Data Acquisition and Timing Error Measurements of Arduino Mega Microcontroller

Sean Tai
QCC Physics Department
mentor Professor R. Armendariz
Undergraduate Research Day 2024

Abstract

The Arduino has a 16MHz oscillator and a 16-bit timer function that can count up to 2^16=65536. Theoretically, by using a function waveform generator, the number of timer counts (TNCT1) between consecutive pulses is equal to: Twaveform generator/TArduino

Where Twaveform generator is pulse period from function waveform generator; TArduino is period of theArduino board oscillator.

However, high frequency pulses which have short periods, result in significant errors in TNCT1 counts.

My research focuses on identifying the factors that influence the error in serial communication and developing methods to reduce it

Materials

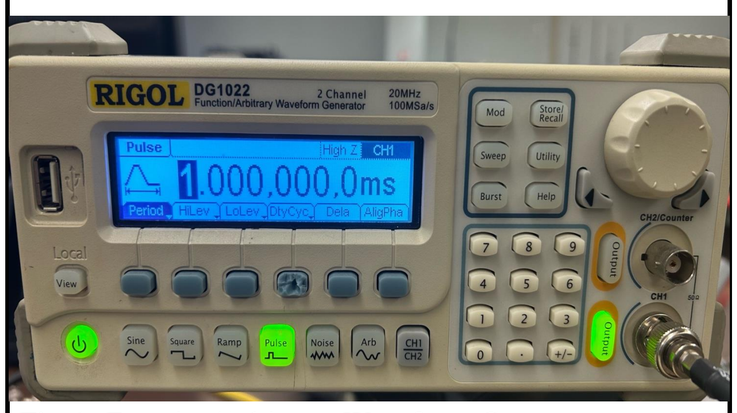

Fig 1. Function/Arbitrary Waveform Generator used to output the voltage pulses

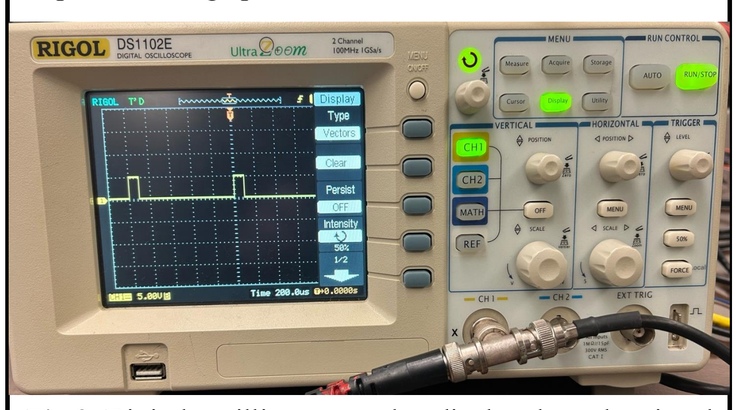

Fig 2. Digital oscilliscope used to display the pulse signal

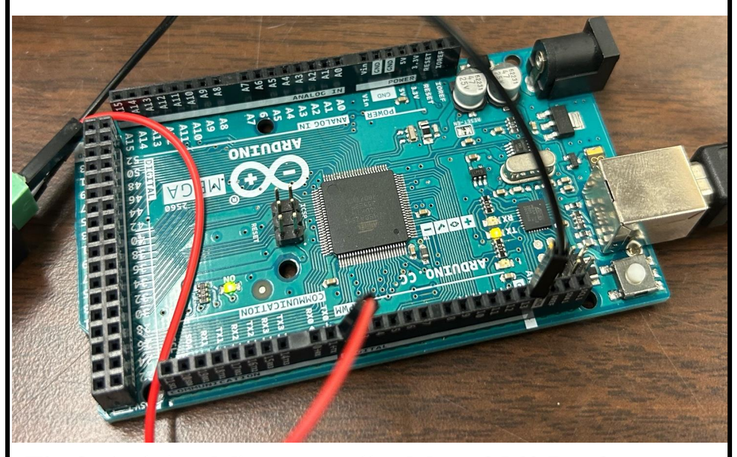

Fig 3. Arduino Microcontroller Mega 2560 Rev3

Method

The waveform generator output is connected with a coaxial cable into two wires, one for the voltage "+" and one for ground "-" to the Arduino's PWM2 input and GND pins. An oscilliscope is also connected in parallel, as shown below

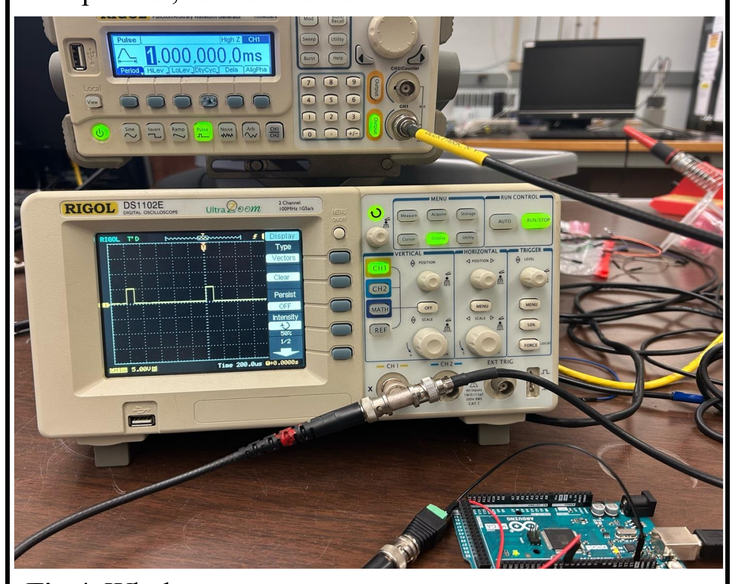

Fig 4. Whole setup

After connecting the Arduino and computer we run the code. The timer count of each pulse is printed out.

```
27    }
28    while(digitalRead(pPS) == LOW){
```

Output Serial Monitor ×

Message (Enter to send message to 'Arduino Mega or Mega 2560' on 'CO

```
1217
17206
33198
49185
65175
15628
31618
47609
63599
14052
30041
```

Fig 5. The code and the printed TNCT1 counts

The Excel spreadsheets is formulated to calculate the time in seconds between pulses from the difference in timer counts, observed periods, theoretical periods, and percent errors.

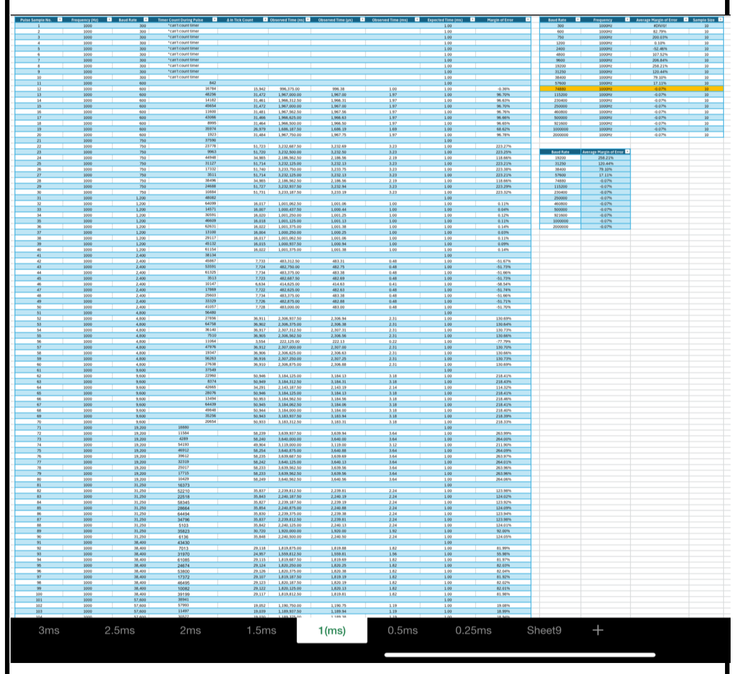

Fig 6. formulated Excel spreadsheet of the data

An Arduino Baud rate of 115200 is required for a pulse frequency of 1500 Hz, which results in approximately a 77:1 ratio.
For pulse frequencies at 2 kHz and higher the baud rate to frequency ratio needs to be higher and higher

Arduino Baud rate	Sig Gen Pulse frequency (Hz)	Baud rate/frequency	% error in measured to expected TCNT1 counts between consecutive pulses
38,400	**500**	**77**	**0.1%**
38,400	600	66	7%
38,400	667	58	19.4%
57,600	**667**	**86**	**negligeable**
74800	**1000**	**75**	**0.1%**
74800	1100	68	2.7%
74800	1200	62	9%
115,200	**1500**	**77**	**negligeable**
115,200	1550	74	6.6%
115,200	1600	72	7.59%
230,400	**2300**	**100**	**negligeable**
230,400	2400	96	4%
250,000	3000	83	22%
460,800	**3000**	**154**	**negligeable**
500,000	4000	125	22.09%
1000,000	4000	250	14.47%
2000,000	**4000**	**500**	**negligeable**

Fig 7. Table of Baud rate, frequency and their ratio

Results and Discussion

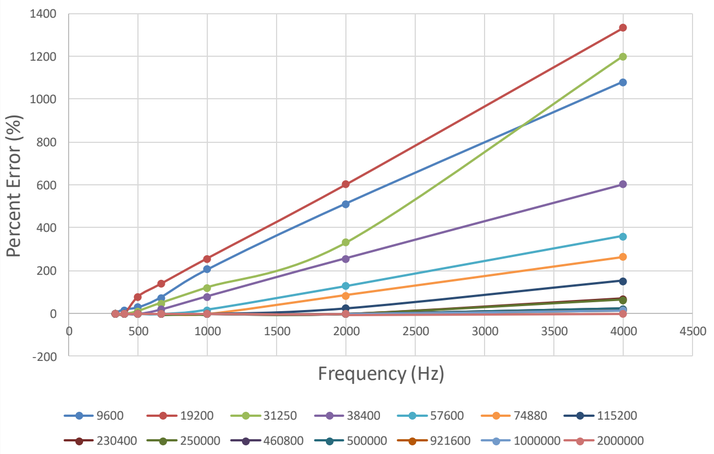

Fig 8. Relations Between Frequency and percent error.

Different plot lines represent different baud rates. As shown as the pulse frequency increases, the percent error increases. However, higher baud rates result in lower percent error.

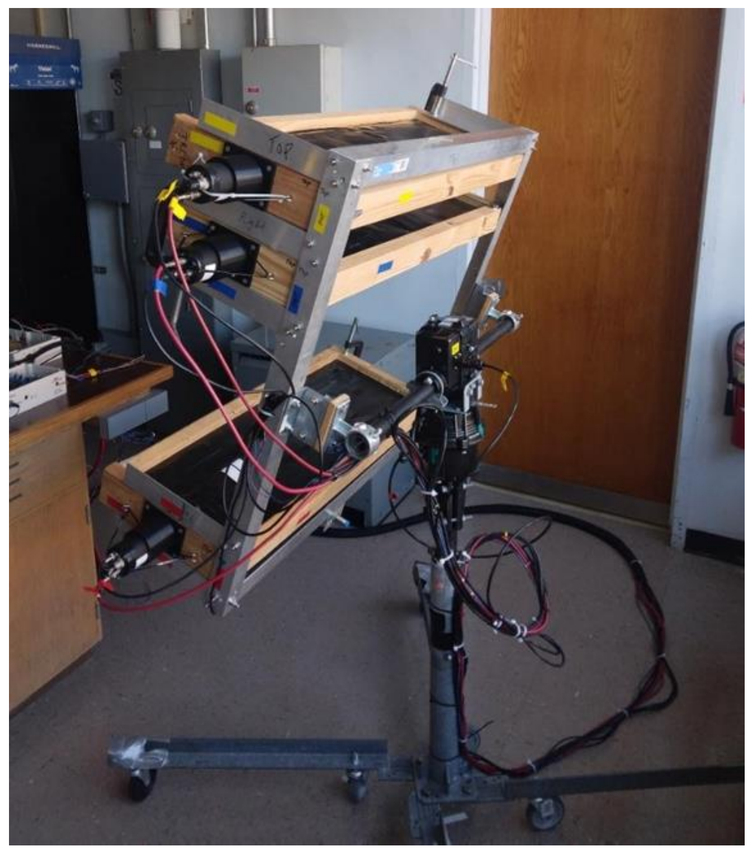

Fig 9. Cosmic ray tracking detector

Fig 10. Voltage pulses from cosmic ray detector

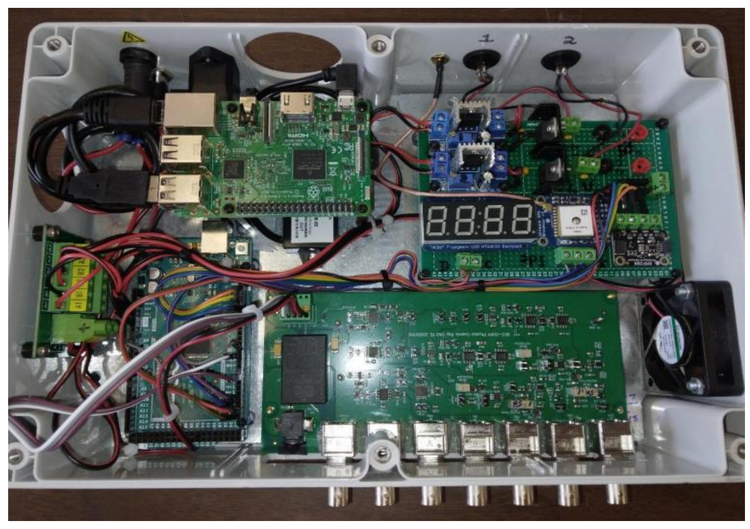

Fig 11. DAQ (Data Acquisition) Box

Conclusion and future work

The shorter the pulse period from the pulse generator requires higher baud rates in communication between computer and Arduino; this requires increasing the baud rate to reduce the error.
I will look into what type of serial communication is used between computer and the Arduino, determine how many bits of data are transferred per each pulse, and see if the results match the table above

Reference

https://forum.arduino.cc/t/maximum-serial-baud-rate-answered/21570/4
https://forum.arduino.cc/t/arduino-bandwidth-rate/26750
https://forum.arduino.cc/t/configuring-timer1-overflow-interrupts/53503

Acknowledgement

We gratefully acknowledge support from the CUNY Research Scholars Program.

QUEENSBOROUGH COMMUNITY COLLEGE | CUNY