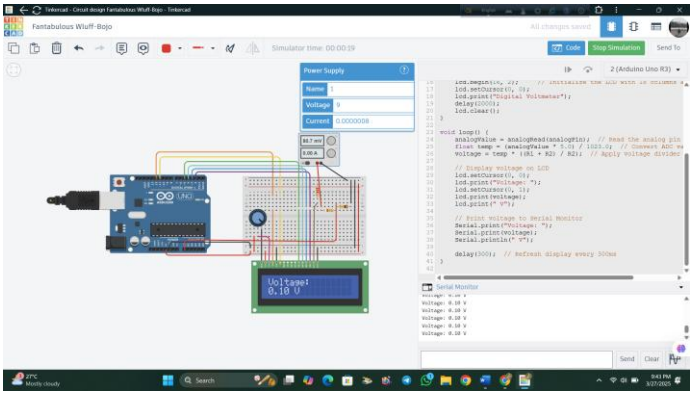
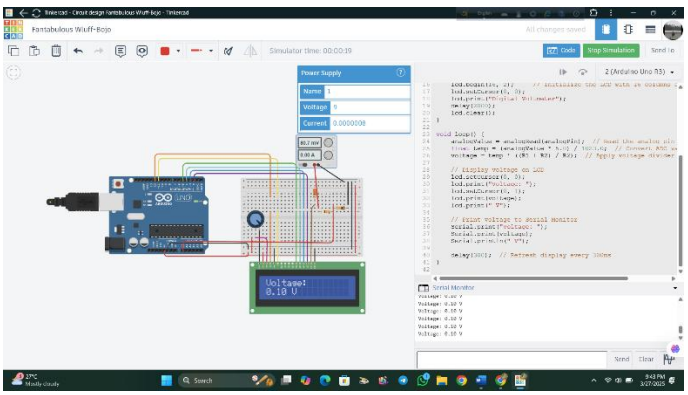
Digital Display & Voltmeter using Arduino

1. **Introduction to Voltmeters**
    - Voltmeters measure voltage or potential difference in a circuit.
    - Two types: **Analog** (pointer-based) and **Digital** (numerical display).
    - Digital voltmeters have higher accuracy and minimal error (<1%).
2. **Project Overview**
    - An **Arduino Uno-based digital voltmeter** capable of measuring up to **50V** is designed.
    - A **16x2 LCD display** is used to show the voltage.
3. **Hardware Components**:
    - Arduino Uno (ATmega328P)
    - 16x2 LCD Display
    - 90.9kΩ & 10kΩ Resistors (Voltage Divider)
    - 250kΩ Potentiometer
    - 220Ω Resistor (LCD Current Limiter)
    - Breadboard & Jumper Wires
4. **Circuit Connections**
    - **Arduino Uno (ATmega328P)** is used.
    - **16x2 LCD** connected with control and data pins.
    - **Voltage divider circuit** (with **90.9kΩ and 10kΩ resistors**) reduces the input voltage to a measurable level.
    - A **potentiometer** adjusts LCD brightness.
    - **5V power supply** is distributed through a **breadboard**.





5. **Voltage Divider Calculation**
    - Since Arduino can measure up to **5V**, a **voltage divider** is used to scale down **50V**.
    - Formula used:

$R_1/R_2>(V_{max}/V_{ref})$
Where V_{max}=50V , V_{ref}=5V

- Chosen values: **R1 = 90.9kΩ**, **R2 = 10kΩ** (Ratio ≈10).

6. Code:

Code -1::

```
#include <LiquidCrystal.h>

int seconds = 0;

LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd_1.begin(16, 2); // Set up the number of columns and rows on the LCD.

  // Print a message to the LCD.
  lcd_1.print("hello world!");
}

void loop()
{
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting
  // begins with 0):
  lcd_1.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd_1.print(seconds);
  delay(1000); // Wait for 1000 millisecond(s)
  seconds += 1;
}
```

Code-2::

```
#include <LiquidCrystal.h>

// Define LCD pins connected to Arduino
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Define resistor values for voltage divider
#define R1 90900  // 90.9kΩ resistor
#define R2 10000  // 10kΩ resistor

int analogPin = A0;  // Pin where voltage is measured
float analogValue = 0;
float voltage = 0;

void setup() {
   Serial.begin(9600);   // Start Serial Monitor with baud rate 9600
```

```
  lcd.begin(16, 2);     // Initialize the LCD with 16 columns and 2 rows
  lcd.setCursor(0, 0);
  lcd.print("Digital Voltmeter");
  delay(2000);
  lcd.clear();
}

void loop() {
  analogValue = analogRead(analogPin);  // Read the analog pin value
  float temp = (analogValue * 5.0) / 1023.0;  // Convert ADC value to voltage
  voltage = temp * ((R1 + R2) / R2);  // Apply voltage divider formula

  // Display voltage on LCD
  lcd.setCursor(0, 0);
  lcd.print("Voltage: ");
  lcd.setCursor(0, 1);
  lcd.print(voltage);
  lcd.print(" V");

  // Print voltage to Serial Monitor
  Serial.print("Voltage: ");
  Serial.print(voltage);
  Serial.println(" V");

  delay(300);  // Refresh display every 300ms
}
```

7. **Arduino Code Explanation**
   - **LiquidCrystal.h** library used for LCD interfacing.
   - **ADC (Analog-to-Digital Conversion)** reads voltage from **A0 pin**.
   - Formula applied to convert ADC value to actual voltage.
   - Voltage values are displayed on both **LCD and Serial Monitor**.
   - **Updates every 300ms** to reflect real-time voltage changes.
8. **Code Overview**:
   - Uses **ADC conversion** to measure input voltage.
   - Displays voltage on **LCD** and **Serial Monitor**.
   - Updates values every **300ms** for real-time accuracy.
9. **Simulation & Testing**
   - The **measured voltage** is displayed on the LCD.
   - Comparison with **actual power supply voltage** shows **high accuracy**.
   - Changing input voltage updates the display in real-time.

10. **Accuracy & Performance**:
    - Measures voltage **up to 50V** with minimal error.
    - Real-time voltage changes reflected instantly.
11. **Brief:**
    - The **Arduino-based voltmeter** provides an accurate voltage reading.
    - It can be integrated into **various electronics projects**.
    - Encourages viewers to **experiment and customize the circuit**.

12. Conclusion:

This project highlights the **importance of software in defining the functionality of hardware**. By keeping the same physical components, Arduino, LCD, and supporting circuitry—we can achieve **different functionalities** just by modifying the code.

When we upload a simple **"Hello, World!"** code, the LCD acts as a basic display, showing a static text message.

When we upload the **digital voltmeter** code, the LCD dynamically displays real-time voltage readings from the circuit.

This demonstrates how **Arduino and microcontrollers provide flexibility in embedded systems**, where a single hardware setup can serve multiple purposes with different programs. It also reinforces the idea that **software is key in controlling hardware behavior**, making Arduino a powerful tool for learning and developing real-world applications in electronics and automation.