Portable And Lightweight Oscilloscope

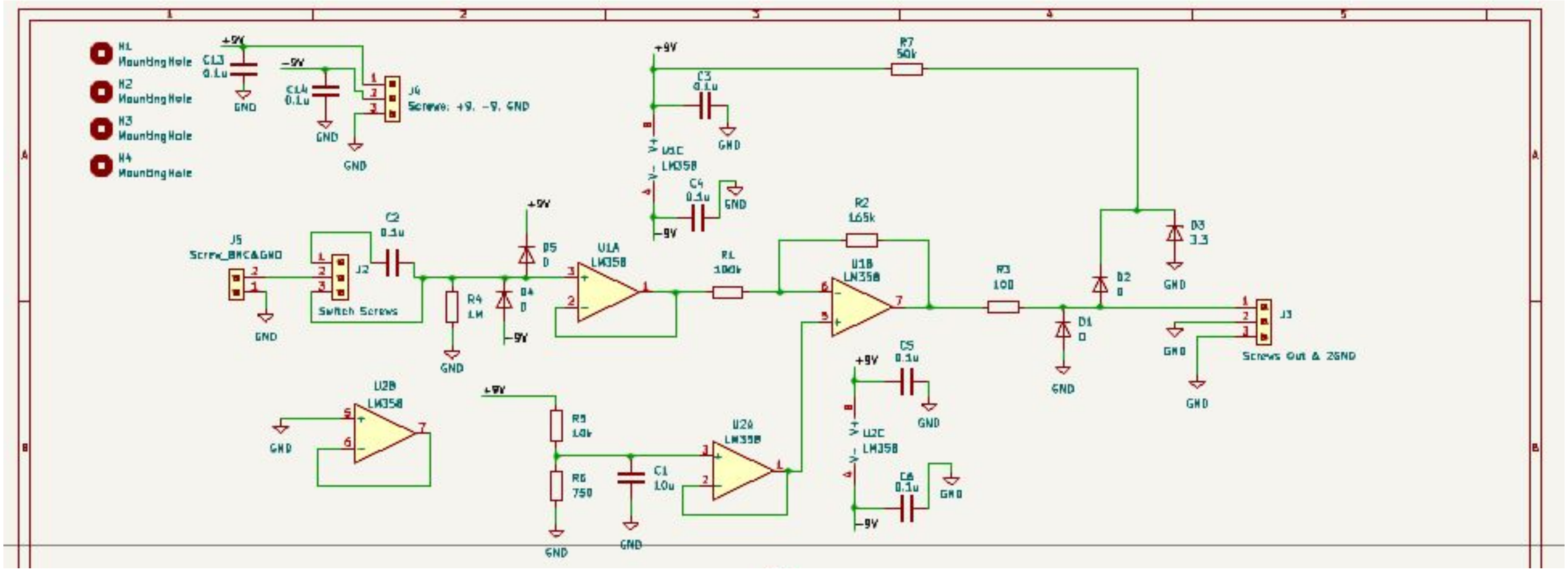
Group 6: Andrew Gondoputro, Luka Radovic, Gavin Le

Project Summary

- Designed a portable, dual-channel digital oscilloscope
- Supports ±10 V analog inputs and 200 kHz sampling per channel
- Built around a Teensy 4.1 with a fully custom analog front end (AFE)
- Real-time signal display on onboard LCD (no PC required)
- Single rotary encoder/button used for all user interaction (pause, trigger, scale)
- Main goals: Low-cost, portable, standalone oscilloscope for student/lab use
- Challenges included:
  - Conditioning ±10 V signals to 0–3.3 V ADC range
  - Achieving stable ADC performance
  - Integrating all blocks into a clean enclosure

Achievements: Verified performance, clean interface

Analog Front End

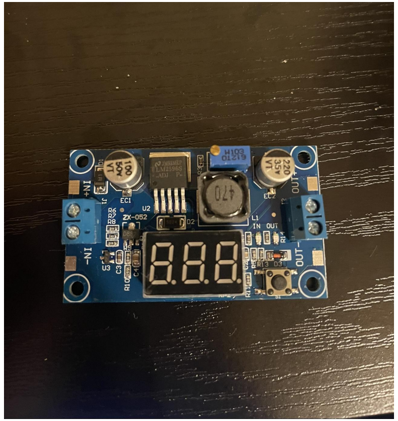


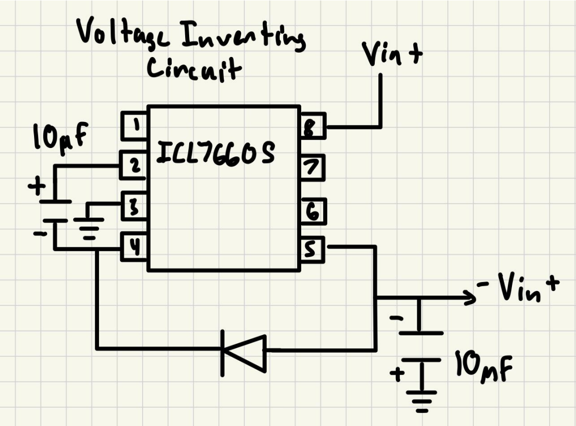
Power Block/Supply and User Interface/Display

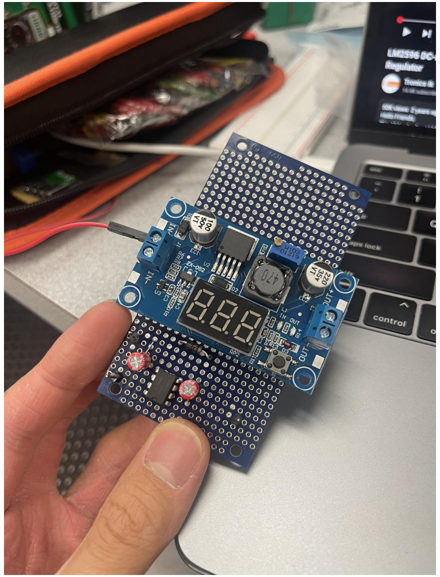
Gavin Le

Power Block/Supply

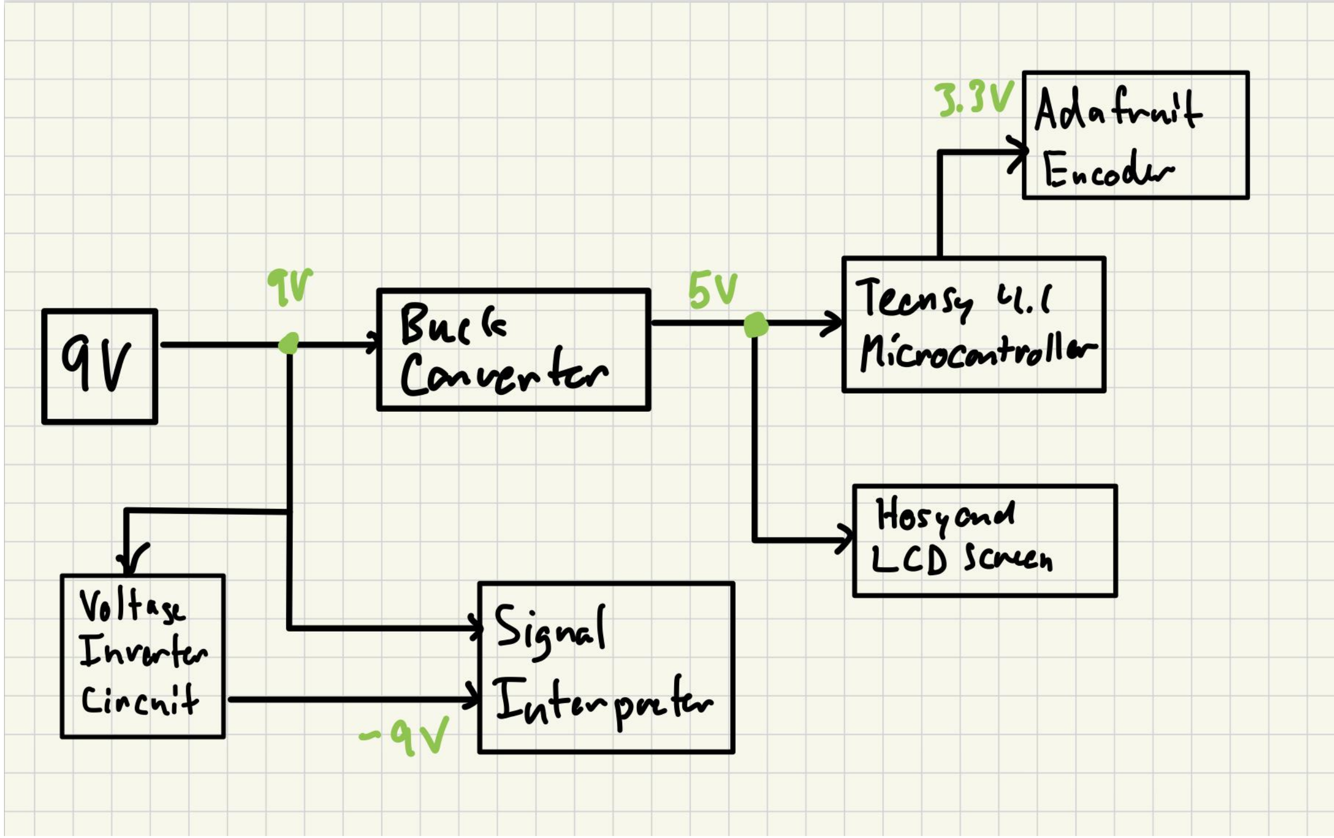
- Step down/convert voltages for the other oscilloscope components
- Eplzon 2596 Buck Converter and Voltage Inverter Circuit
- Each separate block supplies different voltages to different components of the oscilloscope







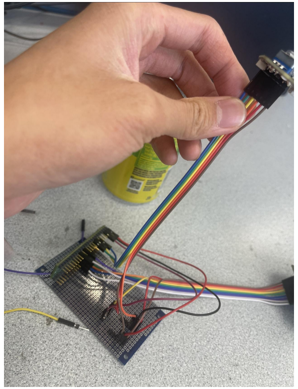
Diagram

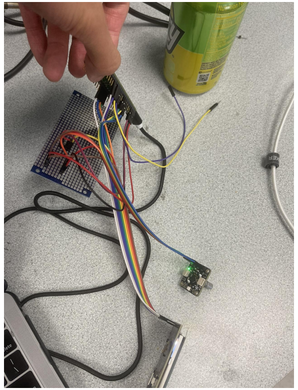


9V
9V
Buck Converter
5V
Teensy 4.1 Microcontroller
3.3V
Adafruit Encoder
Hosyond LCD Screen
Voltage Inverter Circuit
Signal Interpreter
-9V

User Interface/Display

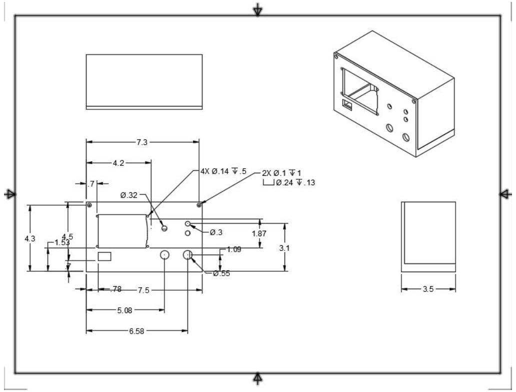
- One Adafruit I2C Stemma QT Rotary Encoder
  - Acts as button and knob
  - Allows for switching between channels, axes, selecting/triggering, and sweeping signals
- Hosyond IPS Capacitive Touch Screen LCD Module
  - Displays time and voltage axes, as well as electric signal
- Teensy 4.1 Microcontroller
  - Handles the code and functions of the oscilloscope
  - Acts as an interface bridge between the encoder and lcd screen



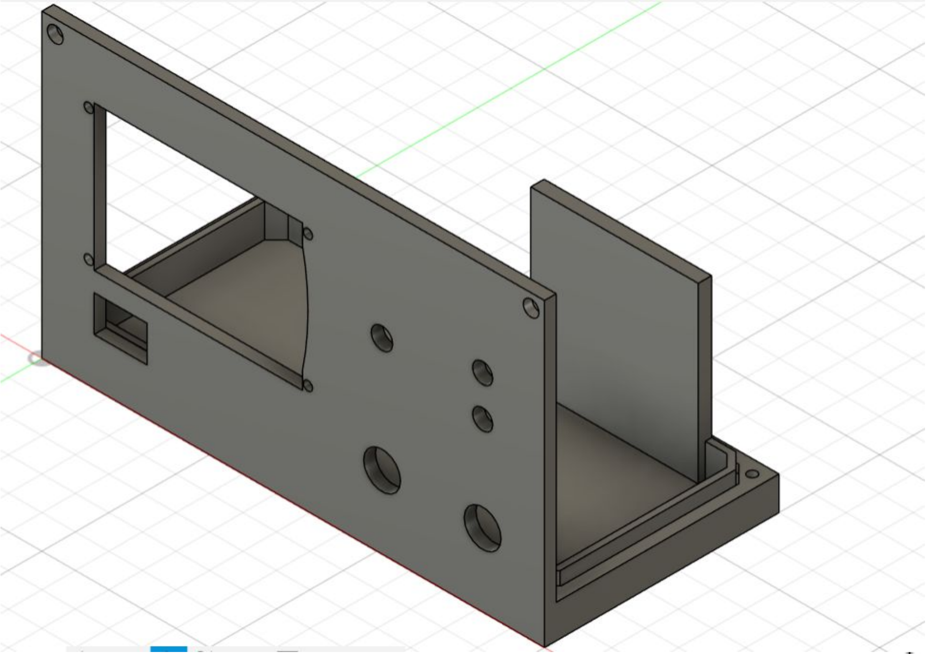


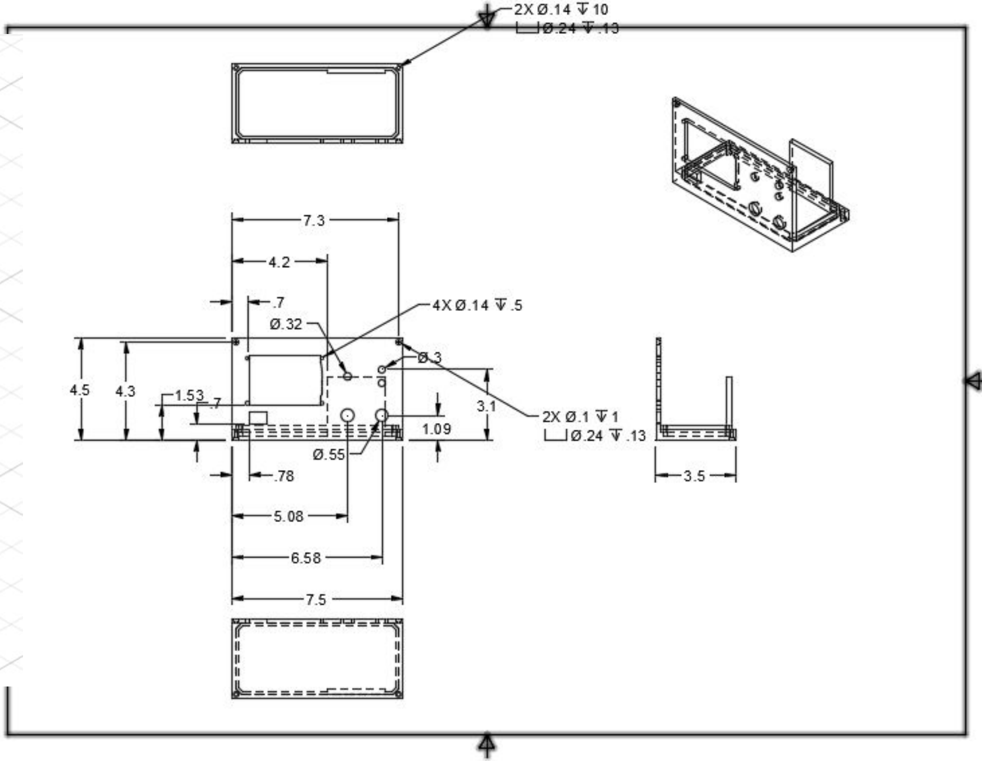
Enclosure

- Made in CAD for exact measurements -> 3D Printed in PLA Plastic
- Sizing of: 3.5 x 4.5 x 7.5in
- Slots for each UI component to fit properly either through tolerances or screws
- Interior holding all other blocks with proper placement

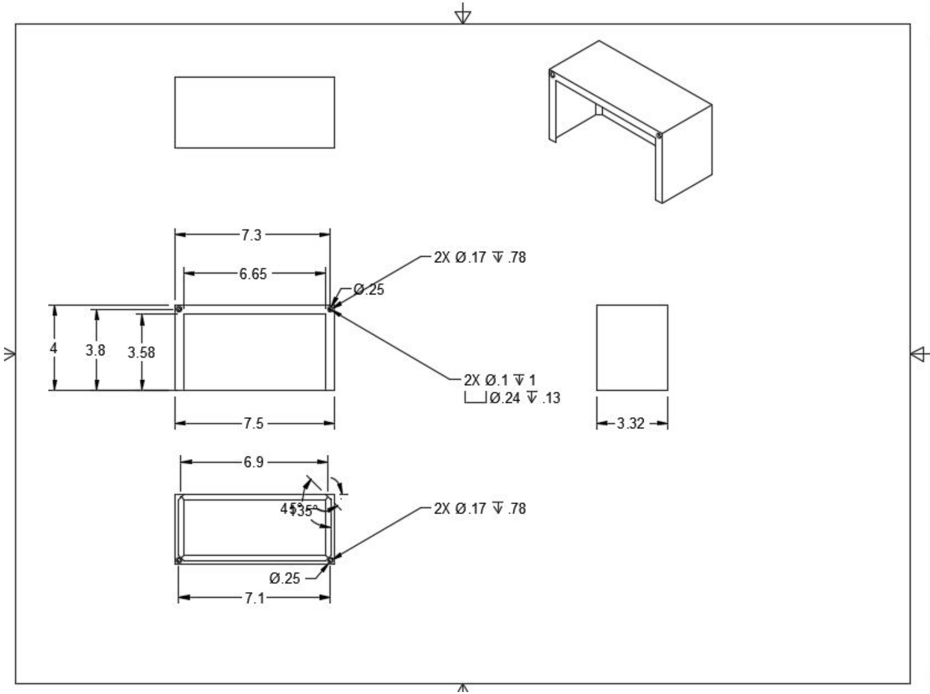


Bottom Piece

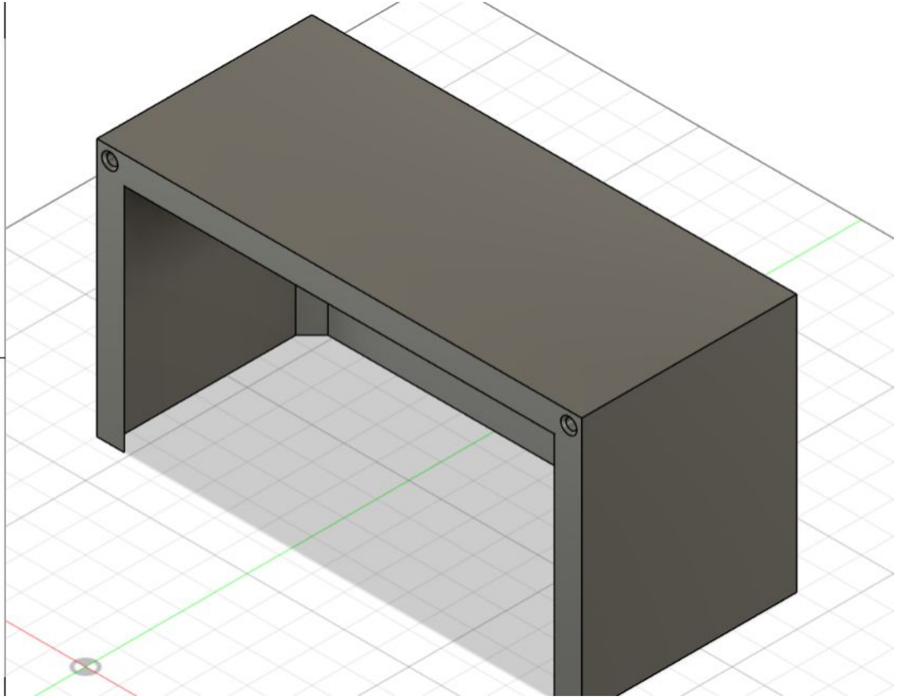




Top Piece



7.3
6.65
2X Ø.17 ⩔ .78
Ø.25
4
3.8
3.58
2X Ø.1 ⩔ 1
⌴Ø.24 ⩔ .13
7.5
3.32
6.9
2X Ø.17 ⩔ .78
Ø.25
7.1



Code

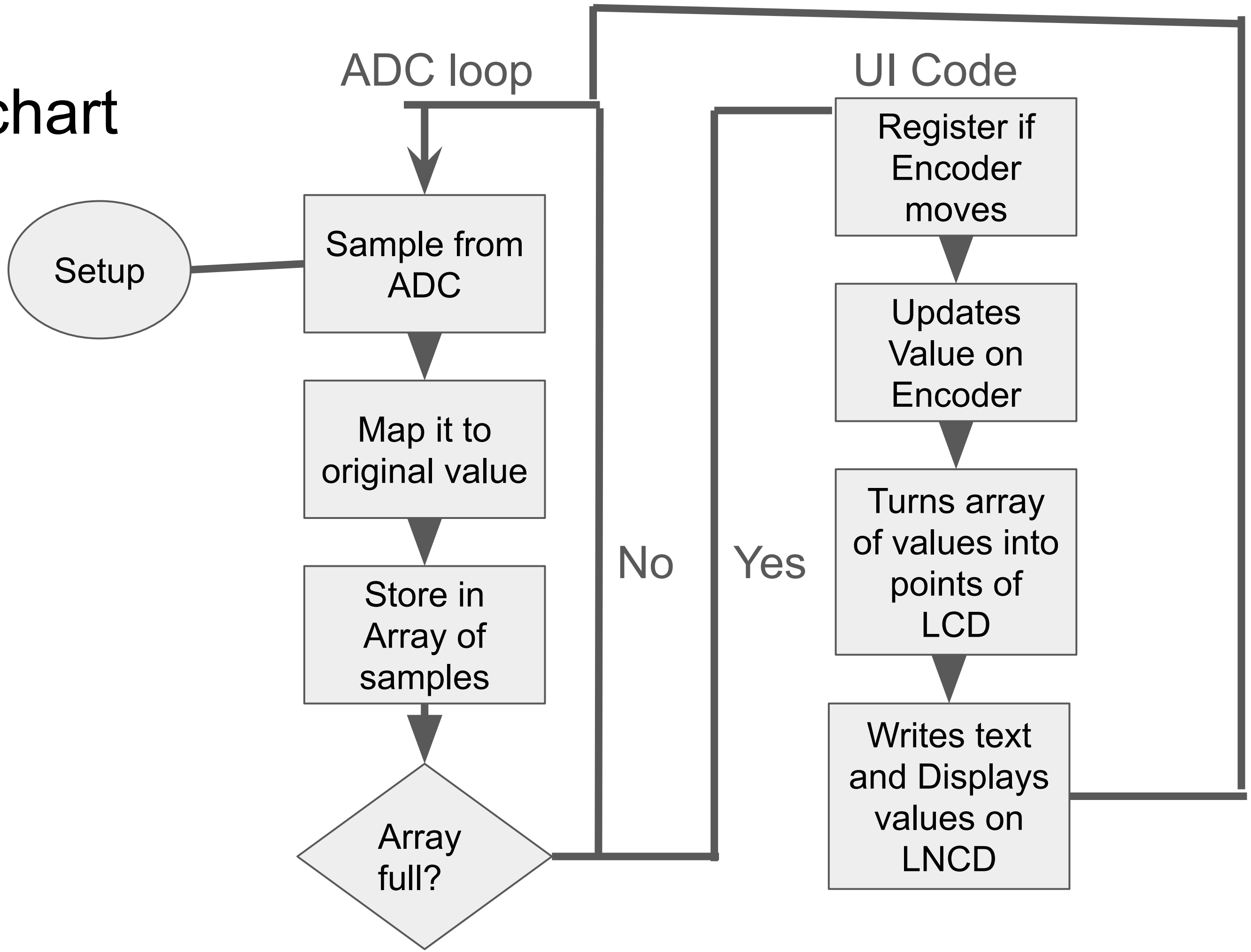
Main Parts:

- **Receiving signals from ADC to sample a high frequency**

- **Receiving User Input from encoder**

- **Coding a User Output on the LCD**

Code Flowchart



Code: Sampling

```
void loop() {
  // DECISION MATRIX POLLING
  if (! ss.digitalRead(SS_SWITCH)) {
    mode += 1 ;
    if (mode > 2){
      mode = 0;
    }
    Serial.print("Mode ");
    Serial.println(mode);
    refresh_screen();
    delay(200);
    if (! ss.digitalRead(SS_SWITCH)) {
      delay(800);
      if (! ss.digitalRead(SS_SWITCH)) {
        trigger();
      }
    }
  }
  int32_t new_position = ss.getEncoderPosition();
  // did we move around
```

```
void gather_samples() {
  //elapsedMicros timer = 0;
  //int x_interval = 5 + mode_values[0];  // time scale micro seconds sampling at 200kHz

  for (int i = 0; i < SAMPLE_COUNT; i++) {
    while (timer < interval_us);  // hold loop until time passed
    timer -= interval;

    ch1_samples[i] = (6.66666*(adc->analogRead(A0, ADC_0))-10);  // Channel 1
    ch2_samples[i] = (6.66666*(adc->analogRead(A1, ADC_1))-10);  // Channel 2
  }
```

Code: User Input

```
void trigger(){
  Trigger = !(Trigger && Trigger);
}

void change_val(int& m, bool increase) {
  if (mode == 0 || mode == 1) { // X or Y scaling
    if (increase) m++;
    else m--;
    m = constrain(m, -10, 10);
  } else if (mode == 2) { // Channel select
    if (increase) m++;
    else m--;
    m = constrain(m, 0, 2);
  }
}
```

```
if (encoder_position > new_position) {
  encoder_position = new_position;
  change_val(mode_values[mode], false);
  Serial.println(mode_values[mode]);
  refresh_screen();
}
else if (encoder_position < new_position){
  encoder_position = new_position;
  change_val(mode_values[mode], true);
  Serial.println(mode_values[mode]);
  refresh_screen();
}
```

Code: User Output

```
void refresh_screen(){

    tft.fillScreen(ILI9341_WHITE);
    tft.setTextColor(ILI9341_BLACK); tft.setTextSize(1);
    tft.setCursor(0, 0);
    tft.print("Mode: ");
    if(mode == 0) tft.println("X Scale");
    else if (mode == 1) tft.println("Y Scale");
    else if (mode == 2) tft.println("Channel Select:");

    tft.print("X_Scale: ");
    tft.print(mode_values[0]);
    tft.print("| Y Scale:");
    tft.print(mode_values[1]);
    tft.print("| Channel ");
    if (mode_values[2] == 2) tft.println("Both");
    else tft.println(mode_values[2]+1);


}
```

```
void draw_sine_wave(float amplitude, float frequency, uint16_t color) {
    tft.fillScreen(ILI9341_WHITE);
    tft.setTextColor(ILI9341_BLACK); tft.setTextSize(1);
    tft.setCursor(0, 0);
    tft.print("Mode: ");
    if(mode == 0) tft.println("X Scale");
    else if (mode == 1) tft.println("Y Scale");
    else if (mode == 2) tft.println("Channel Select:");

    tft.print("X_Scale: ");
    tft.print(mode_values[0]);
    tft.print("| Y Scale:");
    tft.print(mode_values[1]);
    tft.print("| Channel ");
    if (mode_values[2] == 2) tft.println("Both");
    else tft.println(mode_values[2]+1);
    tft.print("Trigger: ");
    if (Trigger) tft.println("On");
    else tft.println("Off");
    // Draw midline
    tft.drawLine(0, SLY / 2, SLX, SLY / 2, ILI9341_LIGHTGREY);
```

Closing

Achieved:

- Safe ±10 V input range
- 200 kHz sampling per channel
- Real-time LCD display with intuitive control
- Fully modular and battery-powered design

Built completely from scratch with protoboards, firmware, and 3D printing
Future improvements:

- Custom PCB for better layout and durability
- Higher-resolution display or waveform features (e.g., FFT mode)
- Expansion to 4+ channels or wireless display integration

Demonstrated that lab-grade tools can be made compact, affordable, and user-friendly