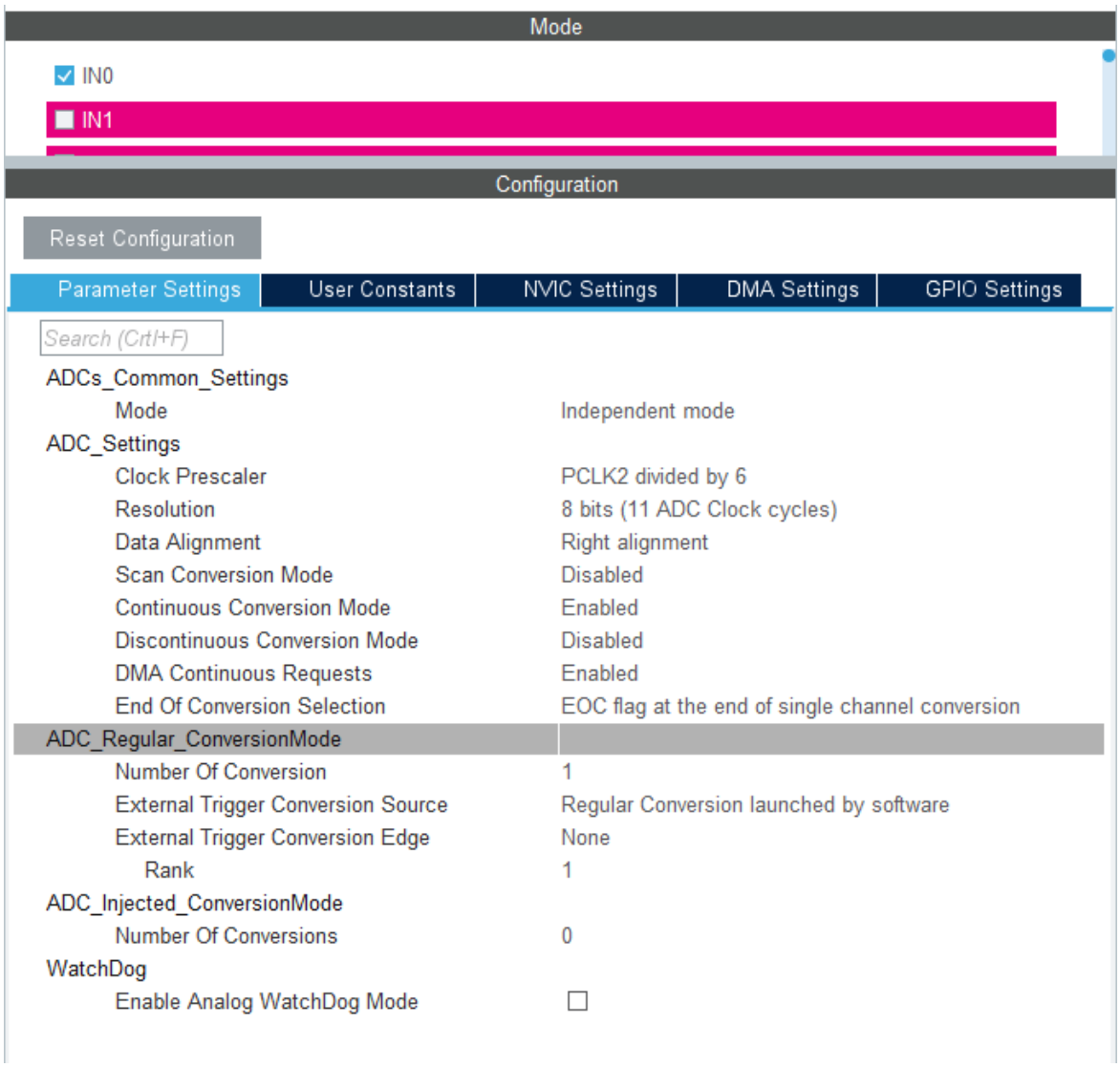



*Figure 61a: ADC CubeMX settings*

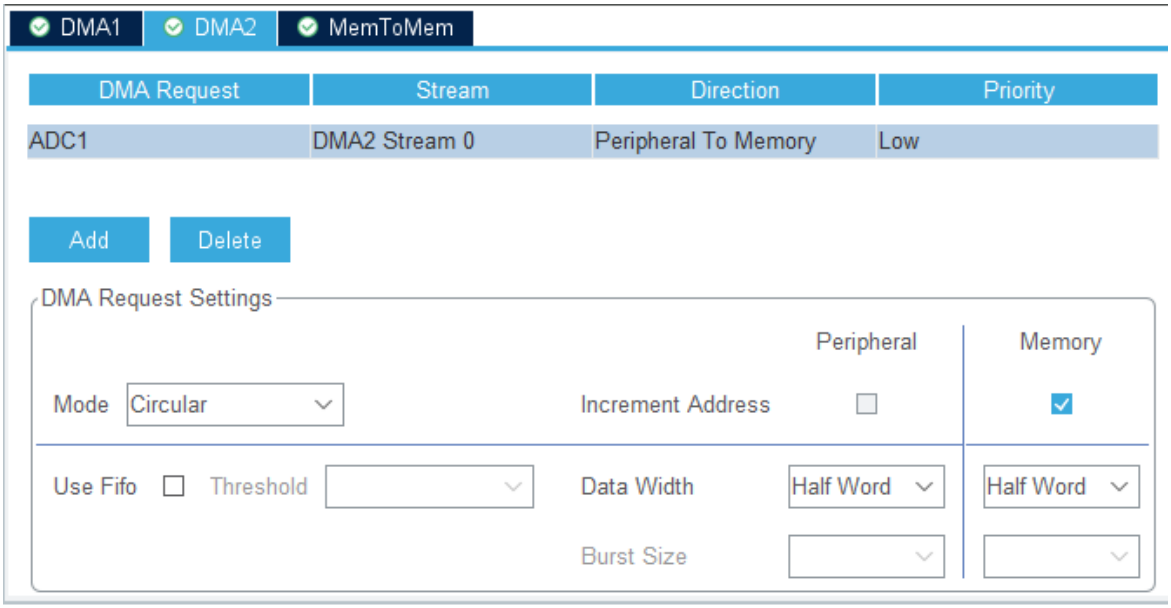



*Figure 61b: DMA CubeMX settings*

**Test Plan:**

Turn the potentiometer all the way to the lowest resistance. Run the firmware in debug mode. Wait a second then pause the debug. Turn the potentiometer all the way the other direction. Restart the debug. Wait a second then pause the debug again.

**Code:**

See Appendix B section 1.

**Pass conditions:**

Values that are in the 240 - 248 range will fill the buffer on the first run as in *figure 62*.

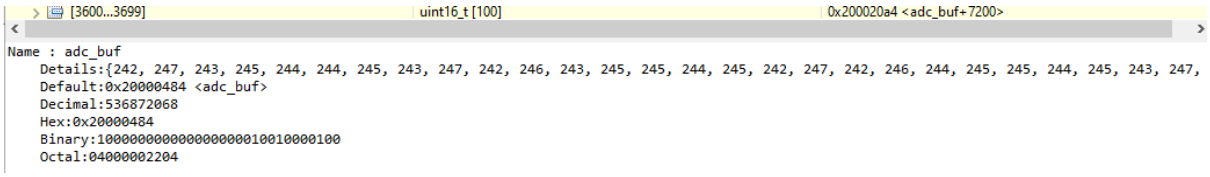



*Figure 62: DMA test*

On the second run, values that are close in magnitude and in the 20-12 ranges as in *figure 63*. Precision of the potentiometer may affect the size of the ranges.

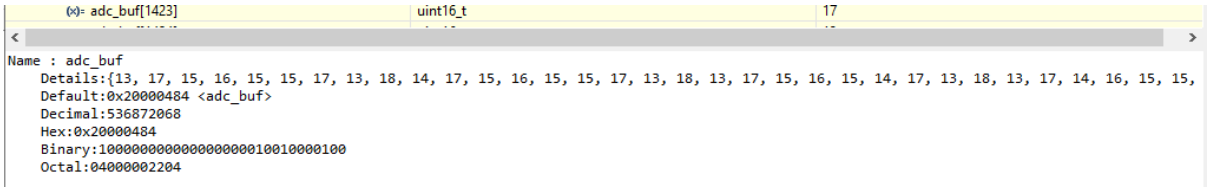



*Figure 63: DMA test 2*

# USART

### Additional Test Equipment

- Micro-B to Type A USB cable

#### Test Setup:

Using STM32CubeMX, configure the USART settings to Asynchronous, RS232 disabled, 9 bits, Odd parity. Leave all other settings at default, as seen in *figure 64*. Connect the Nucleo to a computer using a USB cable via the Nucleo's user USB port. Fill in the provided code.

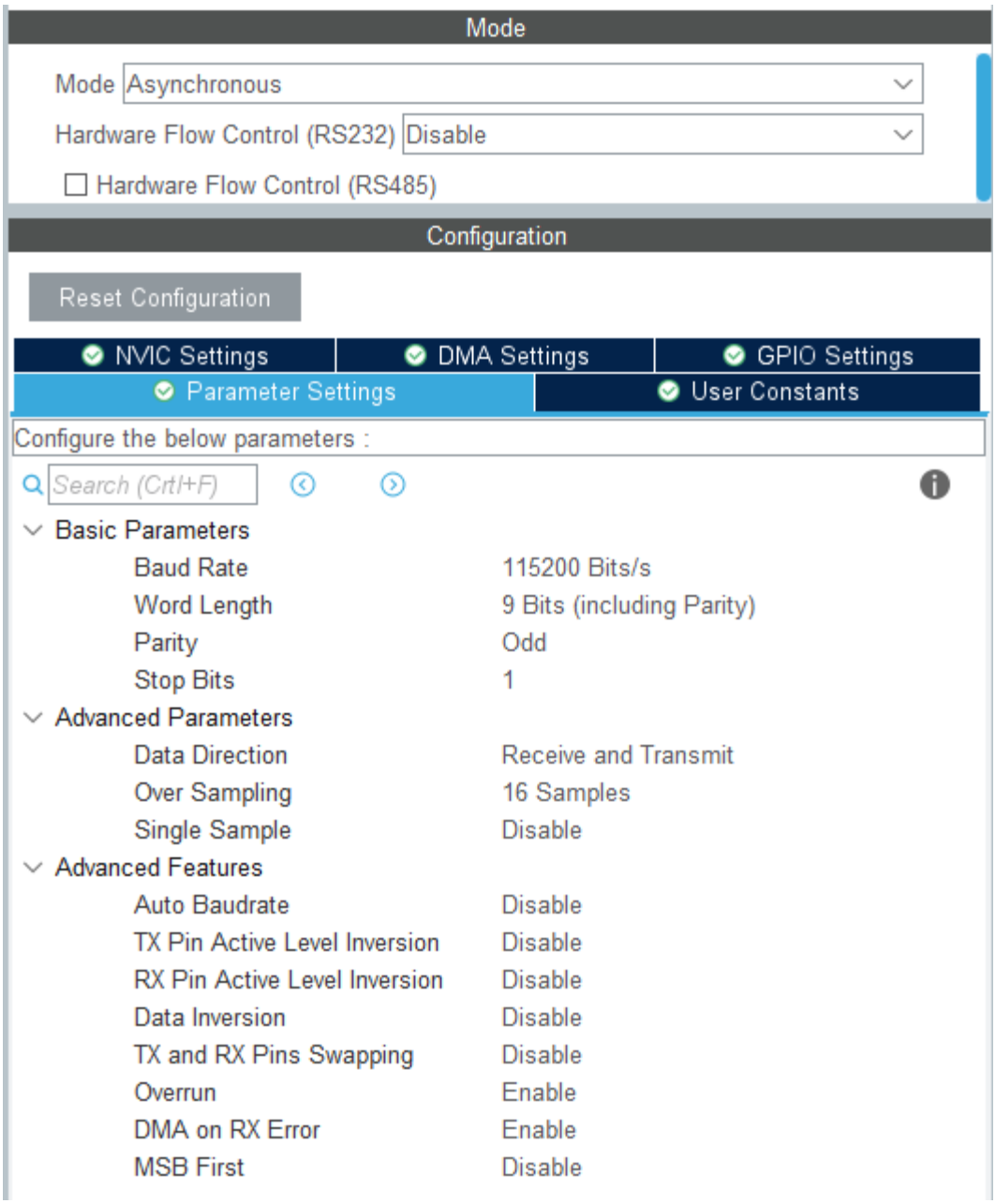



*Figure 64: USART CubeMX settings*

**Test Plan:**

Run code in the debug mode. Open a new command shell console, configure serial port settings to match the USART settings. Select your desired COM port and ISO. Press ok and then run the code. Make sure to close the console after completing the test.

**Code:**

See Appendix B section 2.

**Pass conditions:**

“test ” will be repeated every half second in the command shell console as in *figure 65*.

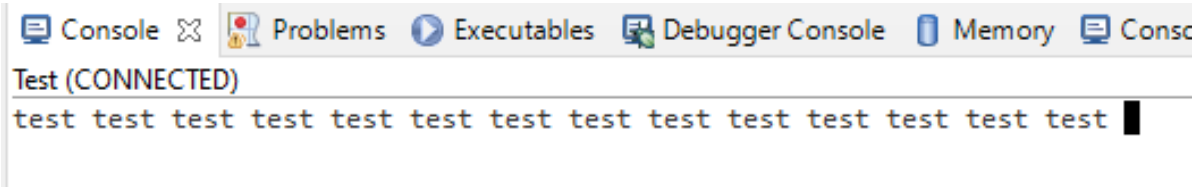



*Figure 65: USART test*

# RTC

**Additional Test Equipment**

- Stopwatch (or device with a stopwatch function)

**Test Setup:**

Using STMCubeMX, configure the RTC settings to 24 hour format with an asynchronous predivider value of 127 and synchronous predivider value of 255. Leave all other settings at default, as seen in *figure 66*. Nucleo board is connected in ST-Link mode to PC. Fill in the provided code.

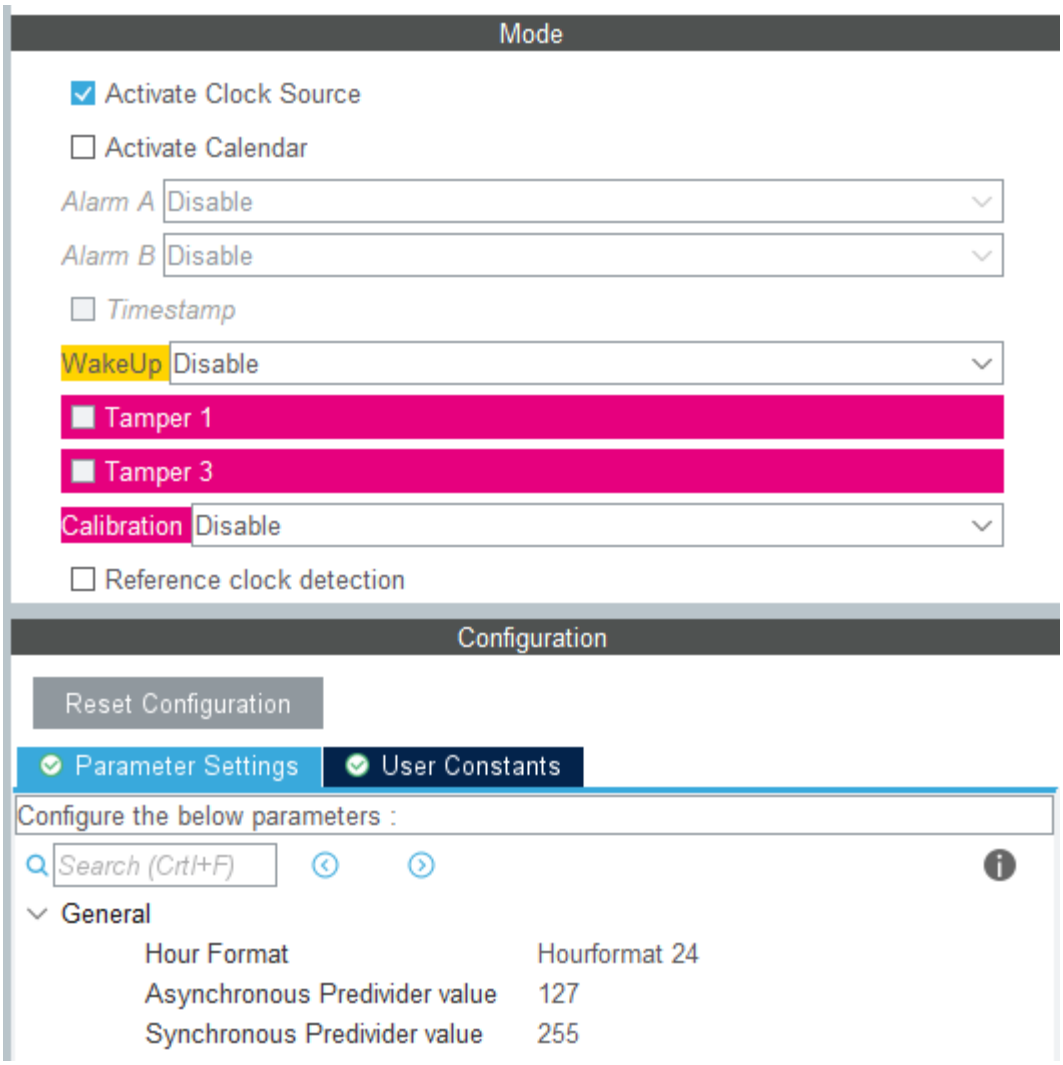



*Figure 66: RTC CubeMX settings*

**Test Plan:**

Place a breakpoint on the line where the **setTime** function is called in **main.c**. Run code in the debug mode. Once the breakpoint is hit, run the code and start an seperate stopwatch simultaneously. Pause the debug and stopwatch simultaneously after a few seconds. Monitor the values in **time** and **date**. Run the debug again, pause and monitor the values in **time** and **date**.

**Code:**

See Appendix B section 3.

**Pass conditions:**

The value in **time** will progress by the same amount measured by the stopwatch within 0.5s to allow for inaccuracies introduced by the debugger as seen in *figures 67* and *68*. Note that the subseconds are on a scale of 0-255, not 0-999. **date** will be set to 01 01 as seen in *figure 69*.

Stopwatch time: 00:15.17

```
Name : time
    Details:"00:00:15:240"
    Default:0x20000478 <time>
    Decimal:536872056
```

*Figure 67: RTC time test 1*

Stopwatch time: 1:00.21

```
Name : time
    Details:"00:01:00:225"
    Default:0x20000478 <time>
    Decimal:536872056
```

*Figure 68: RTC time test 2*

```
Name : date
    Details:"01 01"
    Default:0x20002488 <date>
    Decimal:536880264
```

*Figure 69: RTC date test*

# ADC

**Additional Test Equipment**

- Potentiometer of any value
- 3 jumper wires
- Breadboard
- Micro-B to Type A USB cable

**Test Setup:**

Place the potentiometer on the breadboard with high side connected to Nucleo 3.3V, low side to Nucleo ground and wiper connected to the ADC pin (A0) on the STM32F767ZI using jumper wires. Using STM32CubeMX, configure the ADC2 settings to IN3, PCLK2 divided by 6, 12 bit resolution, right alignment, Continuous conversion mode enabled. Configure the USART settings as per the USART test. Leave all other settings at default, as seen in *figure 70*. Connect the Nucleo to a computer using a USB cable via the Nucleo's user USB port. Leave all other settings as is. Fill in the provided code.

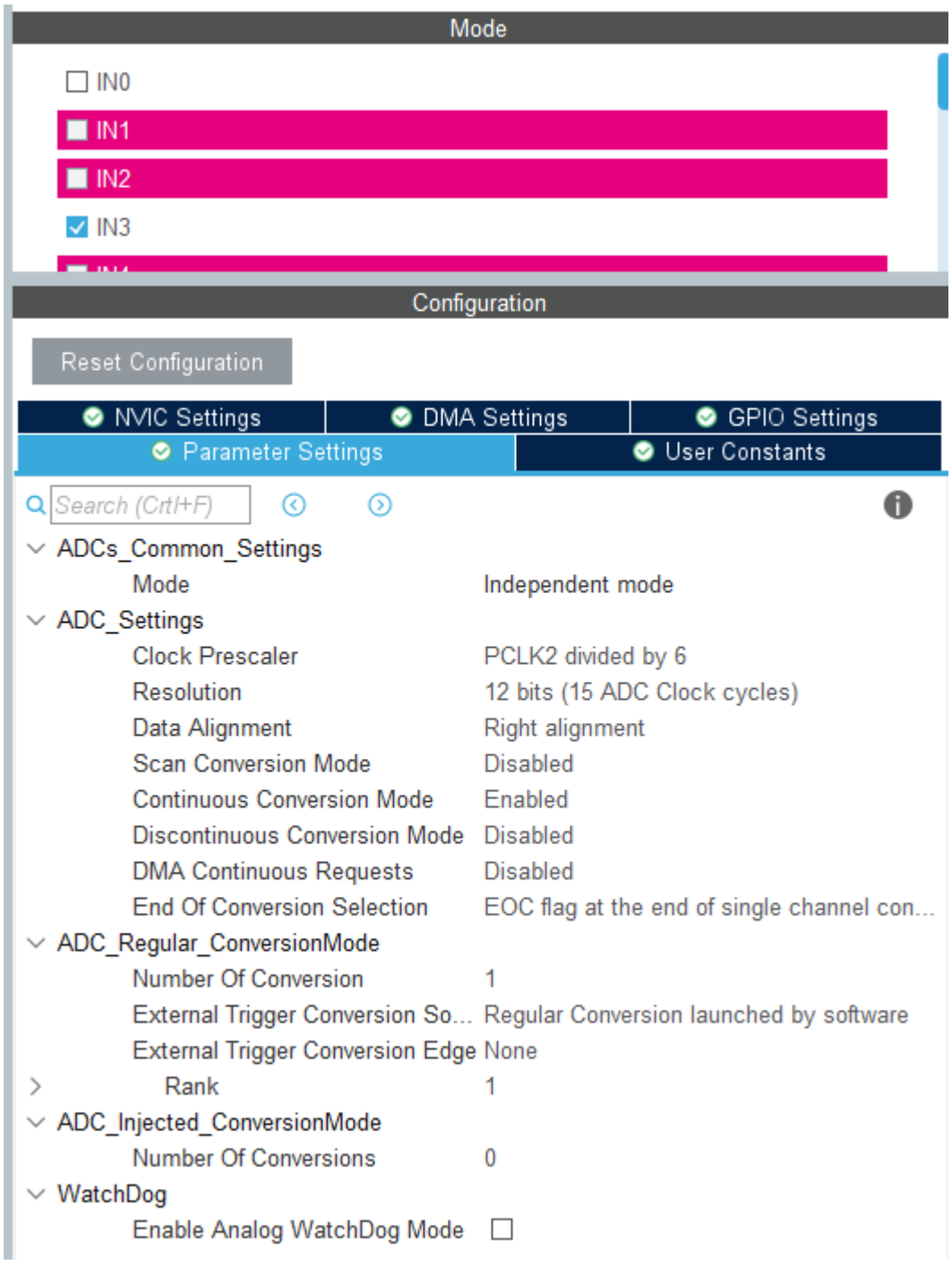



*Figure 70: ADC2 CubeMX settings*

**Test Plan:**

Turn the potentiometer all the way to the lowest resistance. Open a new command shell console, configure serial port settings to match the USART settings. Select your desired COM port and ISO. Press ok and then run the code. Run the firmware in debug mode. In the console

window, the values from the ADC should be listed in real time. Note the values. Turn the potentiometer all the way the other direction. Note the values again.

**Code:**

See Appendix B section 4.

**Pass conditions:**

Values greater than 4080, approaching 4095, as in *figure 71,* will be listed when the potentiometer is at the lowest resistance. When it is at the highest, there will be values less than 225, approaching 0 as in *figure 72*. Precision of the potentiometer may affect the size of the ranges.

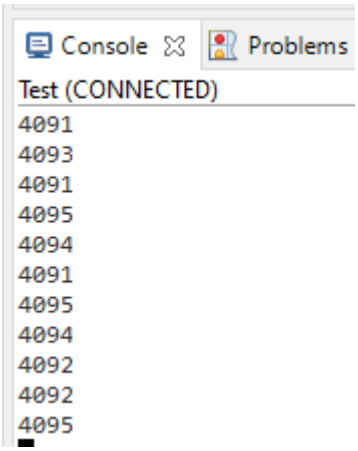



*Figure 71: ADC test 1*

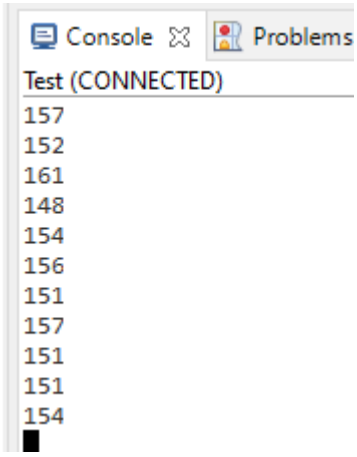



*Figure 72: ADC test 2*

# $I^2C$

**Additional Test Equipment**

- 5 jumper wires
- Relay Analyzer power supply with mains plug
- Oscilloscope with 2 probes

**Test Setup:**

Using STM32CubeMX, configure the I2C1 settings as shown in *figure 73* with PB6 as SCL and PB9 as SDA. Fill in the code provided. Using the tested Power Supply board, connect 3V3 to the DAC pin and 5V to the Enable pin. Connect the power supply to the STM32F767ZI with the following connections: SDA to pin PB9, SCL to pin PB6 and GND to any ground pin. Connect the channel 1 probe from the oscilloscope to the SDA line and channel 2 probe to the SCL line with both ground clamps to the Nucleo's ground. Configure the oscilloscope to single trigger at the start of a I2C transmission and decode I2C in hex. Connect the power supply to mains power.

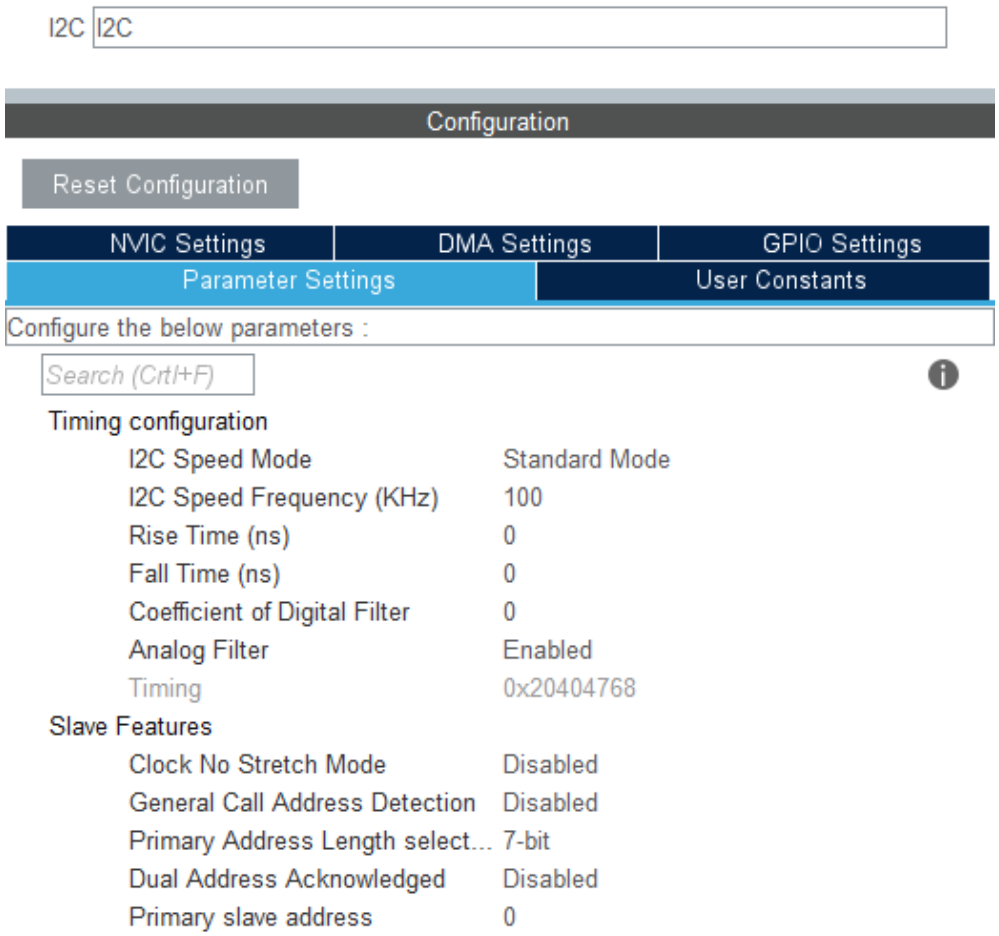



*Figure 73: $I^2C$ CubeMX settings*

**Test Plan:**

Place a breakpoint on the wait command. Run in debug mode. Ensure the oscilloscope has triggered on the signal, and that all 7 bytes are decoded. Adjustment of the time scale and location may be required to achieve this.

**Code:**

See Appendix B section 5.

**Pass conditions:**

4 bytes are written to address 90: 00, 47, 27 as seen in *figure 74*. Then 2 bytes are read from address 91: 47, 27 as seen in *figure 75*.

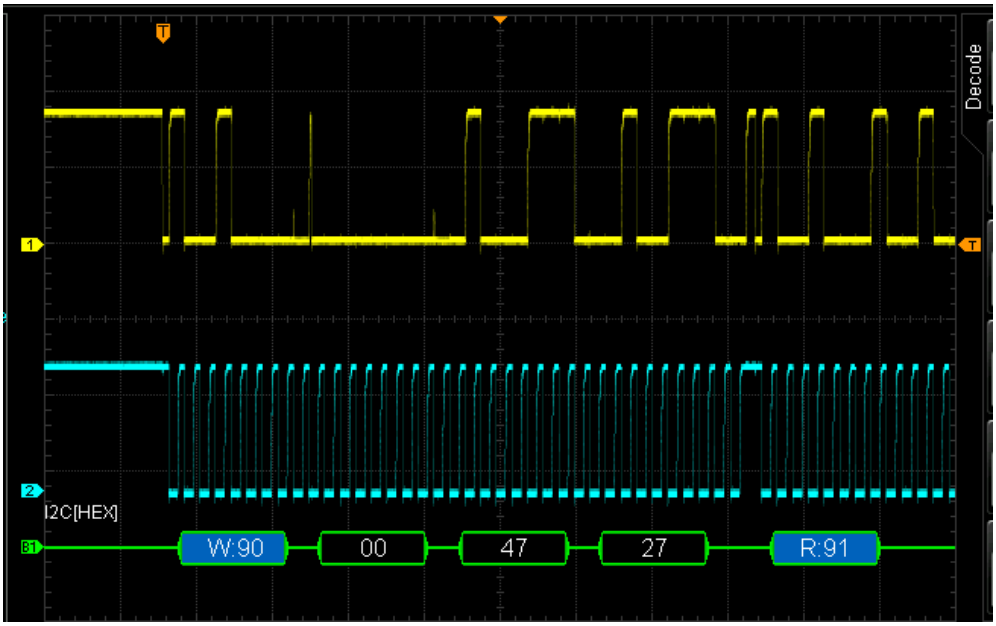



*Figure 74: I²C scope shot*

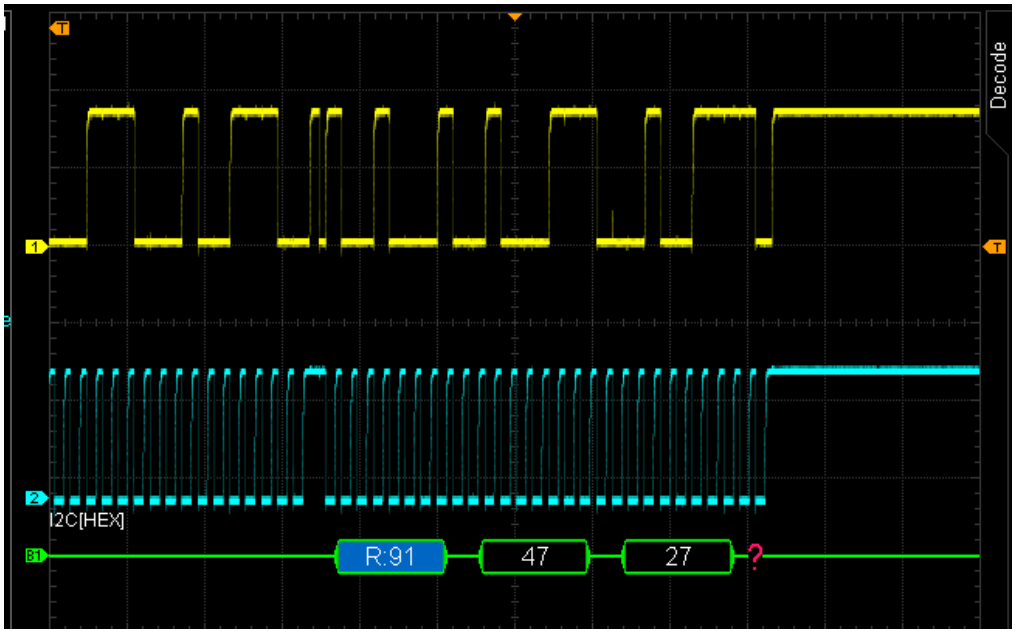



*Figure 75: I²C scope shot*

SPI

**Additional Test Equipment**

- Oscilloscope with 3 probes

**Test Setup:**

Using STM32CubeMX, configure the SPI settings as shown in *figure 76* with PB10 as SCK, PB12 as NSS and PC3 as MOSI. Fill in the code provided. Connect the channel 1 probe from the oscilloscope to pin PC3, the channel 2 probe to pin PB10 and the channel 3 probe to PB12. Configure the oscilloscope to single trigger on the transmission and decode 16 bit SPI in hex with MSB order.

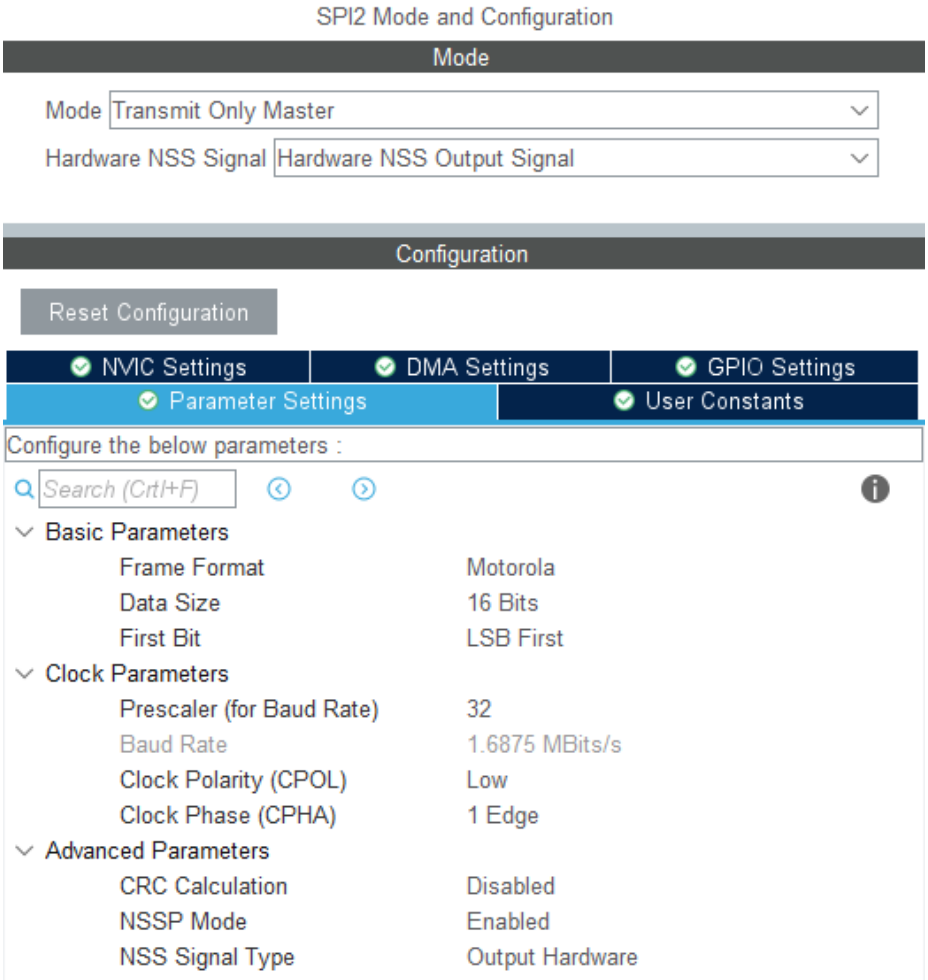



*Figure 76: SPI CubeMX settings*

**Test Plan:**

Place a breakpoint on the wait command. Run in debug mode. Ensure the oscilloscope has triggered on the signal, and that both bytes are decoded. Adjustment of the time scale and location may be required to achieve this.

**Code:**

See Appendix B section 6.

**Pass conditions:**

5A0F is decoded by the oscilloscope. Waveforms matching *figure 77*.

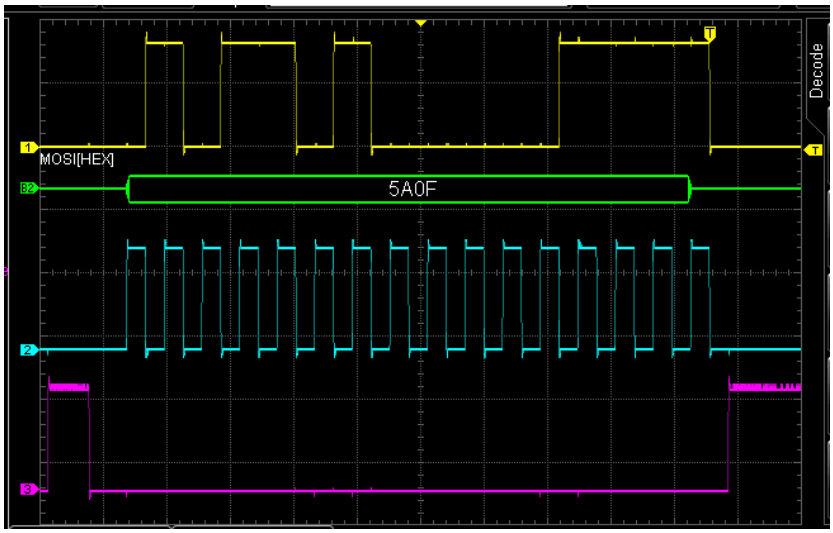



*Figure 77: SPI scope shot*

# Software

The software was tested on-the-fly while it was being developed in parallel with the firmware. This was because of time constraints. Therefore, there was no test plan. This section describes the different tests on the software as they were done.

For the USB part of the software, please see Appendix B.

First, each piece of data that was sent to the software needed to be converted in some way. An example is the bus voltage (coil voltage). It is calculated using a integer value obtained from the INA226, and uses the following formula:

$$Bus\ Voltage\ =\ Returned\ Value\ \times 0.00125$$

Each such value from the Relay Analyzer was tested to make sure that the calculated value was correct.

Next, each input field in the software was tested to make sure that the implemented validation worked correctly.

Lastly, the graphs in the software were tested to make sure that they displayed the right data, and in an easy to understand way.

For the most part, Qt handles most of the GUI and event handling, therefore software testing was kept to a minimum.

## Photos of Project

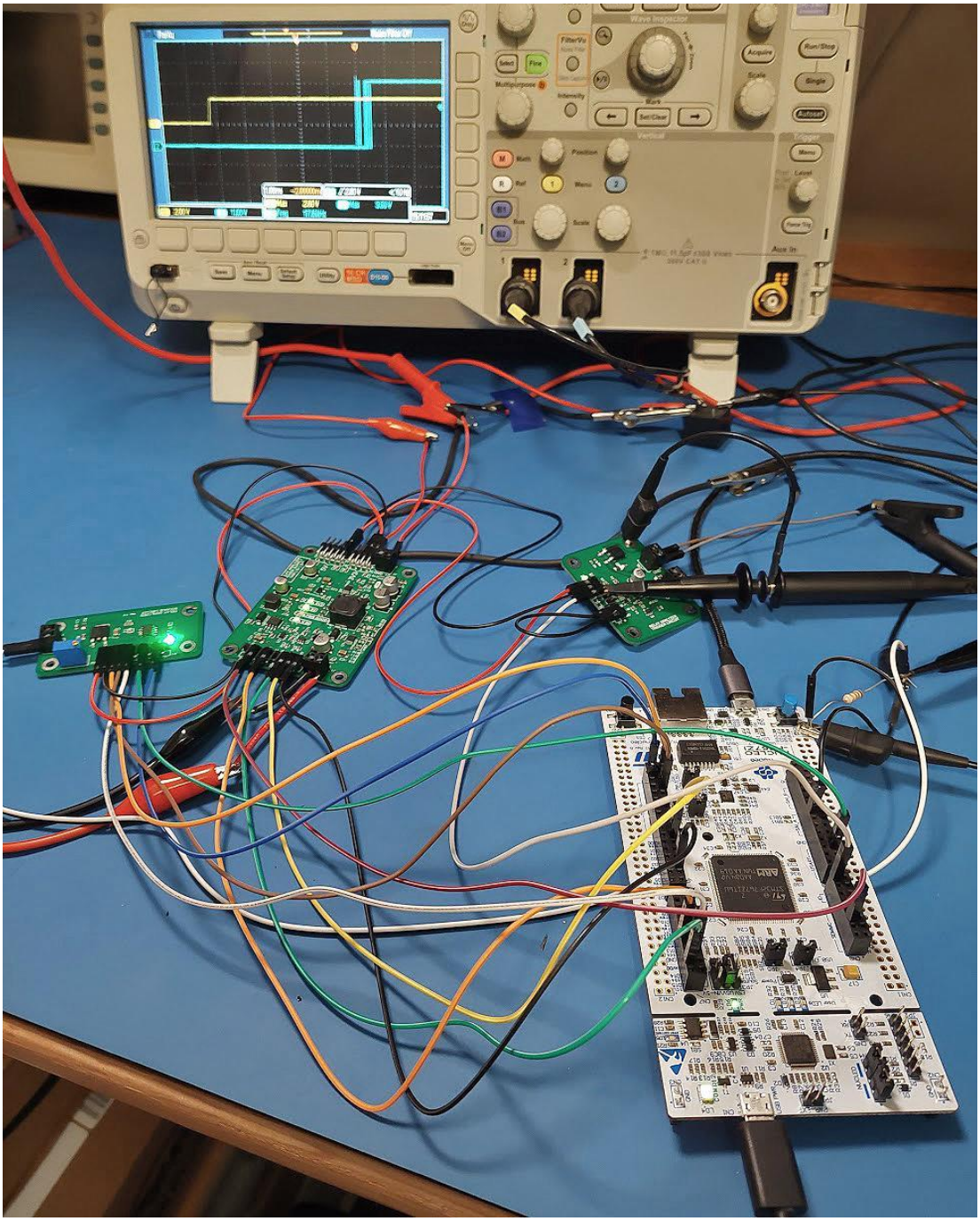


*Figure 78: Photo of final system with scope shot of relay activation*

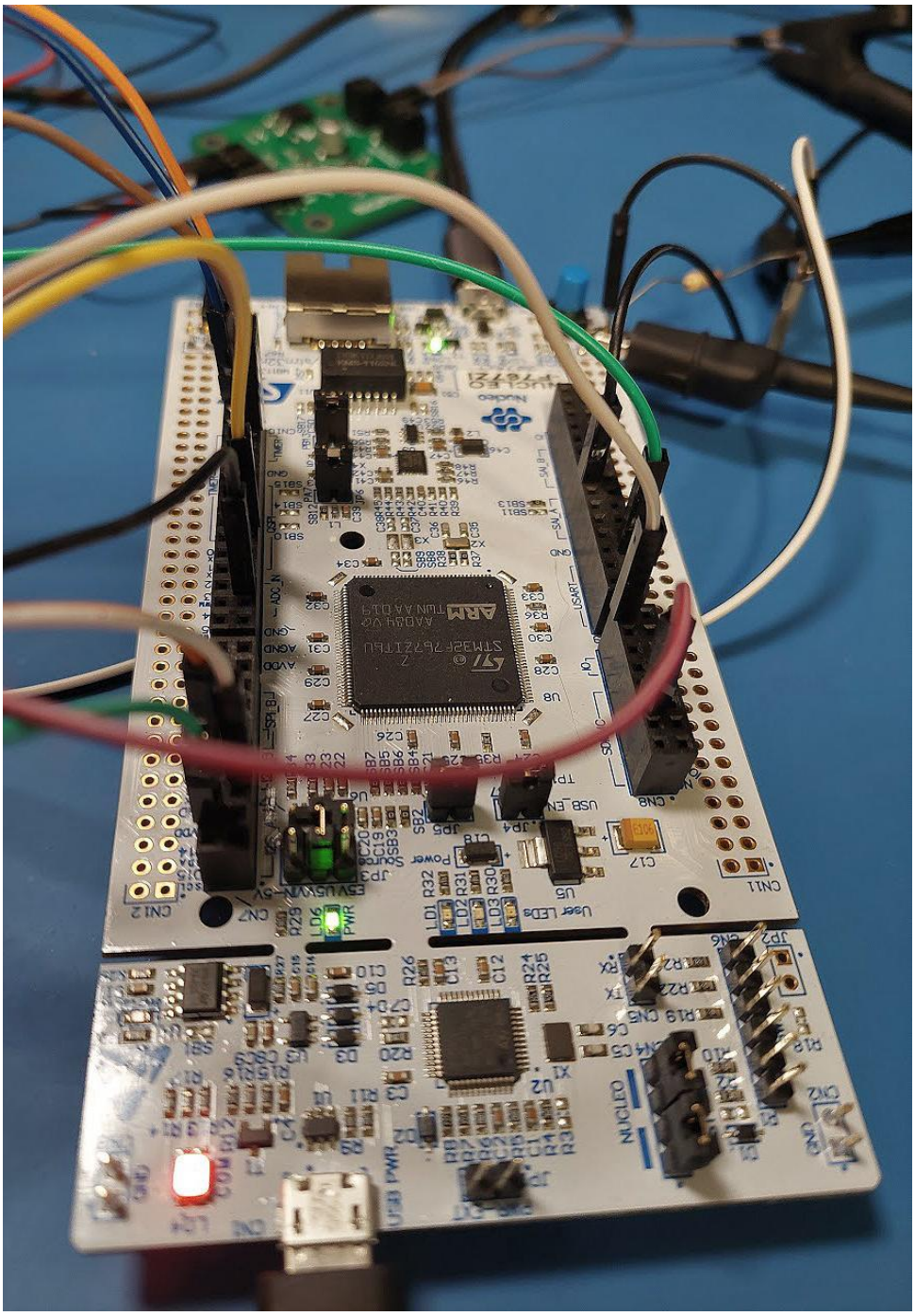


*Figure 79: Photo of STM32 microcontroller development board*

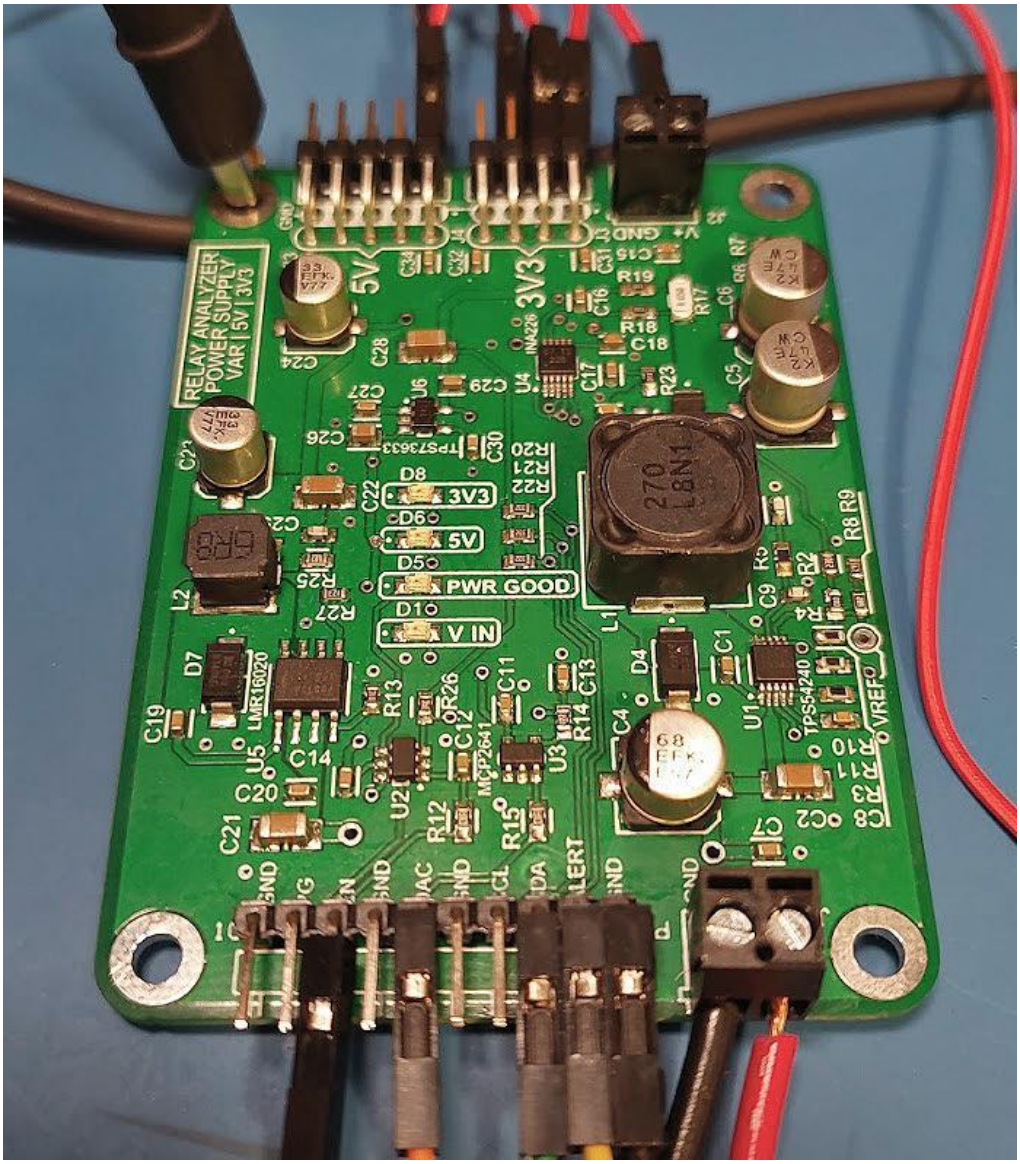



*Figure 80: Photo of power supply PCB, including variable power supply, 3V3 and 5V rails*

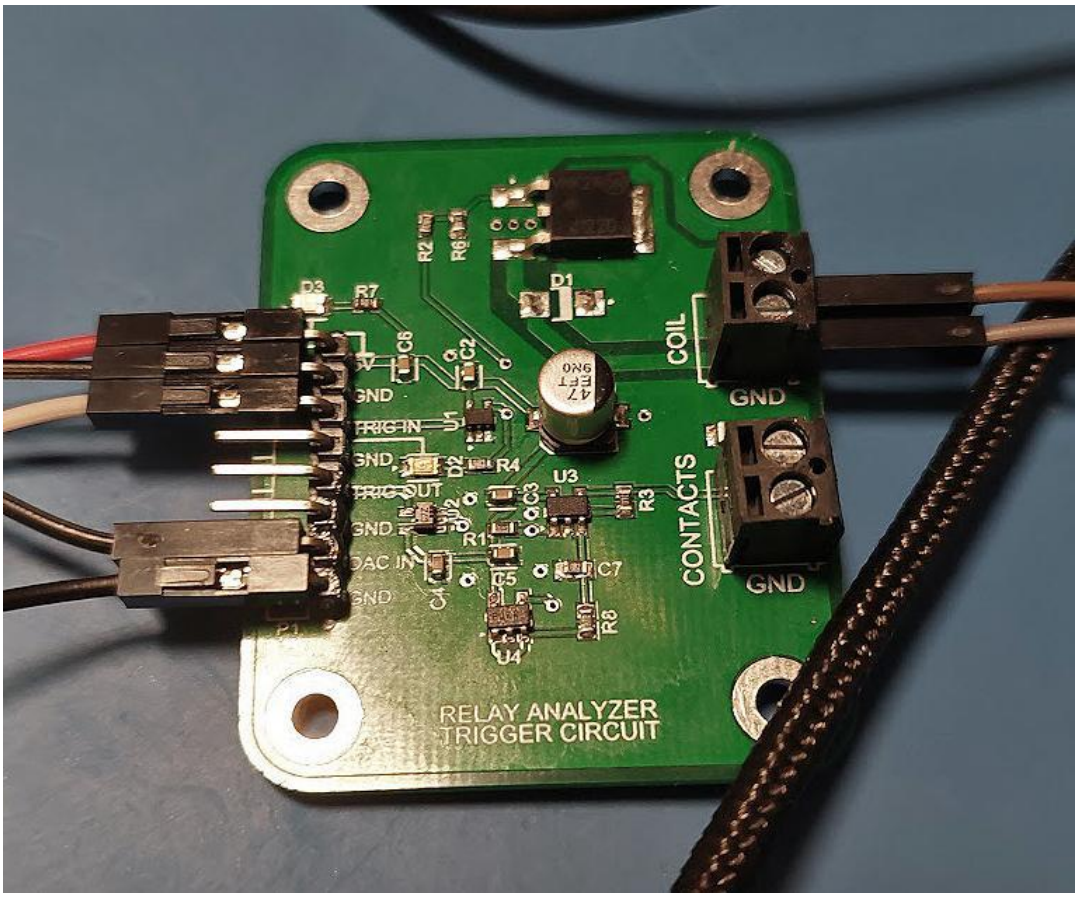



*Figure 81: Photo of trigger PCB*

# Results and Analysis

For the results, two relays were tested, in order to compare their characteristics with those found in the manufacturer datasheets, and to discover some characteristics that are not specified in the datasheets. Since getting noticeable changes in characteristics can take thousands of cycles, during each test, the coil voltage is *decreased* by a small amount. This results in a

noticeable change in several characteristics, including activation and deactivation times, coil current and power, and coil resistance.

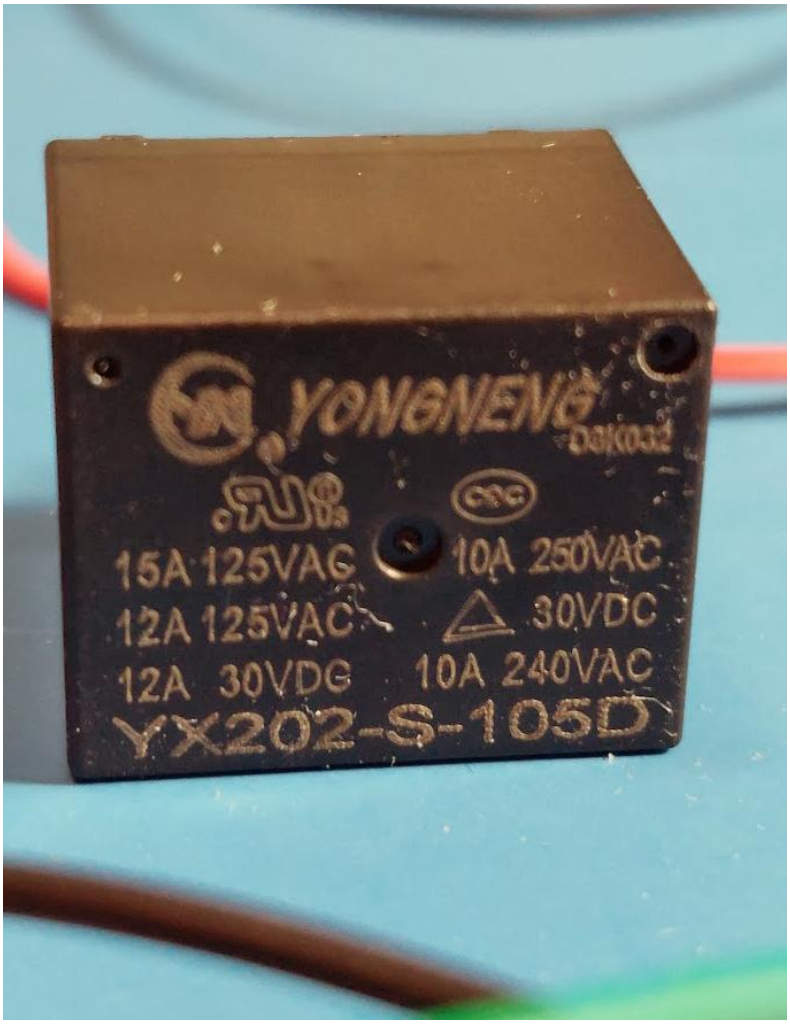



*Figure 82: Photo of 5V Yongneng relay*

The first relay is the Yongneng YX202. This is a standard relay with the following specifications:

| **Characteristic (Datasheet)** | **Specification** |
|---|---|
| Coil voltage | 5V |
| Operating time (activation time) | 8ms max. |

| | |
|---|---|
| Release time (deactivation time) - *without* diode to dissipate back EMF | 5ms max. |
| Coil current | 71.4mA ± 10% |
| Coil resistance | 70Ω ± 10% |
| Rated coil power (approx.) | 0.36W |

*Table 15: Yongneng YX202 relay specifications*

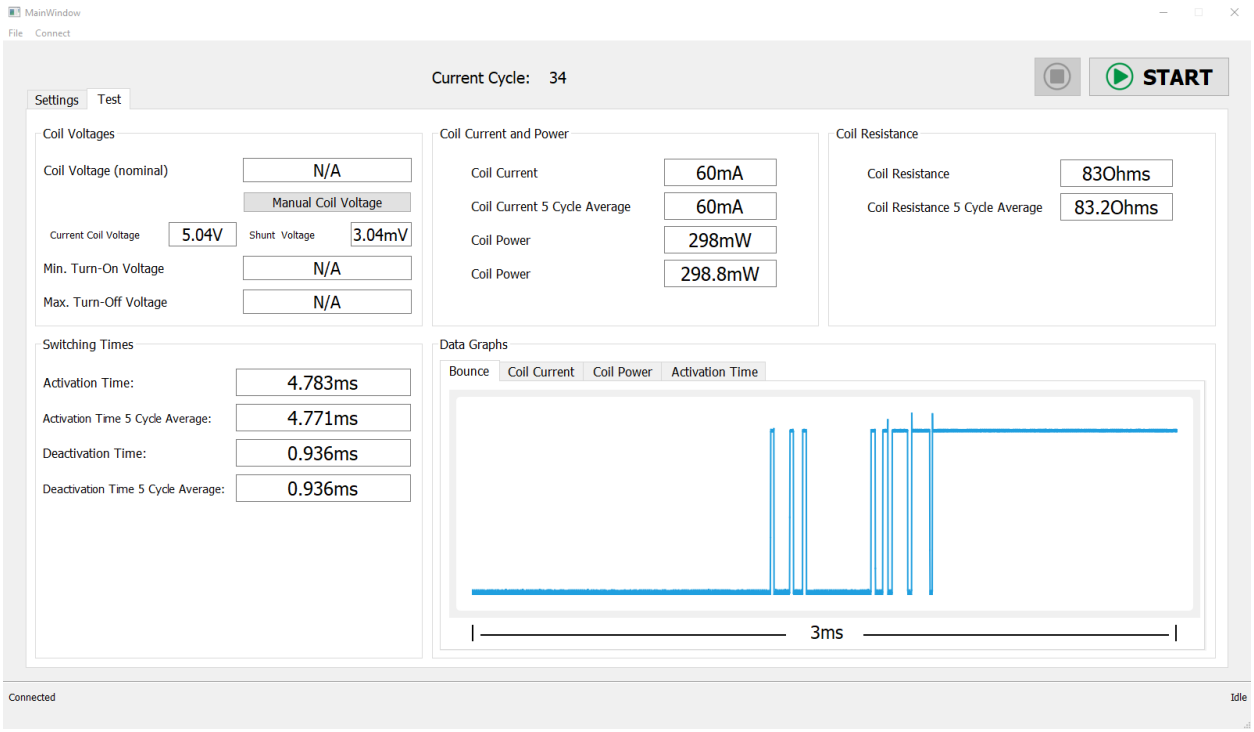



*Figure 83: Screenshot of 5V test results with bounce waveform*

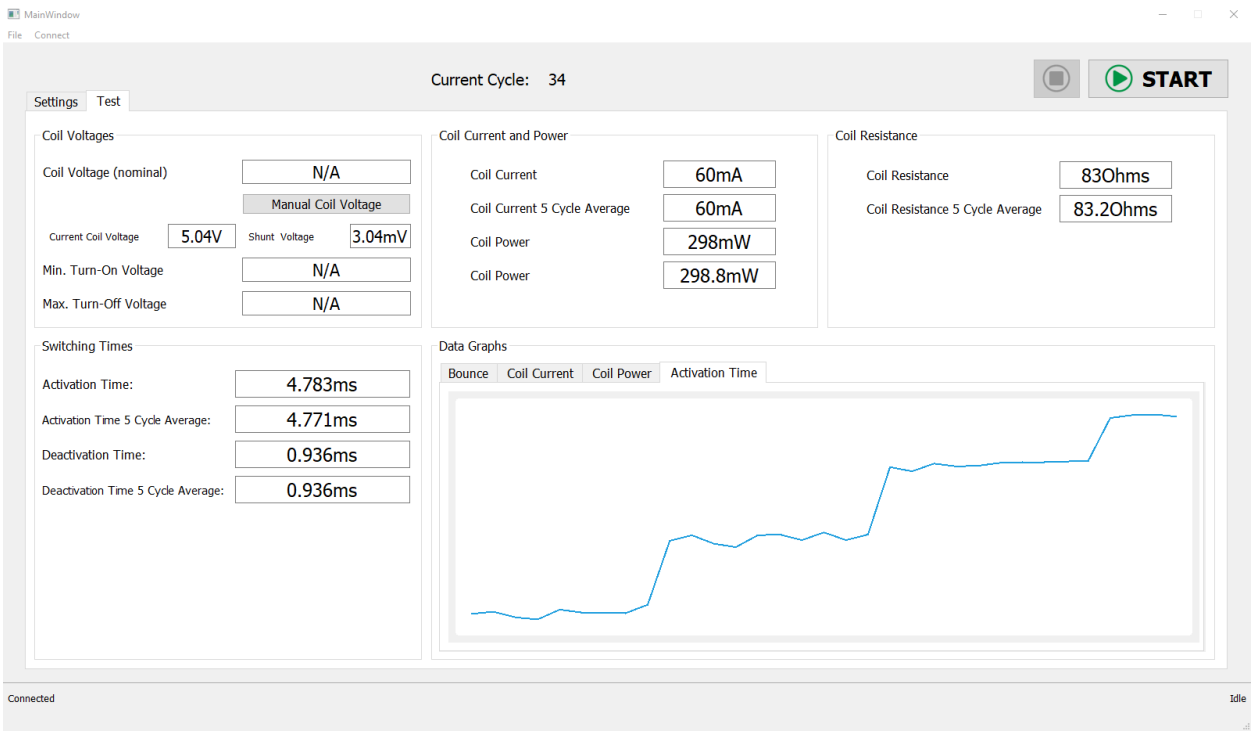



*Figure 84: Screenshot of 5V test results showing change in activation time*

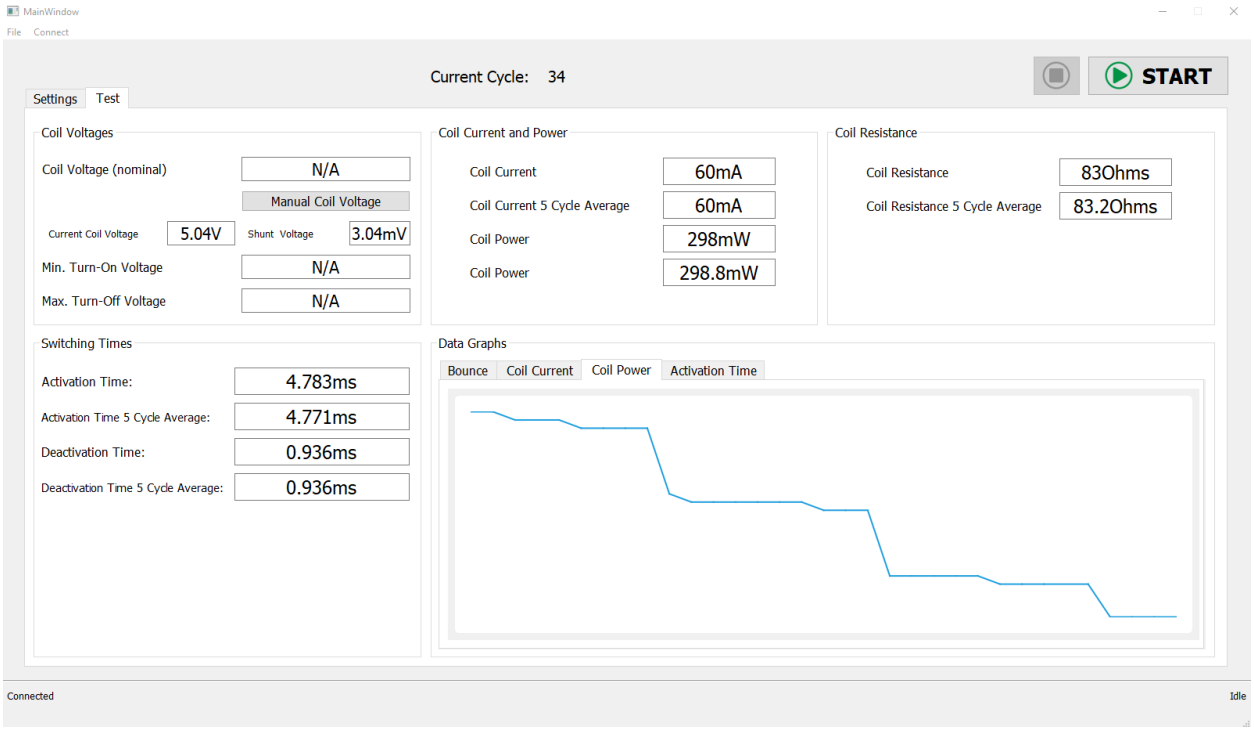



*Figure 85: Screenshot of 5V test results showing change in coil power*

As seen in *Figure 83, Figure 84, and Figure 85*, above, several characteristics were tested for by the Relay Analyzer. The results are listed below, with a calculated difference between the results and the datasheet.

| Characteristic (Relay Analyzer) | Result | Difference |
|---|---|---|
| Coil current | 60mA | 6.63-23.6% |
| Coil power | 0.298W | 17.2% |
| Coil resistance | 83Ω | -18.57% |
| Activation time (operating time) | 4.783ms | |
| Deactivation time (release time) - *with* diode to dissipate back EMF | 0.936ms | |

*Table 16: Yongneng YX202 test results*

From the results, we can see that there is a significant difference between the specifications from the datasheet, and the Relay Analyzer results. However, there are some key points that make this difference acceptable, and even expected. First, there are errors in the data collected by the Relay Analyzer. The most significant of this comes from the coil voltage reading. The INA226 current & voltage monitor on the power supply is directly next to the output capacitors. During the test, there was a significant voltage drop across the wires connecting the power supply and relay coil. Therefore, the relay was, in reality, getting a smaller voltage across the relay. Second, the relay under test was purchased from a company that buys in bulk, and sells components “as is”. Therefore, it is quite possible that the relay under test had significantly different characteristics than those described in the datasheet. However, this is precisely why the Relay Analyzer exists - to test a relay to find out what it’s characteristics are, whether that be a relay directly from a reputable, high-quality manufacturer, or a relay bought in bulk in order to reduce BOM costs as much as possible.

Using a programmable power supply, and a multimeter, the coil current and power were measured. The coil current was measured to be 62mA, and the coil power measured 310mW. Using these, the coil resistance can be calculated to be 80.6Ω. Comparing these measurements to the ones obtained from the Relay Analyzer, we can see that the difference is much smaller.

| Characteristic (Measurement) | Difference |
|---|---|
| Coil current | 3.22% |
| Coil power | 3.87% |
| Coil resistance | -2.98% |

*Table 17: Yongneng YX202 characteristic differences*

In terms of bounce characteristic, we can see that the relay “bounces” 7 times over approx. 0.75ms before settling. Given the measured activation time (from trigger signal to the *first* edge

on the relay contacts) of 4.783ms, and the bounce time of approx 0.75ms, the total activation time is approx 5.53ms, under the 8ms max specified in the datasheet.


*Figure 86: 12V relay without datasheet*

The second relay is a 12V automotive “Bosch style” relay made by GlobalTone. It does not have a datasheet that can be easily obtained. The specifications listed are from a seller's website [7]. However, we can use the Relay Analyzer to verify the specifications we have, and test for those we don’t. The known specifications are:

| | |
|---|---|
| Coil voltage | 12V |
| Max. coil power | 1.8W |

*Table 18: GlobalTone relay specifications*

Using the Relay Analyzer, the following characteristics were gathered about the relay:

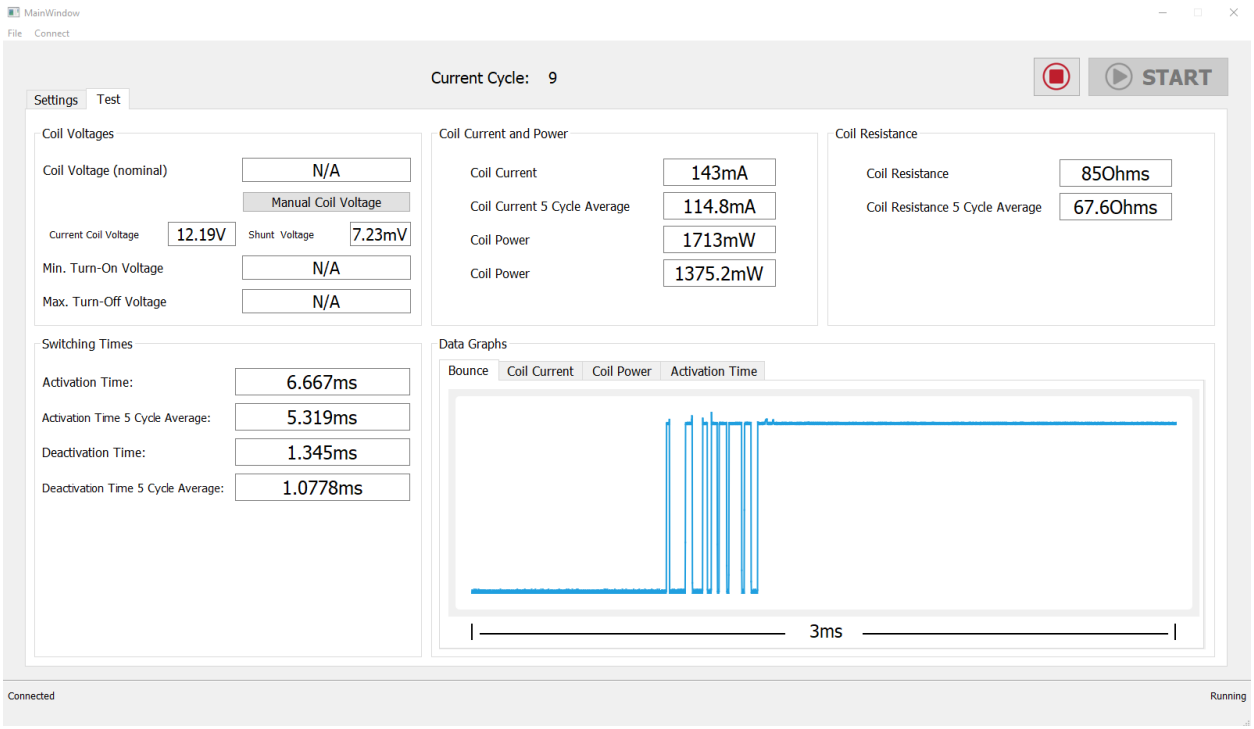



*Figure 87: Screenshot of 12V test results showing bounce waveform*

| **Characteristic (Relay Analyzer)** | **Result** |
|---|---|
| Coil current | 143mA |
| Coil power | 1.71W |
| Coil resistance | 85Ω |
| Activation time | 6.66ms |
| Deactivation time | 1.345ms |

*Table 19: GlobalTone relay test results*

These numbers are compared to those found using a power supply, multimeter & oscilloscope:

| Characteristic (Measurement) | Result |
|---|---|
| Coil current | 152mA |
| Coil power | 1.66W |
| Coil resistance (calculated) | 80.9Ω |
| Activation time | 6.58ms |
| Deactivation time | 1.265ms |

*Table 20: GlobalTone relay measured characteristics*

The results from the Relay Analyzer are very close to those found with test equipment. Also, the one figure given on the website, that is also characterized by the Relay Analyzer, is different by only 90mW, or 5%. At the ~148mA this relay uses, there is even more voltage drop across the wires connecting the relay coil than there was with the 5V relay. Activation and deactivation times are within 7%.

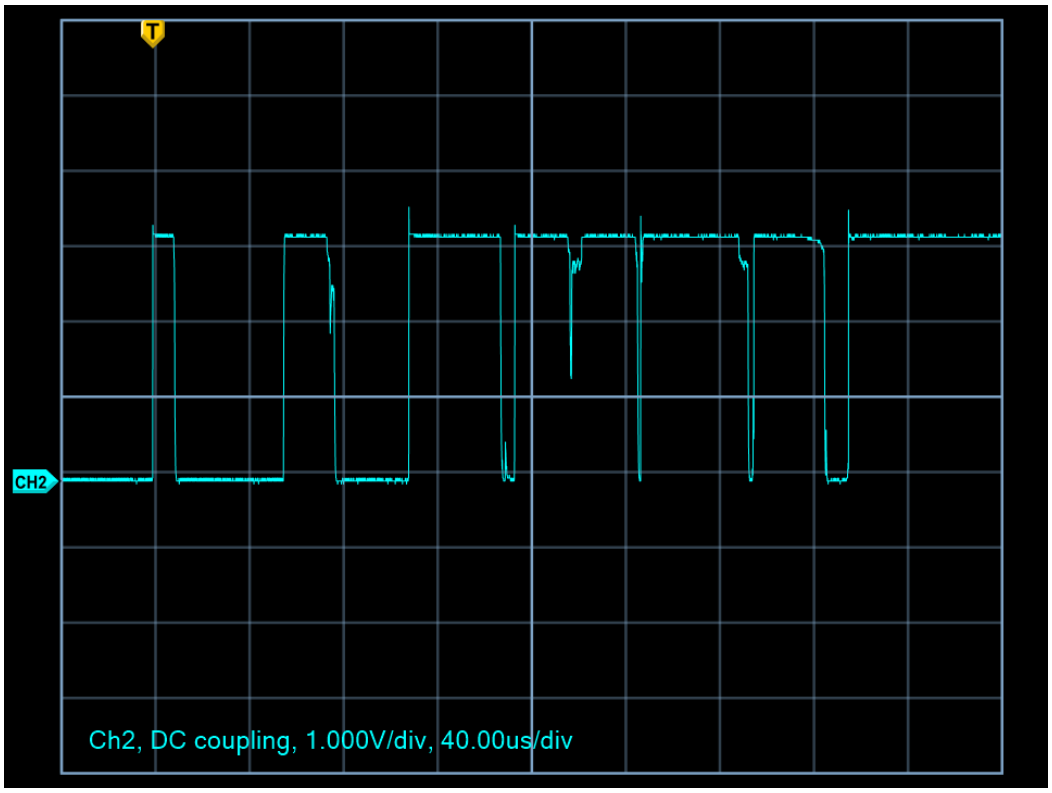



*Figure 88: 12V relay bounce waveform*

The bounce waveform of the 12V relay is shown in *Figure 88*. Bouncing lasts for approx 300µs - similar to the 5V relay previously characterized in terms of time. However, the shape of the waveform, and the location of transitions is different.

Unfortunately, due to time constraints and a bug in the STM32 firmware, we were unable to characterize a relay beyond about 30 cycles, after which the firmware would hardfault. However, the above results show that the Relay Analyzer is at least somewhat accurate, and that accuracy would not change over any number of cycles. Therefore, we are confident that it could measure even small changes in a realys characteristics.

# Conclusion and Recommendations

While this project predominately achieved what it was intended to, there are 2 main components that were unfortunately not finished in time for this report. Firstly, is the relay characteristic database, which had to be dropped in order to ensure the testing functions were fully complete for the deadline. Second are the coil resistance and relay interface boards. The coil resistance circuit would increase precision, but was not required to make the measurements and the interface board made the setup more streamlined and efficient, but was not necessary for function, and required more code to be written. Thus, these were both cut as they needed more testing and verification and to meet time constraints.

Despite being cut from the final working system, both the coil resistance circuit and the relay interface board circuits were designed, PCBs were manufactured and populated, and the PCBs tested and found to be working, as shown in the Prototyping and Testing section of this report.

Lastly, there were three tests that were not included in the firmware and software. These tests were the coil voltage test, the maximum coil turn-off voltage, and the minimum coil turn-on voltage. The coil voltage is something that is written directly on every relay, so it was safe to leave out. The maximum turn-off voltage and minimum turn-on voltage, while important, are relay characteristics that should not be designed near for any application, especially given the changeability in relay characteristics over time.

This project provided a great opportunity to combine many aspects of the ESE program including hardware design, construction and testing of PCBs, firmware development for microcontrollers, software development including GUI design and project management. With the addition of the USB component from the Advanced Elective class, it also provided an opportunity to learn a new communication protocol that is standard in most modern devices. All in all, this project solidified knowledge from previous experiences, and encouraged learning through research, trial and communication.

When taking on a large project like this, it is important to adjust as you go, especially for meeting deadlines. Changing scheduling, tools, and functionality (within reason) were all required as this project evolved. This is a natural part of working on a project, and knowing how to be flexible will create a lot less stress and produce an overall better product. Just remember to document these changes and why they were made.

Some features that could build upon the design in this report include:

- Improve accuracy of measurements
- Include a 4-wire voltage measurement for coil voltage
- Implement full database capabilities
- Design single PCB that combines all PCBs & microcontroller
- Design proper casing for device
    - Hold PCBs
    - Proper connections for device under test
- Widen range of compatible relays

- More user-friendly and feature-rich GUI
- Measure more than one relay at a time
    - Compare multiple relays characteristics

# References

[1] https://www.te.com/usa-en/products/relays-contactors-switches/relays.html

[1] K.C. Yang, “The ins and outs of relay AC performance testing”, *Evaluation Engineering*, Nov. 9, 2018. Accessed on: Feb. 6, 2021. [Online]. Available: https://www.evaluationengineering.com/instrumentation/article/13018718/the-ins-and-outs-of-relay-ac-performance-testing

[2] Matsushita Electric Works, “Relay Technical Information”. Accessed on: Jan. 28, 2021. [Online]. Available: https://media.digikey.com/pdf/Other%20Related%20Documents/Panasonic%20Other%20Doc/Small%20Signal%20Relay%20Techincal%20Info.pdf

[3] Pickering Interfaces, *Life Testing Relays*, Pickering Interfaces, Chelmsford, MA, United States. Accessed on: Jan. 28, 2021. [Online]. Available: https://www.pickeringtest.com/en-ca/kb/hardware-topics/relay-characteristics/life-testing-relays

[4] Applied Relay Testing, *Reflex 10 - Relay Test System*, 27 Cobham Road, Ferndown Industrial Estate, Wimborne, U.K. Accessed on: Jan. 20, 2021. [Online]. Available: http://www.appliedrelaytesting.co.uk/media/datasheets/reflex10.pdf

[5] Applied Relay Testing, *Reflex 51 - Relay Life Test System*, 27 Cobham Road, Ferndown Industrial Estate, Wimborne, U.K. Accessed on: Jan. 20, 2021. [Online]. Available: http://www.appliedrelaytesting.co.uk/media/datasheets/reflex51.pdf

[6] Applied Relay Testing, *Reflex 950 - Filter Connector Test System*, 27 Cobham Road, Ferndown Industrial Estate, Wimborne, U.K. Accessed on: Jan. 20, 2021. [Online]. Available: http://www.appliedrelaytesting.co.uk/media/datasheets/reflex950.pdf

[7] https://temcoindustrial.com/bosch-style-automotive-relay-cn0171-1-qty-12-v-60-80-amp-spdt/

# Appendix A - Code Snippets

## 1. ADC with DMA code

In main.c...

The following is to be placed in USER CODE PD:

```
/* USER CODE BEGIN PD */
#define ADC_BUF_LEN 4096
/* USER CODE END PD */
```

The following is to be placed in USER CODE PV:

```
/* USER CODE BEGIN PV */
uint16_t adc_buf[ADC_BUF_LEN];
/* USER CODE END PV */
```

The following is to be placed in USER CODE PFP:

```
static void MX_USART3_UART_Init(void);
/* USER CODE BEGIN PFP */
void DMATransferComplete(DMA_HandleTypeDef *hdma);
/* USER CODE END PFP */
```

The following is to be placed in USER CODE 2, before the while loop:

```
/* USER CODE BEGIN 2 */
//Start ADC with DMA transfer
HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buf, ADC_BUF_LEN);
/* USER CODE END 2 */
```

## 2. USART code

In main.c...

The following is to be placed before the while loop in main:

```
//USART test
if(HAL_UART_Transmit(&huart3, "test ", 6, 100) != HAL_OK)
    Error_Handler();

HAL_Delay(500);
```

## 3. RTC code

In main.c...

The following is to be placed in USER CODE PFP:

```
84 void setTime (void);
85 void getTime(void);
```

The following is to be placed in USER CODE PV:

```
65 char time[13];
66 char date[6];
```

The following is to be placed before the while loop:

```
128   /* USER CODE BEGIN 2 */
129   setTime();
```

The following is to be placed in the while-loop:

```
155
156       getTime();
157
```

The following is to be placed in the USER CODE 4:

```
705 //Set RTC time
706 void setTime (void){
707     RTC_TimeTypeDef sTime;
708     RTC_DateTypeDef sDate;
709     /*set time*/
710     sTime.Hours = 0x00;         //set hours
711     sTime.Minutes = 0x00;       //set minutes
712     sTime.Seconds = 0x00;       //set seconds
713     sTime.SubSeconds = 0x00;    //set sub seconds
714     sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
715     sTime.StoreOperation = RTC_STOREOPERATION_RESET;
716     if(HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BCD) != HAL_OK)
717         Error_Handler();
718
719     /*set date*/
720     sDate.Date = 0x01;                  //date
721     sDate.Month = RTC_MONTH_JANUARY;    //month (how to set this to current month??)
722     if(HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BCD) != HAL_OK)
723             Error_Handler();
724
725     //assign backup register to maintain time after system reset
726     HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR1, 0x32f2);
727 }
728
729 //get RTC time
730 void getTime(void){
731     RTC_TimeTypeDef gTime;
732     RTC_DateTypeDef gDate;
733     //get current time
734     HAL_RTC_GetTime(&hrtc, &gTime, RTC_FORMAT_BIN);
735     //get current date
736     HAL_RTC_GetDate(&hrtc, &gDate, RTC_FORMAT_BIN);
737     //display time
738     sprintf((char*)time, "%02d:%02d:%02d:%02d", gTime.Hours, gTime.Minutes, gTime.Seconds, gTime.SubSeconds);
739     //display date
740     sprintf((char*)date, "%02d %02d", gDate.Month, gDate.Date);
741 }
```

4. **ADC code**

In main.c...

The following is to be placed in the while-loop:

```
160         //get ADC2 Value
161         HAL_ADC_Start(&hadc2);
162         HAL_ADC_PollForConversion(&hadc2, HAL_MAX_DELAY);
163         reading = HAL_ADC_GetValue(&hadc2);
164
165         //convert adc2 reading to a string and send over uart
166         sprintf(adc2msg, "%hu\r\n", reading);
167         if(HAL_UART_Transmit(&huart3, (uint8_t*)adc2msg, strlen(adc2msg), 100) != HAL_OK)
168                 Error_Handler();
```

### 5. I²C

In main.c...

The following is to be placed before the while loop:

```
107   uint16_t config = 0b0100011100100111;
108   uint8_t SentTable[3] = {0x00, (uint8_t)(config>>8), (uint8_t)config};
109   uint8_t ReceivedTable[2];
```

The following is to be placed in the while-loop:

```
164       if(HAL_I2C_Master_Transmit(&hi2c1, 0x90, SentTable, 3, 50) != HAL_OK)
165                 Error_Handler();
166
167             if(HAL_I2C_Master_Receive(&hi2c1, 0x90, ReceivedTable, 2, 10)!= HAL_OK)
168                 Error_Handler();
169
170             HAL_Delay(800);
```

### 6. SPI

In main.c...

The following is to be placed before the while loop:

```
166   uint8_t SPIData[2] = {0x5A, 0xF0};
```

The following is to be placed in the while-loop:

```
222       //SPI test
223       if(HAL_SPI_Transmit(&hspi2, (uint8_t*)&SPIData, 1, 100) != HAL_OK)
224                 SPI_Error_Handler();
225
226       HAL_Delay(500);
```

# Appendix B - Off-the-Shelf Components

## STM32F767ZI-Nucleo Microcontroller Development Board

life.augmented

**NUCLEO-XXXXZX NUCLEO-XXXXZX-P**
**NUCLEO-XXXXZX-Q**
Data brief

STM32 Nucleo-144 boards

*NUCLEO-H755ZI-Q example. Boards with different references show different layouts. Picture is not contractual.*

| Product status link |
|---|
| NUCLEO-XXXXZX |
| NUCLEO-F207ZG, NUCLEO-F303ZE, NUCLEO-F412ZG, NUCLEO-F413ZH, NUCLEO-F429ZI, NUCLEO-F439ZI, NUCLEO-F446ZE, NUCLEO-F722ZE, NUCLEO-F746ZG, NUCLEO-F756ZG, NUCLEO-F767ZI, NUCLEO-H723ZG, NUCLEO-H743ZI, NUCLEO-H753ZI, NUCLEO-L496ZG, NUCLEO-L4A6ZG, NUCLEO-L4P5ZG, NUCLEO-L4R5ZI. |
| NUCLEO-XXXXZX-P |
| NUCLEO-L496ZG-P, NUCLEO-L4R5ZI-P. |
| NUCLEO-XXXXZX-Q |
| NUCLEO-H745ZI-Q, NUCLEO-H755ZI-Q, NUCLEO-H7A3ZI-Q, NUCLEO-L552ZE-Q. |

arm MBED

### Features

- Common features
  - STM32 microcontroller in LQFP144 package
  - 3 user LEDs
  - 2 user and reset push-buttons
  - 32.768 kHz crystal oscillator
  - Board connectors:
    - SWD
    - ST Zio expansion connector including ARDUINO® Uno V3
    - ST morpho expansion connector
  - Flexible power-supply options: ST-LINK, USB $V_{BUS}$ or external sources
  - On-board ST-LINK debugger/programmer with USB re-enumeration capability: mass storage, Virtual COM port, and debug port
  - Comprehensive free software libraries and examples available with the STM32Cube MCU Package
  - Support of a wide choice of Integrated Development Environments (IDEs) including IAR™, Keil®, and STM32CubeIDE
- Board-specific features
  - External or internal SMPS to generate $V_{core}$ logic supply
  - Ethernet compliant with IEEE-802.3-2002
  - USB OTG full speed or device only
  - Board connectors:
    - USB with Micro-AB or USB Type-C™
    - Ethernet RJ45
  - Arm® Mbed Enabled™ compliant

### Description

The STM32 Nucleo-144 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the internal or external SMPS significantly reduces power consumption in Run mode.

The ST Zio connector, which extends the ARDUINO® Uno V3 connectivity, and the ST morpho headers provide an easy means of expanding the functionality of the Nucleo open development platform with a wide choice of specialized shields.

The STM32 Nucleo-144 board does not require any separate probe as it integrates the ST-LINK debugger/programmer.

The STM32 Nucleo-144 board comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube MCU Package.

TRIAD Magnetics WSU150-1600 AC/DC Power Supply

TRIAD MAGNETICS

**Wall Plug - Ins**

# WSU150-1600


**Electrical Specifications (@25C)**

1. Input Voltage rating: 100-240VAC, 50-60Hz
2. Input Voltage range: 90-264VAC
3. Input current: <0.8A(RMS) @ 115VAC*
4. Max Inrush Current: <50A peak @ 115VAC (Cold start)*
5. Output Voltage: 15VDC
6. Output Current: 1.6A
7. Regulation (line & load): ±5%*
8. Ripple & Noise: 200mVpk-pk Max*
9. No load power (stand by): <100mW*
10. Average Efficiency: ≥86.20%. Meets minimum level VI efficiency.*

**Environmental Specifications**

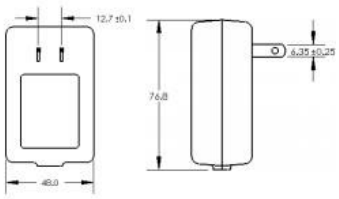
1. Operating Temperature Range: 0°C to +40° C @ full load
2. Storage Temperature Range: -20°C to 60°C
3. Humidity: 5% to 95%, Non-condensing

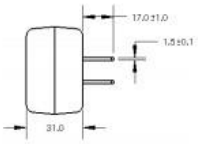
**Reliability Specifications**

1. Leakage Current: <0.25mA (264VAC)
2. Dielectric Strength (Hi-pot): 4242VDC/3secs., 5mA Max
3. Warranty: 5 years

**Mechanical Parameters**

Case Type: Thermoplastic molded enclosure.
Output Cord: 22 AWG, 6 Ft. Long Nom.*



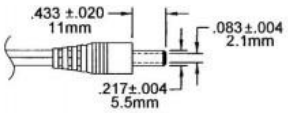




**Safety & EMI**

ETL: 4002961 conforms to UL STD: 60950-1. Certified to CSA.STD C22.2 No. 60950-1 Class II, Double Insulated*
EMI standard: FCC part 15 class B
Over voltage and short circuit protected

Intertek 4002961

FC LPS





Dim.: mm

**EISA 2007/CEC Compliance:** All WSU Power Supplies manufactured after February 10, 2016 will meet the minimum efficiency levels for direct operation as defined by DOE Docket Number EERE-2008-BT-STD-0005-0219. Triad's level VI products will have date code no later than 1605 (YYWW) where 16 is the year and 05 is the 5th week of 2016. In accordance with DOE requirement the label will also contain the Roman numeral VI with a circle.

* These parameters were required to change in order to meet DOE's level VI efficiency requirements.

**RoHS Compliance**: As of manufacturing date February 2016, all standard products meet the requirements of 2015/863/EU, known as the RoHS 3 initiative.

Upon printing, this document is considered "uncontrolled". Please contact Triad Magnetics' website for the most current version.

| | | |
|---|---|---|
| Web: www.TriadMagnetics.com<br>Phone 951-277-0757<br>Fax 951-277-2757 | 460 Harley Knox Blvd.<br>Perris, California 92571 | Publish Date: June 7, 2019 |

# Appendix C - Software APIs & Tools

## Qt

| | |
|---|---|
| **Link** | https://www.qt.io/ |
| **Version used** | 5.12.11 |
| **From the developer** | Qt technology is used by approximately one million developers worldwide. We enable a single software code across all operating systems, platforms and screen types, from desktops and embedded systems to business-critical applications, in-vehicle systems, wearables and mobile devices connected to the Internet of Things. |

## STM32CubeMX

| | |
|---|---|
| **Link** | https://www.st.com/en/development-tools/stm32cubemx.html |
| **Version used** | 6.3 |
| **From the developer** | STM32CubeMX is a graphical tool that allows a very easy configuration of STM32 microcontrollers and microprocessors, as well as the generation of the corresponding initialization C code for the Arm® Cortex®-M core or a partial Linux® Device Tree for Arm® Cortex®-A core, through a step-by-step process. |

## FreeRTOS

| | |
|---|---|
| **Link** | https://www.freertos.org/ |
| **Version used** | Latest STM32CubeMX version included |
| **From the developer** | FreeRTOS is a market-leading real--time operating system (RTOS) for microcontrollers and small microprocessors. Distributed freely under the MIT open source license, FreeRTOS includes a kernel and a growing set of libraries suitable for use across all industry sectors. |

### libusb-win32

| | |
|---|---|
| **Link** | https://sourceforge.net/projects/libusb-win32 |
| **Version used** | 1.2.6.0 |
| **From the developer** | libusb-win32 is a port of the USB library libusb 0.1 (http://sourceforge.net/projects/libusb) to the Microsoft Windows operating systems (Windows 2000, Windows XP, Windows Vista and Windows 7; Windows 98 SE and Windows ME for versions up to 0.1.12.2 ). The library allows user space applications to access many USB device on Windows in a generic way without writing any line of kernel driver code. |

### STM32 USB Firmware

| | |
|---|---|
| **Link** | https://higaski.at/custom-class-for-stm32-usb-device-library/ |
| **From the developer** | Now the curious thing about STM32CubeMX is that you absolutely must pick a device class otherwise the USB library won't be part of code generation. Chances are the class you need is not supported or even if it is, its implementation doesn't do what you want. Now you might get tempted to start with a class similar to your needs and simply edit descriptors, callbacks and so forth till it does work but sadly there are no customization points for editing files like there usually are with STM32CubeMX. So any time you rerun code generation all your changes will be gone. This means that you shouldn't rely on STM32CubeMX to create USB code (apart from hardware initialization) if the functionality you need isn't exactly covered by one of the selectable classes. The good news is that the device library itself is rather extensible and at least mediocre documented. |

# Included Files

This is a list of included files that could not be included in this document.

| ***Description*** | ***Filename or directory name*** |
|---|---|
| *PCB design files, test files & images, and Altium Designer raw files* | */PCBs and Circuits* |
| *Software GUI code* | */Software/relayAnalyzer* |
| *STM32CubeMXIDE Firmware code* | */Firmware/Relay_Analyzer* |