STM32 Based Data Acquisition System

Interdisciplinary Capstone Design

Automation Solutions

Report #3, 4/25/2024

Authors: Jack Hanling (ME), Cooper Allen (ME), Christian Busch (ME, PHYS)
Michael McCabe (ME), Mark Stevens (ME), Arsh Suri (BME, CS)
Professor: Dr. David MacNair

i. Executive Summary

Due to the phasing out of current educational data acquisition devices (DAQ), namely the ELVIS III by National Instruments (NI), the Georgia Tech Mechanical Engineering Instructional Laboratories (ME 3057 Experimental Methods and ME 4056 Systems Lab) seek alternative solutions to their current DAQs. While competing products are currently available that could achieve the same result as the ELVIS III board, it was proposed, in the interest of reduced cost and enhanced user-serviceability, that the new DAQ system be developed in-house. The proposed DAQ is to be based on an STM32 microcontroller and fully compatible with the current experimental ecosystem. To accomplish this, the project is organized into three main subsystems, focused on the DAQ hardware and software. the post-DAQ software - including the GUI and database - and a sensor/actuator-equipped testing and validation platform, respectively.

The process of changing the DAQ from the NI ELVIS III to an STM32 based microcontroller presents several challenges. The new DAQ must contain 10 analog inputs, 8 digital input/output pins, two pulse width modulation (PWM) signal generators, two encoder inputs, two function generator outputs, and two Inter-Integrated Circuit (I2C) inputs. These inputs and outputs are needed so that all current labs (and possible future labs) are compatible with the new DAQ without having to modify existing experimental setups. The DAQ must also be able to read and process the input signals quickly, so that the data is seen by the student in semi real time. The DAQ must have the ability to output a signal so that specified parameters of an experiment can be modified by the student.

In order to accommodate all of the different input signals and protect the STM microcontroller from damage, several different conditioning and protection circuits were designed. In particular the circuit for a 0-3.3V input was designed, built, and tested on a custom-made PCB board. This PCB board includes all the relevant circuitry and pins outs needed for the STM microcontroller, as well as giving a base for the BNC input board to sit on. This entire BNC, PCB and microcontroller combination will be protected by a 3D printed case that will be mounted on the RC Car validation platform.

The STM microcontroller must be able to read real-time data from multiple different input sources and be able to package and export the via ethernet in as little time as possible. The M4 core on the STM will be responsible for taking the HSEM and run through an RTOS (FreeRTOS API) routine and save it SRAM. After a set amount of time, the M4 will give the HSEM to the M7 core for packaging and exporting the data over the ethernet port.

Once the ethernet packets are exported from the M7 to a cloud database. The cloud database will then send the data to the graphical interface that the user accesses on their local machine. To achieve low latency, it was decided to run the two loops, i.e. ethernet to database and database to the graphical interface simultaneously. The database's architecture is crucial to facilitate efficient data transfer. Although the database stores individual student's data, students work within groups for each lab, which means lab data must correspond directly to groups rather than individual users. With that core factor in mind, a database architecture with a core data table that would have groups as the primary key was established, which allows them to directly reference in independent lab tables without data repetition as a foreign key. Each student group and professor group must have different viewing privileges within the GUI, so to manage these various permissions, a JSON table will also be added to include permissions for group categories. Finally, with this database architecture, a basic real-time plotting capability was developed that also allows for modularity between different lab setups.

In order to validate our design decisions, several prototypes were used for the different parts of this report. For the circuit and PCB testing, LTSpice models were first used to simulate the circuits and generate design concepts. These circuits were then built on a breadboard and physically tested. Finally, the PCB design itself was tested after it had been printed. The STM microcontroller was tested iteratively as well and basic prototypes of the firmware have been validated. It's ability to read and write and export data was verified and tested. This is the same case for the GUI interface and cloud code as well. In order to test the real time plotting ability, a python script was used to simulate random ethernet inputs.

Significant progress was made in validating the STM32-based DAQ system through iterative hardware and software development. The 0–3.3V analog and digital input circuitry is fully operational, successfully interfacing with the STM32 and meeting required signal fidelity. Digital outputs, function generator, and PWM output functionalities have also been verified, with adjustable amplitude and frequency working as intended.

The ±10V analog signal path has undergone preliminary testing, with basic functionality confirmed; however, further tuning is needed to ensure reliable operation across all input conditions. Similarly, encoder inputs have demonstrated partial success and are expected to be functional with minor adjustments to the firmware or conditioning circuit.

Conversely, I2C input remains the most challenging subsystem, currently requiring substantial development and debugging to achieve reliable communication. Despite this, the team has built a robust foundation with a working real-time data pipeline and modular architecture that can support future integration of the remaining inputs.

Overall, the DAQ system has reached a stable intermediate stage where core functionality is validated, and outstanding issues have clear pathways forward. With continued refinement, this system remains on track to meet all initial design specifications and support the full suite of ME 3057 and 4056 lab experiments.

ii. Nomenclature

DAQ: Data Acquisition System
GUI: Graphical User Interface
UI: User Interface
STM32: Microcontroller. Specifically, STM32H755ZI-Q
PWM: Pulse Width Modulation
ABET: Accreditation Board for Engineering and Technology
NI: National Instruments
I2C: Inter-Integrated Circuit
I/O: Input/Output
Op-Amp: Operational Amplifier
PCB: Printed Circuit Board
RTOS: Real Time Operating System
HSEM: Hardware Semaphores
DMA: Direct Memory Access
SRAM: Static Random Access Memory
IP: Internet Protocol
TCP: Transmission Control Protocol
MII: Media Independent Interface
PTD: Partial Tandem Duplication
MAC: Medium Access Control
JSON: JavaScript Object Notation
RC: Remote Control
FPGA: Field-Programmable Gate Array

I. Introduction and Background

To reduce reliance on deprecated 3rd-party educational tools, the ME 4056 Systems Laboratory at Georgia Tech's Woodruff School of Mechanical Engineering has need of a new, independently maintainable data-acquisition board (DAQ) based on the STM32 microcontroller, to facilitate education about the application of sensors and actuators to the processes of system identification and data extraction from physical systems. While the DAQ itself is the primary deliverable, a standalone testbed is to be developed in parallel as a platform with which to validate the system's ability to write instructions to actuators and display data from sensors, both analog and digital, across multiple dissimilar channels in synchronous real-time on the web. In large part, the role of the DAQ is to "translate" data and instructions between the student and the in-system sensors and actuators. As the student is not expected to directly program the microcontroller on which the DAQ is based, the scope of the project must also encompass the secure communication between the DAQ and an external web server, as well as the foundation of an interactive cloud-based data management dashboard from which students and instructors will be able to control the parameters of a system and access collected data from a specified lab station.

Thus, the product must address every step in the process of transferring a quantifiable system value – a temperature, for example – to a datum or time-series dataset in a user interface, and, in the opposite direction, the communication of user-defined parameters – motor velocity, for example – to the appropriate actuators, and is as such divided into three primary subsystems. These are the testbed, which stands in for existing and future ME 4056 lab set-ups; the data-acquisition board (DAQ), which communicates between mechatronic components and institute servers; and the off-DAQ software, namely secure DAQ-to-server communication, cloud systems, and a GUI.

II. Existing Products, Prior Art, and Applicable Patents

The current market for engineering laboratory equipment is very broad and there are a wide variety of companies that offer equipment, digital acquisition (DAQ) devices, and computer software to help facilitate engineering laboratory education. One of the most prolific companies is National Instruments, which sells a variety of DAQs, including the myDAQ, myRIO, and NI ELVIS, for different levels of labs and students [1]. Additionally, National Instruments provides a wide range of software and drivers such as LabVIEW to use with their own and third-party measurement devices [2]. However, as of April 2025, National Instruments is phasing out most of its engineering education equipment, which has left universities seeking alternatives to National Instruments for DAQs and software.

There are some alternative companies offering DAQs specifically made for engineering education. National Instruments itself created a new branch called Digilent that is specifically geared towards making DAQs for engineering education. Digilent provides equipment similar to the older myDAQ, myRIO, and NI ELVIS, at a lower cost, and including compatibility with LabVIEW and MATLAB [3]. Keysight is another company that caters to engineering laboratories by offering a wide variety of DAQs for teaching purposes. However, most of their equipment is designed to work with their proprietary software, BenchVue [4], and offers little interoperability. Tektronix also offers general purpose DAQ equipment through its Keithley line, that can be used for teaching purposes as well, although for the system, multiple different units may have to be bought [5]. A common solution for engineering teaching labs and labs in general is also to use an in-house DAQ made from microcontrollers and in-house circuits. Many teaching labs do this, as it can be cheaper than conventional teaching DAQs and can be tailored to work better with the sensors and equipment at hand.

Software products enabling visualization and processing of collected data include LabVIEW, MATLAB, and BenchVue. National Instruments regularly updates LabVIEW, and all of its instruments and DAQs work with it. In addition, there are many tutorials from National Instruments and others that allow for ease of use [6]. Due to this wide availability and access, many instruments and DAQs have some level of LabVIEW support, whether through pre-made packages or drivers. Other software packages include MATLAB and Arduino packages, which also have the ability to communicate with LabVIEW[7,8]. Another software, BenchVue, is also available, but is less supported and primarily made to work with Keysight equipment [9].

III. Market Research/Potential Impact

The potential demand for an STM-based DAQ and software based on inexpensive, open-source materials and software is immense. There are 4773 undergraduate programs currently accredited by the Accreditation Board for Engineering and Technology (ABET) [10]. Every engineering program accredited by ABET is required to offer some form of engineering laboratory experience in its curriculum, so the potential for a low-cost, user-friendly DAQ for lab use is very wide, especially at smaller universities with restrictive laboratory budgets. Even at larger universities with greater financial liberty, the customizable nature of the DAQ allows for more advanced purpose-built labs to be created, facilitating greater institutional control over curriculum.

Beyond the educational engineering laboratory environment, there are many public and private engineering labs that could benefit from a customizable, user-friendly DAQ and accompanying software, saving time, effort, and money in performing unique and varied testing arrangements and procedures.

Non-engineering labs, like those focused on research or natural sciences, can make use of cheap and easily customizable DAQs. Many science programs at universities have low student volume, which can make it difficult to justify investing in the expensive lab equipment currently available, especially in undergraduate laboratories. A cheaper DAQ offering enables allocation of funds towards more advanced and varied experiments in this setting.

IV. Customer Requirements and Engineering Design Specifications

Required Functions

Required functions of the STM32-based DAQ derived from discussion with the sponsor are outlined in the function trees located in Appendix A. At a high level, this project can be broken into five main categories: Data Acquisition and Processing, System Communication and Control, Power Management and Protection, Mechanical and Hardware Design, and Verification/Testing.

The Data Acquisition and Processing function tree breaks down the main task of the STM32. Signals will be input into the STM32, and the STM32 must accept, process, and then export these signals through ethernet to the cloud database.

One of the most expensive components in the DAQ will be the STM32 microcontroller. It is paramount that this microcontroller is protected from excessive voltages and currents to ensure its longevity. The Power Management and Protection function tree breaks down the functions required for the power supply going directly to the STM32 microcontroller, and the current and voltage protection circuit that resides in-between the STM I/O pins and the exterior BNC connectors.

The STM design must be structurally stable, easily manufactured, and easily repaired. The Mechanical and Hardware Design Function tree breaks down the components of ensuring a robust design. The STM32-based DAQ will be enclosed in a 3D printed PLA case, allowing for easy manufacturing. The Gerber files will be provided for the PCB fabrication, allowing for easy fabrication of the electrical components whether made in-house or outsourced.

Another required function is that the STM32 based DAQ system interfaces with a cloud-based Graphical User Interface (GUI). This cloud-based GUI will serve as a web-accessible dashboard to interact with data acquired from the various sensors from the STM32-based DAQ. While the DAQ system must be connected to ethernet to our web servers, for use of students and the minimization lab preparation time, the GUI must be accessible from any Wi-Fi-enabled device through the cloud. The nature of labs also requires that the GUI must facilitate user input back to the DAQ system and Verification Apparatus, which often will be in the form of parameters for lab-related settings. Both data display as well as user input, must be fully synchronous and occur real-time. Cloud based GUI is the end goal, although due to time constraints of this project, the scope was reduced to USB communication. The cloud-based GUI will be emulated through the USB data being send directly to an individual's laptop, with the goal of preparing a prototype of the device that can later integrate ethernet and true cloud based GUI.

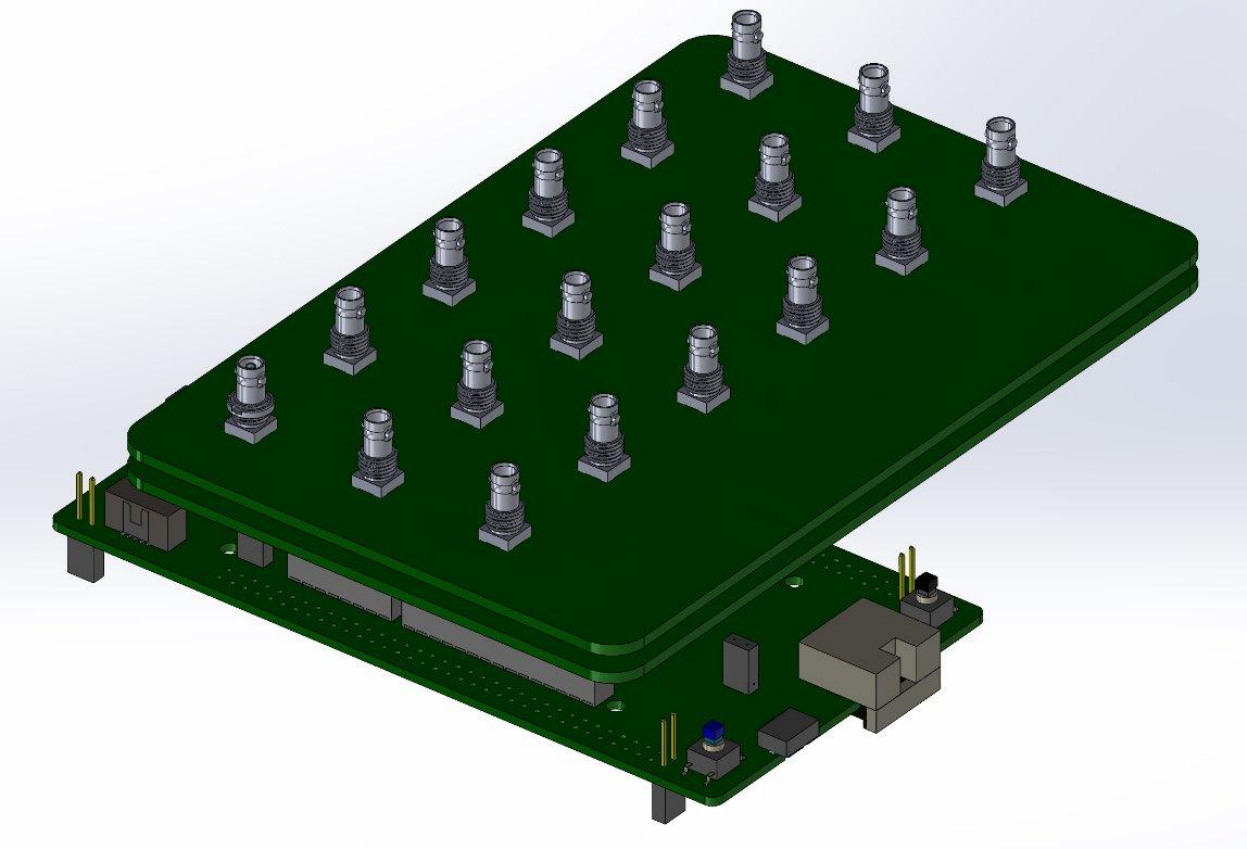


Figure 1: STM-based DAQ Model Initial Ideation (Internal Components)

Verification of the functionality of the STM32-based DAQ is necessary to ensure robustness. The Verification and Testing Function Tree breaks down the testing procedure of the DAQ system. Hardware will be analyzed by characterizing the signal processing and protection capabilities of the Protector PCB. Software will be verified by account of accuracy of data, efficiency of data acquisition, processing, packaging, and transmission to the cloud, as well as latency for the full data display and processing loop to occur.

Design Considerations

Several design options were considered to fulfill the required functions of the STM32-Based DAQ system. The design considerations are outlined in Table 1.

Table 1: Design Considerations

Category	*Potential Solution 1*	*Potential Solution 2*	*Potential Solution 3*
Microcontroller	Off the shelf DAQ	Raspberry Pi	STM32 Microcontroller
Hardware	BNC Connections	Screw Terminals	Combination with Two Interfaces
Software	Cloud via Ethernet	On Computer via USB	
Voltage Conversion	DC-DC Converters	Zener Diodes	
Current Protection	Large Series Resistors	Operational Amplifiers	
Testing Apparatus	F1 RC Car	Current ME4056 Experiment	New ME4056 Experiment

In terms of the microcontroller, options included using another off-the-shelf DAQ, the STM32 Microcontroller, or Raspberry Pi. Many companies make DAQs for engineering and instructional labs. However, the downside of these DAQs is the cost is higher, and they are less customizable for the specific lab uses. The Raspberry Pi has ethernet cables, along with numerous other USB input ports, and comes at a relatively cheap price, but lacks built in Analog to Digital Converters (ADC). The STM32 has three 16-bit resolution, high speed ADCs built in, with 10+ analog input ports. This simplifies the hardware substantially and simplifies the data collection, sorting, and transmission over ethernet. The conclusion was to use the STM32H755ZI-Q microcontroller.

The current hardware of the STM32 interface is BNC connectors. Another consideration was using screw terminals allowing for easier integration of the DAQ into the Testing Apparatus. The conclusion was to use the BNC as the main exterior interface, to allow the easiest integration into the current Systems Lab setup. A side panel with screw terminals will be provided which allows for easy integration into the Testing Apparatus.

For software, the team considered interfacing with the cloud-based GUI with a USB port that could directly connect to any user's laptop USB port. The USB port had the advantage of being simpler. Data can be sent directly to the users computer, and the software can read this data and put it into a GUI. The client exhibited interest in the ethernet connection. Although more complicated, it had the potential of allowing anyone to access the DAQ GUI though the cloud. In addition, having the data transferred through an online database allows for a digital copy to be retained for instructors, and provides additional security benefits. It was decided, as an overarching goal of the project to use the ethernet and the cloud-based GUI. For purposes of this first stage, semester long project, the USB port would be utilized to 'emulate' the ethernet data transmission.

To protect the microcontroller from current surge, both large resisters or operational amplifiers were considered. Operational Amplifiers exhibited the same benefit of high impedance as a powerful resistor but also can be set up with feedback mechanisms to step up or step down the voltage, which will be useful for implementing the 0-3.3V and +/- 10 V inputs and outputs. For this reason, operational amplifiers were used to limit current to input and output ports.

Finally, a means of verifying the STM-based DAQs functionality for quality control, a test bed is needed. Options considered were using a current ME 4056 lab setup, working on developing a new ME 4056 lab setup or using an RC car with sensors installed to test the DAQ. The 1/10th scale RC Car was selected for its secondary use of providing the RoboJackets Team

with real-time telemetry data, thereby expanding the use of this DAQ and proving its general use.

Specifications

Meetings with the sponsor yielded the needs noted in the Specification Sheet in Appendix B, and the User Needs, Specifications, and Constraints Table below. The new STM-based DAQ system must be compatible with the current ME 4056 experimental test setup. High level requirements for compatibility were defined as a need for 10 analog inputs, 8 digital input/output pins, two pulse with modulation (PWM) signal generators, two encoder inputs, two function generator outputs, and two I2C inputs.

Table 2: Needs, Functions, and Constraints

DAQ Subsystem

Needs	Functions	Constraints
10x Analog Input Pins	>35 kHz Sampling Frequency	Input must only be a reference voltage
	16 bit ADC resolution	Voltage must be within 0-4 V
	Capacity for all 10 to be on at once	
	0-3.3V maximum input	
8x Digital Input/Output Pins	Voltage Switch Speeds up to 5 kHz	Input/Output must be a refence voltage
	Provide Constant Voltage Outputs	Voltage must be within 0-5V
	Read Voltage Inputs	
2x PWM Outputs	Generate Output PWM Signals	Output Must be a Reference Voltage
	Duty Cycle from 0 - 100%	Output Voltage Between 0-5 V
	Frequencies up to 50 kHz	
2x Encoder Inputs	Handles up to 50 kHz PWM Frequency Input	Input must only be a reference voltage
		Voltage must be within 0-5 V
2x Function Generator Outputs	Produce Signal Frequencies up to 50 kHz	Output must be a reference voltage
	12 bit DAC resolution	Voltage must be within 0-5 V
2x I2C Inputs	Baud Rate up to 115200 bps	Output must be a reference voltage
		Voltage must be within 0-5 V
Overall Constraints		Maximum Output Current = 20 mA
		Maximum Total Output Current = 140 mA
		Maximum Analog Voltage Input 4 V
		Maximum Digital Voltage I/O 5V
		Maximum Total Injected Current = +/- 25 mA

Discussions with the client concluded that each analog input pin will be required to reach up to 35 kHz sampling frequency. Accuracy of data acquisition was also a topic of interest. High resolution data was considered optimal, and therefore 16-bit ADC resolution for the analog inputs will be used.

The digital Input/Output pins are needed for reading voltage High (5V to 10V)/Low (0V to 5V) signals, and outputting voltage High/Low Signals. Voltage switch speed up to 2 kHz was decided to be sufficient. Two PWM output signals were needed, with the duty cycle reaching between 0 – 100% duty cycle and frequencies up to 50 kHz.

Encoders are required for the internal combustion engine timing and require up to 50 kHz PWM frequency. Two function generators are also introduced into the system, so that varying voltage signals can be sent out to experimental equipment. Producing signal frequencies of up to

50 kHz is sufficient for the signal conditioning lab, and a 12 bit Digital to Analog Converter (DAC) offers sufficient resolution.

Two I2C inputs were likewise required for the current lab setup. It must be able to accept a baud rate up to 115200 bits per second (bps). As seen in Figure 1, the STM32 microcontroller will be located at the bottom, with a PCB plate situated on top taking care of the signal processing and circuit protection. Another PCB plate is applied on top of this with BNC connectors and pins (omitted) for 0-3.3V and +/-10V input/outputs. This entire device will be encased in 3D printed PLA, with an ethernet port accessible to the outside, and a power on button located on the top.

An additional layer of protection is considered in the design of the signal processing board. A breakdown of their functions are in Figures 2 and 3. A Zener diode with a 3.3V and 10V breakdown voltage will direct any excess voltage (ex. 12V power source), with a resistor to ground. This will allow for safe redirection of excessive voltage, protecting the circuit components from high voltages and power that may result in their destruction, in addition to providing an additional layer of protection to the STM32 microcontroller.

The current laboratory setup has +/-10V input and output signals. Although the newer standard of 0-3.3V has become a more prominent standard with sensors and actuators. In addition, the STM microcontroller we use has a threshold of 0-3.3 V for digital Input/Output, and 0-3.3V for analog inputs, which would mean better compatibility. It was concluded that the DAQ must be compatible with both types of inputs, to facilitate the transition between the older +/-10 V to the 0-3.3 V standards. Each input BNC will have an associated jumper, which will allow for the user to select 0-3.3 V and +/-10 V input types.

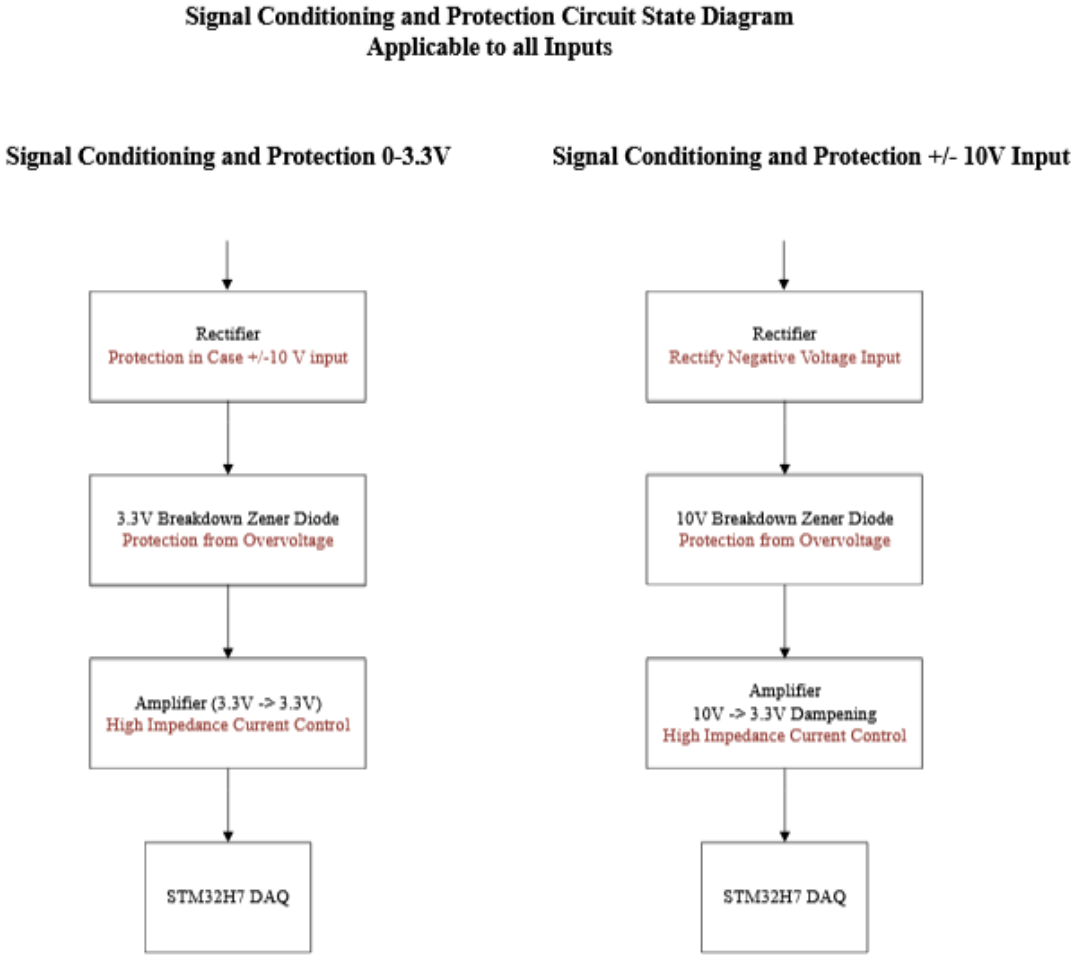


Figure 2: Signal Conditioning and Protection State Diagram Applicable to All Inputs

Another major specification is the power source and power requirements for the STM32 microcontroller. As detailed in the Specification Sheet, the power source provides 12 V, and the STM32 requires 8V and 400 mA for its power input. A DC-DC converter will be inserted to regulate voltage, and the current limiter IC/Circuit will be used to limit the current to 400 mA. A fuse will be used before the STM microcontroller which will protect it in the case of a current surge.

To maintain consistency with the BNC connections used in the current lab setup, the top plate of the STM based DAQ system will have a similar exterior plate to the currently used NI DAQ with BNC connectors for outputs and inputs. A PCB plate on the top will hold the BNC connectors and jumpers for switching between 0-3.3V and +/- 10V input types. Another board will be placed below it containing the signal conditioning circuitry and current/voltage limiting circuitry. Finally, below this PCB plate, the STM will connect to all required inputs/outputs.

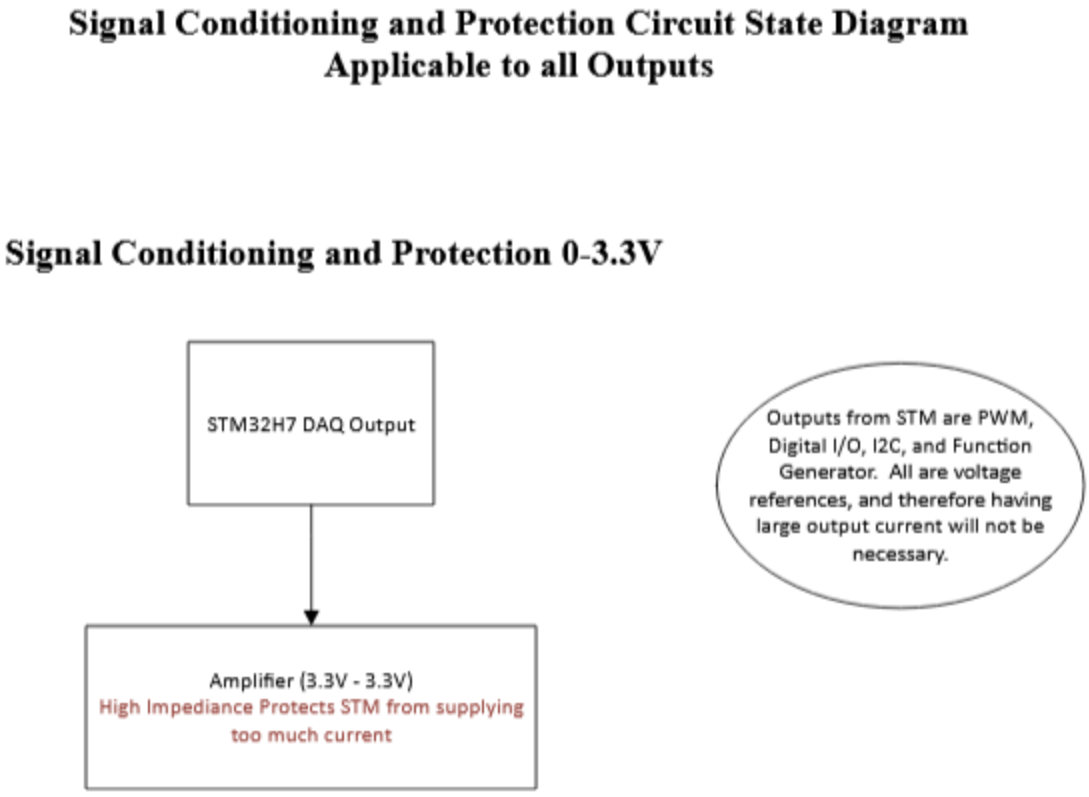


Figure 3: Signal Conditioning and Protection State Diagram Applicable to All Outputs

The exterior of this case will be made out of PLA from 3D printing. This is consistent with the goal of 'ease of manufacturing' and 'ease of repair'.

Finally, a sponsor requirement is that the STM32 based DAQ system interfaces with a cloud-based Graphical User Interface. Data transfer between the DAQ and cloud GUI is additionally required to be through the ethernet, allowing for safe transmission of the data without possible tampering, with backup copies of the data being contained within an online database.

The customer also requires safety measures to protect the DAQ system in case of accidental misuse. A primary concern was someone connecting the power to one of the input or output pins on the DAQ system. With a power source with 12 V, unprotected contact with one of the STM microcontroller pins will almost certainly result in an overdraw of current, resulting in the destruction of the microcontroller.

Validation of the DAQ system will be conducted using a $1/10^{th}$ scale car as a sensor testbed. For the DAQ to work effectively in the ME 3057 and ME 4056 lab environments, it must be able to execute collection of all the aforementioned inputs, display them in real time, and be sufficiently robust to handle the abuse imposed by its use in a student lab. Using the DAQ system aboard a 1/10th scale car to collect data needed to optimize lap time will simulate all of these requirements.

When optimizing the lap time of a car, sensors like the temperature and displacement sensors have an output voltage of 0-3.3V, the encoders use PWM, and the IMU uses I2C. This will replicate all the data types used in the ME labs and provide a benchmark for the validity of

the DAQ system. In addition to data acquisition, the system must also provide inputs to the test equipment. In the ME labs this normally involves control of motor speed. To replicate this the DAQ system will be required to adjust the signal input into the car speed controller and modulate the operational state of all moving components by passing all control inputs through the system.

The Software also has specifications for completely facilitating client requirements. An online cloud-based database that can handle JSON data, the format standard for web communication, must be integrated within the software. This database must be able to accommodate various sensor types through time and have an architecture to support the fetching of different students' data with high efficiency, as well as providing user input back to the STM-based DAQ. Additionally, as the ethernet transfer will provide only voltage values for analog pins, the server is required to have a level of signal processing to convert those outputs into sensible values from the student perspective. To facilitate communication between the back-end server framework and the cloud, an API interface will have to be constructed.

Specifications Importance and Correlations

A House of Quality for the DAQ system is provided in Appendix C. Customer requirements are broken into the general groups of safety, signals, hardware interface, and ethernet. Safety concerns mainly the power supply, voltage I/O, current I/O, and rectifying negative voltages. These are placed at high priority weight of 9 due to the necessity of safe power electronics and to protect the STM32 microcontroller, one of the more expensive and sensitive components of the DAQ.

Signals are placed at a priority weight of 8. These signals are specific requirements from the client, and what is necessary for the DAQ to be successfully implemented into the current ME 4056 lab set-up. Ethernet is tied with a priority weight of 8. Ethernet was strongly recommended by the client and is therefore a high priority on our list. Ensuring smooth and robust data transfer from the microcontroller, through ethernet, and to the UI is essential for functionality of the STM-based DAQ system. The hardware interface is placed at a priority weight of 6-7, as they are essential for ergonomic design of the DAQ.

The power protection circuits are all correlated in their goal to protect the STM32 microcontroller. The Schottky diode, Zener diode, and amplifiers condition the signal to fall within the 0-3.3V input and current limit of the STM microcontroller. This is needed to ensure a robust DAQ design. The BNC connectors and the screw terminals have a slight conflict, as they are both hardware interfaces, but cannot be placed in the same place on the DAQ exterior. For ergonomic purposes, the BNC connectors will reside on top of the DAQ system, where they can be spread out and easily accessible for the System Lab. Since the screw terminal is considered an 'additional' connection for the RC Car Testbed, it will reside on the side, where it can still be accessed for different applications requiring wire in and out, extending the compatibility of the DAQ system to more experimental setups.

V. Design Concept Ideation and Preliminary Selection Justification

The sponsor has already made some key decisions regarding the DAQ system, including the selection of the STM32 for the microcontroller. This platform meets the project's requirements, offering sufficient clock speed for multi-channel real-time I/O, ample analog and digital I/O pins, integrated Ethernet for communication, and security features such as an encryption engine. Communication between the DAQ and backend servers is mandated to occur via Ethernet, ensuring centralized data storage, academic integrity enforcement, and accessibility for students and instructors. USB and wireless alternatives, such as Bluetooth, were deemed unsuitable due to security and integrity concerns.

To support a wide range of sensors, signal conditioning circuitry is required. Existing lab systems condition sensor signals to a ±10V range, but the STM32H7 requires 0-3.3V inputs. Given industry trends favoring 03.3V and the sponsor's preference, this voltage range has been standardized for the DAQ. Future work will be done to implement a conditioning circuit that can convert the +/- 10V input to the acceptable 0-3.3V input range.

Data acquisition for development considered three options: modifying existing lab setups, a standalone digital debugging tool, or a 1/10th RC car testbed. Existing lab equipment was ruled out due to limited access and insufficient complexity, while the RC car was chosen over the digital debugger to ensure real sensor data capture. Additionally, team expertise and interest in RC platforms made it a practical choice.

A robust electrical connection is necessary for sensor integration. While the STM32 supports jumper wire connections, they are too fragile for lab use. BNC connectors were chosen for durability and compatibility with lab equipment, though they are impractical for the RC testbed. Instead, jumper wires with screw terminals will be used for development.

This approach balances system compatibility, robustness, and practicality, ensuring an effective DAQ solution for both lab and development environments.

For the software to facilitate communication from the DAQ to the GUI, several key decisions needed to be made to manage project scope. Full integration with servers will require several weeks of testing and verification, subtracting from total feature development time. As such, communication to the back-end server will occur through using a Docker Container, which can simulate our cybersecurity requirements and functions, rather physical server architecture [12]. This docker container will contain the online database and will be contained on one local device which will be connected to ethernet, which makes development more rapid and flexible.

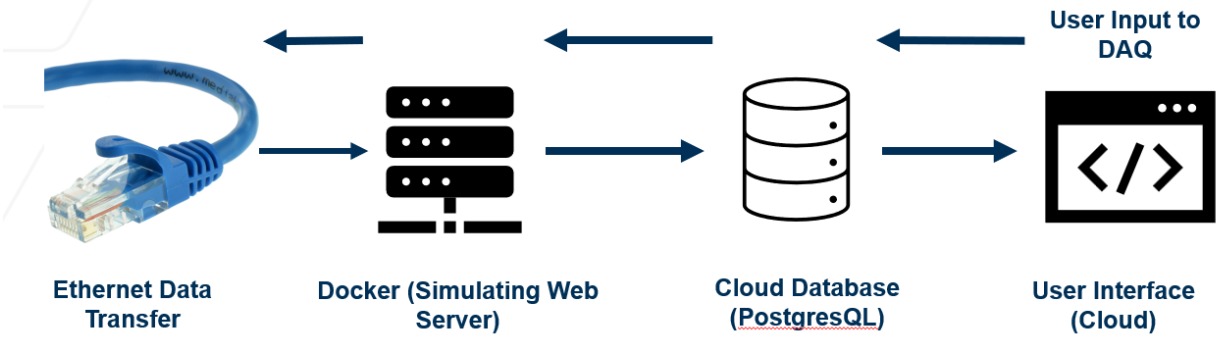


Figure 4: Preliminary Software Pipeline for Cloud-based GUI

With those scope considerations in mind, the overall software pipeline follows the path in Figure 4. The ethernet will transfer to the Docker Container, the simulated web server, which will contain the cloud database, which will store student lab data. This cloud database will be accessible to all teaching staff and will handle data tampering issues as all copies of data will be uploaded to this database. This database will be created with PostgreSQL, due to its synergy with our back-end framework Django, which is an industry standard for facilitating quick-turnaround for small development teams [13]. A MongoDB noSQL database was also considered due to its relative ease in representing complex student group data relationships, however as it lacked compatibility with Django, which would speed up development turnaround, it was not selected. The PostgreSQL will then communicate a GUI hosted within the docker container. User input on the GUI will follow this pipeline back to the ethernet to modify parameters on the DAQ.

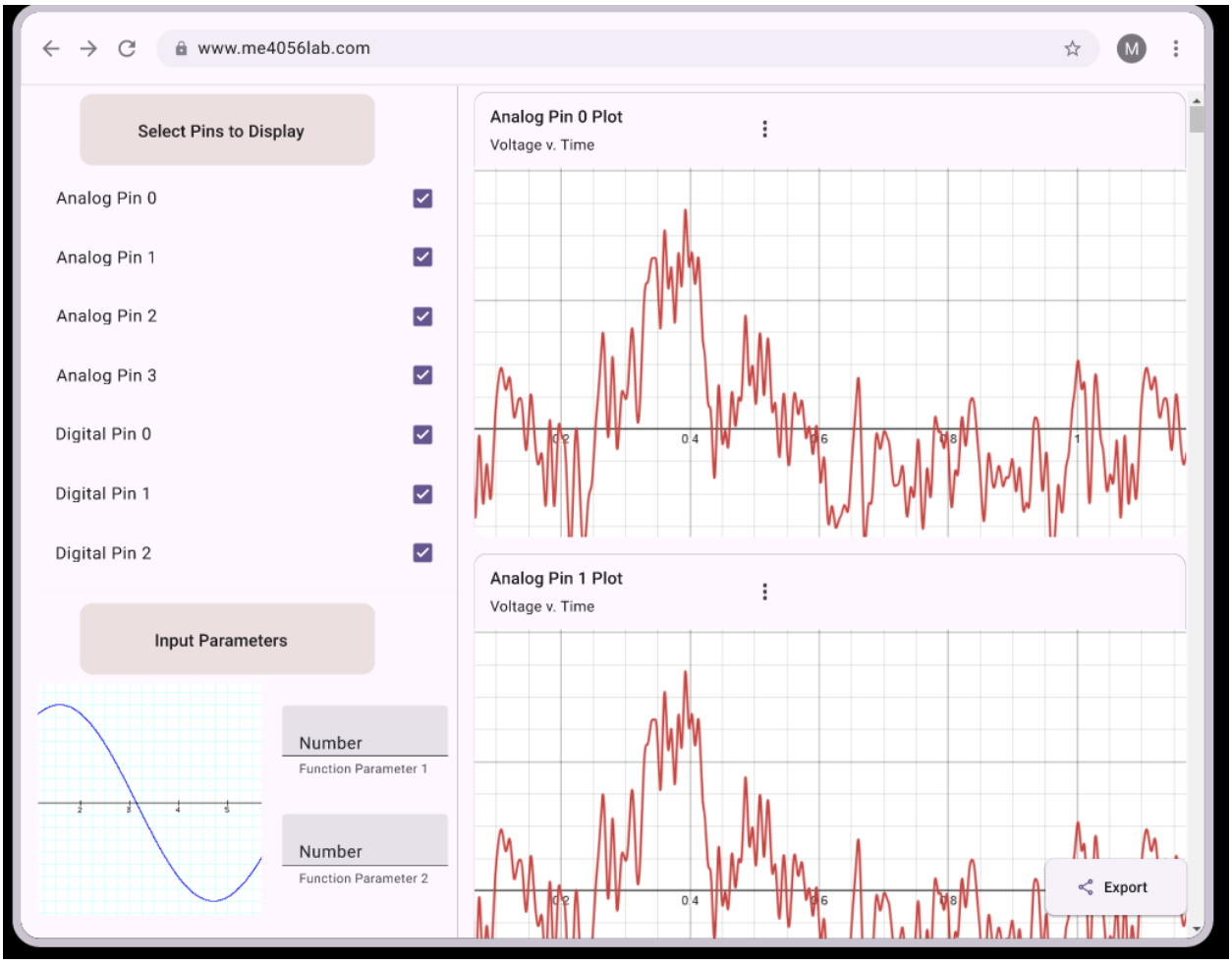


Figure 5: Preliminary Design Concept for GUI

The initial design concept for the cloud-based GUI from the user perspective is displayed in Figure 5. There are user input options to select which pins to display, as all pins may not be needed. Additionally, lab parameters may be controlled by the user. Then, all sensor voltage or measurement values will be plotted separately and may each be modified independently by the

user for different plot formats. On the bottom right, the user can export all their data to an Excel sheet, which they will then use for further lab activities.

VI. Design Features Selection and Justification

An exploded view of the STM32-Based DAQ system is presented below in Figure 6. The main hardware features are the Protector PCB Circuitry for protecting the STM32 microcontroller, the corresponding PCB Design, and the User Interface/Exterior.

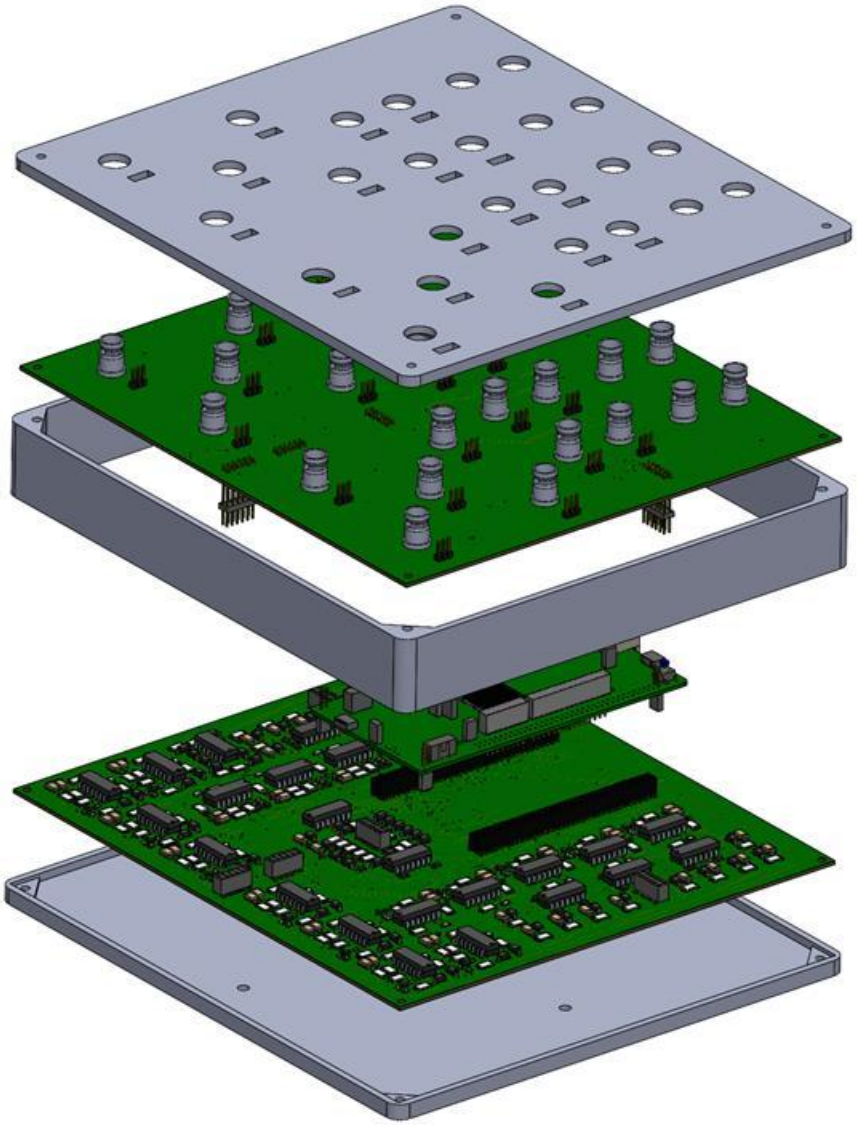


Figure 6: STM32-Based DAQ Exploded View

The software features begin with the STM32 Pinout Configuration as shown in Figure 7. It then moves into a Real Time Operating System (RTOS) for handling simultaneous

requirements, direct memory access (DMA) for accelerated data transfer, hardware semaphores (HSEM) for cross core communication, Ethernet data packaging and transmission, and finally a cloud-based Graphical User Interface (GUI) on the front end.

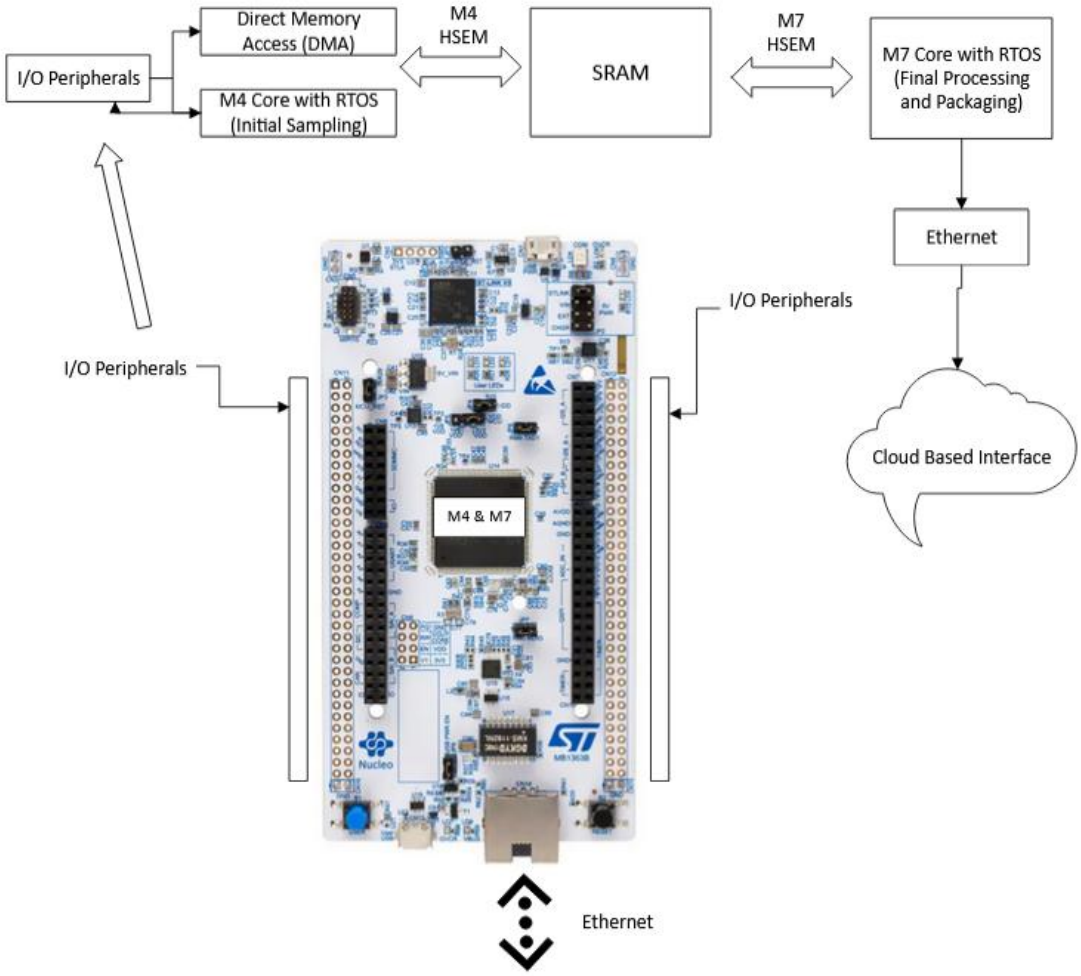


Figure 7: STM32 Software State Diagram

Protector PCB Circuitry: General BNC Inputs

Most signals that will come into the microcontroller will be in the 0-3.3V so that the microcontroller does not get damaged in the process of reading the sensor values. However, in the process of normal use, unexpectedly high voltages may occur. For example, a student may inadvertently plug in a ± 15V voltage supply into an analog channel. Without proper protection, this could severely damage the microcontroller. To protect the microcontroller from such high voltages, several circuit elements were used.

The complete circuit diagram is shown in Figure 8. The voltage input from the sensor comes in from the left and is separated from the two diodes by a 2k resistor. This ensures that the voltage between the source and diodes are separated, otherwise the diodes and source voltages would interfere with one another. This, however, creates a voltage drop from the original signal, which is accounted for by adding a resistor across the op-amp to bias the output voltage back to

the correct level. Interestingly, this feedback resistor was found to have no effect when used with the LM234 op amp, and therefore was dropped in the final circuit design. In between the 2k resistor and op-amp are two different diodes: a Zener with a 3.3V breakdown voltage, which limits the voltage to 3.3V to prevent damage to microcontroller, and a Schottky diode, which better reduces voltages below 0 V, due to its low forward voltage. These, in conjunction with each other, prevent damage to the microcontroller. On the same node, there is also a 1nF capacitor whose purpose, together with the 2k resistor, is to serve as a passive, first-order low-pass filter with a cutoff frequency of ~ 80 kHz. This is integrated to attenuate high frequency (MHz range) noise which might otherwise interrupt the operational amplifier (op-amp). The last part of the circuit is the op-amp, which serves as another layer of protection in case the input current gets too high. It is powered by the +12V supply and is grounded on the negative terminal.

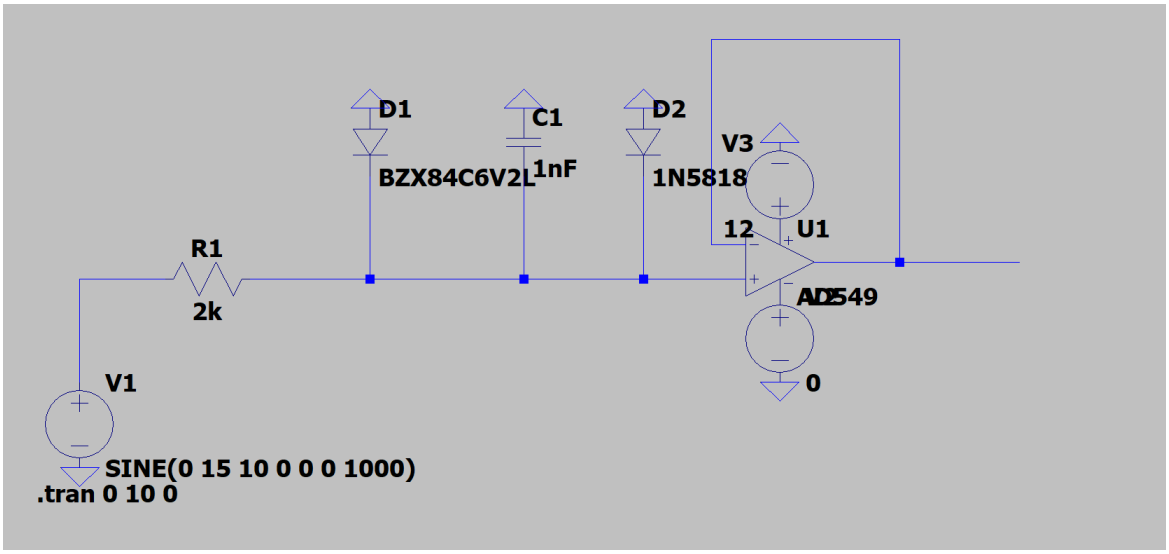


Figure 8: The Circuit Diagram for the 0-3.3V Conditioning and Protection Circuit

In order to verify the correct operation of this circuit, it was first simulated using LTSpice, whose circuit is shown in Figure 9.B. It was tested for the intended input range of 0-3.3V and higher ranges such as -10 to 10V, using a sinusoidal signal. Sinusoidal input was chosen to give a representative picture over the intended range of voltages. The trace for the 0-3.3V sinusoidal input at 10Hz is shown in Figure 9.C.1, and shows a one-to-one correspondence between the input and output, which is exactly the intended need for the circuit to provide accurate data. In Figure 9.C.2, a -15 to 15 V sinusoidal input at 10 Hz was used to test if the circuit can sufficiently protect the microcontroller by limiting the voltage. As clearly can be seen, the negative voltage is completely cut-off, and the positive voltage is limited to ~3.3V, which is below the max microcontroller voltage of 4V.

In terms of power dissipation, the current circuit setup also does exactly what is needed. When the -15 to 15V signal is tested, the diodes each peak at 10 mW of power dissipated, which is well below their limit of 500 mW. The same case also goes for the resistor, which peaks at 100 mW the -15 to 15V signal, which is also well below its limit of 500 mW. This ensures that the protection elements themselves do not get destroyed when the time to protect the circuit comes.

Overall, these traces show that the current circuit does not affect the input signal in the intended 0-3.3V range, which means that the input sensor voltages can be accurately read by the microcontroller. It also shows that if the input voltage is outside of the intended voltage range, the circuit can limit the voltages to a 0-3.3V range and ensure that the microcontroller is not damaged.

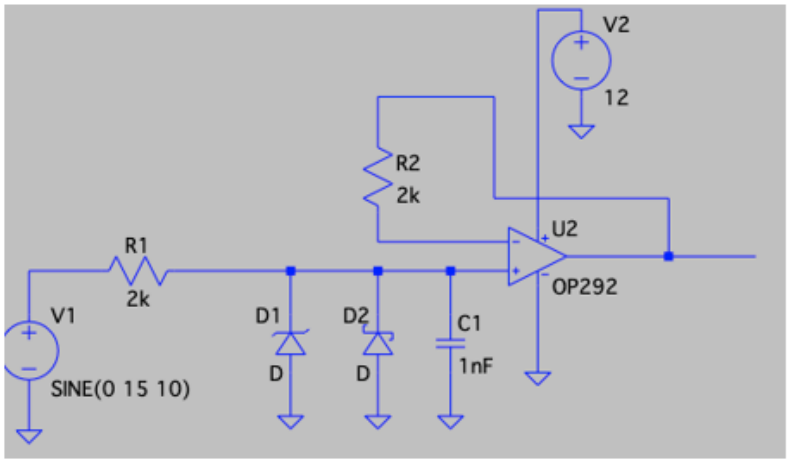


Figure B: LTSpice Circuit used for simulation of the 0-3.3V signal conditioning circuit

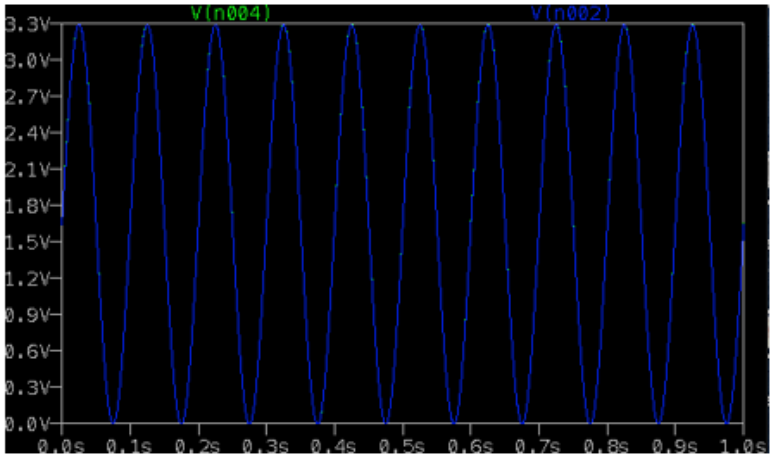


Figure C: LTSpice Simulation Result showing 0-3.3V Sinusoidal input (Green) and output (Blue)

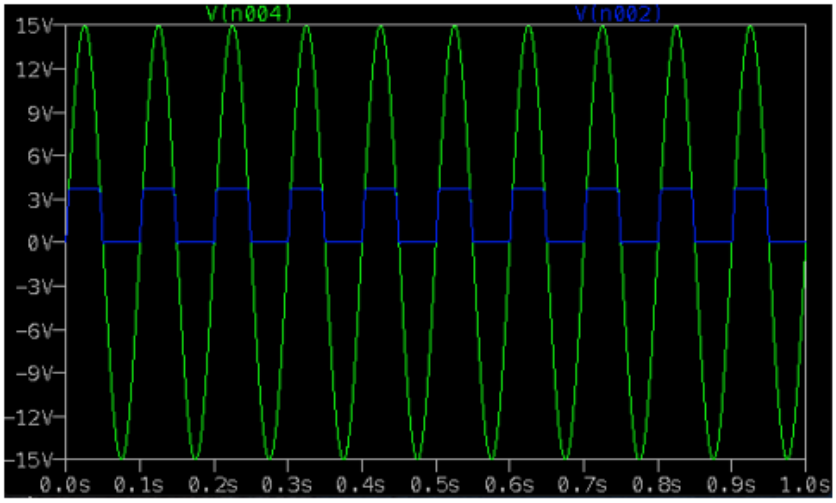
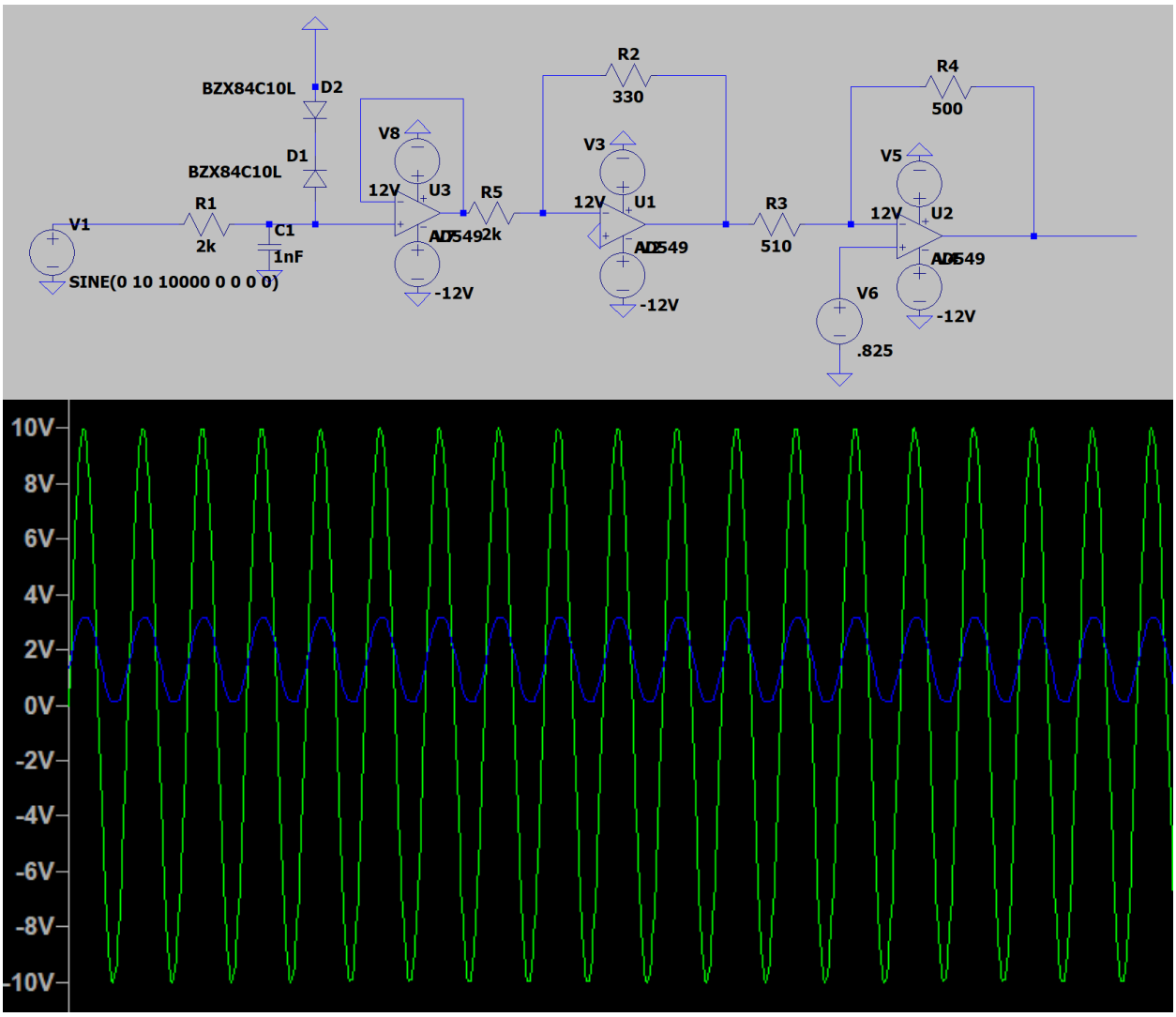
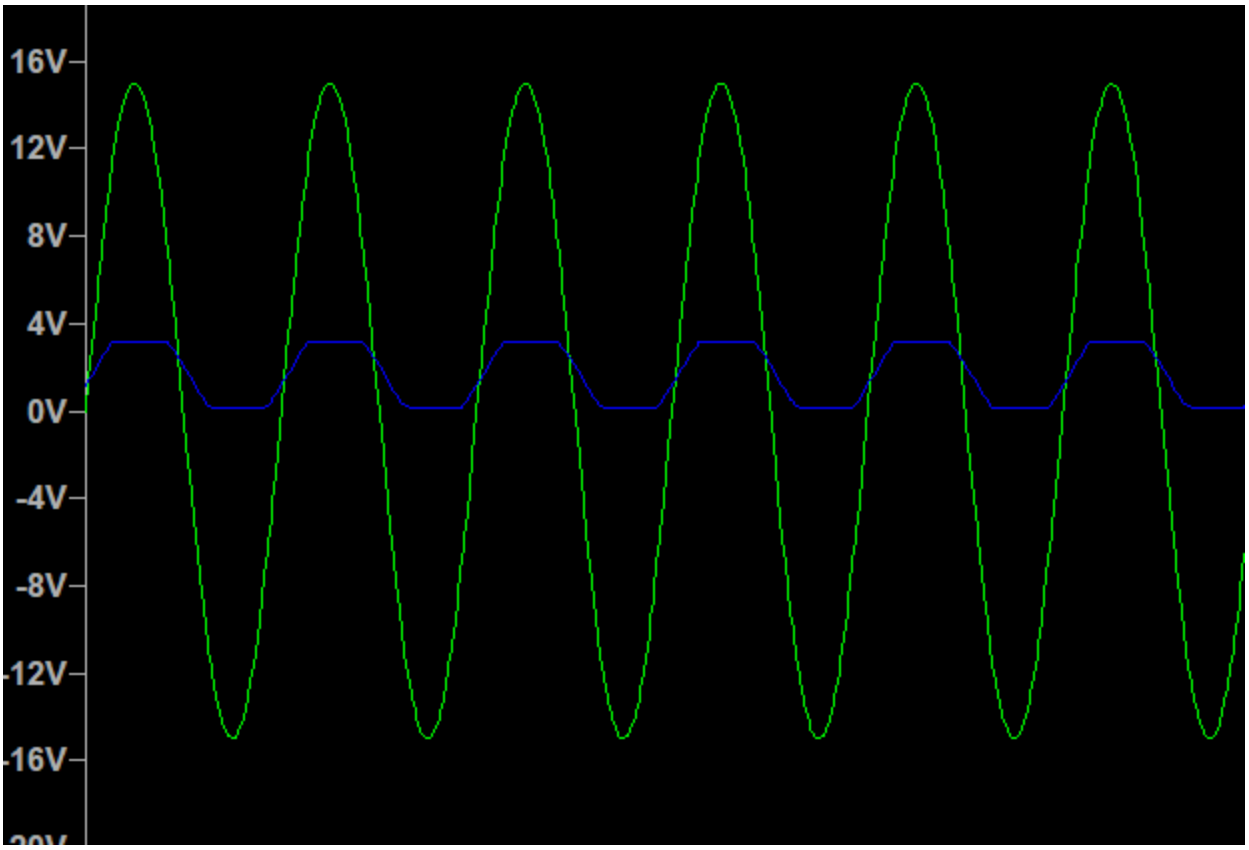


Figure C: LTSpice Simulation Result showing -15 to 15V Sinusoidal input (Green) and output (Blue)

Figure 9: 0-3.3V Input Conditioning Circuit Simulation

A second goal of the PCB was to condition the -10V to 10V signal into a 0-3.3V signal, thereby making the current lab setup and the STM input voltage compatible. The 10V conditioning circuit is presented below in the


BZX84C10L
D2
D1
BZX84C10L
V1
R1
2k
C1
1nF
SINE(0 10 10000 0 0 0)
V8
12V
U3
AD549
R5
2k
-12V
R2
330
V3
12V
U1
AD549
-12V
R3
510
R4
500
V5
12V
U2
AD549
-12V
V6
.825
10V
8V
6V
4V
2V
0V
-2V
-4V
-6V
-8V
-10V



This circuit demonstrates a good ability to condition the 10V to -10V signal to the 0-3.3V range, along with overvoltage protection when too high of a voltage is input. The 0-3.3V conditioning circuit and the 10V to -10V conditioning circuits were apples to every Analog and Digital Input put, allowing for the user to select the input voltage via to toggle switch.

Protector PCB Circuitry: Test Bed Sensor Conditioning Circuits

The three main sensors used on the RC Car testbed that currently send input to microcontrollers are the temperature sensors, force transducers, and IMU. The temperature sensors are a purely analog input signal and need a Wheatstone bridge and difference amplifier circuit in order to read this analog input. However, rather than having to build this circuit, the thermocouple that was chosen came with a premade PCB, the AD8495 Analog Output K-Type Thermocouple Output [17], which serves this purpose without the need for building additional circuitry.

The IMU is in a similar situation. The IMU also comes with signal processing PCB that processes the data of IMU and converts to I2C, which the microcontroller can directly read [18]. This also saves a good amount of space and effort for the creation of a circuit to do this function.

The force transducer represents another difficulty. It also comes with a PCB that converts its input to a digital output. However, this output is at a 3.3V. As such, a small circuit must be built to handle this signal and output an analog signal for the microcontroller to read. Based on the circuit diagram of the strain gauge on the force transducer, a Wheatstone bridge is already part of the sensor, which saves some space and adjusting that is needed for the Wheatstone bridge [19]. The only circuit that needs to be built was a difference amplifier to process the

difference between the Wheatstone bridge outputs so that a single analog output is sent to the microcontroller. A representative circuit diagram of a difference amplifier is shown in Figure 12. The gain of this circuit is

$$V_{out} = \frac{R_2}{R_1(V_2 - V_1)}$$

Where the difference V2 – V1 is the difference between the nodes of the Wheatstone bridge. The difference amplifier shown in Figure 12.I is that it has a high input impedance, which can affect the sensor voltage. If this is found to be a large issue, an instrumentation amplifier will be used instead, whose circuit diagram is shown in Figure 12.J, and whose gain is:

$$V_{out} = \frac{\left(\frac{2R_2}{R_1} + 1\right) R_4}{R_3}(V_2 - V_1)$$

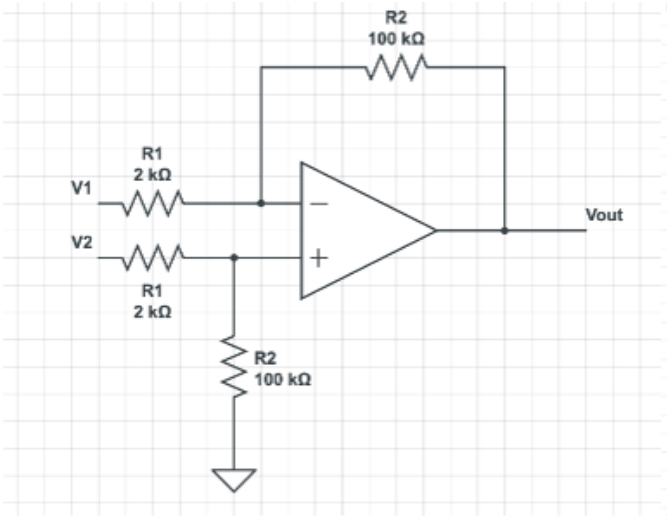


Figure I: Difference Amplifier Circuit

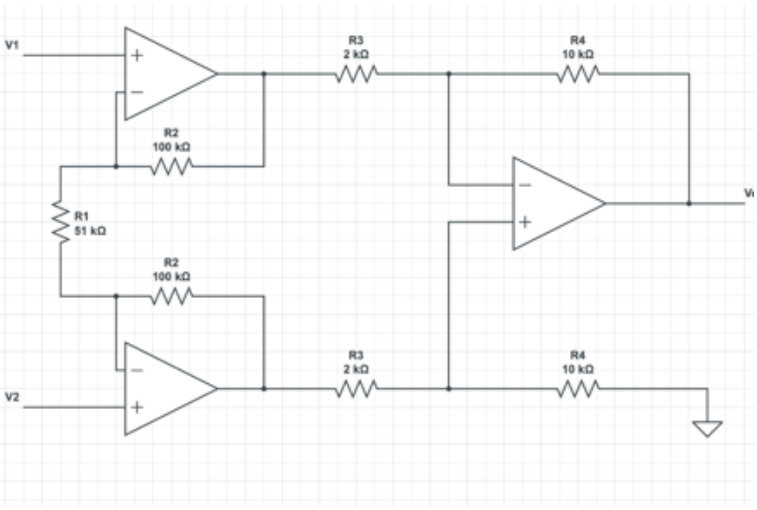


Figure J: Instrumentation Amplifier Circuit

Figure 12: Test Bed Sensor Conditioning Circuits

PCB Design

The 3D model below in Figure 13 gives a perspective of what the final protector PCB will look like. The actual PCB design is shown in Appendix E and is designed to handle power voltage of 12V to 15V. The board is designed to withstand only +/- 15 V inputs as a maximum. A 50 mil trace for power was selected for this purpose, as it can handle approximately 3 Amps of current, which is well below what the STM32 microcontroller and operational amplifiers will pull.

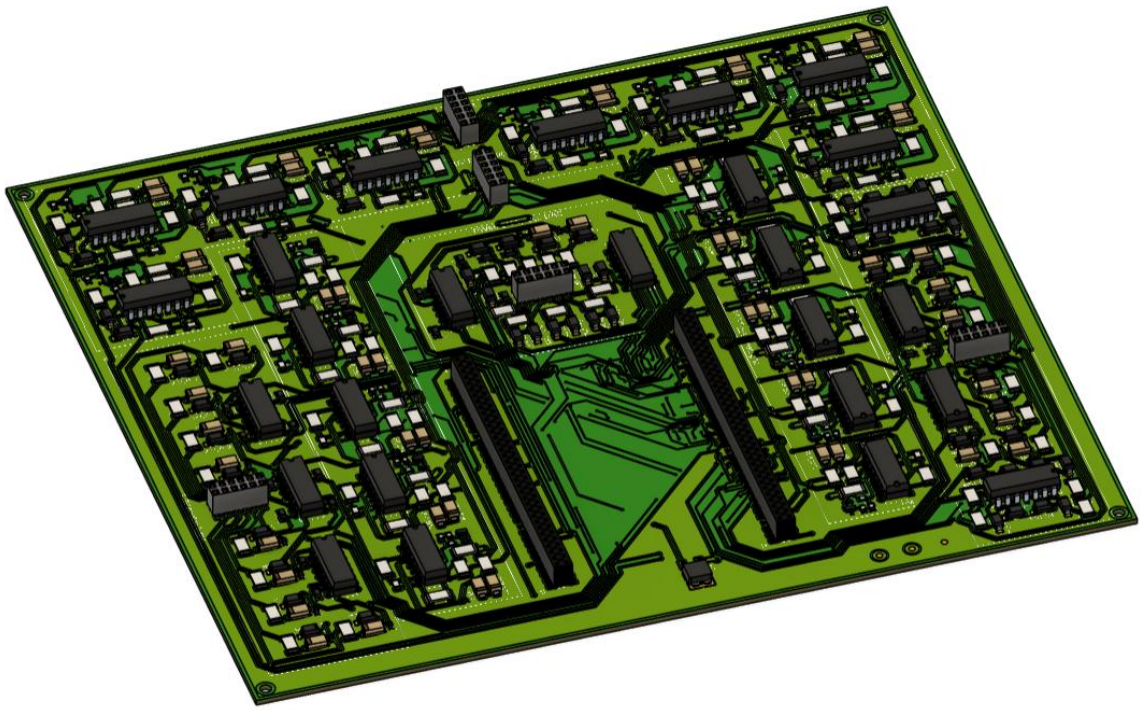


Figure 13: Protector Board PCB CAD

The quiescent current from the LM358 Operational Amplifiers is $300\mu A$ per channel, which is negligible even across 44 channels. The stm32 microcontroller can only pull a maximum of 500 mA. To protect the stm32 microcontroller, an 8 V voltage regulator will bring down the voltage to the proper 8V input, and a 400 mA fuse will be an added layer of protection in the case that the STM32 tries to pull more current than it can handle.

The second iteration of the PCB presented in Figure 13 was designed to handle both 0-3.3V and +/-10V inputs from all of the digital and analog inputs.

The protector board uses four 2-row, 12-pin socket headers spread out across the board. These socket headers are connected to the BNC connector board with pin header extenders. Due to the solid connection between these two plates, the assembly is simplified and has an added layer of durability. The Protector PCB will first be screwed into the bottom of the case, then the STM32 microcontroller will be inserted onto it, as shown in the assembling drawing in Appendix H. Finally, the BNC connector can simply be placed on top of the protector board and pushed down to insert the headers into the sockets. Once this connection is made, the BNC connector board will be meshed with the case surface, and the second set of M4 screws can be used to fasten it to the case.

A set of two 2-row, 70-pin socket headers are also integrated into the protector board. The STM32 microcontroller will have two 2-row, 70-pin pin headers inserted through their I/O holes. The user must solder these pin headers onto the STM32 microcontroller. Once soldered, the STM32 microcontroller can simply be pushed onto the socket header, which automatically routes all of the necessary I/O pins to their respective conditioning circuits. This allows for quick and easy implementation of the STM32 board into the case. This also was designed so that if a STM32 microcontroller were to malfunction, one would only need to quickly replace the STM32, upload the firmware, and have a new DAQ ready for use. The BNC connector board

was constructed with BNC connectors for each input and output, consistent with the needs of the current ME 4056 Systems Lab experimental setup. The durability and noise reduction provided by the BNC connectors, along with the reduced cost in replacing all other connectors currently in the lab, outweigh their added cost.

The BNC connector board and the conditioning circuits in the protector board use trace widths of 24 mil, allowing it to handle currents up to 2 amps. Since this wire is directly connected to the protection board, which has a 2k Ohm resistor as its first interface, it can easily handle the current drawn with even the maximum input voltage of 15V. The reason for these highly durable traces is so that the electrical components, specifically the operational amplifiers, will break before anything else. The operational amplifiers will use sockets. This will allow for easy replacement of any destroyed op amps without the need of soldering, adhering to the goal of 'ease of repair'. Wires past the first 'buffer' were reduced in trace width for compactness, as after that point the current was expected to be negligible.

User Interface and Exterior

For the DAQ system, a protective and functional case for the STM-32 DAQ system was developed, ensuring both accessibility and protection for the components. This can be seen in Figure 14. The design process began with creating an assembly in SolidWorks that included the STM-32 microcontroller enclosed between two custom-designed PCB boards. The bottom PCB, named the 'Protection PCB', contains additional electronics and circuits designed to protect the STM-32 from excessive current and voltage spikes, while the top PCB features BNC connectors, allowing for convenient switching of input and output connections. This modular approach ensures that the system remains adaptable to different desired scenarios while maintaining robust protection.

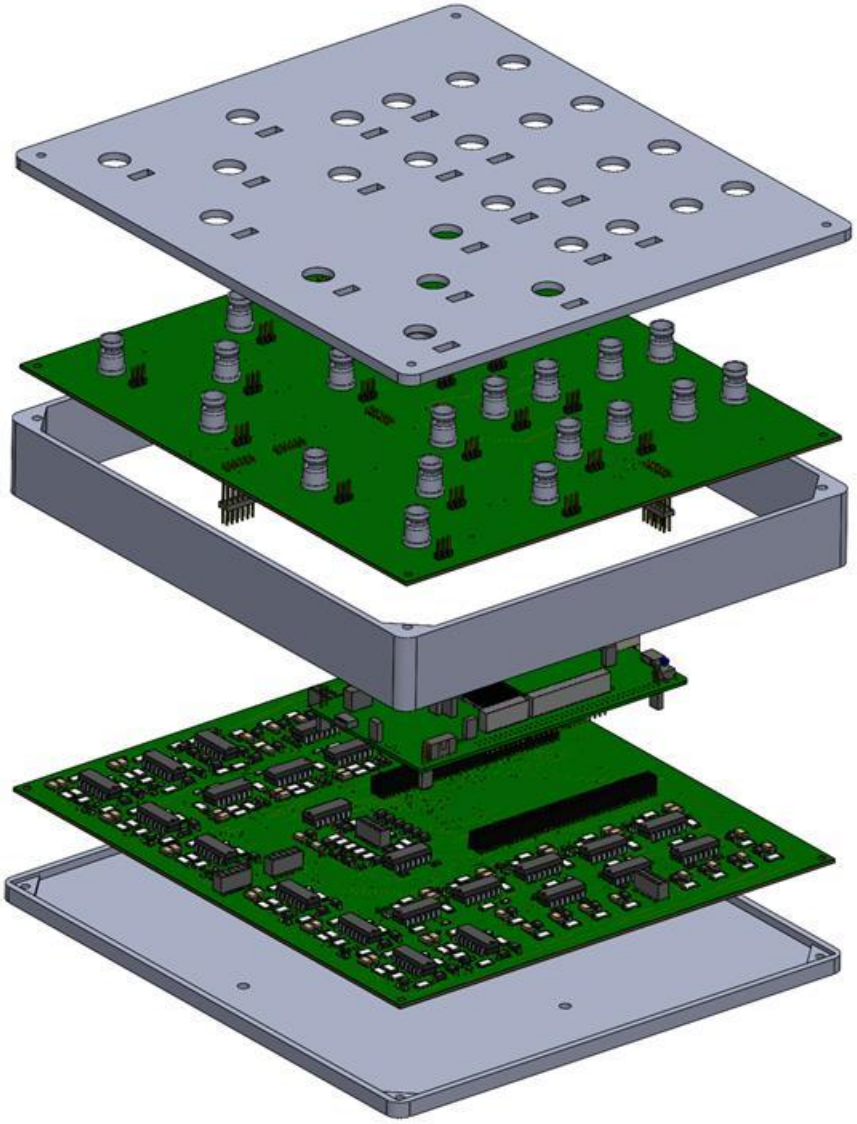


Figure 14: STM32-Based DAQ Assembly

With the SolidWorks assembly complete, a custom case that houses and secures all components effectively was designed around that assembly. The case consists of three main parts: a bottom section, a middle section, and a top cover. The assembly process follows a bottom-to-top approach, beginning with small standoffs with heat inserts to support the first PCB. Once the bottom PCB is mounted, the second PCB is placed on top and secured using M4 screws with heat inserts. The middle section of the case includes precisely placed cutouts for essential interfaces, such as banana plug connections, Ethernet, and USB-C ports. These design

features allow for easy external connectivity to key systems while maintaining a compact and enclosed structure. Additionally, the bottom part of the case includes pre-drilled holes that allow it to be securely mounted onto the RC car test bed. The case is designed to fit on a flat aluminum bar fit to the car, ensuring stability during testing while minimizing vibrations and movement.

The top PCB, featuring the BNC connection ports, is securely fastened while allowing just the connection ports exposed for modularity. The case is designed to be both protective and user-friendly, ensuring that all critical connections are easily reachable without compromising the system's integrity. By using heat inserts and screw fastenings, the case achieved a balance between structural durability and ease of assembly. Overall, the case design provides reliable housing for the STM-32 DAQ system while facilitating seamless operation and component access, while its mounting system ensures a secure and stable attachment to the RC car.

STM32 Pinout Configuration

Pinout configuration provided in Appendix # shows the current pin names for each of the I/O channels on the STM32 base DAQ. Each of these pinouts were tested by setting them all simultaneously high and testing for functionality.

High Level Core Scheduling

As can be seen in Figure 15, the M4 core will first take the Hardware Semaphore (HSEM). It will then run through its Real Time Operating System (RTOS) routine and sample data, mainly through Direct Memory Access (DMA), with some help of the CPU, and export this data directly to the region of SRAM memory designated by the HSEM. Once it has run for a specified amount of time (ex. 0.5 seconds), the M4 core will give the HSEM, allowing the H7 Core to take it, read that data, and delete the data from SRAM. Immediately after this, the M7 core will give the HSEM back to the M4 core so that it can begin writing data to SRAM memory again. During this time, the M7 core will begin its final processing of the signal data, in addition to organizing, packaging, and exporting it via ethernet to the cloud. The architecture used on the

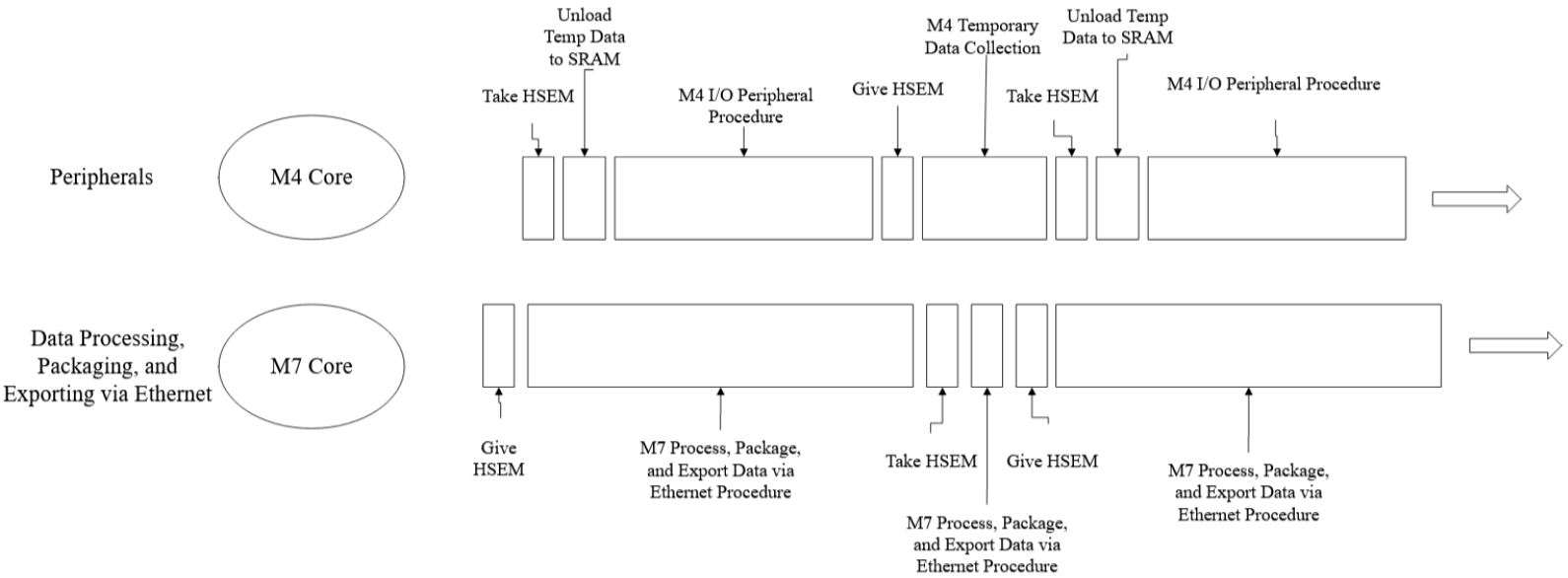


Figure 15: M4 and M7 Threads

STM followed this concept, to allow future developments if needed allowing for two way communication. Currently, one way communication is used, with the M4 core sending digital pin information to the M7 core. Former hardware semaphores also were not used due to the simplicity of this transaction. Rather, the HSEM data was stored in a structure with the data that the M4 was sending to the M7, and those were used to synchronize the cores.

A further breakdown of what is happening during the M4 I/O Peripheral Procedure is provided in Figure 16. Encoder and Digital Inputs are processed through the CPU, as DMA is not an option for these pins. These values will be likewise written into SRAM, and transferred on a timed schedule to the M7 core. While the M7 core has the HSEM, the M4 will load data into a buffer array which will be unloaded first into SRAM when the M4 core takes the semaphore.

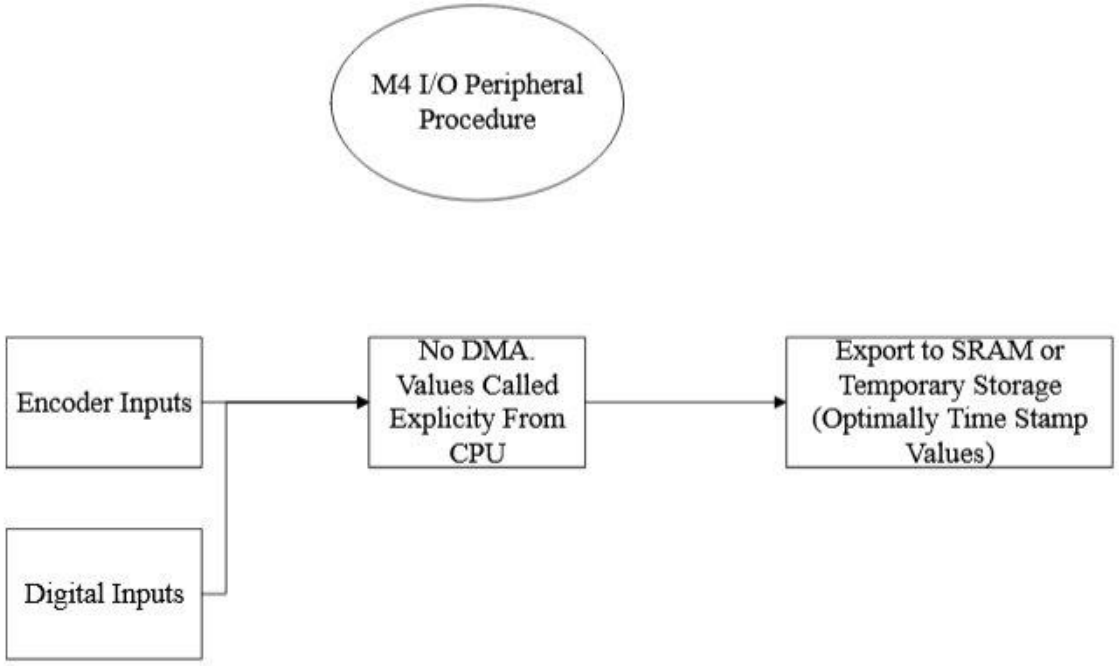


Figure 16: M4 Input/Output Peripheral Procedure

The M7 core will take the semaphore, read the data from SRAM, delete the data on SRAM, and then give the semaphore back to M4. It will read the analog input pins using Direct Memory Access (DMA), thereby reducing the computational load on the M7 core. Likewise, it will use RTOS to simultaneously poll for I2C inputs, as I2C will be configured on the M7. It will then process the incoming data and export it through ethernet to the cloud. Likewise, data will come in through ethernet from the cloud on how to set peripherals including digital outputs, PWM signals, I2C outputs, and the function generator output. This is most easily done directly on the M7.

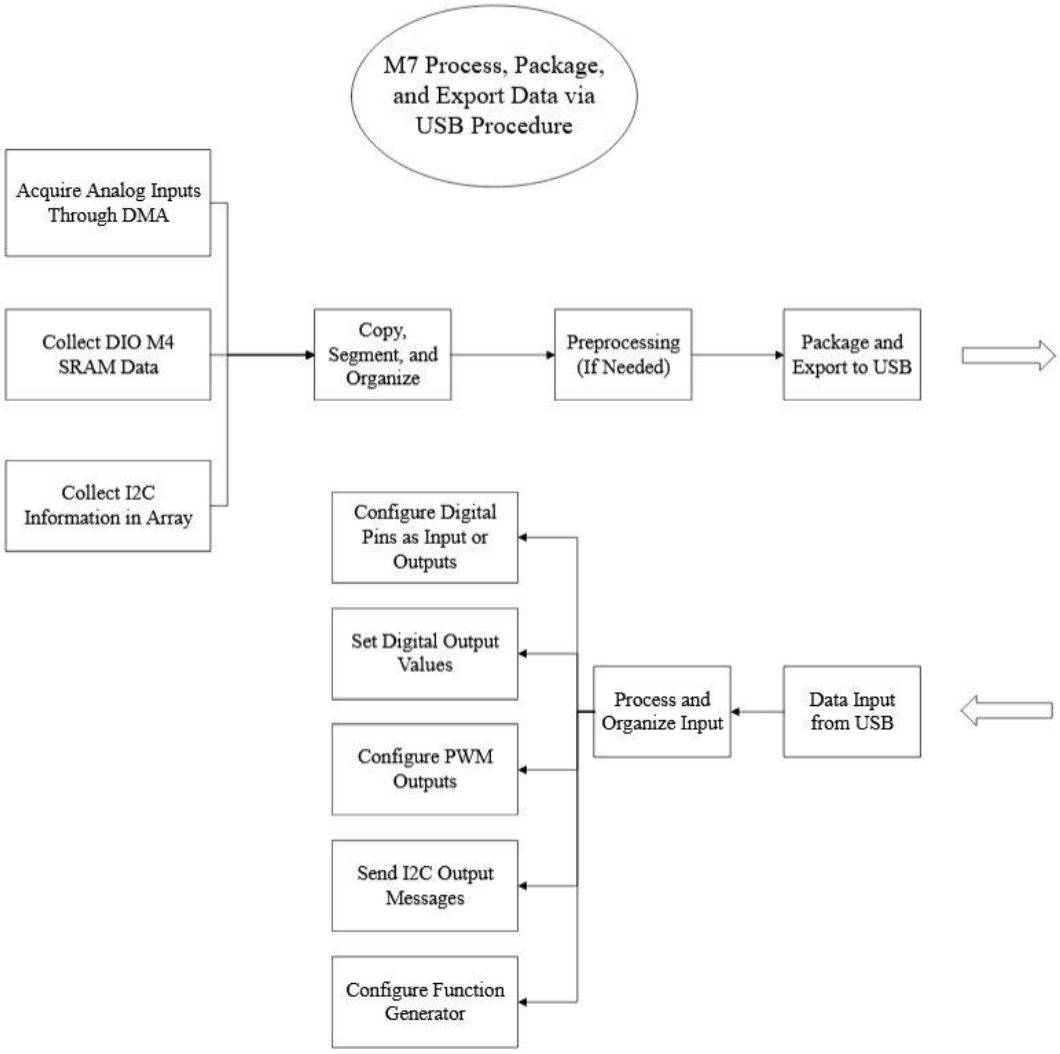


Figure 17: M7 Core General Procedure

Real Time Operating System (RTOS)

To ensure real time collection of data, a real time operation system, namely the FreeRTOS API, will be implemented into the firmware of the STM32 microcontroller. FreeRTOS was selected for this project due to its ease of implementation with the STM32 microcontroller, and for fulfilling real-time needs. This will allow for multiple threads to be sequenced one after another, to check each channel, and collect data from a given channel for a specified amount of time.

The RTOS implemented on the M7 core is most easily visualized in Figure 17. Likewise, every box in this model represents an RTOS thread, each communicating with each other simultaneously to collect, modify, and export/import data. A thread will poll for when the M4 core releases the HSEM, and will then take it, gaining access to the SRAM. The M7 will then use that thread to read, delete, and transfer that data from SRAM. Likewise, simultaneous threads will be used for reading I2C input, processing data, transmitting and receiving via ethernet, and then setting output for digital signals, PWM, I2C, and function generator.

Direct Memory Access (DMA)

Direct Memory Access (DMA) is used in the initial collection of data from analog sensors. Data is collected continuously by analog sensors and written into memory with the STM32's built-in DMAs, instructed by the M4, running in circular Peripheral-to-Memory mode. A similar reference schematic of an M3 core with two DMAs is presented in Figure 18. The concept is the same, the DMA pulls information directly from the peripherals and inserts it directly into SRAM. This is the same mechanism that the M4 will use.

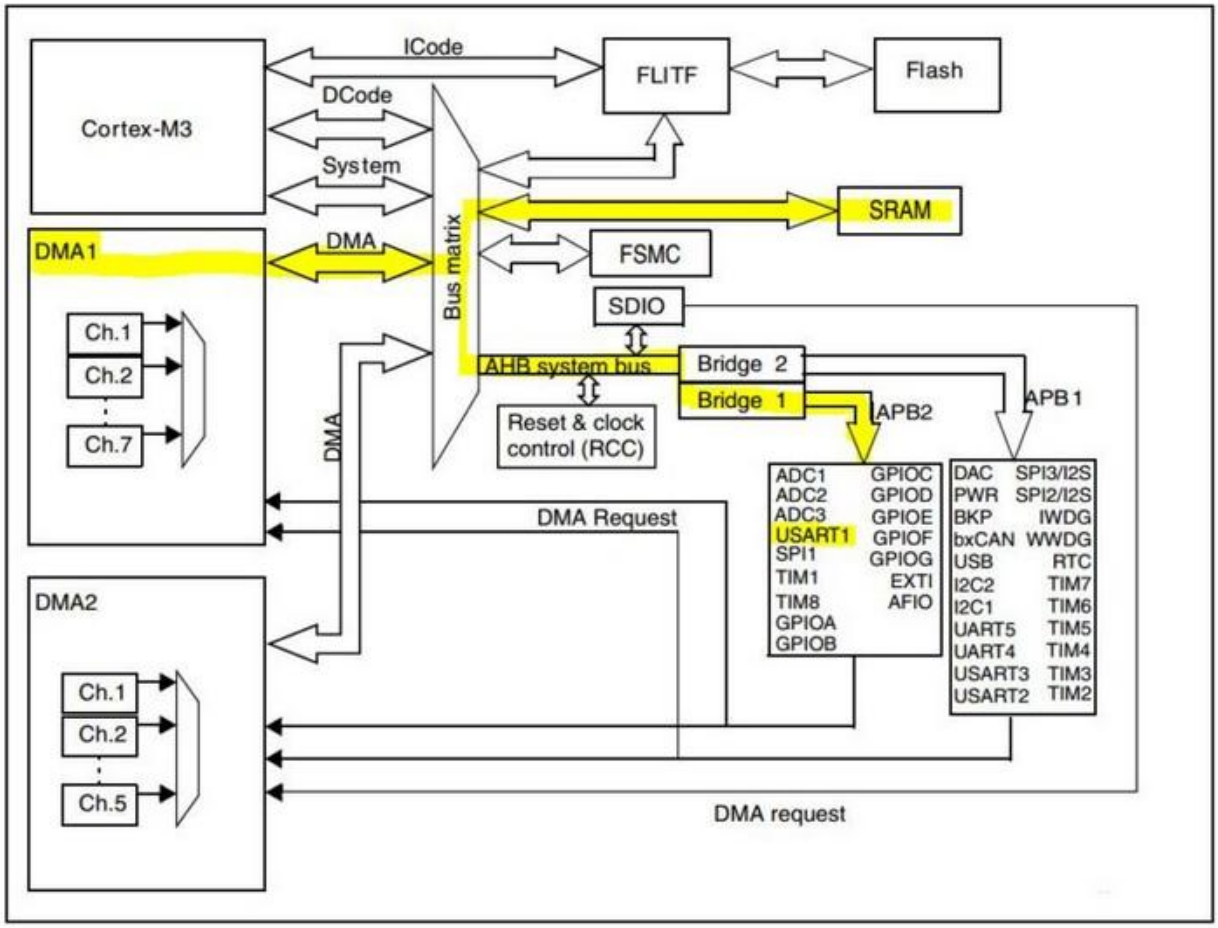


Figure 18: Dir.ect Memory Access (DMA) Path Within M3 Core [16]

Hardware Semaphores (HSEM) for Dual Core Communication

To permit the M7 access to the data from the M4 only once it is safe to do so, and to prevent the M7 from attempting to access data prematurely, semaphores are implemented. The semaphores permit cross-core communication, enabling the cores to communicate to each other that a specified segment of memory has become reserved or available. Figure 19 shows the breakdown of the M4 and M7 cores in the STM32h755zi-q microcontroller. As highlighted, the M7 and the M4 have common access to SRAM4, and therefore this location in memory will be used to transfer information from the M4 to the M7. Assuming that both the M4 and M7 are simultaneously editing SRAM4, semaphores are used to dictate who has access when.

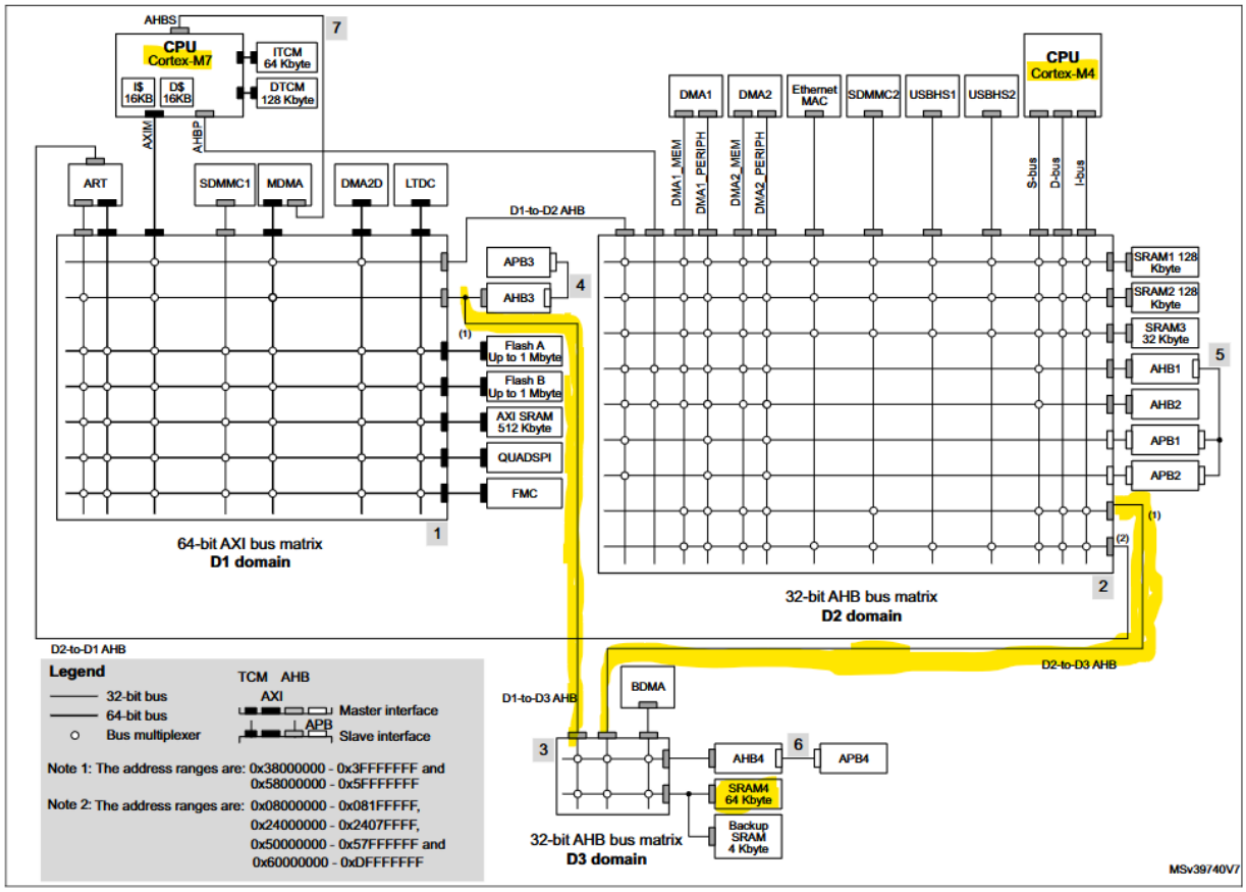


Figure 19: M4 and M7 CPU Paths to Shared SRAM4. With Trade Access using Hardware Semaphores (HSEM)

Ethernet & Cloud Based Interface

Ethernet packets are prepared for export by the M7. The M7 creates a packet buffer in memory, then copies data into that buffer from where it was originally stored when collected. Headers are created for the appropriate protocols. In this case, an IP header containing source/destination IP addresses, protocol type (TCP), etc.; a TCP header containing protocol-specific data; and an Ethernet Frame header including MAC addresses, ethernet frame type, etc.

Several ethernet protocols were also considered. When deciding the correct media-independent interface (MII) to connect to a Fast Ethernet, several factors were considered. A core client requirement is the facilitation of real-time user input and data fetching from the STM32-based DAQ, which requires a continuous two-way communication loop is required. Additionally, the incoming data must be properly time-tamped for student analysis, so precision timekeeping is a must. Taking these into consideration, we have preliminarily selected the MII PTP Synchro interface protocol, established by the IEE, for our STM32-based DAQ to GUI communication, as it provides synchronous, accurate timestamped data [14].

The method of which ethernet data will flow to the cloud-based interface is also crucial to its development. The initial data from the ethernet must first undergo signal processing on the server, then be uploaded to the database. The interface will then fetch this data and display it on the web. This information loop must be real-time and have low latency. As such, it was decided to opt for splitting data flow into two independent loops, rather than a continuous data flow as

that would impede development due to its high co-dependency on the STM32-based DAQ and database communication, which is graphically shown in Figure 20.

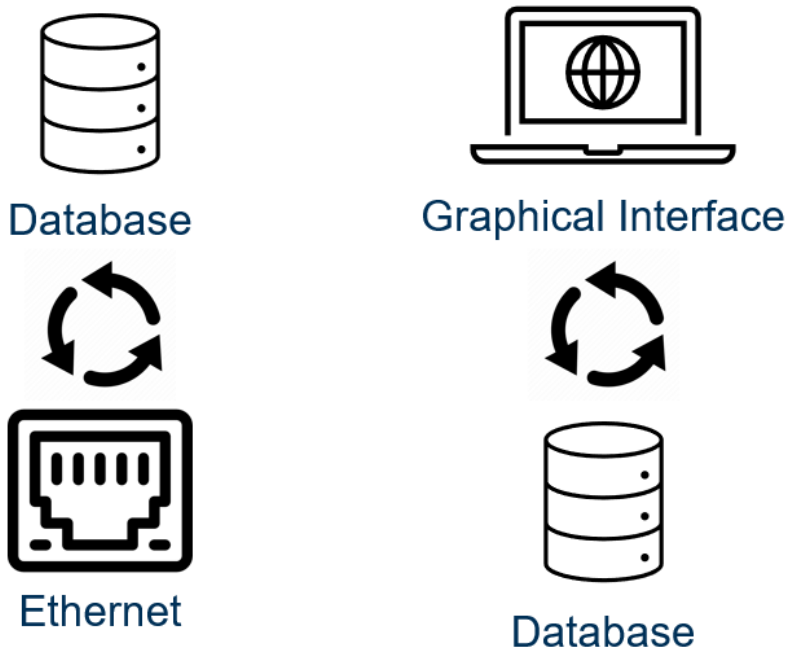


Figure 20: Double Thread for Cloud GUI

These two loops would be configured completely independently, with the database acting as the liaison between the interface and the ethernet. This allows signal processing and upload from the ethernet to occur simultaneously as data is being fetched. The database will then include tables to store user input as well as dictate which pins on the DAQ the ethernet should actively fetch data from.

The database's architecture is crucial to facilitate efficient data transfer. Several iterations of the database architecture were proposed. Although the database stores individual student's data, students work within groups for each lab, which means lab data must correspond directly to groups rather than individual users. With that core factor in mind, a database architecture with a core data table that would have groups as the primary key was established, which allows them to directly reference in independent lab tables without data repetition as a foreign key. We can extend the group system to accommodate not only students, but also professors and teaching assistants as they also need access to points of data. Each student group and professor group must have different viewing privileges within the GUI, as professors must be able to view everything, and students should only have access to their own group's data. To manage these various permissions, a JSON table will also be added to include permissions for group categories.

Using this proposed database architecture, basic real-time plotting capability was developed. This real-time plot allows for multiple to be created and also moved, allowing modular customization of the interface between various labs. This was built in React.js, with the charts being completed within Chart.js, which facilitates real-time functionality [15]. The chart

used a minimal implementation of an API, using Django, to fetch data from the PostgreSQL database, which was configured to occur on a set interval. An example default chart is shown in Figure 21.

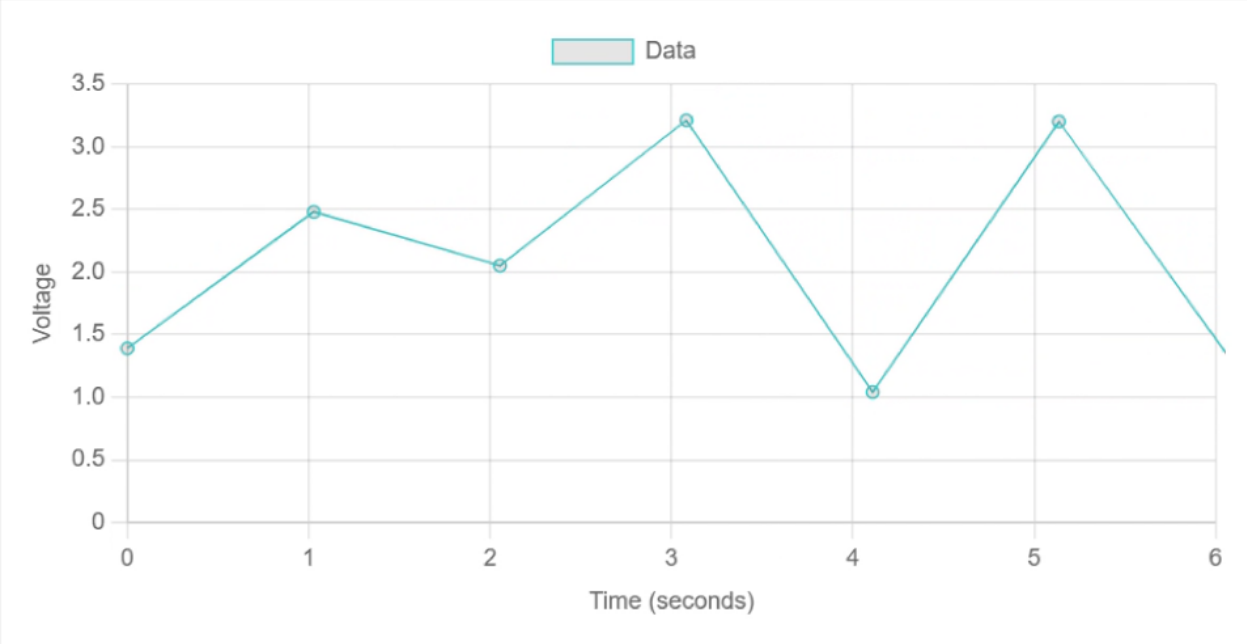


Figure 21: Real Time Data Collection GUI

RC Car Testbench

To validate the data acquisition system, an RC car testbench equipped with sensors that cover the full input type range will be used to simulate a lab experiment. The sensors, used to optimize lap time, consist of thermocouples with an analog output, force transducers with a digital output, encoders with a PWM output, and an IMU with an I2C output. In addition to these sensors, also equipped are speed controllers, and fans that the DAQ can control. All of these inputs and outputs allow for full testing of the DAQ in an easily manageable package as opposed to the large experiments normally used in labs.

The thermocouples will be used to measure the temperature of the motor and speed controller, the force transducers will record the force transmitted through each wheel, the encoders will indicate wheel speed, and the IMU will provide velocity, acceleration, and positional data. These factors will aid the driver of the car in better understanding the car's dynamic and help improve lap time.

The integration of these sensors can be seen in Figure 22 below. Full designs of the mounting solutions can be seen in the attached data package.

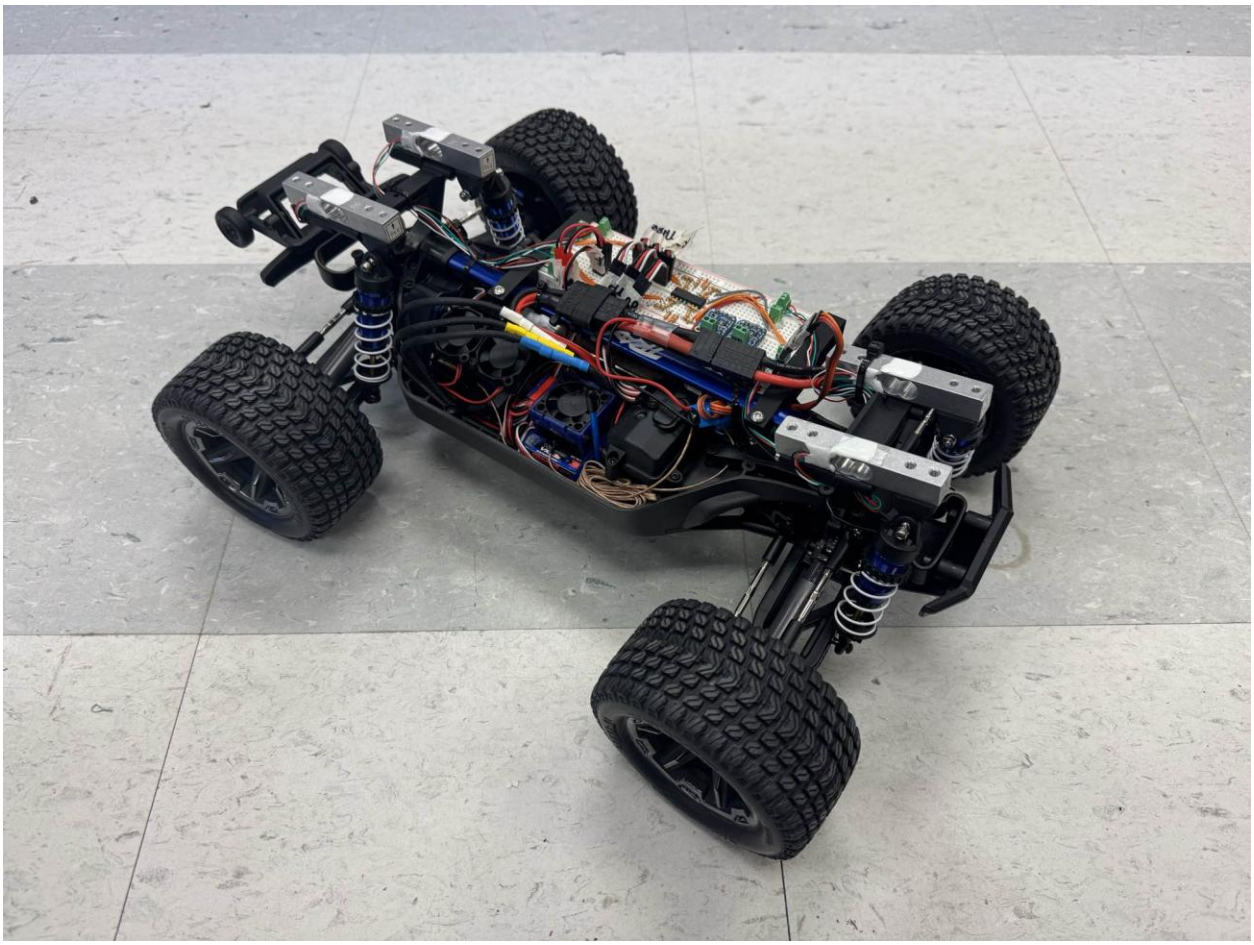


Figure 22: RC Car Testbench

VII. Engineering Analysis and Experiments

Circuit and PCB Testing:

The circuits are designed to handle a maximum of +/- 15V, and frequencies up to 50 kHz. Exact requirements for each of the input and output peripherals are outlined in the Needs, Functions, and Constraints table below.

DAQ Subsystem

Needs	Functions	Constraints
10x Analog Input Pins	>35 kHz Sampling Frequency 16 bit ADC resolution Capacity for all 10 to be on at once 0-3.3V maximum input	Input must only be a reference voltage Voltage must be within 0-4 V
8x Digital Input/Output Pins	Voltage Switch Speeds up to 5 kHz Provide Constant Voltage Outputs Read Voltage Inputs	Input/Output must be a refence voltage Voltage must be within 0-5V
2x PWM Outputs	Generate Output PWM Signals Duty Cycle from 0 - 100% Frequencies up to 50 kHz	Output Must be a Reference Voltage Output Voltage Between 0-5 V
2x Encoder Inputs	Handles up to 50 kHz PWM Frequency Input	Input must only be a reference voltage Voltage must be within 0-5 V
2x Function Generator Outputs	Produce Signal Frequencies up to 50 kHz 12 bit DAC resolution	Output must be a reference voltage Voltage must be within 0-5 V
2x I2C Inputs	Baud Rate up to 115200 bps	Output must be a reference voltage Voltage must be within 0-5 V
Overall Constraints		Maximum Output Current = 20 mA Maximum Total Output Current = 140 mA Maximum Analog Voltage Input 4 V Maximum Digital Voltage I/O 5V Maximum Total Injected Current = +/- 25 mA

Figure 23: Needs, Functions, and Constraints

Based on the requirements, an initial set of tests are proposed for testing of the prototype, provided in Figure 23 and 24. These tests verify functionality at different voltages and frequencies that are required for functionality of the DAQ and have associated passing criteria outlined. In the case that a peripheral is unable to pass all categories, a further analysis will be done on the particular peripheral in order to define the 'bounds of successful operation', and an analysis into the reason for failure will be conducted. Encoder testing follows the same rubric as the Digital Pin Testing. The first round of testing will be done using the 0-3.3V protector board. If time permits, another PCB containing +/-10V conditioning circuits will also be implemented and tested by similar criteria. I2C performance will be dictated by successful communication with an I2C compatible sensor in the RC Testbed.

Analog Pins Testing (0-3.3V)	DC Voltage (0 Hz)	Low Frequency (100 Hz)	High Frequency (35 kHz)
Low Voltage (200 mV Amplitude)	Pass/Fail	Pass/Fail	Pass/Fail
Medium Voltage (3.3V Amplitude)	Pass/Fail	Pass/Fail	Pass/Fail
High Voltage (15V Amplitude)	Pass/Fail	Pass/Fail	Pass/Fail
Pass: Voltage Input RMSE < .05V/period			

Digital Pin Testing (0-3.3V)	DC Voltage (0 Hz)	Low Frequency (50 Hz)	High Frequency (2 kHz)
Low Voltage (LV)	Pass/Fail	Pass/Fail	Pass/Fail
High Voltage (HV)	Pass/Fail	Pass/Fail	Pass/Fail
Pass: LV Range within [0, 1.7] V			
Pass: HV Range within [1.6, 3.3] V			

PWM Signal Testing	Low Frequency (100 Hz)	High Frequency (50 kHz)
Low Duty Cycle (10%)	Pass/Fail	Pass/Fail
Medium Duty Cycle (50%)	Pass/Fail	Pass/Fail
High Duty Cycle (90%)	Pass/Fail	Pass/Fail
Pass: Duty Cycles within 2% Accuracy		
Pass: Frequencies within 2% Accuracy		

Function Generator Testing	DC Voltage (0 Hz)	Low Frequency (100 Hz)	High Frequency (50 kHz)
Low Voltage (0-1.65V)	Pass/Fail	Pass/Fail	Pass/Fail
Medium Voltage (0-3.3V)	Pass/Fail	Pass/Fail	Pass/Fail
Pass: Voltage Output RMSE < .05V/period			

Figure 24: DAQ Peripherals Testing Procedure

These initial tests will be done using an oscilloscope and function generator, to obtain clean numerical data that can be processed in MATLAB. A final presentation of the device will use the RC Car testbed to demonstrate the functionality of the STM-Based DAQ system with real time data collection from on board sensors.

Testing of the firmware on the STM32 microcontroller will be done incrementally. First, a pinout configuration will be selected and verified. Then the RTOS, DMA, and HSEM functionalities will be individually tested with UART outputs to a laptop. Then they will be tested in conjunction with Ethernet.

STM32 pinout configuration confirmation

Pinout configuration provided in Appendix # shows the current pin names for each of the I/O channels on the STM32 base DAQ. Each of these pinouts were tested by setting them all simultaneously high and testing for functionality. A second test, whereat each pin was set to its functionality simultaneously, was performed, although two pins currently failed to meet inspection requirements, and will therefore be modified during the second Protector Board PCB iteration.

STM32 RTOS configuration confirmation

The STM32 RTOS configuration will be tested on the M4 core using UART output 3 in the stm32 which will allow for the signal to be collected through the usb onto the computer. The signal transferred will be read and analyzed on a MATLAB script. If RTOS is functioning properly, then the signal coming from a particular thread will correspond to the thread associated with the user input.

Testing for DMA functionality

DMA will be initially tested by pulling analog pin information directly from memory and passing them through UART 3 to a testing computer, which will use a MATLAB script to collect and analyze the inputs. Each of the analog pins will be fed different inputs, and corresponding outputs on the MATLAB script will indicate functionality of the DMA. A test using 5 channels from ADC3 has already been performed, as shown in Figure 25. Future tests will analyze all 10 analog channels simultaneously and analyze the resulting data according to Figure 25.

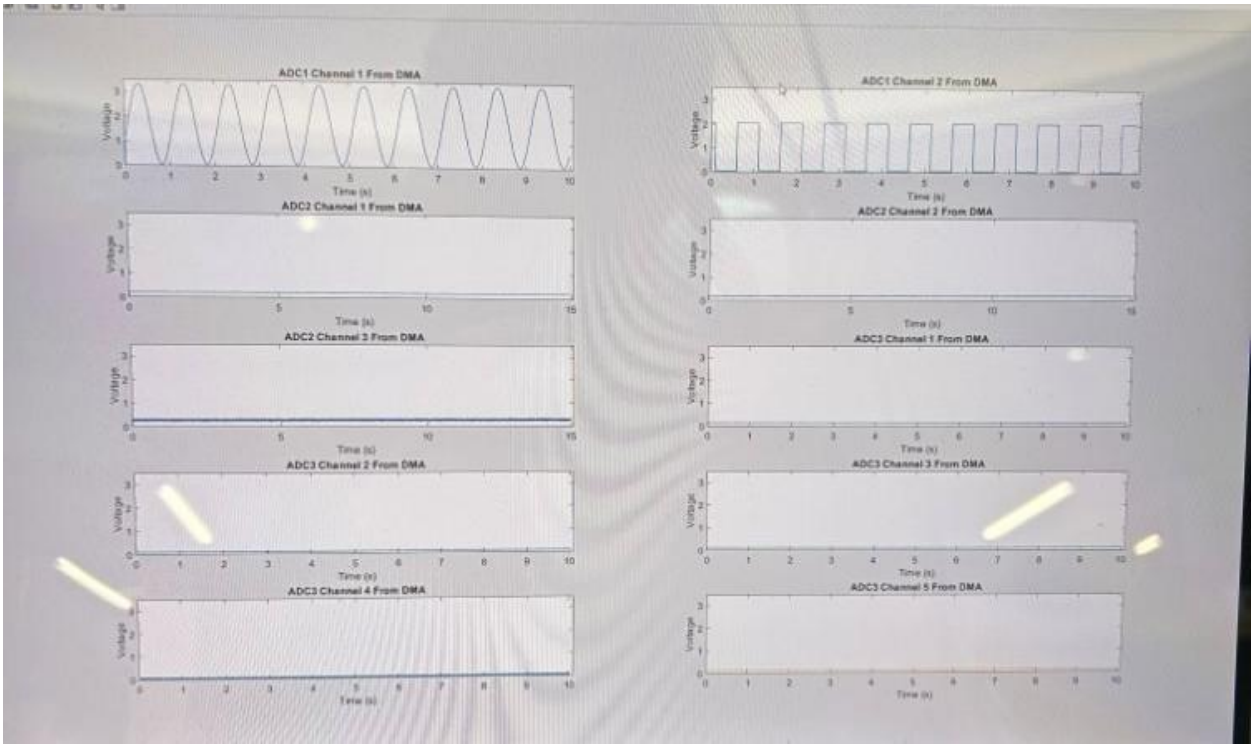


Figure 25: DMA Used to Collect Data from 5 Channels of ADC3 Simultaneously

Testing for HSEM functionality

The underlying principle for testing the functionality of the hardware semaphores is to claim a semaphore with the M4, start writing some data, then try to read that data with the M7. If the M4 and M7 can trade the HSEM and corresponding SRAM data for multiple cycles of then the HSEM cross core communication will be considered a success. Testing will be performed by exporting obtained data on the M7 through UART 3 to a laptop for a MATLAB analysis.

Testing of RTOS, DMA, and HSEM

Prior to sending the information over the ethernet, RTOS, DMA, and HSEM will be configured and the full operation of data collection, processing, and exporting through UART will be conducted. The outputs will be analyzed in MATLAB. This is to confirm full functionality of the STM32 microcontroller prior to data export to the cloud.

Testing for the ethernet packaging and transfer functionality

The STM will be programmed with a known array of data stored in its program memory. When the program is run, all the relevant headers and data packets will be generated for ethernet transmission. However, instead of sending those packets via ethernet, the data will be exported via USART so that the developer can be manually verify (in the debugger) that the prepared packet matches the expected packet.

Testing for verifying signal transmission from STM to software

Once it is verified that the packets produced by the packaging routine behave according to expectations, a test routine can be implemented to verify functionality of the ethernet transmission itself. When the program is run, the data will be moved into the ethernet queue buffer and sent, along with the appropriate headers, across the ethernet connection. If the expected array of data is received at the other end of the connection, the ethernet can be confirmed to function as expected.

Testing of Real-time Plotting Capability

To simulate ethernet data transfer, without the need for the physical ethernet to be fully considered, a Python script was developed that would function as the ethernet, generating random signal voltage data and uploading it within the database. This functions as the ethernet to database loop and allows for development and testing of the interface to occur without a fully functional ethernet. Using this script, plotting functionality was verified as the charts accurately updated in real-time with almost no latency.

Analog Input Tests

The circuits for both the 3.3V and ± 10V signal conditioning circuits were first tested using a breadboard and LTSpice simulations as detailed above. Once the final PCB was assembled, each analog port was tested for both options at varying amplitudes and frequencies as detailed in Figure 26.

Analog Pins Testing (0-3.3V)	DC Voltage (0 Hz)	Low Frequency (100 Hz)	High Frequency (35 kHz)
Low Voltage (200 mV Amplitude)	Pass/Fail	Pass/Fail	Pass/Fail
Medium Voltage (3.3V Amplitude)	Pass/Fail	Pass/Fail	Pass/Fail
High Voltage (15V Amplitude)	Pass/Fail	Pass/Fail	Pass/Fail
Pass: Voltage Input RMSE < .05V/period			

Digital Pin Testing (0-3.3V)	DC Voltage (0 Hz)	Low Frequency (50 Hz)	High Frequency (2 kHz)
Low Voltage (LV)	Pass/Fail	Pass/Fail	Pass/Fail
High Voltage (HV)	Pass/Fail	Pass/Fail	Pass/Fail
Pass: LV Range within [0, 1.7] V			
Pass: HV Range within [1.6, 3.3] V			

PWM Signal Testing	Low Frequency (100 Hz)	High Frequency (50 kHz)
Low Duty Cycle (10%)	Pass/Fail	Pass/Fail
Medium Duty Cycle (50%)	Pass/Fail	Pass/Fail
High Duty Cycle (90%)	Pass/Fail	Pass/Fail
Pass: Duty Cycles within 2% Accuracy		
Pass: Frequencies within 2% Accuracy		

Function Generator Testing	DC Voltage (0 Hz)	Low Frequency (100 Hz)	High Frequency (50 kHz)
Low Voltage (0-1.65V)	Pass/Fail	Pass/Fail	Pass/Fail
Medium Voltage (0-3.3V)	Pass/Fail	Pass/Fail	Pass/Fail
Pass: Voltage Output RMSE < .05V/period			

Figure 26: The testing criteria for the various channels being used and tested on the final PCB and STM design. The digital and analog channels were also tested in the ± 10V as well

Each Analog pin was first tested with the 3.3V option at 100Hz. to verify proper operation. Shown in Figures 27, 28, and 29 are representative plots from analog channel 1. Figure 27 and 29 are for the ideal input range of the signal and shows that the circuit's output is able to accurately match the circuits input as is required for the analog channels. Figure 28 is for a signal outside of the ideal range and shows the circuit's inbuilt protection does its job at limiting both the high and negative voltages and thus preventing damage to the STM microcontroller.

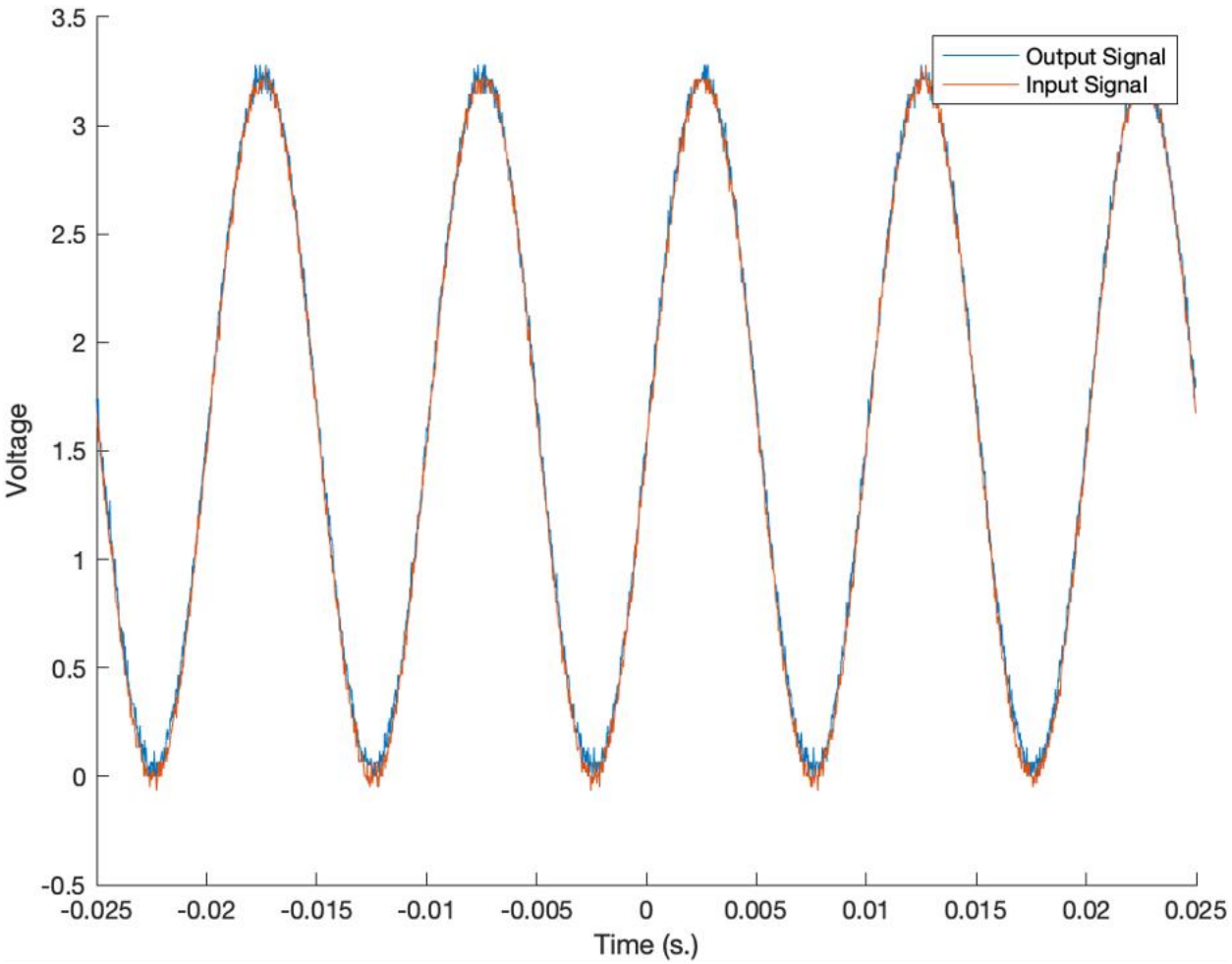


Figure 27: 3.3V input at 100Hz on Analog Channel 1

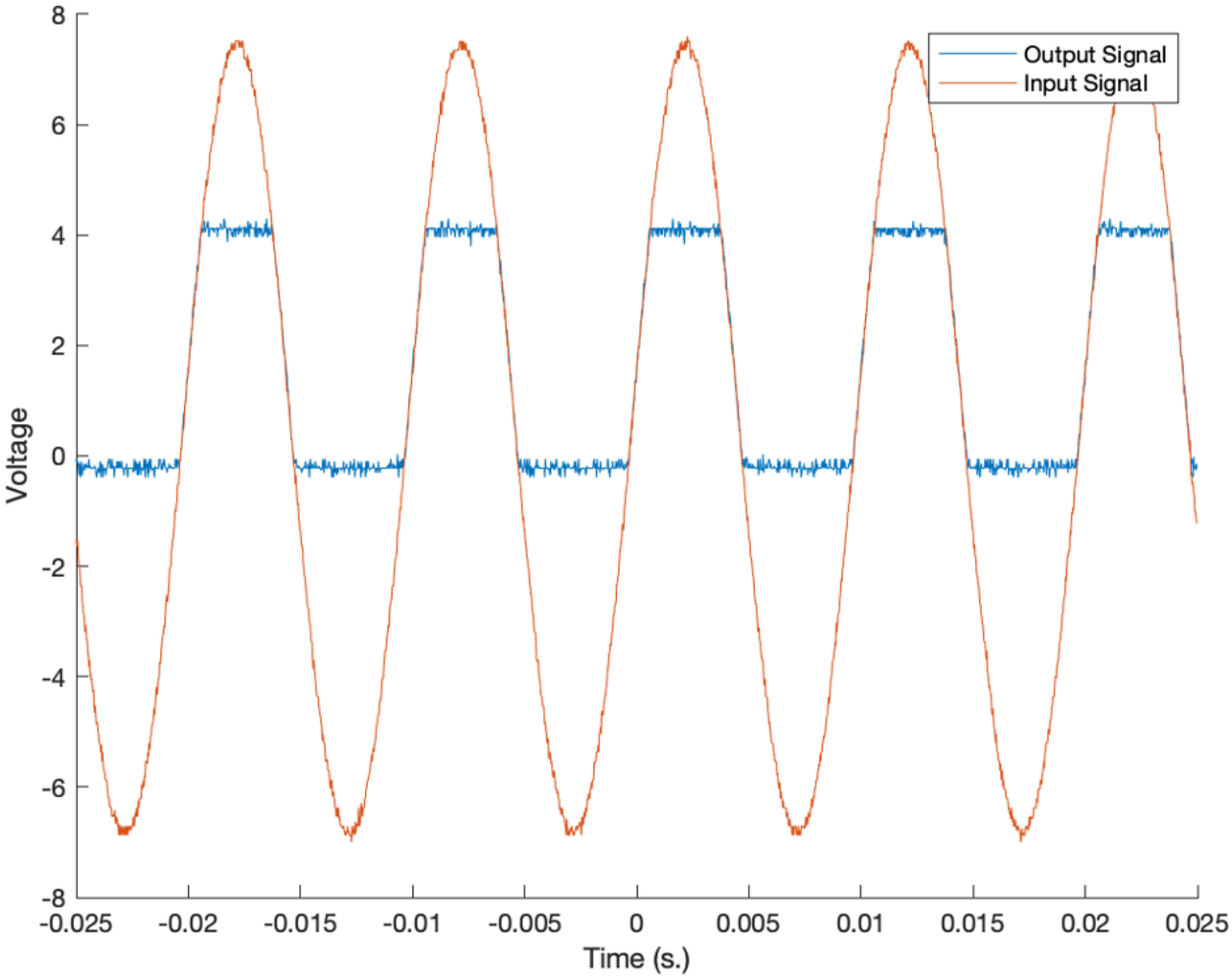


Figure 28: 15V input at 100Hz on Analog Channel 1

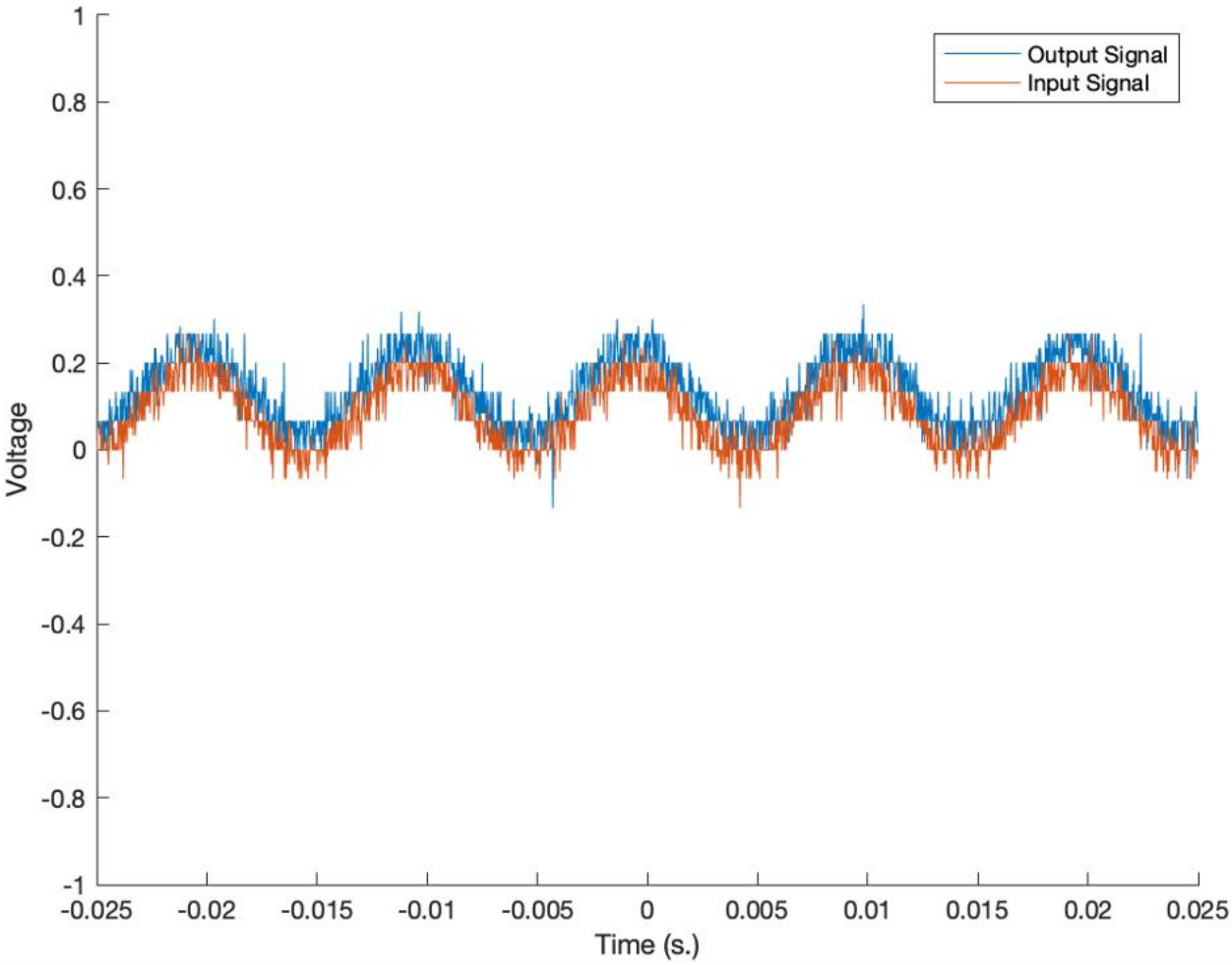


Figure 29: 200mV input at 100Hz on Analog Channel 1

The analog inputs were also tested at low (DC signal), medium (100 Hz), and high (35 kHz) signals. Shown below in Figures 30, 31, and 32 are representative plots from analog channel 1 that shows the different frequencies with a 3.3V signal. Figure 30 shows a 3.3V DC input which behaves as expected with minimal signal loss. Figure 31 shows much the same as Figure 27 above. Figure 32 for the high frequency signal shows some problems, however. The input signal is phase shifted and damaged, which shows that the signal conditioning circuit still has some problems at higher frequencies

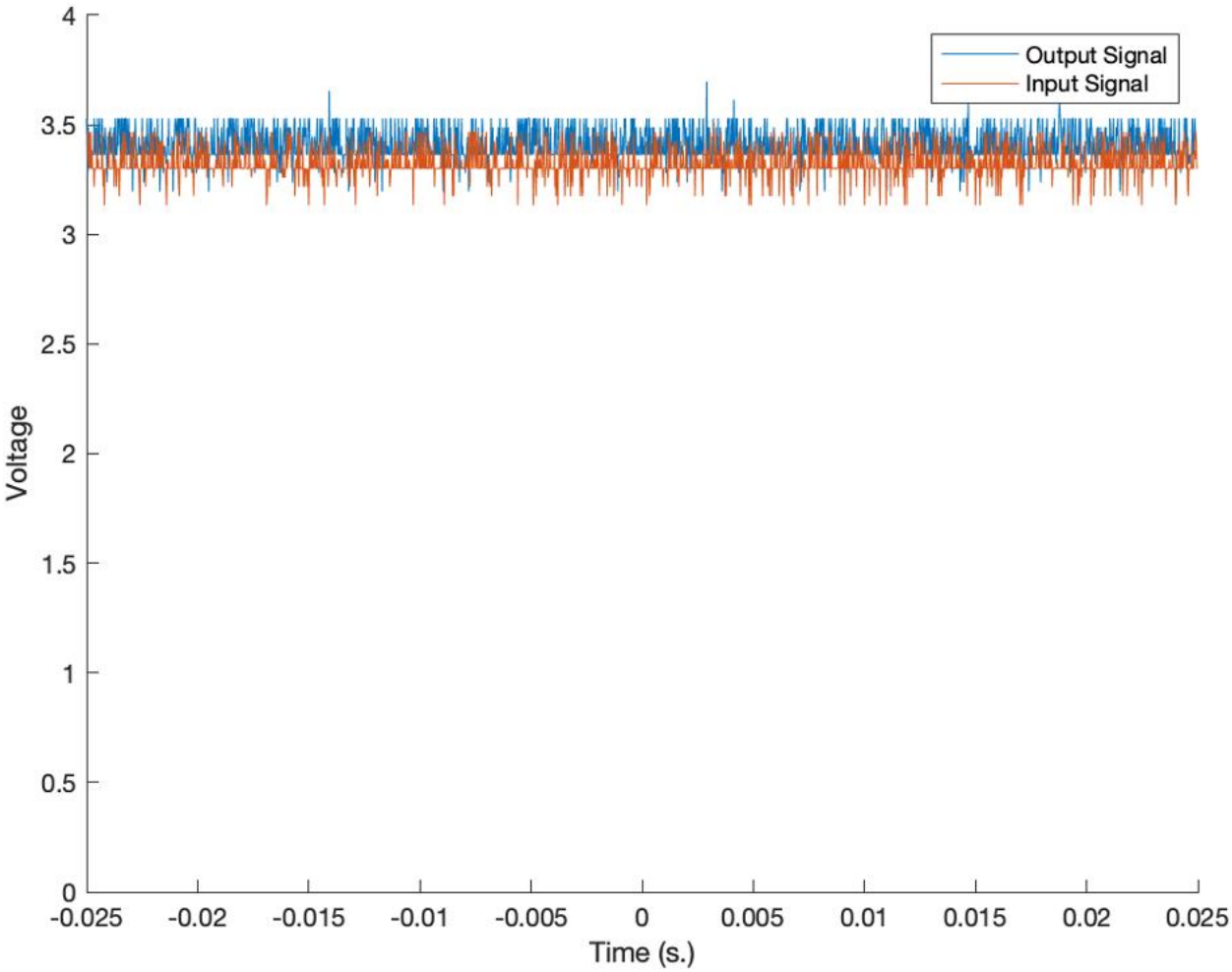


Figure 30: 3.3V DC Input on Analog Channel 1

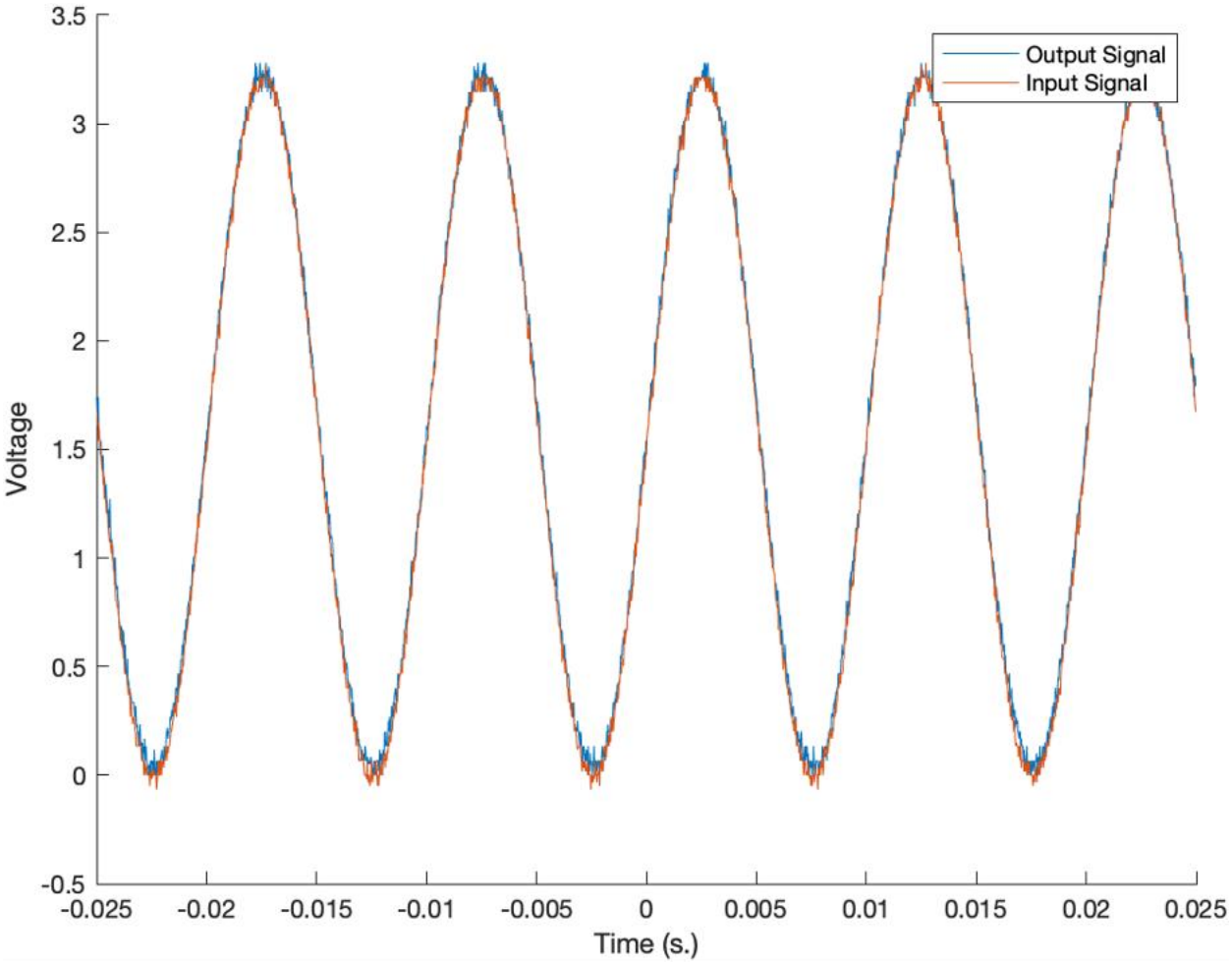


Figure 31: 3.3V 100Hz. Input on Analog Channel 1

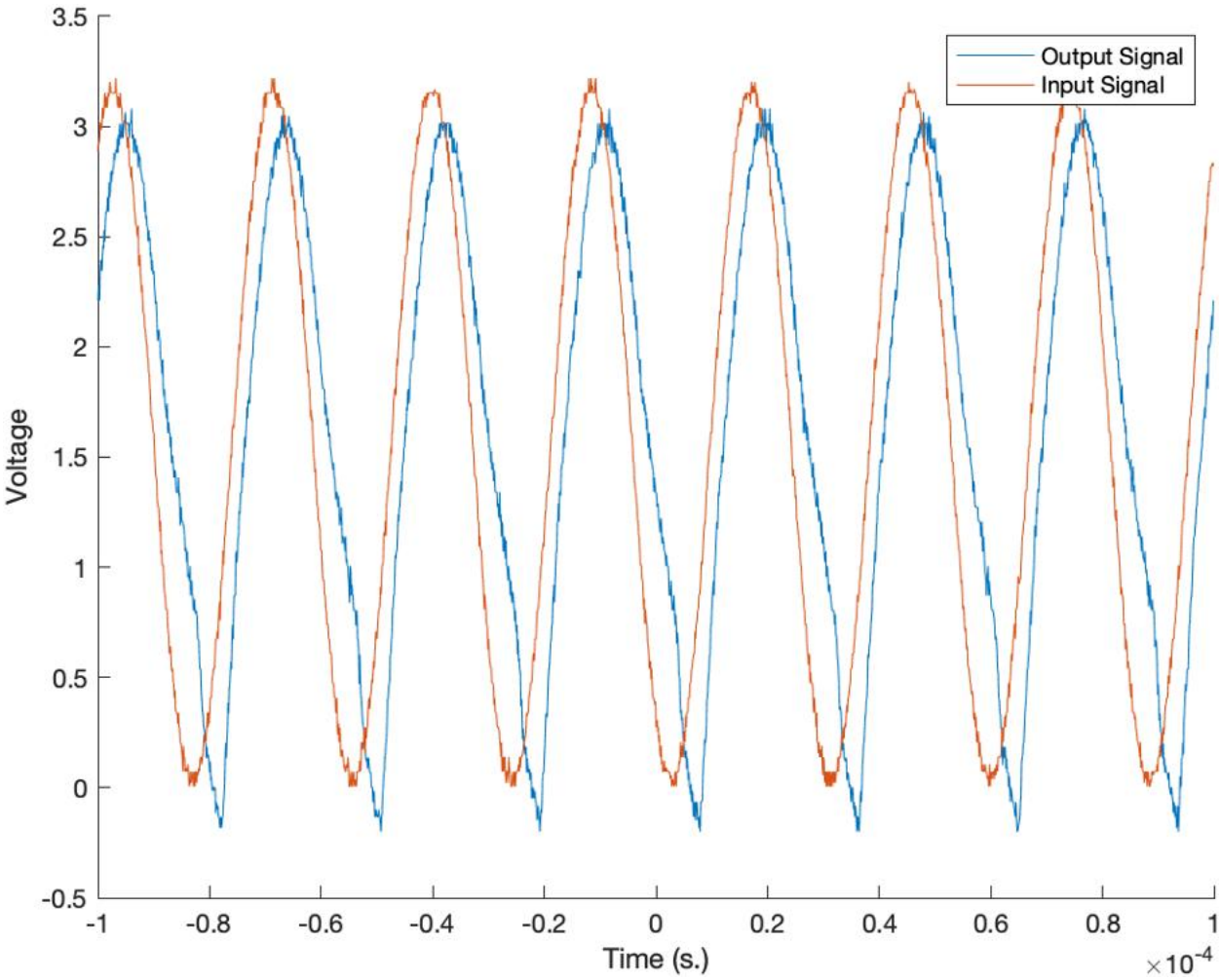


Figure 32: 3.3V 35kHz Input on Analog Channel 1

The ± 10V analog input option was tested as well with following similar guidelines to that in Figure 26, except the voltages tested were 3.3Vpp, 10Vpp and 15Vpp with the three frequencies remaining the same. Figures 33, 34, and 35 show representative plots for 3.3Vpp at varying frequencies, Figures 36, 37, and 38 show representative plots for 10Vpp at varying frequencies, and Figures 39 and 40 shows representative plots for 15Vpp at 100 Hz and 35kHz. No DC voltage is shown for 15V since the function generator could not produce a DC voltage that high. Regardless, there is one clear problem with each signal, and that is the fact that regardless of the input, the output signal is biased around 4V, even though the signal is truncated to the appropriate 0-3.3V range. This problem renders the ± 10V analog input option useless on the current PCB since using it would damage the STM microcontroller. Additionally, as seen in Figures 35, 38, and 40, the outputs for high frequencies inputs lose their signal coherence as in the 3.3V case. The output voltage is approximately 4V when overvolted, which is not consistent with what was seen during preliminary testing on the breadboard. Further testing will be needed to identify the underlying cause of this inconsistency.

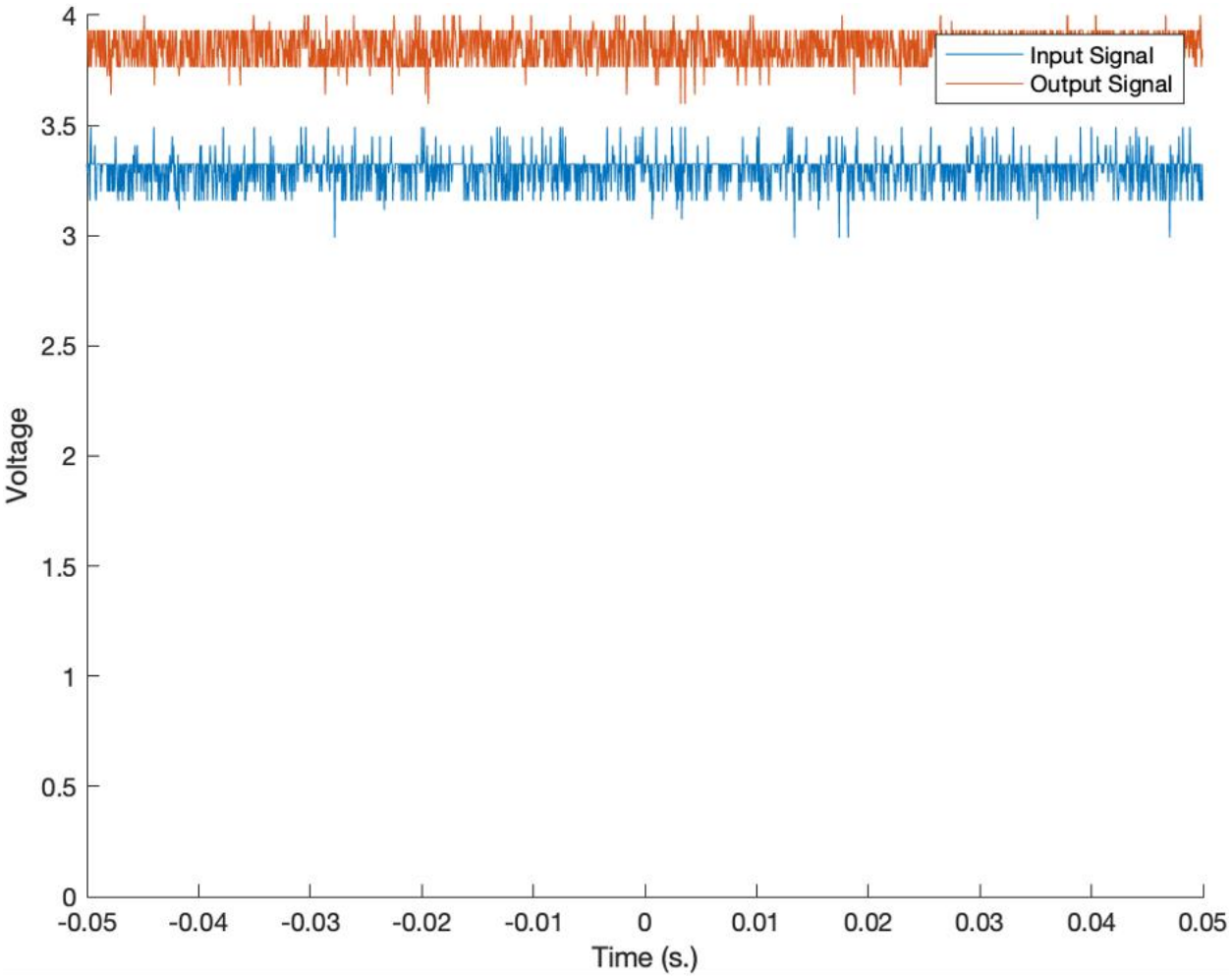


Figure 33: 10V DC Input on Analog Channel 1

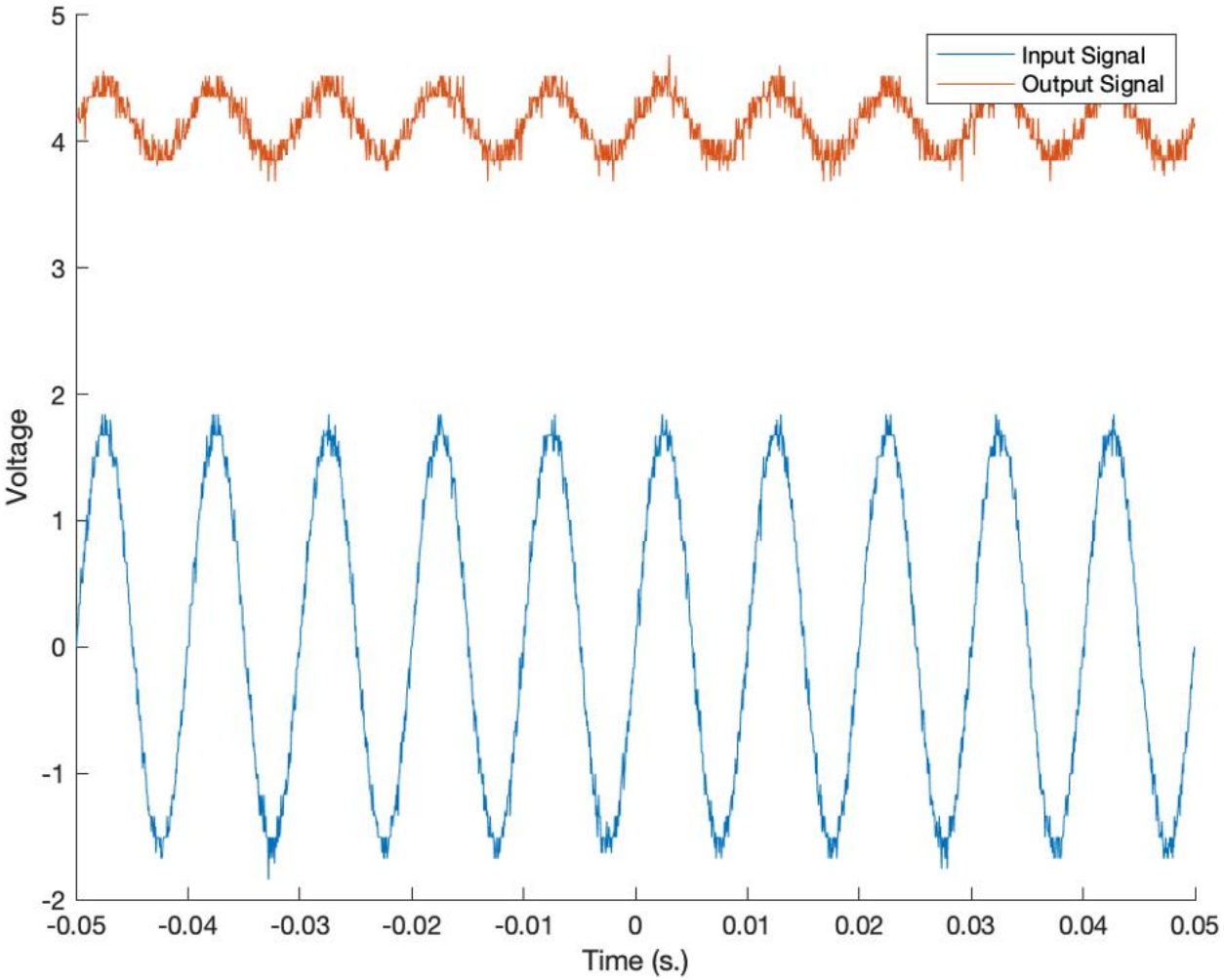


Figure 34: 10V 100Hz. Input on Analog Channel 1

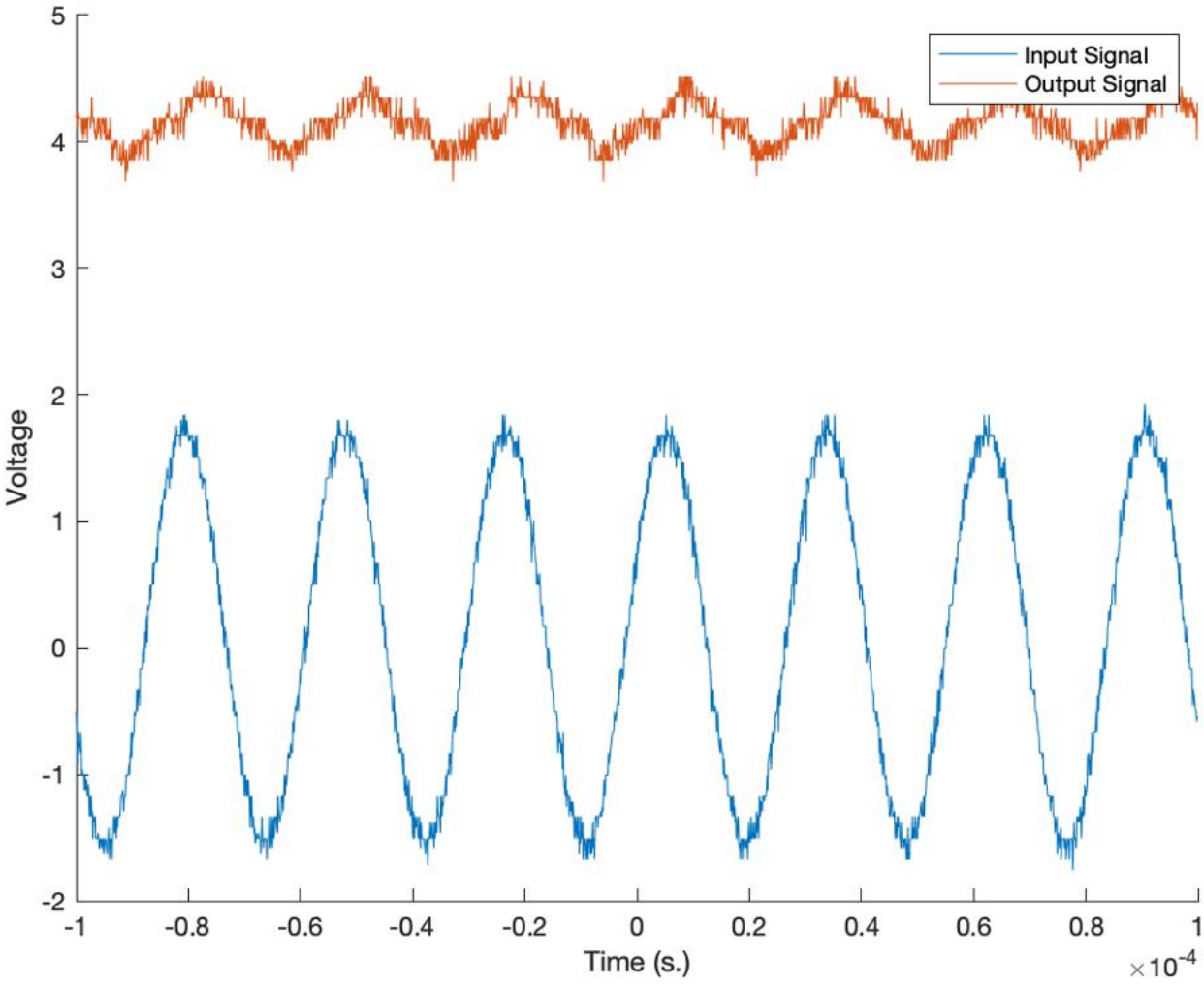


Figure 35: 10V 35kHz Input on Analog Channel 1

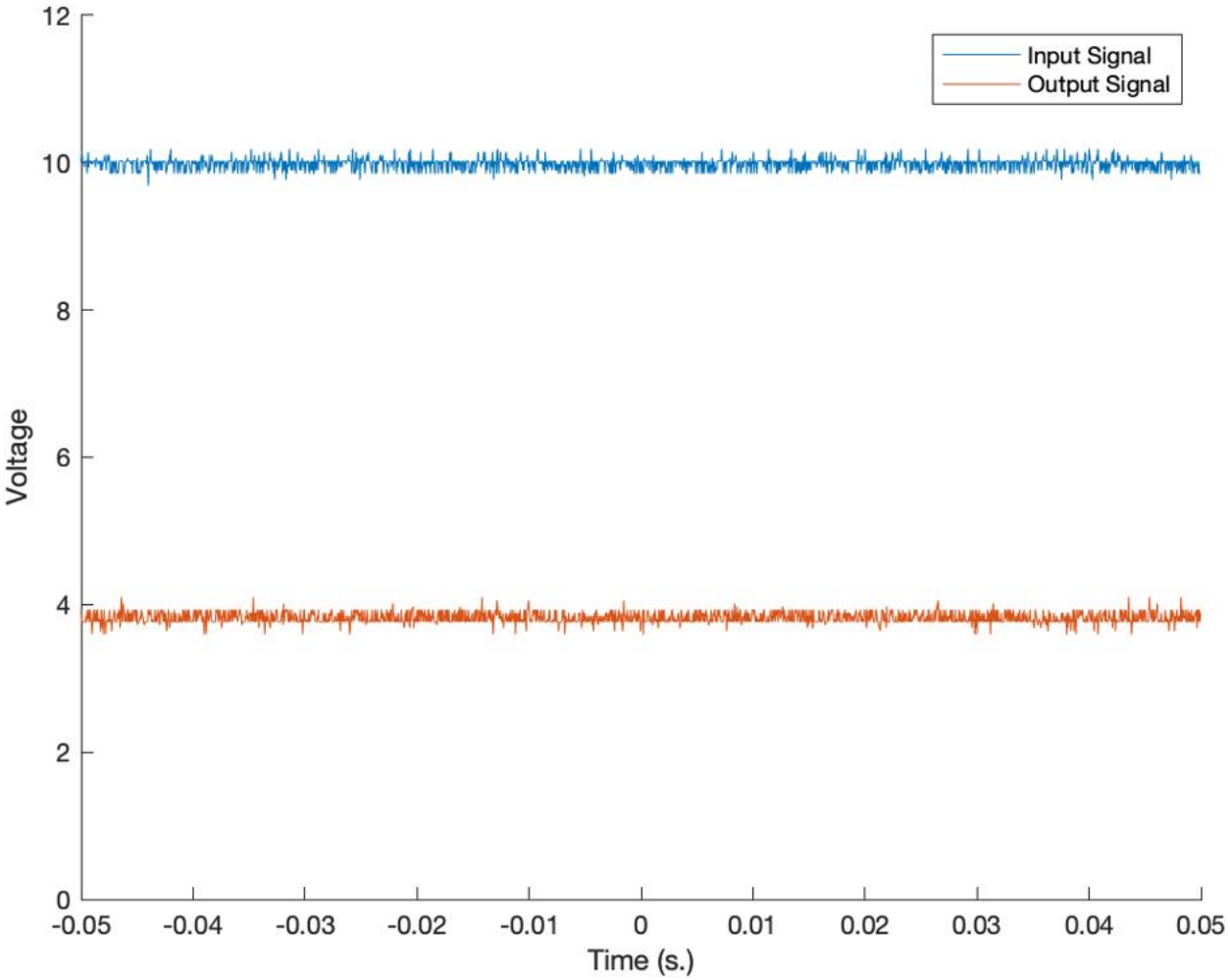


Figure 36: 10V DC Input on Analog Channel 1

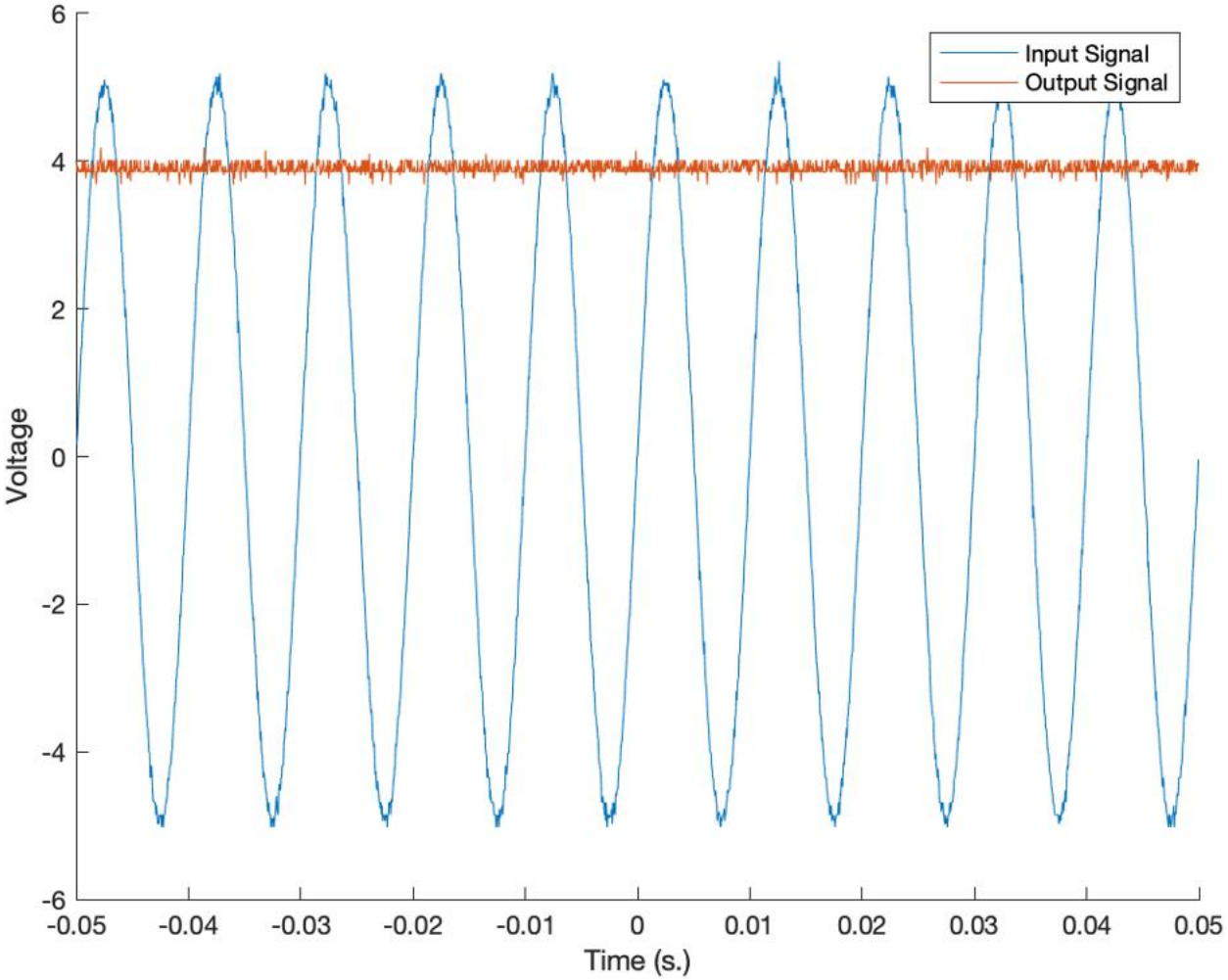


Figure 37: 10V 100Hz Input on Analog Channel 1

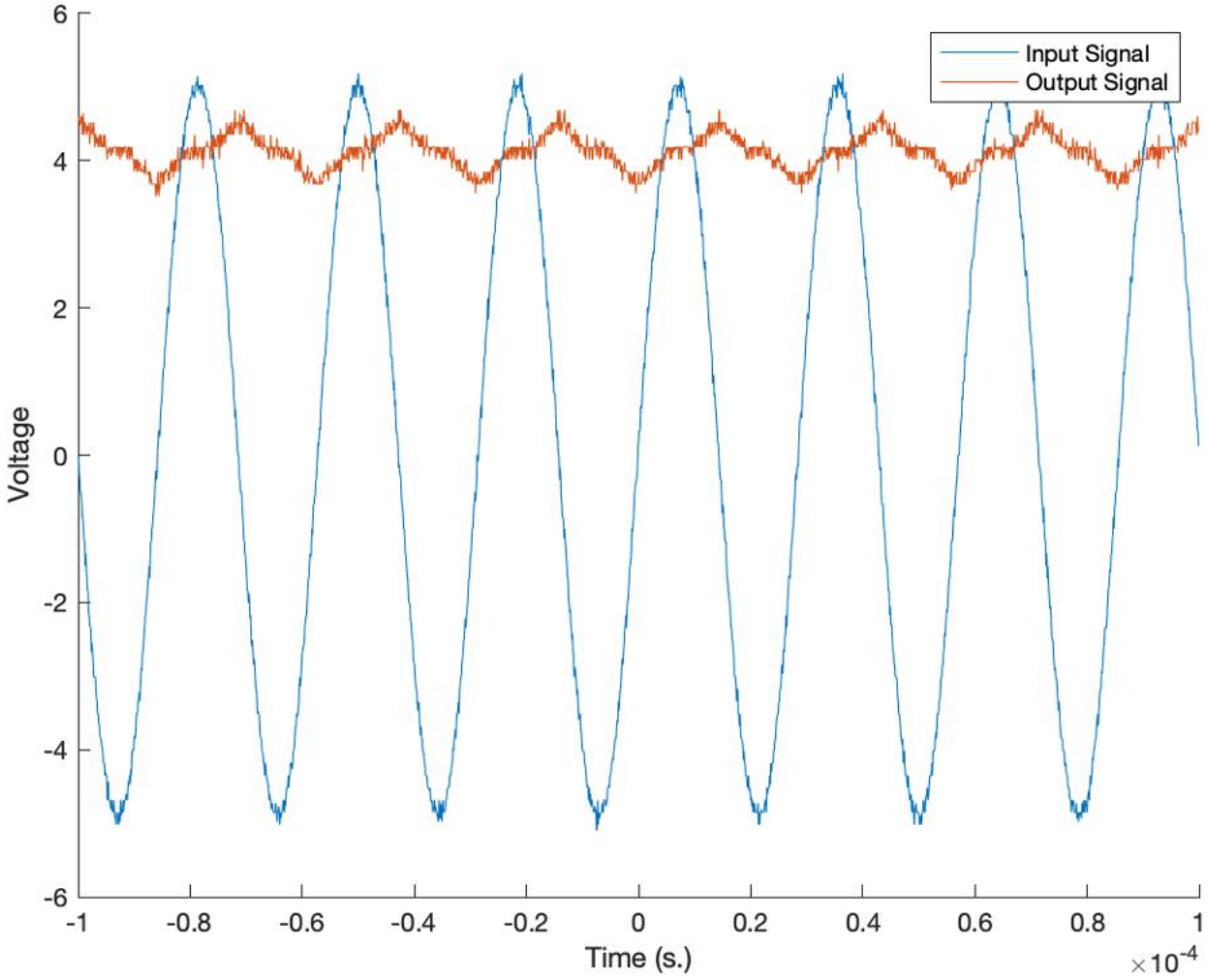


Figure 38: 10V 35kHz Input on Analog Channel 1

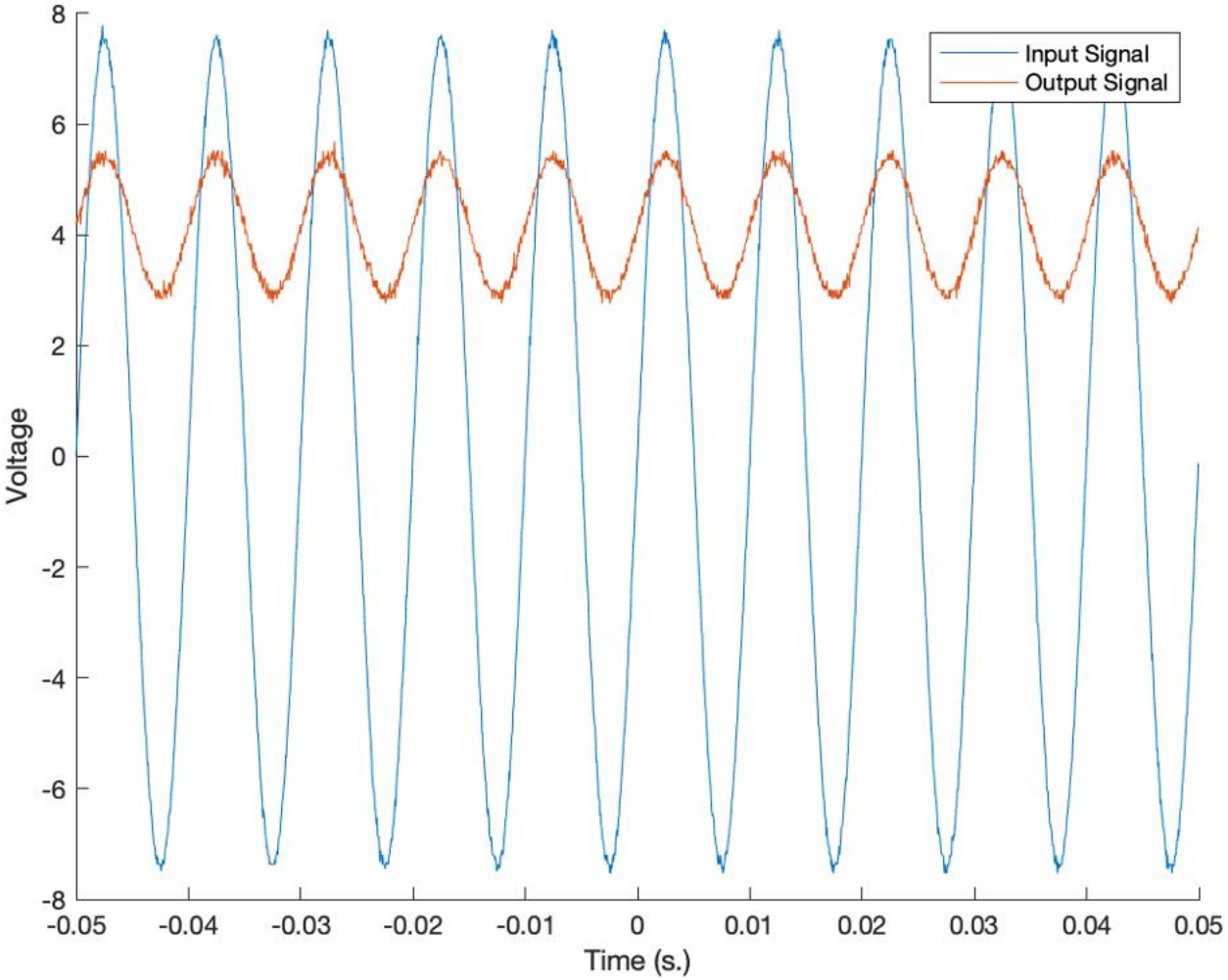


Figure 39: 15V 100Hz Input on Analog Channel 1

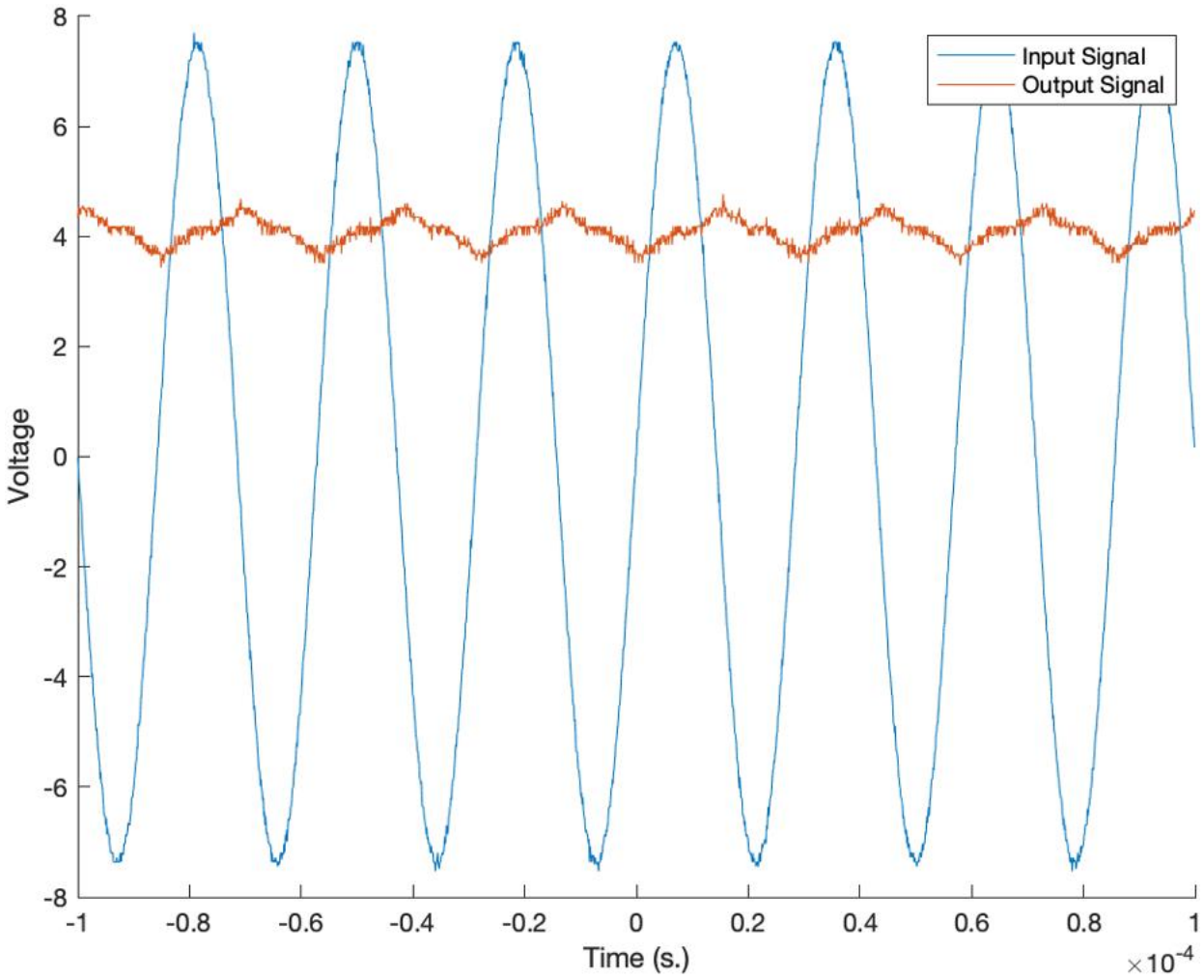


Figure 40: 15V 35kHz Input on Analog Channel 1

Figure 39 demonstrates the ability of the signal processing to be conducted correctly. Optimally, a 20V input (ranging from -10V to 10V) would have been used, although the capabilities of the function generation were limited. An interesting features of Figures 33 and 36 are that the DC steady state rests just below 4V. This would indicate a problem with the biasing, which is only meant to bias the signal by .825V. This would indicate a problem with the circuit at this location. Measuring resistance values around the shunt regulator revealed abnormalities. Further testing will need to be conducted to identify the underlying problem with the bias voltage and correct it. It would also appear from Figure 40 that the low pass filter may be damping the output signal at high frequencies too much. It would be worth using lower capacitor values to reduce this dampening effect.

Digital Input/Output Tests

Once again following the guide in Figure 41, the digital inputs were tested for both 3.3V and ± 10V digital inputs. Each signal was also tested at two different frequencies, 100Hz and 35kHz. For the 3.3V input option, Figures 41 and 42 show representative plots for a 3.3V and 15V input signal at 100 Hz respectively. It shows that the 3.3V input signal is minimally affected by the circuitry, while the 15V signal is effectively clipped at high and negative voltages in order

to protect the STM. However, as shown in Figure 43 and 44, the signal also begins to lose its coherence to the original input signal as in the analog case.

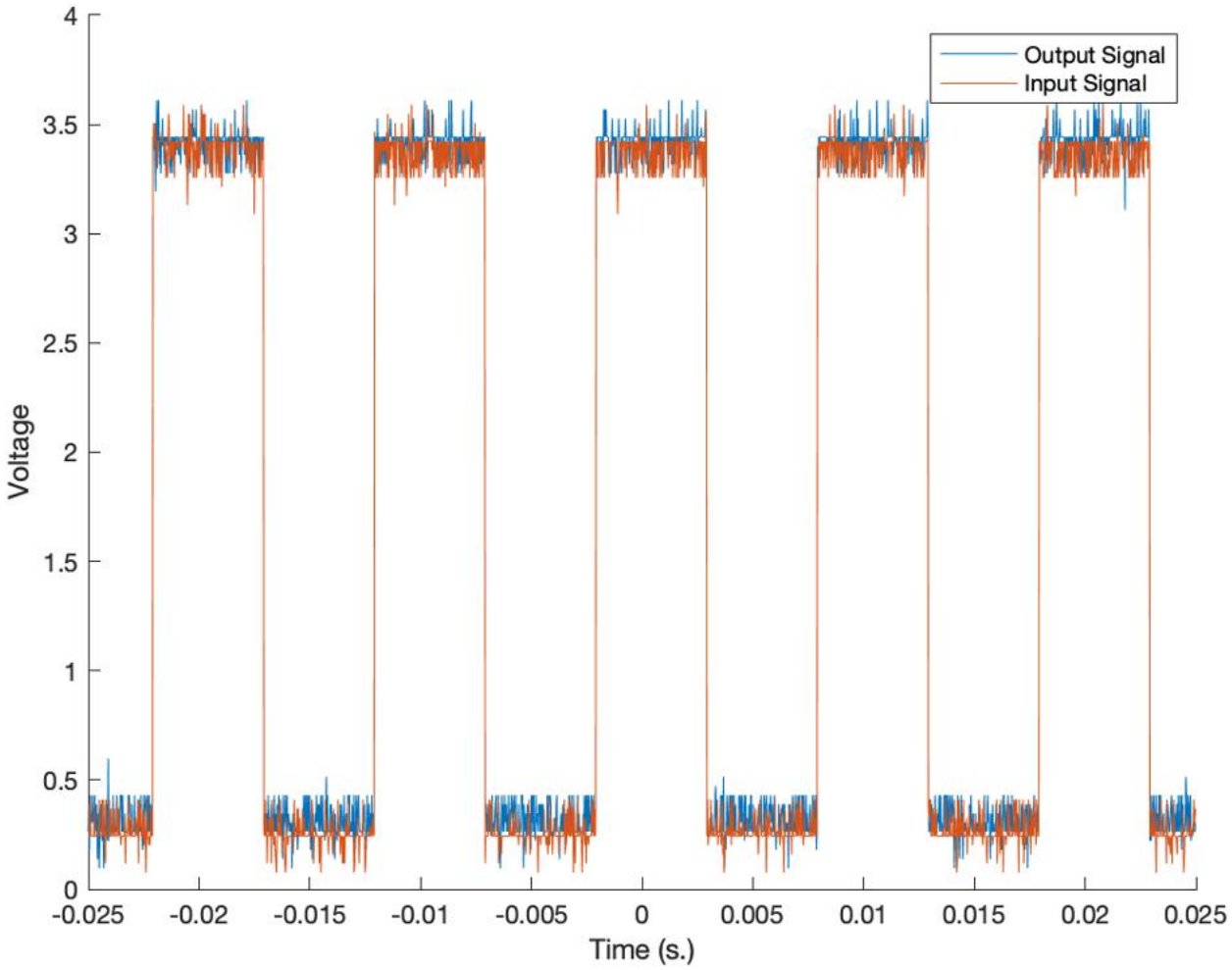


Figure 41: 3.3V 100Hz Digital Input on Digital Input 1

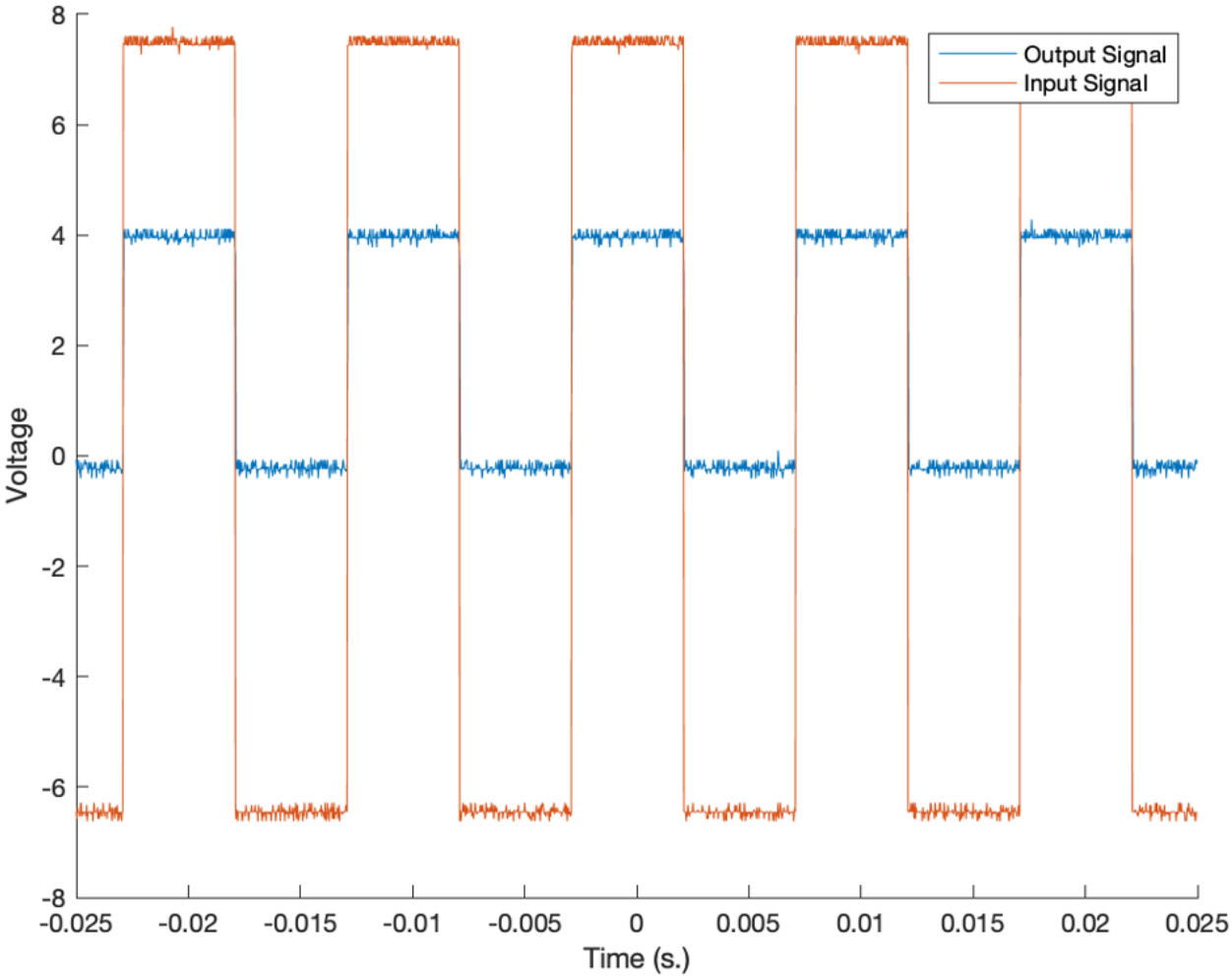


Figure 42: 15V 100Hz Digital Input on Digital Input 1

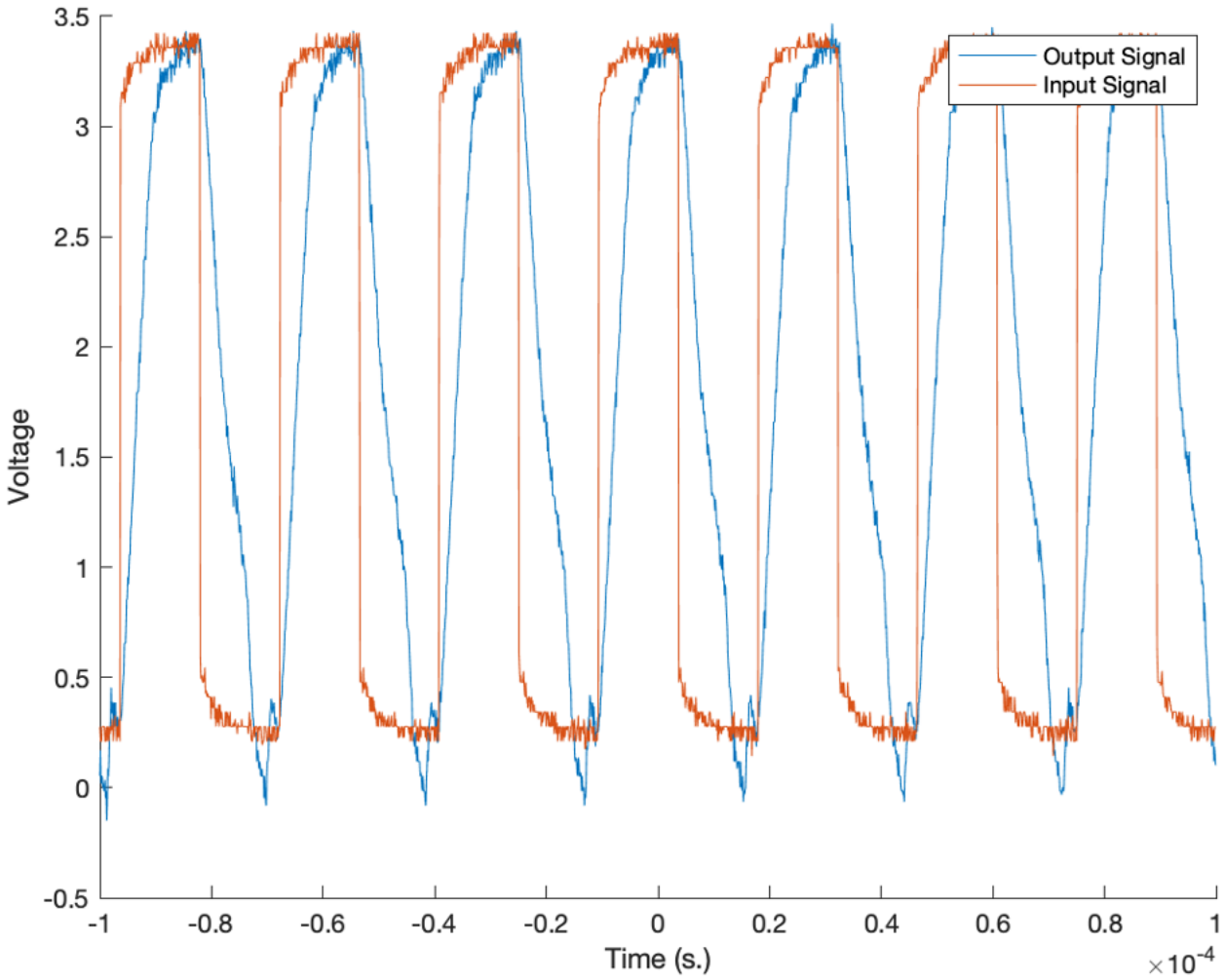


Figure 43: 3.3V 35kHz Digital Input on Digital Input 1

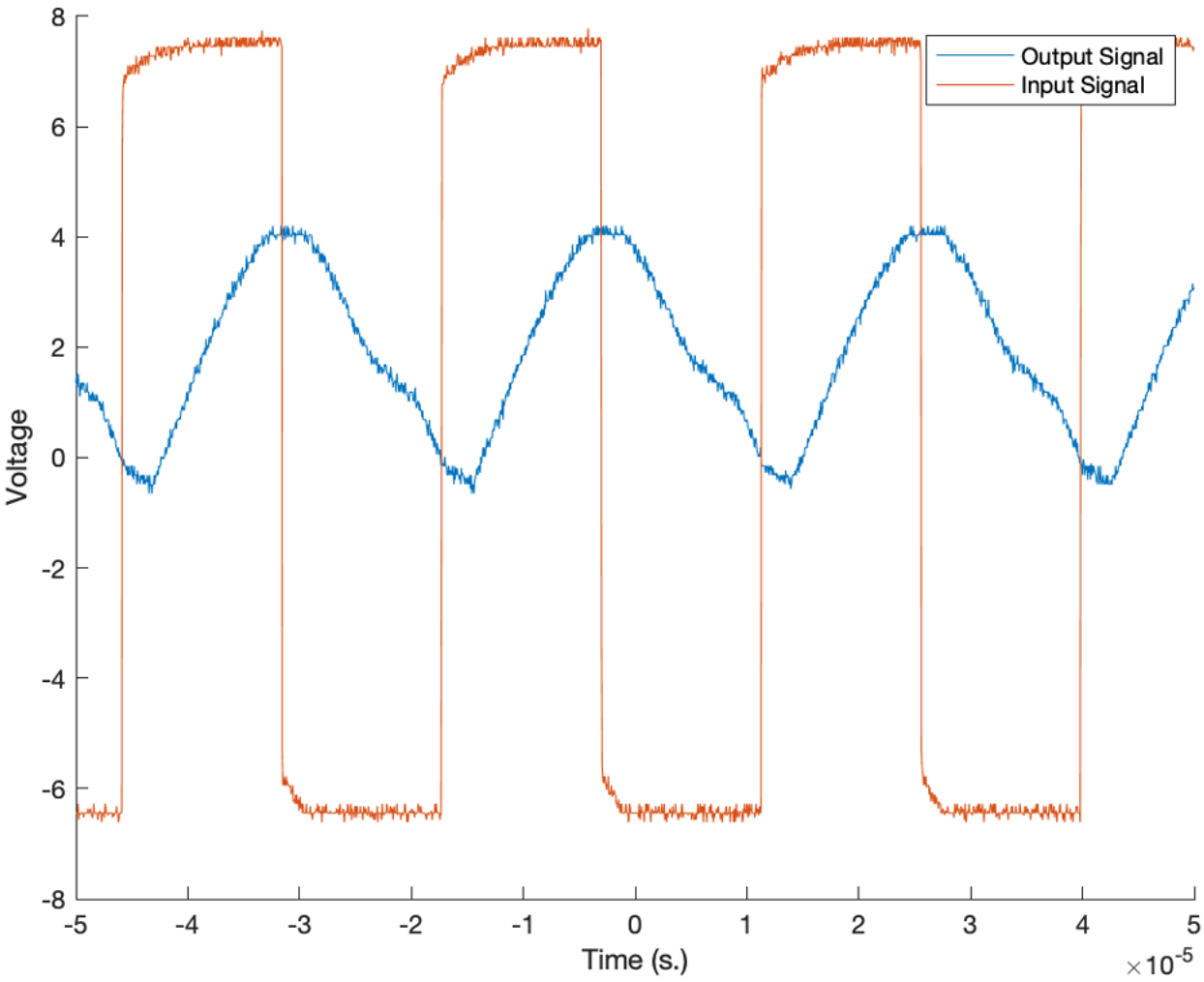


Figure 44: 15V 35kHz Digital Input on Digital Input 1

The ± 10V digital input option was tested as well. Representative plots at for a 10Vpp signal at 50 Hz and 2kHz are shown in Figures 45 and 46 for digital channel 1. Once again, the signal suffers from the same bias problem as the analog inputs, even though the output is properly truncated. Additionally, the high frequency signal in Figure 46 also suffers from the decoherence problem that all the other high frequency inputs have suffered from thus far.

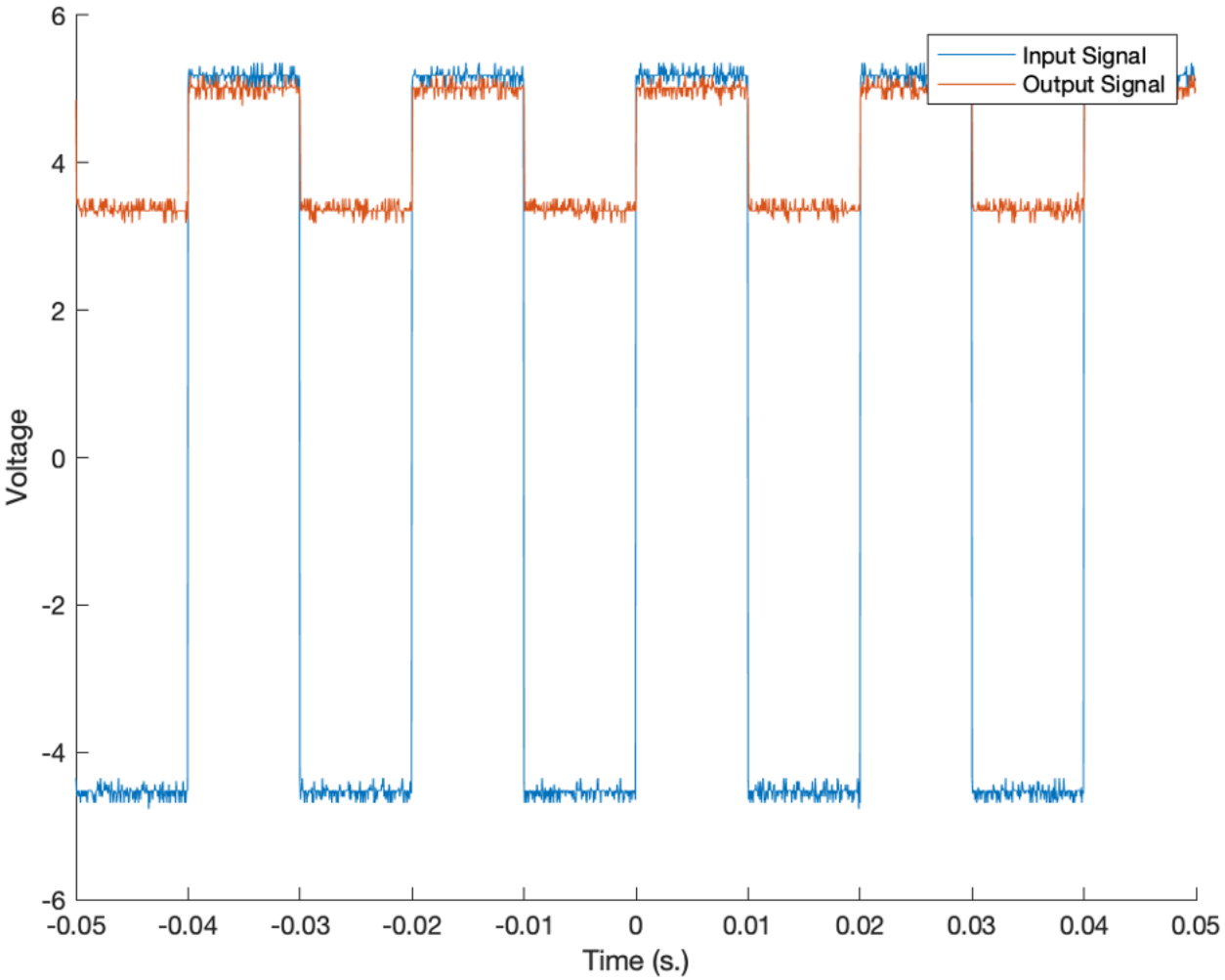


Figure 45: 10Vpp 50Hz Digital input on Digital channel 1

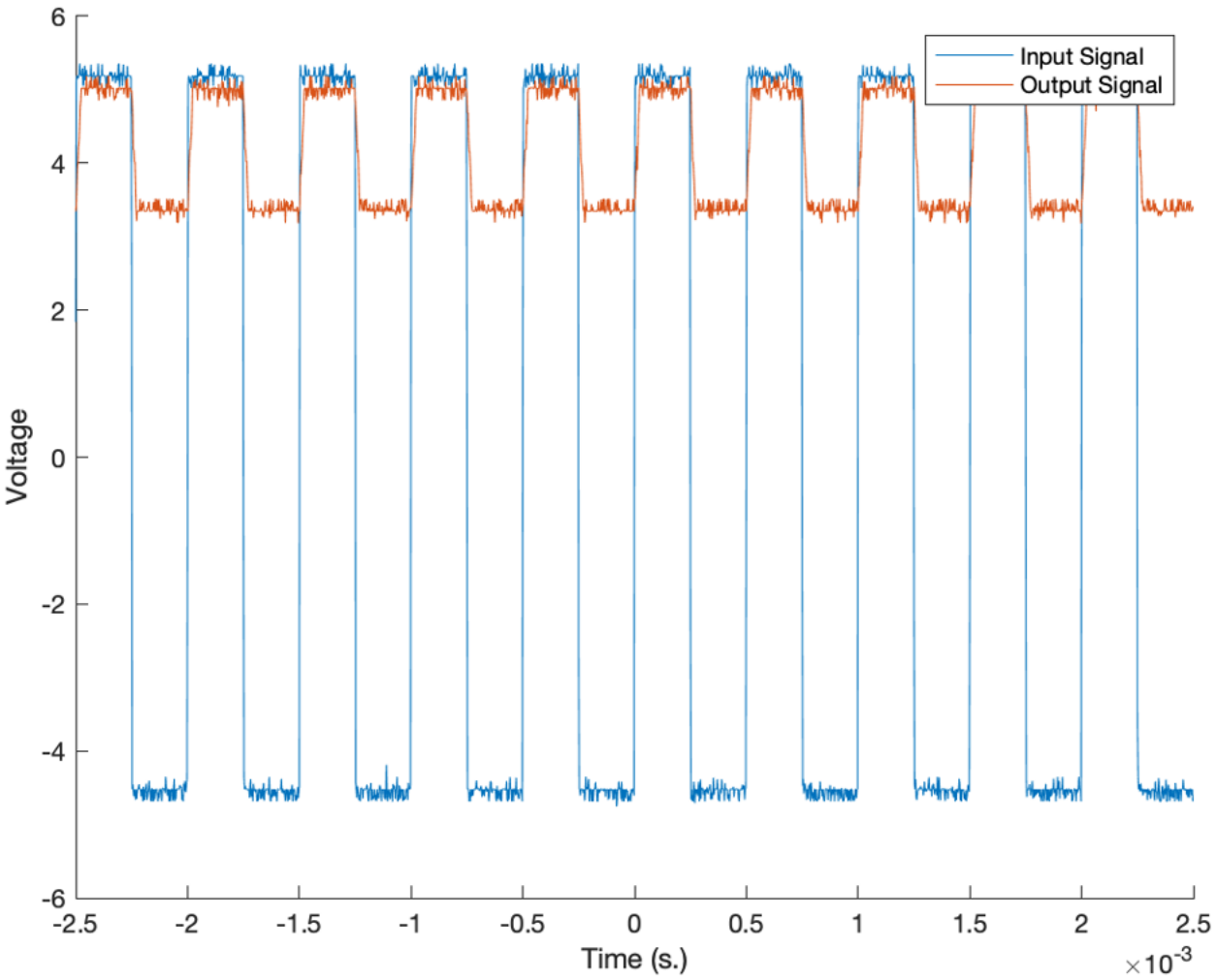


Figure 46: 10Vpp 2kHz Digital input on Digital channel 1

The digital output capabilities were also tested using a couple of LEDs. The digital channels sent pulses to the LEDS to flash on and off in a particular order. This was done to verify the verify the functionality of the digital outputs.

Function Generator and PWM Tests

Both the function generator and PWM outputs were tested. The Function Generator was able to output sine waves of varying amplitude and frequencies as intended. The PWM was also able to output pulses at varying duty cycles and frequencies. A sample PWM signal is shown in Figure 41.

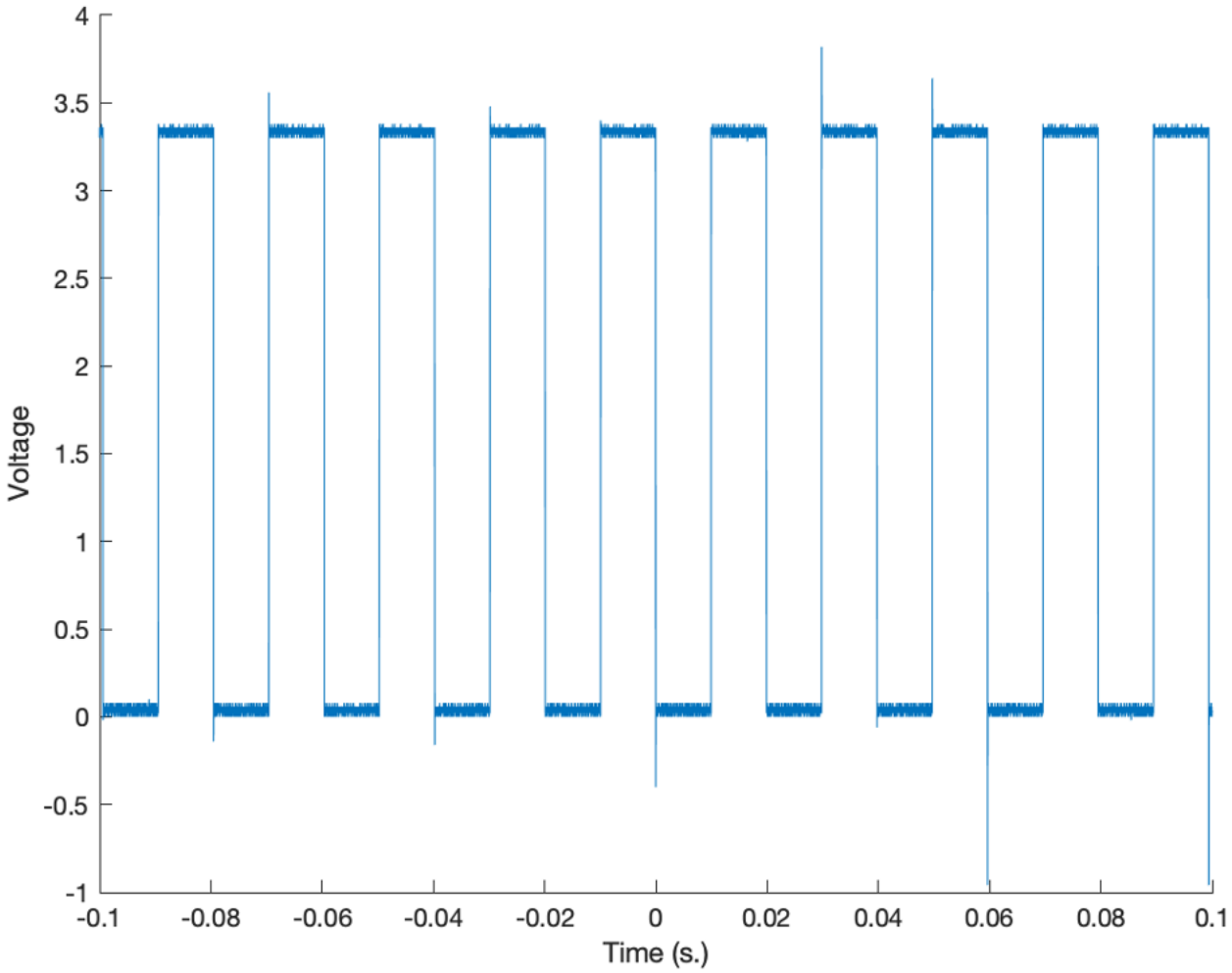


Figure 47: A trace of the PWM output

Encoder and I2C Tests

The Encoder and I2C capabilities are not implemented at the time of this report. The encoder circuit is fully functional and should be relatively easy to implement on the STM microcontroller side. The I2C circuit and STM microcontroller functional is much more difficult to implement and as such has not been tested or implemented. The functionality of these inputs is a future goal of this project.

VIII. Applications, Constraints, and Future Work

The STM-Based DAQ can also be used for general-purpose labs outside of Georgia Tech Instructional Labs. Broader applications considered include implementation in instructional labs at other universities and the provision of DAQ kits to future GT Capstone Design Expo groups as simple, plug-and-play systems with which to collect data to facilitate their R&D processes. In this application, the STM-Based DAQ offers more technical capability than products like Arduino, but a more beginner-friendly package than an STM32, at a significantly lower cost than existing DAQs like the NI ELVIS III.

As a USB connection is required to program the device, and due to constraints in what was technically feasible before the Capstone Expo, the device, as of the publication of this report, communicates with the server with a USART connection carried over USB, rather than with the planned Ethernet connection. For cybersecurity purposes relating to both student academic integrity and institute security from external factors, the transition to Ethernet-based communication remains a top priority for future development. This is planned to be implemented using the TCP/IP stack, chosen over the UDP stack because it is not susceptible to packet loss. The devices must be able to perform their duties without a USB connection, although a USB will still be required to alter DAQ firmware.

Another core feature on the horizon is the inclusion of I2C sensors, which rely on the same digital pins that have already been proven but have not yet independently demonstrated full functionality. This implementation is possible with modification only to the device firmware, and, while potentially tedious, is not expected to pose a significant technical challenge.

A future design team with a larger prototyping budget can investigate buying very low tolerance electrical components. This will cause the circuit elements to behave more consistently with the Spice circuit models, which will allow for overall cleaner data. Secondly, the board can be improved with better operational amplifiers with reduced noise. This will allow for more accurate data acquisition than was possible with prototyped Protector Board circuit.

An experienced team can improve this project by using a Field Programmable Gate Array (FPGA), which will allow for more parallel computing, resulting in faster data acquisition and on-board analysis.

One notable constraint of this device is that it cannot analyze very high frequency signals in the MHz range. This makes it used in some more advanced applications limited. Another constraint is that modification of the I/O pins is not trivial. Adding more I/O pins will require changes to the circuitry and firmware.

IX. Conclusions

Summary Conclusion

This project is designed around the premise of making a DAQ system using the STM32H7 Microcontroller that can fulfill all necessary requirements of the experimental setups in the ME 4056 systems lab. The STM based DAQ is also being designed on the basis of open-source use, meaning that it will remain relevant for a variety of test scenarios, both inside and outside the scope of the ME 4056 curriculum.

The goal of this project is to develop a robust STM based DAQ system with ease of manufacturing and ease of repair. The customer requirements are that the DAQ must have x10 analog inputs, x8 digital inputs/outputs, x2 PWM, x2 signal generators, x2 Encoder readers, and x2 I2C communication ports. These criteria were based on the previous NI DAQ system, requirements of the current lab setup, and discussion with the sponsor. This design challenge is being approached using the STM32 microcontroller. It is compatible with all of these imports. To ensure robustness, rectifiers, operational amplifiers, and Zener diodes will be used for signal conditioning and over-voltage/-current protection. All of the previously discussed inputs and outputs will be easily accessible with BNC connectors of the top of the DAQ system. The data will be collected by the STM32 microcontroller, processed and packaged, and then exported through ethernet to a cloud-based GUI. This setup will allow any student laptop that can access the Georgia Tech cloud to access the GUI to both see and send information to the DAQ system. Ease of manufacturing is also considered and implemented with a 3D printable PLA case, and only two custom PCBs, making up the majority of the device.

Ease of repair is ensured by implementing IC sockets, allowing for easy repair of any destroyed IC components. The STM32H7 microcontroller will be replaceable, and there will be access to the programming port to reload firmware if necessary. The DC-DC Converter and breaker will be commercial off the shelf, allowing for easy and safe repair of power supply components. The case CAD file and PCB Gerber files will be available for remaking components in case of damage or destruction.

Validation

Significant progress was made in validating the STM32-based DAQ system through iterative hardware and software development. The 0–3.3V analog and digital input circuitry is now fully operational, successfully interfacing with the STM32 and meeting required signal fidelity. Digital outputs, function generator, and PWM output functionalities have also been verified, with adjustable amplitude and frequency working as intended.

The ±10V analog signal path has undergone preliminary testing, with basic functionality confirmed; however, further tuning is needed to ensure reliable operation across all input conditions. Similarly, encoder inputs have demonstrated partial success and are expected to be functional with minor adjustments to the firmware or conditioning circuit.

Conversely, I2C input remains the most challenging subsystem, currently requiring substantial development and debugging to achieve reliable communication. Despite this, the team has built a robust foundation with a working real-time data pipeline and modular architecture that can support future integration of the remaining inputs.

Overall, the DAQ system has reached a stable intermediate stage where core functionality is validated, and outstanding issues have clear pathways forward. With continued refinement, this system remains on track to meet all initial design specifications and support the full range of ME 4056 lab experiments.

Jack Hanling

Wrote the sections related to the STM and PCB in the report. Created many of the relevant graphics as well as the PCB design files.

Cooper Allen

Wrote the introduction of the report and assisted Jack with the STM portions of the report. Helped create many of the related graphics.

Christian Busch

Wrote the market research, existing products, and executive summary sections of the report. Wrote and worked on the sections related to the circuit design.

Michael McCabe

Wrote the sections related to the RC Car and hardware/sensors. Created the CAD and helped create the bill of materials related deliverables.

Mark Stevens

Assisted with RC Car sections and the CAD model of the case. Wrote the sections related and helped trim down sections related to the last report.

Arsh Suri

Wrote the sections related to the software in the customer requirements and created the related graphics.

References/Citations

[1] National Instruments. Accessed: Feb. 2, 2025 https://www.ni.com/en/shop/engineering-education/portable-student-devices/mydaq/what-is-mydaq.html

[2] National Instruments. Accessed: Feb. 2, 2025 https://www.ni.com/en/support/downloads/software-products/download.LabVIEW.html#559067

[3] National Instruments. Accessed: Feb. 2, 2025 https://digilent.com/shop/academic/

[4] Keysight Technologies. Accessed: Feb. 2, 2025 https://www.keysight.com/us/en/products/modular/data-acquisition-daq.html

[5] Tektronix. Accessed: Feb. 2, 2025 https://www.tek.com/en/products/keithley/data-acquisition-daq-systems

[6] National Instruments. Accessed: Feb. 2, 2025 https://www.ni.com/en/support/downloads/software-products/download.LabVIEW.html#559067

[7] Mathworks. Accessed: Feb. 2, 2025 https://www.mathworks.com/products/connections/product_detail/LabVIEW.html

[8] National Instruments. Accessed: Feb. 2, 2025 https://www.ni.com/gate/gb/GB_EVALTLKTLVARDIO/US

[9] Keysight Technologies. Accessed: Feb. 2, 2025 https://www.keysight.com/us/en/products/software/pathwave-test-software/benchvue-software.html

[10] Accreditation Board for Engineering and Technology. Accessed: Feb. 2, 2025 https://www.abet.org/about-abet/

[11] 1“Nucleo-H755ZI-Q,” STMicroelectronics Available: https://www.st.com/en/evaluation-tools/nucleo-h755zi-q.html

[12] LabEx. Accessed: Feb. 7, 2025. https://labex.io/tutorials/cybersecurity-how-to-build-a-docker-image-for-cybersecurity-server-simulation-purposes-414485

[13] Medium. Accessed: Feb. 7, 2025. https://medium.com/@kanithkar_baskaran/essential-features-of-django-that-make-it-a-developers-favorite-6c916692c890

[14] University of Delaware. Accessed: Mar. 14, 2025. https://www.eecis.udel.edu/~mills/ptp.html

[15] Chart.js. Accessed: Mar. 14, 2025. https://www.chartjs.org/docs/latest/

[16] https://deepbluembedded.com/stm32-dma-tutorial-using-direct-memory-access-dma-in-stm32/

[17] https://www.adafruit.com/product/270#technical-details

[18] https://learn.adafruit.com/st-9-dof-combo?view=all

[19] https://cdn-shop.adafruit.com/product-files/4541/C14641+C14642+C14643+datasheet.png

Appendices

Appendix A: Function Trees

Figure A1: Top of Function Tree

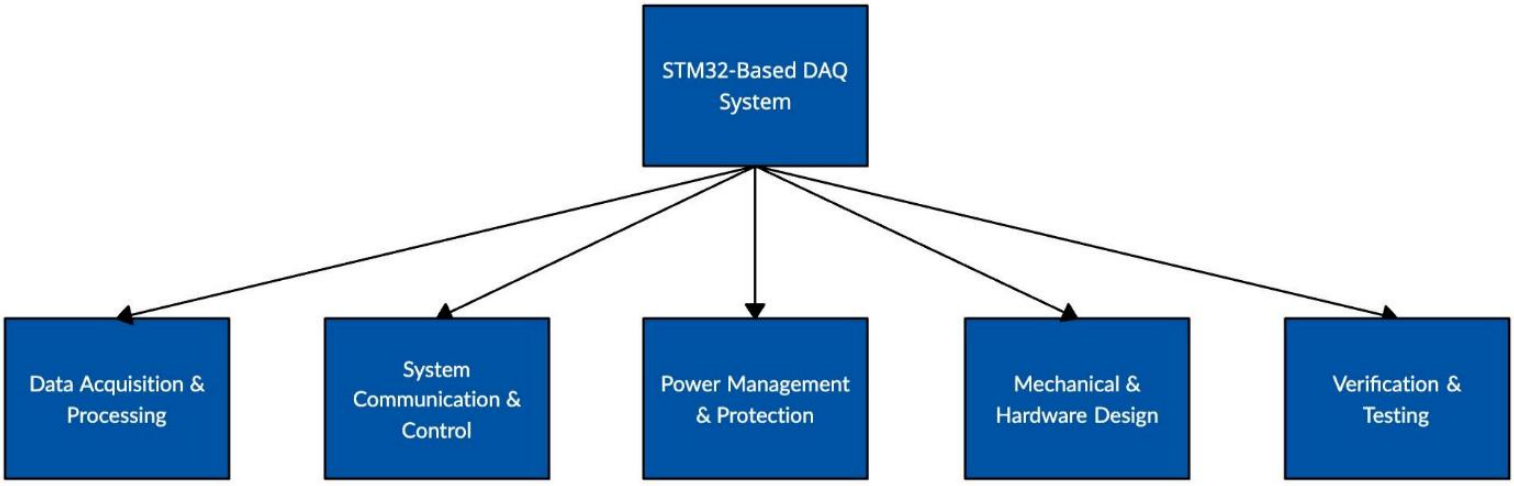


Figure A2: Data Acquisition & Processing Branch of Function Tree

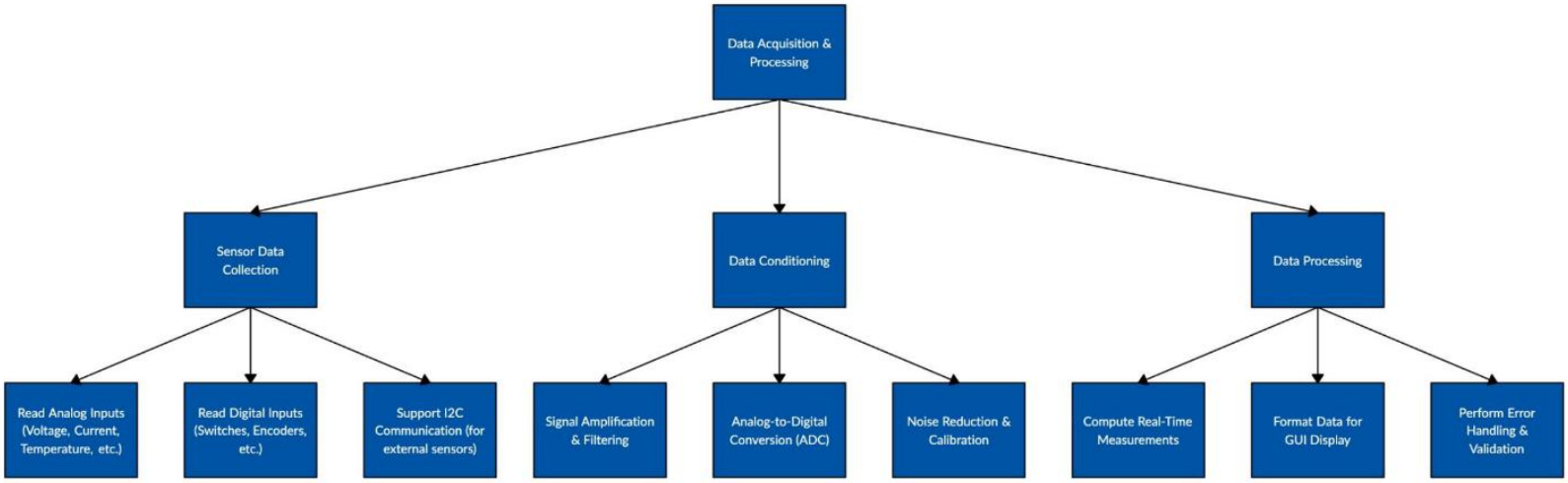


Figure A3: System Communication & Control Branch of Function Tree

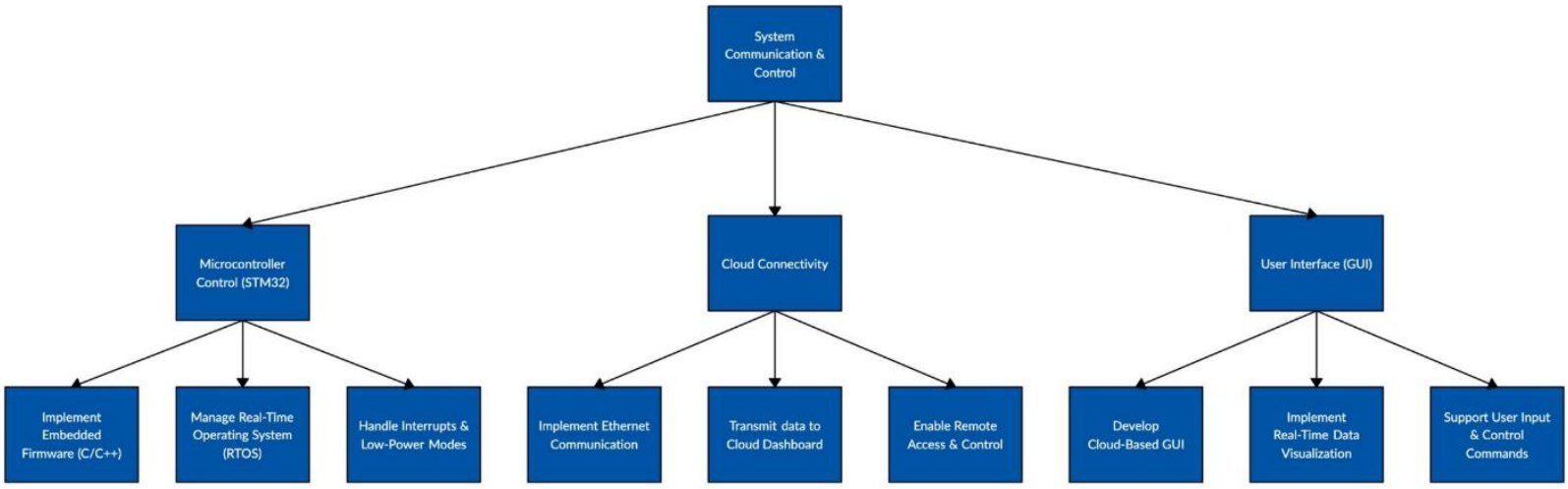


Figure A4: Power Management Protection Branch of Function Tree

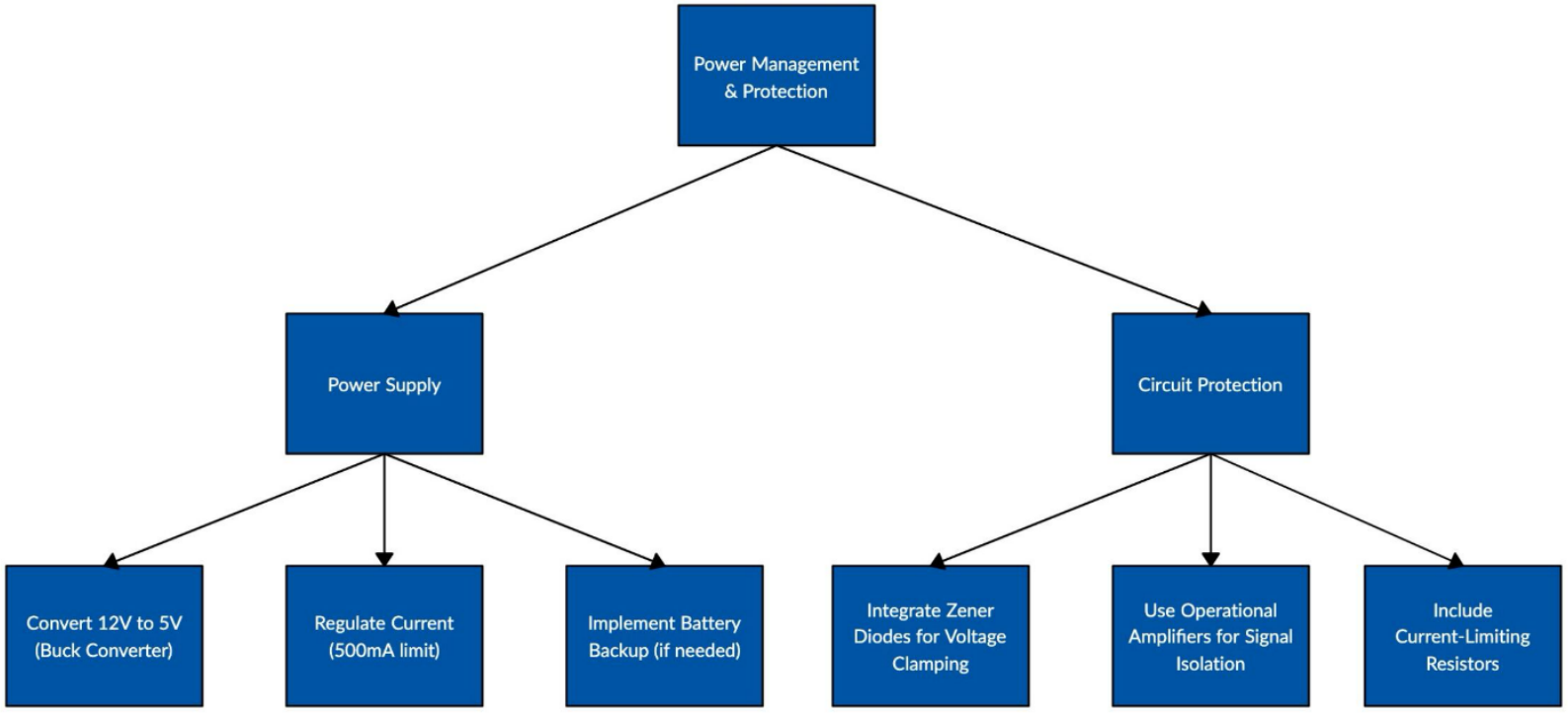


Figure A5: Mechanical Hardware Design Branch of Function Tree

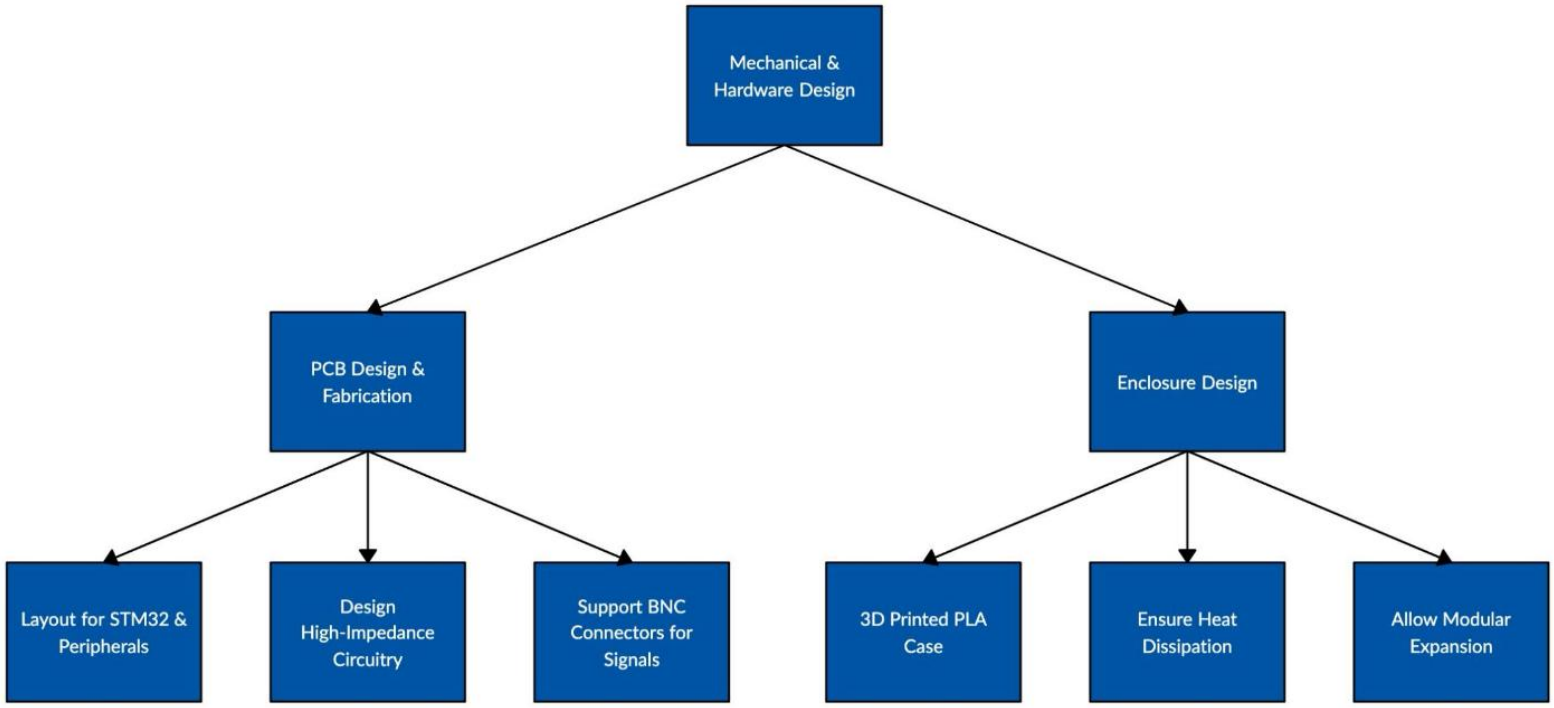
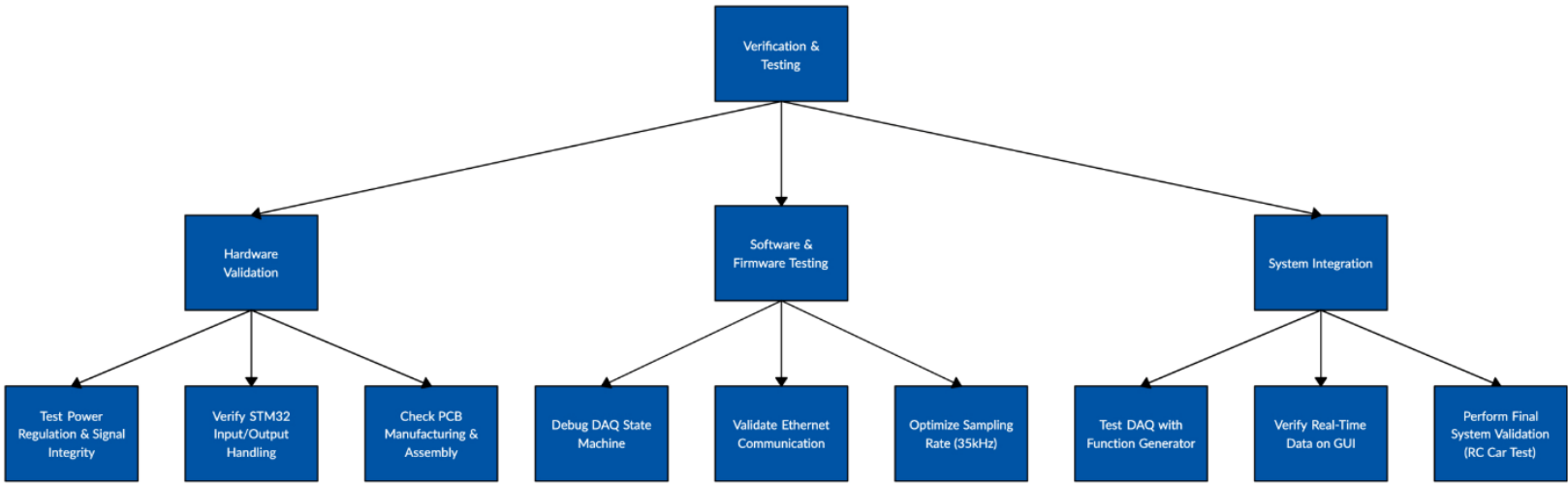


Figure A6: Verification & Testing Branch of Function Tree



Appendix B: Specification Sheet

Specifications			Issued: 1/28/2025	
		For: DAQ Subsystem		Page 1
Changes	D/W	Requirements	Resp.	Source
		DAQ Subsystem		
		Energy:		
2/2/2024	D	Power 12 V	JH/CA	Sponser
2/2/2024	D	STM32 Needs 5V and 500 Amp	JH/CA	Sponser
2/2/2024	D	5V References May Need 5V with Additional Current From Power Source	JH/CA	Sponser
		Material:		
2/2/2024	W	PCB for Circuits and Top	JH/CB	Team
2/2/2024	D	Metal BNCs for Connectors	JH/CB	Sponser
2/2/2024	W	3D Printed PLA Exterior	JH/CB	Team
		Circuits/Electronics:		
2/2/2024	W	Operational Amplifiers High Impediance - Current Protection	JH/CB	Team
2/2/2024	W	DC-DC Converter Voltage Conversion (12 V - 5 V) Voltage Conversion (12 V - 10 V)	JH/CB	Team
2/2/2024	W	STM32H7 MCU	JH/CA	Sponser
2/7/2024	W	Zener Diodes (5V and 10V breakdown voltages)	JH/CB	Team
2/7/2024	W	Rectifier	JH/CB	Team
		Communication:		
2/2/2024	D	Ethernet From MCU to Cloud	JH/CA	Sponser
		DAQ Inputs		
2/2/2024	D	Encoder PWM In Signals x2 5V Voltage and Ground References	JH/CA	Sponser
2/2/2024	D	I2C Communication x2 Register Code for Access (TBD) Baud Rate Max - 115200 bit/s	JH/CA	Sponser
2/2/2024	D	Analog Input x10 High Impediance Voltage Input (0-4V) 16 bit ADC resolution >35 Hz Sampling Frequency	JH/CA	Sponser
2/2/2024	D	Digital Input/Output x8 Low/High = 0V/5V Switch Speeds up to 5 kHz	JH/CA	Sponser
		DAQ Outputs		
2/2/2024	D	PWM Out Signals x2 Duty Cycle 0-100% Frequencies up to 50 kHz	JH/CA	Sponser
2/2/2024	D	Function Generators x2 12 bit DAC with 0-5V signal output Signal Frequencies up to 50 kHz	JH/CA	Sponser
2/2/2024	D	*Costs:* Minimize Price	All	Sponser/Team

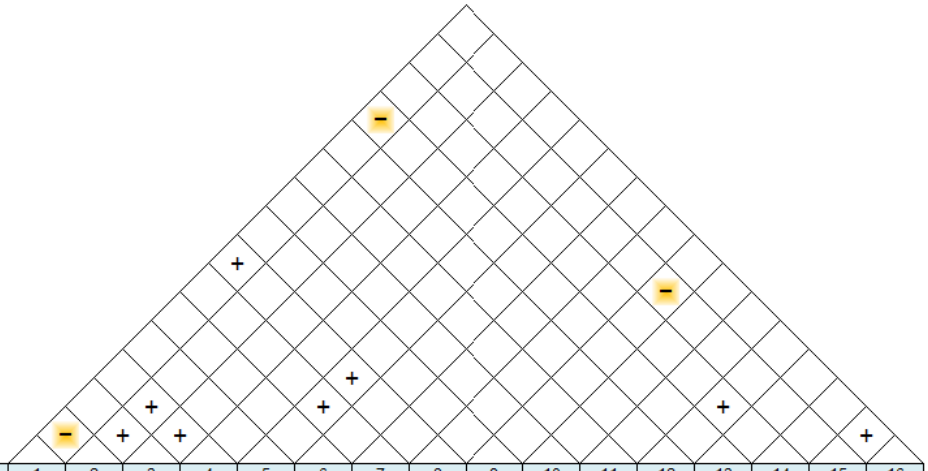
Appendix C: House of Quality

Project: DAQ Subsystem
Revision: 2/3/2025
Date: 2/3/2025

Correlations	
Positive	+
Negative	−
No Correlation	

Relationships	
Strong	●
Moderate	○
Weak	▽

Direction of Improvement	
Maximize	▲
Target	◇
Minimize	▼



Category	Weight	Customer Requirements (Explicit and Implicit) \ Engineering Requirements	1 Buck Converter 12V -> 5V and 12 -> 10 V	2 Zener Diode (5V and 10V Breakdown)	3 Operational Amplifier Before Input/Output	4 Rectifier	5 16 Bit ADC Resolution & 35 kHz Sampling Rate	6 5V/10V High and 0 V Low & Switch Speed 5 kHz	7 Handles 50 kHz PWM Frequency Input	8 Produces 50 kHz PWM Frequency Outputs	9 115200 bps Baud Rate	10 12 DAC Resolution	11 CAD Files and PLA 3D Printer	12 BNC Connectors and Customer PCB Plate	13 Labelled Jumpers for 0-5V I/O and +/-10V I/O	14 Printable PCBs with IC Connectors for Easy Repair	15 Agreed Standard Data Packaging and Transfer Routine	16 STM32H7 Ethernet Cable Output
Safety	9	STM32H7 Power Supply	●	○		▽												
	9	Input/Output Voltage Protection	○	●	▽													
	9	Input/Output Current Protection			●													
	9	Rectify Negative Input Voltages		▽		●												
Signals	8	Analog Inputs x10		○	○	○	●											
	8	Digital Input/Output x8		○	○	○		●										
	8	Encoder Input x2		○	○	○			●	▽								
	8	PWM Output x2		○	○	○			▽	●								
	8	I2C Communication x2		○	○	○					●							
	8	Function Generator x2		○	○	○						●						
Hardware Interface	6	3D Printable PLA Case											●					
	7	Signal Input/Output Interface												●				
	6	Jumpers (0-5 V Signals & +/- 10 V Signals)												○	●			
	7	Ease of Manufacture and Repair											○	○	○	●		
Ethernet	8	Signal Sampling and Packaging															●	
	8	Output via Ethernet									▽						○	●

Appendix D: Planned Schedule (Gantt Chart)

RC Car Subsystem	17-Jan	24-Jan	3-Feb	10-Feb	17-Feb	24-Feb	31-Feb	7-Mar	14-Mar	21-Mar	28-Mar	5-Apr	12-Apr	19-Apr
Tasks	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	**Week 9**	**Week 10**	**Spring Break**	**Week 12**	**Week 13**	**Week 14**	**Week 15**	**Week 16**
Finilize data sets needed to optimize lap times														
Spec sensors to collect desired data (confirm compatibility with DAQ)														
Order sensors and requried hardware														
Design sensor and DAQ mounting														
Manufacture mounting parts														
Install sensors and DAQ mounting hardware														
Wire sensors for interface board compatibility														
Integrate DAQ system onto car														
Validate and Debug system														
Verify/Debug real time dashboard														

Figure D1: Legend for Gantt Charts

Legend	
	Task
	Deliverable
	Subsection Crossover

Figure D2: Gantt Chart for DAQ Subsystem

DAQ Subsystem	20-Jan	27-Jan	3-Feb	10-Feb	17-Feb	24-Feb	31-Feb	7-Mar	14-Mar	21-Mar	28-Mar	5-Apr	12-Apr	19-Apr
Tasks	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	**Week 9**	**Week 10**	**Spring Break**	**Week 12**	**Week 13**	**Week 14**	**Week 15**	**Week 16**
Prepare DAQ Visio Diagram														
Complete Problem Statement & Organization														
Complete User Needs, Specifications, Codes and Standards														
Re-Learn KiCAD formally with Tutorial														
Investigate use of RTOS with our Project														
Create Initial Circuit Diagram														
Audience Analysis Deliverable														
Team Charter Deliverable														
Generate Initial I/O STM Code														
CAD first iteration for STM and PCBs														
Revise Circuit Diagram														
Order Parts														
3D Print First Prototype of Case														
Expo Teaser Deliverable														
Assemble First DAQ Prototype														
Test I/O Pins with STM Debugger and Function Generator														
Prepare Ethernet Visio Diagram														
Code into STM data packaging routine for Ethernet														
Work with Arsh on Ethernet Data Transfer														
Debug STM Code														
Debug Circuit Diagrams														
Reprint PCBs if necessary														
Modify CAD if necessary														
Manufacture and Assemble Second DAQ Prototype														
Verify Functionality														
Implement DAQ into RC Car Test														
Spend Time Debugging and Ensuring Robustness														
Create Testing Report to Prove Functionality of All Components														

Figure D3: Gantt Chart for RC Car Subsystem

Figure D4: Gantt Chart for Cloud Based Software Subsystem

Cloud Based Software Subsystem	17-Jan	24-Jan	3-Feb	10-Feb	17-Feb	24-Feb	31-Feb	7-Mar	14-Mar	21-Mar	28-Mar	5-Apr	12-Apr	19-Apr
Tasks	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	**Week 9**	**Week 10**	**Spring Break**	**Week 12**	**Week 13**	**Week 14**	**Week 15**	**Week 16**
Create System Architecture for data flow from Dashboard to DAQ														
Research Visualization Frameworks														
Finalize Tech Stack														
Design Database Table Structuture														
Implement Database Locally														
Develop & Plan RESTful API Points														
Create Front-end Skeleton														
Verify/Debug Ethernet Data Transfer														
Implement Dynamic Pin Selection based on User Inputs														
Verify/Debug Real-time Data Fetching														
Add Real-time Data Visualizations (Graphs, Tables)														
Integrate Front-end with Back-end APIs														
Verify Front-end/Back-end Integration														
Develop Functionaliity for Pin Selection to Modify Data Visualizations														
Implement Interactive Elements to Data Visualizations														
Improve UI/UX for ease of use and clarity														
Conduct User Testing for UI/UX														
Measure Data Transfer Cycle Length														
Collaborate to Optimize Data Transfers Overall														
Design Docker Container for Consistent Deployment														
Dockerize Entire Software System														
Verify/Debug Software System														

X. Appendix E: Circuit Schematics

Figure 1: PCB Trace Layout for 0-3.3V Protector Board

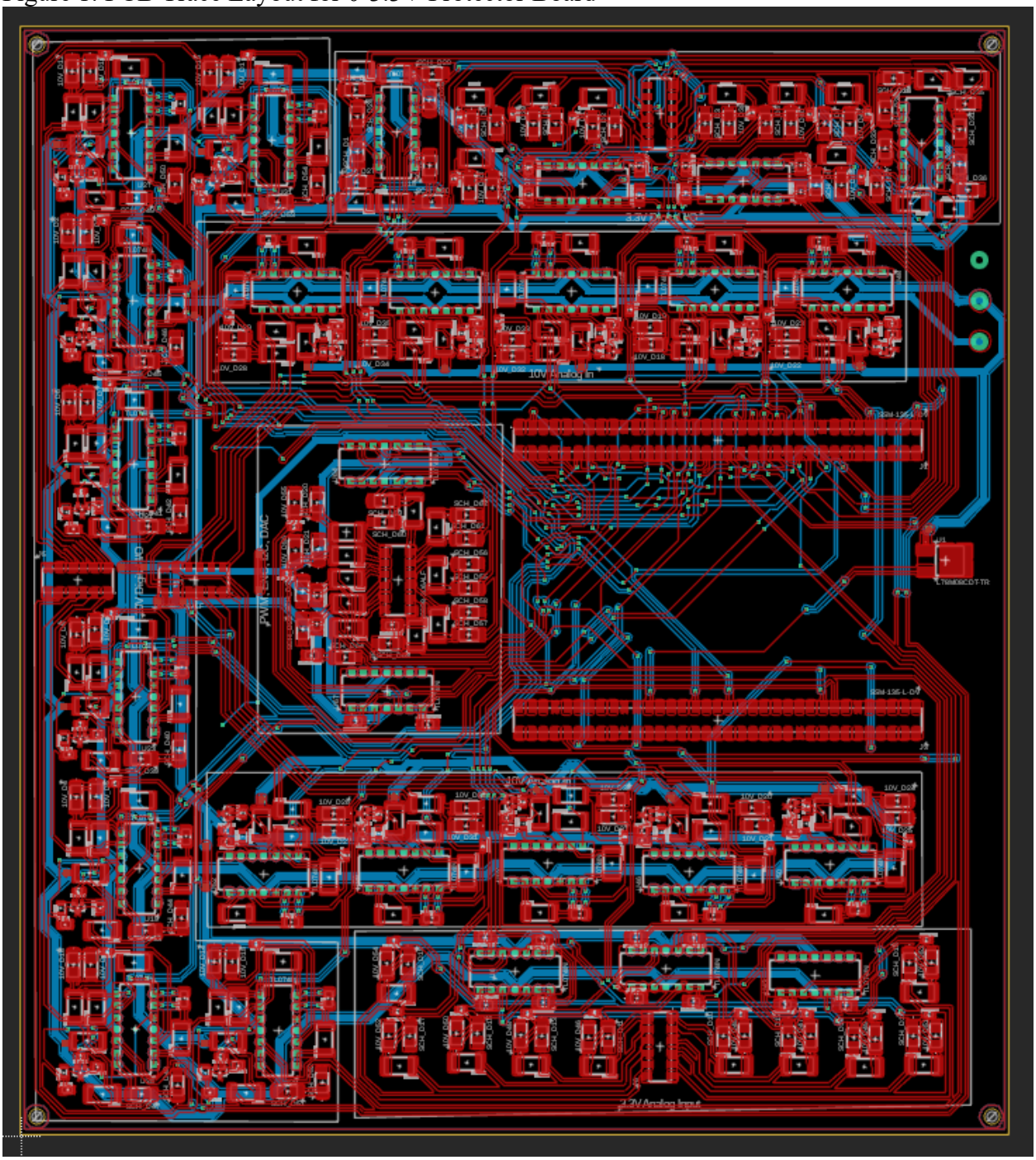
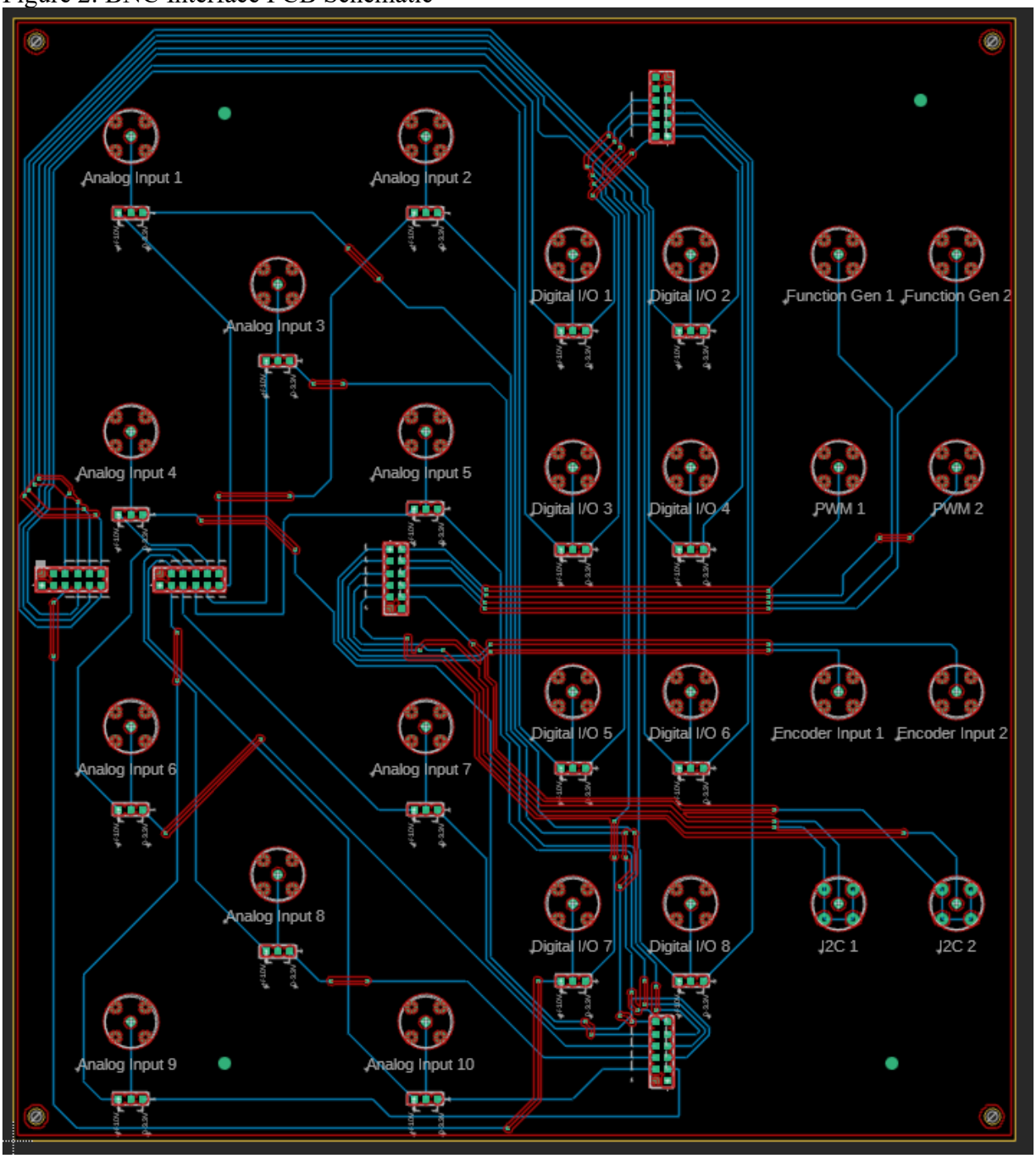
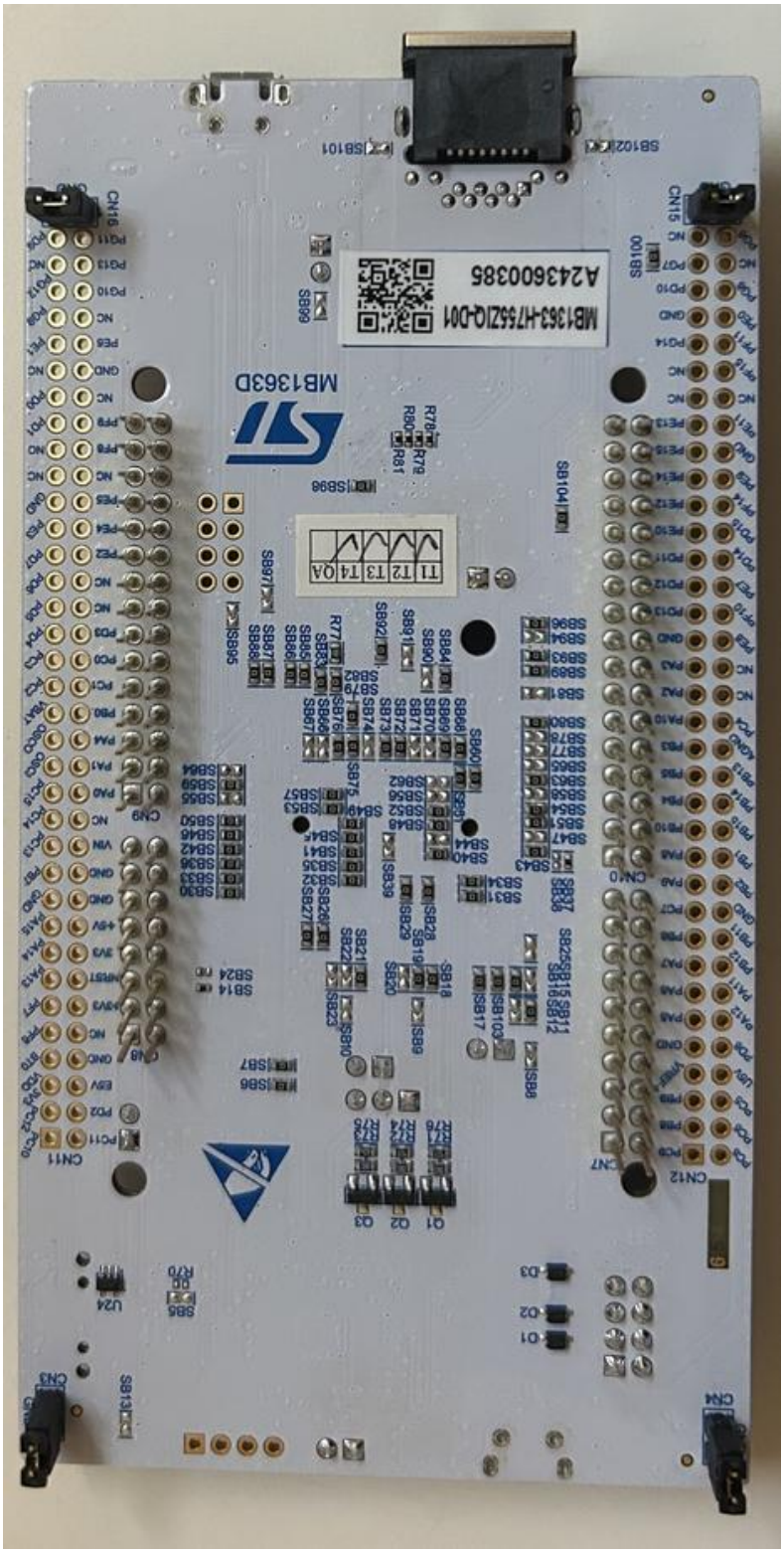


Figure 2: BNC Interface PCB Schematic



XI. Appendix F: STM32 Pinout

Pin Name		Pin Type	Notes
PA6		A	Analog In 1
PF11		A	Analog In 2
PF6		A	Analog In 3
PF7		A	Analog In 4
PF8		A	Analog In 5
PF9		A	Analog In 6
PF10		A	Analog In 7
PC0		A	Analog In 8
PF14		A	Analog In 9
PA3		A	Analog In 10
PD15		D	Digital In 1
PD12		D	Digital In 2
PD13		D	Digital In 3
PF15		D	Digital In 4
PE12		D	Digital In 5
PE13		D	Digitial In 6
PE14		D	Digital In 7
PE15		D	Digital In 8
PE9		D	Digital Out 1
PD14		D	Digital Out 2
PA10		D	Digital Out 3
PB15		D	Digital Out 4
PE7		D	Digital Out 5
PE8		D	Digital Out 6
PG6		D	Digital Out 7
PD11		D	Digital Out 8
PE1		D	Encoder In 1
PB14		D	Encoder In 2
PB6	PB7	I2C1	I2C In 1
PB8	PB9	I2C2	I2C In 2
PC6	(Timer 3)	PWM	PWM Out 1
PB3	(Timer 2)	PWM	PWM Out 2
PA4		DAC	Function Out 1
PA5		DAC	Function Out 2



XII. Appendix G: PCB Bill of Materials

Protector Circuit BOM

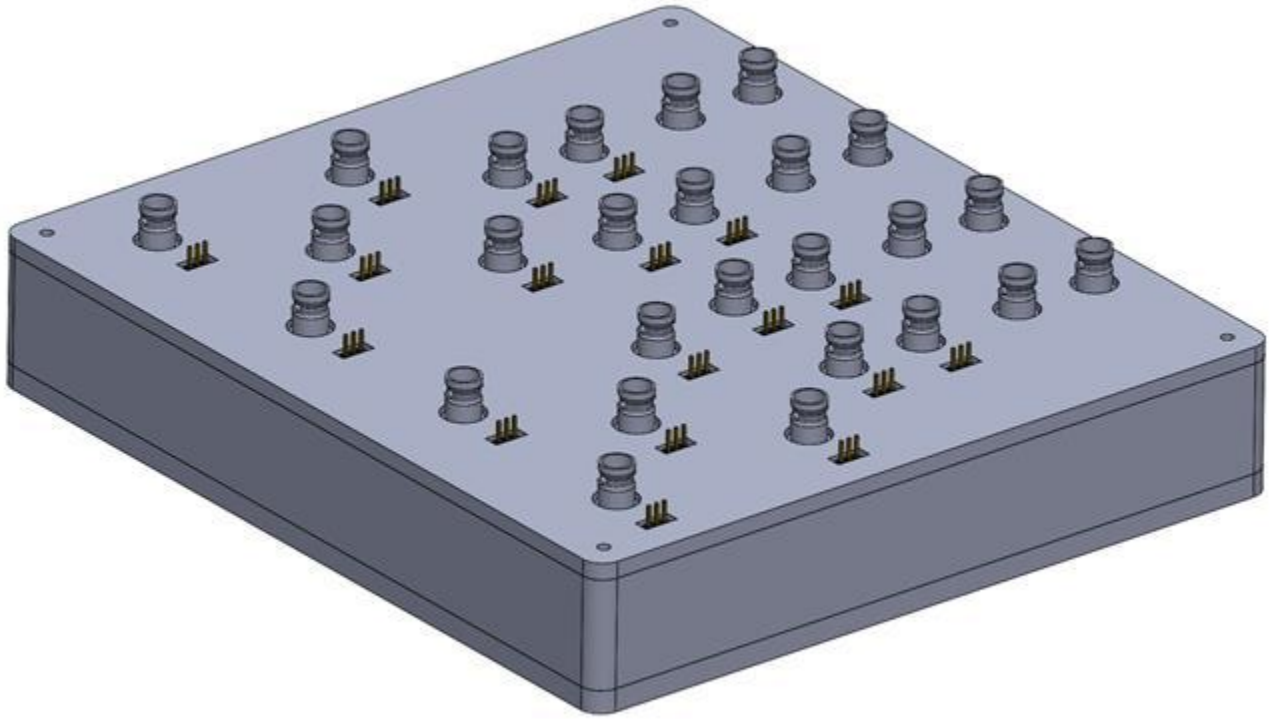
Quantity	Manufacturer Part Number	Manufacturer Name	Description	Digi-Key Part Number 1	Unit Price 1
27	TL074IN	Texas Instruments	IC OPAMP JFET 4 CIRCUIT 14DIP	296-7200-5-ND	0.70480
5	89898-306ALF	Amphenol ICC (FCI)	CONN RCPT 12POS 0.1 GOLD SMD	609-5156-1-ND	1.13500
2	SSM-135-L-DV	Samtec Inc.	CONN RCPT 70POS 0.1 GOLD SMD	SAM1146-35-ND	11.12000
18	TL431AIDBZR	Texas Instruments	IC VREF SHUNT ADJ 1% SOT23-3	296-17329-1-ND	0.12760
56	RC2512FK-072KL	YAGEO	RES 2K OHM 1% 1W 2512	13-RC2512FK-072KLCT-ND	0.08060
18	RC1206FR-07330RL	YAGEO	RES 330 OHM 1% 1/4W 1206	311-330FRCT-ND	0.01120
78	RMCF2512FT510R	Stackpole Electronics Inc	RES 510 OHM 1% 1W 2512	RMCF2512FT510RCT-ND	0.07510
56	SMAZ10-13-F	Diodes Incorporated	DIODE ZENER 10V 1W SMA	SMAZ10-FDICT-ND	0.15500
20	GRM1885C1H432JA01D	Murata Electronics	CAP CER 4300PF 50V C0G/NP0 0603	490-GRM1885C1H432JA01DCT-ND	0.04040
18	RNCF0603BTE32K0	Stackpole Electronics Inc	RES SMD 32K OHM 0.1% 1/6W 0603	738-RNCF0603BTE32K0CT-ND	0.06780
18	RC0805FR-07100KL	YAGEO	RES 100K OHM 1% 1/8W 0805	311-100KCRCT-ND	0.01120
64	B140-13-F	Diodes Incorporated	DIODE SCHOTTKY 40V 1A SMA	B140-FDICT-ND	0.10050
18	GRM1885C1H102JA01D	Murata Electronics	CAP CER 1000PF 50V C0G/NP0 0603	490-1451-1-ND	0.01520
27	SA143000	On Shore Technology Inc.	CONN IC DIP SOCKET 14POS GOLD	ED3014-ND	0.56324
24	BZT52B3V3 RHG	Taiwan Semiconductor Corporation	DIODE ZENER 3.3V 500MW SOD123F	BZT52B3V3RHGCT-ND	0.04200
18	RCS080510K0FKEA	Vishay Dale	RES SMD 10K OHM 1% 1/2W 0805	541-2845-1-ND	0.03960

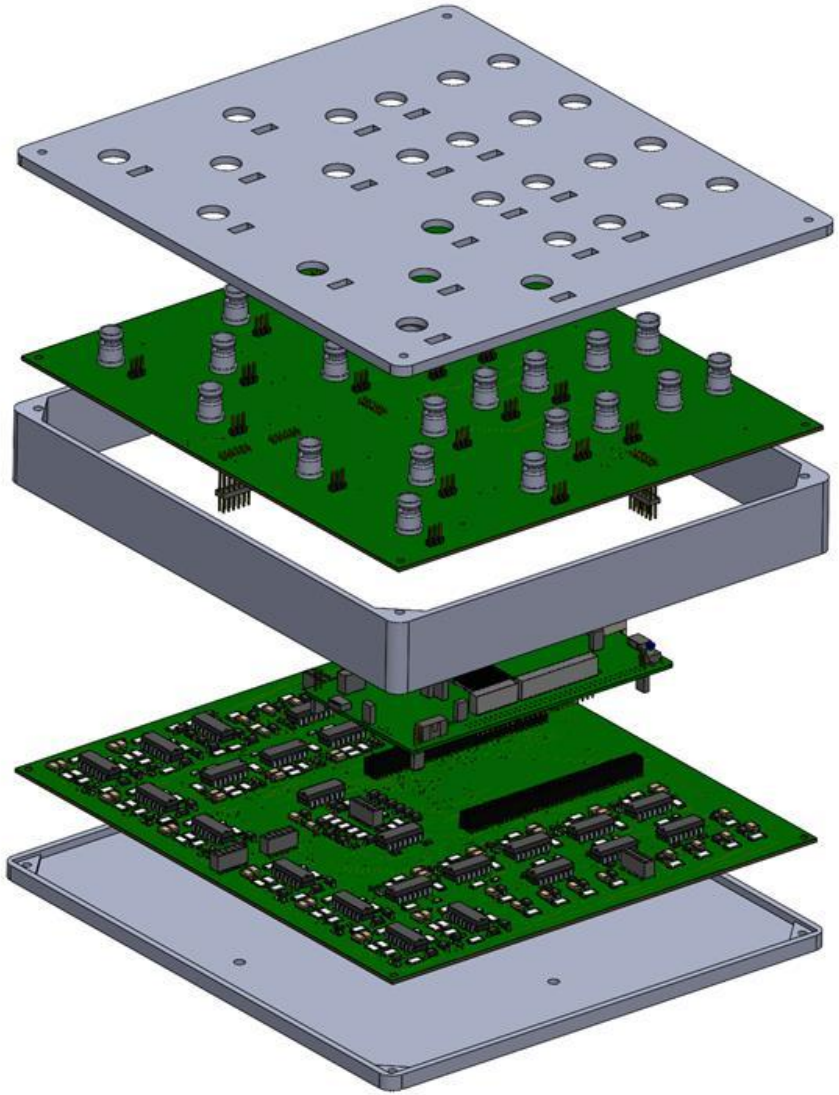
18	RMCF0805FT30K0	Stackpole Electronics Inc	RES 30K OHM 1% 1/8W 0805	RMCF0805FT30K0CT-ND	0.01200
1	L78M08CDT-TR	STMicroelectronics	IC REG LINEAR 8V 500MA DPAK	497-1206-1-ND	0.50000
2	108-0906-001	Cinch Connectivity Solutions Johnson	CONN BANANA JACK SLDR TABS ORAN	J356-ND	1.41000
1	108-0908-001	Cinch Connectivity Solutions Johnson	CONN BANANA JACK SLDR TABS BROWN	J357-ND	1.53000

BOM BNC Interface Board

Quantity	Manufacturer Part Number	Manufacturer Name	Description	Digi-Key Part Number 1	Unit Price 1
5	DW-06-12-L-D-650	Samtec Inc.	CONN HDR 12POS 0.1 STACK T/H	612-DW-06-12-L-D-650-ND	1.49900
26	CONBNC001	TE Connectivity Linx	BNC CONNECTOR JACK, FEMALE SOCKE	343-CONBNC001-ND	1.84000
18	PH1-03-UA	Adam Tech	CONN HEADER VERT 3POS 2.54MM	2057-PH1-03-UA-ND	0.03700
18	60900213421	Würth Elektronik	JUMPER W/TEST PNT 1X2PINS 2.54MM	732-2678-ND	0.27900

XIII. Appendix H: STM32-Based DAQ CAD Models





XIV. Appendix I: Oscilloscope Traces for 0-3.3V Analog Input Range tested with a 15Vpp Signal

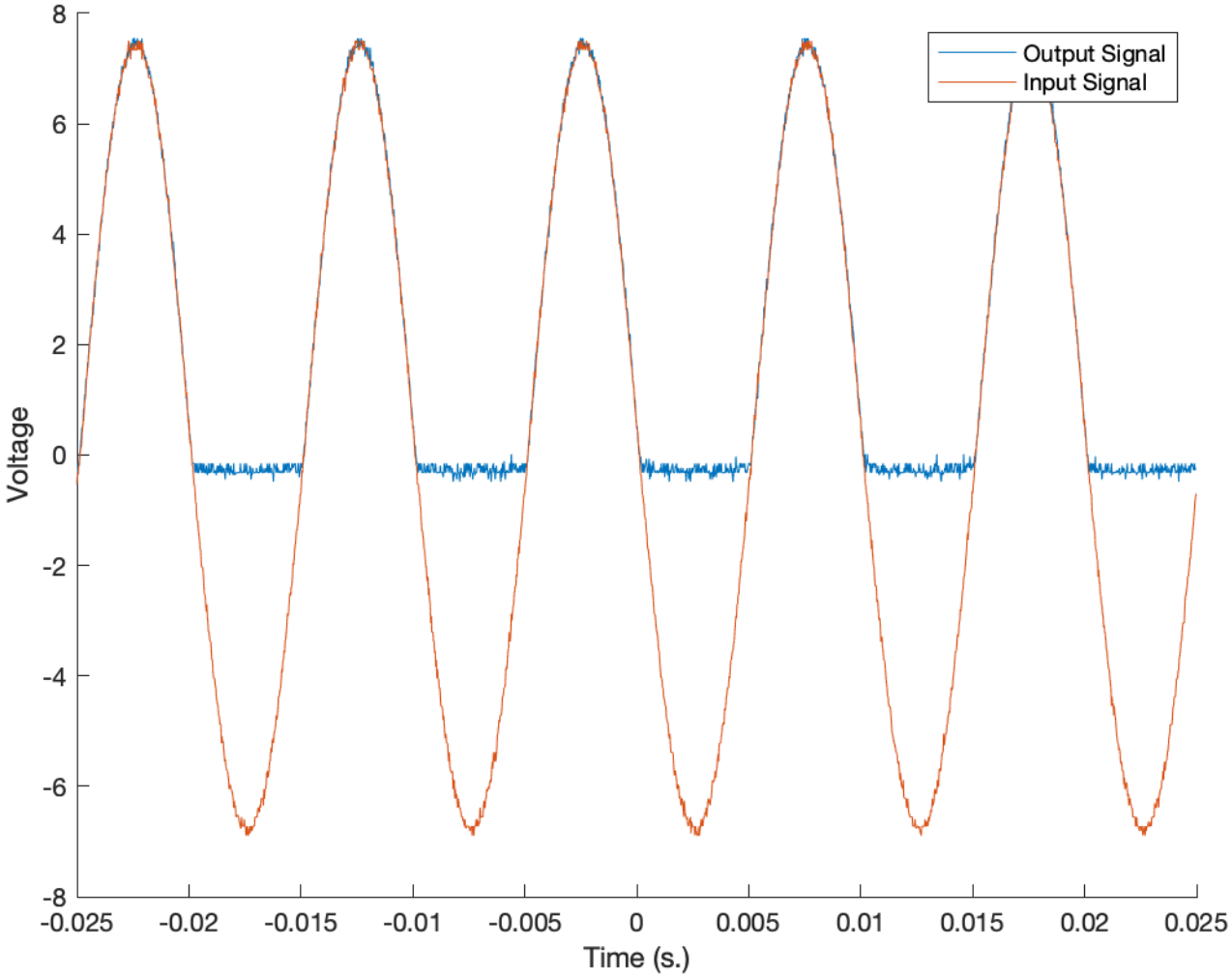


Figure I1: Analog Port 2

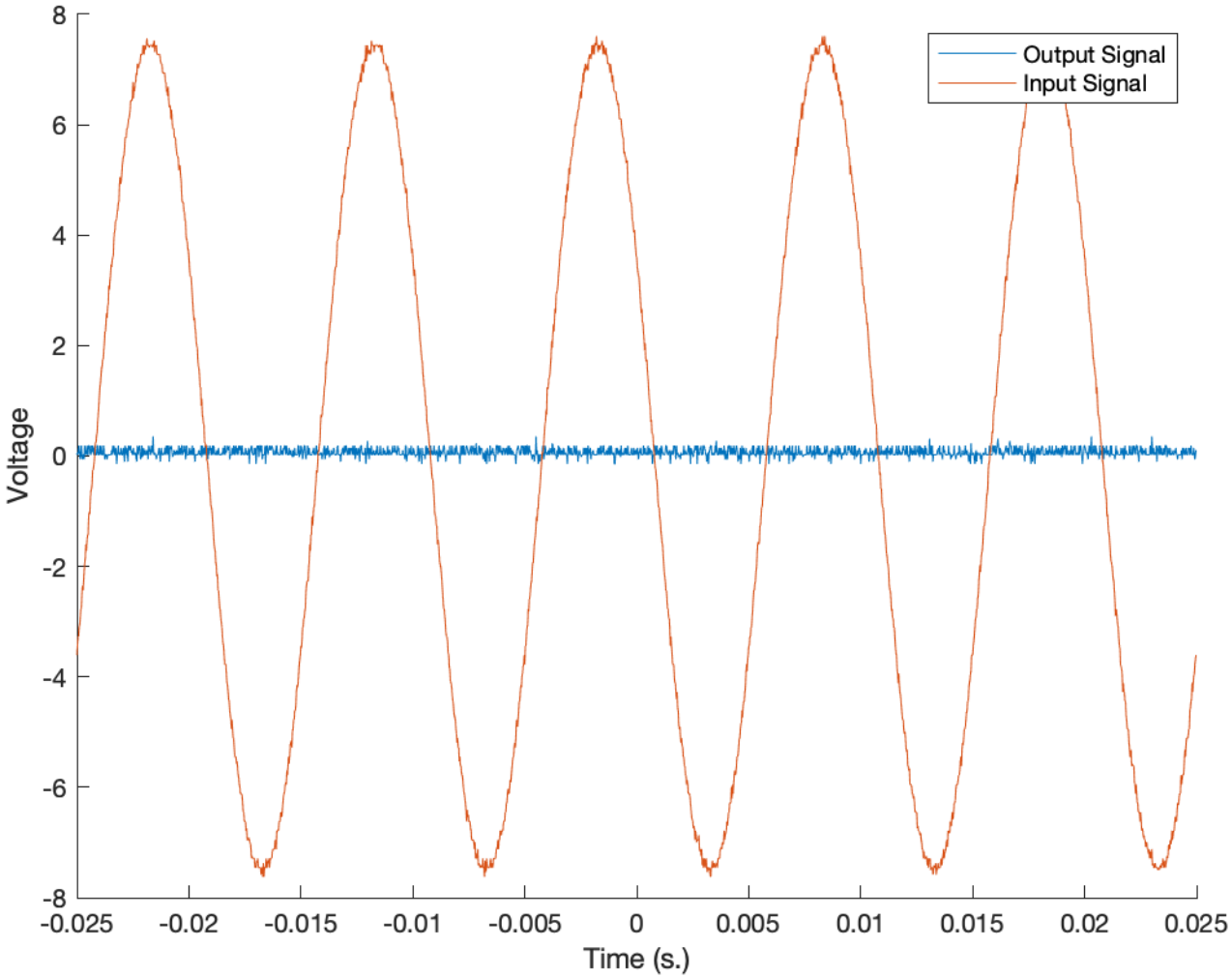


Figure I2: Analog Port 3

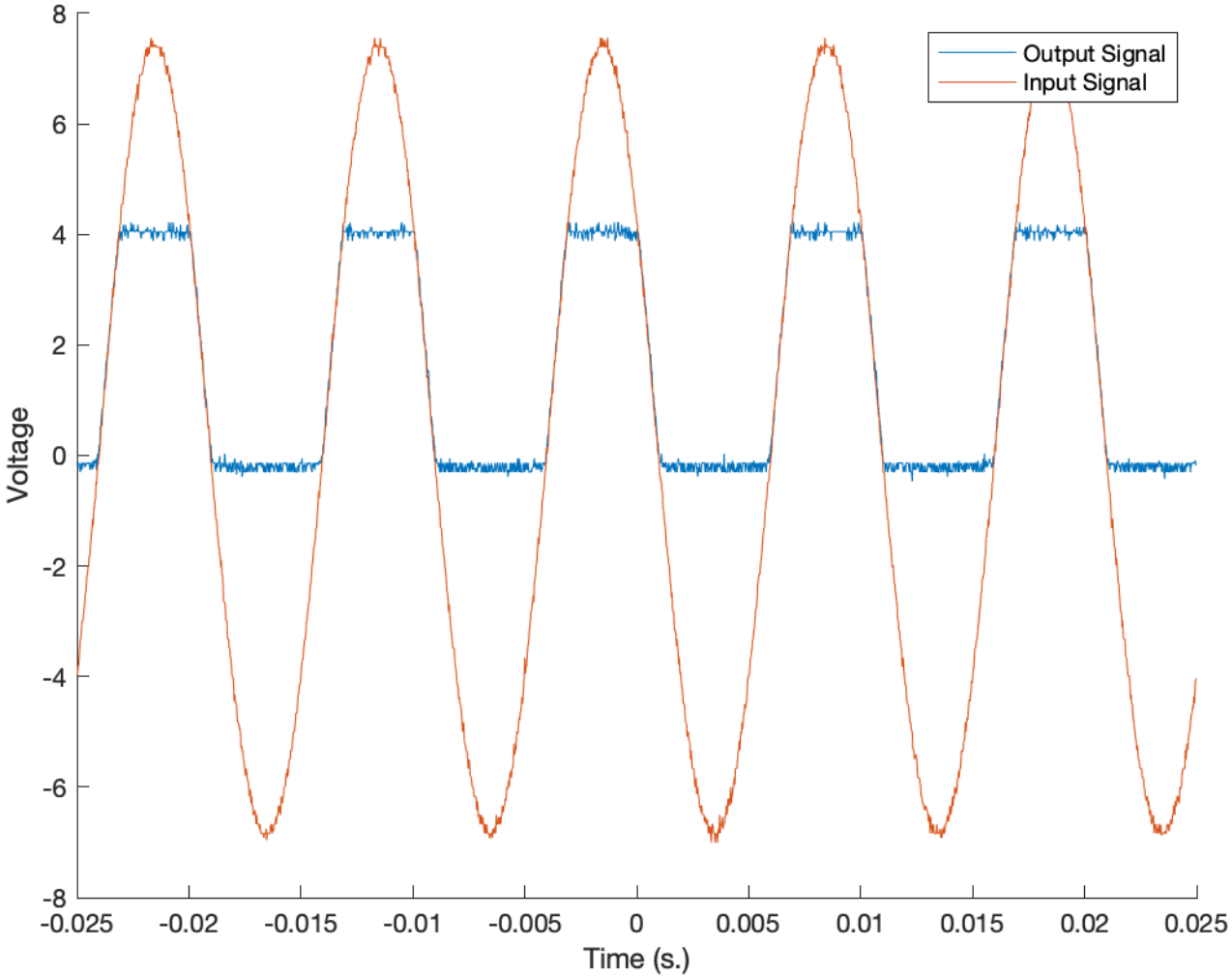


Figure I3: Analog Port 4

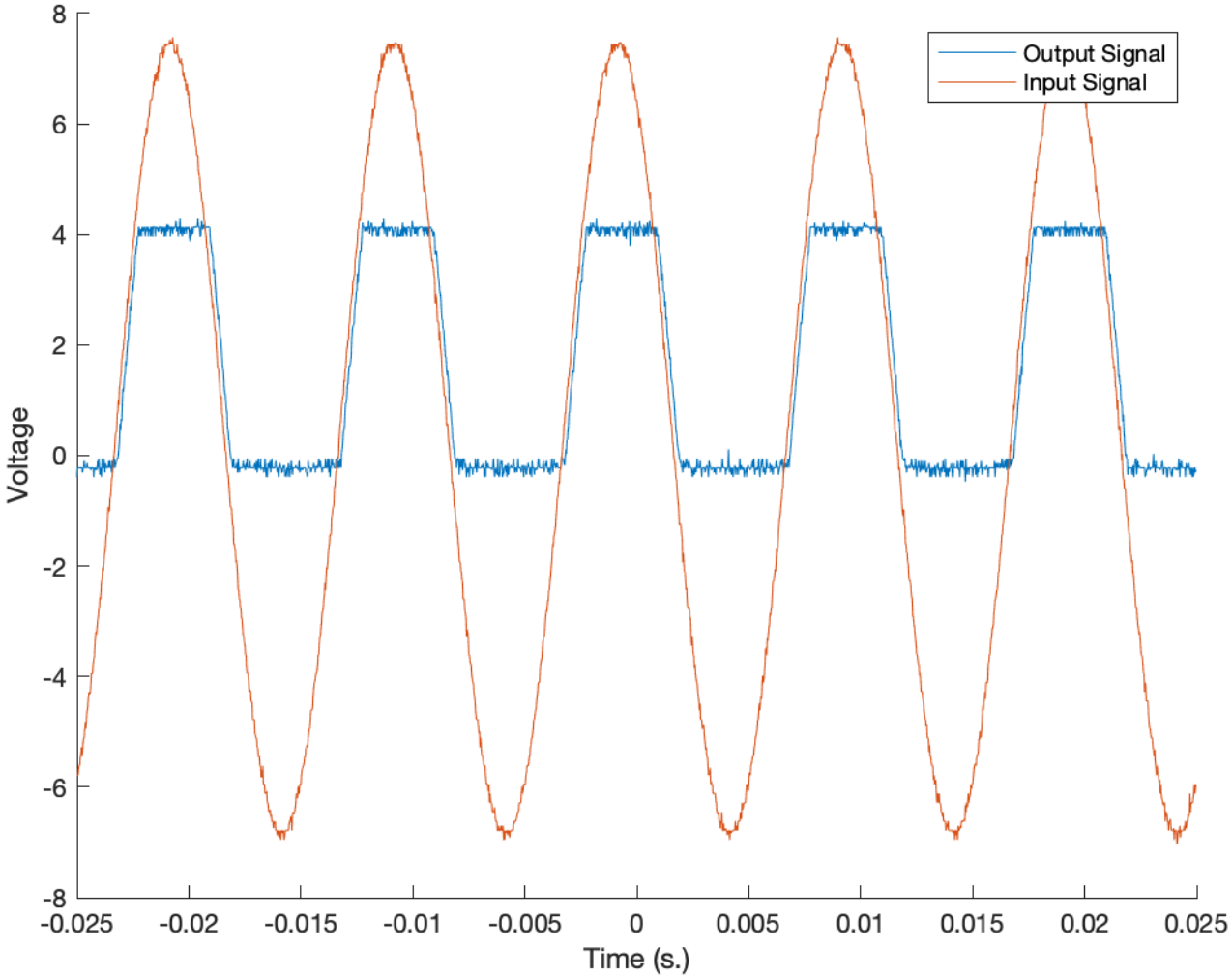


Figure I4: Analog Port 5

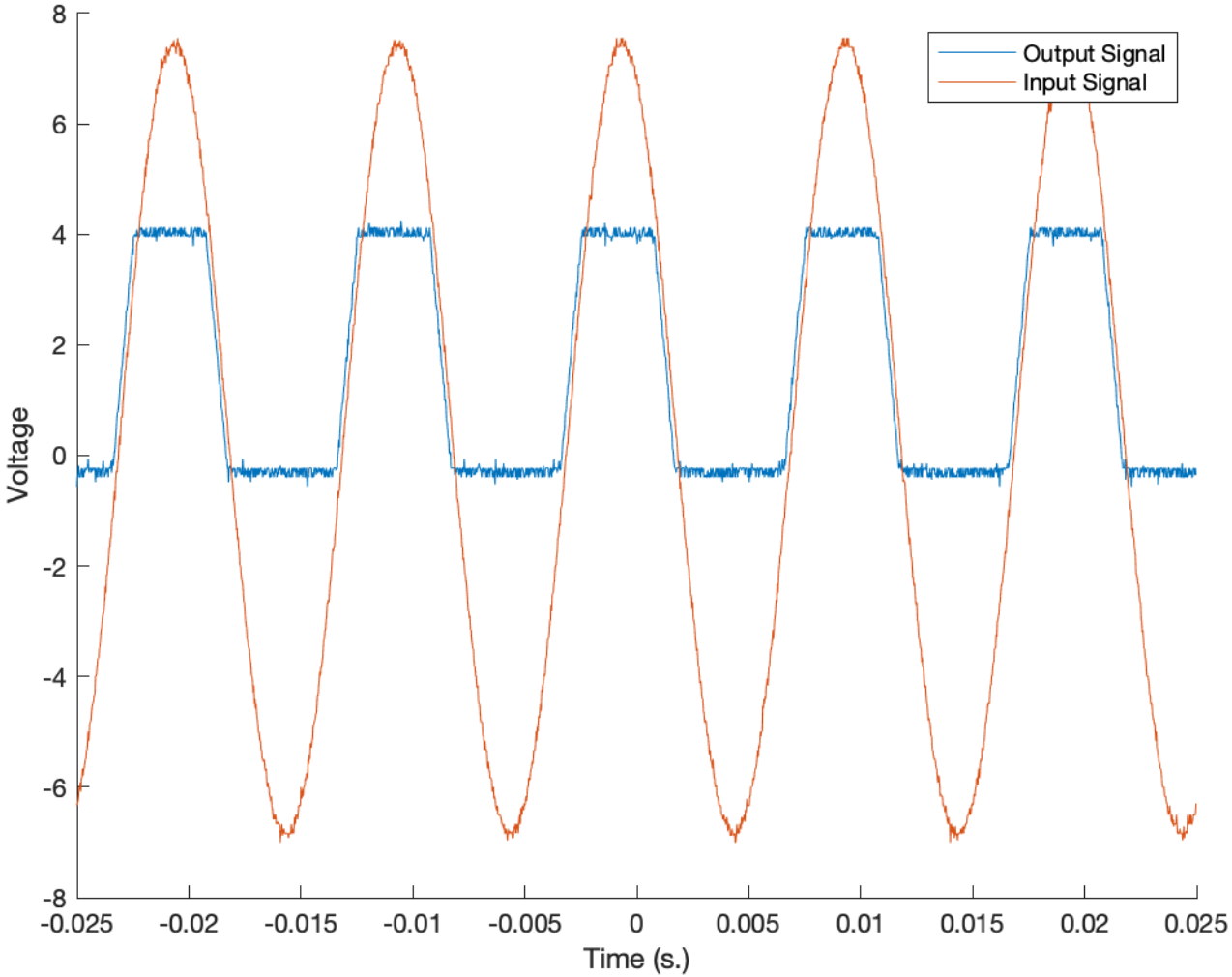


Figure I5: Analog Port 6

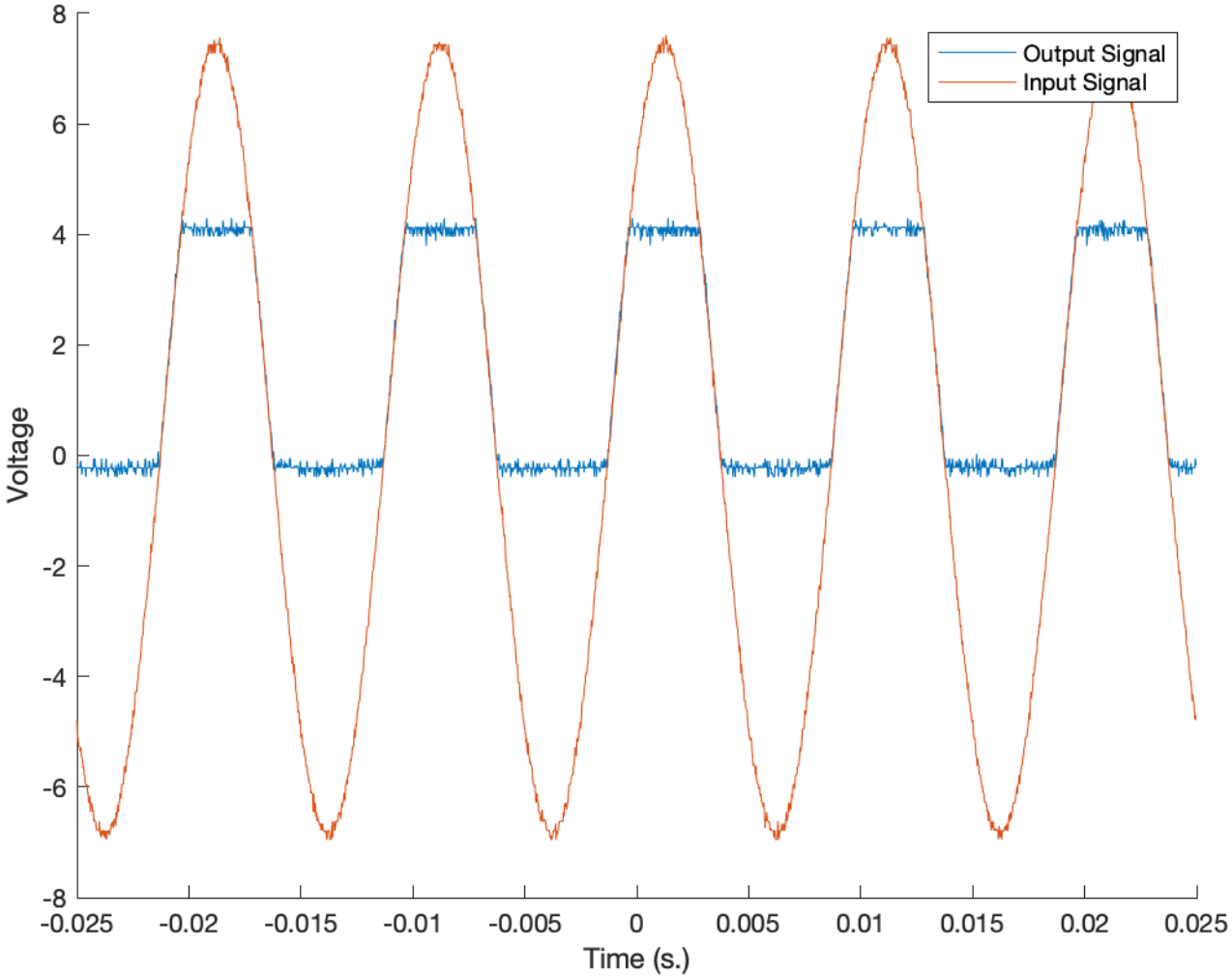


Figure I6: Analog Port 7

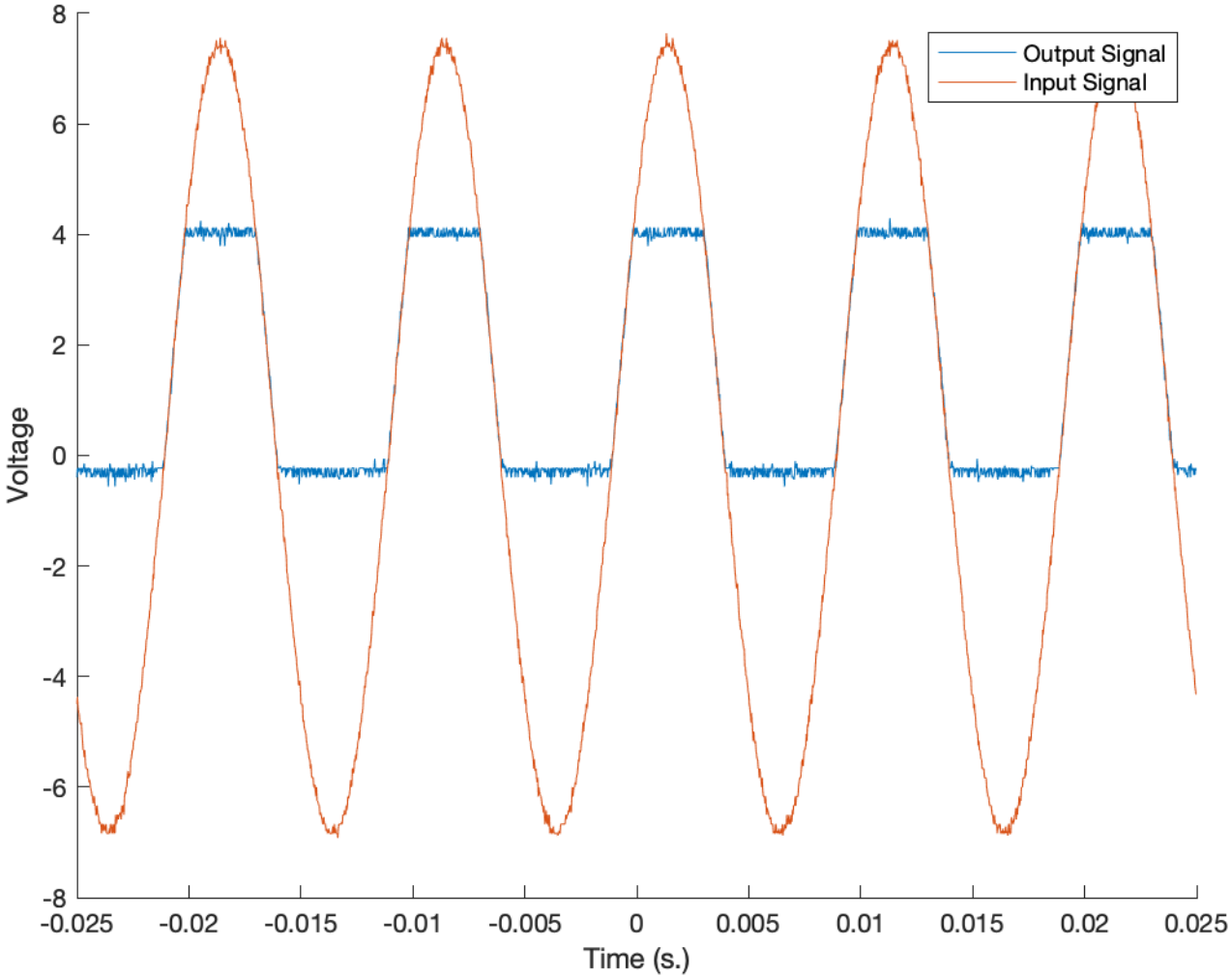


Figure I7: Analog Port 8

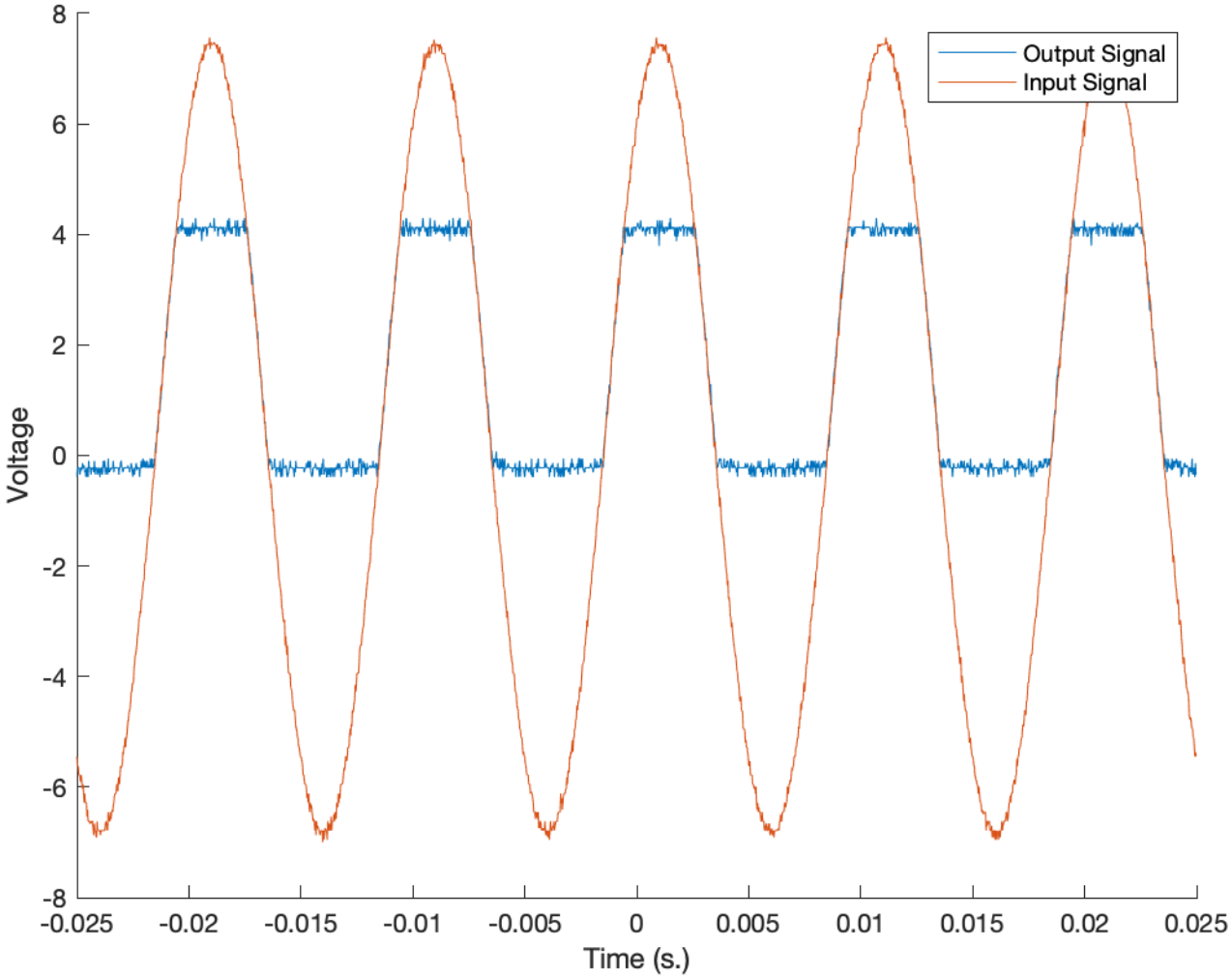


Figure I8: Analog Port 9

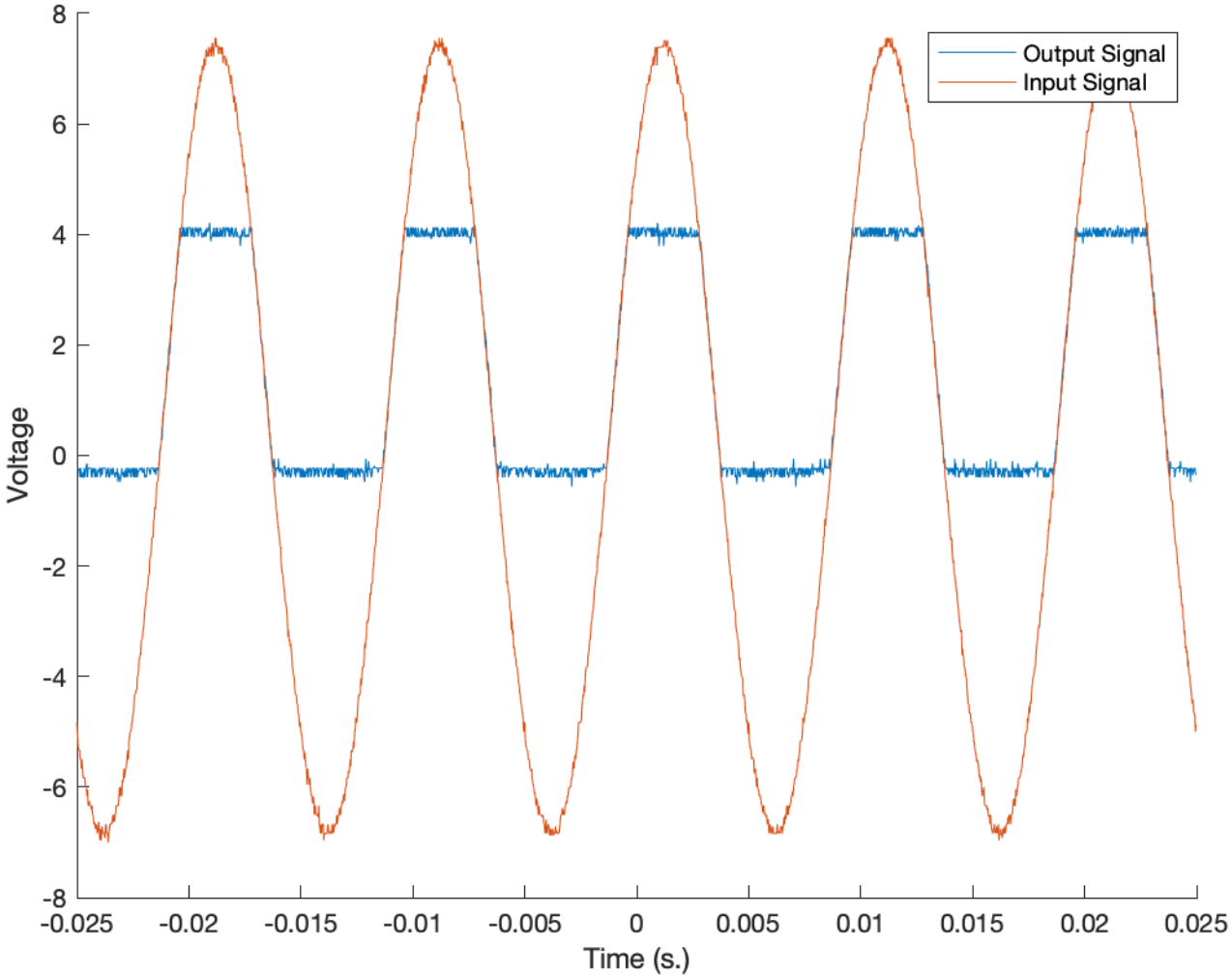


Figure I9: Analog Port 10

XV. Appendix J: Oscilloscope Traces for 0-3.3V Digital Input Range tested with a 0-3.3Vpp Signal

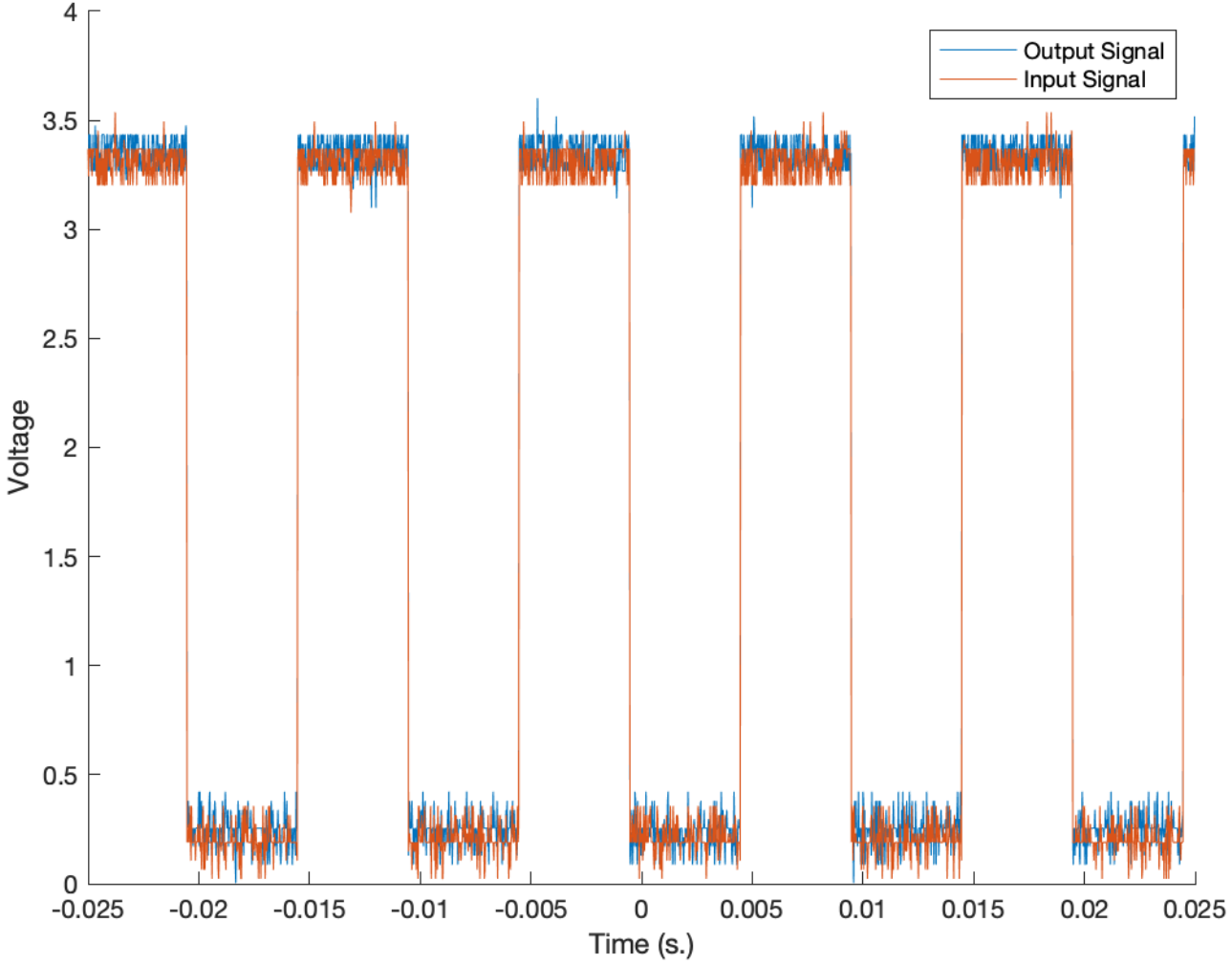


Figure J1: Digital Port 2

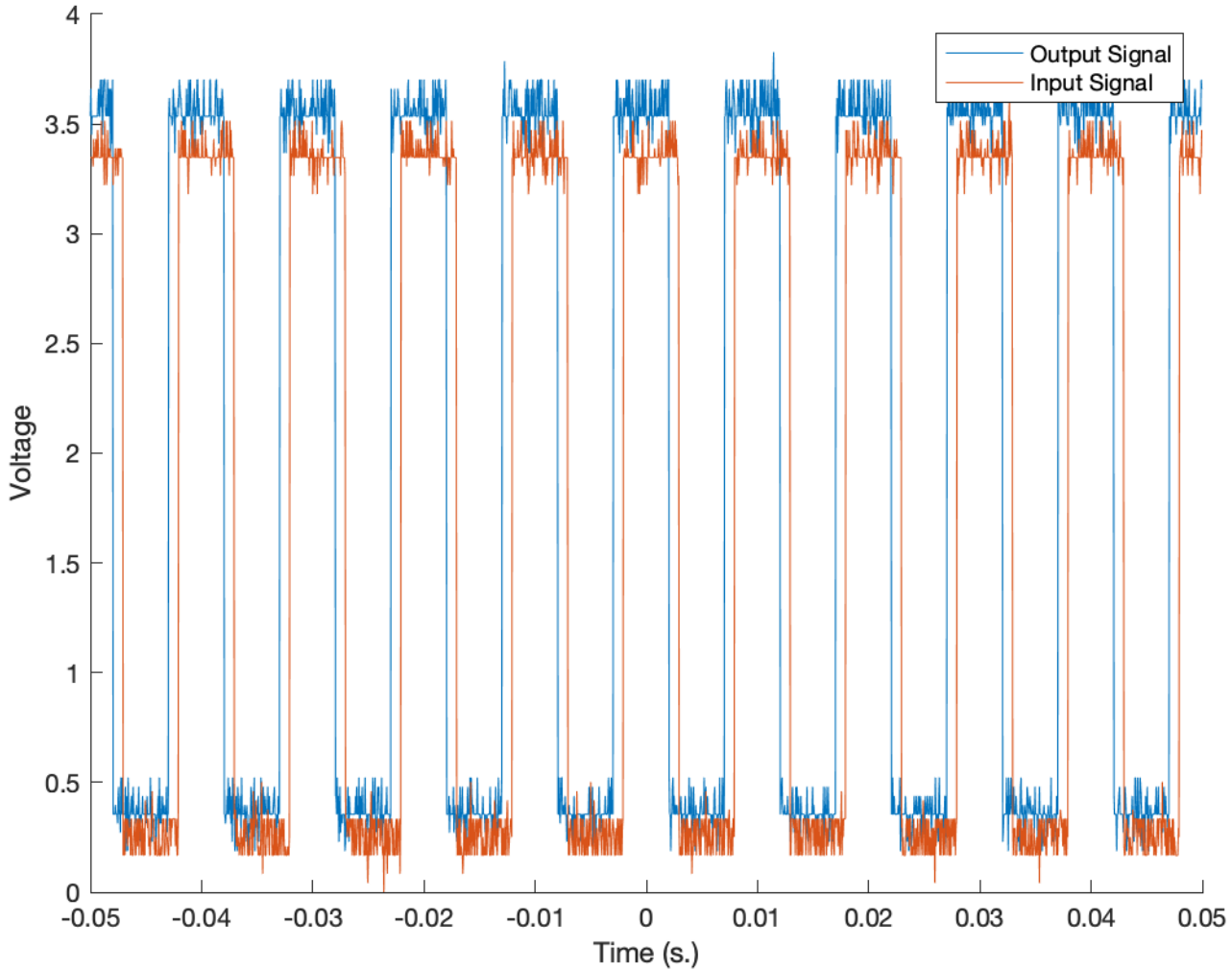


Figure J2: Digital Port 3

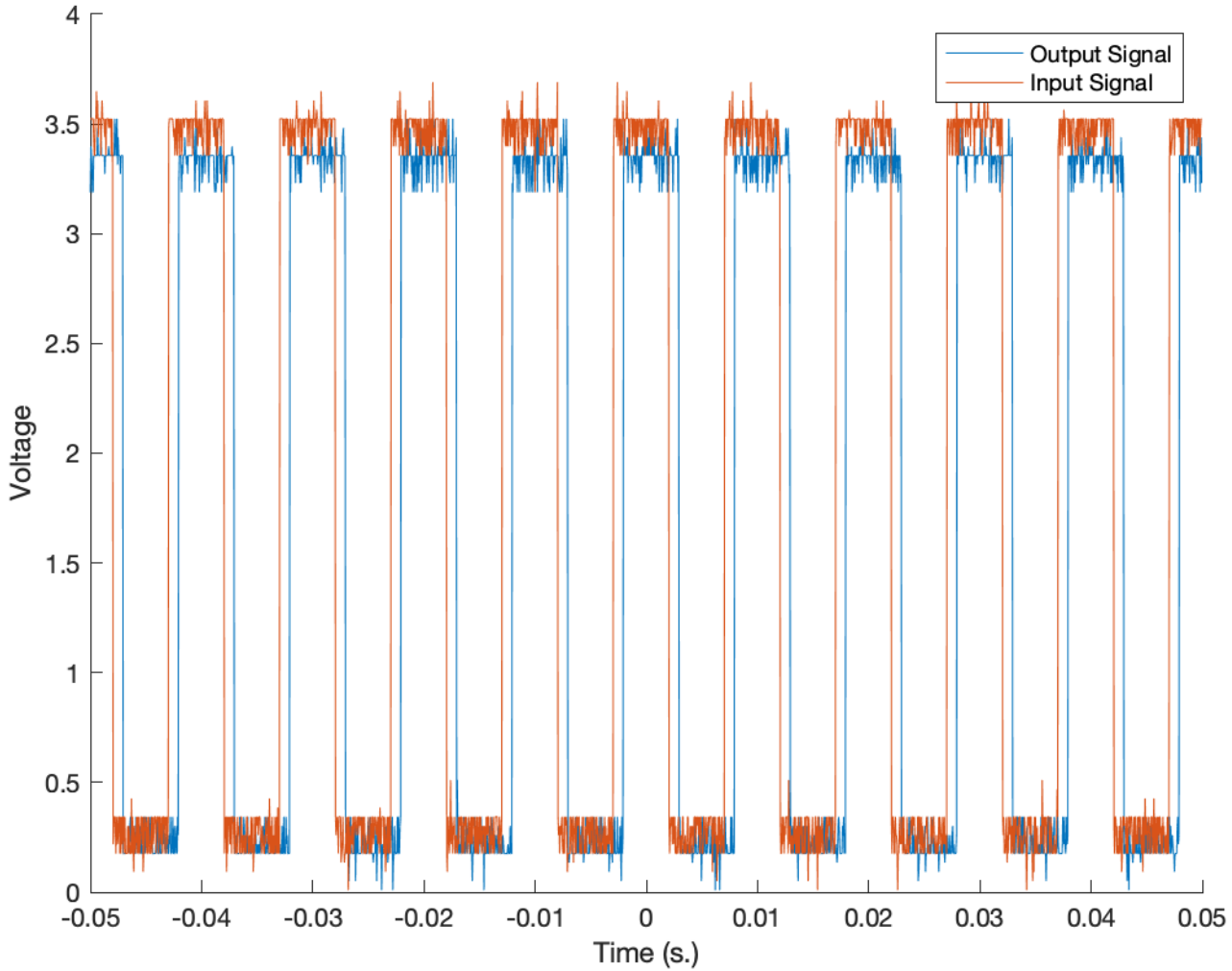


Figure J3: Digital Port 4

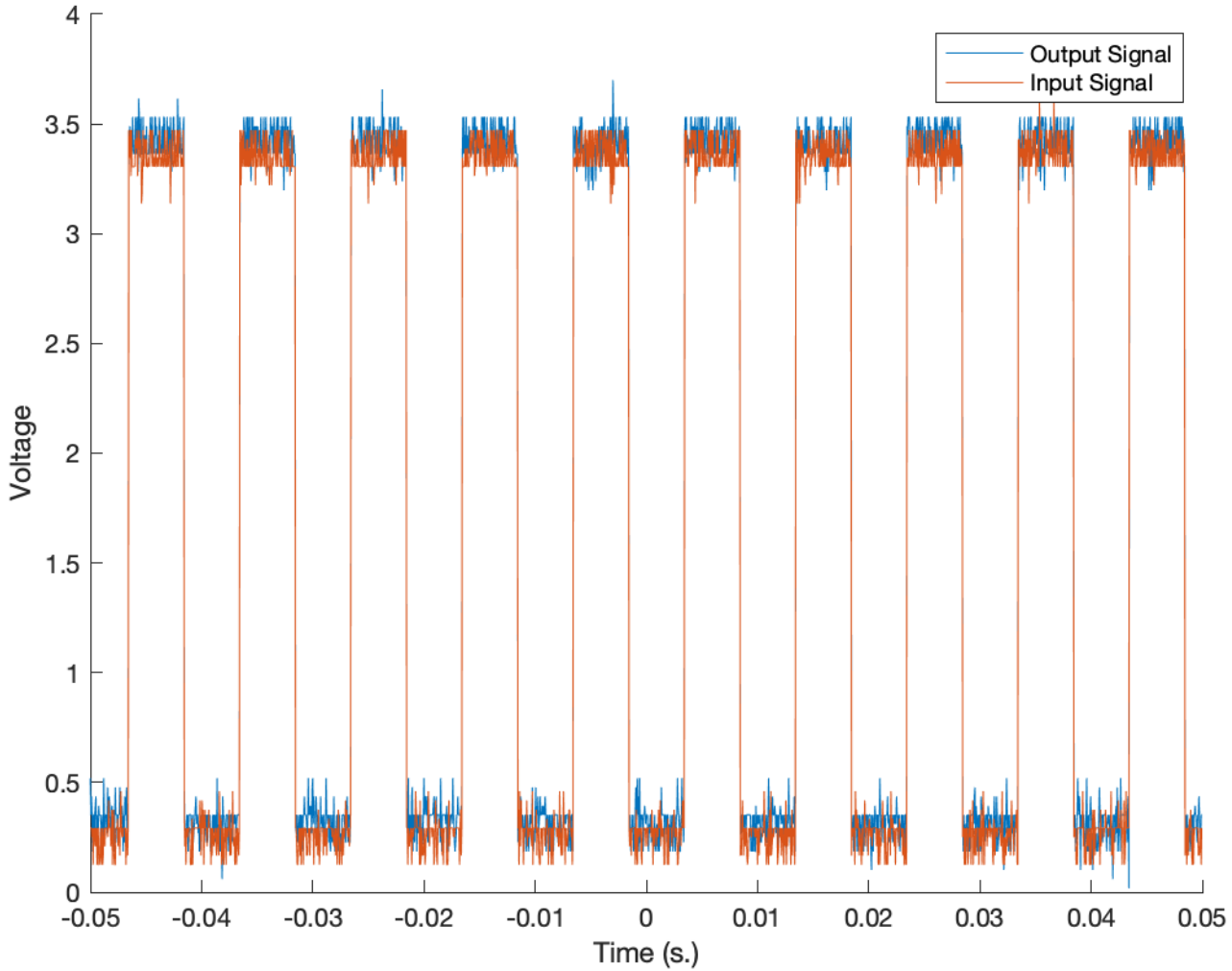


Figure J4: Digital Port 5

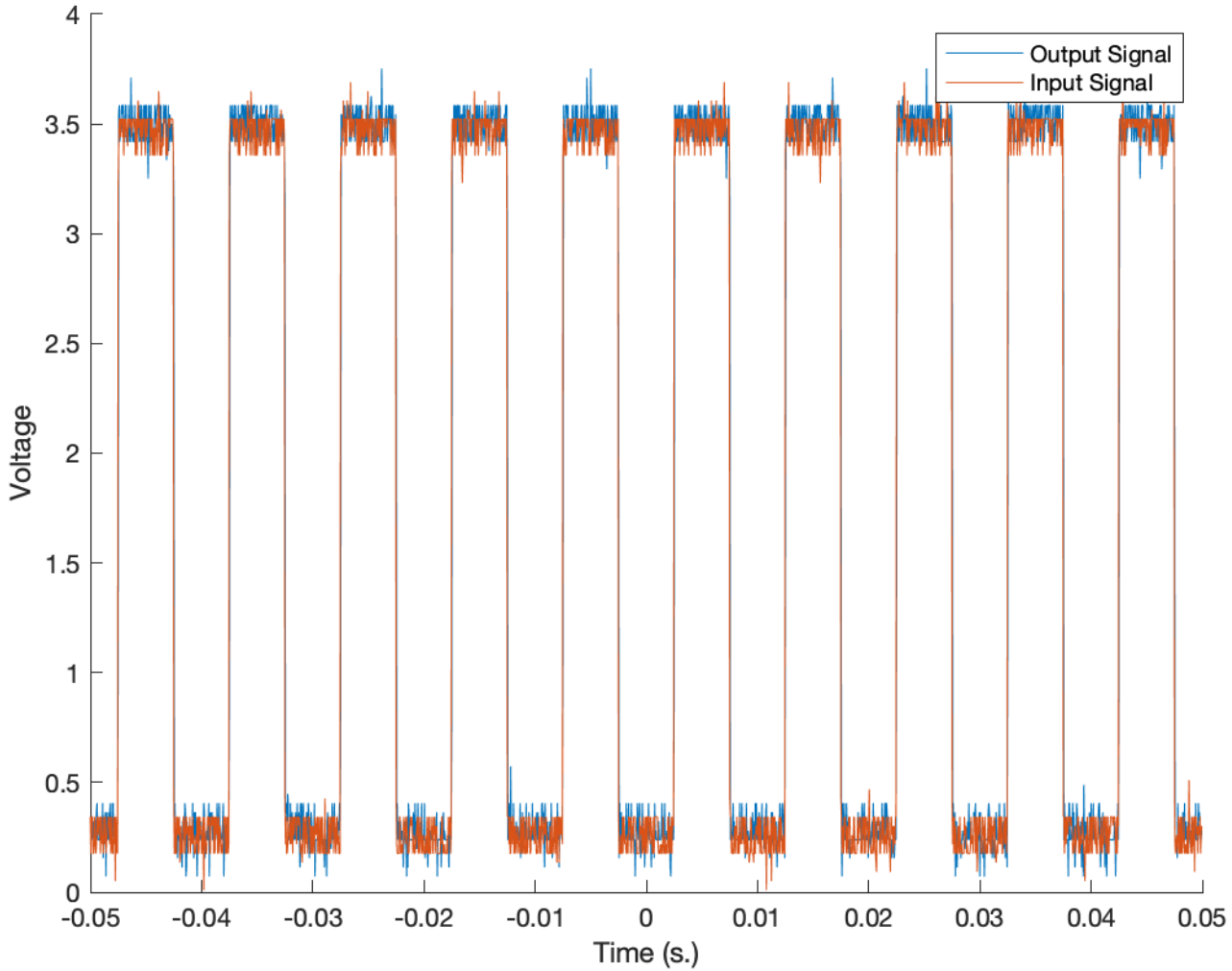


Figure J5: Digital Port 6

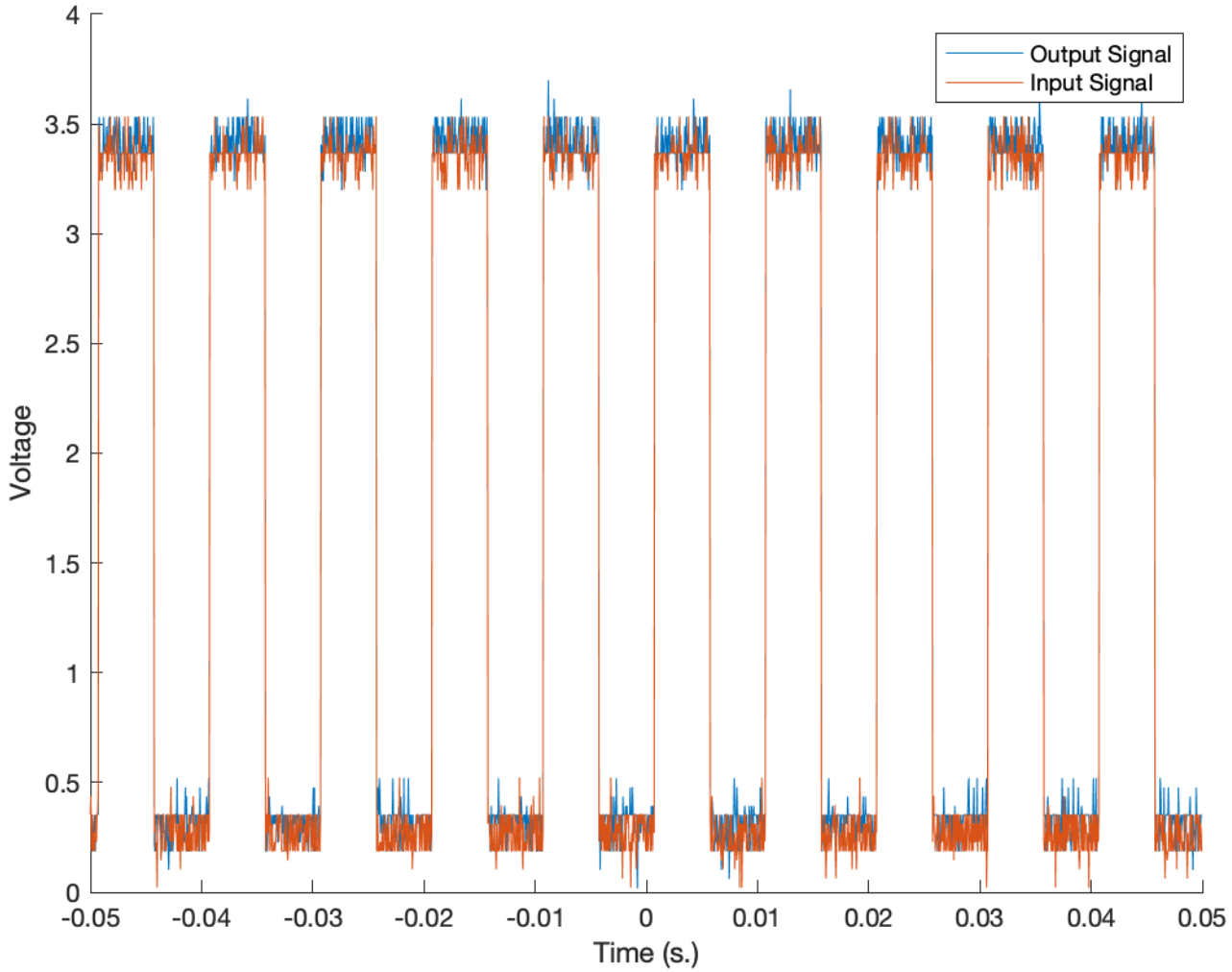


Figure J6: Digital Port 7

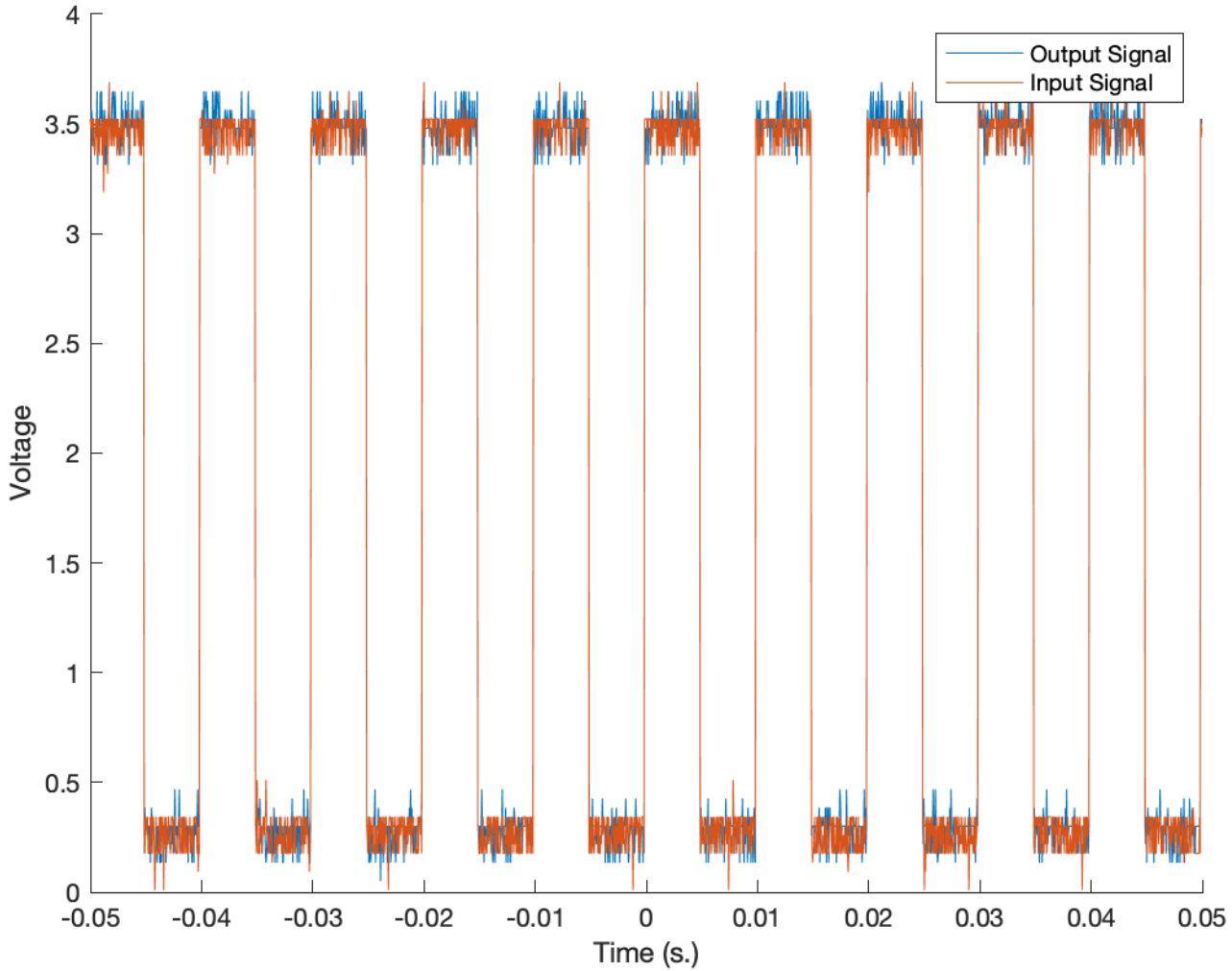


Figure J7: Digital Port 8