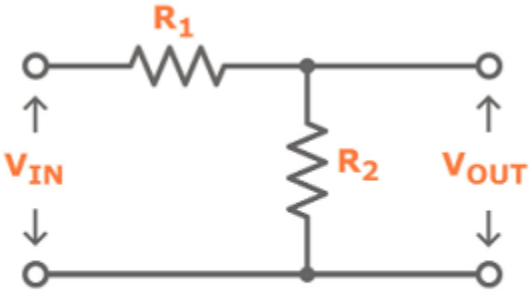
Assignment-1 (Multimeter)

-Shivansh Goyal (Roll no: 251026)

Task A: Voltage Divider

A voltage divider is used to produce a smaller voltage from a large voltage. It can also be used to measure the value of unknown resistances.

The following is the voltage divider circuit:



$$V_{OUT} = \left(\frac{R_2}{R_1 + R_2}\right) V_{IN}$$

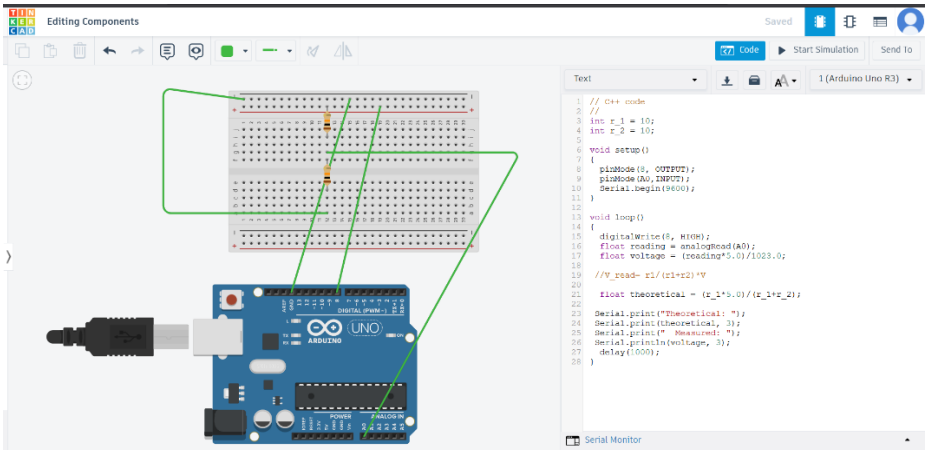
ADC reading to voltage conversion is done by:
voltage =(reading/1023) * 5.00

Below is the tinkercad implementation of the voltage divider circuit. The Arduino code is also given along.

Three readings are taken as follows:

	Resistance 1(kohms)	Resistance 2(kohms)	Theoretical voltage	Practical voltage
Reading 1	10	10	2.5	2.493
Reading 2	7.30	2.30	3.802	3.783
Reading 3	6.20	4.80	2.818	2.805

Given below are the tinkercad circuits as well as the Arduino codes.



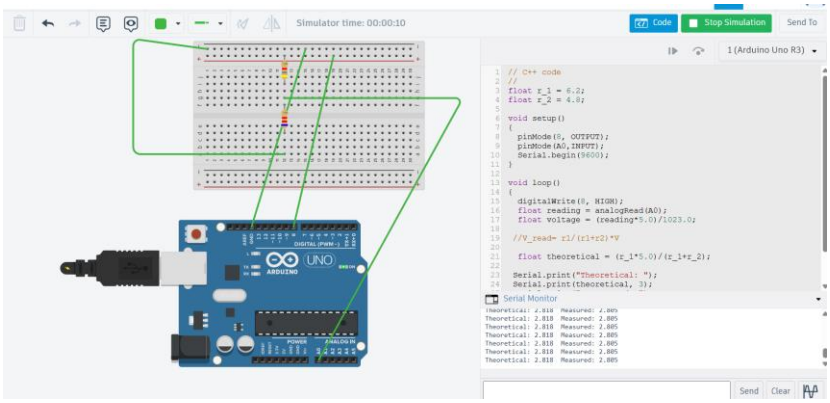
Serial Monitor

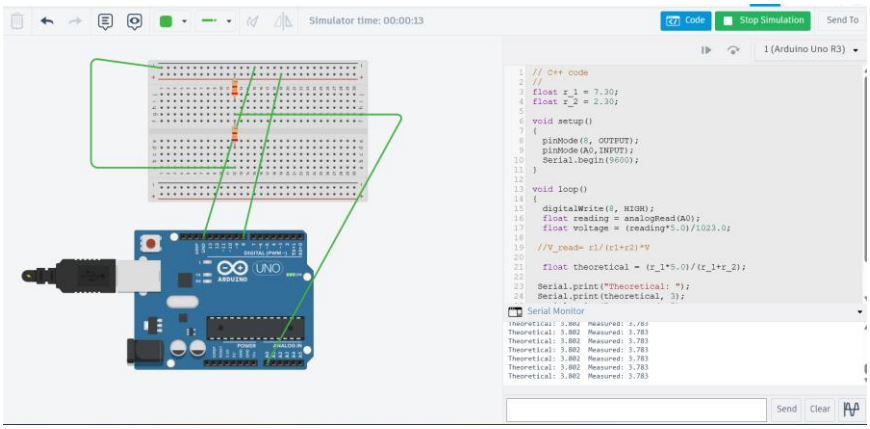
```
Theoretical: 2.500  Measured: 2.493
Theoretical: 2.500  Measured: 2.493
Theoretical: 2.500  Measured: 2.493
Theoretical: 2.500  Measured: 2.493
Theoretical: 2.500  Measured: 2.493
Theoretical: 2.500  Measured: 2.493
Theoretical: 2.500  Measured: 2.493
```

The code measures the output voltage in the analog pins on the Arduino board (pin A0).

The voltage to the circuit is provided by the digital pin 8. (the board itself is simulated to be powered by the USB port.)

The converts the analog ADC reading to voltage value and outputs it to the serial monitor. The serial monitor also outputs the theoretical voltage.





The error is for each reading is complied below:

	Theoretical voltage	Practical voltage	Error	Percentage error
Reading 1	2.5	2.493	0.007	0.28%
Reading 2	3.802	3.783	0.019	0.49%
Reading 3	2.818	2.805	0.013	0.46%

Sources for error are:

Supply voltage not exactly constant

- Arduino “5V” is typically **4.8–5.1 V**
- USB supply fluctuates with load
- Theory assumes **perfect 5.000 V**

ADC quantization error

- Arduino UNO ADC is **10-bit**
- Resolution:

$$\frac{5V}{1023} = 4.88 \text{ mV per step}$$

- Output is rounded to nearest step

Thermal noise and temperature effects

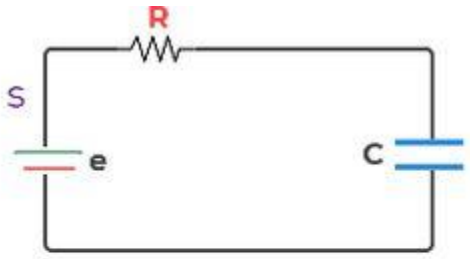
- Resistance changes with temperature
- Long measurements → self-heating
- Ambient temperature variation

The deviation between theoretical and practical voltage in a voltage divider arises due to non-ideal components, measurement limitations, and environmental effects. It is important to implement calibration and error analysis in real engineering systems.

Task 4.2

Capacitance measurement using RC time constant.

The capacitor resistor circuit is shown below.



$q = Ce\ (1 - e^{-t/RC})$

Capacitive Time Constant

$\tau = RC$ (dimensions of time)

$q = Ce\ (1 - e^{-RC/RC})$

$q = Ce\ (1 - e^{-1})$

$q = 0.63\ Ce$

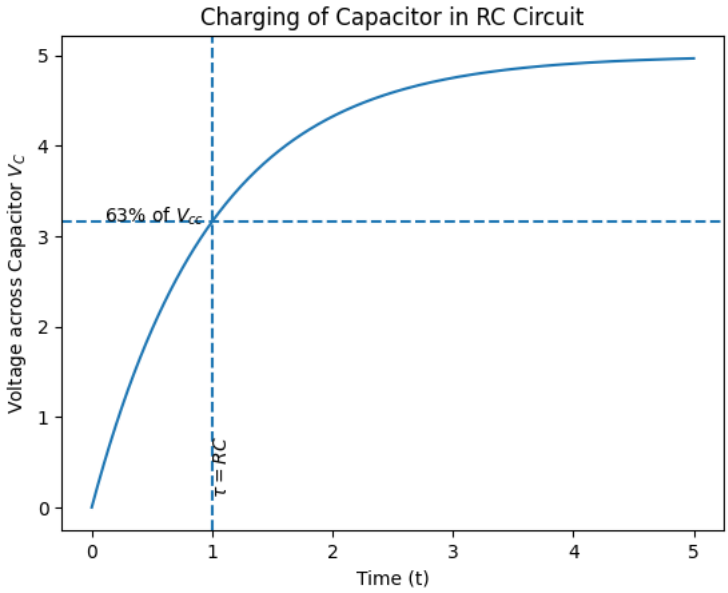
$qo = Ce$ *Since emf "e" is max potential difference across capacitor, Ce or qo is max charge*

We use this circuit for unknown capacitance measurement.

The potential difference across the capacitor is given as:

V= V0(1 – e^(-t/RC))

We wait for one time constant as the potential difference rises to 63% of its max value. The time needed for that is RC. Since the resistance is known, capacitance is calculated using C=t/ R. Shown below is the graph for charging of a capacitor. One time constant is marked on the graph.



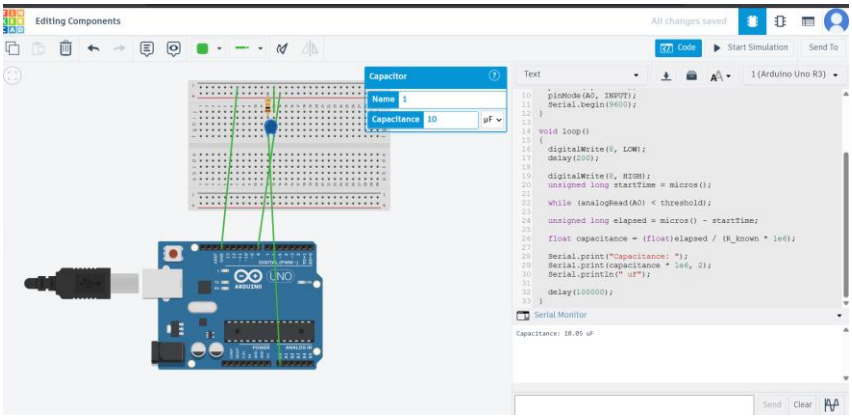
Given below is the code for the same. Threshold is 63% of max voltage. The resistance used is 10kohm.

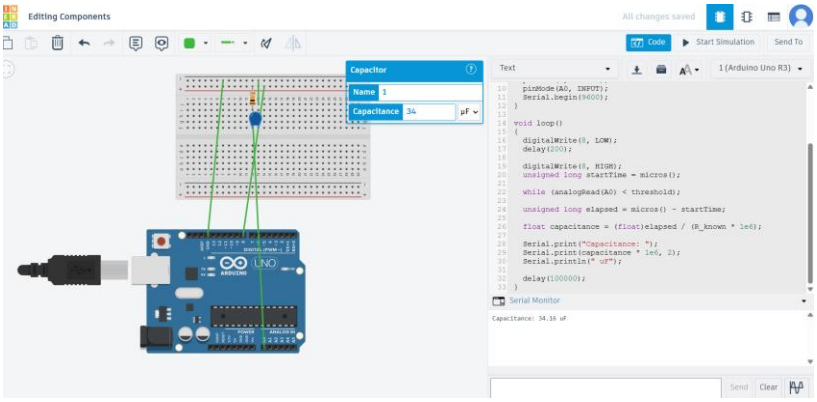
Once threshold id attained, the micros() function calculates one time constant in microseconds. the value of capacitance is calculated for each of the two cases listed after that.

Text 1 (Arduino Uno R3)

```
// C++ code
//

 float R_known = 10000.0;  // 10 kohm resistance
 int threshold = 647;      // 63.2% of 1023

void setup()
{
  pinMode(8, OUTPUT);
  pinMode(A0, INPUT);
  Serial.begin(9600);
}

void loop()
{
  pinMode(chargePin, OUTPUT);
  digitalWrite(8, LOW);
  delay(200);

  digitalWrite(8, HIGH);
  unsigned long startTime = micros();

  while (analogRead(A0) < threshold);

  unsigned long elapsed = micros() - startTime;

  float capacitance = (float)elapsed / (R_known * 1e6);

  Serial.print("Capacitance: ");
  Serial.print(capacitance * 1e6, 2);
  Serial.println(" uF");

  delay(1000);
}
```

Serial Monitor





The below table encapsulates the theoretical value of time constant, measured value, actual capacitance and measured capacitance:

	Measured time	Theoretical time	Actual capacitance	Measured capacitance.
Reading 1	0.1005 seconds	0.1 seconds	10 uF	10.05 uF
Reading 2	0.3416 seconds	0.34 seconds	34 uF	34.16 uF

The sources for error in this experiment are:

Resistor tolerance

- The known resistor Rhas tolerance (±1%, ±5%)
- Theory assumes exact resistance
- Error in Rdirectly causes proportional error in calculated C

$$\frac{\Delta C}{C} = \frac{\Delta R}{R}$$

ADC quantization error

- 10-bit ADC resolution ≈ 4.88 mV
- 63% threshold corresponds to a **range of ADC codes**
- Threshold crossing is not exact

ADC sampling and software delay

- analogRead() takes ~110 µs
- Loop and comparison overhead adds delay
- Timing error increases for **small capacitors**

Task 4.3:

Ohmmeter circuit using voltage divider.

This method uses the voltage divider circuit shown above to calculate the value of an unknown resistor using a known resistor. The known resistor in all three cases used in 5 kiloohm resistor.

Voltage divider equation:

$$V_{out} = V_{in} \times \frac{R_x}{R_{known} + R_x}$$

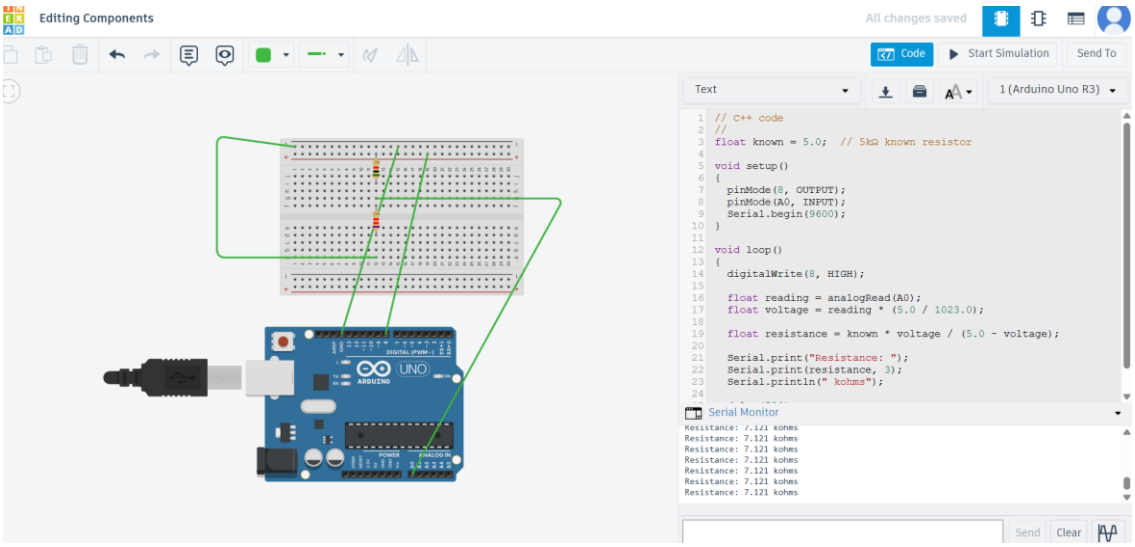
Solving for the unknown resistance R_x:

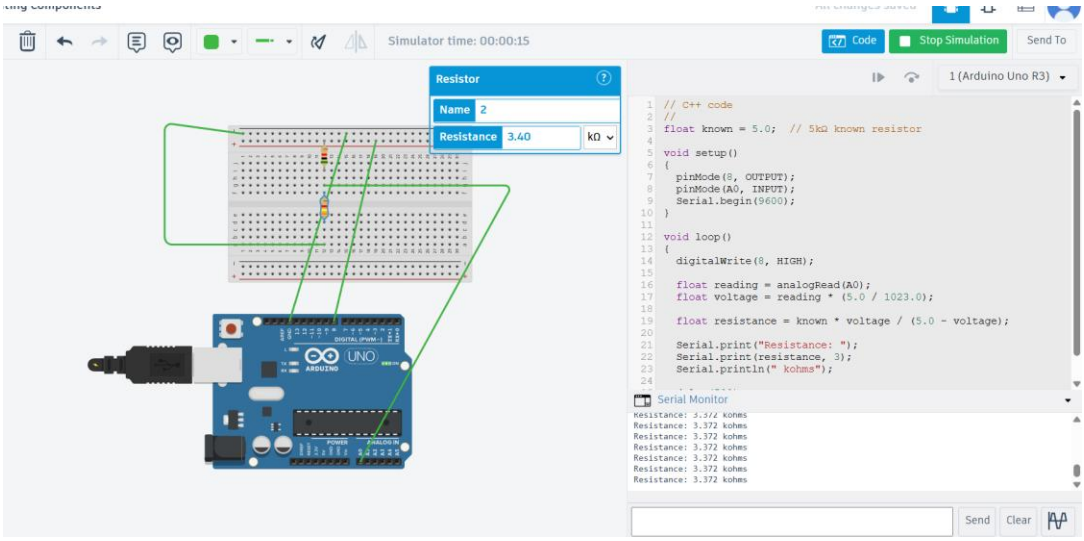
$$R_x = R_{known} \times \frac{V_{out}}{V_{in} - V_{out}}$$

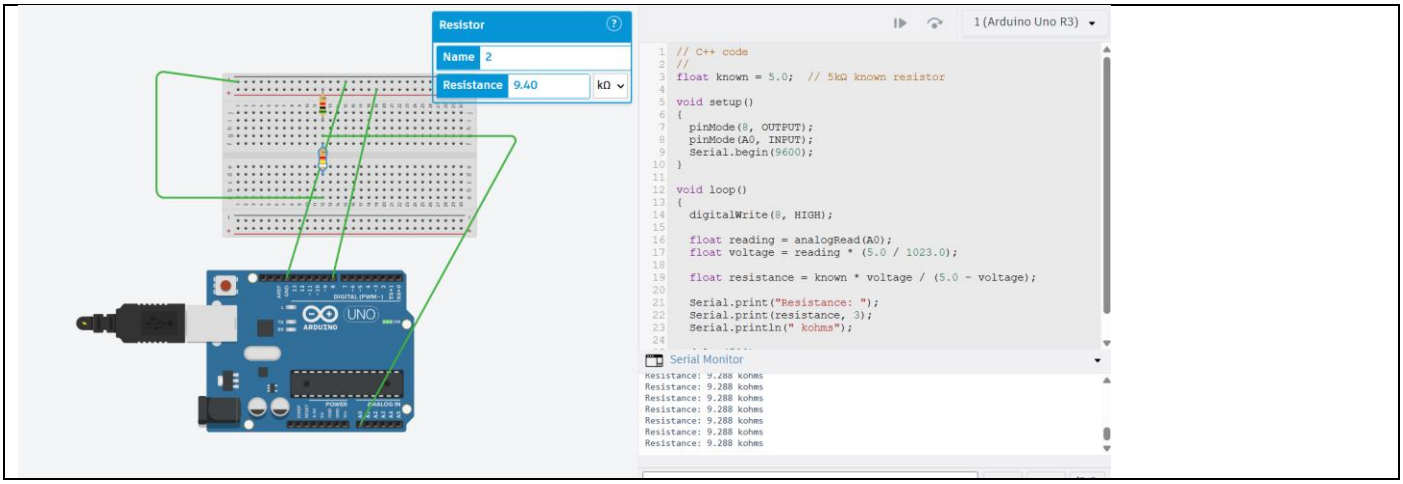
the Arduino code is given for all cases.

The voltage is read using the analog read function as in task 4.1

The value of resistance is displayed in the serial monitor.







	Actual resistance(kohms)	Measured resistance(kohms)	Error percentage
Reading 1	7.2	7.121	1.09%
Reading 2	3.4	3.372	0.82%
Reading 3	9.4	9.288	1.19%

For resistances far away from 5 kohms, the error percentage is somewhat large, close to 1%. The voltage divider setup works best for resistances close to each other.

Sources of error:

Tolerance of the known resistor

- The reference (known) resistor has a tolerance (±1%, ±5%)
- Theory assumes its value is exact
- Any error in the known resistor directly transfers to the measured resistance

Supply voltage variation

- Supply voltage (Arduino 5 V) is not perfectly constant
- USB supply fluctuates with load and time
- Divider output voltage scales with supply voltage

ADC quantization error

- Arduino ADC is 10-bit (0–1023)
- Voltage resolution ≈ 4.88 mV per step
- Measured voltage is rounded to nearest ADC count

Leads to step-like variation in calculated resistance

Noise and interference

- Electrical noise on ADC input

- USB noise from computer
- Switching noise from digital pins