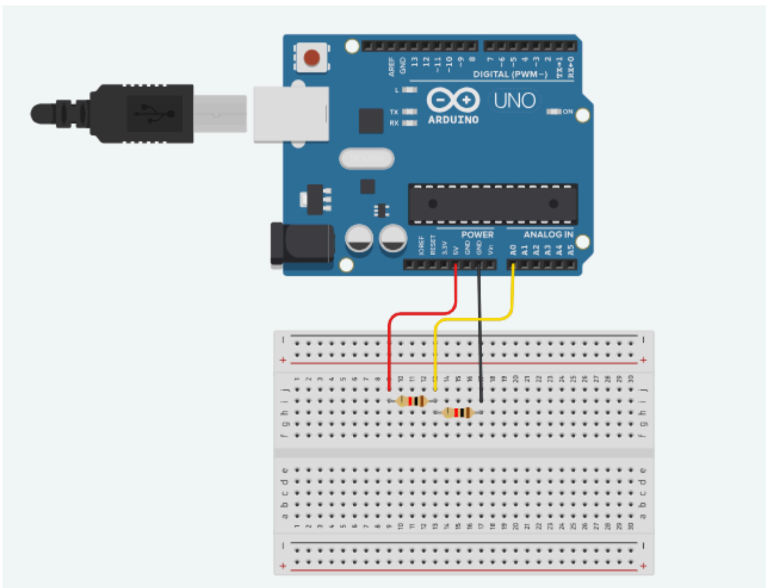
Project Title: Smart Multimeter Using Microcontroller System
Assignment:1
Student Name: Vanshika Khandelwal
Roll Number:241141
Department:EE
Date:14/12/25

#Introduction:

This project seeks to build a simple monitoring system with an Arduino that helps grasp at the principles under which digital instruments measure electrical values. Experiment Goals: The goal of this experiment is to use basic theory of voltage dividers, RC time constants, and Ohm's laws to simulate measurements of several voltages, capacitance, and resistance.

Wiring Practical circuits are developed to transform analog electric signals into digital values using Tinkercad and the Arduino's ADC. These experiments illustrate the basic functionality of a common digital multimeter and show physical phenomena including ADC resolution, component variation, and measurement error.

Task A — Voltage Divider Analysis and Measurement Module:




```
// C++ code
//
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int adc=analogRead(A0);
  float voltage=adc*(5.0/1023.0);
  Serial.print("Voltage: ");
  Serial.println(voltage);
  delay(1000);
}
```

For R1=R2=10 kΩ

Serial Monitor

```
Voltage: 2.50
Voltage: 2.50
Voltage: 2.50
Voltage: 2.50
```

For R1=4.7kΩ & R2=10kΩ

Serial Monitor

```
Voltage: 3.40
Voltage: 3.40
Voltage: 3.40
Voltage: 3.40
Voltage: 3.40
Voltage: 3.40
Voltage: 3.40
```

For R1=1kΩ &R2=15kΩ

Serial Monitor

```
Voltage: 4.69
Voltage: 4.69
Voltage: 4.69
Voltage: 4.69
Voltage: 4.69
Voltage: 4.69
Voltage: 4.69
```

-ADC to Voltage Conversion Formula:(10 bits)

$$V=ADC*(5/1023)$$

- Voltage Divider Formula:

$$V_{out} = V_{in}*[R2/(R1+R2)]$$

Resistance values	Theoretical	Tinkercad Measured
R1=10kΩ R2=10kΩ	2.5 V	2.5 V
R1=4.7kΩ R2=10kΩ	3.4013 V	3.4 V
R1=1kΩ R2=15kΩ	4.6875 V	4.69 V

Measured output voltages closely match the theoretical values calculated using the voltage-divider formula. Small deviations are observed due to the Arduino's 10-bit ADC resolution, electrical noise, and resistor tolerances etc.

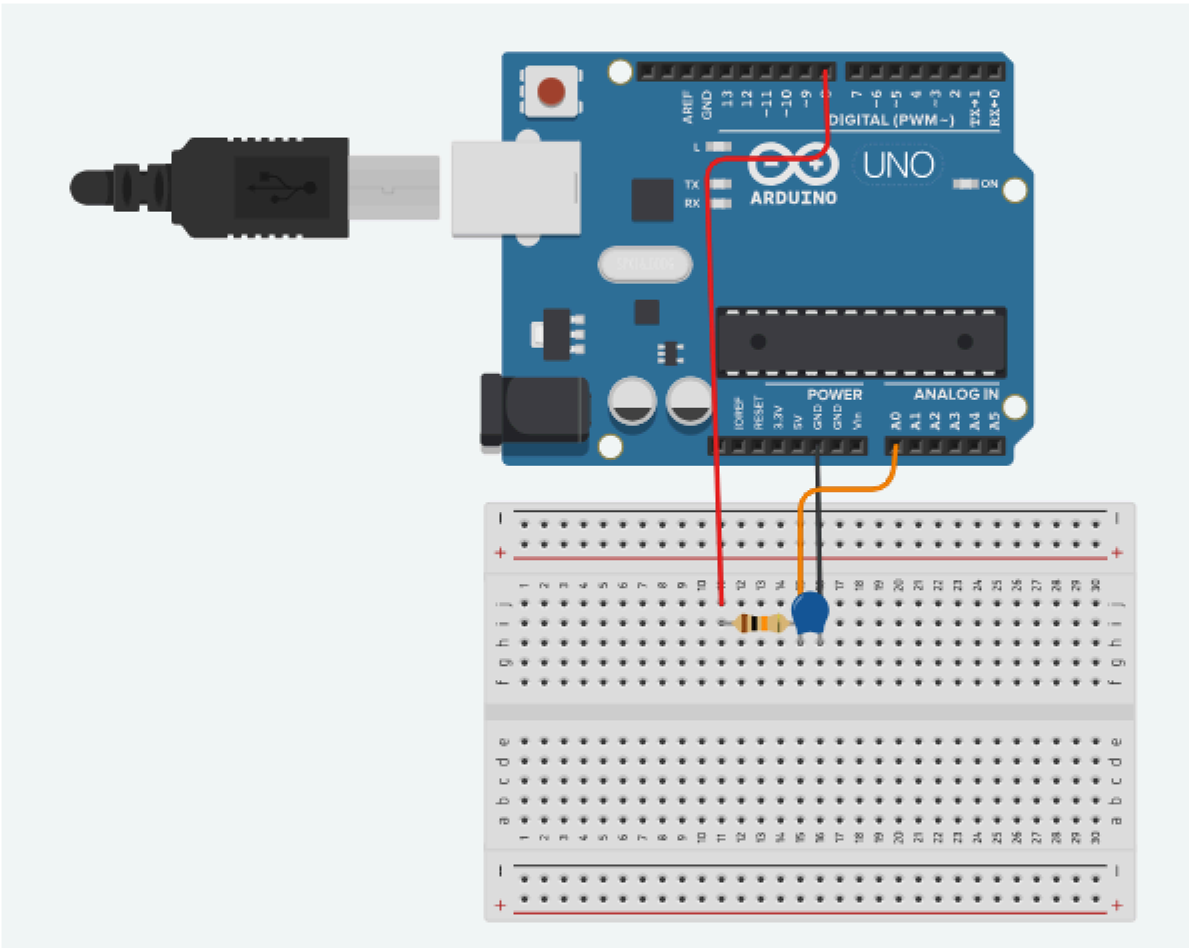
Purpose of a voltage divider in measurement systems :

→ A voltage divider reduces a higher input voltage to a safe level so that it can be accurately measured by the Arduino's ADC.

Observations :

→ Small fluctuations were observed in the measured voltage due to ADC quantization error, electrical noise and minor variation in resistor values.

Task 2— Capacitance Measurement Using RC Time Constant:



```
const int chargePin=8;
const int analogPin=A0;
const float R=10000.0;

void setup(){
  Serial.begin(9600);
  pinMode(chargePin,OUTPUT);

  digitalWrite(chargePin,LOW);
  delay(2000);

  digitalWrite(chargePin,HIGH);
  unsigned long startTime = millis();

  while(analogRead(A0)<0.63*1023){
    // wait
  }

  unsigned long elapsed=millis()-startTime;
  float C=elapsed/R;   // in millisecond/ohm
   Serial.print("Time(ms): ");
  Serial.println(elapsed);
  Serial.print("Capacitance (approx): ");
  Serial.println(C);
}

void loop() {}
```

For C=10 μF = 0.01 ms/Ω

Serial Monitor

```
Time(ms): 101
Capacitance (approx): 0.01
```

For C=100 μF = 0.1 ms/Ω

Serial Monitor

```
Time(ms): 1000
Capacitance (approx): 0.10
```

Capacitance Value	Measured Time	Expected Time(RC)
10 µF	101 ms	100 ms
100 µF	1000 ms	1000 ms

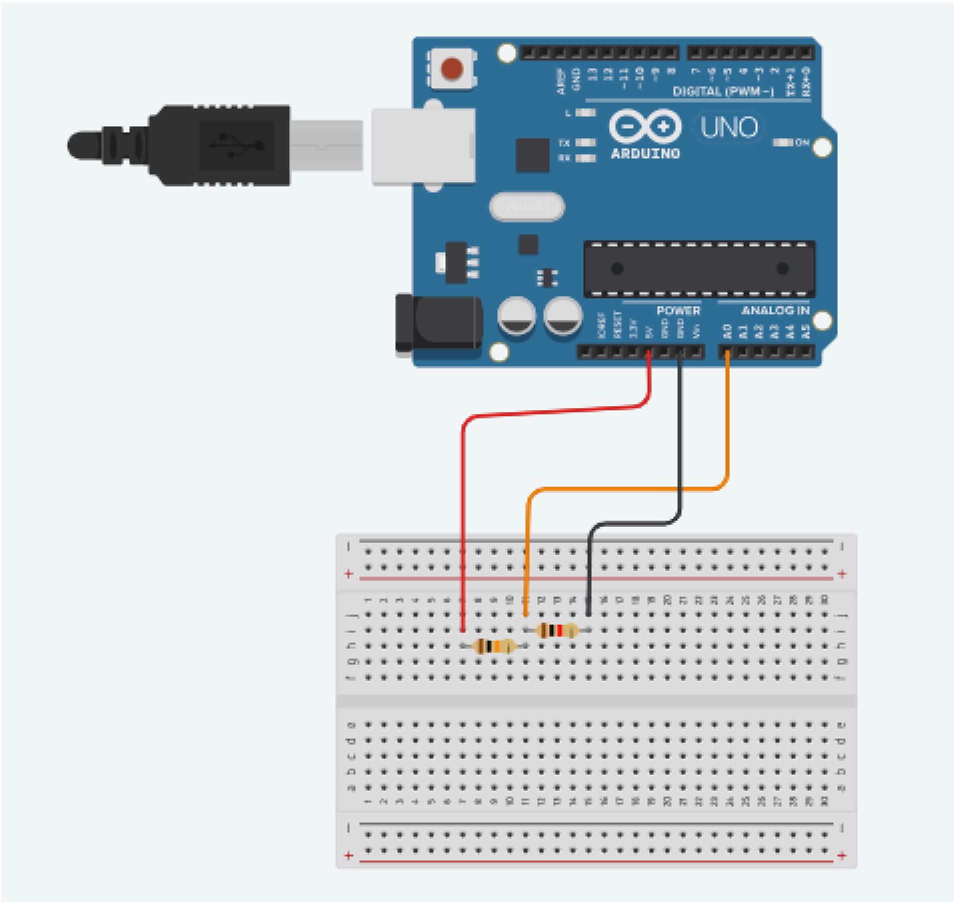
① The RC time constant ($\tau = RC$) is the time required for a capacitor to charge to approx 63% of the supply voltage. This comes from the exponential charging law.

$$V_c(t) = V\left(1 - e^{-\frac{t}{RC}}\right)$$

② Error Source:

Error arise due to resistor & capacitor tolerance, electrical noise, ADC resolution & time limitations of millis().

#Task C — Beginner Ohmmeter Prototype:



```cpp
// C++ code
const int analogPin = A0;
const float R1 = 10000.0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int adc = analogRead(analogPin);
  float Vout = adc * (5.0 / 1023.0);

  float Rx = R1 * (Vout / (5.0 - Vout));

  Serial.print("Measured Resistance (ohm): ");
  Serial.println(Rx);

  delay(1000);
}

```

For R1=10k**Ω**,Rx=Unknown:

Serial Monitor

```
Measured Resistance (ohm): 1000.00
Measured Resistance (ohm): 1000.00
Measured Resistance (ohm): 1000.00
Measured Resistance (ohm): 1000.00
Measured Resistance (ohm): 1000.00
Measured Resistance (ohm): 1000.00
```

For R1=10k**Ω**,Rx=Unknown:

Serial Monitor

```
Measured Resistance (ohm): 11006.16
Measured Resistance (ohm): 11006.16
Measured Resistance (ohm): 11006.16
Measured Resistance (ohm): 11006.16
Measured Resistance (ohm): 11006.16
Measured Resistance (ohm): 11006.16
Measured Resistance (ohm): 11006.16
```

For R1=10k**Ω**,Rx=Unknown:

Serial Monitor

```
Measured Resistance (ohm): 4208.33
Measured Resistance (ohm): 4208.33
Measured Resistance (ohm): 4208.33
Measured Resistance (ohm): 4208.33
Measured Resistance (ohm): 4208.33
Measured Resistance (ohm): 4208.33
Measured Resistance (ohm): 4208.33
```

Known R1	Unknown Rx (Actual)	Unknown Rx (Measured)
10k**Ω**	1k**Ω**	1k**Ω**
10k**Ω**	11k**Ω**	11.00616 k**Ω**
10k**Ω**	4.2k**Ω**	4.20833 k**Ω**

① Step-by-Step Calculation:

The Arduino first converts the ADC reading into voltage using ADC formula.

ADC to Voltage conversion formula:

$$V_{out} = ADC \times \left(\frac{5}{1023}\right)$$

The unknown resistance is then calculated by rearranging the voltage divider eq.

$$V_{out} = V_{in}\left(\frac{R_x}{R_1 + R_x}\right)$$

$$R_x = R_1 \times \left(\frac{V_{out}}{V_{in} - V_{out}}\right)$$

② Measurement Uncertainty:

Small errors occur due to resistor tolerance, ADC resolution, electrical noise & reference voltage variation.