NANYANG TECHNOLOGICAL UNIVERSITY

School of Mechanical and Aerospace Engineering

MA4830 Realtime Software for Mechatronic Systems

Major CA Project Report

From left to right:

Group Members:	**Matric Number:**
Lim Jia Jing	**U2021224D**
Liu Qi Yuan	**U2020275D**
Song Ke Yan	**U2020352K**
Daniel Low Teck Fatt	**U2022053E**

Table of Contents

List of figures

List of tables

1. Introduction

This project showcases a program written in C programming language that generates 4 different types of waveforms: sine wave, square, saw-tooth wave, and triangle wave. This program obtains inputs from keyboard, switches, and potentiometer to perform various function such as, changing the waveform, frequency and amplitude of the wave and switching the program on and off.

1.1. Key Features

1. Wave parameters setting – The wave parameters include waveform, amplitude, and frequency of the wave, which can be changed via either keyboard input or analog input.
    a. Keyboard input: The program activates the user input function by toggling the user input switch (SW2 – SW4). The user can change the waveform and the frequency of the wave by entering new value in the command prompt. Any changes to the wave parameters will trigger a sound alert to notify the user of the modification.
    b. Analog input: The program allows flexibility in altering the amplitude of the wave. The user can easily adjust the amplitude by turning the potentiometer, whose resolution matches the amplitude scaling factor that ranges between 0x00 and 0xffff.
2. Read/write file on hard-disk/removeable disk – The program automatically saves the waveform data at every timestep into an array, which is then written to a readable text file to ensure the repeatability of the wave. This file is stored in the same folder as the program file and can be copied to other directories.
3. User Interface (UI) – A user-friendly UI is programmed to guide the user step-by-step in setting the wave parameters. All instructions are displayed in the command prompt, and the range of values for each parameter is stated clearly. The program will generate an error message to notify the user of incorrect inputs and allow multiple attempts until the correct input is entered.
4. Real-time techniques – This program implements real-time programming techniques such as threading, interrupt, and mutex. Multiple threads are used to run multiple functions while keeping the other threads running.
5. Sounds to alert users that change has been made – While potentiometer reading or toggle switch is changed, an alarm sound will be emitted.

2. Program Description

This section introduces the functions involved in wave generator program. Overall, the program functions can be categorized into normal functions and threading functions, working collaboratively to generate desired waveform with user specified frequency and amplitude.

2.1. Normal functions

Table 1 Normal functions brief description

Function name	**Descriptions**
void interrupt_handler();	Serves as a signal handler for interrupts, particularly the "Ctrl + C" signal (SIGINT).
void reset_settings();	Perform cleanup and reset operations before exiting the program. Unreachable while generating wave.
void parse_arguments(int argc, char *argv[]);	Responsible for extracting and validating command-line arguments passed to the program when it is executed. - argc: number of command-line arguments. - argv: an array of strings containing the arguments.
void display_usage();	Informing users about correct usage of the program. When called, it prints out command line argument options.
void update_settings();	Responsible for updating the waveform type and frequency settings based on user input. It displays the available options and waiting for the correct user input: - Update Options: "1" for waveform / "2" for frequency. - Available waveform: sine / triangle / sawtooth / square. - Valid frequency range: 1-1000 Finally, it signals the **check_input()** thread to continue executing and wait for user input or changes in settings.
void check_waveform();	Responsible for validating and setting the waveform type based on user input. Upon checking invalid user input, the function continuous to loop until receives a valid input. - Valid inputs (string): sine / square / triangle / sawtooth

void check_freq();	Responsible for validating and setting the frequency based on user input. Upon checking invalid user input, the function continuous to loop until receives a valid input. - Valid inputs: (int) 1 – 1000
void check_toggle();	Responsible for handling digital port functions, specifically toggling switches. It monitors the state of the toggle switch, determining when to interrupt the program or accept new input.
void check_pot();	Handles the ADC input from a potentiometer. It reads from potentiometer in terms of voltage and convert it into amplitude of wave. Have a fluctuation range to avoid constant detection.
void create_data_arr();	Responsible for dynamically allocating memory for storing waveform data points. It calculates the number of data points from the desired frequency and allocate memory for the dynamic data size.

2.2 Realtime Technique

This program accepts user inputs and respond to the changes in real time. To perform real time operations, multi-threading is required. Multiple functions will be using different threads to operate separately, without waiting for completion of a function before proceeding.

void* check_input();

void* generate_data();

int generate_wave();

Mutex is used to lock the global variables such that there is no concurrent access of data or other resources. Conditions variables are used to control (lock) the execution of the threads until a unlock signal is released. There are 3 conditions: 0: initial, 1: generate wave, 2: generate data

The main() function is the primary thread that is running. This main() function returns the generate_wave() function, and creates two other threads for check_input() and generate_data() using pthread_create().

check_input() is an infinite function that will check for input signals forever. When a change is detected, it will change the condition to 2 and release unlock signal to generate_data() to start generating data.

generate_data() will run once at the start to generate data for generate_wave(). Once the wave is being generated, the condition will be set to 1, then this thread will be locked and will wait for condition change and unlock signal from check_input(). Meanwhile, it will write to a file named "wave.txt" that saves the generated data.

generate_wave() will continuously output data to the board to generate wave on oscilloscope. It will run indefinitely except when condition is 2 which is when generate_data() is running, or when a keyboard input is triggered which is speculated to block all the input/output streams. It will read data from "wave.txt" by iterating through each line in the file.

2.3 Frequency conversion

We discovered that the generate_wave() function will not output the desired frequency with its default for loop output structure. This is because each iteration will take a constant time to be executed, and the time to output all the data in the for-loop to the board determines the period of the wave, hence the frequency of the wave. By testing, the relationship between the number of iterations in the for loop and the actual frequency is tabulated in Table y.

Table 2 - freq variable in program vs actual output frequency

Freq (in program)	Actual frequency (Desired frequency by user)
87000	1
8700	10
870	100
87	1000

It is found that there is an inverse proportional relationship between the freq variable in program and the actual output frequency in oscilloscope. The variable to determine the number of iterations will be renamed to freq_points, which is the number of points/iterations needed. Let x = user input frequency, and y = number of iterations needed. The relationship is governed by the equation:

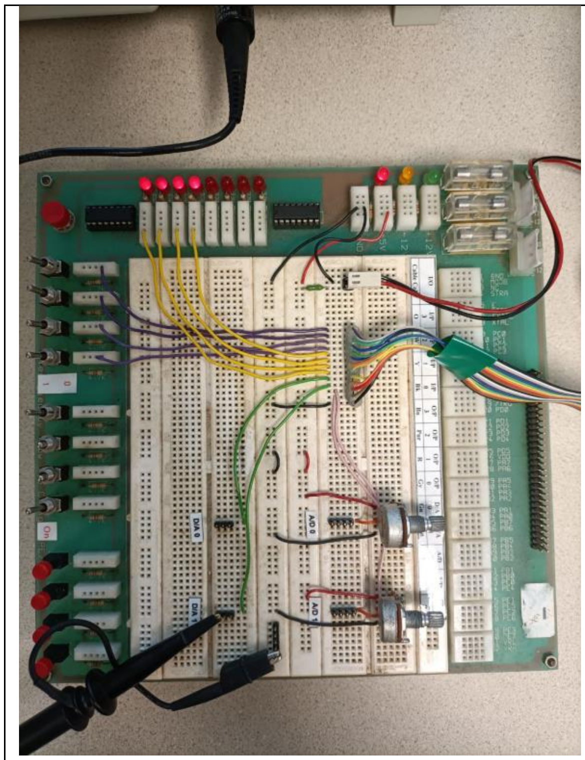
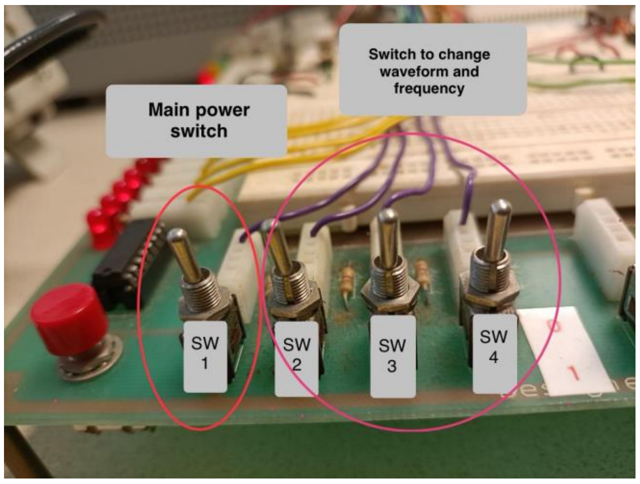
$$y = \frac{87000}{x}$$

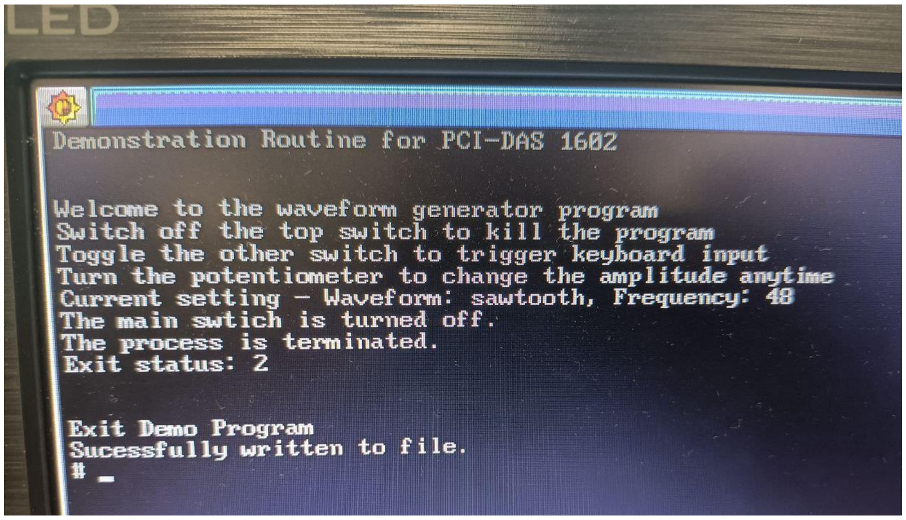
3. Program Execution

3.1 Command Line Arguments

Before the start of the program, ensure that the main power switch (SW1) is on by switching to the left as seen in Figure 2. The toggle switches are on when it is toggled to the left, and off when it is toggled to the right. When the main switch is off, the program cannot start shown in Figure 2. The program can be started by following Command Line Arguments (CLA) usage: ./ca2 [waveform] [frequency].

Table 3 - PCI Board port inputs

 *Figure 1 - Board layout*	 *Figure 2 - Switch labels*



```
Demonstration Routine for PCI-DAS 1602

Welcome to the waveform generator program
Switch off the top switch to kill the program
Toggle the other switch to trigger keyboard input
Turn the potentiometer to change the amplitude anytime
Current setting - Waveform: sawtooth, Frequency: 48
The main swtich is turned off.
The process is terminated.
Exit status: 2


Exit Demo Program
Sucessfully written to file.
# _
```

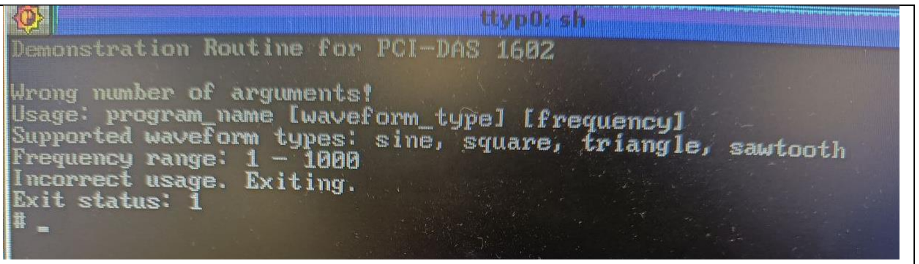
Figure 3 - Output when the main switch is not on

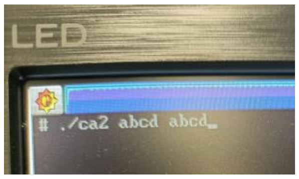
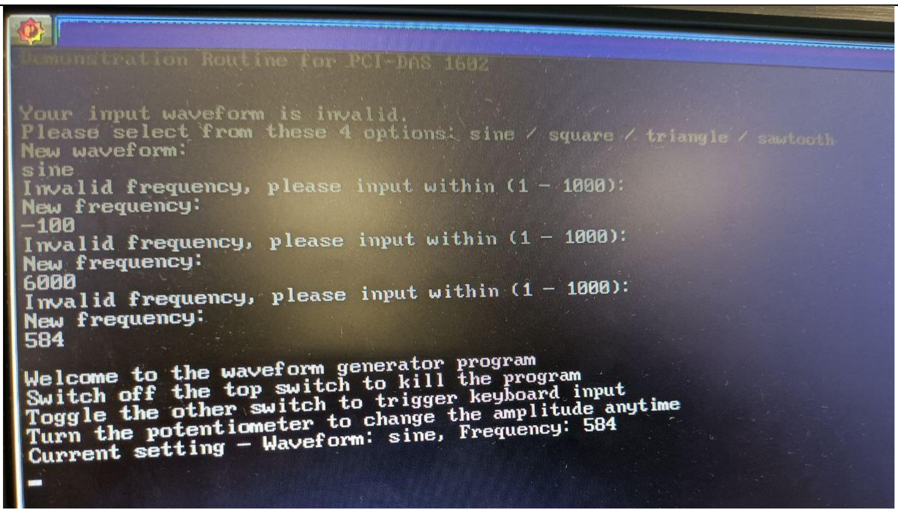
After switching on the main switch, and trying to execute the program with the wrong command *./ca2*, the following outputs will be shown to the user with a few information. Firstly, the UI will let the user know the correct usage, the available waveform options, the accepted frequency range, and the exit status. Exit status = 1 means that the program exits with incorrect CLA error. Table 4 Figure 4 is a screenshot of the output.

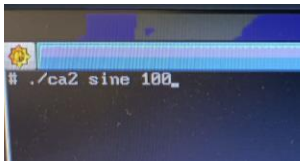
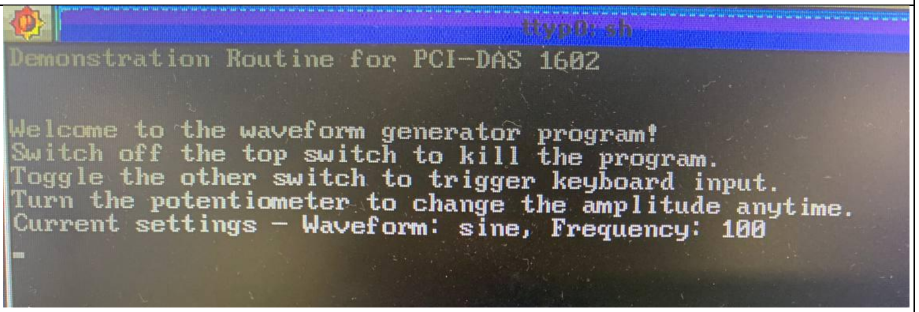
If the user inputs the correct number of arguments but invalid arguments, the program will prompt the users with available options again and ask for input until user inputs correctly. For waveform, the input must be from the 4 options: sine, square, triangle, and sawtooth; while for the frequency argument, the input must be within the range from 1-1000. When all the inputs are correct, a welcome message with instructions for use to change amplitude and waveform, and the current settings. Table 4 Figure 6 shows the screenshot output of UI prompting for correct inputs repeatedly.

When the correct arguments are given, the program will directly show the welcome message. Table 4 Figure 8 shows the screenshot output of welcome message from correct prompts.

Table 4 - CLAs vs outputs

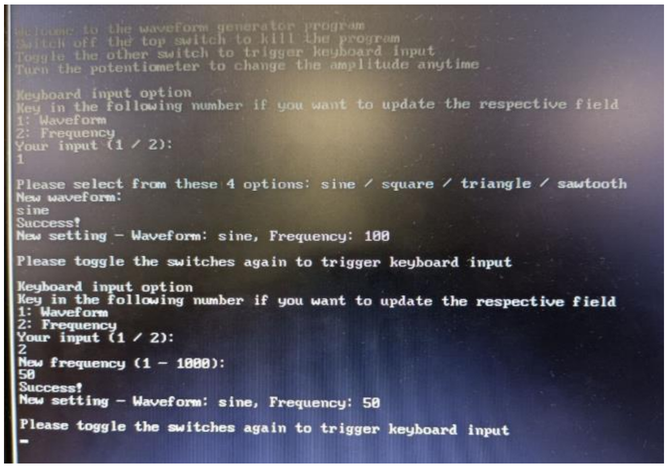
./ca2	 `ttyp0: sh`
	`Demonstration Routine for PCI-DAS 1602`
	
	`Wrong number of arguments!`
	`Usage: program_name [waveform_type] [frequency]`
	`Supported waveform types: sine, square, triangle, sawtooth`
	`Frequency range: 1 - 1000`
	`Incorrect usage. Exiting.`
	`Exit status: 1`
	`# _`
	
	Figure 4 – Output for wrong number of arguments

./ca2 abcd abcd
 `# ./ca2 abcd abcd_`
Figure 5 - Invalid arguments
	 `Demonstration Routine for PCI-DAS 1602`
	
	`Your input waveform is invalid.`
	`Please select from these 4 options: sine / square / triangle / sawtooth`
	`New waveform:`
	`sine`
	`Invalid frequency, please input within (1 - 1000):`
	`New frequency:`
	`-100`
	`Invalid frequency, please input within (1 - 1000):`
	`New frequency:`
	`6000`
	`Invalid frequency, please input within (1 - 1000):`
	`New frequency:`
	`584`
	
	`Welcome to the waveform generator program`
	`Switch off the top switch to kill the program`
	`Toggle the other switch to trigger keyboard input`
	`Turn the potentiometer to change the amplitude anytime`
	`Current setting - Waveform: sine, Frequency: 584`
	`_`
	
	Figure 6 - Output for invalid arguments

./ca2 sine 100
 `# ./ca2 sine 100_`
Figure 7 - Correct arguments
	 `ttyp0: sh`
	`Demonstration Routine for PCI-DAS 1602`
	
	`Welcome to the waveform generator program!`
	`Switch off the top switch to kill the program.`
	`Toggle the other switch to trigger keyboard input.`
	`Turn the potentiometer to change the amplitude anytime.`
	`Current settings - Waveform: sine, Frequency: 100`
	`_`
	
	Figure 8 - Output for correct arguments

3.2 Port and Keyboard inputs

The program can receive port digital and analog inputs, and keyboard input. The toggle switches are digital port inputs, potentiometers are analog port inputs. When any of the switches SW2-SW4 is toggled, it will prompt the user to change the setting for 1: waveform type and 2: frequency as shown in the diagram. When the user entered the wrong input, it will re-prompt the user to key in the right waveform or frequency value shown in Figure 9.

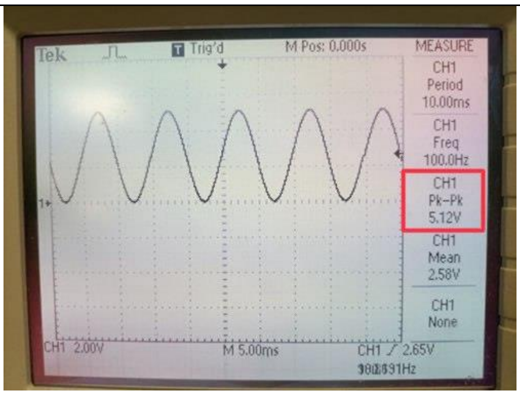
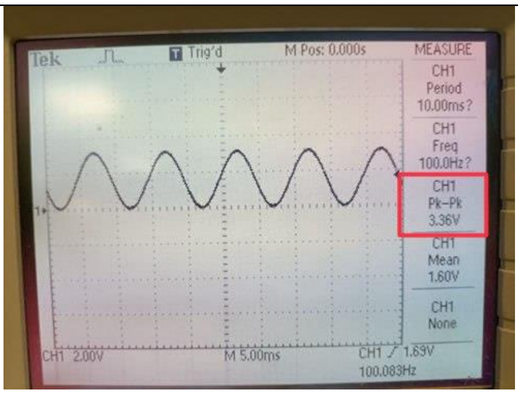


```
Welcome to the waveform generator program
Switch off the top switch to kill the program
Toggle the other switch to trigger keyboard input
Turn the potentiometer to change the amplitude anytime

Keyboard input option
Key in the following number if you want to update the respective field
1: Waveform
2: Frequency
Your input (1 / 2):
1

Please select from these 4 options: sine / square / triangle / sawtooth
New waveform:
sine
Success!
New setting - Waveform: sine, Frequency: 100

Please toggle the switches again to trigger keyboard input

Keyboard input option
Key in the following number if you want to update the respective field
1: Waveform
2: Frequency
Your input (1 / 2):
2
New frequency (1 - 1000):
50
Success!
New setting - Waveform: sine, Frequency: 50

Please toggle the switches again to trigger keyboard input
_
```

Figure 9 - Keyboard input

The amplitude can be changed anytime via the potentiometer and the changes will be reflected on the oscilloscope. The potentiometer ranges from 0x00 to 0xffff, which will be directed to the value of amplitude. The result can be shown in Table 5 Figures 10 and 11 where it shows the same wave with different peak value.

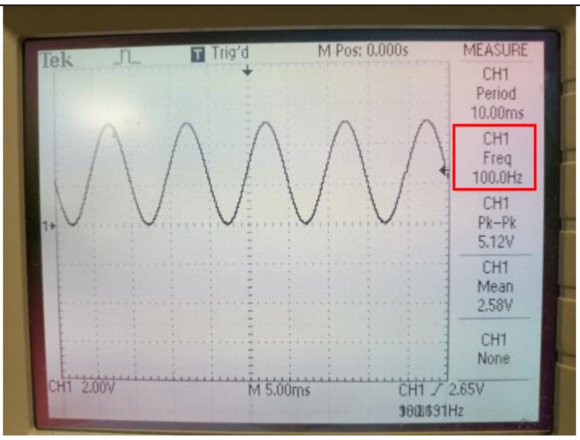
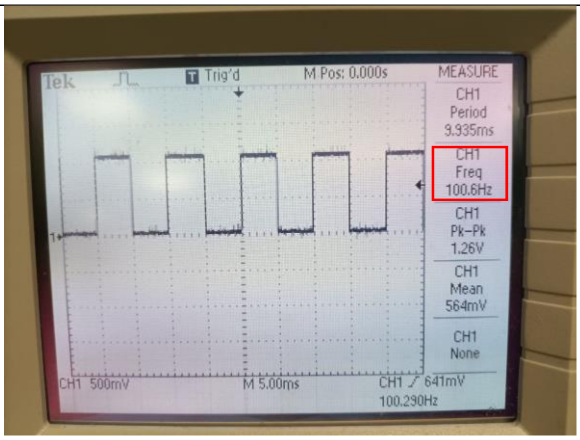
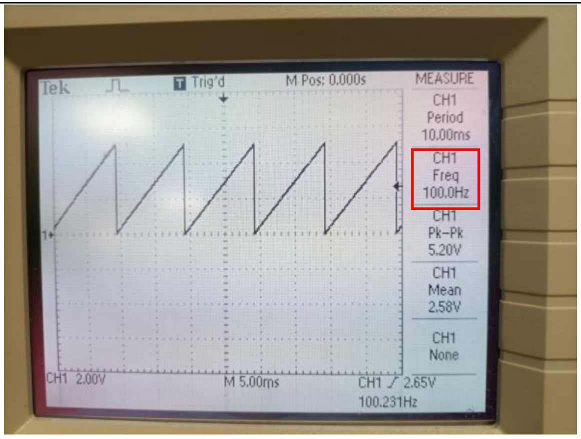
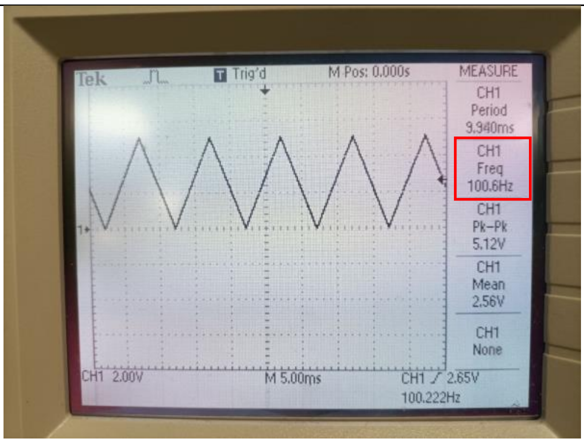
Table 5 - Comparison of different amplitudes

 Tek Trig'd M Pos: 0.000s MEASURE CH1 Period 10.00ms CH1 Freq 100.0Hz CH1 Pk-Pk 5.12V CH1 Mean 2.58V CH1 None CH1 2.00V M 5.00ms CH1 2.65V *Figure 10 - Full amplitude*	 Tek Trig'd M Pos: 0.000s MEASURE CH1 Period 10.00ms? CH1 Freq 100.0Hz? CH1 Pk-Pk 3.36V CH1 Mean 1.60V CH1 None CH1 2.00V M 5.00ms CH1 1.69V 100.083Hz *Figure 11 - Changed amplitude*

3.3 Waveform Outputs

Shown here are all the waveforms running at 100 Hz. Refer to the Appendix A for the mathematical representation of each waveform and Appendix B for figures of all waveforms running at different frequencies.

Table 6 - Waveform outputs

 Tek Trig'd M Pos: 0.000s MEASURE CH1 Period 10.00ms CH1 Freq 100.0Hz CH1 Pk-Pk 5.12V CH1 Mean 2.58V CH1 None CH1 2.00V M 5.00ms CH1 2.65V *Figure 12 - Sine wave*	 Tek Trig'd M Pos: 0.000s MEASURE CH1 Period 9.935ms CH1 Freq 100.6Hz CH1 Pk-Pk 1.26V CH1 Mean 564mV CH1 None CH1 500mV M 5.00ms CH1 641mV 100.290Hz *Figure 13 - Square wave*
 Tek Trig'd M Pos: 0.000s MEASURE CH1 Period 10.00ms CH1 Freq 100.0Hz CH1 Pk-Pk 5.20V CH1 Mean 2.58V CH1 None CH1 2.00V M 5.00ms CH1 2.65V 100.231Hz *Figure 14 - Sawtooth wave*	 Tek Trig'd M Pos: 0.000s MEASURE CH1 Period 9.940ms CH1 Freq 100.6Hz CH1 Pk-Pk 5.12V CH1 Mean 2.56V CH1 None CH1 2.00V M 5.00ms CH1 2.65V 100.222Hz *Figure 15 - Triangle wave*

4. Strength and Limitations

4.1. Strengths

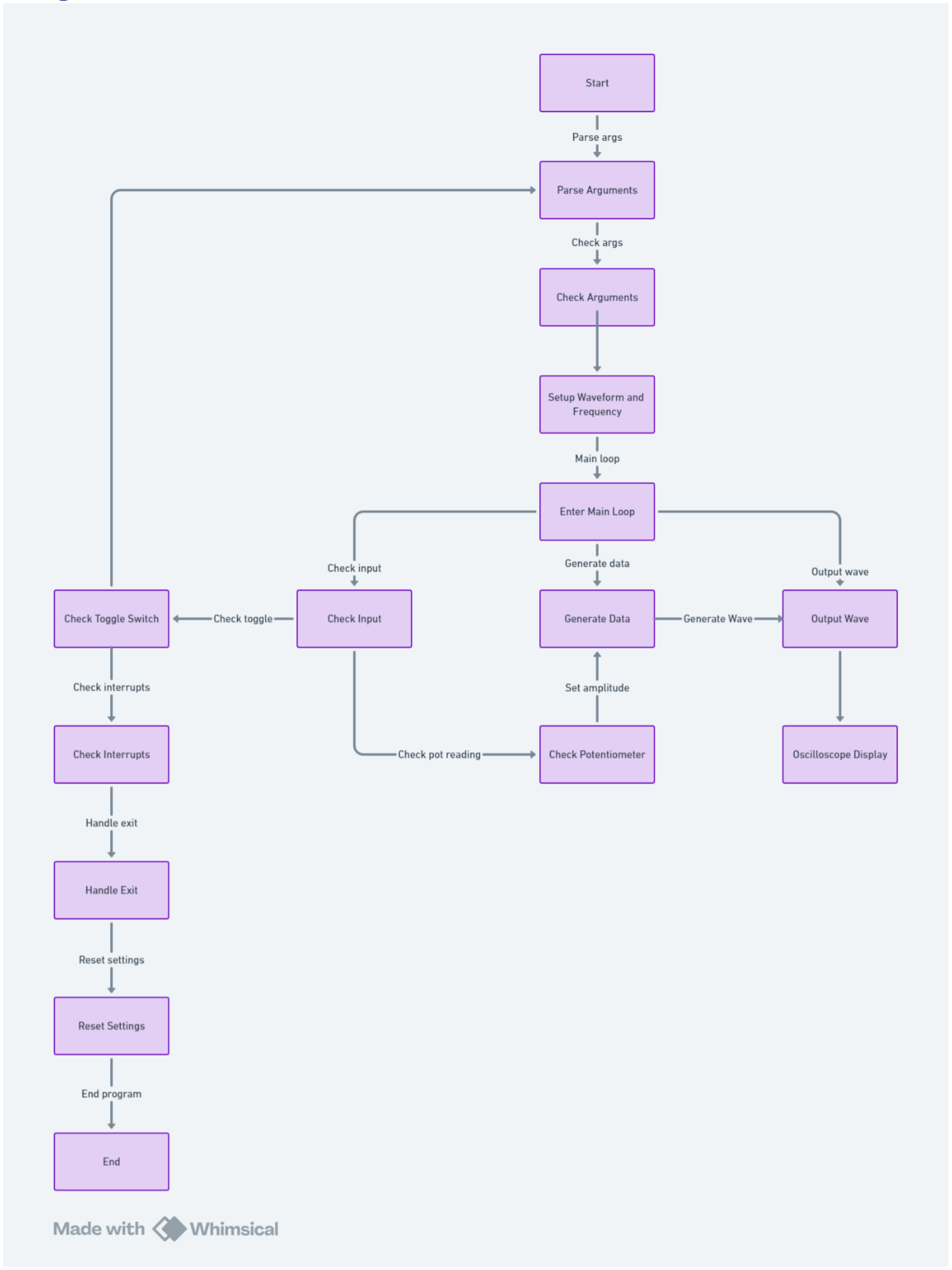
- The output waveform is very smooth and have exact frequency with less than 1% of deviation.
- The real time methods allow user to key in inputs anytime, and a new wave will be generated in real time.
- The program is very stable, it will not crash halfway due to unexpected errors.
- The program handles all sorts of errors such as wrong user input and memory faults. particularly when using scanf() and file IO operations.
- The program handles all kinds of terminating signal. It detects when main switch is off and interrupt signal (Ctrl + C) which provides flexibility for user operation.
- The program allows user to save and retrieve generated waveform data to and from a predefined text file, ensuring repeatability of the desired waveform.
- The program accepts multiple kinds of inputs.
- User Interface (UI) is simple, and the instructions are clearly written to guide user in setting the waveform parameters.
- UI can detect and display errors in user input, and it will continuously prompt the user for input until the correct information is entered.

4.2. Limitations

- When asking for user keyboard input, the user input function scanf() will block the input stream stdin. Hence, the other thread operations will halt despite running on multiple threads.
- Only a potentiometer is used to change a setting which is the amplitude. The other potentiometer could be used to change other settings such as frequency.
- The program requires initialisation with user-input waveform parameters in the argument, as it does not support reading from default settings or previously saved parameters to generate the waveform.
- The users cannot change both waveform and amplitude settings using UI at the same time, they are required to change each setting once at a time.
- More combinations of toggle switches (16 possible combinations) could be used to change settings such as waveform.

5. Appendix

A: Program Flowchart

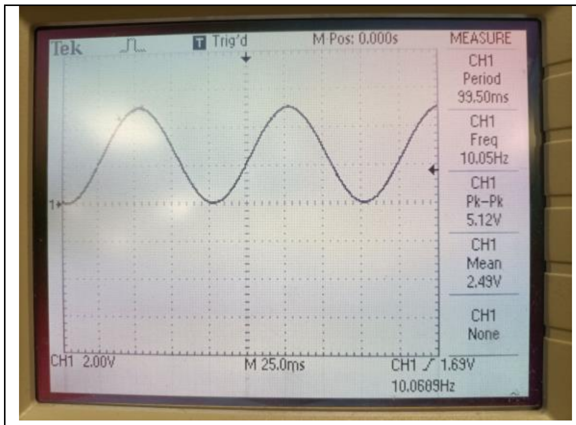
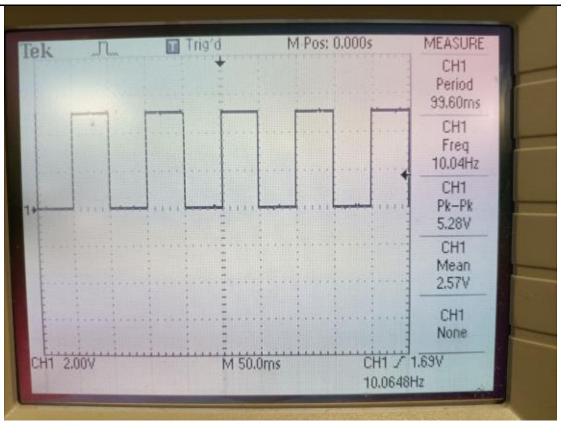
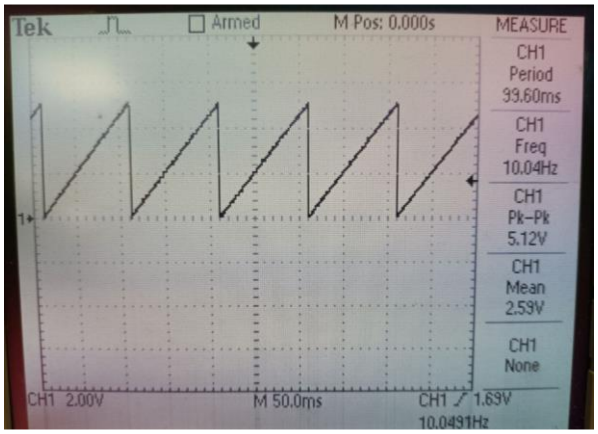
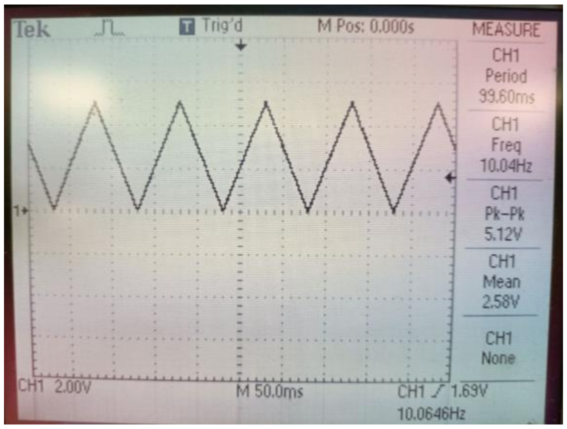


B: Mathematical Representation of Waveforms

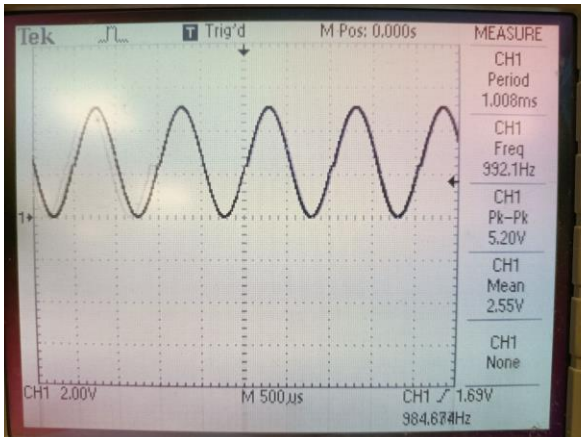
Waveform	**Mathematical Expression**
Sine wave	$\boldsymbol{V = \left[sin(\frac{2\pi}{n} i) + 1\right] \times \frac{amp}{2}}$ Where n is the total number of points (starting at 1), i is the index of the point (starting at 0), amp is the amplitude.
Square wave	$\boldsymbol{V = \begin{cases} 0 \text{ } if \text{ } i < \frac{n}{2} \\ amp \text{ } if \text{ } i \geq \frac{n}{2} \end{cases}}$
Triangle wave	$V = \begin{cases} \frac{\boldsymbol{i}}{\boldsymbol{n/2}} \times amp \;\; \boldsymbol{if\ i < \frac{n}{2}} \\ \frac{\boldsymbol{n-1-i}}{\boldsymbol{n/2}} \times amp \;\; \boldsymbol{if\ i \geq \frac{n}{2}} \end{cases}$
Sawtooth wave	$\boldsymbol{V = \frac{amp}{n} \times i}$

C: All Waves Output

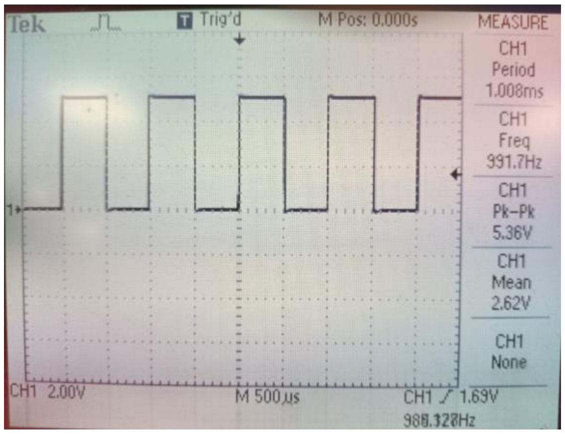
10 Hz

 Tek Trig'd M Pos: 0.000s MEASURE CH1 Period 99.50ms CH1 Freq 10.05Hz CH1 Pk-Pk 5.12V CH1 Mean 2.49V CH1 None CH1 2.00V M 25.0ms CH1 ∕ 1.69V 10.0689Hz **Sine**	 Tek Trig'd M Pos: 0.000s MEASURE CH1 Period 99.60ms CH1 Freq 10.04Hz CH1 Pk-Pk 5.28V CH1 Mean 2.57V CH1 None CH1 2.00V M 50.0ms CH1 ∕ 1.69V 10.0648Hz **Square**
 Tek Armed M Pos: 0.000s MEASURE CH1 Period 99.60ms CH1 Freq 10.04Hz CH1 Pk-Pk 5.12V CH1 Mean 2.59V CH1 None CH1 2.00V M 50.0ms CH1 ∕ 1.69V 10.0491Hz **Sawtooth**	 Tek Trig'd M Pos: 0.000s MEASURE CH1 Period 99.60ms CH1 Freq 10.04Hz CH1 Pk-Pk 5.12V CH1 Mean 2.58V CH1 None CH1 2.00V M 50.0ms CH1 ∕ 1.69V 10.0646Hz **Triangle**

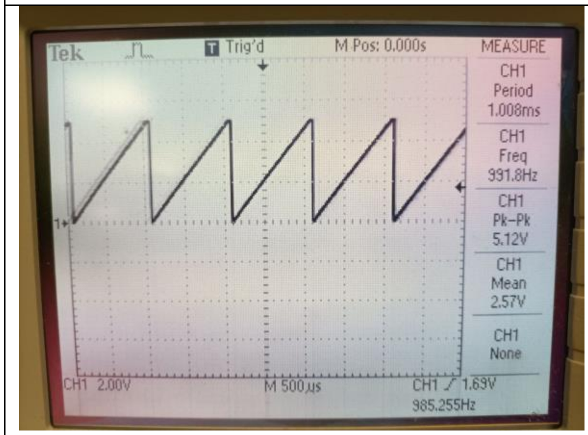
1000 Hz



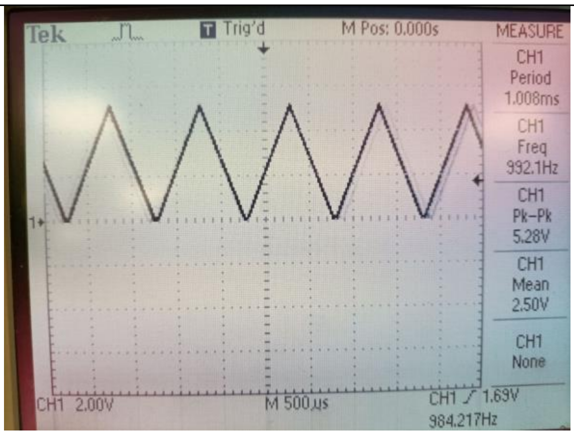
Sine



Square



Sawtooth



Triangle