INSTRUMENTATION SYSTEM DESIGN

PROJECT REPORT

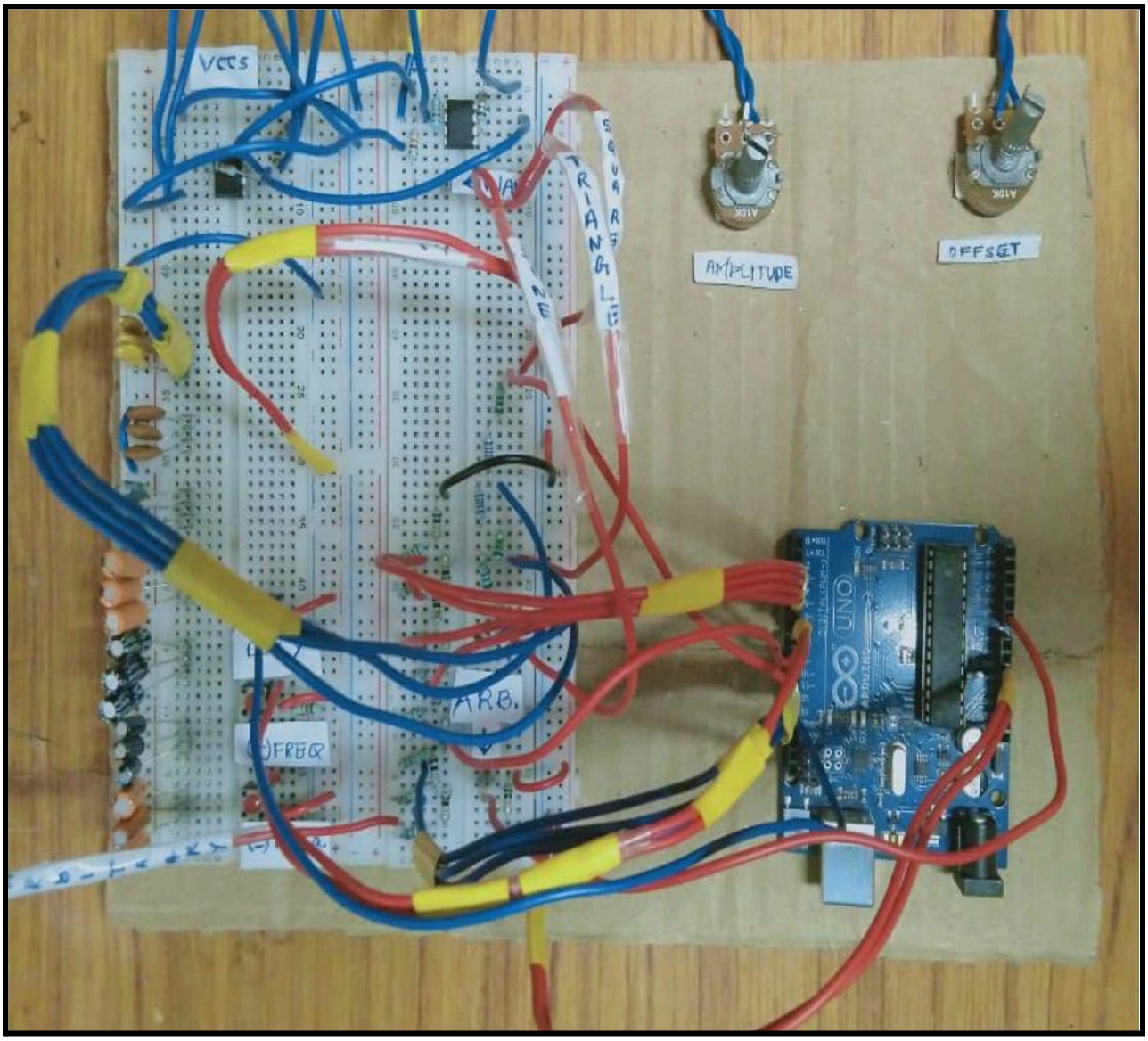
GROUP-16

22IE10052 - Gourav Pramanik

22IE10053 - Chakradhar Reddy Yatham

22IE3EP01 - Ankit Kumar Patel

Standard and Arbitrary Waveform Generator with Voltage Controlled Current Source



Executive Summary

This report documents the design of a **standard and arbitrary waveform generator with a voltage-controlled current source (VCCS)** aimed at providing an affordable and versatile educational tool for undergraduate students. Using **Arduino, RC circuits, an R-2R DAC ladder, and Op-Amps**, we created a design that allows adjustable frequency, amplitude, offset and duty cycle, with high-quality waveform generation.

The device's main features include:

- **Standard Waveform Generation**: Produces square, triangle, and sine waves with adjustable frequency and duty cycle. An inverting amplifier controls amplitude and offset.
- **Arbitrary Waveform Generation**: A switch allows toggling to arbitrary waveform mode, where predefined waveforms can be displayed, and we successfully observed these arbitrary waveforms on an oscilloscope.
- **Voltage-Controlled Current Source (VCCS)**: A separate op-amp configuration allows current output control based on input voltage, operating independently of the waveform generator.

Our prototype demonstrated successful generation of square, triangle, sine, and arbitrary waveforms at 1 kHz, confirmed on an oscilloscope.

Objective

The objective of this project is to design and develop a standard waveform generator (sine, square, triangle) and arbitrary waveform generator with a voltage-controlled current source (VCCS). The system will meet specified amplitude and frequency requirements, providing undergraduate students with a versatile tool for signal generation and analysis.

Motivation

- **Educational Need:** Undergraduate students require practical, firsthand experience with both standard and custom waveforms, as well as with controlled sources, in order to gain a comprehensive understanding of fundamental electrical concepts.
- **Limitations of Existing Devices:** Presently, devices available on the market are limited in their functionality. While some can generate specific waveforms and others can provide controlled sources, there is currently no single device that seamlessly integrates all these capabilities into one comprehensive unit.

- **Cost Barrier for Institutions:** The financial implications of having to purchase multiple devices with distinct functionalities pose a significant barrier for educational institutions. The high costs involved make it challenging for these institutions to provide individual devices for each student.

State of the Art

HTC FG-2002

The HTC FG-2002 is a function generator designed for a range of signal testing and electronic applications. With a frequency range of 0.1 Hz to 2 MHz, it supports multiple waveform outputs like sine, square, triangle, and sawtooth, making it versatile for both analog and digital testing.

Siglent SDG1032X

The Siglent SDG1032X is a versatile dual-channel function and arbitrary waveform generator designed for both professional and educational applications. It operates up to a 30 MHz maximum frequency with high precision. With a sampling rate of 150 MSa/s and 14-bit vertical resolution, it delivers high-fidelity signals that meet the needs of various testing scenarios.

LongWei LW-3010E

The LongWei LW-3010E is a reliable DC power supply designed for laboratory, educational, and industrial use, featuring an adjustable output range of 0-30V and 0-10A. This model is known for its stability and precision, with a 4-digit LED display for accurate voltage and current readouts. Additionally, it supports both 110V and 220V inputs, making it versatile for international use.

Market Gap

- **Lack of a Unified Device**: At present, there does not exist a singular, cost-effective piece of equipment available for purchase that integrates all the essential features required for both the generation of waveforms and functioning as a voltage-controlled current source; these features are crucial for carrying out experimental work at the undergraduate academic level.
- **High Cost:** The current solutions that are available in the market provide a wide array of sophisticated features and functionalities. However, the financial demand associated with acquiring or implementing these solutions is significantly

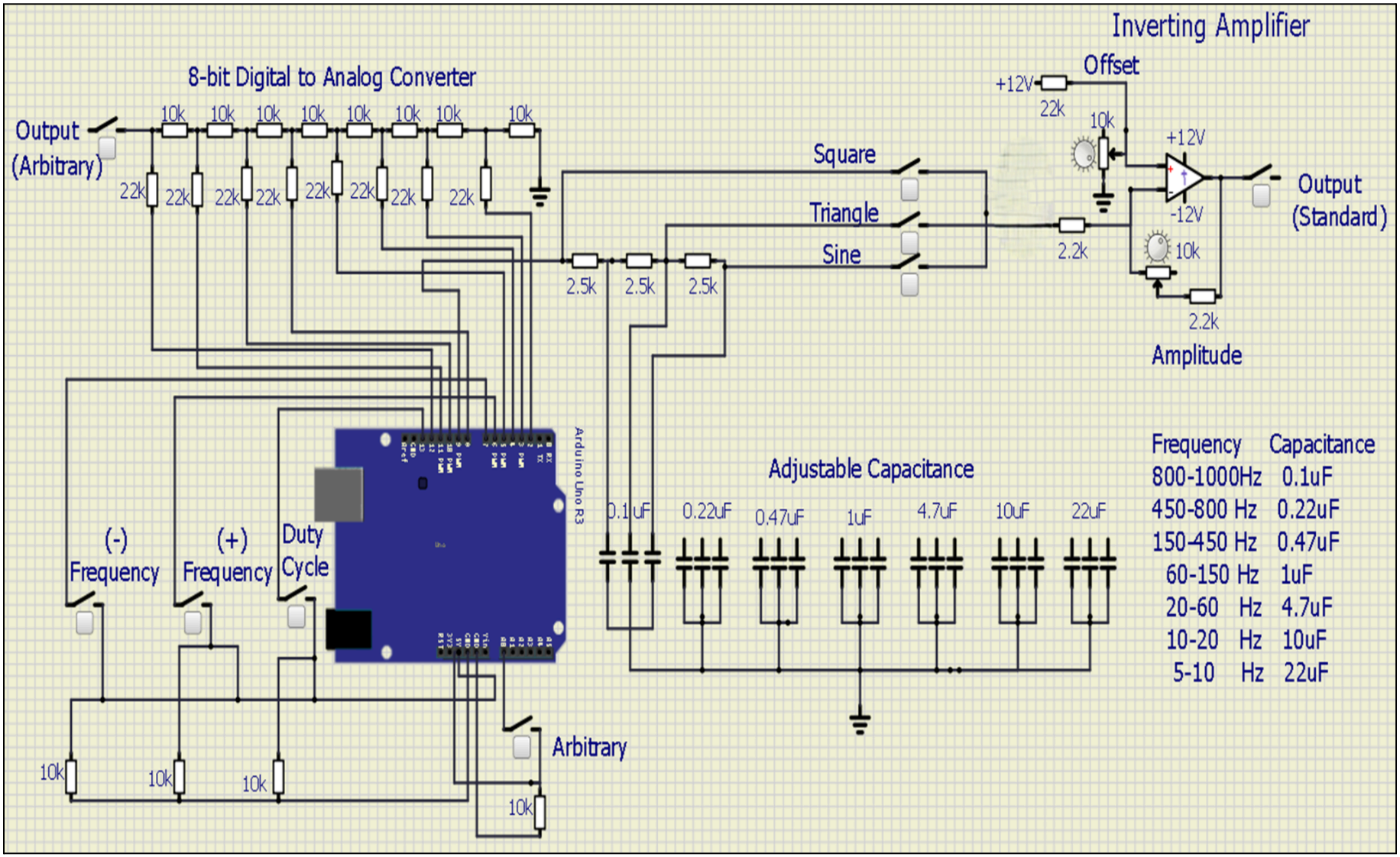
elevated, rendering them unaffordable for the majority of educational institutions operating within India.

- **Feature Overload:** The Siglent SDG1032X, while offering a comprehensive range of high-performance capabilities, is often not fully utilized by undergraduate students who generally do not make use of its advanced features. As a result, purchasing such devices tends to place a substantial financial strain on these students.
- **Need for a Student-Focused Solution:** There exists a considerable demand for a device that is both simplified and affordable, tailored specifically to cater to the unique requirements of undergraduate education. Such a device would ideally enhance accessibility to educational resources and improve overall learning outcomes for students pursuing their undergraduate studies.

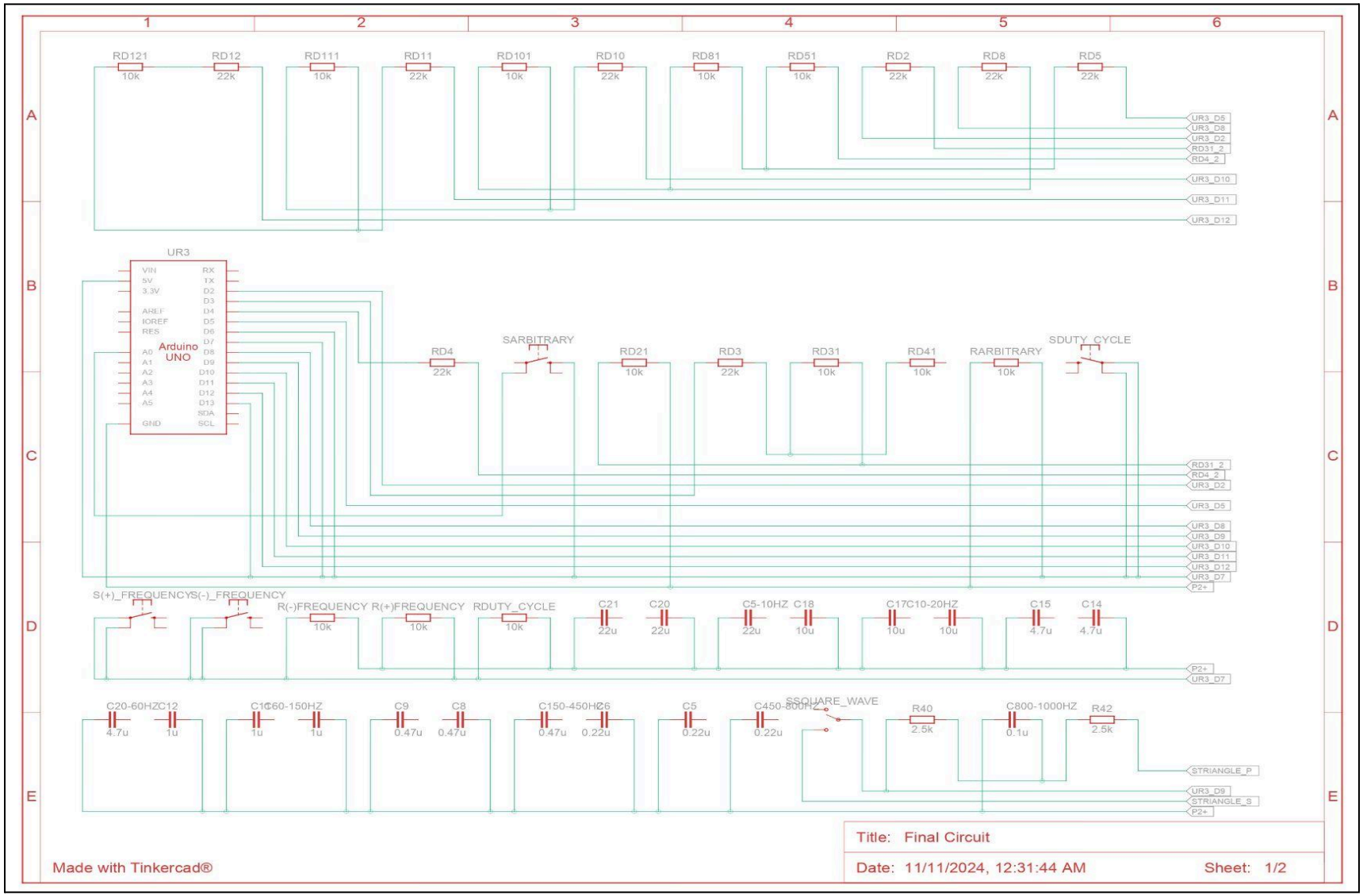
Desired Specifications

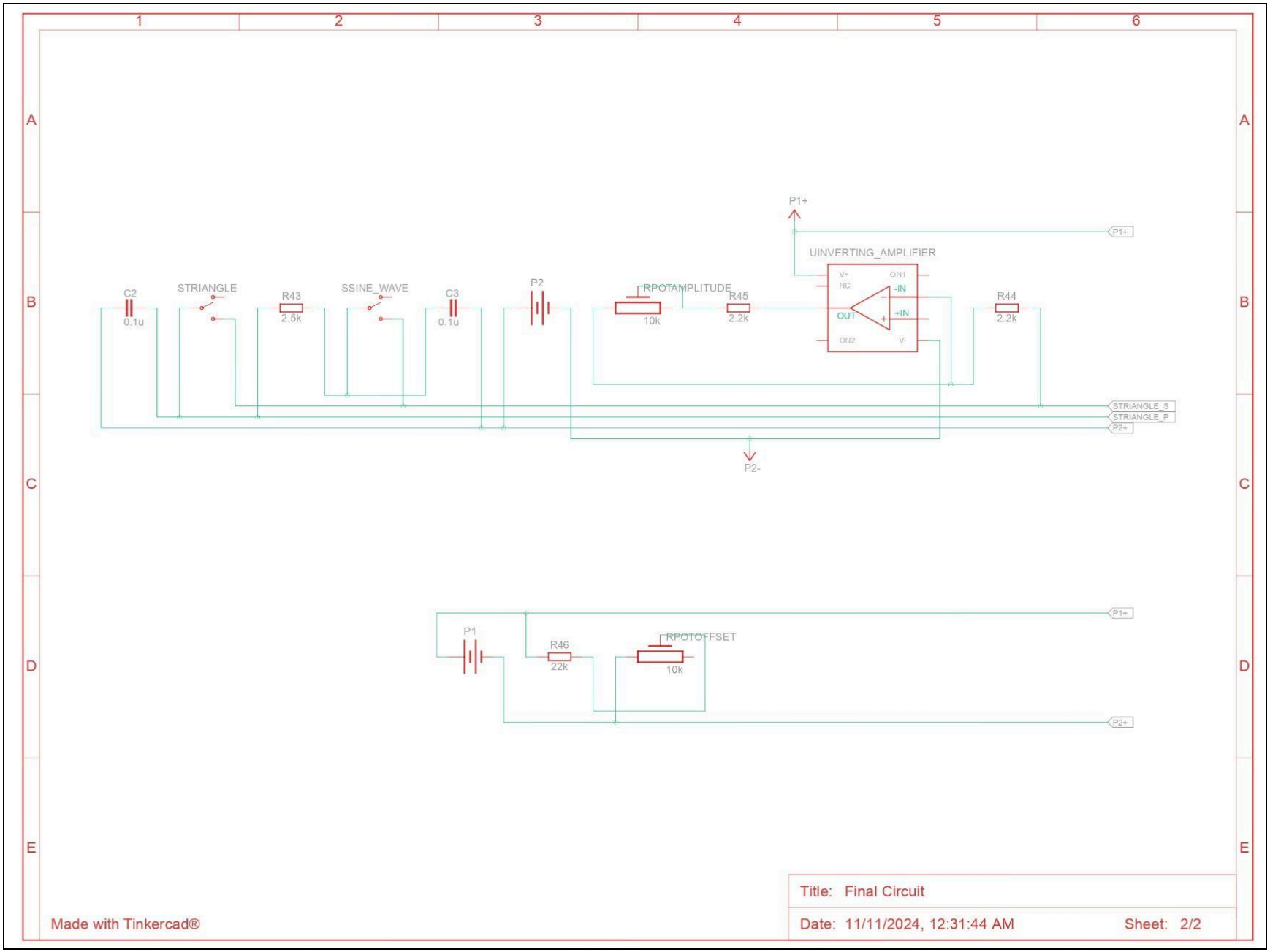
Function/ Arbitrary Waveform Generator	
Waveform Types:	Square, Triangle, Sine, Arbitrary
Frequency Range:	1Hz to 1kHz
Amplitude Range:	1Vpp to 10Vpp
DC Offset:	±5 V
Duty Cycle:	0% to 100%
Range of Operation:	±15 V
Voltage-Controlled-Current Source	
Power supply:	DC 12 V
Input:	0-5V
Output:	0-5mA

Circuit Diagram of Standard and Arbitrary Waveform Generator

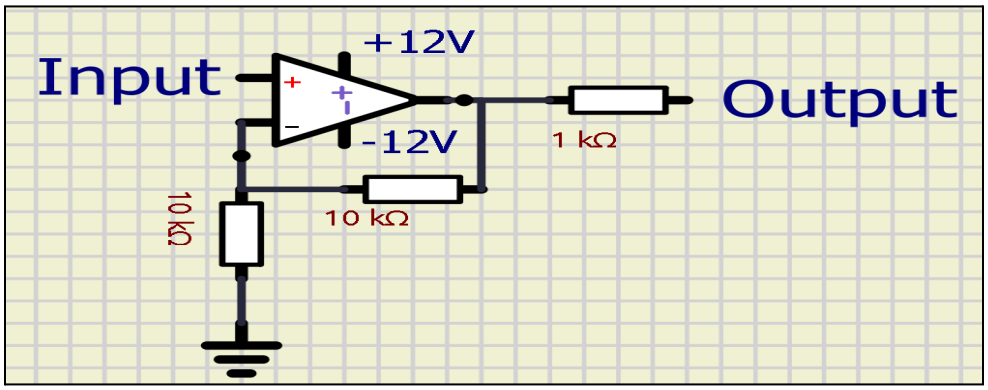


Schematic Diagram of Standard and Arbitrary Waveform Generator




1
2
3
4
5
6
A
B
C
D
E
P1+
UINVERTING_AMPLIFIER
V+
NC
ON1
-IN
+IN
OUT
ON2
V-
C2
0.1u
STRIANGLE
R43
2.5k
SSINE_WAVE
C3
0.1u
P2
RPOTAMPLITUDE
10k
R45
2.2k
R44
2.2k
STRIANGLE_S
STRIANGLE_P
P2+
P2-
P1
R46
22k
RPOTOFFSET
10k
Made with Tinkercad®
Title: Final Circuit
Date: 11/11/2024, 12:31:44 AM
Sheet: 2/2

Circuit Diagram of Voltage-Controlled Current Source



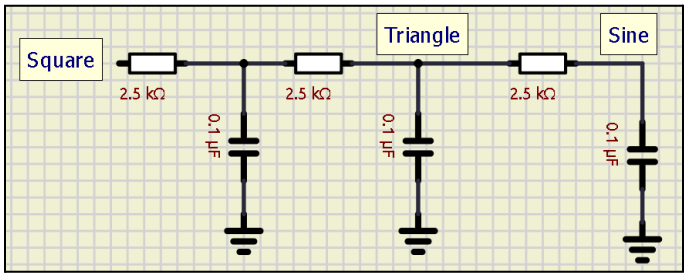
Working Principle

1. Square Wave Generation

The Arduino is used to generate square waves. The frequency and duty cycle of the square wave are controlled using two push buttons:

- The "**(+)Frequency**" button increases the frequency by 10 Hz with each press. Holding the button for a longer duration increases the increment rate.
- The "**(-)Frequency**" button decreases the frequency similarly.
- The duty cycle increases by 5% with each press of the button and resets to 0% when it exceeds 100%.

2. Waveform Transformation: Square to Triangle to Sine



- Square-to-triangle-to-sine conversion uses **RC** filters to smooth waveforms. A square wave is passed through an **RC** integrator, attenuating higher harmonics to produce a triangle wave. Further, a **3rd-order RC low-pass filter** removes remaining harmonics, creating a near-pure sine wave.
- Fourier series of square, triangle and sine wave with amplitude **A** and angular frequency ω_o:

$$x_{\text{square}}(t) = \frac{4A}{\pi} \sum_{n=1,3,5,\ldots}^{\infty} \frac{1}{n} \sin(n\omega_0 t)$$

$$x_{\text{triangle}}(t) = \frac{8A}{\pi^2} \sum_{n=1,3,5,\ldots}^{\infty} \frac{(-1)^{(n-1)/2}}{n^2} \sin(n\omega_0 t)$$

$$x_{\text{sine}}(t) = A \sin(\omega_0 t)$$

- Magnitude Response of the **n-th harmonic** after **N** stages of **RC** filter:

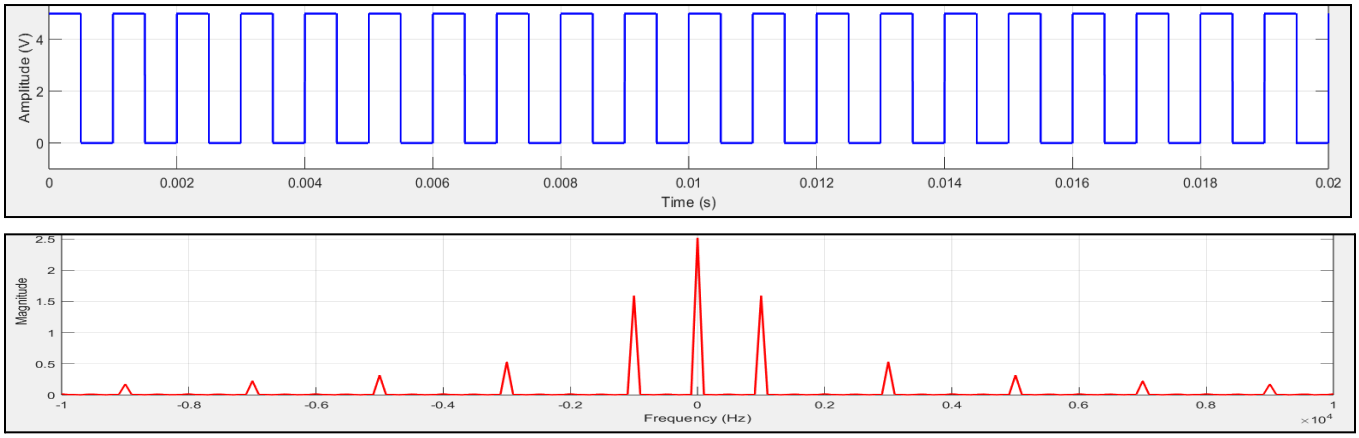
$$A_n = \frac{\frac{4A}{\pi n}}{\left(\sqrt{1 + (n\omega_0 RC)^2}\right)^N}$$

- For higher harmonics the amplitude is inversely proportional to n^3 after the 2nd Stage, resembling a triangle wave. Similarly, for higher harmonics the amplitude is inversely proportional to n^4 after the 3rd Stage, resembling a sine wave.
- At higher angular frequencies of square wave, the attenuation increases because the magnitude is inversely proportional to $\omega_o RC$. To achieve the proper shape and amplitude for triangle and sine waveforms, the **RC** value must be adjusted accordingly. For higher frequencies, decrease the **RC** value. For lower frequencies, increase the **RC** value.
- It is more convenient to vary the capacitance since all the capacitors share a common ground terminal. Therefore, capacitance is adjusted to accommodate different frequency ranges.
- Cut-off frequency of **RC** filter:

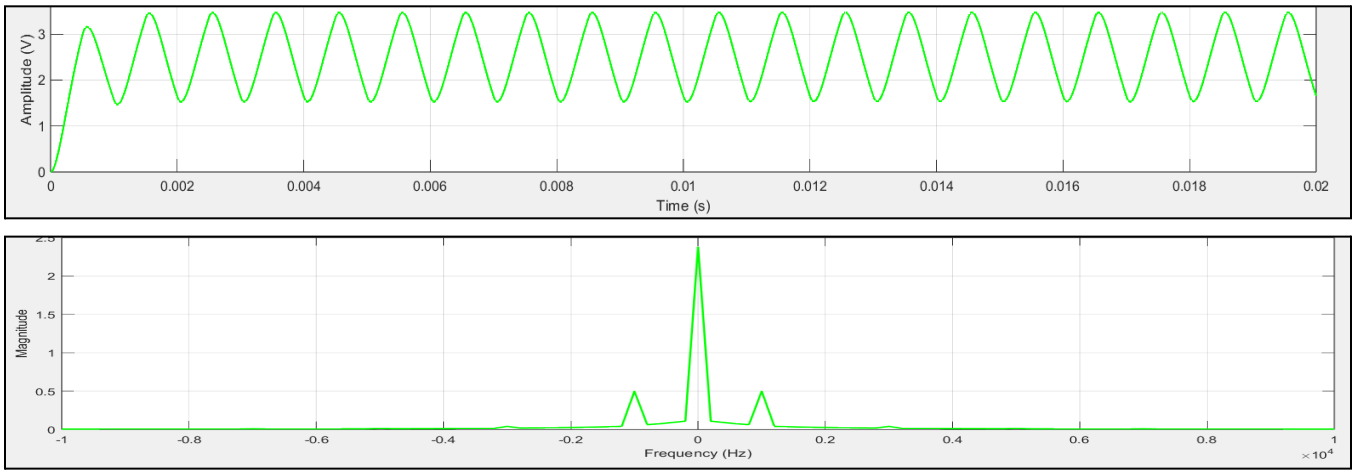
$$f_c = \frac{1}{2\pi RC}$$

- Roll-off rate for **2nd Order RC filter** beyond the cut-off frequency: **-40** $\frac{dB}{decade}$
- Roll-off rate for **3rd Order RC filter** beyond the cut-off frequency: **-60** $\frac{dB}{decade}$

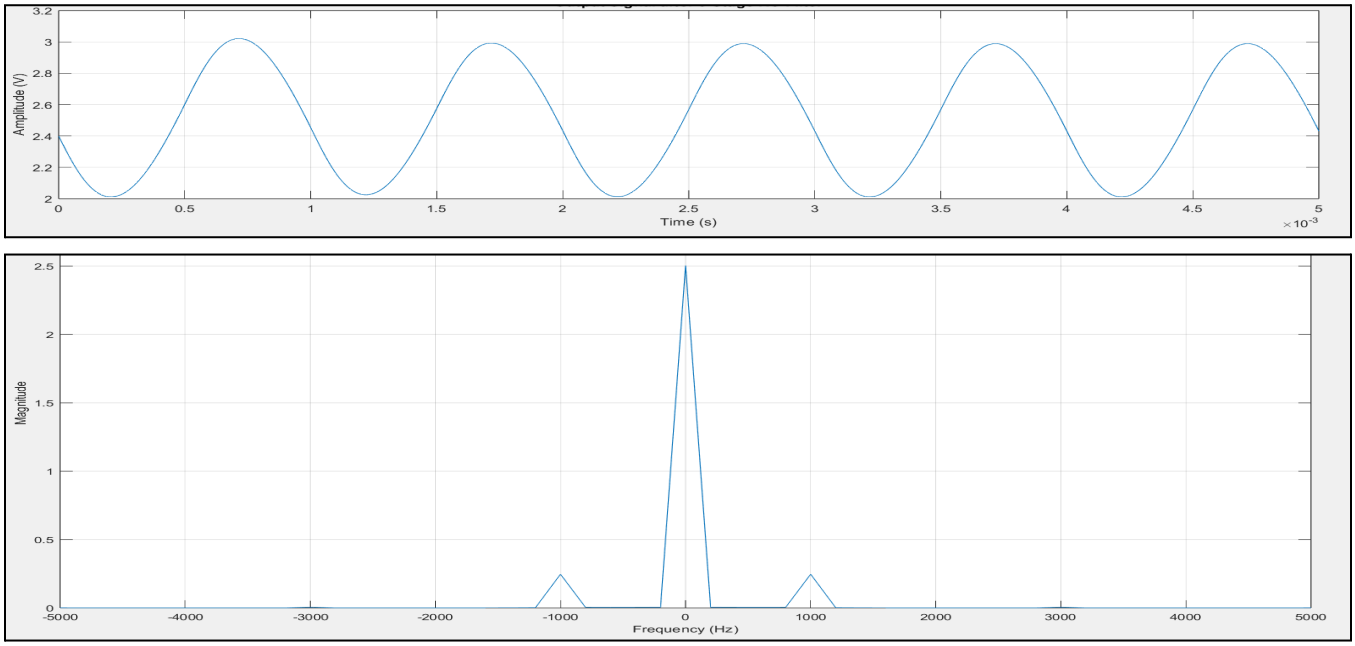
Input signal to the 3rd-order RC filter and its Fourier Transform:



Output signal after 2nd RC stage and its Fourier Transform



Output signal after 3rd RC stage and its Fourier Transform

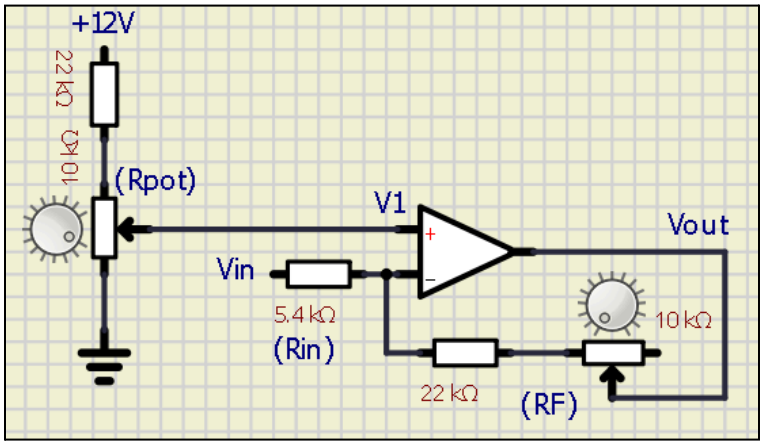


3. Amplitude and Offset Control

An inverting amplifier circuit is used to control the amplitude and offset of the square, triangle, and sine waves. The design involves:

- The gain of the inverting amplifier is varied using a potentiometer between the inverting and output terminals.
- A resistor in series with this potentiometer prevents loading effects when the potentiometer resistance is set very low.

- A potentiometer connected between a **+12V DC** supply and ground adjusts the voltage at the non-inverting terminal of the op-amp. Hence, controlling the offset.
- A resistor in series with the potentiometer minimizes the voltage drop across the potentiometer.



$$V_{out} = -\frac{R_F}{R_{in}}V_{in} + V_{offset} \quad where\ V_{offset} = V_1 \times (\frac{R_F}{R_{in}} + 1)$$

$$V_1 = \frac{R_{pot}}{32k\Omega} \times 12V$$

$$For\ V_{in} = 1V_{pp}$$

$$Minimum\ V_{out} = \frac{22\,k\Omega}{5.4\,k\Omega} \times 1V_{pp} \approx 4.074\,V_{pp}$$

$$Maximum\ V_{out} = \frac{22\,k\Omega + 10\,k\Omega}{5.4\,k\Omega} \times 1V_{pp} \approx 5.926\,V_{pp}$$

$$Minimum\ V_{offset} = (\frac{0\,k\Omega}{32\,k\Omega} \times 12V) \times (\frac{22\,k\Omega}{5.4\,k\Omega} + 1) = 0V$$

$$Maximum\ V_{offset} = (\frac{10\,k\Omega}{32\,k\Omega} \times 12V) \times (\frac{32\,k\Omega}{5.4\,k\Omega} + 1) = 25.973V$$

$$Maximum\ V_{offset} = 12V\ (Since\ OpAmp's\ output\ voltage\ is\ limited\ by\ its\ power\ supply\ rails)$$

4. Arbitrary Waveform Generation

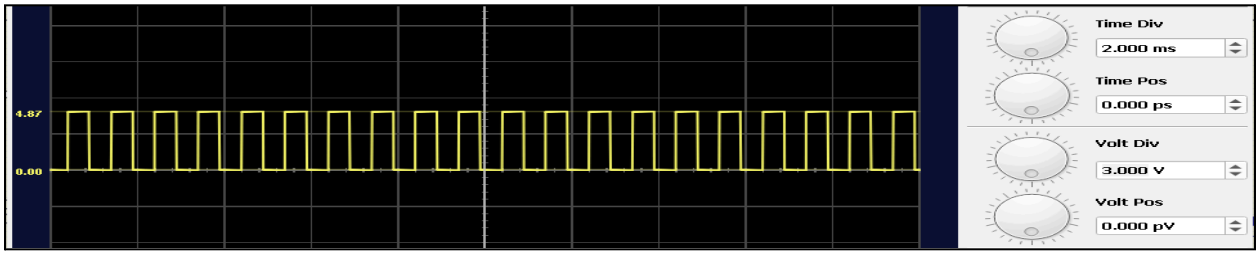
- The Arduino generates arbitrary waveforms by using an array to store user-defined values.
- An **8-bit R-2R ladder DAC** is connected to 8 digital pins of the Arduino, converting digital signals to analog signals. These signals are displayed as waveforms on an oscilloscope.

5. Voltage-Controlled Current Source

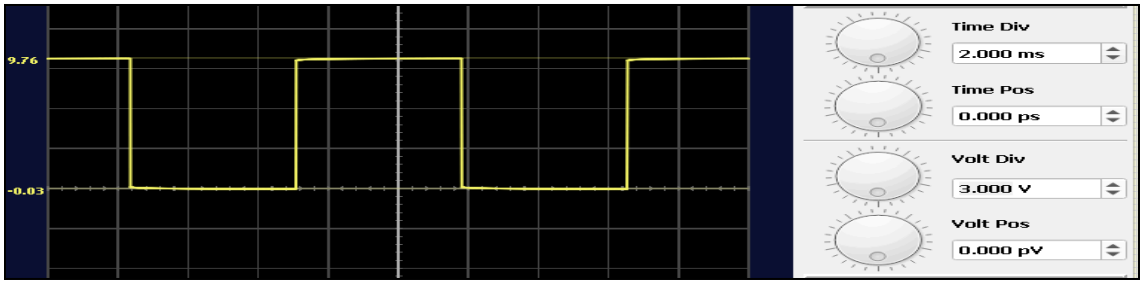
The **VCCS** is designed using an op-amp. The op-amp maintains a zero potential difference between its inverting and non-inverting terminals, and the current output is determined by: $\mathbf{I_{out}} = \frac{V_{in}}{0.5\ k\Omega}$. The output current varies in response to the input voltage.

Simulations

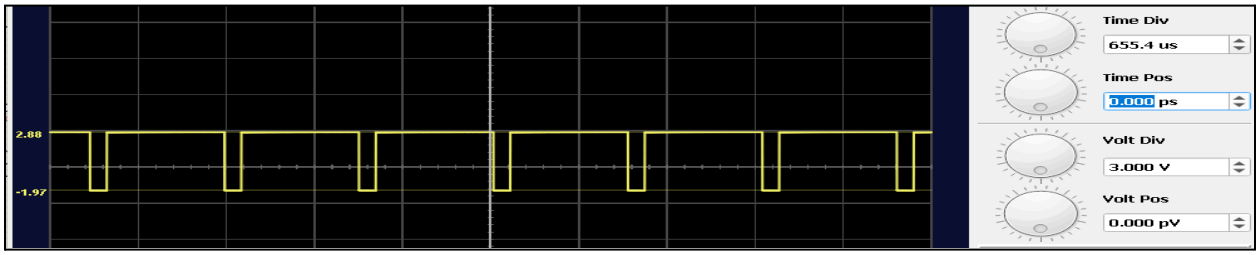
Square Wave



Frequency: 1 kHz, Amplitude: 4.87V_{pp}, Offset: 0V, Duty Cycle: 50%

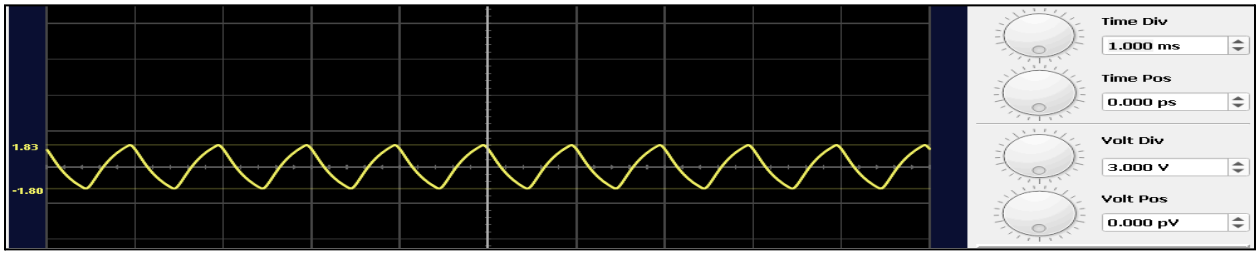


Frequency: 105.8 Hz, Amplitude: 9.79V_{pp}, Offset: -0.03V, Duty Cycle: 50%

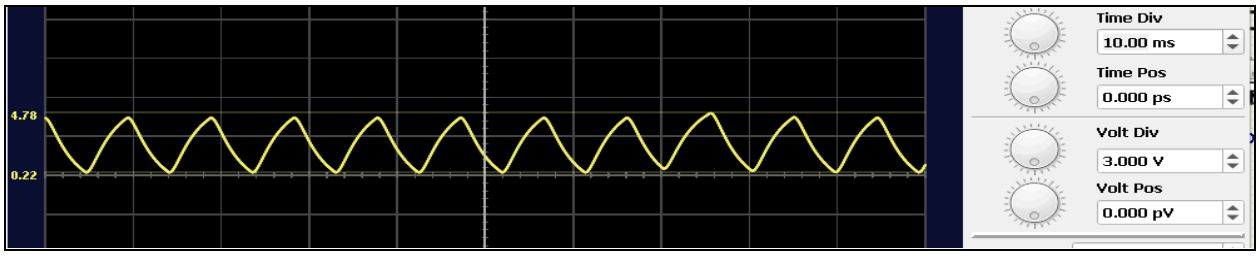


Frequency: 1 kHz, Amplitude: 4.85V_{pp}, Offset: -1.97V, Duty Cycle: 85%

Triangle Wave

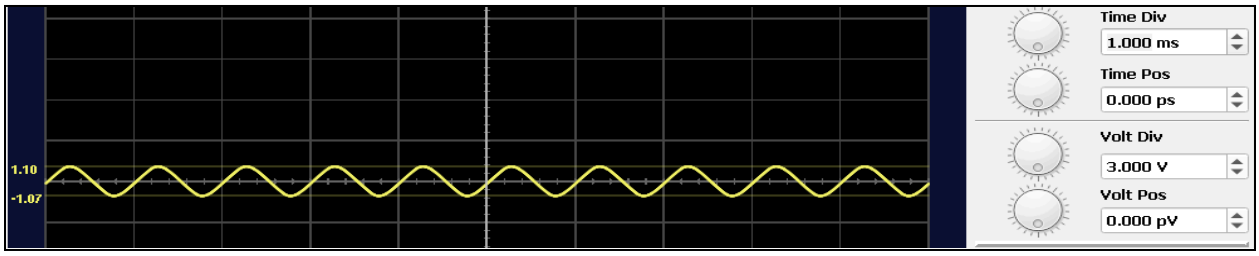


Frequency: 1 kHz, Amplitude: 3.63V_{pp}, Offset: 0.02V

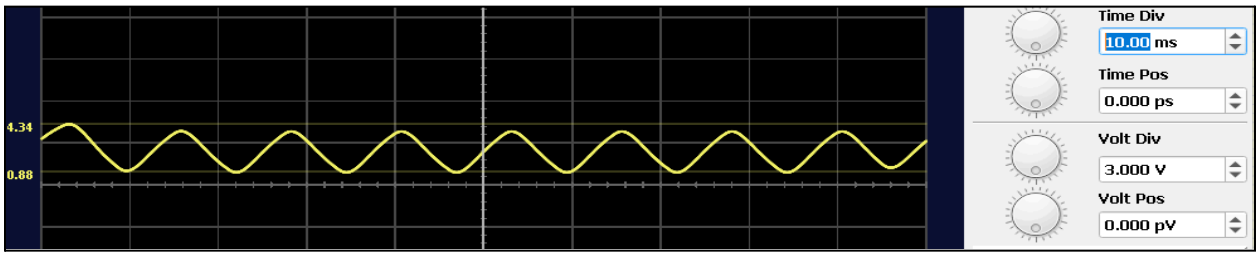


Frequency: 105.9 Hz, Amplitude: 4.56V_{pp}, Offset: 2.06V

Sine Wave



Frequency: 1 kHz, Amplitude: 2.27V_{pp}, Offset: 0.06V

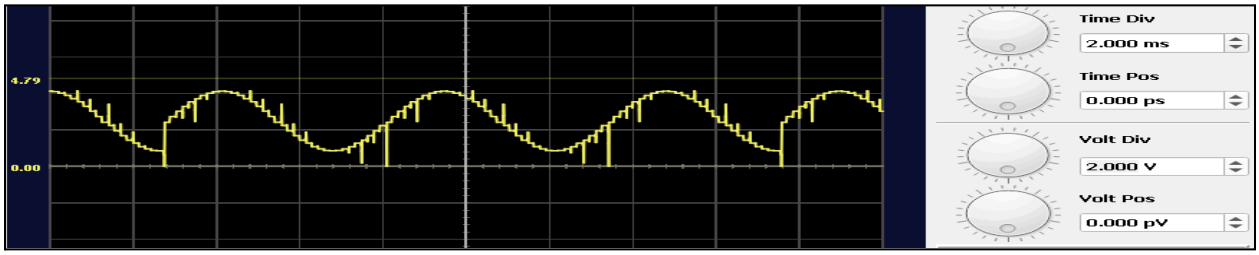


Frequency: 80.32 Hz, Amplitude: 3.46V_{pp}, Offset: 2.61V

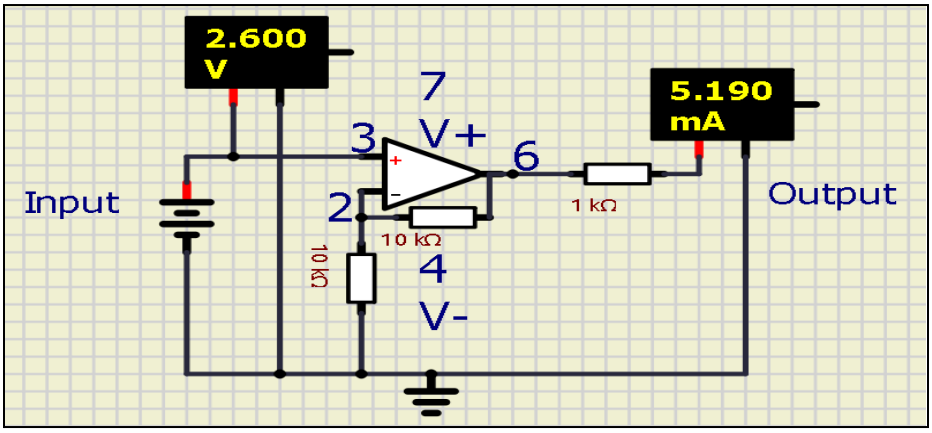
Arbitrary Wave Input

128, 143, 157, 171, 183, 194, 203, 209, 213, 214, 213, 209, 203, 194, 183, 171, 157, 143, 128, 113, 99, 85, 73, 62, 53, 47, 43, 42, 43, 47, 53, 62, 73, 85, 99, 113, 128, 143, 157, 171, 183, 194, 203, 209, 213, 214, 213, 209, 203, 194, 183, 171, 157, 143, 128, 113, 99, 85, 73, 62, 53, 47, 43, 42, 43, 47, 53, 62, 73, 85, 99, 113, 128, 143, 157, 171, 183, 194, 203, 209, 213, 214, 213, 209, 203, 194, 183, 171, 157, 143, 128, 113, 99, 85, 73, 62, 53, 47, 43, 42

Arbitrary Wave Output



Voltage-Controlled Current Source



Components Selection

1. Arduino UNO R3

Acts as the central controller of the circuit. It generates PWM signals for waveforms and provides control over the frequency, duty cycle, and waveform type. It generates arbitrary waveforms based on user input.

2. Resistors (various values)

Resistors in this circuit serve multiple purposes, such as:

- Limiting current to prevent damage to components.
- Setting gain and control parameters in the op-amp circuit.
- Pull-up or pull-down configurations for buttons and switches.

3. Capacitors (0.1 µF to 20 µF)

- **Purpose**: Capacitors form RC (resistor-capacitor) circuits that filter the PWM output of the Arduino to produce more stable analog waveforms, especially for

sine and triangle waves. The capacitance values range from 0.1 μF to 20 μF to allow a wide range of frequency outputs, as shown in the table in the circuit.

4. Switches and Push Buttons

Allow the user to manually select waveform type (sine, square, triangle, arbitrary) and control frequency and duty cycle.

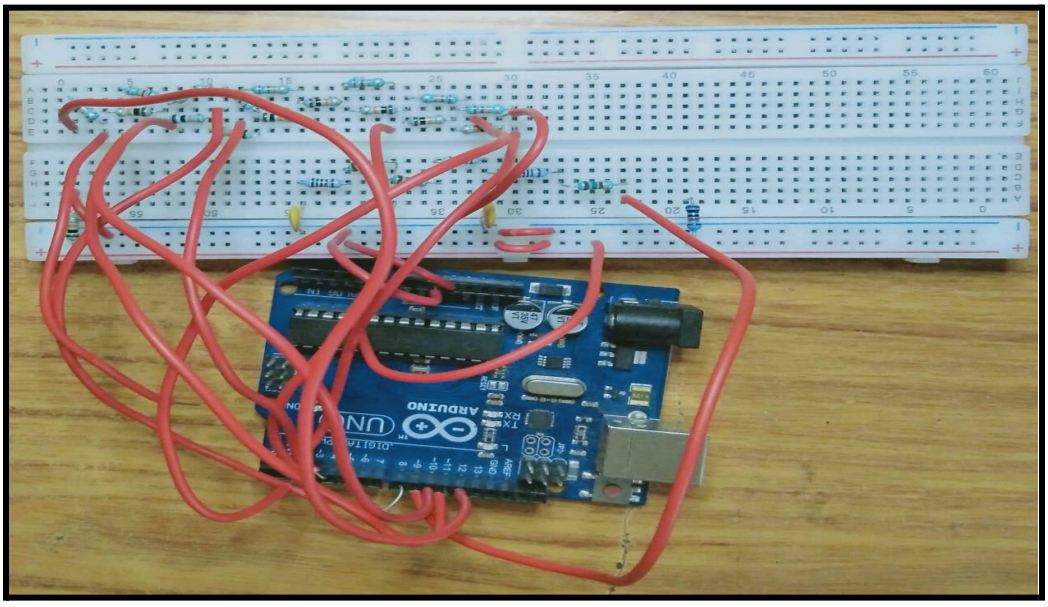
5. Potentiometers (various values)

Used for adjusting parameters like amplitude and offset in real-time.

Amplitude Potentiometer: Controls the voltage gain in the op-amp, affecting the amplitude of the output waveform.

Offset Potentiometer: Connected to the 12V power supply, it adjusts the DC offset of the waveform, allowing you to move the waveform up or down relative to ground.

Prototype of Standard and Arbitrary Waveform Generator



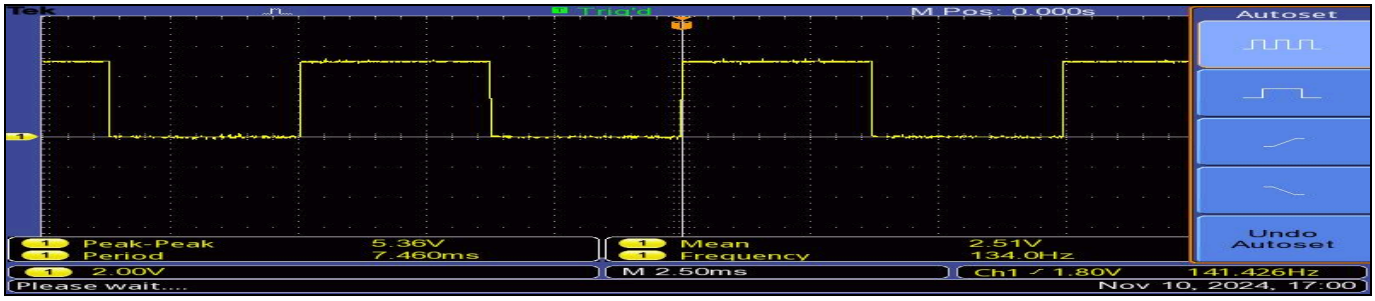
Final Implementation

Due to time constraints and approaching end-of-semester exams, the entire circuit was implemented on a breadboard rather than a finalized PCB. The breadboard and Arduino were secured onto cardboard for added stability, and all components and connections were labeled directly on the breadboard and wires for clarity.

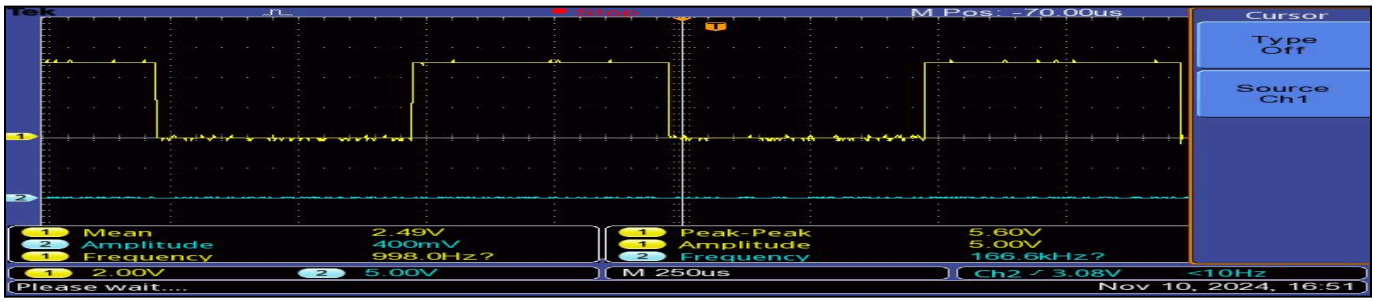
1. **Code Finalization and Documentation:** The final code was streamlined for efficiency, minimizing delays when switching waveforms and adjusting frequency or duty cycle.
2. **Final Testing on Oscilloscope:** Limited testing to 1 kHz outputs for square, sine, triangle, and arbitrary waveforms to verify essential functionality.
3. **Packaging and Deployment:** Mounted the breadboard and Arduino on cardboard for support and labeled each component for easy identification.

Results

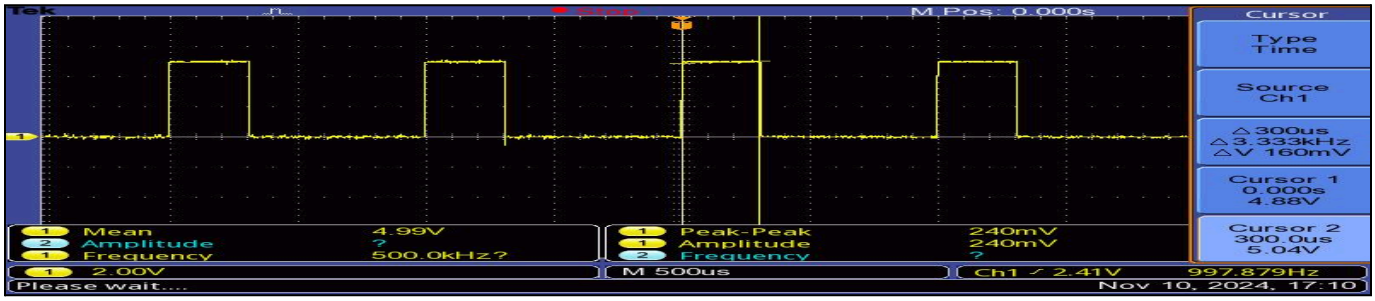
Square Wave



Amplitude = 5.36 V_{pp}, Frequency = 141.426 Hz, Duty Cycle = 50%

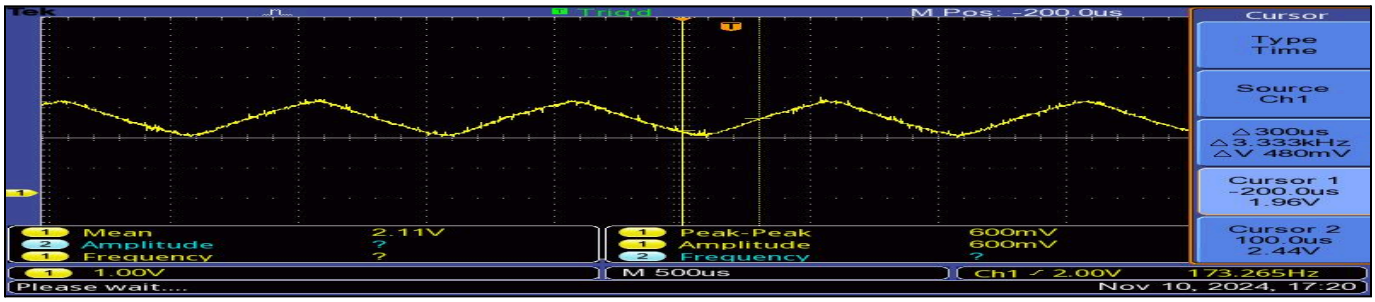


Amplitude = 5.60V_{pp}, Frequency = 998.0 Hz, Duty Cycle = 50%

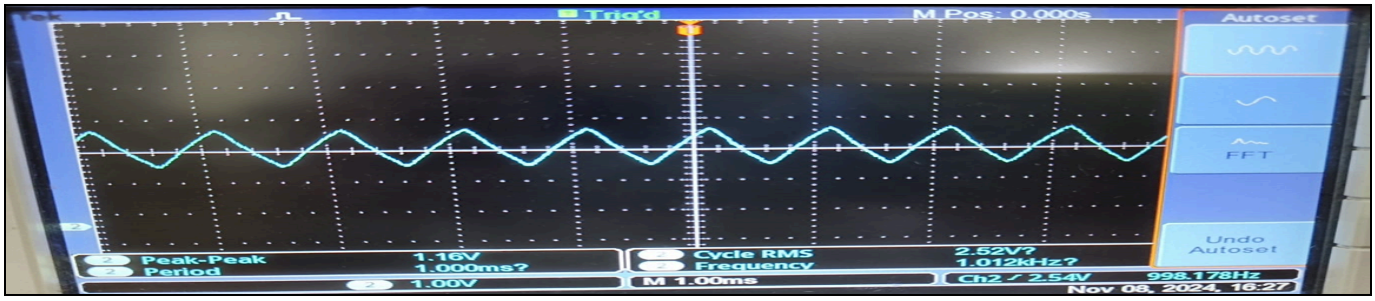


Amplitude = 5.04V_{pp}, Frequency = 997.879 Hz, Duty Cycle = 30%

Triangle Wave

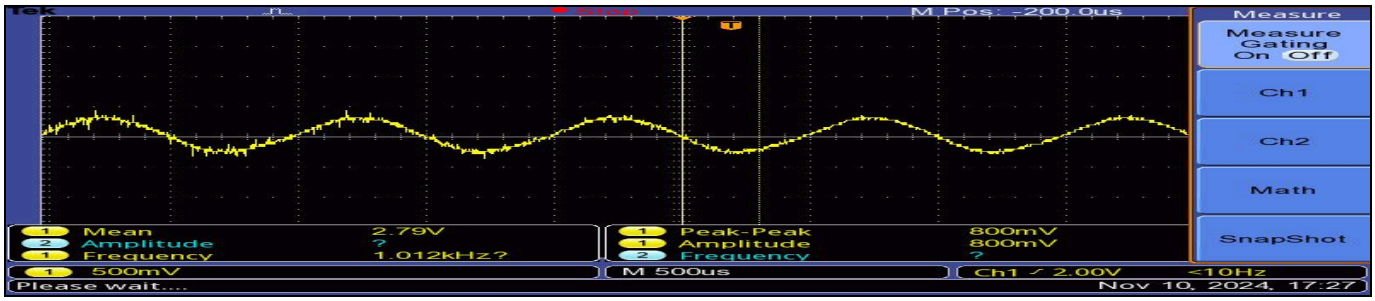


Amplitude = $0.6V_{pp}$, Frequency = 173.265 Hz



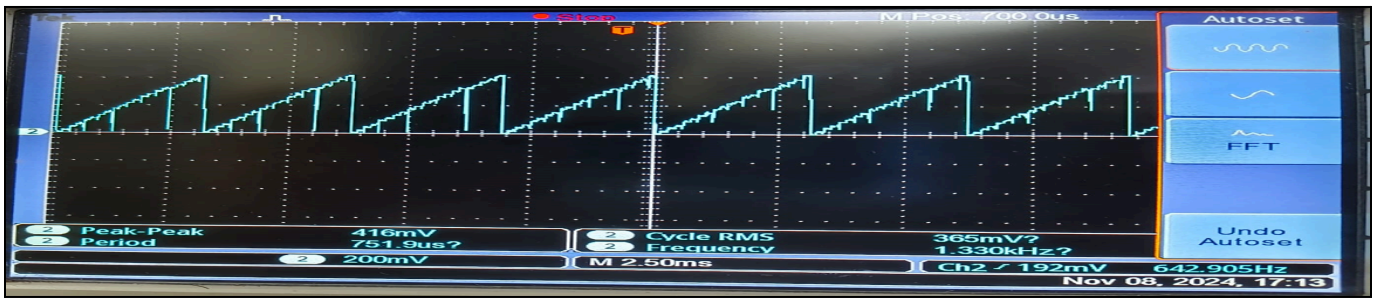
Amplitude = $1.16V_{pp}$, Frequency = 1.012 kHz

Sine Wave



Amplitude = $0.8V_{pp}$, Frequency = 1.012 kHz

Arbitrary Wave



Amplitude = 416mV$_{pp}$, Frequency = 1.330 kHz

Amplitude = 2.24V$_{pp}$, Frequency = 354.5 Hz

Output Specifications

Waveform	Frequency	Amplitude	Duty Cycle	DC Offset
Square Wave	5Hz - 1kHz	5V$_{pp}$ - 10V$_{pp}$	0 -100%	±5V
Triangle Wave	10Hz - 1kHz	2V$_{pp}$ - 8V$_{pp}$	-	±5V
Sine Wave	10Hz - 1kHz	1V$_{pp}$ - 4V$_{pp}$	-	±5V
Arbitrary Wave	5Hz - 1kHz	1V$_{pp}$ - 5V$_{pp}$	-	-

Challenges and Component Limitations

- **Discontinuity and Lag**: When generating continuous waveforms, we observed a lag after each cycle, likely due to the processing delays in the Arduino.
- **RC Filter Spikes**: Using an RC filter to convert the Arduino's square wave into sine and triangle waves introduced unexpected spikes.
- **Frequency Control Issues:** Initially, a potentiometer was used to control the frequency, but small adjustments caused large, uneven frequency changes. This was replaced with push buttons: one to increment and the other to decrement the frequency.
- **Limited Input Options:** Without dedicated input devices like a keypad or touchscreen, a laptop was required to take input values for arbitrary waveforms, limiting the design's standalone functionality and user-friendliness.
- **Limitations of Using an 8-bit R-2R Ladder:** The 8-bit R-2R ladder accepts only 256 discrete input values, causing a stepped output waveform. A 12-bit R-2R ladder could provide 4096 values for smoother transitions, but the limited number of available Arduino pins prevented its implementation.

Bill of Materials

S.No	Component	Specification	Qty.	Unit Cost (Rs.)	Amount (Rs.)
1.	Operational Amplifier	LM741	2	15.00	30.00
2.	Arduino UNO R3		1	380.00	380.00
3.	Resistor	1 kΩ	4	1.00	4.00
		1.5 kΩ	3	1.00	3.00
		2.2 kΩ	2	1.00	2.00
		10 kΩ	14	1.00	14.00
		22 kΩ	9	1.00	9.00
4.	Potentiometer	10 kΩ	2	10.00	20.00
5.	Capacitor	0.1 µF	3	4.00	12.00
		0.22 µF	3	4.00	12.00
		0.47 µF	3	4.00	12.00
		1 µF	3	4.00	12.00
		4.7 µF	3	4.00	12.00
		10 µF	3	4.00	12.00
		22 µF	3	4.00	12.00
6.	Breadboard		2	60.00	120.00
7.	Push Button		3	5.00	15.00
8.	9V Battery with Connector		4	35.00	140.00
9.	Wires				50.00
10.	Miscellaneous (Cardboard, Tape, etc)				30.00
		Total	67		901.00

Arduino Code

```
#include <TimerOne.h>
unsigned long t = 1000, f, k = 512;
byte k1, kn, kn1, kn2;
int drive, drive0;
const int dacPins[8] = {2, 3, 4, 5, 8, 10, 11, 12};
const int waveform[] = { 128, 131, 134, 137, 140, 143, 146, 149, 1
158, 161, 164, 167, 170, 173, 176, 179, 182, 185,
188, 191, 194, 197, 200, 203, 206, 209, 212, 215,
218, 221, 224, 227, 230, 233, 236, 239, 242, 245,
248, 251, 254, 255, 254, 251, 248, 245, 242, 239,
236, 233, 230, 227, 224, 221, 218, 215, 212, 209,
206, 203, 200, 197, 194, 191, 188, 185, 182, 179,
176, 173, 170, 167, 164, 161, 158, 155, 152, 149,
146, 143, 140, 137, 134, 131, 128, 125, 122, 119,
116, 113, 110, 107, 104, 101, 98, 95, 92, 89,
86, 83, 80, 77, 74, 71, 68, 65, 62, 59,
56, 53, 50, 47, 44, 41, 38, 35, 32, 29,
26, 23, 20, 17, 14, 11, 8, 5, 2, 1,
0, 1, 2, 5, 8, 11, 14, 17, 20, 23,
26, 29, 32, 35, 38, 41, 44, 47, 50, 53,
56, 59, 62, 65, 68, 71, 74, 77, 80, 83,
86, 89, 92, 95, 98, 101, 104, 107, 110, 113,
116, 119, 122, 125
};
const int waveformLength = sizeof(waveform) / sizeof(waveform[0]);
void setup()
{
  for (int i = 0; i < 8; i++) {
    pinMode(dacPins[i], OUTPUT);
  }
  pinMode(A0, INPUT);
  pinMode(9, OUTPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  pinMode(13, INPUT);
}
void loop()
{
  if (digitalRead(A0) == HIGH) {
    for (int i = 0; i < waveformLength; i++) {
      outputToDAC(waveform[i]);
      delayMicroseconds(100);
  }
}
else {
  Timer1.initialize(t);
  Timer1.pwm(9, k);
  kn = digitalRead(6);
  kn1 = digitalRead(7);
  kn2 = digitalRead(13);
  if (kn == HIGH) {
    drive++;
    if (drive < 10) {
      t -= 50;
    }
    else if (drive >= 10 && drive < 30 ) {
      t -= 100;
    }
    else if (drive >= 30 && drive < 60) {
      t -= 500;
    }
    else {
      t -= 1000;
    }
  } else {
    drive = 0;
  }
  if (kn1 == HIGH) {
    drive0++;
    if (drive0 < 10) {
      t += 50;
    }
    else if (drive0 >= 10 && drive0 < 30 ) {
      t += 100;
    }
    else if (drive0 >= 30 && drive0 < 60) {
      t += 500;
    }
    else {
      t += 1000;
    }
  } else {
    drive0 = 0;
  }
```

```
    if (t == 0 || t > 300000) {
      t = 1;
    }
    if (t > 200000 && t < 300000) {
      t = 200000;
    }

    f = 1000000 / t;
    k1 = k * 100 / 1024;
    if (kn2 == HIGH) {
      if (k >= 32) {
        k -= 32;
      } else {
        k = 1024;
      }
    }
    delay(100);
  }
}
void outputToDAC(int value) {
  for (int i = 0; i < 8; i++) {
        digitalWrite(dacPins[i], (value >> i) & 1);
  }
}
```

1. Global Variables and Setup

- `t`, `f`, and `k` are timing and fill factor variables for controlling the square wave's frequency and duty cycle.
- `dacPins[]` defines the 8 pins used for outputting an 8-bit value to the DAC.
- `waveform[]` is an array containing values to generate a smooth waveform (e.g., sine wave).
- In `setup()`, the DAC pins are set as outputs, and other control pins are configured for various input and output tasks.

2. Loop Function

The main `loop()` function alternates between two modes based on the state of pin `A0`:

- **Arbitrary Waveform Mode** (when `A0` is HIGH):
    - The loop iterates through the `waveform[]` array, outputting each value to the DAC pins via `outputToDAC()`.
    - `delayMicroseconds(100);` controls the speed (frequency) of the waveform by adding a delay after each output value.

- **Square Wave Mode** (when `A0` is LOW):
    - `Timer1.initialize(t);` sets the period of the square wave.
    - `Timer1.pwm(9, k);` generates the square wave with the specified duty cycle (fill factor `k`).
    - **Frequency Adjustment**:
        - Buttons connected to pins `6` and `7` allow for incrementing or decrementing the pulse period `t`, which directly changes the frequency.
        - Depending on how long the button is held, the adjustments occur in steps (small steps for short presses, larger steps for long presses).
    - **Fill Factor Adjustment**:
        - A button on pin `13` reduces the fill factor `k` in steps, wrapping around to 100% if it reaches 0%.
    - The `delay(100);` at the end of the `loop()` function adds a short delay to prevent rapid changes.

3. `outputToDAC()` Function

- This function outputs an 8-bit value from `waveform[]` to the DAC pins by writing each bit to the corresponding pin in `dacPins[]`. It shifts each bit of the `value` to the right and writes it to the respective pin using `digitalWrite()`.

User Manual

1. Setting Up Hardware Connections

- Connect the Arduino to your laptop or an external power supply to turn it on.
- Connect the wire labeled **sine**, **square**, or **triangle** to the positive terminal of the oscilloscope. These wires will output the respective waveforms.
- Connect the negative oscilloscope terminal to the **ground** wire.
- After powering up, observe the waveform on the oscilloscope.

2. Switching to Arbitrary Waveform Mode

- To switch the Arduino to arbitrary waveform generation mode, connect the wire labeled **arbitrary** to the point indicated on the breadboard.
- This will instruct the Arduino to switch to generating custom waveforms.

- Connect the positive probe of the oscilloscope to the wire labeled **arbitrary** on the breadboard. This will allow the oscilloscope to display the custom waveform.

3. Setting Up the Arduino IDE

- If you haven't already, download and install the Arduino IDE software on your laptop from the official Arduino website.
- Connect the Arduino board to your laptop via a USB cable. The Arduino IDE will automatically recognize the device.

4. Programming the Arduino

- Open the Arduino IDE and write or load a code that includes an array called **waveform**. This array will store the custom waveform values as integers (no floating-point values). For example: `int waveform[] = {0, 512, 1023, 512};` Ensure that the waveform array is populated with the desired integer values representing the custom waveform you want to generate. The values typically range from 0 to 1023, corresponding to the PWM output range on the Arduino.

Project Timeline

Phase	Completion Date
Project Proposal	August 13, 2024
Initial Design & Research	August 30, 2024
Mid-Term Presentation	October 1, 2024
Prototyping	October 18, 2024
Testing & Calibrations	November 1, 2024
Final Implementation	November 8, 2024
Final Presentation and Demonstration	November 16, 2024

Acknowledgements

Prof. Siddhartha Mukhopadhyay guided this project with invaluable insights, while Teaching Assistants **Sativada Manikanth**, **Shivansh Tekam** and **KV Koteswara Rao** provided technical support and feedback. We extend our deepest gratitude to **Prof. Siddhartha Mukhopadhyay**, Teaching Assistants **Sativada Manikanth**, **Shivansh Tekam** and **KV Koteswara Rao** for their guidance and support throughout this project.