Smart Multimetre Ass1

Tarun Goyal

December 14, 2025

1 Task A: Voltage Divider Analysis and Measurement Module

1.1 Voltage Divider

The main purpose of using the voltage divider circuit is that the ADC of the arduino can only read the voltages in the range of 0V to 5V but the input voltage can be out of this range. So, we use the voltage divider circuit to convert that high voltage into the safe measurable voltage range using the formula:

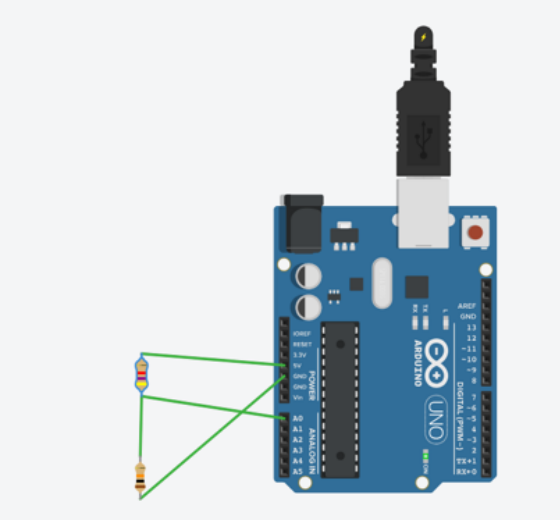
$$V_{\text{out}} = V_{\text{in}} \times \frac{R_2}{R_1 + R_2}$$

Now to convert the ADC reading into voltage we know that the voltage value will lie in 0-5 and the analog ADC reading will go from 0-1023 so the simple formula to convert the ADC reading into volatge is:

$$V = \text{ADC} \times \frac{5.0}{1023.0}$$

1.2 Screenshots showing the circuit, code, and output readings

1.2.1 Circuit



Circuit Description:
This is the circuit diagram of the voltage divider used for Task A. The input voltage is 5V from the Arduino, and the divided voltage at the node between R1 and R2 is measured using analog pin A0.

Figure 1: Task A – Circuit and Description

1.2.2 Code

Code Description:
This code reads the analog value from A0 using `analogRead()`, converts it to voltage using the formula $V = \text{ADC} \times (5.0/1023)$, and prints the measured voltage to the Serial Monitor.

Figure 2: Task A – Code and Description

1.2.3 Output Values

Vin (V)	R1 (kΩ)	R2 (kΩ)	Vout (Theoretical) (V)	Vout (Experimental) (V)
5.0	10	10	2.5000	2.4976
5.0	4.7	10	3.4013	3.4018
5.0	1	15	4.6875	4.6872

Table 1: Comparison of Theoretical and Experimental Vout Values for Task A

1.3 Observations: Error, Noise, and Unexpected Results

The experimental value is not exactly same as the theoretical one. There are some deviations. There are a number of factors which contribute to this error:

1) The resistances used in the circuit are not ideal and have a general error of 1 to 5 %.

2) The arduino uses a 10-bit ADC, which divides the range 0-5 V into 1024 steps which is 4.88mV per step which can result in some roudning off error.

3)There can be some noise from the USB port and ADC sampling which can cause slight error in readings of different measurements.

2 Task B: Capacitance Measurement Using RC Time Constant

2.1 RC Time Constant and the Significance of 63%

The RC time constant (denoted as τ, tau) is defined as:

$$\tau = R \times C$$

It is basically a method of representation of the amount of time it takes for a capacitor to charge or discharge through a resistor. The charging of capacitor form a Voltage source follows an exponential curve represented by the formula:

$$V(t) = V_{\text{in}} \left(1 - e^{-t/(RC)}\right)$$

Now, according to this formula after one time constant (τ), the capacitor charges up to approximately 63% of the input voltage. Mathematically:

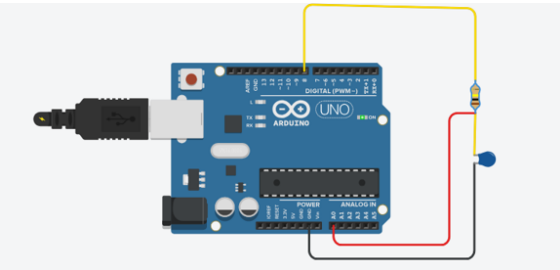
Substituting $t = RC$:

$$V(RC) = V_{\text{in}} \left(1 - e^{-1}\right) \approx 0.632 \times V_{\text{in}}$$

So at $t = \tau$, the capacitor has reached 63.2% of its final value.
This value is significant because:

- It is a standard reference point used to describe how fast an RC circuit responds.
- It allows us to calculate capacitance using measured time, as we can see when it reaches 63 % and that time will be the exactly one time constant.
- It is independent of the input voltage and depends only on R and C.

2.2 Circuit



Circuit Description:
This is the circuit diagram of the DC RC circuit used for Task A. The input voltage is taken through digital pin 8 from the Arduino. When the D8 is HIGH it acts as 5V resulting in charging and when the D8 is LOW it acts as 0V resulting in discahrging.

Figure 3: Task A – Circuit and Description

2.3 Code

```
const int pin = A0;
void setup()
{
  Serial.begin(9600);
  pinMode(Volt,OUTPUT);
  digitalWrite(Volt,LOW);
  delay(10);
}
void loop()
{
  pinMode(Volt,OUTPUT);
  digitalWrite(Volt,LOW);
  delay(100);
  digitalWrite(Volt,HIGH);
  unsigned long starttime = micros();
  float val = analogRead(pin);

  while(val < threshold) {
    val = analogRead(pin);
    if ((micros() - starttime) > 2000000UL) break;
  }
  unsigned long elapsed_time_us = micros() - starttime;
  const float R_Value_Ohm = 100000.0;
  float time_sec = elapsed_time_us/ 1e6;
  float C_Value_nF = (time_sec*1e9)/R_Value_Ohm;

  Serial.print("ADC reading: "); Serial.print(val);
  Serial.print("  Time(us): "); Serial.print(elapsed_time_us);
  Serial.print("  C = "); Serial.print(C_Value_nF, 3); Serial.pr

  pinMode(Volt,OUTPUT);
  digitalWrite(Volt,LOW);
  delay(500);
```

Code Description:
Firstly we set the D8 LOW for some time to ensure that the capacitor is unchared before beginning the charging experiment. Then we set the D8 HIGH to start the charging and use the micros() function to measure the exact time at which the charging starts. Then we wait until the analog value reaches the threshold of the 63%. When it reaches the threshold we again capture the exact time using micros() and calculate the time elasped which gives us the T_{63}. Tehn by dividing this T_{63} by R gives us the Capacitance.

Figure 4: Task A – Code and Description

2.4 Table showing the measured time vs. expected time and value of Capacitance

T_{63} Expected (us)	T_{63} Experimental (us)	$C_{original}$ (nF)	$C_{experimental}$ (nF)	R Used (KΩ)
10000	10028	100	100.28	100
10000	1056	10	10.56	100
700	728	70	72.8	10
56000	56060	700	700.75	80

Table 2: Comparison of Expected and Experimental T_{63} and Capacitance Values for Task B

2.5 potential error sources

There are several factors that can cause differences between the expected T_{63} values and the experimental measurements:

- **Resistor and Capacitor Tolerance:** The resistor and capacitance are not ideal. So the actual values may not match the values written on them. So, a tolerance of 5% or 10% can change the RC time constant and the measured T_{63}.
- **ADC Resolution:** The Arduino uses a 10-bit ADC, which means it can only represent voltages in 1024 discrete steps. This limited resolution can cause slight errors in detecting the exact moment the capacitor reaches 63% of the supply voltage.
- **Electrical Noise:** There can be some noise from the USB power supply, breadboard connections, and surrounding components which can cause the analog voltage readings to vary and affect the measured T_{63}.
- **Incomplete Discharge:** If the capacitor is not fully discharged before a new measurement, the starting voltage will not be zero, which will result in shorter charging time than expected.
- **Timing Inaccuracy:** The `micros()` function has its own resolution and may introduce small timing errors during fast charge cycles.

3 Task C: Beginner Ohmmeter Prototype

Step-by-Step Calculations

I am showing the calculation by taking an example.

Example:

Reference resistor: $R_{\text{ref}} = 10\,\text{k}\Omega$
ADC reading: 512

1. **Read the ADC value**
   The Arduino gives the raw analog value between 0 and 1023.
   $$ADC = 512$$
2. **Convert ADC reading to voltage**
   $$V_{\text{out}} = ADC \times \frac{5}{1023}$$
   Substitute:
   $$V_{\text{out}} = 512 \times \frac{5}{1023}$$
   $$V_{\text{out}} \approx 2.50\ \text{V}$$

3. **For the Rx we are using the voltage divider formula:**

$$V_{\text{out}} = 5 \times \frac{R_x}{R_{\text{ref}} + R_x}$$

The rearranged formula is:

$$R_x = R_{\text{ref}} \times \frac{V_{\text{out}}}{5 - V_{\text{out}}}$$

4. **Substitute the values**

$$R_x = 10000 \times \frac{2.50}{5 - 2.50}$$

$$R_x = 10000 \times \frac{2.50}{2.50}$$

5. **Final calculation**

$$R_x = 10000\ \Omega$$

So the unknown resistor value is:

$$R_x = 10\ \text{k}\Omega$$

3.1 Code and Circuit

the circuit is the same as the cirucit used in task A and the code is also almost the same we are just rearranging the equation of voltage divider to find the unknown resisntace in place of Vout.

3.2 Comparison table: Actual vs. Measured values.

R_ref ()	**R_Unknown_original ()**	**R_Unknown_experimental ()**
10000	10000	9980.469
5000	5000	4990.234
5000	100000	99387.75
5000	75000	74921

Table 3: Comparison of Actual vs. Measured Values for Task C

3.3 Reflection on Measurement Uncertainty

There are small differences between the actual resistor values and the measured values. These causes of these differences are similar to the other two tasks before.The reference resistor and the unknown resistor are not ideal. They have a general tolerance os 1% to 5% which cause error in the measured value of resistance. The Arduino ADC also has limited resolution, which means the voltage is quantized and cannot be measured perfectly. Noise in the breadboard connections and power supply can also affect the ADC reading slightly. Because of these factors, the measured resistance values are close to the actual values but not exactly the same.

NOTE- the error sources for all the tasks are basically the same. The non ideality of the components. The ADC resolution and the noise from the components and the surrounding.