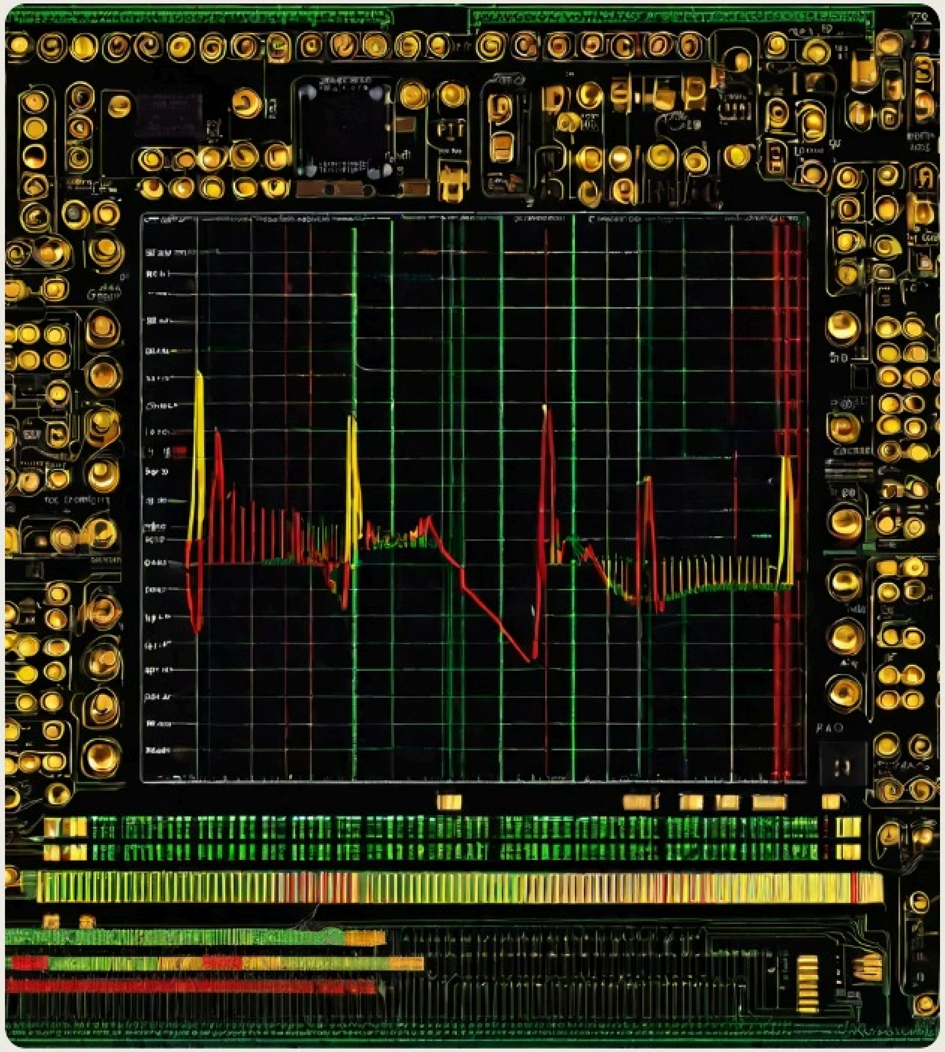
Digital Oscilloscope using Arduino Uno



Ctrl+K

Prepared by

Group PG4

Priyanshu Verma (2022ceb1024)

Raj Shrivastav (2022ceb1025)

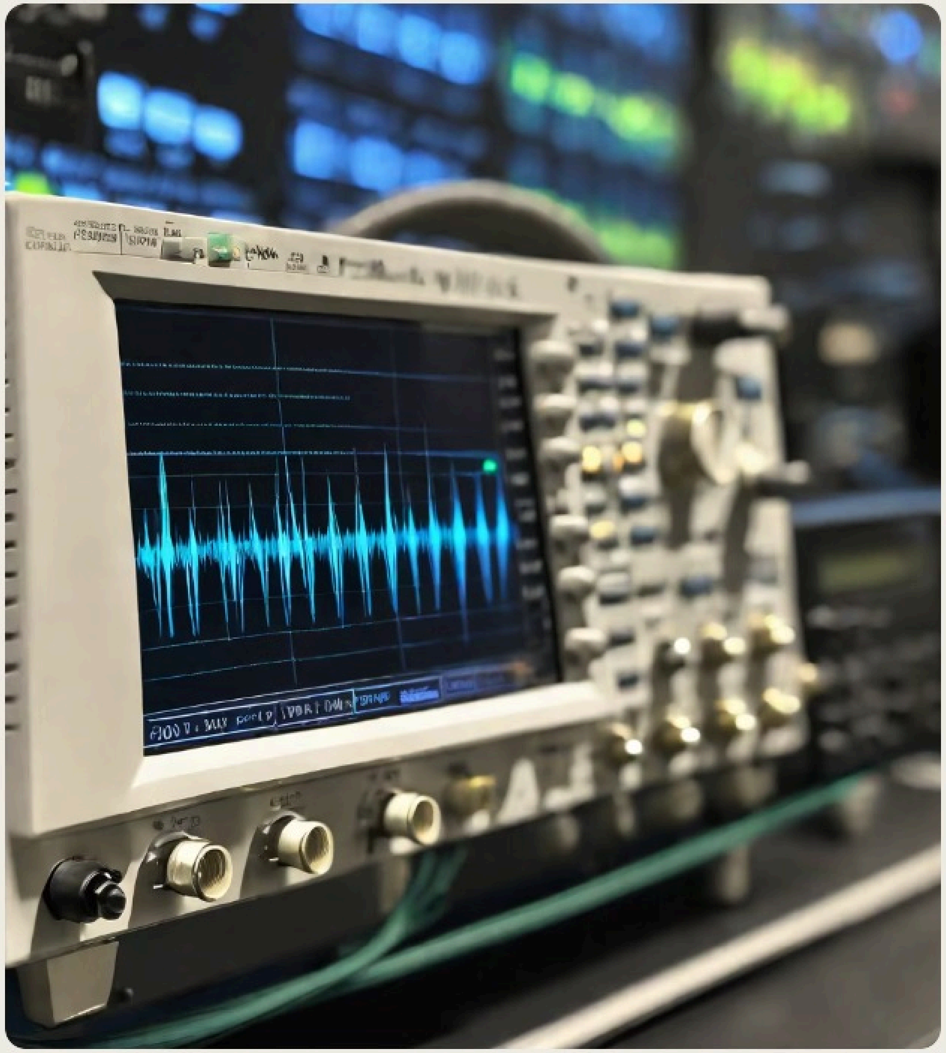
Rajat Gupta (2022ceb1026)

Rishabh Pratap Singh (2022ceb1027)

Rishi Gautam (2022ceb1028)

Introduction to Digital Oscilloscope

A digital oscilloscope is an electronic instrument used to visualize and analyze the waveform of electronic signals. It is an essential tool for anyone working with electronics, from hobbyists to professionals.



Project Overview

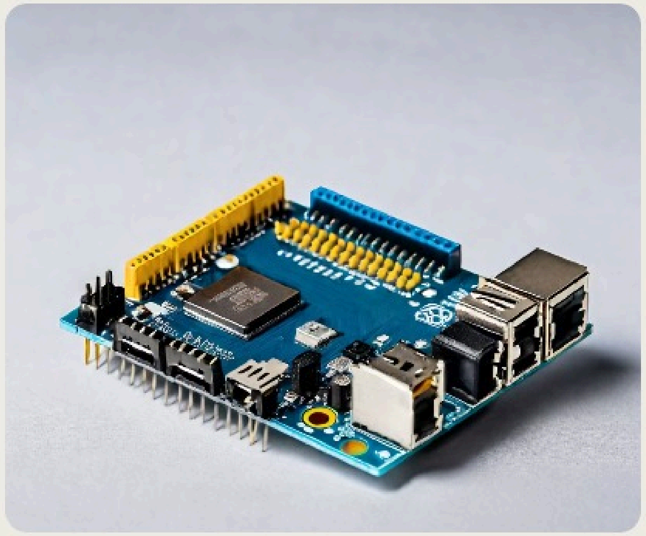
Digital Oscilloscope with Arduino Uno

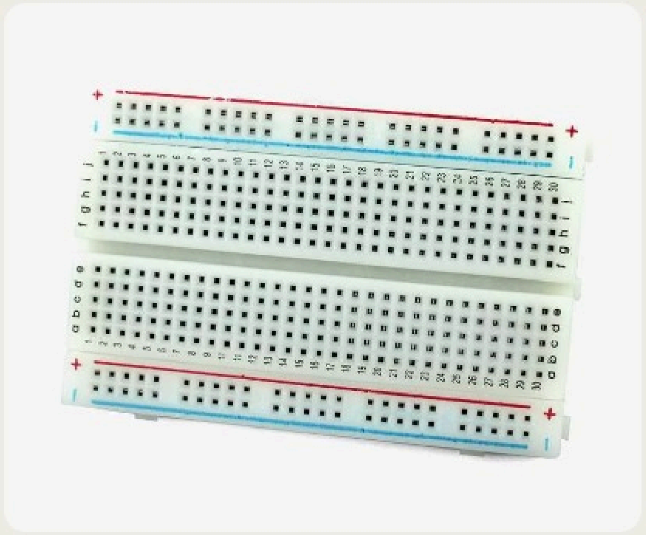
Our project involves creating a digital oscilloscope using an Arduino Uno microcontroller. This oscilloscope will be able to display and measure electronic signals, making it a versatile tool for electronics enthusiasts and professionals alike.

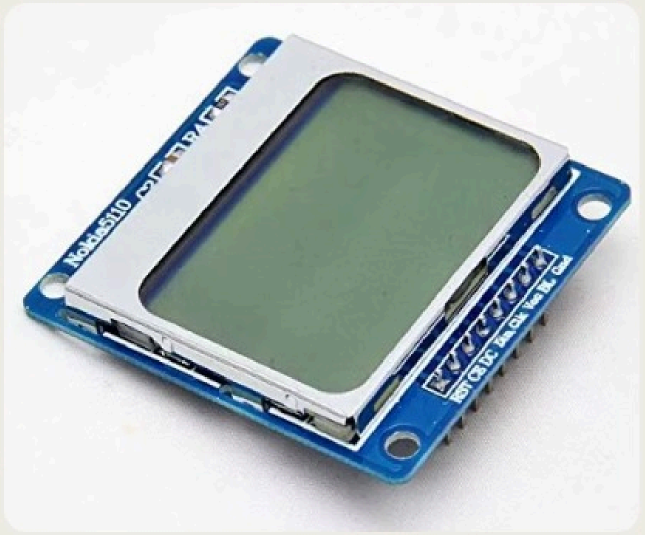
The project involves designing and building the hardware components, including the circuit board and display screen, as well as programming the software that will allow the oscilloscope to function. We will also be conducting tests and calibrations to ensure accurate measurements and reliable performance.

Hardware Requirements

To build the digital oscilloscope using an Arduino Uno, the following hardware components are required:







Arduino Uno Microcontroller Board

The Arduino Uno is the main microcontroller board that serves as the brain of the digital oscilloscope. It has a USB port for programming and communication with the computer, as well as digital and analog input/output pins for connecting to other hardware components

Breadboard with Wires

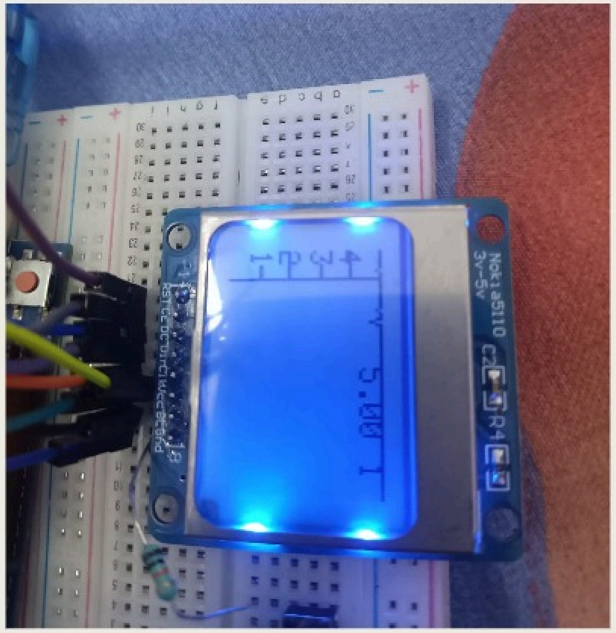
The breadboard and wires are used to create the necessary connections between the various hardware components. The breadboard provides a platform for easy prototyping and testing of the circuit, while the wires allow for flexible connections between the components

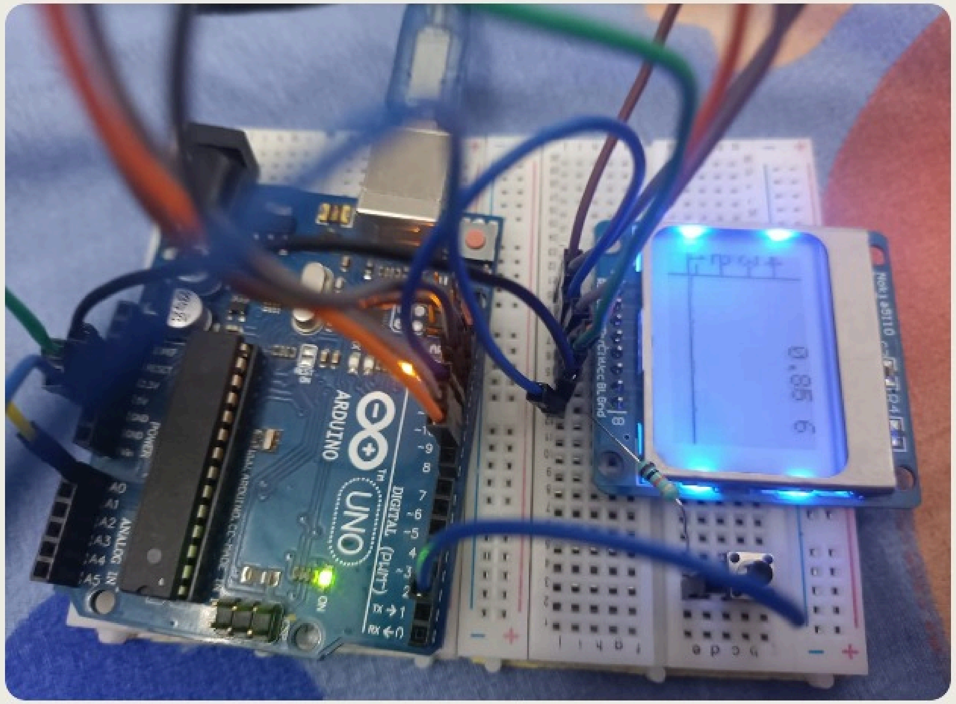
LCD Display Screen

The LCD display screen is used to display the waveforms captured by the digital oscilloscope. It provides a visual representation of the signal being measured and allows for easy analysis of the waveform characteristics.

Working of the Oscilloscope

The digital oscilloscope works by measuring and displaying voltage signals over time. It does this by taking samples of the input signal at regular intervals and converting them into digital values. These digital values are then plotted on a graph with time on the x-axis and voltage on the y-axis.





Challenges and Solutions

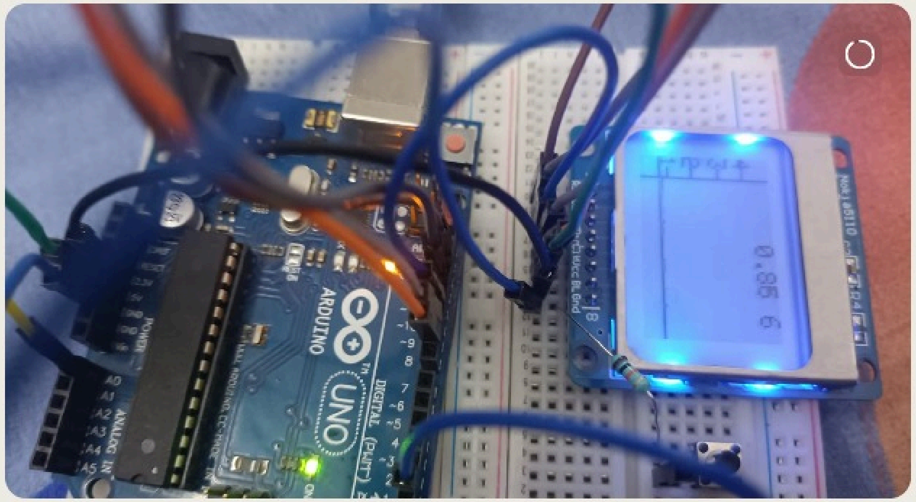
Challenges

Initially, we were aiming to design an oscilloscope with a function generator so we could give input as well. However due to the unavailability of the components we were not able to achieve our aim. But we came up with the idea of an oscilloscope with the probe in it.

Solution

This probe can take input and we can display the measured voltage on the TFT Display. We can generate sine wave function, square wave function, etc. but we have to program a different code for that.

Results and Observations



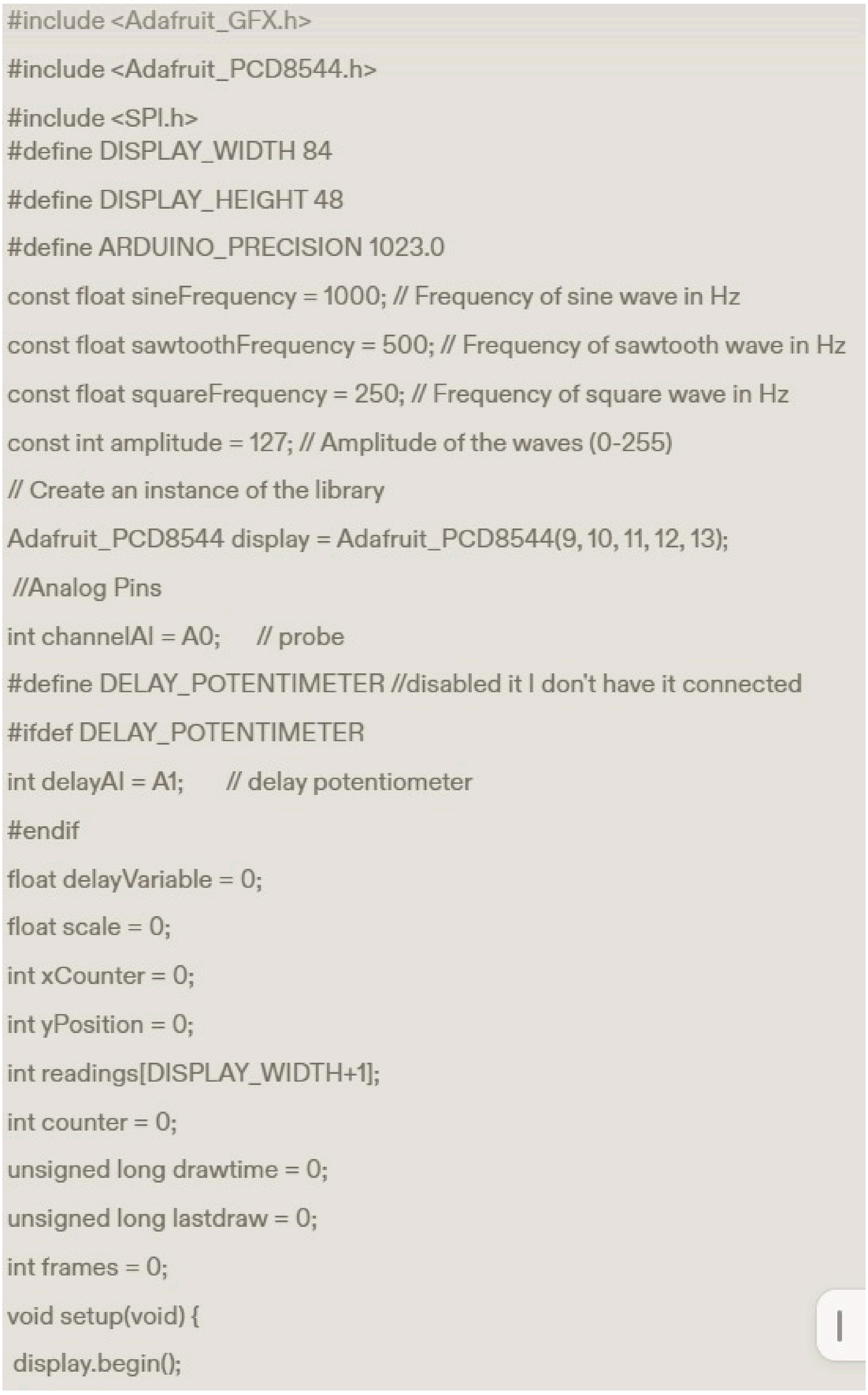
The digital oscilloscope project was successful in producing a functional device that can display waveforms on a screen.

During testing, we observed that the device was able to accurately display waveforms up to a certain frequency. Beyond that frequency, the device was not able to accurately display the waveforms and showed distortions.

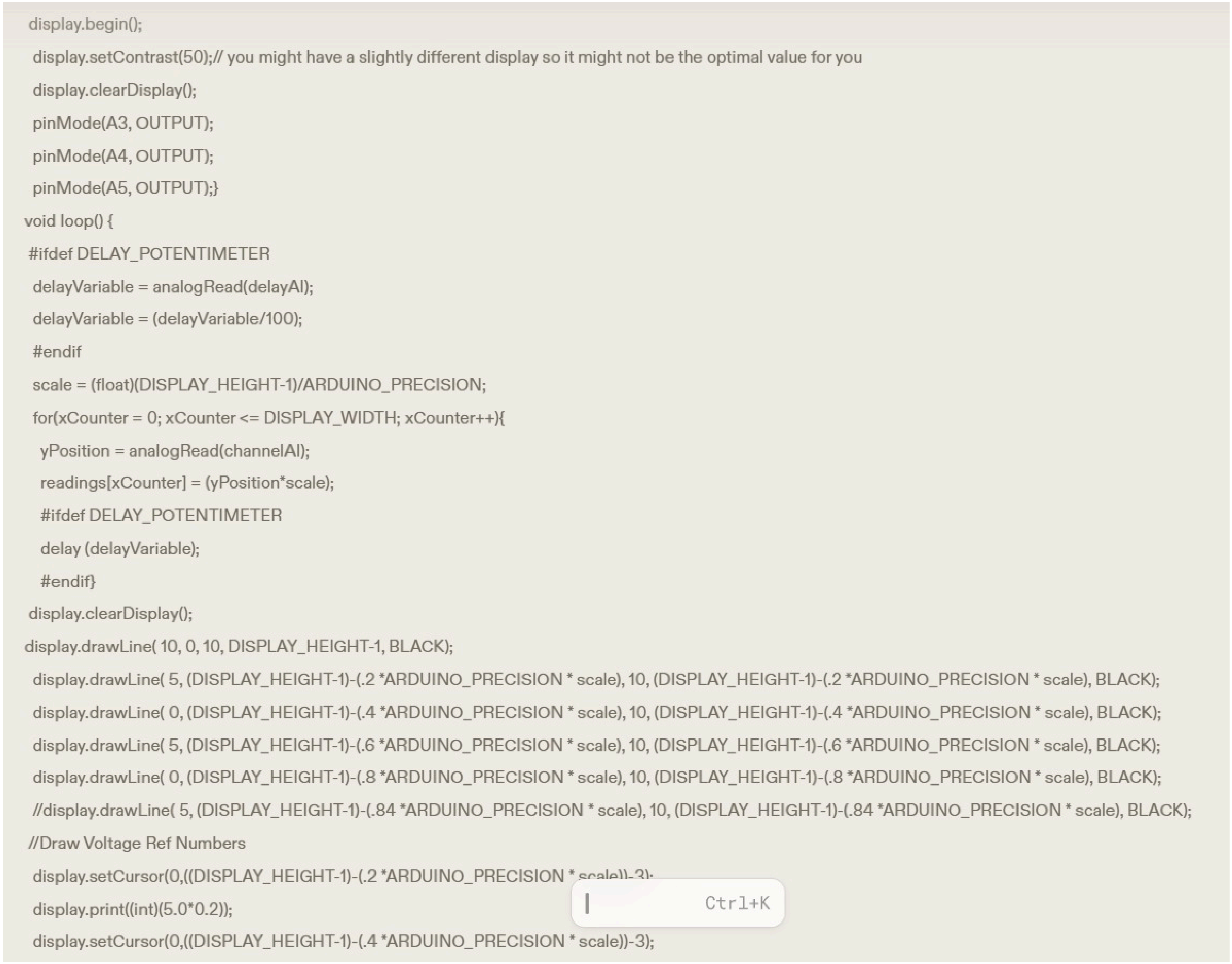
Future Improvements

To improve the performance of the digital oscilloscope, we plan to upgrade the microcontroller to a more powerful one that can handle higher frequencies and provide better accuracy.

We also plan to add additional features such as triggering and measurement capabilities to make the device more versatile and useful for a wider range of applications.



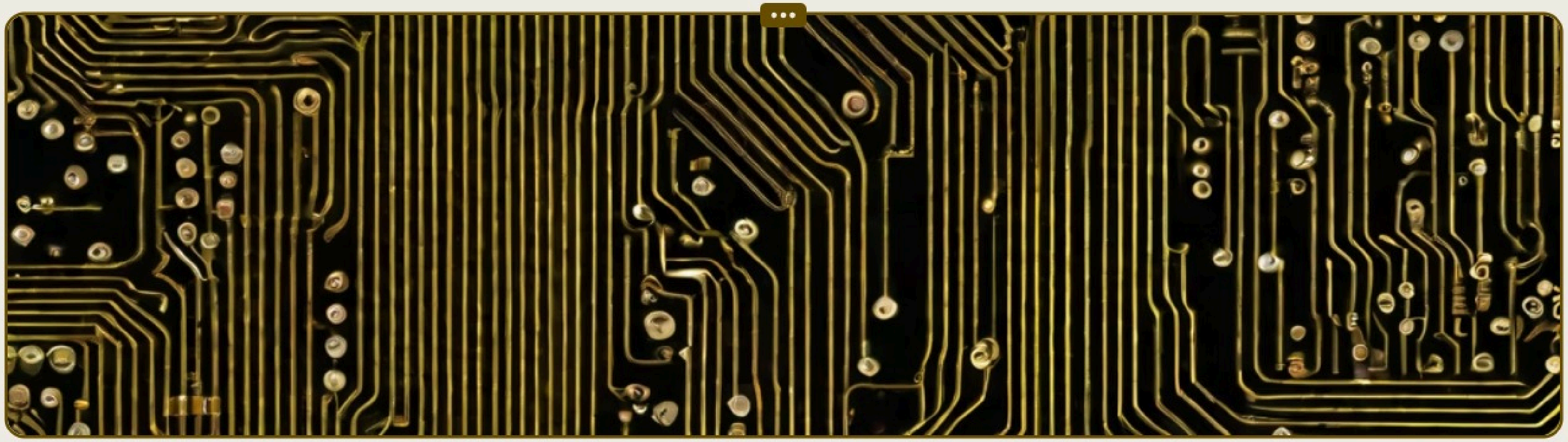
```
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
#include <SPI.h>
#define DISPLAY_WIDTH 84
#define DISPLAY_HEIGHT 48
#define ARDUINO_PRECISION 1023.0
const float sineFrequency = 1000; // Frequency of sine wave in Hz
const float sawtoothFrequency = 500; // Frequency of sawtooth wave in Hz
const float squareFrequency = 250; // Frequency of square wave in Hz
const int amplitude = 127; // Amplitude of the waves (0-255)
// Create an instance of the library
Adafruit_PCD8544 display = Adafruit_PCD8544(9, 10, 11, 12, 13);
 //Analog Pins
int channelAI = A0;     // probe
#define DELAY_POTENTIMETER //disabled it I don't have it connected
#ifdef DELAY_POTENTIMETER
int delayAI = A1;     // delay potentiometer
#endif
float delayVariable = 0;
float scale = 0;
int xCounter = 0;
int yPosition = 0;
int readings[DISPLAY_WIDTH+1];
int counter = 0;
unsigned long drawtime = 0;
unsigned long lastdraw = 0;
int frames = 0;
void setup(void) {
 display.begin();
```



```
 display.begin();
  display.setContrast(50);// you might have a slightly different display so it might not be the optimal value for you
  display.clearDisplay();
  pinMode(A3, OUTPUT);
  pinMode(A4, OUTPUT);
  pinMode(A5, OUTPUT);}
void loop() {
 #ifdef DELAY_POTENTIMETER
  delayVariable = analogRead(delayAI);
  delayVariable = (delayVariable/100);
  #endif
  scale = (float)(DISPLAY_HEIGHT-1)/ARDUINO_PRECISION;
  for(xCounter = 0; xCounter <= DISPLAY_WIDTH; xCounter++){
   yPosition = analogRead(channelAI);
   readings[xCounter] = (yPosition*scale);
   #ifdef DELAY_POTENTIMETER
   delay (delayVariable);
   #endif}
 display.clearDisplay();
display.drawLine( 10, 0, 10, DISPLAY_HEIGHT-1, BLACK);
  display.drawLine( 5, (DISPLAY_HEIGHT-1)-(.2 *ARDUINO_PRECISION * scale), 10, (DISPLAY_HEIGHT-1)-(.2 *ARDUINO_PRECISION * scale), BLACK);
  display.drawLine( 0, (DISPLAY_HEIGHT-1)-(.4 *ARDUINO_PRECISION * scale), 10, (DISPLAY_HEIGHT-1)-(.4 *ARDUINO_PRECISION * scale), BLACK);
  display.drawLine( 5, (DISPLAY_HEIGHT-1)-(.6 *ARDUINO_PRECISION * scale), 10, (DISPLAY_HEIGHT-1)-(.6 *ARDUINO_PRECISION * scale), BLACK);
  display.drawLine( 0, (DISPLAY_HEIGHT-1)-(.8 *ARDUINO_PRECISION * scale), 10, (DISPLAY_HEIGHT-1)-(.8 *ARDUINO_PRECISION * scale), BLACK);
  //display.drawLine( 5, (DISPLAY_HEIGHT-1)-(.84 *ARDUINO_PRECISION * scale), 10, (DISPLAY_HEIGHT-1)-(.84 *ARDUINO_PRECISION * scale), BLACK);
 //Draw Voltage Ref Numbers
  display.setCursor(0,((DISPLAY_HEIGHT-1)-(.2 *ARDUINO_PRECISION * scale))-3);
  display.print((int)(5.0*0.2));
  display.setCursor(0,((DISPLAY_HEIGHT-1)-(.4 *ARDUINO_PRECISION * scale))-3);
```

```
//Draw Voltage Ref Numbers
 display.setCursor(0,((DISPLAY_HEIGHT-1)-(.2 *ARDUINO_PRECISION * scale))-3);
 display.print((int)(5.0*0.2));
 display.setCursor(0,((DISPLAY_HEIGHT-1)-(.4 *ARDUINO_PRECISION * scale))-3);
 display.print((int)(5.0*0.4));
 display.setCursor(0,((DISPLAY_HEIGHT-1)-(.6 *ARDUINO_PRECISION * scale))-3);
 display.print((int)(5.0*0.6));
 display.setCursor(0,((DISPLAY_HEIGHT-1)-(.8 *ARDUINO_PRECISION * scale))-3);
 display.print((int)(5.0*0.8));
  for(xCounter = 0; xCounter <= DISPLAY_WIDTH; xCounter++){
  display.drawPixel(xCounter, (DISPLAY_HEIGHT-1)-readings[xCounter], BLACK);
  if(xCounter>1){
    display.drawLine(xCounter-1, (DISPLAY_HEIGHT-1)-readings[xCounter-1], xCounter, (DISPLAY_HEIGHT-1)-readings[xCounter], BLACK);}}
 //Draw FPS
 display.setCursor((DISPLAY_WIDTH-1)-11,0);
 display.print(frames);
 //Draw Voltage
 display.setCursor(((DISPLAY_WIDTH-1)/2),0);
 display.print(analogRead(channelAI)/ARDUINO_PRECISION*5.0);
display.display();
 //Calculate FPS
 drawtime = micros();
 frames=1000000/a second//(drawtime-lastdraw);
 lastdraw = drawtime;}
```

Future Scope



Function Generator

In the future, we plan to add wireless connectivity to the digital oscilloscope, allowing for remote monitoring and control of the device.

Type something...

Individual Contributions

Rishabh Pratap Singh(2022CEB1027), Raj Shrivastav (2022CEB1025) and Rajat Gupta(2022ceb1026) were involved in the connection procedure, on the other hand, Rishi Gautam(2022CEB1028) and Priyanshu Verma (2022 CEB1024) handled programming section