SPEED OF LIGHT MEASUREMENT USING TDR ON RP2040

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Spencer Cameron Davis

MEng Field Advisor: Dr. Adams

Meng Outside Advisor: Dr. Land

Degree Date: May 2024

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title:

Speed of Light Measurement Using TDR on RP2040

Author:

Spencer Cameron Davis

Abstract:

This project is an exploratory experiment to determine the capacity of an RP2040 microcontroller to measure the speed of light using the concepts exploited by a time domain reflectometer (TDR). TDRs leverage mismatches in a cable to measure the speed of light. This can be used to determine qualities like the length and quality of the cable and termination impedance. A handheld TDR can typically cost a user anywhere from $200 – $2,000, whereas a Raspberry Pi Pico costs approximately $4. This project aims to explore ways to implement a TDR using the RP2040 and to explore the boundaries in accuracy and precision in measuring the speed of light. The project involves designing and implementing a pulse generator and wave measurement scheme and thoroughly testing the system's capabilities to determine its usability in various experimental setups. Listing out the system's capabilities gives insight into its use cases classified by price vs. accuracy tradeoff. The project will continue to include the development of a straightforward user interface that allows users to quickly and easily pick up one of these and connect it to any BNC-terminated cable, allowing for a simplified experience.

Executive Summary

The purpose of this project was to measure the speed of light on the Raspberry Pi Pico by means of time-domain reflectometry. Using the PIO state machines that guarantee single-cycle instruction execution, this was possible with an accuracy of less than 1% on a 1000-foot cable. This was done by initializing two state machines to check the status of the shorted cable once every other cycle. The instructions for the PIO state machines only allow the machine to check the state of a pin every other cycle, requiring that two state machines be initiated. This would guarantee that the state of the wire was verified on each cycle of the processor. This allows us to have precision equal to the processor speed.

After developing a measurement scheme, productizing this project was the next step. Designing a user experience that allows for an entirely contained device is optimal, allowing a similar experience to a handheld TDR. A standardized connector like a BNC jack allows for easy connectivity to any cable with the same connector. A keypad for user entry allows for easy device configuration in a test environment. Looking at a cost vs. accuracy analysis of the TDR on RP2040, it is clear that this device has a niche in low-risk environments requiring confirmation of a cable connection or a generalized failure location. The project does have limitations as it is accurate to 0.5% on a 1000-foot cable and under 3% when measuring a 500 and 1500-foot cable. The cable being measured must be shorted at the far end and the TDR is only capable of measuring the speed of light over a cable and the length of the cable.

Introduction

A time domain reflectometer (TDR) is a device that measures reflections on a medium. It works by sending a pulse along a wire. As discontinuities in the medium are realized, portions of that incident pulse are reflected to the source. This research project aims to implement this technology on a Raspberry Pi Pico. Time domain reflectometers typically cost upwards of $300 for a handheld device. The Raspberry Pi Pico costs approximately $4, making it considerably cost-effective.

A TDR generates a measurement by sending a pulse over the wire and measuring the time for a reflection to occur. The reflection timing and size depend on the cable's characteristics and the termination method. The TDR on the Raspberry Pi will need to generate a pulse on a pin, measure the time between the incident pulse and reflected pulse, and display the measured speed of light constant calculated and the accuracy of that measurement. This information will give insight into the range of use of this TDR implementation and show the hardware cost vs. measurement accuracy. This paper will include the necessary background in understanding the physics concepts observed by a TDR, issues to be addressed, solutions to those issues, and the development of a self-contained device for user interaction.

Background

A time domain reflectometer (TDR) is a piece of equipment that measures energy reflections over some medium. A typical setup for a TDR built from laboratory equipment and RG6 coaxial cable is shown in Figure 1. The signal generator applies a step to a medium, passing through the oscilloscope to the end of the cable. When the pulse encounters a discontinuity in the medium's impedance or the cable termination, a portion of the wave is reflected by a quantity described as the reflection coefficient. The reflection coefficient (ρ) is a defined as a complex ratio of the voltage of a reflected wave (v^-) to the incident wave (v^+). The reflection coefficient is described as follows:

$$\rho = \frac{v^-}{v^+} = \frac{Z_t - Z_0}{Z_t + Z_0}$$

where Z_0 is the characteristic impedance of the transmission line and Z_t is the impedance of the termination on the far end of the transmission line. If the cable is terminated by means of an open circuit (i.e., $Z_t = \infty$), ρ is 1. If, instead, the cable is shorted to itself, ρ is -1.

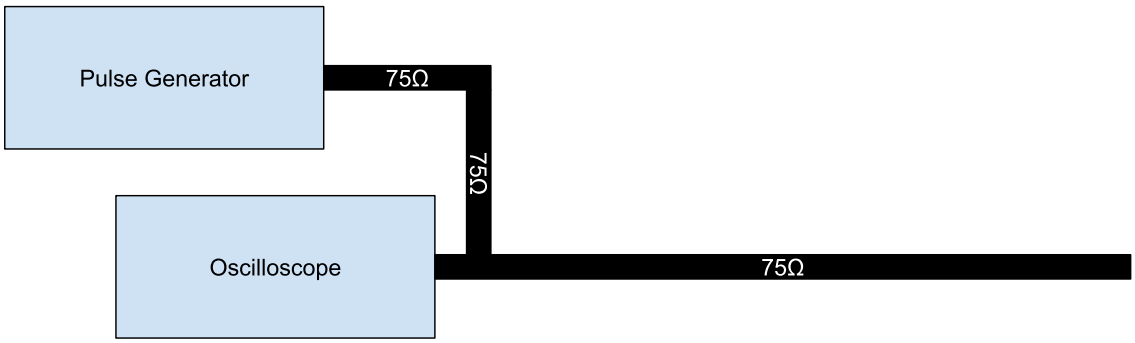


Figure 1: TDR Constructed from Typical Lab Equipment

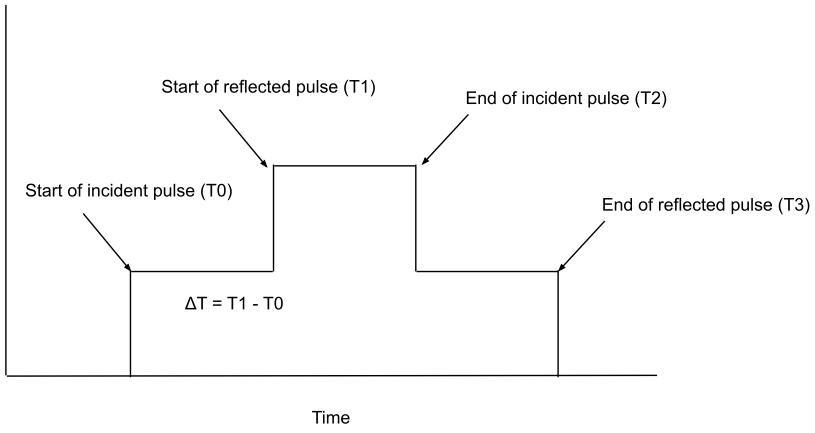


Figure 2: Timing Diagram of a Cable with an Open Circuit Termination

Above is a diagram showing a typical waveform of a cable with an open circuit termination. At T_0, the pulse generator sends the incident pulse along the cable. Before the pulse generator toggles the output off, the signal has propagated to the end of the cable and has reflected to the input, shown at T_1. The magnitude at this point has doubled assuming the cable termination has infinite resistance (i.e., an open circuit). At T_2, the pulse generator has turned off, and the signal read by the oscilloscope at that point is strictly from the reflected pulse. At T_3 the reflected pulse ends. ΔT is the time for the signal to make a round trip from one end of the cable to the other and back. The signal's round trip is shown below on a setup similar to Figure 1 using an RG6 cable.

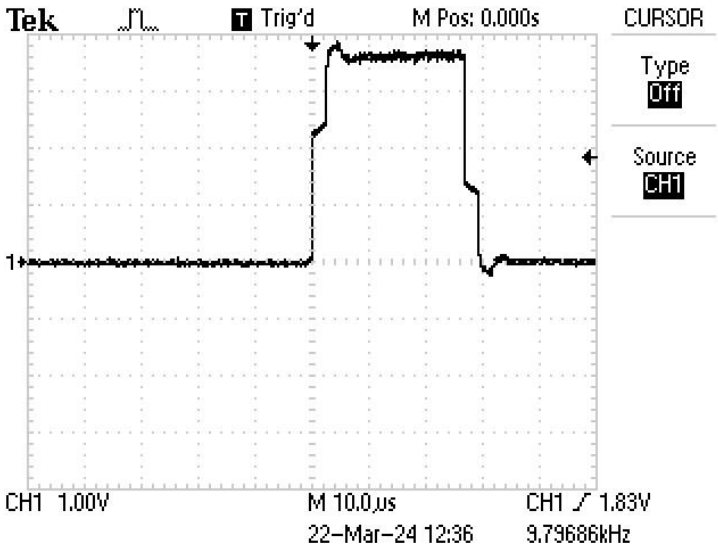


Figure 3: Signal Reflection Shown on Lab Setup Open Circuit Termination

To calculate the speed of light in a vacuum c, we can use the velocity factor (V_f) of a specific cable and the length of the cable (l). This is shown in the following equation:

$$c = \frac{2l}{V_f \Delta T}$$

The formula can be reorganized to calculate various aspects of a setup, like the length of the cable, or to compare the listed velocity factor of a specific type of cable to its actual value. Comparing the theoretical length of a cable to the measured length can aid in determining if a cable has been disconnected or damaged at some intermediate point.

Issues to be Addressed

While computing a value for the speed of light, a major challenge is guaranteeing the execution time of instructions on the Raspberry Pi Pico. The speed of light in a vacuum is 299,792,458 $\frac{m}{s}$ (983.571 $\frac{ft}{\mu s}$). The processor runs at a base frequency of 125 MHz (RP2040 Datasheet, 2023). To obtain the best possible measurement, we would want to check the status of the reflection once per clock tick (125 times a microsecond). Second, we want to synchronize the start of the pulse generation and the start of our measurement schemes to avoid loss in precision due to delay in signaling.

After the base software to implement a TDR has been developed, the system should be scalable to allow an increase in processor speed. Increasing processor speed is fundamental to allow for a more precise measurement. Once the system has reached its maximum sampling frequency, measurements must be taken to evaluate its accuracy across various cable lengths and qualities. Given the processor clock frequency limitations, measuring various cable lengths to determine its range of use would be necessary. As the cable decreases in length, the accuracy of the TDR will also decrease.

After the pulse generation and measurement scheme have been optimized, understanding the extent to which this device can operate is essential. For example, depending on the accuracy of the measurements, the device could be used for various applications that a TDR could be used, like measuring the length of a cable, determining where a cable has been damaged, and determining the termination method of a cable.

An interesting metric also produced by this project includes a breakdown of hardware cost vs. measurement accuracy. The cost of a handheld TDR may not be justified for applications requiring little accuracy in calculating the speed of light or cable termination location. Using a product similar to the TDR on RP2040 may be much better suited to these situations.

Approach and Expected Results

We'll need to generate a count on each processor cycle to measure the time between the incident and the reflected pulse. Fortunately, the RP2040 incorporates a peripheral known as the programable input/output device (PIO) (RP2040 Datasheet, 2023). This co-processor, programmed in PIO assembly (pioasm), can evaluate one instruction per clock tick. Since the processor's clock timing is heavily constrained, the execution time of the PIO instructions can be guaranteed. The PIO state machines execute independently of the central processor, meaning there is zero overhead to run these peripherals. With these peripherals, we can reliably generate a signal and measure the state of the cable once per clock cycle. The Raspberry Pi Pico allows for overclocking up to 250MHz, reliably allowing us to double the precision of our measurements without extra effort.

The Raspberry Pi Pico was chosen for a variety of reasons. An alternative method to measuring the speed of light over a medium could employ an FPGA. The main downside of using an FPGA is the cost. FPGAs are much more expensive than the RP2040, making the Raspberry Pi Pico ideal for this project.

Since each state machine can only execute a single instruction per clock cycle, and pioasm does not support decrementing and measurement in the same instruction, a full state machine measurement program takes two clock cycles. This requires using two state machines, where one measures during the other state machine's decrementing stage to get single-cycle fidelity. This is shown in more detail below in Table 1. An alternative to measuring using the PIO state machines would be to use the ADC. The ADC has a sampling rate of 500kHz, making it much too slow to measure the speed of light effectively (RP2040 Datasheet, 2023).

To take the cycle count calculated by each state machine and convert it to a cycle-high value, we must determine whether it is an even or odd count. If PIO state machine 0 is equal to PIO state machine 1, that means it is an odd count, and the number of cycles high is equal to $(starting\ count - end\ count) * 2 - 1$. In the table below, looking at line 5, the starting count is 100, and the current count at line 5 is 97, so the count would be $(100 - 97) * 2 - 1 = 5$. If it is an even count, that would result in PIO state machine 1 being larger than PIO state machine 0. For example, this can be seen in the table's final row. The conversion from the PIO state machine

count to a cycle-high count would be $(starting\ count - end\ count) * 2$. In the final row, since it is an even count indicated by PIO State machine 1 being greater than PIO state machine 0, that means current count is 97, therefore the calculation would be $(100 - 97) * 2 = 6$.

Number of Cycles Pulse is High	**PIO State Machine 0**		**PIO State Machine 1**	
	Instruction	**Count**	**Instruction**	**Count**
0 (pulse goes high)	wait	99	wait	99
1	wait satisfied	99	wait satisfied	99
2	Count - 1	98	No Operation	99
3	Measure	98	Count - 1	98
4	Count - 1	97	Measure	98
5	Measure	97	Count - 1	97
6	Count - 1	96	Measure	97

Table 1: PIO State Machine Program Flow Example

The PIO state machines are similar other than a no operation instruction in PIO state machine 1. This allows each state machine to be offset by one instruction from each other. The state machines are passed an initial count value, equal to UINT32_MAX (0xFFFFFFFF), which is decremented every other cycle, as shown above in Table 1.

Additionally, since we must distinguish when the reflection has occurred and the utilized PIO implementation does not communicate with ADCs, the reflection must result in a binary on or off state. This can be achieved by shorting the end of the transmission line to result in 0 voltage when the reflected wave returns to the source.

A method to minutely increase the precision of the measurement is to reduce noise caused by a breadboard. Developing a PCB to connect the cable directly to the signal generation and measurement system can help eliminate sources of noise. Implementing a standard cable connection like a BNC directly on the PCB would improve the user's experience when utilizing the device.

In doing preliminary calculations to determine the maximum possible accuracy of this product, a processor running around 250 mHz should produce a measurement of the speed of light with less than 1% error given a cable 1000 feet long. When running at 250MHz, in a vacuum, light will have traveled 3.93 feet per sample. Since the signal is not traveling through a vacuum but instead a cable, the time for the signal to travel along the cable is prolonged. Testing

this product with shorter and shorter cable lengths will test the viability of this project in such situations.

Results

To test the program described in the previous sections, the setup consisted of the Raspberry Pi Pico connected to a 1000-foot RG6 cable shorted to its ground conductor at the far end (shown below in Figure 4). In this diagram, pin 20 acts as the signal generator, and pin 19 monitors the state of the wire. Pin 18 is one of the Raspberry Pi Pico's ground cables.

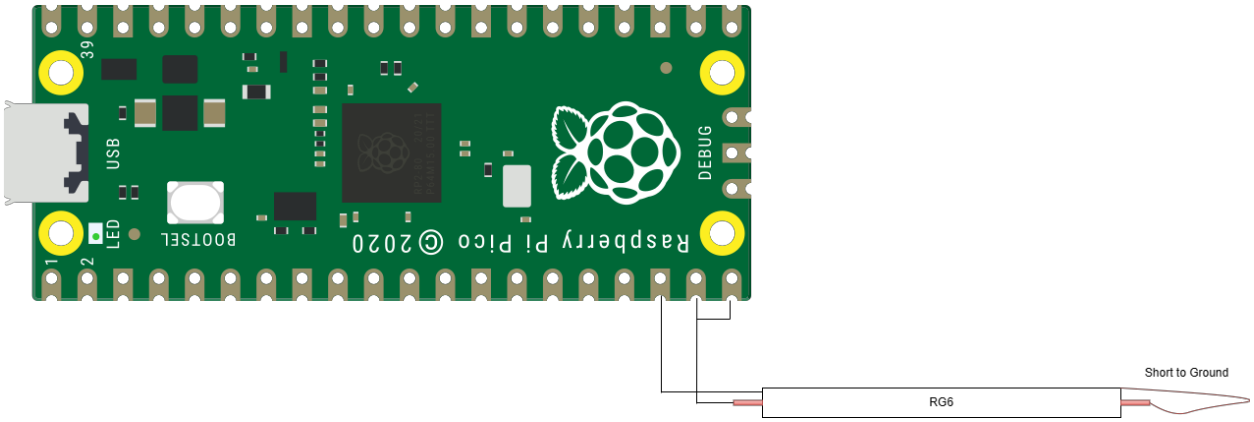


Figure 4: Lab Setup for Initial TDR Readings

Below is a table containing the results of the breadboard implementation of the TDR against various RG6 cable lengths. The source of the error can be theorized to be caused by a few different sources. Verifying the length of the cable would prove quite tricky as the manufacturer does not list the tolerance of the cable length, and unwinding the cable would be tedious. Second, the velocity coefficient varies in different cables up to 5% (Duffy, n.d.).

True Cable Length	Number of Cycles Counted	Number of µs	Measured Speed of Light (feet/µs)	|Error| (%)
500	308	1.232	1002.084	1.88222
1000	626	2.504	986.0766	0.25409
1500	956	3.824	968.5418	1.52803

Table 2: Speed of Light on Breadboard Setup

True Cable Length	Number of Cycles Counted	Number of µs	Measured Cable Length (feet/µs)	|Error| (%)
500	308	1.232	490.7626	3.84738
1000	626	2.504	994.2723	0.5727
1500	956	3.824	1523.276	1.55175

Table 3: Cable Length on Breadboard Setup

Productization

The next step of this project would be to transform this device into something user-friendly instead of a lab setup. A small handheld device that allows users to enter parameters such as the cable length and velocity factor would allow them to tune this device's output to their specific setup. Since the equation we use to calculate the speed of light depends on the cable's length, we can manipulate this equation to calculate the cable's length. The equation:

$$c = \frac{2l}{V_f \Delta T}$$

becomes

$$l = \frac{cV_f \Delta T}{2}$$

The device's tolerance hovers around 4 feet on a 1000-foot RG6 cable. Looking at a Cost vs Resolution plot, the TDR on RP2040 sits low on the cost scale of the graph and is competitive when looking at the resolution. Note that the compared devices' resolutions are relative to the range measured and are compared against their standard resolution on a 750-foot cable.

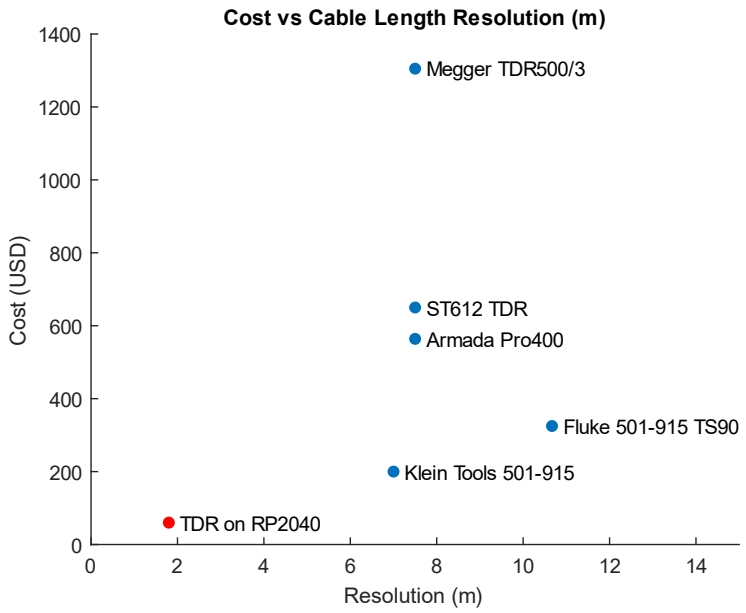


Figure 5: Cost vs Cable Length Resolution Plot

Since this device's measurement scheme depends on the wire's binary high/low state, the precise impedance along the wire is impossible to measure. This device would prove helpful when the user needs to verify the connectivity of a wire between two points. Detecting a break in the wire would be accurate up to 5 feet on a 1000-foot cable. Another downside to this product is the limited range in output impedance. Since this device features a 75 Ω BNC jack, setups read by this device are limited to this characteristic impedance. The Raspberry Pi Pico also has a maximum signal amplitude of 3.3v at 12mA on a single pin. This puts an upper limit on the overall length of cable that the TDR can measure on RP2040.

Creating a user experience allowing for easy use of this device is next. The goal would be to create an entirely contained device, allowing the user to plug this device into any BNC cable and test a setup. As mentioned above, developing a PCB to reduce possible error sources is also necessary. To design a product for the customer, we first design a much simpler version of the device housing a BNC jack, Raspberry Pi Pico, and a USB to UART converter. This rendition of the device allows the user to carry around a laptop as an interface with the Raspberry Pi TDR and use the device in various settings. The wiring schematic and PCB are shown below.

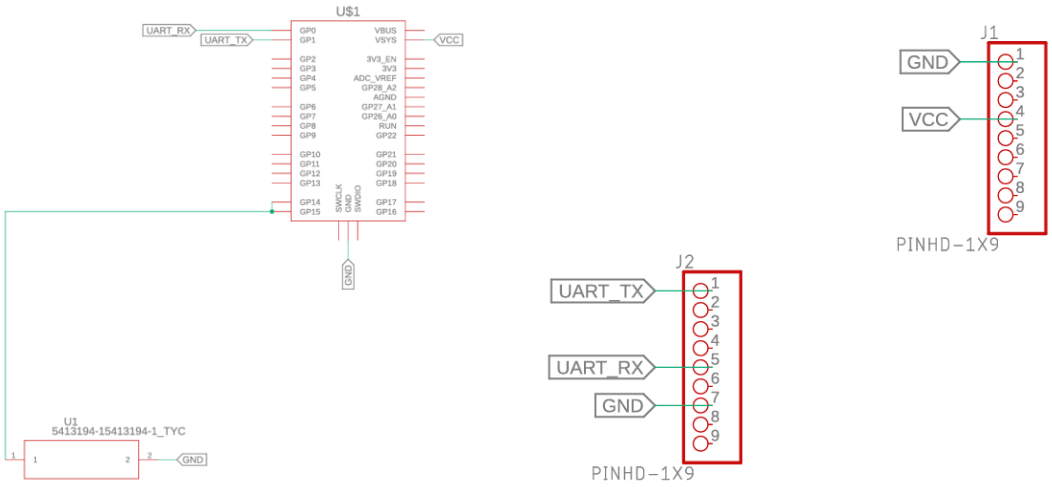


Figure 6: TDR on RP2040 Version 1 Wiring Diagram

The UART to USB converter used was the SparkFun USB to Serial Breakout FT232RL. A breakout board was chosen to reduce the possibility of errors in producing a custom PCB and interchangeability. The wiring diagram can be found in Appendix A. Figure 8 below is the testbench used to verify the wiring before ordering the PCB.

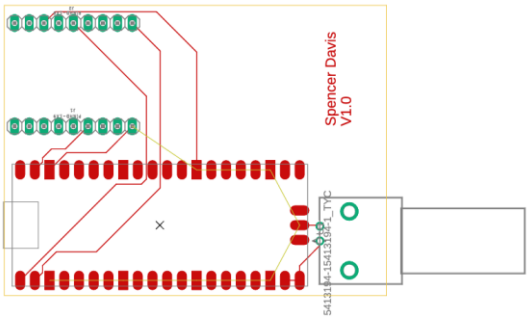


Figure 7: TDR on RP2040 Version 1 PCB Diagram

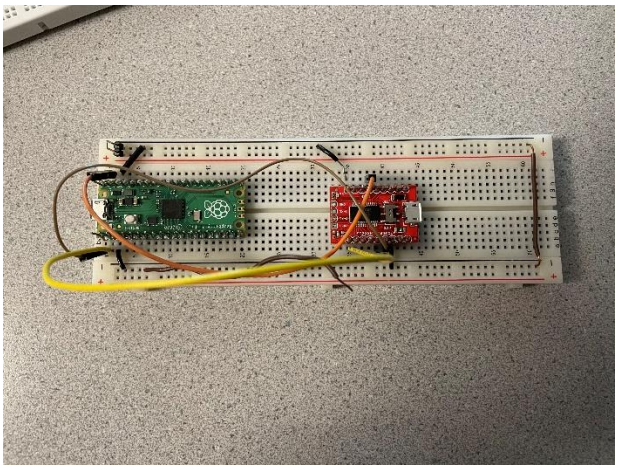


Figure 8: TDR on RP2040 Version 1 Testbench

One major flaw with this device is its usability. Since users must initiate a UART connection with an external program like PuTTY, they must also spend extra time searching for the correct Windows COMM port. Once the user has set up the initial connection with the TDR on RP2040, the initial start menu won't appear to the user until they press a character on the keyboard. This occurs because, on startup, the RP2040 sends the initial terminal menu over UART to the computer before the PC opens a communication port. This adds extra confusion when using the device. Lastly, without substantial effort, button presses on the PC keyboard do not appear on the terminal menu. A simple solution to these few issues would be to add a TFT display and keypad for the user. The new version of the PCB will house an Adafruit TFT LCD, Adafruit Keypad Switches, the Raspberry Pi Pico, and a BNC jack. The wiring diagram and PCB for this is shown below.

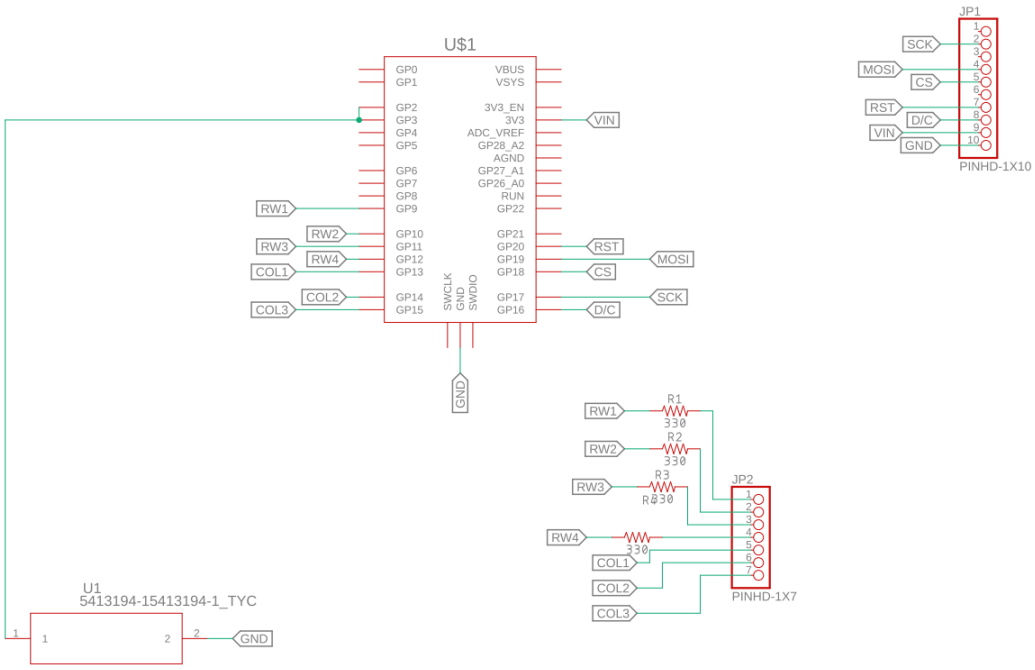


Figure 9: TDR on RP2040 Version 2 Wiring Diagram

The keypad chosen for this is the Adafruit 3x4 Phone Style Matrix Keypad. This device was chosen due to the abundant documentation and the program to decode a button press that had already been written and was quickly implemented into the codebase. The TFT used in this project is the 2.2" 18-bit color TFT LCD from Adafruit. This device uses SPI drivers found online utilizing a PIO state machine and requires little effort to connect to the Raspberry Pi Pico. Below, shown in Figure 11, is the test bench used to verify the wiring before ordering the PCB.

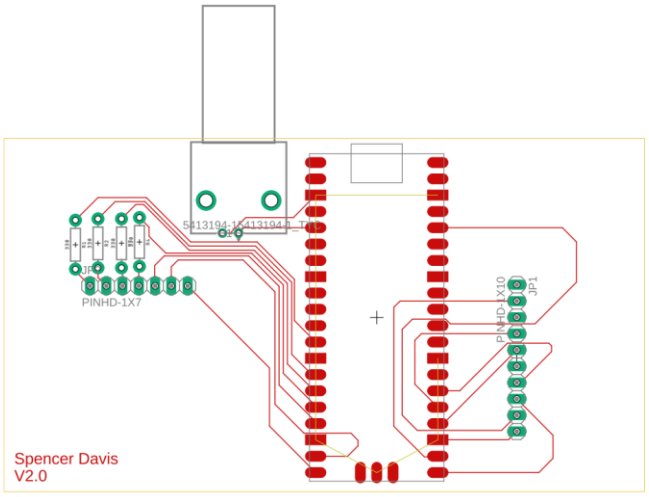


Figure 10: TDR on RP2040 Version 2 PCB Diagram

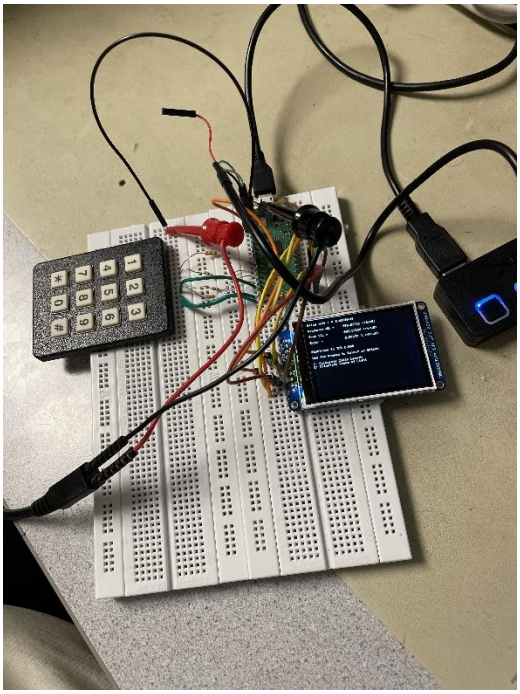


Figure 11: TDR on RP2040 Version 2 Testbench

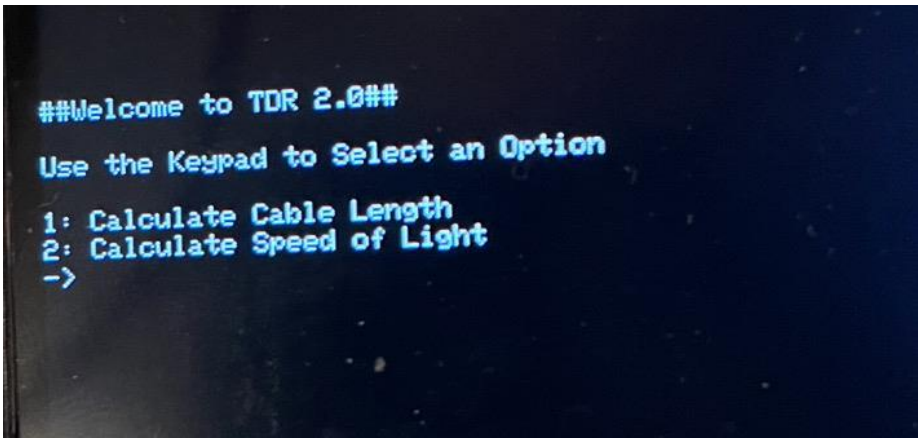


Figure 12: TDR on RP2040 Version 2 UI

This new version of the TDR on RP2040 is significantly more straightforward to use and requires less external effort to read a measurement. The user powers the device by plugging a micro-USB cable into the port, and the user interfaces with their setup by connecting a cable via the BNC connector. The user is given instructions on the screen asking what they want to measure. Depending on their selection, additional user input is required, such as the cable length and the velocity factor of the cable being measured. Once the user enters all the requested inputs, the readings are displayed on the TFT. Some example outputs are displayed below.

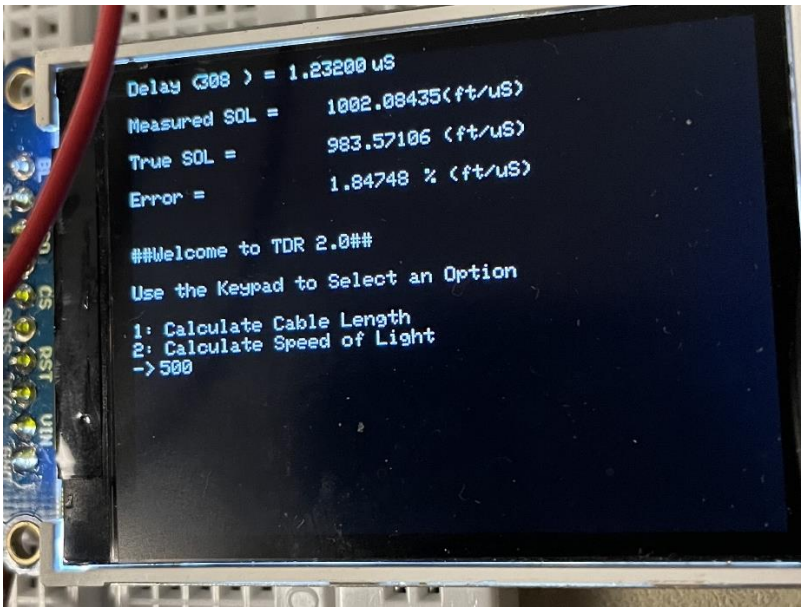


Figure 13: TDR on RP2040 Version 2 Speed of Light Measurement Output

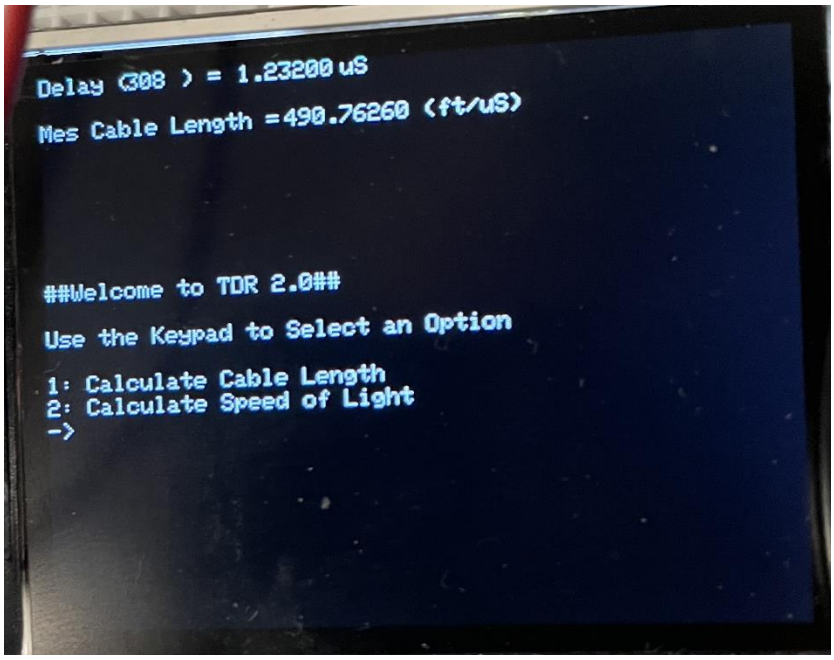


Figure 14: TDR on RP2040 Version 2 Cable Length Measurement Output

Conclusion

This project aimed to develop a program to measure the speed of light over a cable using the RP2040 using time domain reflectometry (TDR). TDRs use the time between an incident signal sent over a wire and the reflected signal off the termination to determine a cable's length. The RP2040 was chosen due to its price point and its PIO co-processors. The PIO co-processors allow for single-cycle instruction execution, maximizing this device's sampling rate. When overclocking the processor, the RP2040 can safely execute instructions at 250mHz. The instructions for controlling the PIO state machines allow us to sample a pin state every other cycle. Using two state machines, we can sample the state of a pin every cycle.

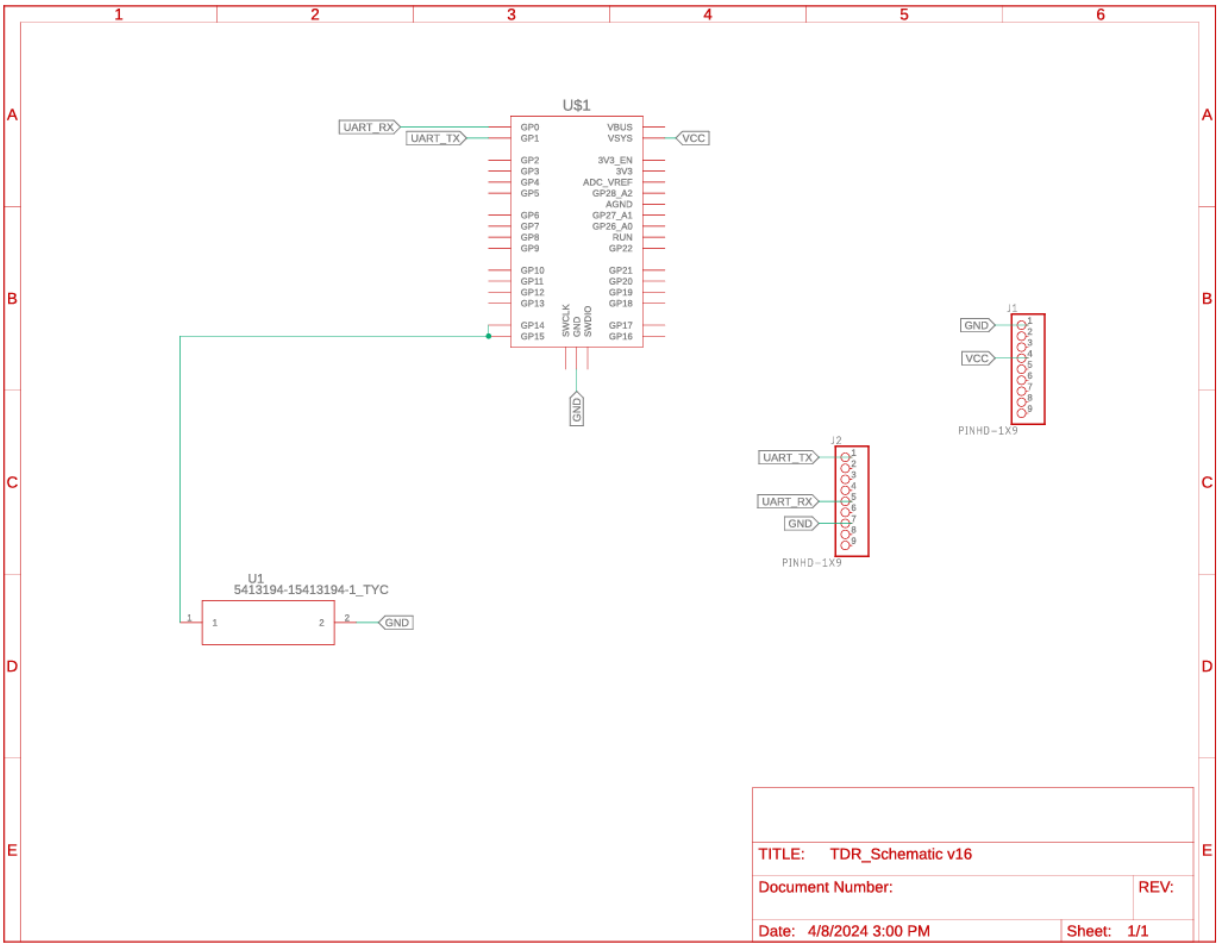
A downside of this measurement scheme and device choice is the precision of the measurement. For a 1000-foot cable, the RP2040 could measure the speed of light within 0.5%. On a 500-foot and 1500-foot cable, the TDR could measure the speed of light within 2% and cable length within 3%. This makes this product ideal for continuity tests or low-risk, high-error-tolerant environments. The measurement scheme requires that the cable be short-circuited to itself, as the PIO state machines can only measure the binary state of a pin. If the cable is shorted to itself, when the reflected signal reaches the signal generator, the magnitude of the signal falls below the voltage, transitioning the high state of the pin to low. This measurement scheme dictates that only the length of the cable and the speed of light through the cable can be measured, not the impedance along the wire. Because of this, the use cases for a device such as this are limited.

After developing a measurement scheme and testing its accuracy, a comfortable, straightforward user interface was next. The device should be able to connect to a variety of cables through a standardized connector. A BNC connector is typically used for devices like this. The device should also have a user interface, allowing a user to understand how to operate the device quickly. A UART to USB converter was used for the first iteration of the device. The main issue is that a user must have a computer and a USB port. This increases the overall cost of using the device and requires users to open a COMM port on their PC. Since the TDR sends out the initial instructions when powering up, the user cannot read the messages until they have opened the COMM port. This is not very clear to the user. A second version of the TDR was developed

to be entirely contained. It houses a TFT display to view results and a keypad for user input. The user can enter cable length and cable velocity factor. The BNC connector used in this version of the PCB is a 75 Ω jack. Looking at the cost of the product versus the accuracy of the readings provided by the device, it is clear that this product has its niche. The device can read a cable's length within 3% of the total length and is priced at under $65. The TDR on RP2040 can successfully calculate the speed of light within 2% of its true value through a cable.

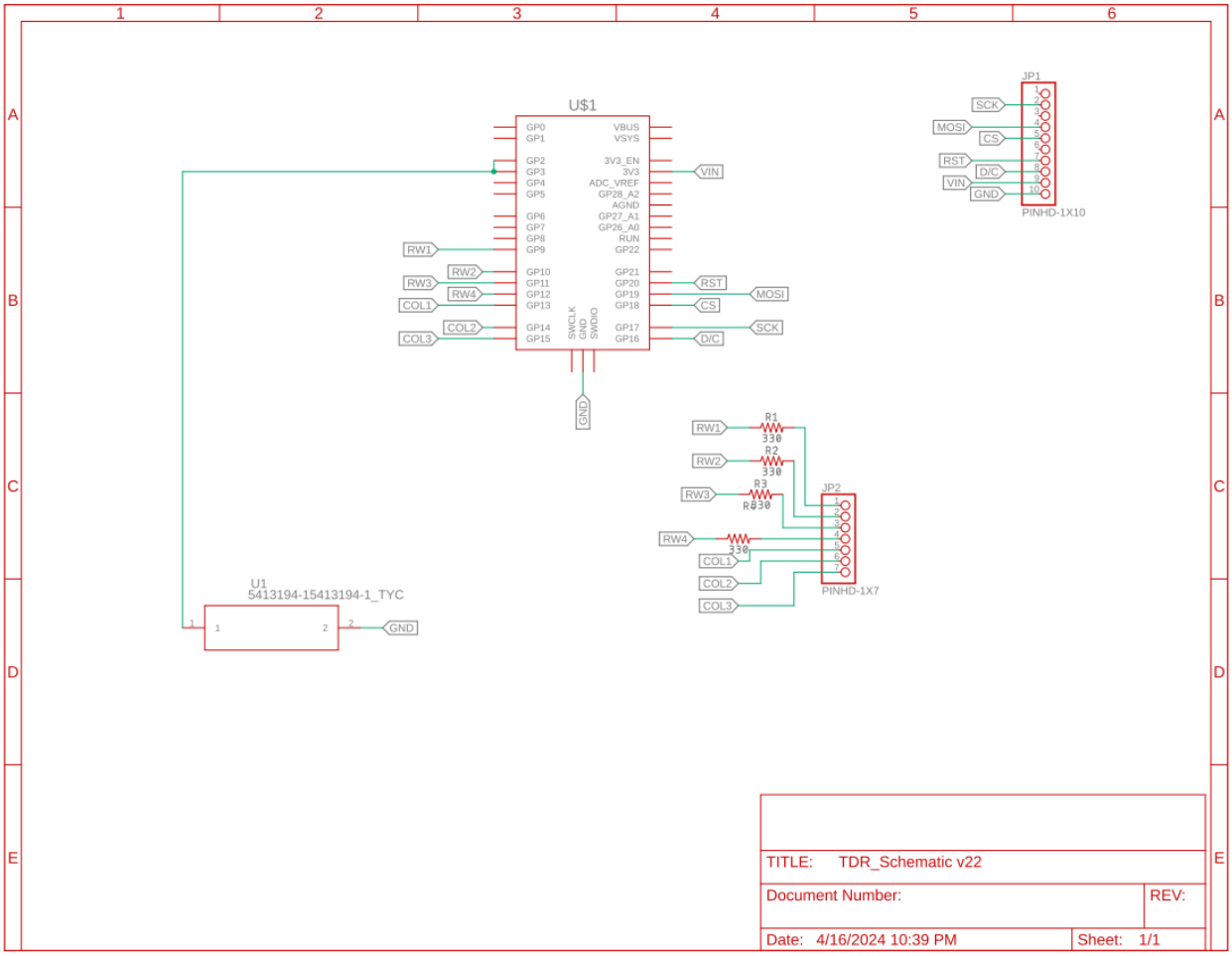
Appendix A

TDR on RP2040 Version 1 Wiring Diagram



Appendix B

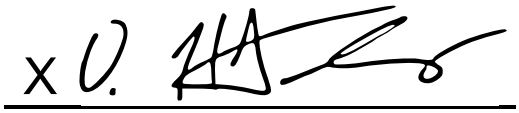
TDR on RP2040 Version 2 Wiring Diagram



Appendix C

Codebase Comments

The code for this project is excluded from submission with approval from the project advisor Professor Adams.



Van Hunter Adams
Dr

References

Duffy, O. (n.d.). *Measuring the velocity factor of coaxial transmission line*. https://owenduffy.net/transmissionline/concept/mvf/index.htm

Raspberry Pi Ltd. (2023). RP2040 Datasheet. Raspberry Pi. https://www.raspberrypi.org/documentation/rp2040/rp2040-datasheet.pdf