1. Abstract

This project focuses on designing and implementing an ohmmeter using an Arduino Nano and an LCD display to measure the value of an unknown resistor. The core purpose of the project is to accurately determine the resistance of a component using the principles of a voltage divider circuit. The methodology involves connecting a known resistor in series with the unknown resistor, measuring the voltage drop across the unknown resistor using Arduino's analog input pins, and calculating its resistance using the voltage divider formula. The measured voltage is processed in the Arduino, and the computed resistance value is displayed on an LCD screen for user convenience. The results demonstrate the accuracy and efficiency of this low-cost, microcontroller-based ohmmeter in determining resistance values. This project highlights the practical application of electrical principles and embedded systems in creating simple, reliable measurement tools.

---

2. Introduction

Background

The digital ohmmeter is an essential instrument for determining the resistance of electrical components, playing a critical role in electronics and engineering applications. By utilizing the fundamental principles of Ohm's Law, the device operates by measuring current flow through a known resistor to determine the unknown resistance. Understanding its working mechanism not only enhances practical skills in circuit design but also provides valuable insights into the principles of analog measurement techniques and their applications in real-world scenarios.

This project addresses the need for a simple, cost-effective, and accurate solution to measure resistance, leveraging the versatility of microcontroller-based systems such as the Arduino Nano.

Objectives

The primary objective of this project is to design and implement an ohmmeter capable of measuring resistances in the range of 1 Ω to 10 MΩ with a precision error margin of no more than ±5%. The system aims to provide accurate results, display the measured values in real-time on an LCD screen, and ensure ease of use for practical applications. Secondary objectives include enhancing comprehension of electronic components, such as resistors, milliammeter sensitivity, and switches, and developing a systematic approach to circuit design, calibration, and performance testing.

Scope

This project focuses on the development of a working prototype for resistance measurement using a voltage divider circuit. It will employ an Arduino Nano for processing signals and performing calculations, along with an LCD module for displaying data. The focus will not be on advanced features like auto-ranging, measuring reactive components such as inductors or capacitors, or integrating with external data logging systems. Instead, the priority will be on ensuring reliable resistance measurement within a specified range and accuracy.

---

3. Methodology

Block Diagram

The entire project can be broken down into three main components: the Resistance Sensor Unit (Voltage Divider), the Processor Unit (Arduino Nano), and the Display Unit (LCD Screen). The Sensor Unit receives an unknown resistor as input. When this resistor is connected, the output voltage varies between 0 and 5V, which is proportional to the resistance value. The Processor Unit accepts this input voltage and calculates the resistance based on the reference resistor 1 kΩ. It then sends a 10-bit data signal to the Display Unit, which contains the calculated resistance value. Finally, the Display Unit takes this 10-bit data and presents the resistance value with 8-bit on a 16x2 LCD display.

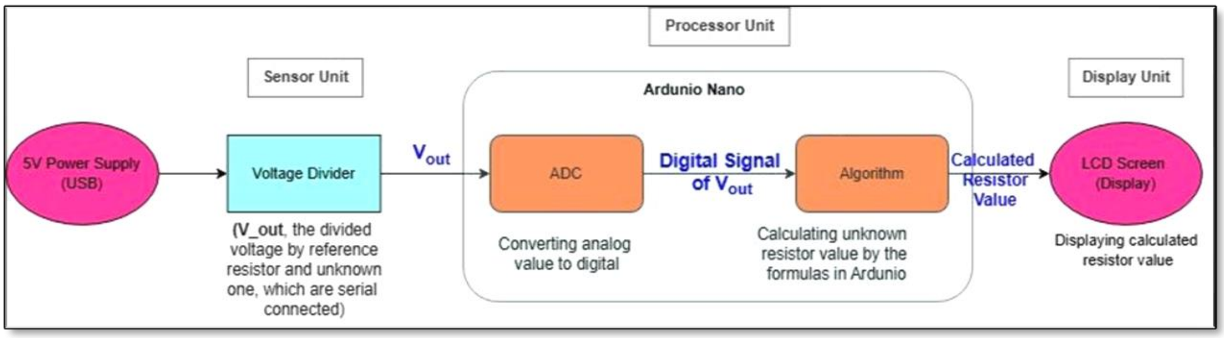

Processor Unit
Sensor Unit
Ardunio Nano
Display Unit
5V Power Supply (USB)
Voltage Divider
V_{out}
ADC
Digital Signal of V_{out}
Algorithm
Calculated Resistor Value
LCD Screen (Display)
(V_out, the divided voltage by reference resistor and unknown one, which are serial connected)
Converting analog value to digital
Calculating unknown resistor value by the formulas in Ardunio
Displaying calculated resistor value

Figure 1 Block Diagram of the circuit with refernce resistor 1 kΩ

1. **5V Power Supply (USB):** The system is powered by a 5V USB supply, which provides energy to all components.

2. **Voltage Divider:** A voltage divider circuit is used to divide the input voltage using a reference resistor and the unknown resistor connected in series. This division produces V_{out} ,

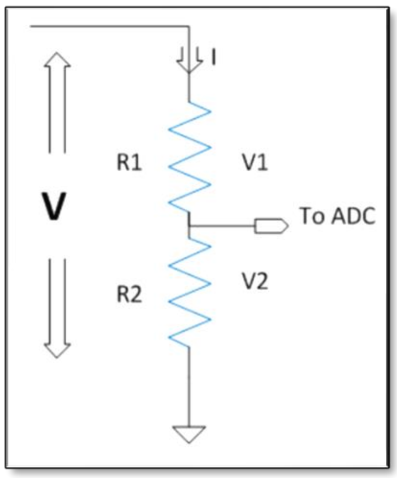


Figure 2 Voltage Divider circuit configuration

which is fed to the Arduino Nano's analog pin (A0) for voltage measurement.
In the voltage divider circuit shown in the image, the input voltage 'V' is divided between the resistors R1 and R2, resulting in voltages V1 and V2. The relationships can be expressed as:

$$R1 = \frac{R1}{\left(\frac{V}{V1} - 1\right)}$$

Assuming R2 is the unknown resistance, we know the input voltage 'V' is applied, and we can measure the voltage V1. With the value of R2 known, we can calculate R1. Let's consider R2's maximum value as 1M ohm. Given that we're applying 5V as V and that V1 should be at least 4.88 mV (**the ADC Voltage Resolution for this project**), we can substitute these values into the equation to find R2:

$$R1 = \frac{10^6}{\left(\frac{5}{4.88 \times 10^{-3}} - 1\right)} \approx 976\,\Omega$$

Therefore, using a reference 1K resistor is appropriate. This analysis ensures that the circuit operates efficiently within the specified parameters, allowing for accurate resistance measurements.

3. **Voltage Measurement by Analog Pin (A0):** Reads the divided voltage V_{out}.
4. **ADC (Analog-to-Digital Converter):** The Arduino's built-in ADC (Analog-to-Digital Converter) converts this analog voltage into a digital value. The analog signal connected to the

ADC must be between 0 and 5 volts. The ADC converts this voltage into a digital value, producing a number between 0 and 1023 (with 10-bit resolution).

5. **Algorithm (Arduino Nano):** The Arduino processes this digital data using an algorithm to compute the value of the unknown resistor based on the voltage divider formula.

6. **LCD Screen (Display):** Finally, the calculated resistance value is displayed on an LCD screen for user convenience.

Design Process

In terms of the selection of components, since the primary requirement was to measure resistances between 1 ohm and 10 megaohms with ±2% accuracy, a voltage divider circuit was chosen as the foundation due to its simplicity and effectiveness in calculating unknown resistances.

Component selection followed, with the Arduino Nano chosen as the processing unit because of its compact size, affordability, and built-in ADC capabilities. The comparison between Arduino Nano and Ardunio Uno yielded the results as both the Arduino Nano and Arduino Uno utilize the same microcontroller, the ATmega328P. This implies that both boards have equivalent processing power and input/output capabilities. Nonetheless, we preferred for the Arduino Nano in our project due to its more compact size and the adequacy of the number of pins required for our application.

When it comes to the selection of the LCD Screen (16x2); in our circuit, we utilized a Liquid Crystal Display (LCD) in 8-bit mode to present the measured resistance values in real time. The choice of 8-bit mode over 4-bit mode for the LCD was made because it facilitates faster data transfer.

In this circuit, the resistors (R1 and R2) are employed to create a voltage divider essential for measuring resistance. R1 typically serves as a fixed reference resistor, while R2 is the resistor whose value is being measured. Additionally, the circuit is powered by a 5V power supply, which is adequate for both the Arduino and the LCD.

In this project, a low-tolerance fixed resistor was used as the reference resistor to ensure measurement accuracy. A known resistor was 1 kΩ and were used as reference resistors for their suitability in minimizing error across the required measurement range. Also a 2.2 kΩ resistor was used to adjust the contrast of the LCD screen. The choice of which resistor to use as the resistance value of the resistor to adjust the contrast of the LCD screen is decided based on our research on online sources. Still, after having some discrepancies, we need to test this decision. After measurements by the potentiometer in the laboratory, we observed that 2.2k ohm was entirely appropriate to integrate into the circuit for the contrast resistor.

The selection was based on the principle that reducing the R1/R2 ratio decreases the error caused by resistance tolerance, as lower ratios lead to more minor voltage deviations due to tolerance differences. However, the ratio cannot be reduced indefinitely due to practical limitations, such as ensuring non-zero resistance values and maintaining resolution. These specific values balance accuracy and practicality, allowing for precise resistance measurements across the target range.

When we move to the next station, the circuit design phase connected the reference resistor and unknown resistor in series, with the midpoint of the voltage divider linked to the Arduino's A0 pin to read V_{out}. The LCD pins were interfaced with the Arduino's digital pins to facilitate a real-time display of the resistance value. Finally, an algorithm was developed and programmed into the Arduino to compute the resistance value of the unknown resistor using the voltage divider formula.

Hardware Setup

To elaborate on the connections between the Arduino Nano and the LCD, we connected the RS pin (which allows the LCD to determine whether the information being sent is a command or data) to Arduino pin D2, and the E pin (which sends a HIGH (1) to LOW (0) pulse, enabling the LCD to detect and process the transmitted data) to Arduino pin D3. The connection of LCD pins D4-D7 to Arduino Nano pins D4-D7 is a choice for optimizing the circuit design and ensures clarity and ease of debugging, making the circuit straightforward to implement.

Regarding the resistor connections, a voltage divider circuit was set up with a 1 kΩ resistor and an unknown resistor. The output of the voltage divider circuit was connected to the A0 analog input pin of the Arduino Nano to receive the analog data.

Software Development

Using the Arduino IDE, the LiquidCrystal library was utilized to control the LCD screen. Voltage measurements were taken from the A0 analog pin, and the unknown resistor value was calculated using Ohm's Law:

$$\boxed{\mathrm{R\_}x = \frac{\mathrm{V\_out * R\_ref}}{\mathrm{V\_in - V\_out}}}$$

Here R_x is an unknown resistor, R_{ref} is reference resistor (1 kΩ), V_{in} is input voltage (5V) to the circuit while V_{out} being voltage across the reference resistor.

The coding steps for this process begin with reading the voltage from the analog pin. This voltage value is then used to calculate the resistance of the unknown resistor. Once the calculation is complete, the resulting resistance value is displayed on an LCD screen. This process encompasses both the measurement and display phases, allowing the user to see the value of the resistor easily. The code is shown in the Appendix part.

Circuit Schematic

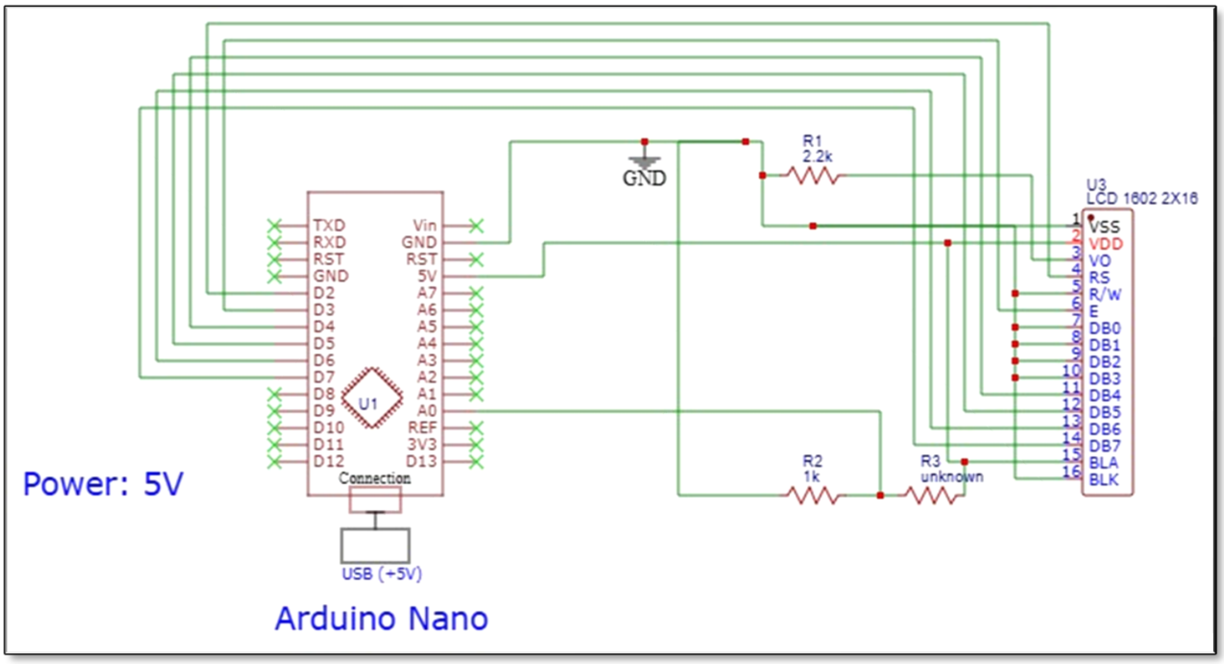


Figure 3 Circuit schematic

Components

Arduino Nano: This microcontroller board is the central processing unit of the circuit. It processes the input from the resistance measurement circuit and drives the LCD to display the results.

LCD Screen (16x2): The Liquid Crystal Display (LCD) is used to show the measured resistance values. In this schematic, it operates in 8-bit mode, allowing it to communicate with the Arduino using fewer pins.

Resistors (R1, R2): The resistors in this circuit are used to create a voltage divider, which is essential for measuring resistance. R1 is typically a fixed reference resistor, while R2 is the resistor whose value is being measured.

Analog Input Pin (A0): This pin reads the voltage from the voltage divider formed by R1 and the variable resistor (R2). The voltage at this point is proportional to the resistance of R2.

Power Supply: The circuit is powered by a 5V supply, which is sufficient for the Arduino and the LCD.

At its core, the Arduino Nano serves as the central processing unit. A voltage divider circuit with a fixed reference resistor (R1) and a test point for the unknown resistor (R2) is used to generate the input voltage V_{out}. This voltage is processed through the Arduino's analog input pin. An LCD module is connected to the Arduino's digital pins to display the resistance values. The entire system is powered through a USB connection, which provides the necessary 5V supply for all components.

Implementation and Debugging

The implementation began with assembling the circuit and verifying the functionality of individual components. The LCD display and the Arduino Nano were tested to ensure they were functioning as intended, and proper connections were established. Known resistor values were then used to validate the voltage divider circuit and confirm the accuracy of the Arduino's analog input readings.

One of the main challenges encountered during debugging was selecting appropriate reference resistors (such as 100 Ω, 1 kΩ and 100 kΩ) to achieve accurate resistance measurements across the desired range. Initially, the chosen reference resistor values led to inconsistent results, especially near the lower and upper limits of the measurement range. This issue was addressed by reevaluating the R1/R2 ratio and selecting 1 kΩ a reference resistor. By doing so, we achieved a measurement range of approximately **180 ohms to 33 kΩ**. This narrower range was chosen deliberately to ensure higher accuracy for the most relevant resistances, as a wider range would compromise precision.

Another challenge involved the system's inability to provide consistent performance during the initial tests. The problem was traced back to inaccuracies in the algorithm used for resistance calculation. By refining the mathematical expressions and calibrating the Arduino's ADC readings, the system was stabilized to deliver reliable results.

Final testing involved iterative improvements, where resistors of varying tolerances and values were measured to ensure the system's accuracy met the approximately ±2.44% error margin. The refined design consistently delivered precise and repeatable results, confirming the effectiveness of the implemented solution in that range.

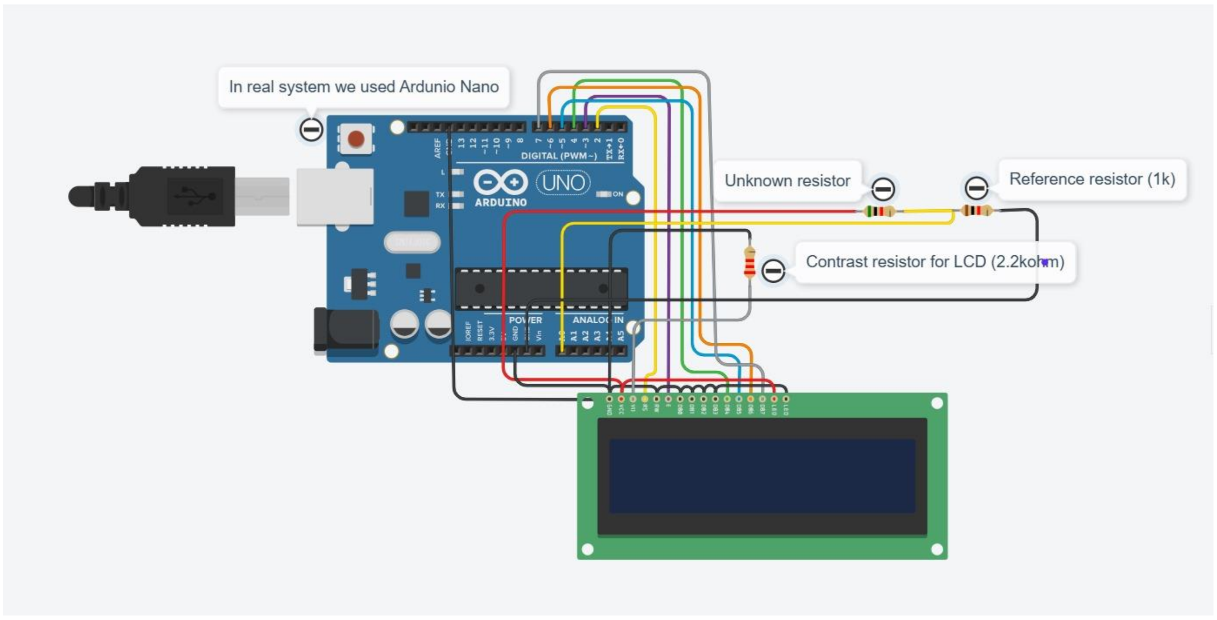

In real system we used Ardunio Nano
Unknown resistor
Reference resistor (1k)
Contrast resistor for LCD (2.2kohm)
UNO
ARDUINO

Figure 4 Circuit design on ThinkerCad

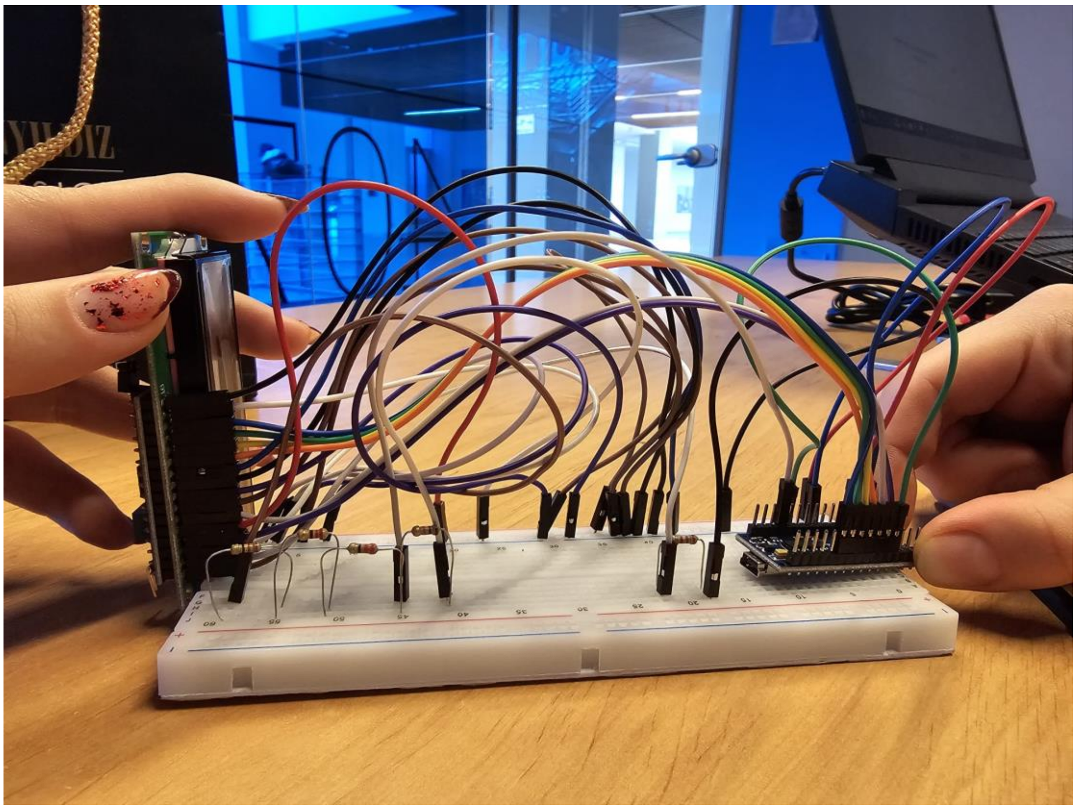


Figure 5 The circuit configuraiton on breadboard

4. Results and Discussion

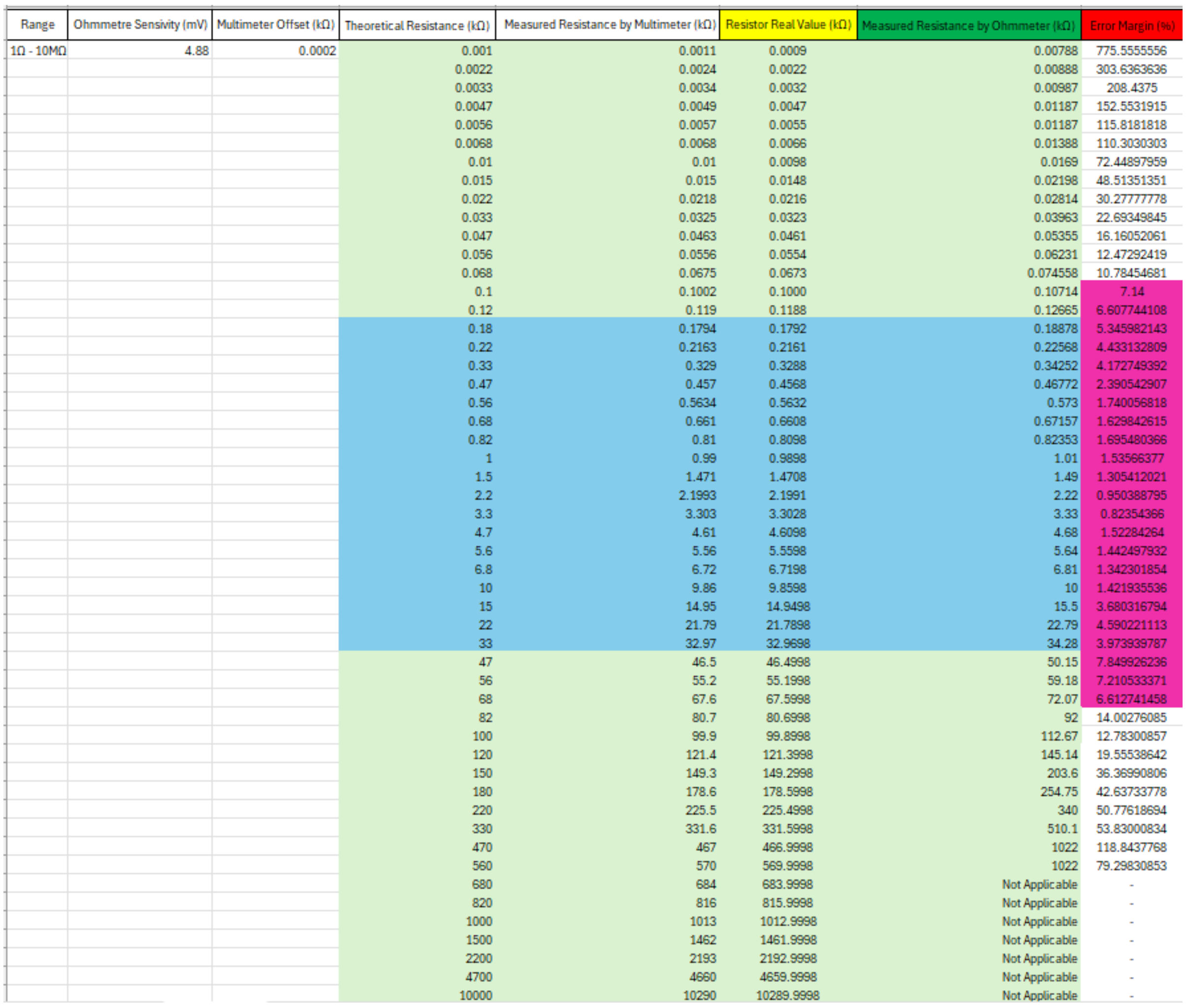
Results

Measurements were taken for various resistances ranging from 1 Ω to 10 MΩ under laboratory conditions and compared with the actual resistance values. Before starting the measurements, the multimeter's offset was measured as 0.0002 kΩ, and the error margin value was accepted as 0. The Resistor Real Value displays the Measured Resistance by Multimeter values minus the Multimeter Offset values, shown in Table 1 below. The ohmmeter was able to measure resistances in the range of approximately 180 ohms to 33 kΩ within an acceptable error margin of about ±2.44% and a maximum of 5%.

The aim of this project by making an ohmmeter was being able to measure the resistors between 1 Ω to 10 MΩ; however, our circuit gives correlated measurement results in the range of 180 Ω to 33 kΩ due to the fact that our circuit design is configured with 1 kΩ reference resistor. In the selection of the reference resistor, it was considered that adding more reference resistors would complicate the circuit and make management more difficult. More complex circuit designs can increase the likelihood of errors and negatively impact the reliability of measurements. Therefore, considering the scope and objectives of the project, the use of 1 kΩ reference resistor were preferred.

When these points are taken into consideration, measurements taken with the 1 kΩ reference resistor have provided an adequate level of accuracy to meet the project's requirements and have proven to be a suitable solution for our application. By avoiding complex circuit designs, a more reliable and manageable measurement process has been achieved. The use of a voltage divider circuit allowed the Arduino Nano to process the input voltage and accurately calculate the resistance based on Ohm's Law. The data sheet of the measurements and the graphs that show the margin of error and the real value of resistors' values vs. measured resistor values by our ohmmeter are shown below.

Table 1 Measurement data



Range	Ohmmetre Sensivity (mV)	Multimeter Offset (kΩ)	Theoretical Resistance (kΩ)	Measured Resistance by Multimeter (kΩ)	Resistor Real Value (kΩ)	Measured Resistance by Ohmmeter (kΩ)	Error Margin (%)
1Ω - 10MΩ	4.88	0.0002	0.001	0.0011	0.0009	0.00788	775.5555556
			0.0022	0.0024	0.0022	0.00888	303.6363636
			0.0033	0.0034	0.0032	0.00987	208.4375
			0.0047	0.0049	0.0047	0.01187	152.5531915
			0.0056	0.0057	0.0055	0.01187	115.8181818
			0.0068	0.0068	0.0066	0.01388	110.3030303
			0.01	0.01	0.0098	0.0169	72.44897959
			0.015	0.015	0.0148	0.02198	48.51351351
			0.022	0.0218	0.0216	0.02814	30.27777778
			0.033	0.0325	0.0323	0.03963	22.69349845
			0.047	0.0463	0.0461	0.05355	16.16052061
			0.056	0.0556	0.0554	0.06231	12.47292419
			0.068	0.0675	0.0673	0.074558	10.78454681
			0.1	0.1002	0.1000	0.10714	7.14
			0.12	0.119	0.1188	0.12665	6.607744108
			0.18	0.1794	0.1792	0.18878	5.345982143
			0.22	0.2163	0.2161	0.22568	4.433132809
			0.33	0.329	0.3288	0.34252	4.172749392
			0.47	0.457	0.4568	0.46772	2.390542907
			0.56	0.5634	0.5632	0.573	1.740056818
			0.68	0.661	0.6608	0.67157	1.629842615
			0.82	0.81	0.8098	0.82353	1.695480366
			1	0.99	0.9898	1.01	1.53566377
			1.5	1.471	1.4708	1.49	1.305412021
			2.2	2.1993	2.1991	2.22	0.950388795
			3.3	3.303	3.3028	3.33	0.82354366
			4.7	4.61	4.6098	4.68	1.52284264
			5.6	5.56	5.5598	5.64	1.442497932
			6.8	6.72	6.7198	6.81	1.342301854
			10	9.86	9.8598	10	1.421935536
			15	14.95	14.9498	15.5	3.680316794
			22	21.79	21.7898	22.79	4.590221113
			33	32.97	32.9698	34.28	3.973939787
			47	46.5	46.4998	50.15	7.849926236
			56	55.2	55.1998	59.18	7.210533371
			68	67.6	67.5998	72.07	6.612741458
			82	80.7	80.6998	92	14.00276085
			100	99.9	99.8998	112.67	12.78300857
			120	121.4	121.3998	145.14	19.55538642
			150	149.3	149.2998	203.6	36.36990806
			180	178.6	178.5998	254.75	42.63733778
			220	225.5	225.4998	340	50.77618694
			330	331.6	331.5998	510.1	53.83000834
			470	467	466.9998	1022	118.8437768
			560	570	569.9998	1022	79.29830853
			680	684	683.9998	Not Applicable	-
			820	816	815.9998	Not Applicable	-
			1000	1013	1012.9998	Not Applicable	-
			1500	1462	1461.9998	Not Applicable	-
			2200	2193	2192.9998	Not Applicable	-
			4700	4660	4659.9998	Not Applicable	-
			10000	10290	10289.9998	Not Applicable	-

To observe the accuracy of this ohmmeter, we measured all the resistors that we could find in the lab ranging 1Ω to 10 MΩ. After the analysis we found that the data indicate a significant correlation between used reference resistor and the range that can be measurable accurately via this ohmmeter. During this analysis, we assume that the multimeter that we used to measure the real values of the resistor has zero error margin. And also, the error margin calculations are based on this formula [(**measured resistance value by ohmmeter) − (real resistor value)] /real resistor value×100.**

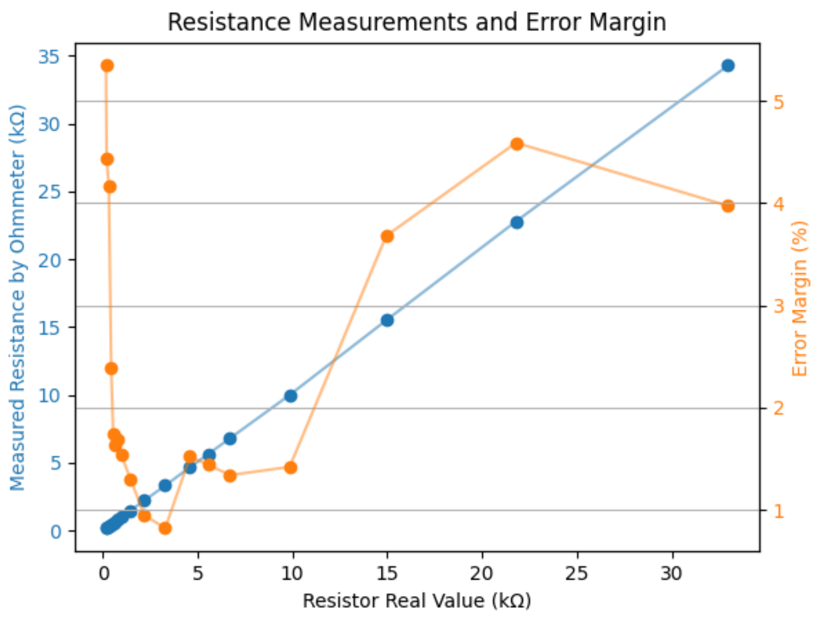


Figure 6 The graph showing the relation between real resistor value and measured resistor value by ohmmeter and the error margin in linear scale

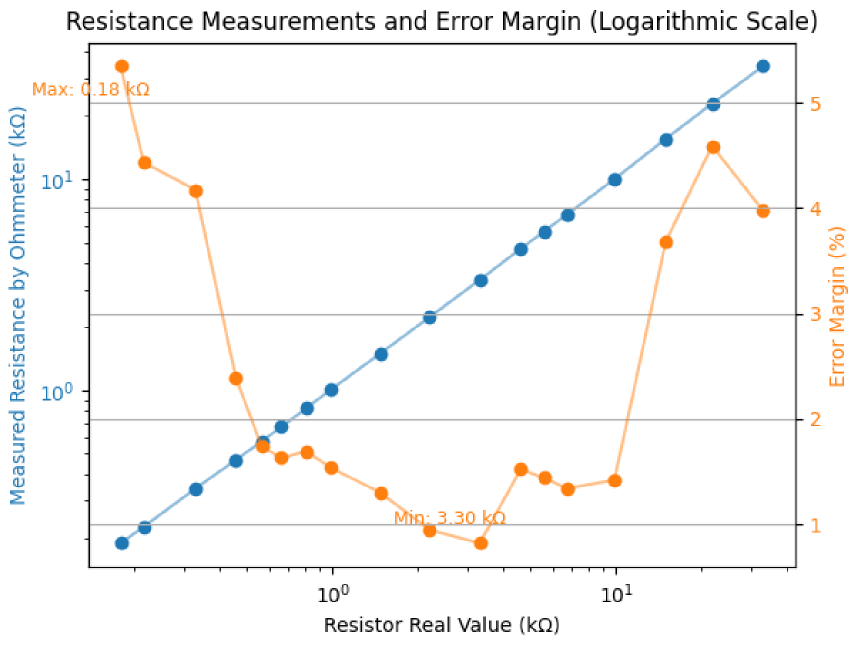


Figure 7 The graph showing the relation between real resistor value between the ±5% margin of error and measured resistor values by ohmmeter and the error margin in linear scale with the min and max values in logarithmic scale

There are two graphs shown above showing the interval between 180 Ω to 33 kΩ resistor values. First one has Real Resistor Values on x-axis, Measured Ohmmeter Values on primary y-axis, and error margin on secondary y-axis. Due to the fact that it is difficult to observe the values between 0 to 5 kΩ on the x-axis, we generate a second graph that shows the zoomed in version of the first one and indicates the resistor values which have maximum and minimum error margins.

In analyzing the graph depicting the relationship between the real resistor values (x-axis) and the measurements obtained from our ohmmeter (y-axis), we observe that the lowest error margin occurs near the reference resistor value of 1 kΩ. However, the minimum error margin is actually recorded at a resistor value of 3.3 kΩ. Under ideal conditions, one would expect this minimum error margin to correspond to the 1 kΩ reference resistor. Thus, the observed deviation is noteworthy. Several factors may contribute to this discrepancy such as the error margin and offset errors of the Arduino system, as well as the inherent inaccuracies of the multimeter used.

Furthermore, as we move to the left and right of this minimum error margin, we observe that the error margin increases. This indicates that we are approaching the extremes of our selected range, specifically towards 180 Ω and 33 kΩ. To accurately measure higher resistor values, it is necessary to shift this minimum error margin interval to the right, which entails using a reference resistor with a higher resistance value in the circuit. Detailed examinations related to this topic are discussed under the 'Future Works' section.

Mistakes

During the design stage, one significant challenge was the selection of appropriate reference resistors. Initially, improper resistor values caused incorrect measurements, especially at the extremes of the resistance range. By reevaluating the R1/R2 ratio, 1 kΩ resistor was chosen, narrowing the range to 180 ohms to 33 kΩ, which improved measurement precision.

Another issue was the algorithm for resistance calculation, which initially led to inaccurate results due to rounding errors and incorrect handling of ADC values. This was resolved by refining the mathematical model by ensuring proper conversion from digital values to resistance. Additionally, during implementation, some minor wiring mistakes caused inconsistent readings. These errors were resolved by carefully reviewing the circuit and testing connections systematically.

Discussion

The project successfully achieved its objective of measuring resistances with an average ±2.44% error margin, demonstrating the effectiveness of the voltage divider and Arduino-based approach. The selected range, although narrower than initially intended, provided high accuracy and repeatability, which was a key strength of the system.

However, the limitation of the measurement range (180 ohms to 33 kΩ) excludes very low and very high resistances, which could be addressed in future iterations by incorporating additional reference resistors or range selection mechanisms. Another limitation was the dependence on fixed tolerance levels of resistors; using resistors with tighter tolerances could further enhance accuracy.

---

5. Conclusion

The project successfully met its objectives, resulting in the design and assembly of an ohmmeter capable of measuring resistances across a range of 180 ohms to 33 kΩ with an accuracy of approximately ±2.44%. The system demonstrated reliability and accuracy, with measurements consistently within the expected tolerance. Through this project, our understanding of electronic components, particularly resistors, and measurement techniques was significantly enhanced. The hands-on experience with circuit design, calibration, and data analysis contributed to a deeper appreciation of how to optimize electronic measurement systems.

Future Works

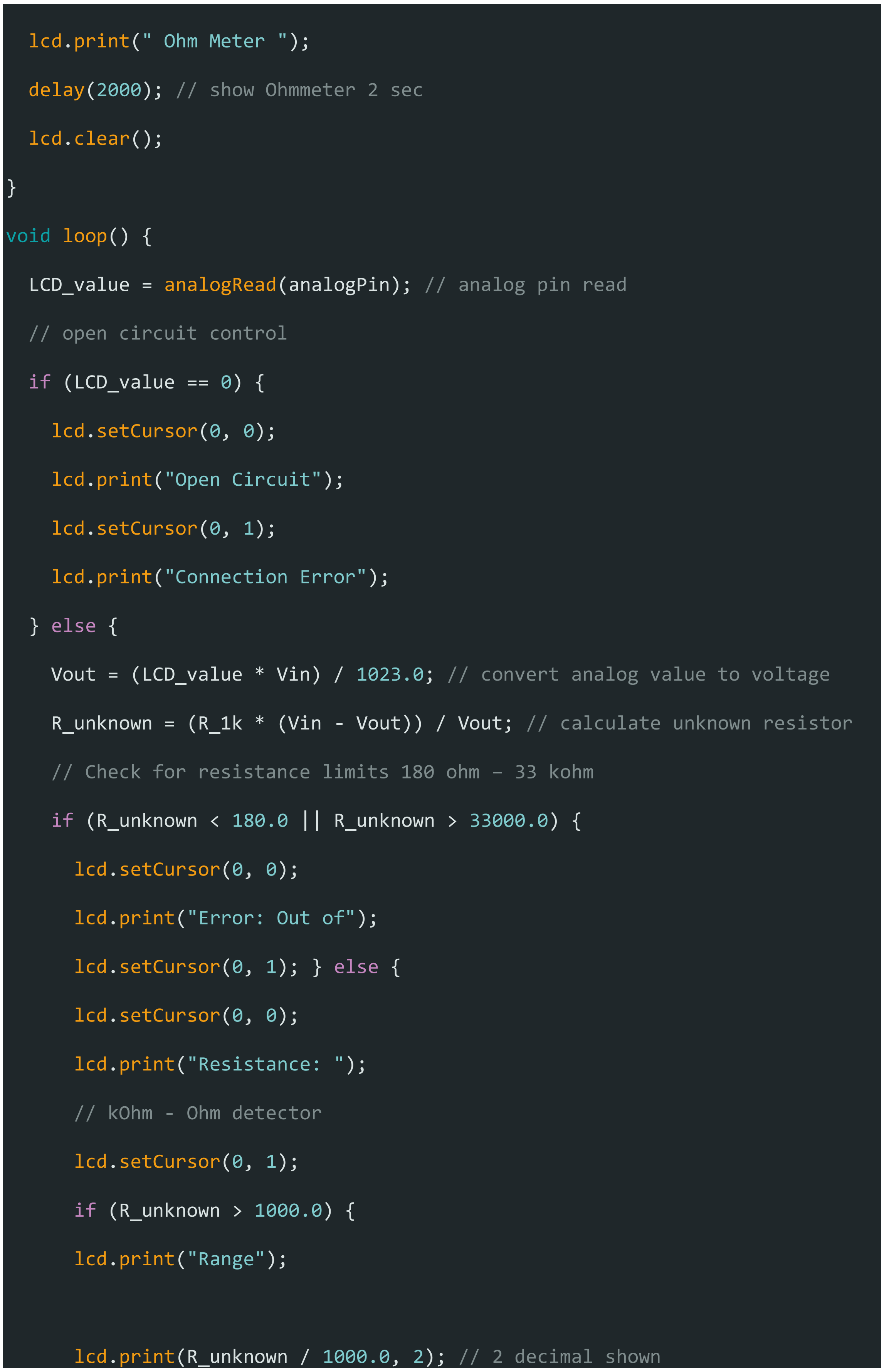
As we look to expand the capabilities of our ohmmeter to measure larger resistors, it is essential to carefully select appropriate reference resistor values to maintain accuracy and minimize error margins. Reference resistor values of 100 Ω, 100 kΩ and 1 MΩ can be added to the circuit, enabling the measurement of resistance values below 180Ω and above 33 kΩ (approximately between 1Ω and 100 MΩ) with a maximum error rate of 5%. Our experience with a 1 kΩ reference resistor allowed us to measure resistors between 180 Ω and 33 kΩ with a maximum error margin of 5%. Building on this analysis, we can derive suitable reference resistor values for larger resistors. For instance, while measuring between approximately 1 Ω and 180 Ω by this ohmmeter, using a reference value of 100 Ω will yield a more effective result. In larger reference resistor values, the range in which the error value is acceptable can be approximately found using the value ±16.410 Ω ((33 kΩ - 180 Ω)/2). However, this value corresponds to a very small range for reference values such as 100 kΩ and 1MΩ. According to our assumptions, operations can be performed using 100 kΩ reference resistor for measuring the range of 15 kΩ - 330 kΩ, and 1 MΩ reference resistor for the range of 470 kΩ - 10 MΩ. By determining the reference resistor limits, the Arduino code must be adjusted accordingly, the section which shows the resistor interval that can be measured correctly with that reference resistor. These adjustments will require careful consideration of the measurement system’s limitations and the impact of the reference resistor on overall accuracy. By systematically analyzing our previous results and implementing these new reference resistor values, we can enhance the reliability and precision of our ohmmeter for a broader spectrum of resistance measurements.

Additionally, ADC calibration can be improved by using an ADC model with a resolution of 12 bits or higher, instead of the 10-bit resolution ADC found on the Arduino Nano. To minimize noise in the system, an RC filter can be applied to the analog pin connection. Also, for the same purpose we can enclose the circuit in a metal casing to shield it from electromagnetic interference. For enhanced data accuracy and resolution, the Arduino Due microcontroller can be used thanks to its superior capabilities like having 32-bit processor while Arduino Nano have 8-bit processor. Organizing cable connections, unlike in Figure 5, results in a cleaner layout that facilitates error detection and resolution. Using short, high-quality cables also help minimize noise affecting analog signals. For accurate measurements, it's crucial to insulate the circuit from external factors like temperature, humidity, and dust, and to house it in a protective box. Adding features like data logging or wireless connectivity can enhance functionality and expand application scope, making the circuit more user-friendly for its intended use.

6. Appendix

According to our experimental results, we have adjusted the Arduino code below to measure only the range that provides a maximum 5% error margin (180 Ω - 33 kΩ). In this way, the possibility of users making significant errors in their measurements while using the code has been prevented.

```
#include <LiquidCrystal.h>

// LCD digital connection pins (RS, E, D4, D5, D6, D7)

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

int analogPin = A0; // analog pin

float Vin = 5.0;

float Vout = 0.0;   // voltage drop on reference resistor

float R_1k = 1000.0;     // reference resistor

float R_unknown = 0.0;    // unknown resistor

int LCD_value = 0;    // analog to digital converter value / LCD value


void setup() {

  lcd.begin(16, 2);

  lcd.setCursor(0, 0); // adjusting rows and columns on the LCD screen (row,
column)
```



```
  lcd.print(" Ohm Meter ");

  delay(2000); // show Ohmmeter 2 sec

  lcd.clear();

}

void loop() {

  LCD_value = analogRead(analogPin); // analog pin read

  // open circuit control

  if (LCD_value == 0) {

    lcd.setCursor(0, 0);

    lcd.print("Open Circuit");

    lcd.setCursor(0, 1);

    lcd.print("Connection Error");

  } else {

    Vout = (LCD_value * Vin) / 1023.0; // convert analog value to voltage

    R_unknown = (R_1k * (Vin - Vout)) / Vout; // calculate unknown resistor

    // Check for resistance limits 180 ohm – 33 kohm

    if (R_unknown < 180.0 || R_unknown > 33000.0) {

      lcd.setCursor(0, 0);

      lcd.print("Error: Out of");

      lcd.setCursor(0, 1); } else {

      lcd.setCursor(0, 0);

      lcd.print("Resistance: ");

      // kOhm - Ohm detector

      lcd.setCursor(0, 1);

      if (R_unknown > 1000.0) {

      lcd.print("Range");


      lcd.print(R_unknown / 1000.0, 2); // 2 decimal shown
```

Figure 8 The Arduino Code