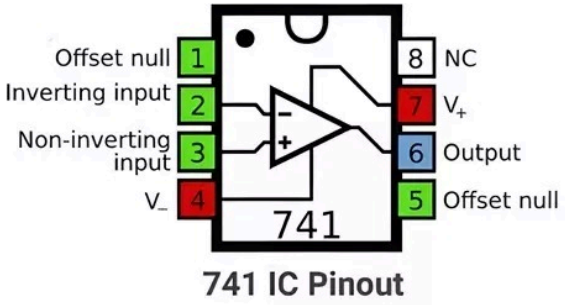
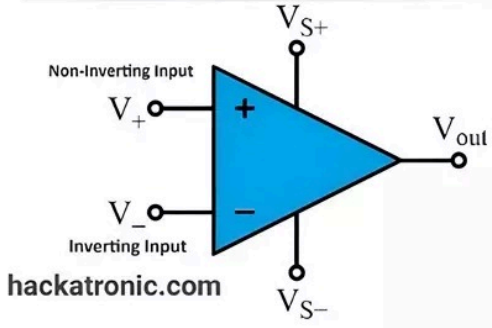
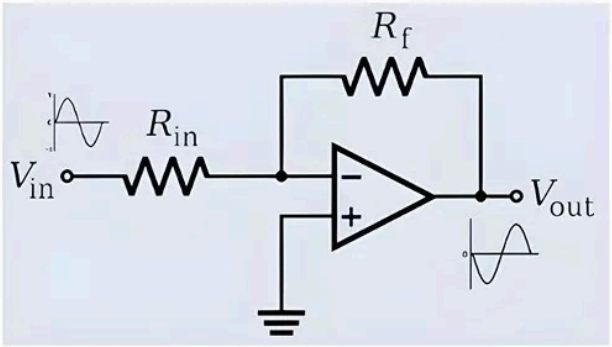
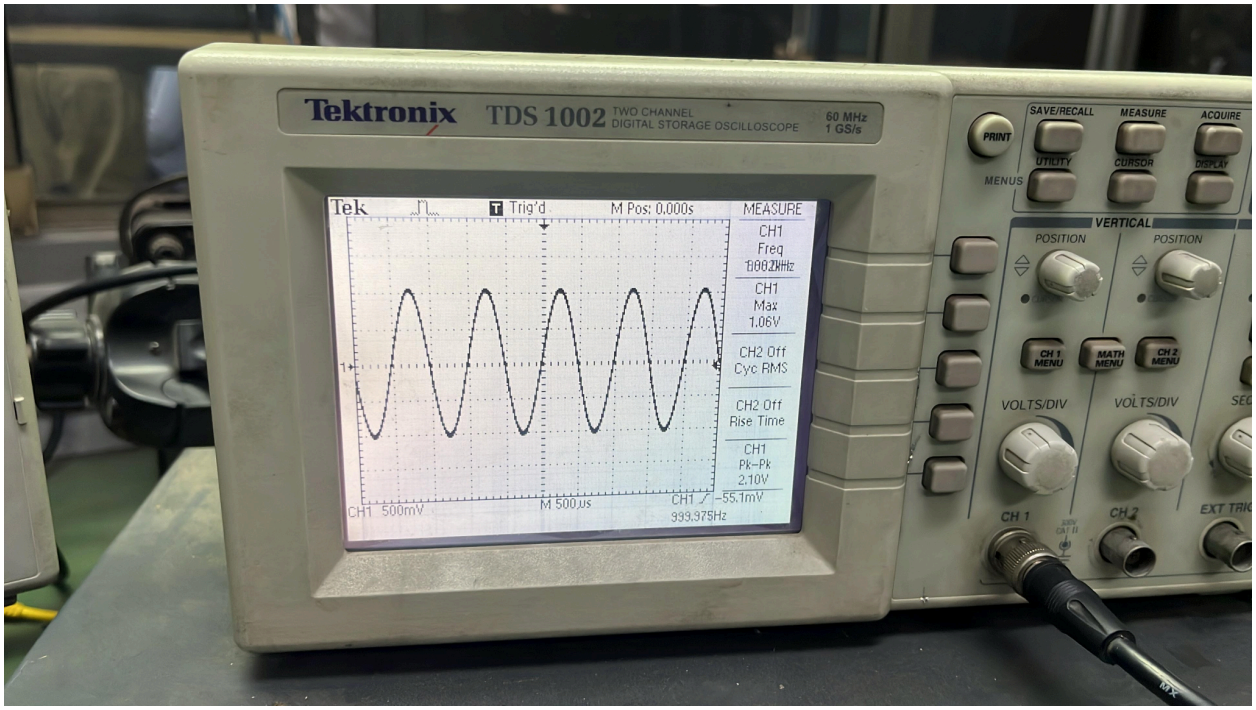
Task-1: Amplifier characteristics, to observe if it will be able to provide sinusoid till which frequency range.
Assigned to: Sushant and Gandharva
On careful analysis of ELP101 experiments, it was found that no experiment exceeds frequency of 400kHz, therefore we need an OpAmp which works in the range of 100Hz-400kHz
Configuring the IC LM741CN
Datasheet exploration:
LM741CN
slew rate - 0.5 V/μs.
3db bandwidth (gain bandwidth product) - 1MHz
link - https://www.ti.com/lit/ds/symlink/lm741.pdf

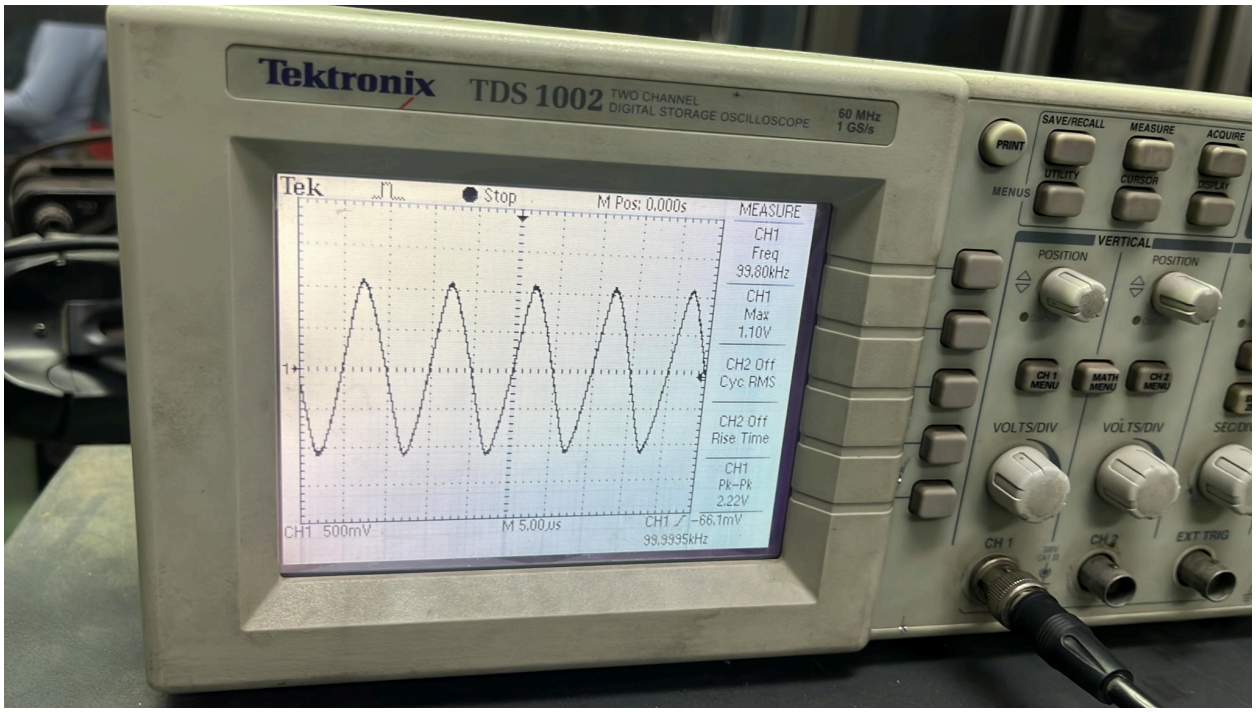
LM358
slew rate - 0.3 V/μs.
3db bandwidth (gain bandwidth product) - 0.7MHz
link - https://www.ti.com/lit/ds/symlink/lm358.pdf?ts=1738724243245&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLM358

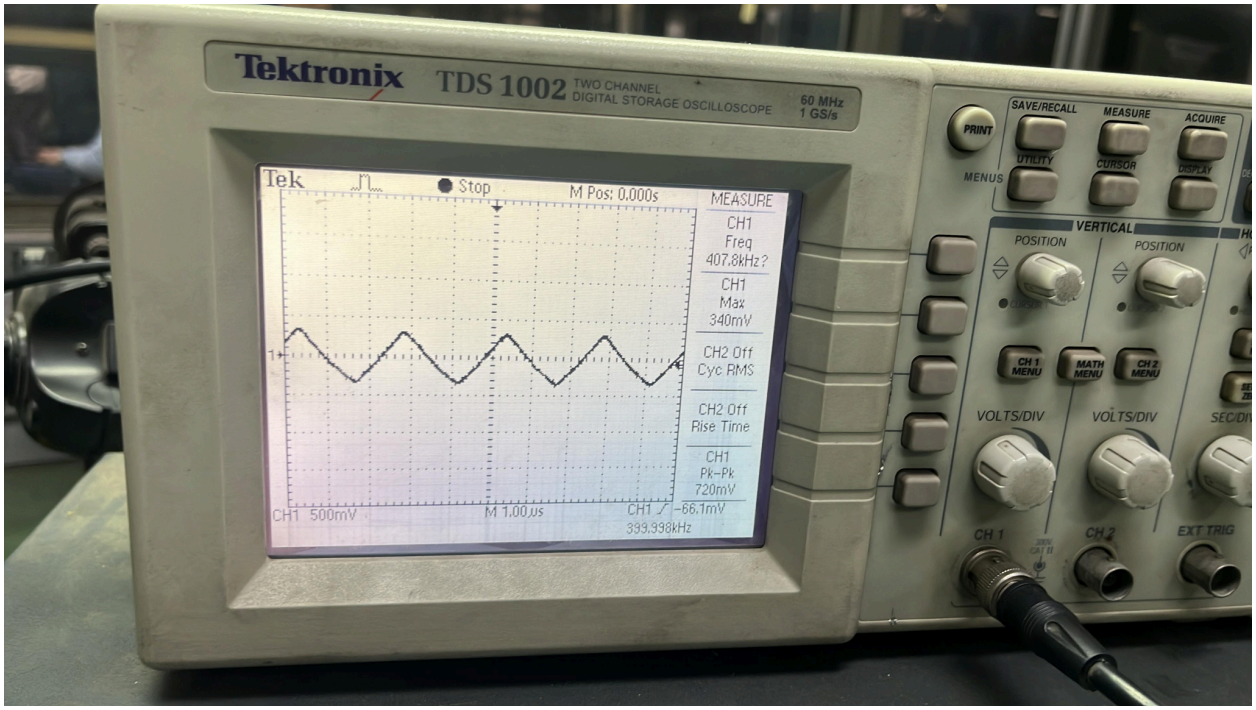


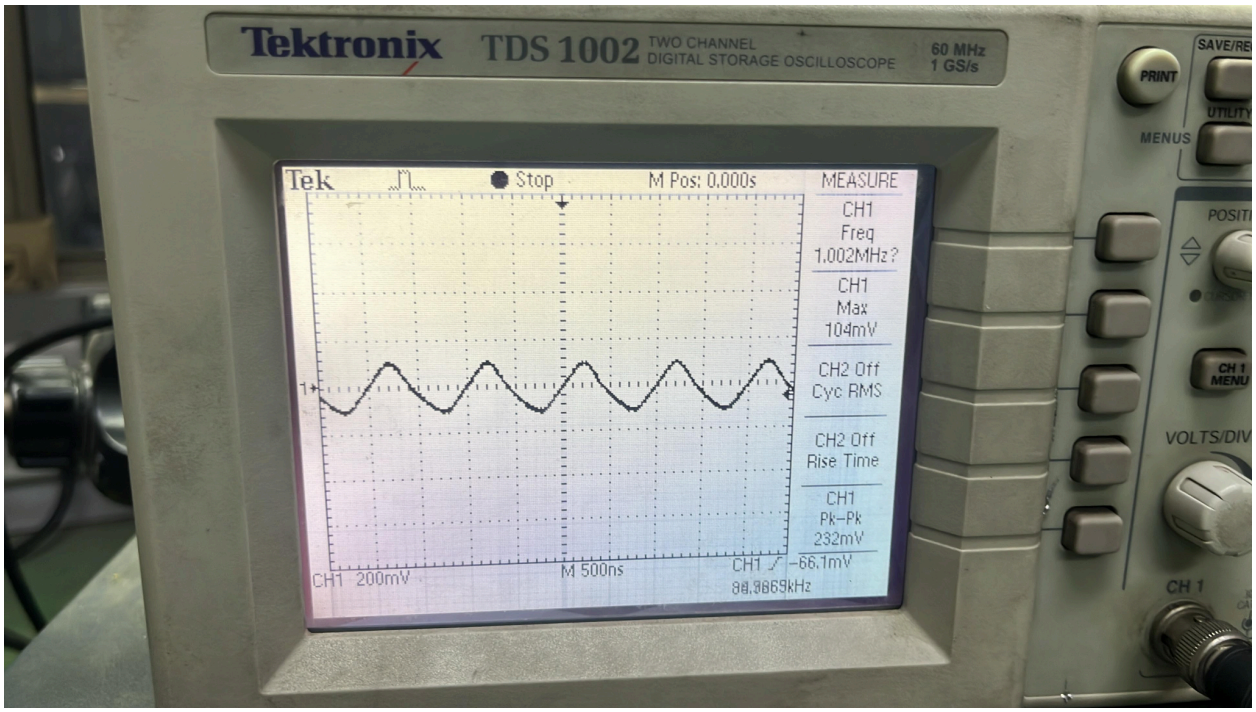





Tektronix TDS 1002 TWO CHANNEL DIGITAL STORAGE OSCILLOSCOPE 60 MHz 1 GS/s
Tek Trig'd M Pos: 0.000s
MEASURE
CH1 Freq 1.00kHz
CH1 Max 1.06V
CH2 Off Cyc RMS
CH2 Off Rise Time
CH1 Pk-Pk 2.10V
CH1 500mV M 500µs CH1 -55.1mV 999.975Hz
SAVE/RECALL MEASURE ACQUIRE
UTILITY CURSOR DISPLAY
PRINT
MENUS
VERTICAL
POSITION POSITION
CH 1 MENU MATH MENU CH 2 MENU
VOLTS/DIV VOLTS/DIV
CH 1 CH 2 EXT TRIG


Tektronix TDS 1002 TWO CHANNEL DIGITAL STORAGE OSCILLOSCOPE 60 MHz 1 GS/s
Tek Stop M Pos: 0.000s
MEASURE
CH1 Freq 99.80kHz
CH1 Max 1.10V
CH2 Off Cyc RMS
CH2 Off Rise Time
CH1 Pk-Pk 2.22V
CH1 500mV M 5.00µs CH1 -66.1mV 99.9395kHz
SAVE/RECALL MEASURE ACQUIRE
UTILITY CURSOR DISPLAY
PRINT
MENUS
VERTICAL
POSITION POSITION
CH 1 MENU MATH MENU CH 2 MENU
VOLTS/DIV VOLTS/DIV
CH 1 CH 2 EXT TRIG



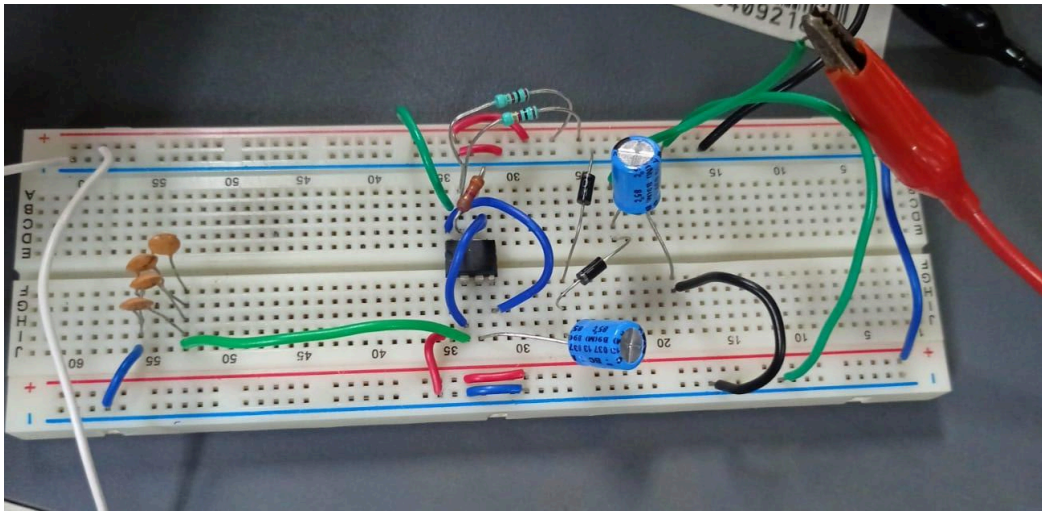


Frequency	Input Voltage	Output Voltage	Shape
100Hz	2.2V(p/p)	2.2V(p/p)	Sinusoid
1000Hz	2.2V(p/p)	2.2V(p/p)	Sinusoid
10kHz	2.2V(p/p)	2.2V(p/p)	Sinusoid

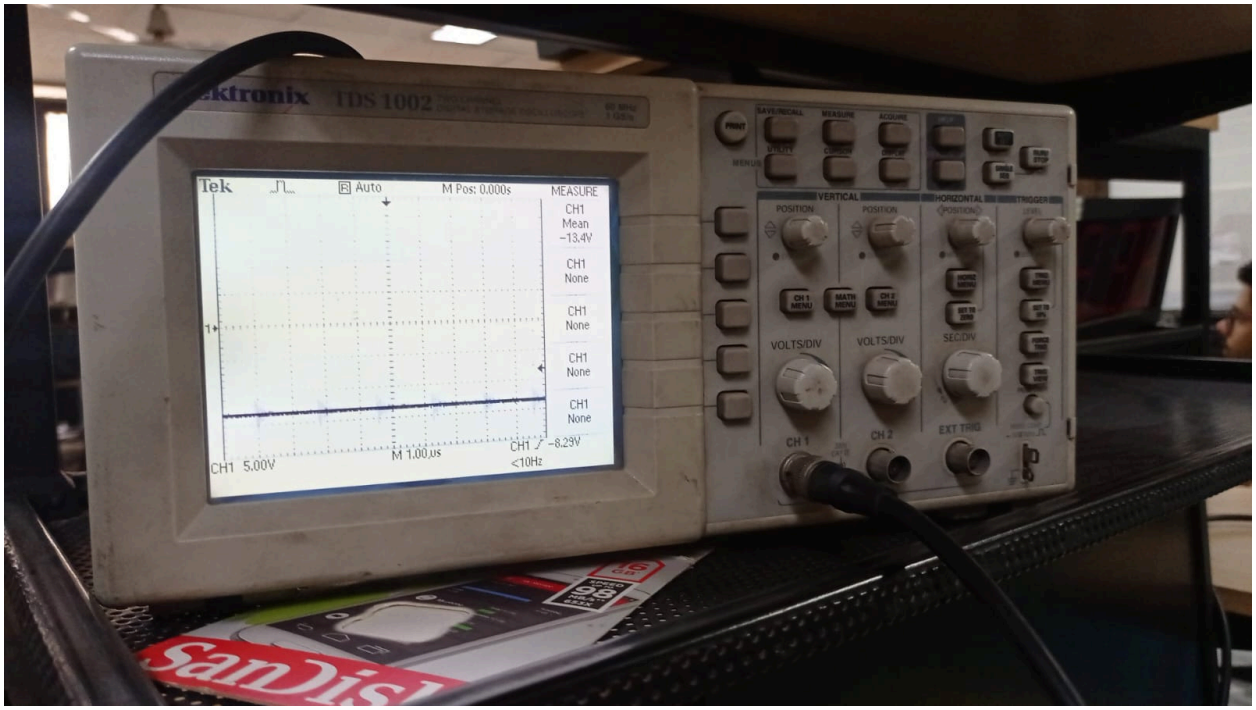
100kHz	2.2V(p/p)	2.2V(p/p)	Sinusoid
400kHz	2.2V(p/p)	0.72V(p/p)	Improper
1MHz	2.2V(p/p)	0.232V(p/p)	Improper

CONCLUSION:
We have ordered another OpAmpLM358, but wil be less suitable given the worse slew rate, ot we have to use this OpAmp LM 741CN, we will limit its use till 200kHz range for amplification.

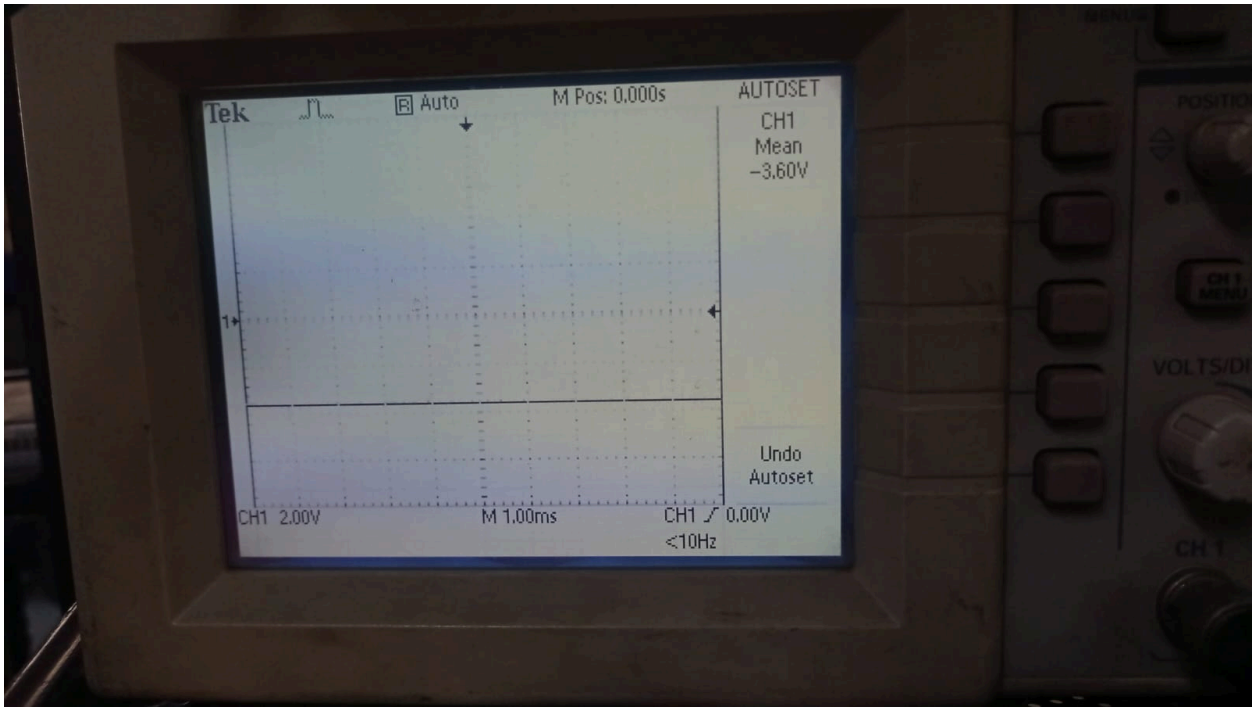
Task-2: Inverting the input
Assigned to: Dhanashri and Divyanshu



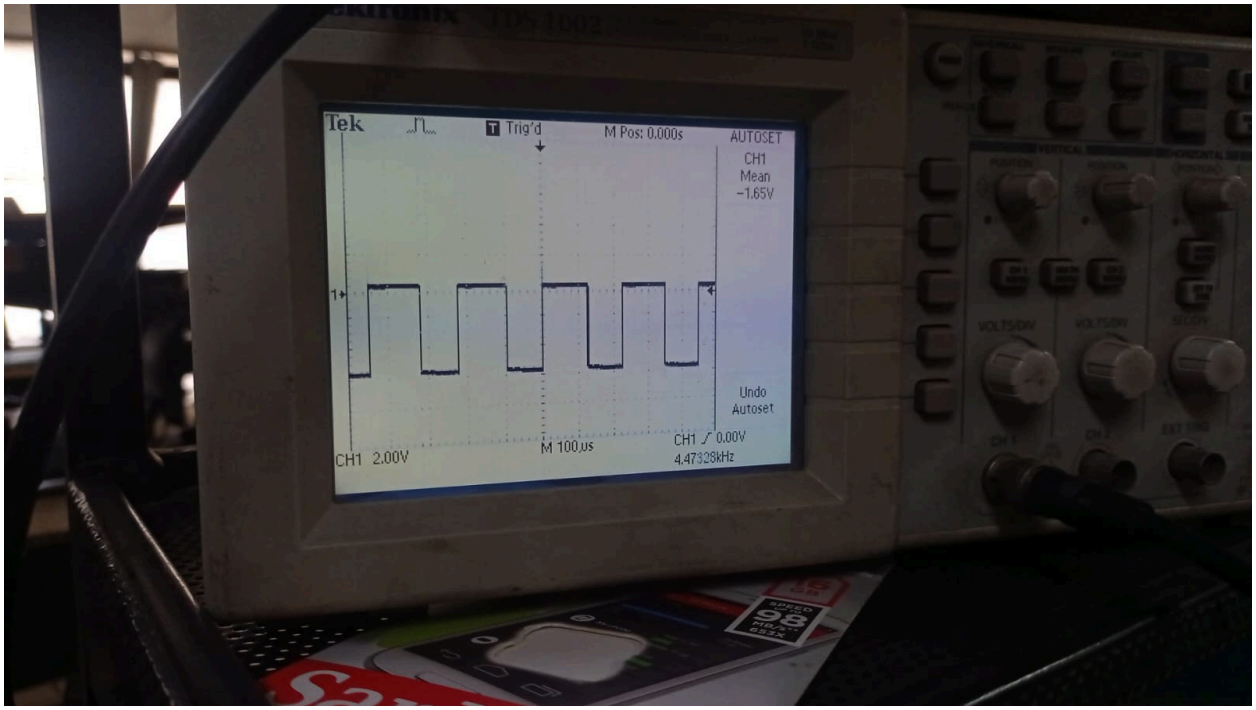
Circuit Diagram of Inverter using 555 Timer IC



Input voltage: 15V Output Voltage: -13.4 V



Input voltage: 5V Output Voltage: -3.6 V



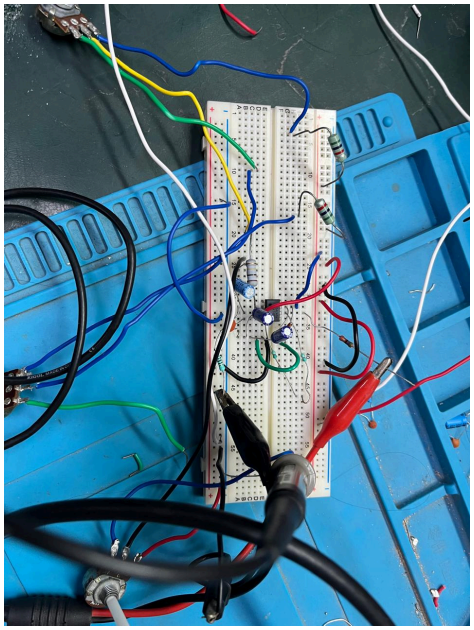
OUTPUT WAVE FROM 555 timer

Problem

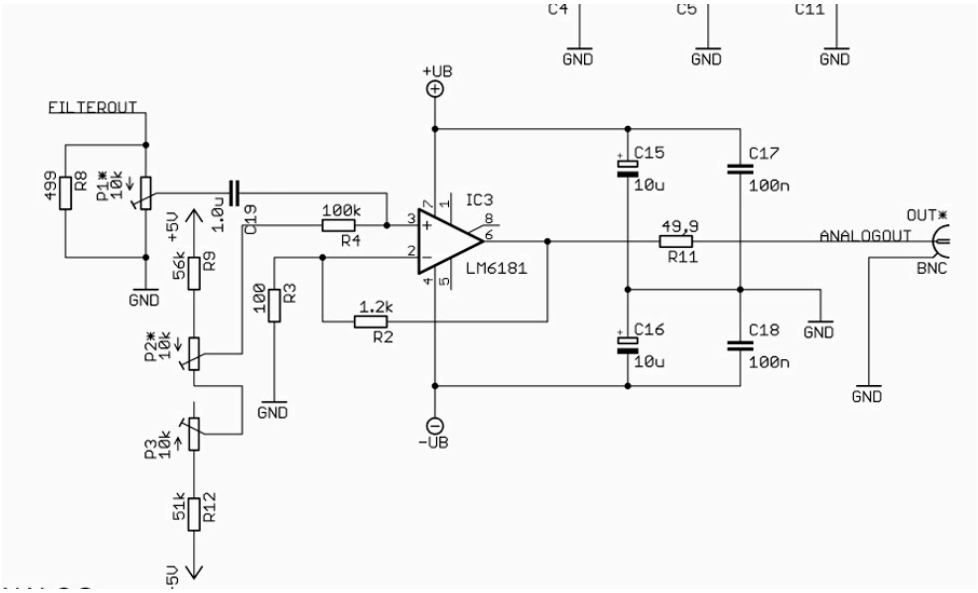
Input voltage	Output Voltage IntendeD(IN Volts)	Output Voltage Expected(in Volts)	error
15V	-13.4	-15	10%
5V	-3.6	-5	28%

Solution: Using IC 7905 AND 7915 for negative voltage regulation
Cost: appox, rs. 450
Given that we can use lab power supply for demonstration purposes-> we will not incur this cost

Task-3: Offset Circuitry



BREADBOARD Picture



SCHEMATIC

P1:control amplitude

P2& P3: control offset

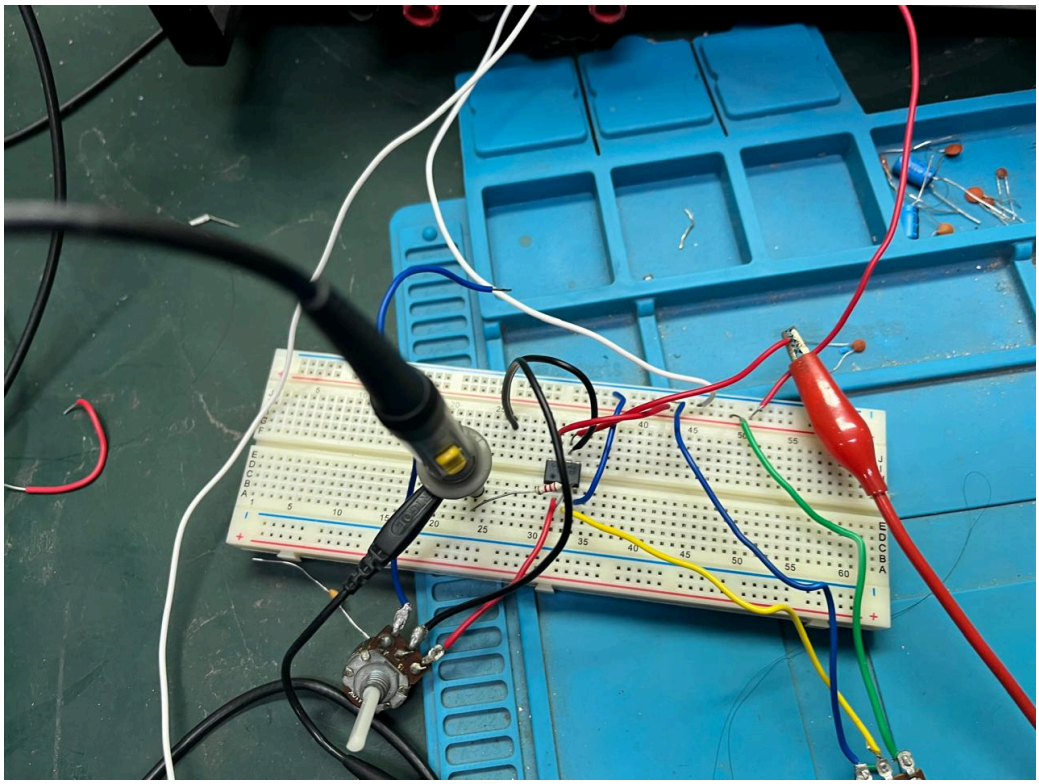
On changing resistance of potentiometer, change was observed in the amplitude and the offset

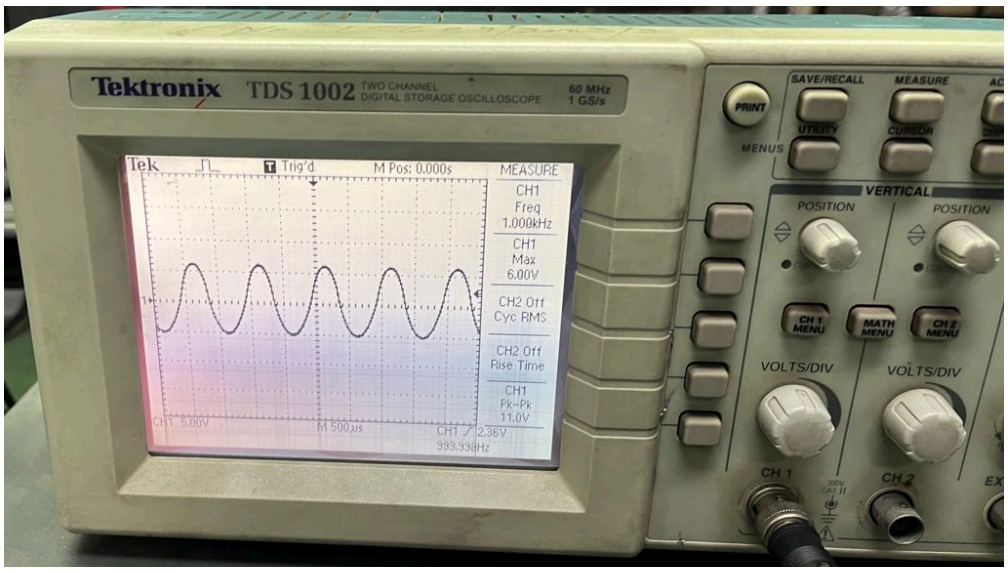
To note that amplitude of sinusoid <amplitude of offset, otherwise clipping will occur

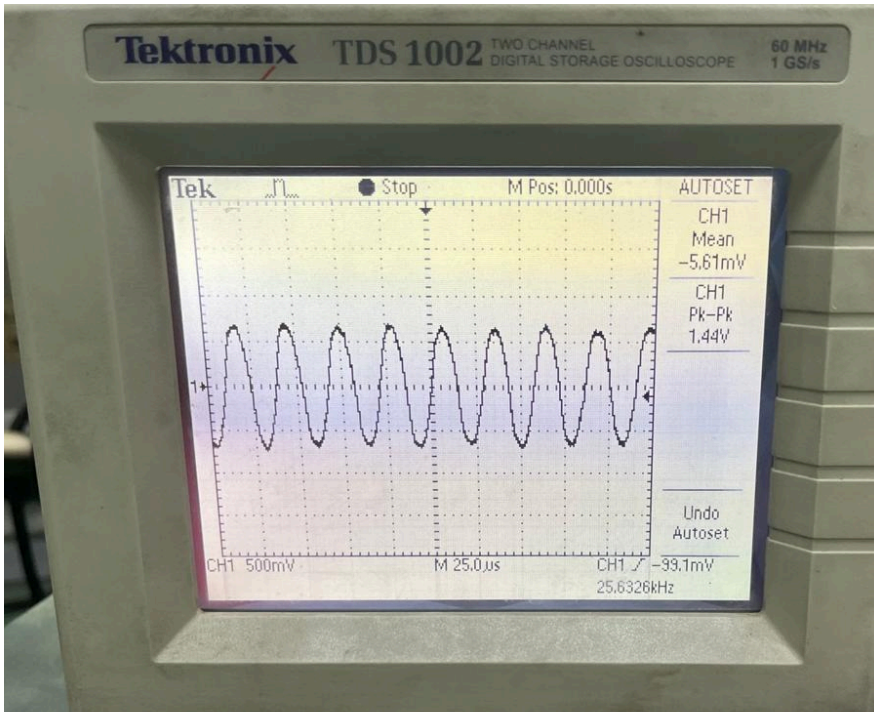
Shortcomings: Offset control was not working

Sources of error: due to the circuit being extensive, few unwanted wires might touch and get shorted.

Approach-2: Using 2 potentiometers at op amp terminals, for offset and feedback control. A more stable wave was obtained and offset control of the range +-1V was obtained

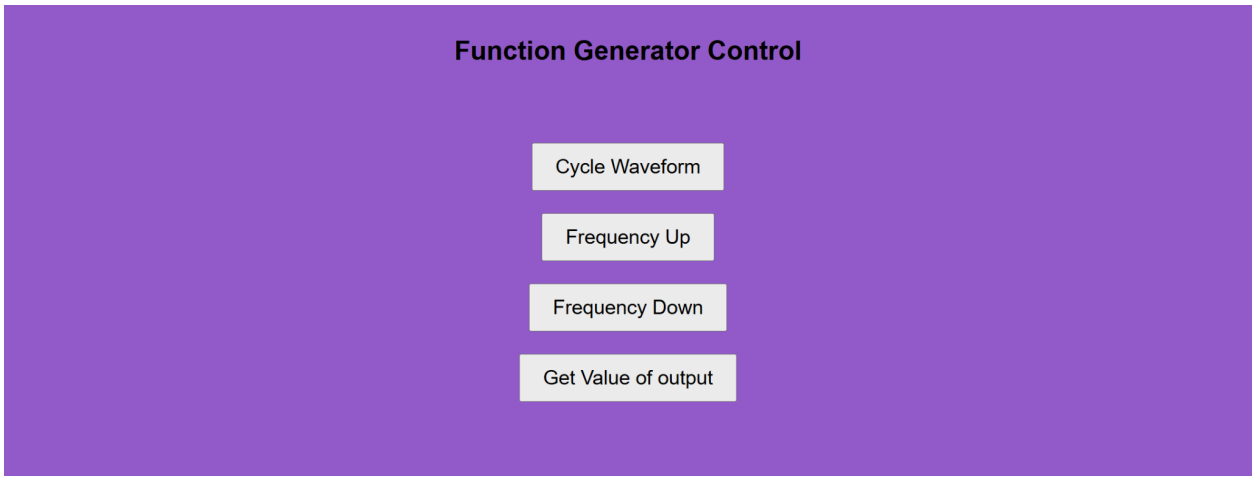



Tektronix
TDS 1002
TWO CHANNEL
DIGITAL STORAGE OSCILLOSCOPE
60 MHz
1 GS/s
PRINT
SAVE/RECALL
MEASURE
UTILITY
CURSOR
MENUS
VERTICAL
POSITION
POSITION
CH 1
MENU
MATH
MENU
CH 2
MENU
VOLTS/DIV
VOLTS/DIV
CH 1
CH 2
Tek
Trig'd
M Pos: 0.000s
MEASURE
CH1
Freq
1.000kHz
CH1
Max
6.00V
CH2 Off
Cyc RMS
CH2 Off
Rise Time
CH1
Pk-Pk
11.0V
CH1 5.00V
M 500 µs
CH1 2.36V
999.938Hz



Next intended steps:
Simulation of Offset circuitry to provide +-5V offset
Soldering components for accurate circuitry.

TASK-4: Future scope of software design
Incorporating an input port, which will measure DC voltage, use esp32 WiFi to send the voltage reading to the software, for tabulation as frequency sweep happens automatically .
Assigned to: Dhruv Belawat



The frequency can also be controlled directly making it increase or decrease as per convenience.
The waveforms will also be switched upon clicking the cycle waform button wherein it will shuffle between the many types that will include square wave, triangle wave , sine wave etc.

This code sets up an ESP32 microcontroller to serve a web page that allows users to control a function generator and Creates a web server on port 80.

```
#include <WiFi.h>
#include <WebServer.h>

// WiFi credentials
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

// Create web server on port 80
WebServer server(80);

// HTML Webpage for the Function Generator Control
const char MAIN_page[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html>
<head>
    <title>Function Generator Control</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: rgb(157, 89, 208);
        }
        h1 {
            font-size: 24px;
            margin-bottom: 30px; /* Add spacing after the title */
        }
        button {
            display: block;
            margin: 20px auto; /* Increase spacing between buttons */
            padding: 10px 20px;
            font-size: 18px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <br>
    <br>
    <h1>Function Generator Control</h1>
    <br>

    <button onclick="sendCommand('cycle_waveform')">Cycle Waveform</button>
    <button onclick="sendCommand('freq_up')">Frequency Up</button>
    <button onclick="sendCommand('freq_down')">Frequency Down</button>
    <button onclick="sendCommand('getvalue')">Get Value of output</button>

    <script>
        function sendCommand(command) {
            fetch('/' + command)
            .then(response => console.log(command + ' command sent'))
            .catch(error => console.error('Error:', error));
        }
    </script>
</body>
</html>
)rawliteral";

// Global variables
float frequency = 1.0;
int waveform = 0; // 0: Triangle, 1: Square, 2: Sine

// Handle web requests
void handleRoot() {
    server.send(200, "text/html", MAIN_page);
}

// Function to handle button commands
```

```
void handleCommand() {
    String command = server.uri().substring(1); // Get command from URL
    Serial.println("Received command: " + command);

    if (command == "cycle_waveform") {
        // Cycle waveform
        waveform = (waveform + 1) % 3; // Cycle through 0, 1, 2
        String waveFormName = waveform == 0 ? "Triangle" : waveform == 1 ? "Square" : "Sine";
        Serial.println("Waveform: " + waveFormName);
    } else if (command == "freq_up") {
        // Increase frequency
        frequency *= 10;
        Serial.println("Frequency: " + String(frequency));
    } else if (command == "freq_down") {
        // Decrease frequency
        frequency /= 10;
        Serial.println("Frequency: " + String(frequency));
    } else if (command == "getvalue") {
        // Get voltage value at port 1 (GPIO1)
        int analogValue = analogRead(1);
        float voltage = analogValue * (3.3 / 4095.0); // Assuming 12-bit ADC and 3.3V reference
        Serial.println("Voltage at GPIO1: " + String(voltage) + " V");
        server.send(200, "text/plain", "Voltage at GPIO1: " + String(voltage) + " V");
    }
}
```

```
void setup() {
    Serial.begin(115200);

    // Connect to WiFi
    WiFi.begin(ssid, password);
    Serial.print("Connecting to WiFi...");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected to WiFi!");
    Serial.println(WiFi.localIP()); // Print ESP32's IP Address

    // Start server
    server.on("/", handleRoot);
    server.onNotFound(handleCommand);
    server.begin();
}

void loop() {
    server.handleClient();
}
```

The code displays an html file when opened on the esp private server. It is to be run on the aurdino IDE after connecting with the esp32.

```
#include <WiFi.h>
#include <WebServer.h>

// WiFi credentials
const char* ssid = "ESP32";
const char* password = "PASSWORD";

// Create web server on port 80
WebServer server(80);

// HTML Webpage for the Function Generator Control
const char MAIN_page[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html>
<head>
  <title>Function Generator Control</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      background-color: rgb(157, 89, 208);
    }
    h1 {
      font-size: 24px;
      margin-bottom: 30px; /* Add spacing after the title */
    }
    button {
      display: block;
      margin: 20px auto; /* Increase spacing between buttons */
      padding: 10px 20px;
      font-size: 18px;
      cursor: pointer;
    }
  </style>
</head>
<body>
  <br>
  <br>
  <h1>Function Generator Control</h1>
  <br>

  <button onclick="sendCommand('cycle_waveform')">Cycle Waveform</button>
  <button onclick="sendCommand('freq_up')">Frequency Up</button>
```

```
    <button onclick="sendCommand('freq_down')">Frequency Down</button>
    <button onclick="sendCommand('getvalue')">Get Value of output</button>

    <script>
        function sendCommand(command) {
            fetch('/' + command)
            .then(response => console.log(command + ' command sent'))
            .catch(error => console.error('Error:', error));
        }
    </script>
</body>
</html>
)rawliteral";

// Global variables
float frequency = 1.0;
int waveform = 0; // 0: Triangle, 1: Square, 2: Sine

// Handle web requests
void handleRoot() {
    server.send(200, "text/html", MAIN_page);
}

// Function to handle button commands
void handleCommand() {
    String command = server.uri().substring(1); // Get command from URL
    Serial.println("Received command: " + command);

    if (command == "cycle_waveform") {
        // Cycle waveform
        waveform = (waveform + 1) % 3; // Cycle through 0, 1, 2
        String waveFormName = waveform == 0 ? "Triangle" : waveform == 1 ? "Square" : "Sine";
        Serial.println("Waveform: " + waveFormName);
    } else if (command == "freq_up") {
        // Increase frequency
        frequency *= 10;
        Serial.println("Frequency: " + String(frequency));
    } else if (command == "freq_down") {
        // Decrease frequency
        frequency /= 10;
        Serial.println("Frequency: " + String(frequency));
    } else if (command == "getvalue") {
        // Get voltage value at port 1 (GPIO1)
        int analogValue = analogRead(1);
```

```
        float voltage = analogValue * (3.3 / 4095.0); // Assuming 12-bit ADC and 3.3V reference
        Serial.println("Voltage at GPIO1: " + String(voltage) + " V");
        server.send(200, "text/plain", "Voltage at GPIO1: " + String(voltage) + " V");
    }
}

void setup() {
    Serial.begin(115200);

    // Connect to WiFi
    WiFi.begin(ssid, password);
    Serial.print("Connecting to WiFi...");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected to WiFi!");
    Serial.println(WiFi.localIP()); // Print ESP32's IP Address

    // Start server
    server.on("/", handleRoot);
    server.onNotFound(handleCommand);
    server.begin();
}

void loop() {
    server.handleClient();
}
```