Nanyang Technological University

MA4830 – Realtime Software for Mechatronic Systems

Major Programming Assignment

Supervisor:

Prof. Gerald Seet

Student Name:

Jin Zihang U1822185F
Bryant Juspi U1820821E
Cai Yuxin U1822214D
Dylan Yeo U1922111H

Contents

Abstract

A metronome is a device that produces a steady pulse to help musicians play in time. A metronome is commonly used as a practice tool to help maintain a steady tempo while learning difficult passages. It is also used in live performances and recording studios to ensure an accurate tempo throughout the performance or session.

In this report, we will explain on the theory behind a metronome, what are the relevant parameters and how does the program work.

Theory

Wave Generation

We are tasked to produce 4 different types of waves on the oscilloscope: sinusoidal, rectangle, triangle and sawtooth. These generated signals are used as a stimulus for electronic measurements.

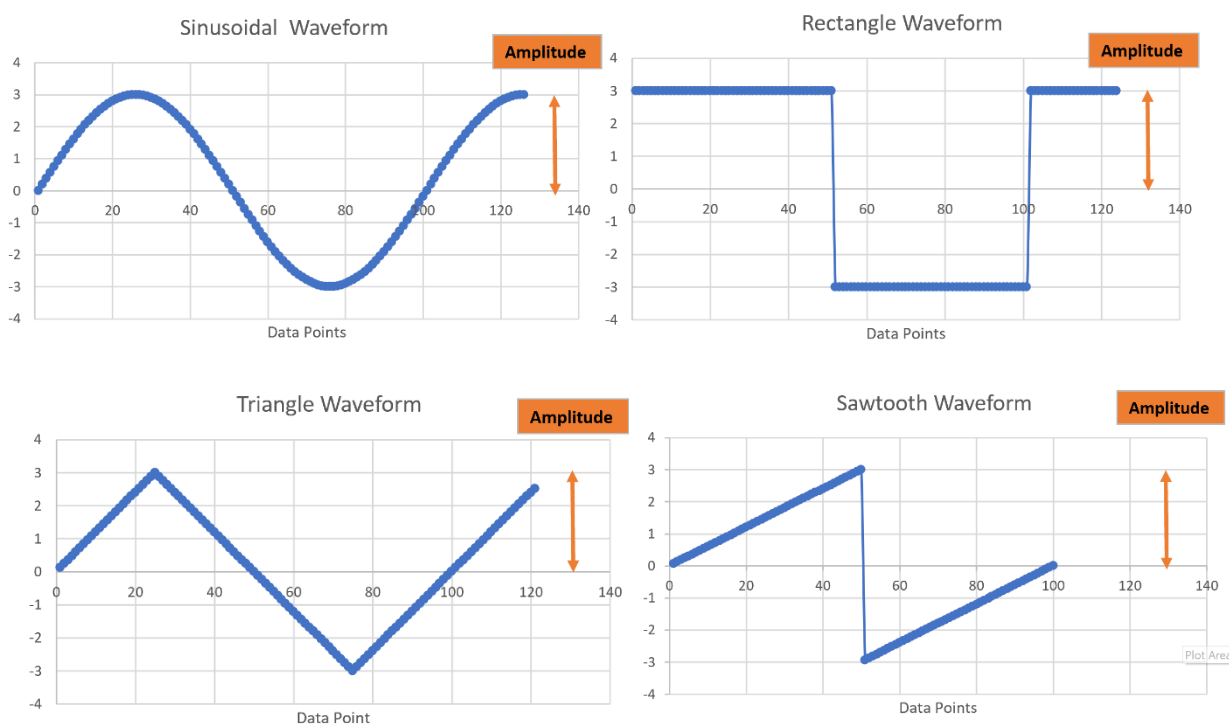


Figure 1

To output the waveform into the oscilloscope, it is only required to calculate for 1 full cycle. For each type of waveform, we calculate for 1 full cycle $0 - 2\pi$ in 100 steps. Throughout trial and error, we found that 100 sample points is a reasonable number of sample points for our purposes.

Relationship between delay & frequency/period

$$delta = \frac{T}{(No.Sample\ Points\ -\ 1)}$$

$$T = \frac{1}{frequency}$$

$$delay = delta \times 1000$$

delta is the time between data points
delay is the time between two consecutive points
f is the number of waves produced by a source per second, it is measured in hertz (Hz).
T is the time it takes for one complete oscillation; it is measured in seconds.

Sinusoidal Waveform

The Sine function has a range of $(-amplitude,\ +amplitude)$ and needs to be scaled from $(0 - 5V)$ for a unipolar output range. We are tasked to connect the D/A output to an oscilloscope and send data to D/A port to visualize the data points on oscilloscope.

- Wave data array calculation

$$data[i] = Amplitude\ *\ sin(\frac{2\pi}{No.Sample\ Points} \times i)$$

$$No.Sample\ Points\ =\ 100$$

$$i = index\ of\ data\ point \in [0, No.Sample\ Points)$$

- Wave data array re-scaling

$$data[i]\ = (data[i] +\ Amplitude) \times \frac{2^N\ -\ 1}{Maximum\ Reference\ Range}$$

$$Maximum\ Reference\ Range = 5$$

$$N = Number\ of\ bits\ in\ DAC0\_data\ port = 12bits$$

Rectangle Waveform

The Square wave has only two levels, $(-\ amplitude)$ for first part of the duty cycle and $(+\ amplitude)$ value for the second part. Another parameter that controls the waveform is the duty cycle which ranges from $0 - 100\%$.

<u>For first part of the cycle:</u>

$$data[i] = -amplitude + amplitude = 0$$

<u>For second part of the cycle:</u>

- Wave data array calculation

$$data[i] = amplitude$$

$$No.Sample\ Points\ =\ 100$$

$$i = index\ of\ data\ point\ \in\ (\frac{No.Sample\ Points}{2}, No.Sample\ Points)$$

- Wave data array re-scaling

$$data[i] = (data[i] + amplitude) \times \frac{2^N - 1}{Maximum\ Reference\ Range}$$

$$Maximum\ Reference\ Range = 5$$

$$N = Number\ of\ bits\ in\ DAC0\_data\ port = 12bits$$

Triangle Waveform

The Triangle wave increases from $(0, +amplitude)$ for the first half of the cycle and decreases to 0 for the second half of the cycle.

<u>For first half of the cycle:</u>

- Wave data array calculation

$$data[i] = \frac{2 \times amplitude}{\frac{No.Sample\ Points}{2}} \times i$$

$$No.Sample\ Points = 100$$

$$i = index\ of\ data\ point \in [0, \frac{No.Sample\ Points}{2})$$

- Wave data array re-scaling

$$data[i] = data[i] \times \frac{2^N - 1}{Maximum\ Reference\ Range}$$

$$N = Number\ of\ bits\ in\ DAC0\_data\ port = 12bits$$

$$i = index\ of\ data\ point \in [0, \frac{No.Sample\ Points}{2})$$

$$Maximum\ Reference\ Range = 5$$

<u>For second half of the cycle:</u>

- Symmetry based on the first half

Sawtooth Waveform

The Sawtooth wave ramps up from 0 to maximum value for one full cycle and repeats again for the next cycle.

- Wave data array calculation

$$data[i] = \frac{2 * amplitude * i}{No.Sample\ Points - 1}$$

$$No.Sample\ Points\ =\ 100$$

$$i = index\ of\ data\ point\ \in\ [0, No.Sample\ Points)$$

- Wave data array re-scaling

$$data[i]\ = data[i]\ \times \frac{2^N\ -\ 1}{Maximum\ Reference\ Range}$$

$$N\ = Number\ of\ bits\ in\ DAC0\_data\ port\ =\ 12bits$$

$$i = index\ of\ data\ point\ \in\ [0, No.Sample\ Points)$$

$$Maximum\ Reference\ Range = 5$$

Analog Input

Potentiometer 0 (Amplitude)

Potentiometer 0 is used to change the amplitude of the wave in real-time.

- Read in the analog value, re-scaling and update the wave amplitude.

$$\frac{2^N}{Maximum\ Reference\ Range} = \frac{Analog\ Input\ Voltage}{Digital\ Output\ (Amplitude)}$$

$$N\ = Number\ of\ bits\ in\ ADC\ data\ port\ =\ 16bits$$

$$Maximum\ Reference\ Range = 2.5$$

$$Analog\ Input\ Voltage = in16(ADC\_Data)$$

- Include an analog input filter to better handle noises or small analog changes. This filter checks for consecutive changes of amplitudes. The update and visualization will only proceed if the difference is greater than the threshold value.

$$amplitude = \begin{cases} f\left(\mathrm{adc}_{\mathrm{in}[1]}\right), & abs(\mathrm{previous}_{\mathrm{adc1}} - \mathrm{adc}_{\mathrm{in}[1]}) > \mathrm{threshold} \\ f\left(\mathrm{previous}_{\mathrm{adc1}}\right) & abs(\mathrm{previous}_{\mathrm{adc1}} - \mathrm{adc}_{\mathrm{in}[1]}) < \mathrm{threshold} \end{cases}$$

$$f\ is\ the\ re-scaling\ function$$

Potentiometer 1 (Duty Cycle)

Potentiometer 1 is used to change the duty cycle of the rectangle wave.

- Read in the analog value, re-scaling and use it to update the duty cycle.

$$\frac{2^N}{Maximum\ Reference\ Range} = \frac{Analog\ Input\ Voltage}{Digital\ Output\ (duty\ cycle)}$$

$$N\ = Number\ of\ bits\ in\ ADC\ data\ port\ =\ 16bits$$

$$Maximum\ Reference\ Range = 100$$

$$Analog\ Input\ Voltage = in16(ADC\_Data)$$

Digital Input

Switch (Waveform)

Switch is used to change the waveform of the wave.

- Read in digital value from in8(DIO_Data).
- Update LED light out8(DIO_Data, dio_switch) and waveform of the wave.

Arrow Key (Frequency)

Keyboard arrow key is used to change the frequency of the wave.

- Read in the keyboard arrow key digital value.
- Update the frequency of the wave.

Programme Description

Program Flow

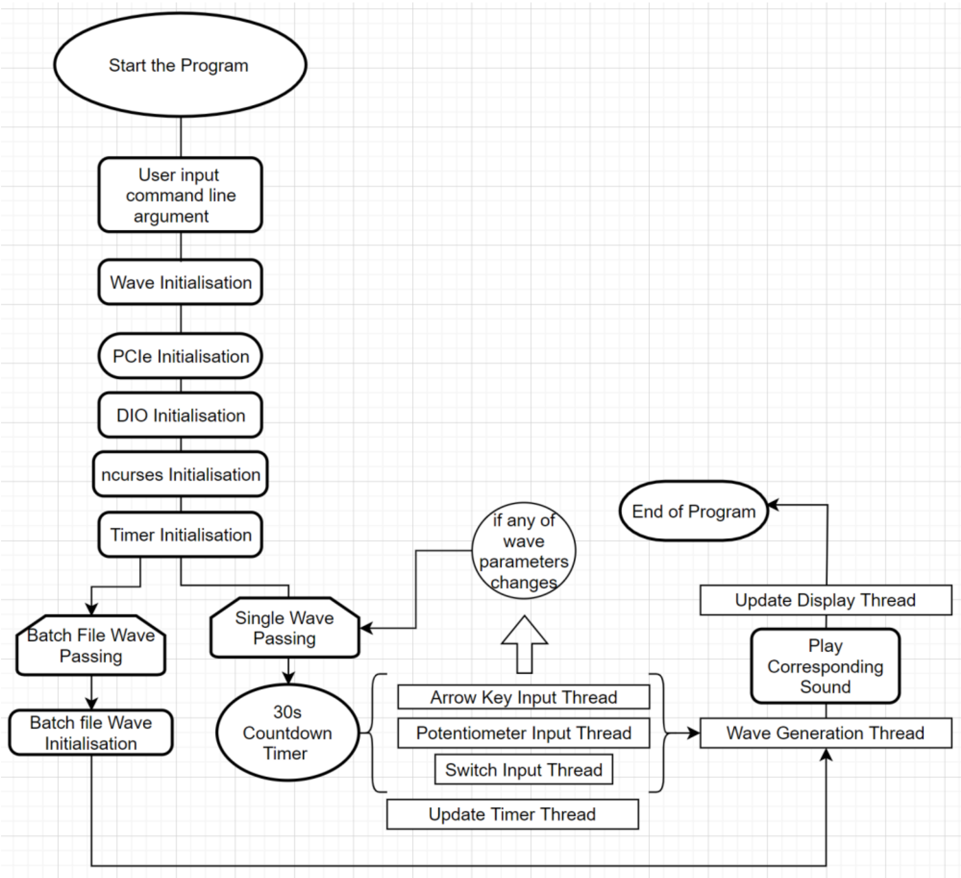


Figure 2. Program Flow

Features

Multiple Execution Modes

User can select different modes based on different command line arguments. There are 3 execution modes:

- Streaming Processing mode, which allow users to user the sensors to adjust the wave parameters in real time.
- Setting parameter mode, which allow users to pass multiple parameters through command line arguments.
- Batch File mode, which allow user to pass batch wave configuration in one time and display in sequence.

Oscilloscope display

Users can see 4 different types of waveforms visualized based on the specified wave configuration on oscilloscope. **Vpp** defined is set to **5V** as maximum.

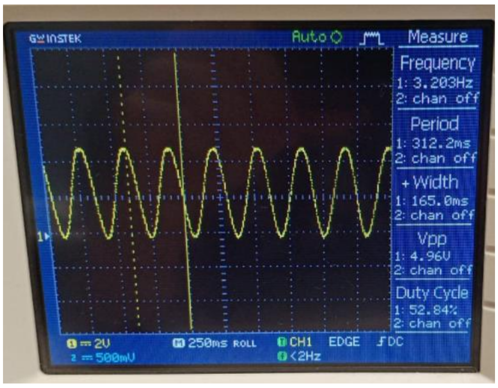
<table>
<tr><th>Type of Wave</th><th>Display waveform on oscilloscope</th><th>Type of Wave</th><th>Display waveform on oscilloscope</th></tr>
<tr><td>Sine</td><td>

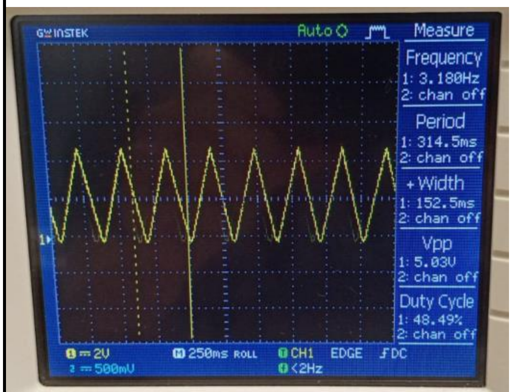
Figure 3. Display of Sine wave</td><td>Triangle</td><td>

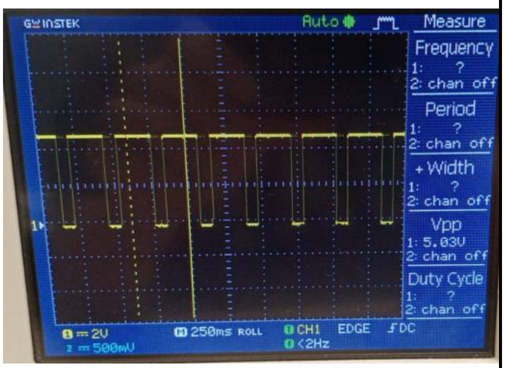
Figure 4. Display of Triangle wave</td></tr>
<tr><td>Rectangle</td><td>

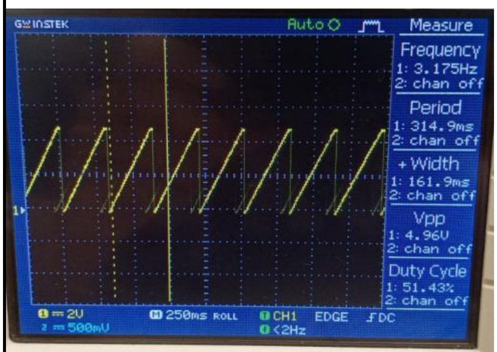
Figure 5. Display of Rectangle wave</td><td>Sawtooth</td><td>

Figure 6. Display of Sawtooth wave</td></tr>
</table>

Dynamic input (PCIe Board)

Users can change wave configurations from sensors in real-time. The wave configuration on oscilloscope will be updated in real-time.

- 4 Switches to select which waveform is displayed.
- 2 Potentiometer to adjust amplitude and rectangle wave's duty cycle respectively.
- 4 LED lights to indicate which waveform is being displayed.
- Up and down arrow key on keyboard to adjust frequency.

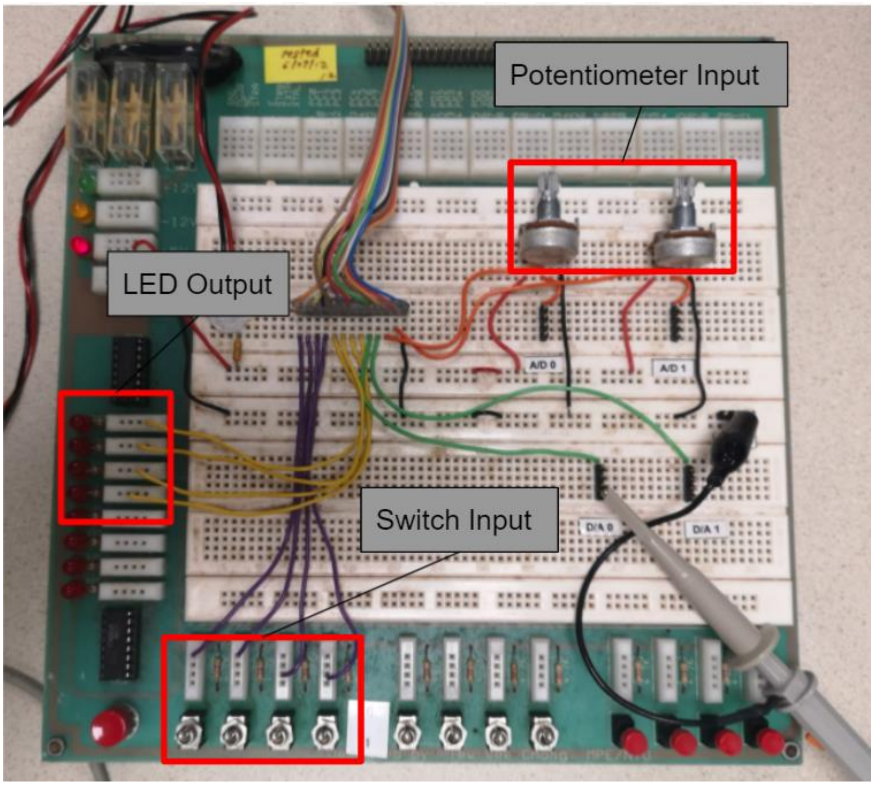


Figure 7. Electrical Circuits

Sound output

- A tone will be played at the peak of each type of wave.
- The peak of each wave is calculated based on the period/frequency.
- The pitch of the tone can be adjusted by changing the amplitude of the wave.
- The duration between each tone is determined by the period/frequency of the wave.
- The visual output displayed on the oscilloscope and ncurses display are well synchronised with the auditory sound module.

Highlights

Multithreading

Multiple threads are being ran asynchronously and coordinated using "*Mutex*":

- Read switches on PCIe board.

- Read potentiometer on PCIe board.
- Read arrow keys on keyboard.
- Generate waves to be displayed on oscilloscope.
- Update ncurses display.
- Update Timer for 30 second countdown.

Independent threads can access the same variables in memory. Threads can share the same memory space and read or write global data which updates the real time wave configuration.

Noise Handling

To handle random analog noises, a simple filtering method is used, in which the analog input must be greater than the specified **threshold** to be considered as an input for the oscilloscope.

Wave Generation Optimization

Wave data array are only generated/calculated once for every pair of parameters. By doing these, it reduces redundant operation by calculating the same data array. The wave generation is strictly limited by a timer count down.

Robustness

To increase the robustness of the application, various signal handler and timeout system are used.

- Trapping **Ctrl + C** using *SIGINT* to terminate the application safely.
- Using a default timeout to automatically exits the application.
- When the wave configuration from user argument is not fully defined, the program would auto-fill up the undefined parameters with default setting.

Logging System

To give feedback to user regarding various errors and warning, logging system are used. Which are displaying various errors and warnings to users. For example: *Exceed defined limit, Invalid parameters, etc.*

Limitations

Parameter constraints:

1. Amplitude constraint: 0 - 2.5
2. Frequency constraint: 0 – 300
3. Number of samples → 100

Flow Chart

Wave Initialisation

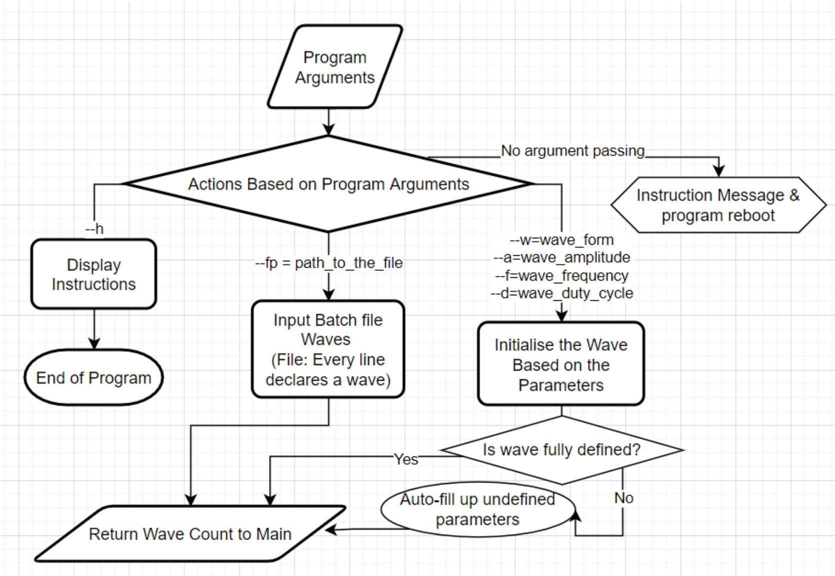


Figure 8. Wave Initialisation Flowchart

Hardware Update Input (3 threads – switch, potentiometer, and arrow key)

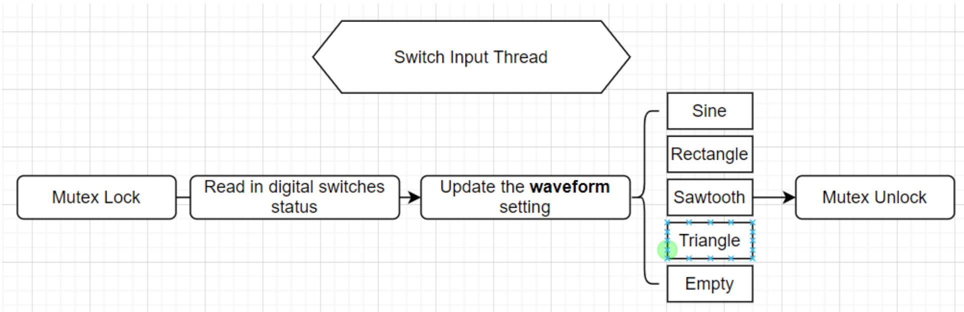


Figure 9. Switch Input Thread Flowchart

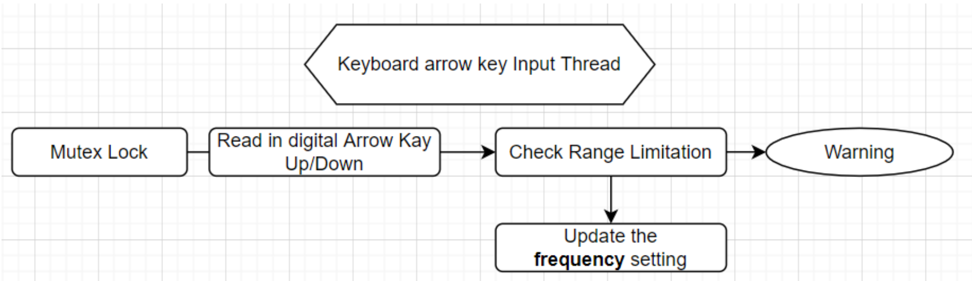


Figure 10. Arrow key input thread Flowchart

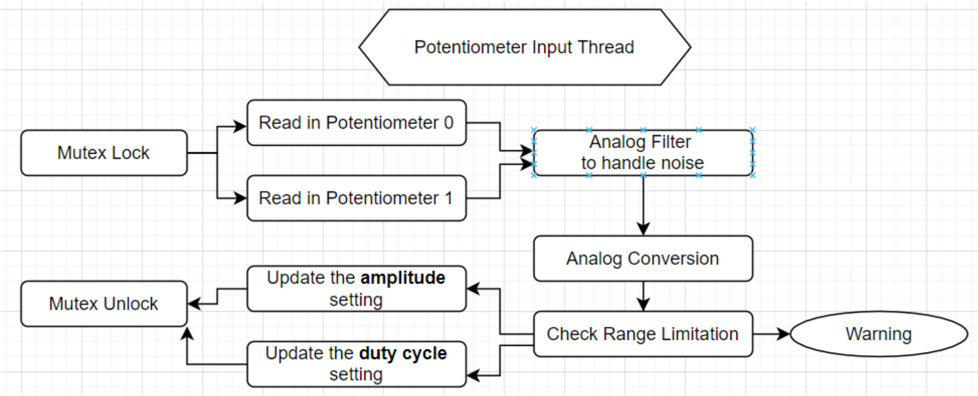


Figure 11. Potentiometer Input thread Flowchart

Wave generation thread + Sound module

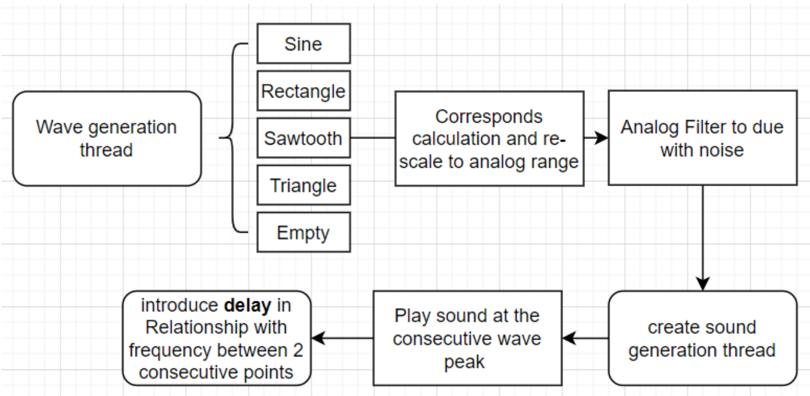


Figure 12. Wave generation thread Flowchart

Sample Run

In this section, we show sample runs of the program. This includes sample runs for each mode and sample runs for various invalid inputs.

Setup

To generate an executable that can be run. Compile and link the source and header files as instructed below.

```
cd ca2
su -> system
chmod +x compile.sh
./compile.sh
```

Or

```
cc -c functions/helper.c functions/initialization.c functions/input.c functions/logging.c functions/pcie_control.c functions/sound.c functions/timer.c functions/wave_generator_pcie.c
cc -lm helper.o initialization.o input.o logging.o pcie_control.o sound.o timer.o wave_generator_pcie.o -o main main.c -lncurses
```

To look at the application instructions run the instruction below.

```
./main --h
```

Stream Processing

```
./main
```

Batch Processing

Through command line argument

```
./main --w=sine
./main --w=triangle --f=30 --a=2
./main --w=rectangle --f=10 --a=1 --d=50
```

Through file input

```
./main --fp=./data.dat
```

Application GUI

Normal UI:
This is the standard ncurses GUI users will see on screen. As users changing the parameters through PCIe input, synchronised wave configuration will be updated on screen in real-time.

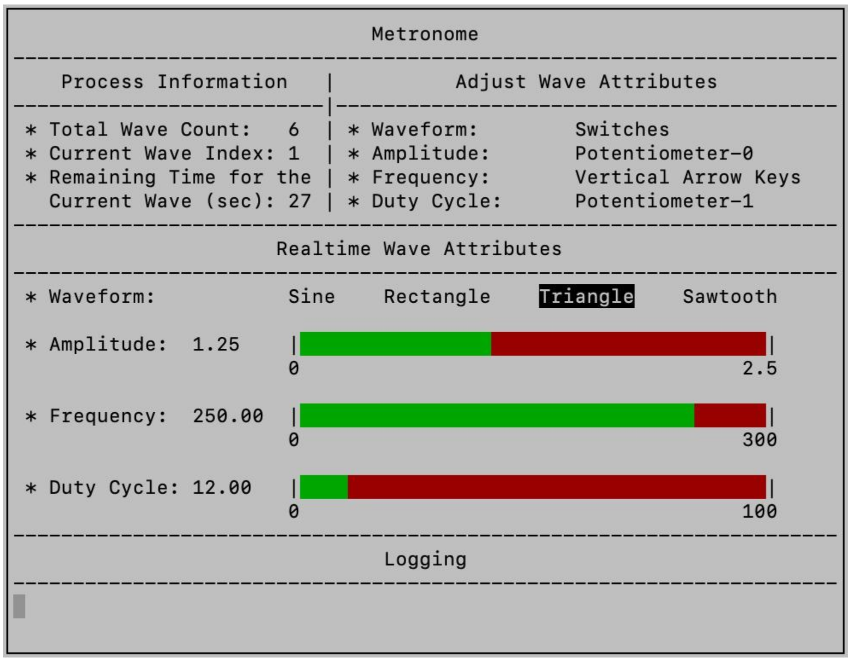


Figure 13. Normal Application Interface

Error message UI:
This is the error message UI users will see on screen. Once errors occur such as, the batch file failed to open, failed to create a timer, Invalid program arguments and Invalid parameter value exists, there will be error message posted on the UI with instructions. User needs to restart the program once they solve the problem following the help mode. More details please refer to logging.c.

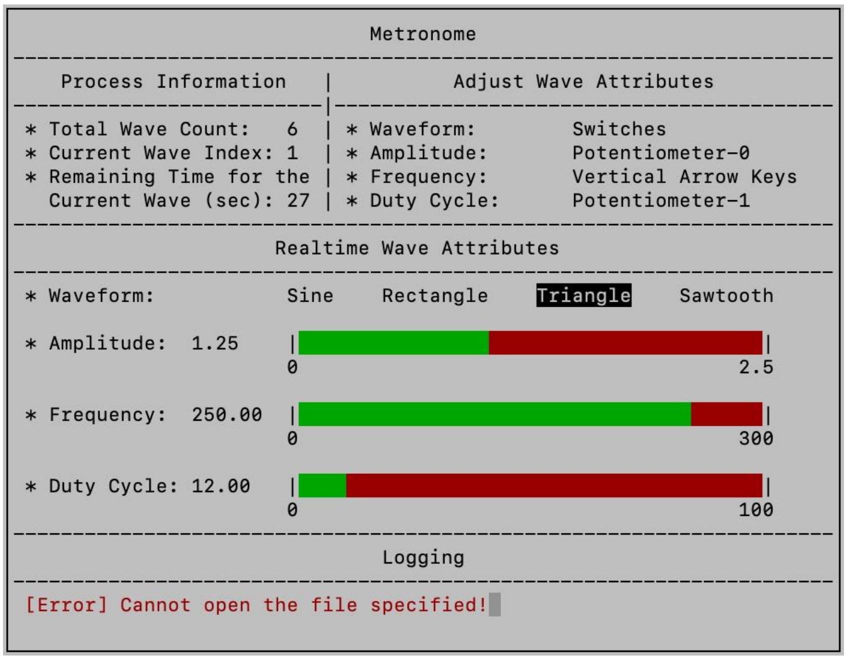


Figure 14. Error Message Application Interface

Info message UI:
Info message will only be triggered once the program is terminated. This UI provide a summary for both batch processing and stream processing.

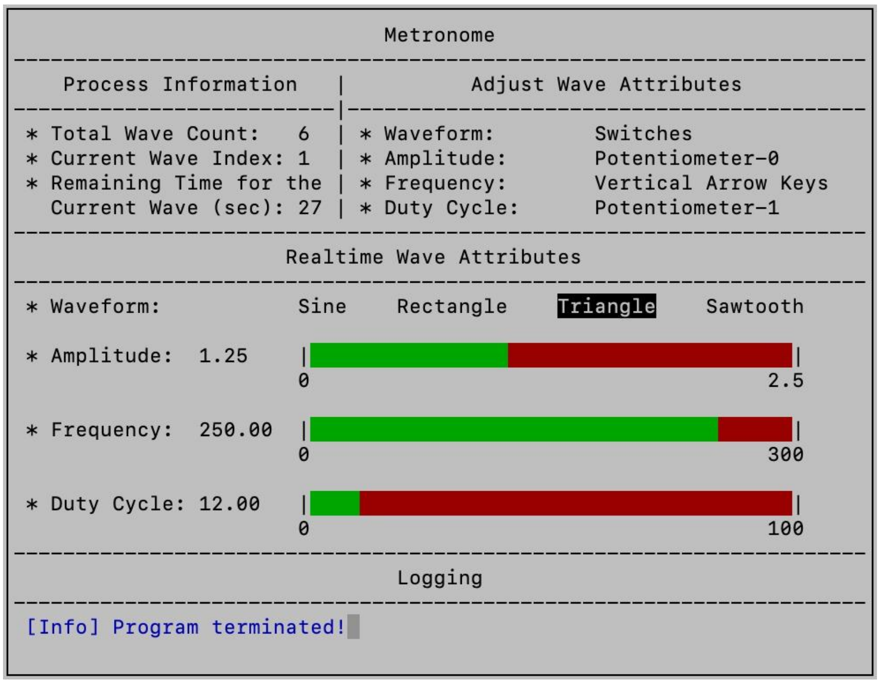


Figure 15. Info Message Application Interface

Sample Video

https://drive.google.com/file/d/11guEUN7RYbAJ2clX02FJ1CldGUrrDjim/view

Appendix 1: Source Code

```
D:.
│   compile.sh
│   data.dat
│   main.c
│   main.h
│
├───datatypes
│       enum.h
│       struct.h
│
└───functions
        helper.c
        helper.h
        initialization.c
        initialization.h
        input.c
        input.h
        logging.c
        logging.h
        pcie_control.c
        pcie_control.h
        sound.c
        sound.h
        timer.c
        timer.h
        wave_generator_pcie.c
        wave_generator_pcie.h
```

Files		**Description**
	compile.sh	All commands needed to compile and link the program
	data.dat	Batch file format
	main.c	Main Program to run
	main.h	
Datatypes	enum.h	Define type of waveform to enumerate
	struct.h	Define global variable wave structs contains all configuration parameters
Functions	helper.c	Helper Functions to prevent code duplication
	helper.h	
	initialization.c	Initialization Functions for different execution mode
	initialization.h	
	input.c	Read digital and analog input from switches, potentiometers, and keyboard arrow key
	input.h	
	logging.c	Ncurses Related Functions: - Initialize ncurses window with default static attributes - Clear one logs line in the logging box - Logs an error if there is an error when slicing a string - Logs help instructions for using the application
	logging.h	
	pcie_control.c	PCIE Initialization Functions: - Initialize PCIe base registers - io registers configuration
	pcie_control.h	
	sound.c	Sound module related functions: - Generate sound at consecutive wave peak
	sound.h	
	timer.c	Timer related functions: - Initialize a timer with default configuration values - update timer countdown if there is a change in wave parameters
	timer.h	
	wave_generator_pcie.c	Wave Generator Functions

	wave_generator_pcie.h	- GenerateWave: General generate wave functions including sine, rectangle, triangle, sawtooth waves

Table 1. Descriptions of file directories

Appendix 2: Team Members

Bryant Juspi

Jin Zihang

Cai Yuxin

Dylan Yeo