Kalyani Government Engineering College

Department of Electronics & Communication Engineering

B. Tech 3rd Year (Semester: 6) Mini Project

A PROJECT REPORT

ON

Implementation of Auto Ranging Ohmmeter using Arduino Uno

Project Guide: *Dr. Aritra Acharyya*

Submitted by:

Rohit Biswas (10200322025)

Soutik Mondal (10200322008)

Sourav Sarkar (10200322019)

Debarghya Paul (10200322016)

ACKNOWLEDGEMENT

We want to start by sincerely thanking **Dr. Aritra Acharyya**, our project supervisor, for his essential advice, inspiration, and unwavering support during this project. His knowledge, perceptive criticism, ideas and endurance were invaluable in determining the course and effective conclusion of our job. We are also grateful to the teachers and staff of the Department of Electronics and Communication Engineering, Kalyani Government Engineering College for offering instruments and a welcoming learning atmosphere that substantially benefited our research and practical endeavours.

Rohit Biswas*

(10200322025)

Soutik Mondal*

(10200322008)

Sourav Sarkar*

(10200322019)

Debarghya Paul*

(10200322016)

Dr. Aritra Acharyya

Assistant Professor,

Department of Electronics & Communication Engineering,

Kalyani Government Engineering College

*6th Semester students in the Department of Electronics & Communication Engineering, Kalyani Government Engineering College

LIST OF FIGURES AND TABLES

List of Tables

List of Figures

TABLE OF ABBREVIATIONS

Serial Number	Abbreviations	Meaning
1	ADC	Analog-to-Digital Converter
2	CMOS	Complementary Metal-Oxide-Semiconductor
3	DIY	Do It Yourself
4	ESP32	Espressif Systems on a Chip 32-bit
5	IC	Integrated Circuit
6	IDE	Integrated Development Environment
7	PCB	Printed Circuit Board
8	USB	Universal Serial Bus

TABLE OF CONTENTS

ABSTRACT

In this project, an Arduino Uno microcontroller is used to design and implement an Auto Ranging Ohmmeter. The goal is to create an efficient and reasonably priced digital resistance measurement tool that automatically chooses the right range for precise results. By adding an auto ranging mechanism, this project overcomes the drawback of traditional ohmmeters' manual range selection process, which can be cumbersome and prone to mistakes. The voltage divider rule is the foundation upon which the system operates. We determine the value of the unknown resistance by choosing several reference resistances and measuring the voltage at the intersection of the chosen reference and unknown resistance values. The Arduino IDE's serial monitor shows the measured resistance in real time. The project shows how embedded systems and fundamental electronics can be used practically. It may also be expanded to integrate into other systems where resistance monitoring is required.

Key Words: Arduino Uno, Auto Ranging Ohmmeter, voltage divider rule, embedded systems.

INTRODUCTION

An ohmmeter is an essential electronic instrument used to measure the resistance of a component or circuit. Resistance, measured in ohms (Ω), is a key parameter in determining how much a material or component opposes the flow of electric current. Ohmmeters are widely used in electrical and electronic applications for tasks such as verifying resistor values, checking circuit continuity, and diagnosing faulty components.

The need for accurate resistance measurement is critical in various fields, including circuit design, maintenance, repair, and testing. Whether for educational, industrial, or DIY electronics purposes, having a reliable resistance measuring tool is indispensable.

However, traditional ohmmeters, especially those in analog multimeters or basic digital types, often suffer from several limitations. One of the major issues is manual range selection. The user must estimate the value of the unknown resistance and choose a suitable range before measurement. Selecting the wrong range can lead to inaccurate readings or an overload display, requiring repeated adjustments. This makes the process inefficient, especially for users who are not experienced with electronic components. Additionally, traditional meters may lack the sensitivity or resolution required for precise low or high resistance measurements.

To overcome these limitations, this project proposes the design and development of an auto ranging ohmmeter using Arduino Nano. An auto ranging ohmmeter automatically detects the magnitude of the unknown resistance and adjusts its measurement range accordingly. This eliminates the need for manual range selection, reduces user error, and speeds up the measurement process.

In this project, the Arduino Uno acts as the core controller. Its selects different predetermined reference resistance one by one and measures the voltage at the junction of the unknown voltage. Using the values of voltage it determines the correct range by an auto ranging algorithm. This auto ranging system not only enhances ease of use and measurement accuracy but also demonstrates how microcontrollers can be effectively used to modernize traditional test equipment.

OBJECTIVES

The main objective of this project is to design and implementation of an auto ranging ohmmeter using the Arduino Uno that can measure resistance across a wide range of values without manual range selection by the user. The specific goals include:

1. To develop a microcontroller-based ohmmeter capable of measuring unknown resistances.
2. To implement an auto ranging algorithm that dynamically adjusts the measurement range for optimal accuracy.
3. Simplify operation for users of all skill levels by automating range selection, reducing the chance of user error.
4. To build a cost-effective and portable device that can be used for educational and practical electronic applications.
5. Allow testing of various components and circuits—from low-resistance wires to high-resistance insulators—without manual adjustments.

SPECIFICATIONS AND TECHNICAL REQUIREMENTS

In this section, all the required components which are used on the development of this project is listed and with their very brief description and quantity.

TABLE 1: Component Specification List

Component	Quantity	Description
Arduino Uno	1	Main microcontroller board
CD74HC154EN	1	4 to 16 Decoder
SN74LS04N	2	Hex Inverter
1N4007	5	PN junction diode
Reference Resistors	5	Known carbon resistors (2,177,987,14760,98000)
Breadboard	2	For circuit assembly
Jumper Wires	Several	For connections
Power Supply (5V USB)	1	To power the Arduino
Unknown Resistor (test samples)	Several	For testing the ohmmeter (0.8 ohm to 2M ohm)

METHODOLOGY

Working Principle

Principle of **Voltage Division Rule** is used to calculate the unknown voltage.

The voltage divider is the series of resistors or capacitors that can be tapped at any intermediate point to generate a specific fraction of the voltage applied between its ends. It consists of an electric circuit composed of two resistors and one input voltage supply. The below figure shows a simple voltage divider. In this circuit, two resistors are connected in series. The output voltage of the voltage divider is a function of the input voltage. This circuit helps to determine how the input voltage divides among the components in the circuit.

The voltage divider formula is given by,

$$V_0 = \frac{R_2}{R_1 + R_2} V_s \qquad \ldots\ldots\ldots\ldots (1)$$

V_0: output voltage
V_s: input voltage
R_1, R_2: resistance of the resistors

From equation (1) we can find the value of R_2, in terms of R_1 using

$$R_2 = \frac{V_0}{V_s - V_0} R_1 \qquad \ldots\ldots\ldots\ldots (2)$$

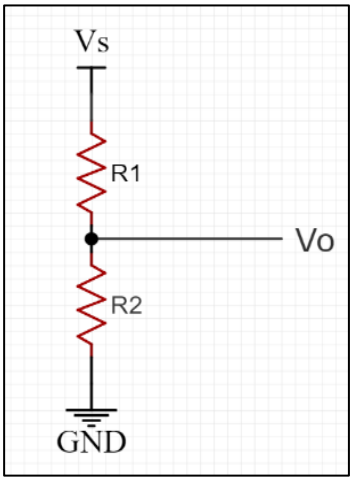


Fig 1: Voltage Divider Circuit

Circuit of Auto-ranging Ohmmeter

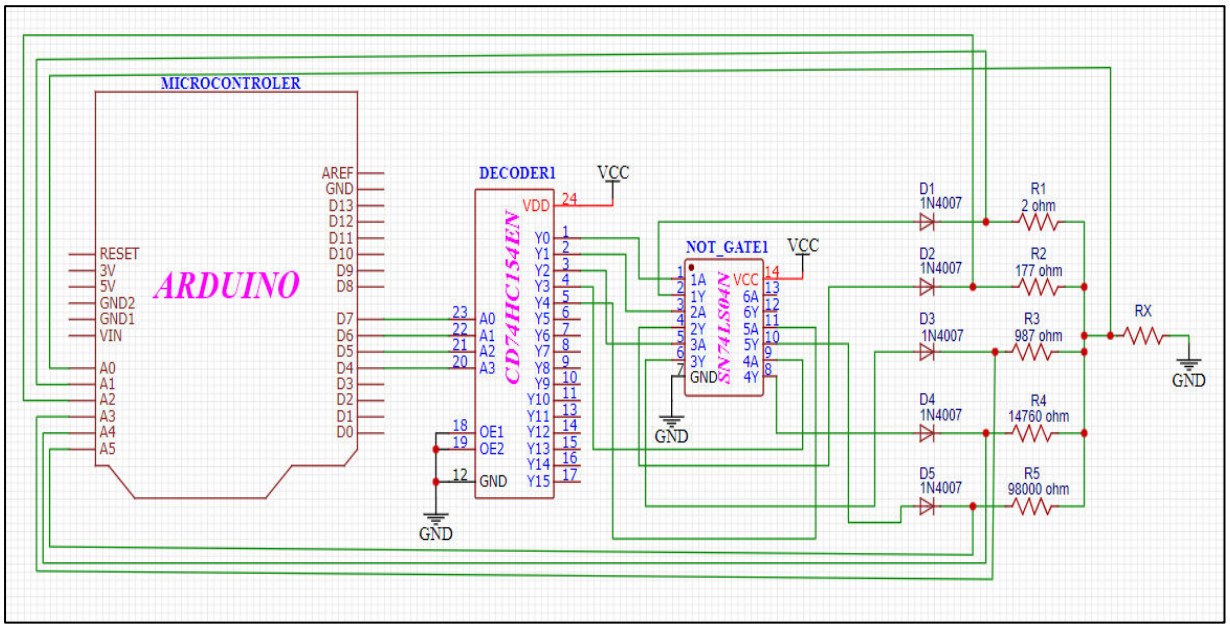


Fig 2: Schematic Diagram of Auto Ranging Ohmmeter

The hardware circuit consists of the following components:

- **Microcontroller (Arduino Nano)**: Arduino Uno is a well-known Microcontroller Board with an open-source architecture and IDE. In our project it is used to handle ADC conversion, range switching and calculation of resistance value of unknown resistor.

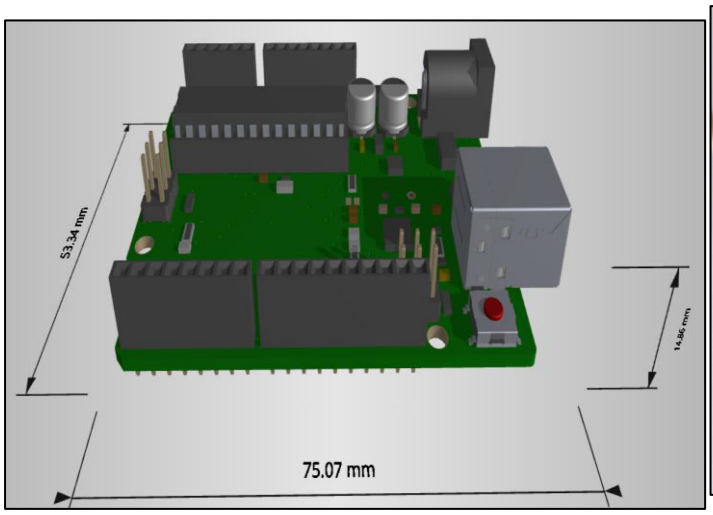


Fig: 3.1

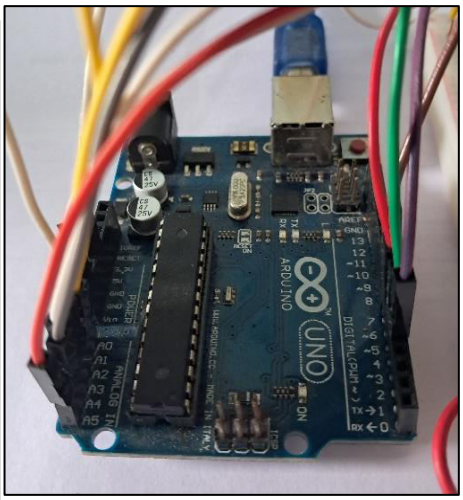


Fig: 3.2

Fig 3: (3.1) Arduino Uno 3D model with dimensions, (3.2) Arduino Uno used in the project

- **4-to-16 Decoder (CD74HC154EN)**: It is a high-speed 24 pin CMOS logic 4 to 16 Line Decoder IC. It operates in the temperature range of -55 to 125 ˚C [1]. It is used to select the reference resistors as the per select line inputs by Arduino digital pins.

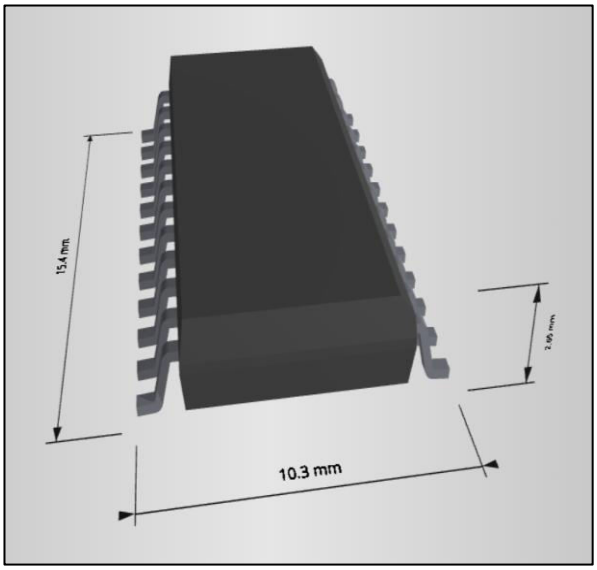


Fig: 4.1

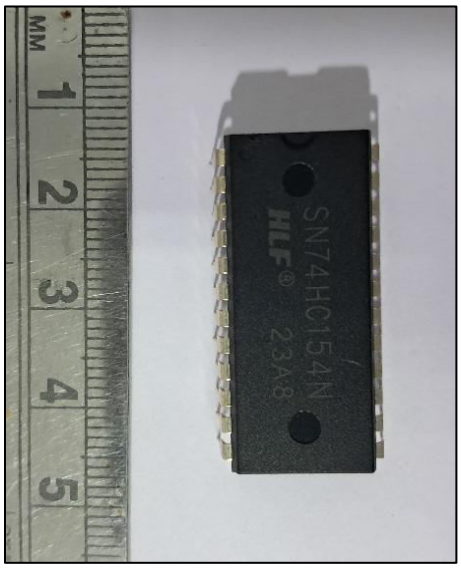


Fig: 4.2

Fig 4: (4.1) CD74HC154EN 3D model with dimensions, (4.2) CD74HC154EN used in the project

- **Hex Inverter (SN74LS04N)**: It is a 14 pin Hex Inverter i.e. this IC contains 6 NOT gates. It operates in the temperature range of 0 to 70 ˚C [2]. As the decoder output is active low to make it active high when the corresponding select inputs for that reference line is given, inverter is used.

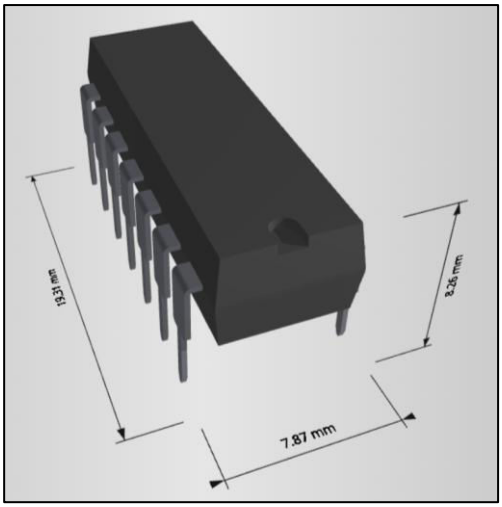
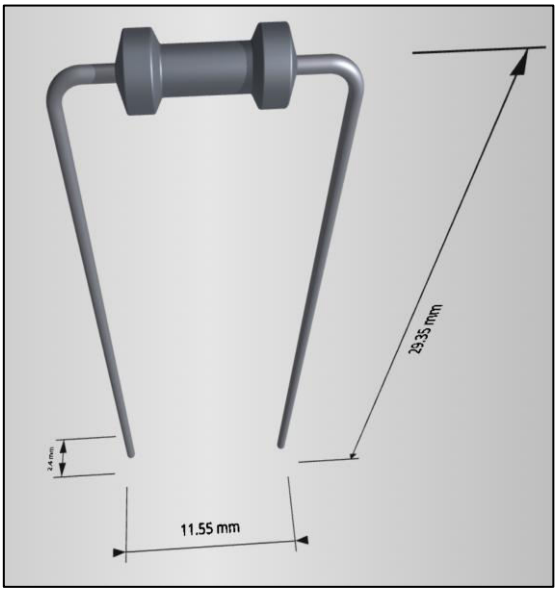


Fig: 5.1 Fig: 5.2

Fig 5: (5.1) SN74LS04N 3D model with dimensions, (5.2) SN74LS04N used in the project

- **Reference Resistors**: Carbon resistors are one of the most common types of electronics used. They are made from a solid cylindrical resistor element with embedded wire leads. Some known valued resistors (e.g., 2 Ω, 177 Ω, 987 Ω, 14780 Ω, 98000 Ω) are used for different measurement ranges.



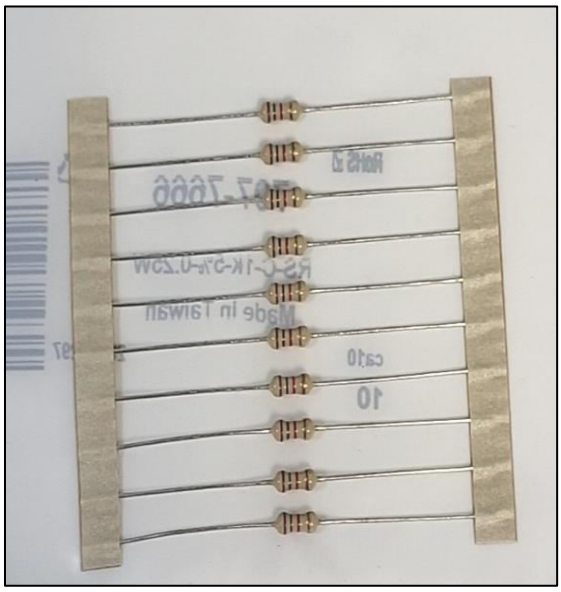
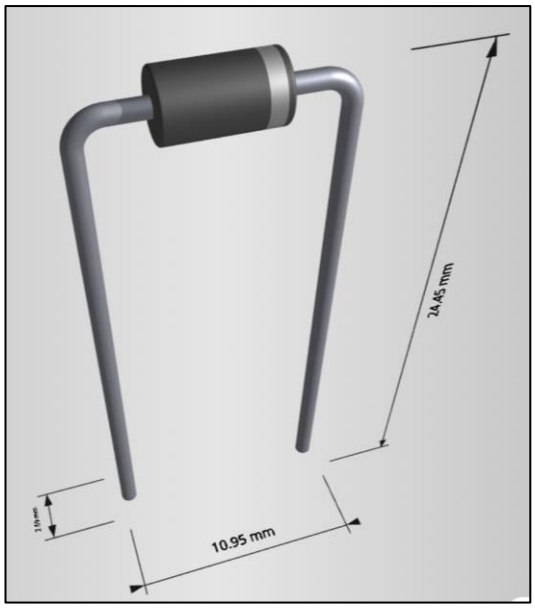


Fig: 6.1 Fig: 6.2

Fig 6: (6.1) Carbon resistor 3D model with dimensions, (6.2) Carbon resistors used in the project

- **Diodes (1N4007)**: It is a PN junction diode. It can handle a maximum junction temperature up to 150 ˚C [3]. It is used between reference resistors and output pin of SN74LS04N. Used to avoid unselected reference resistors becoming parallel with the unknown resistor.



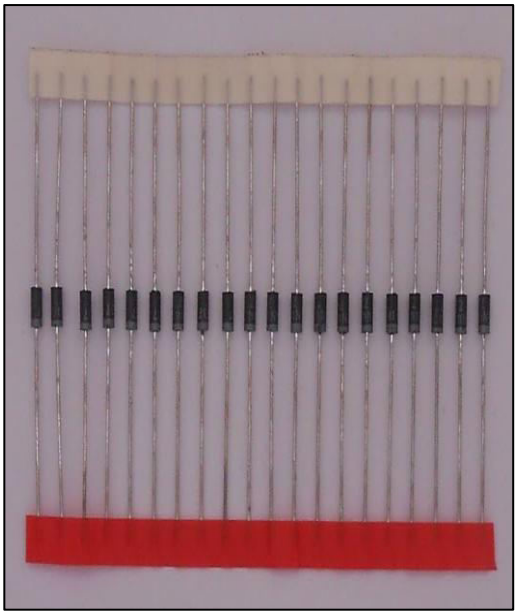


Fig: 7.1 Fig: 7.2

Fig 7: (7.1) 1N4007 3D model with dimensions, (7.2) 1N4007 used in the project

- **Power Supply**: 5V regulated supply for the microcontroller and analog circuitry.

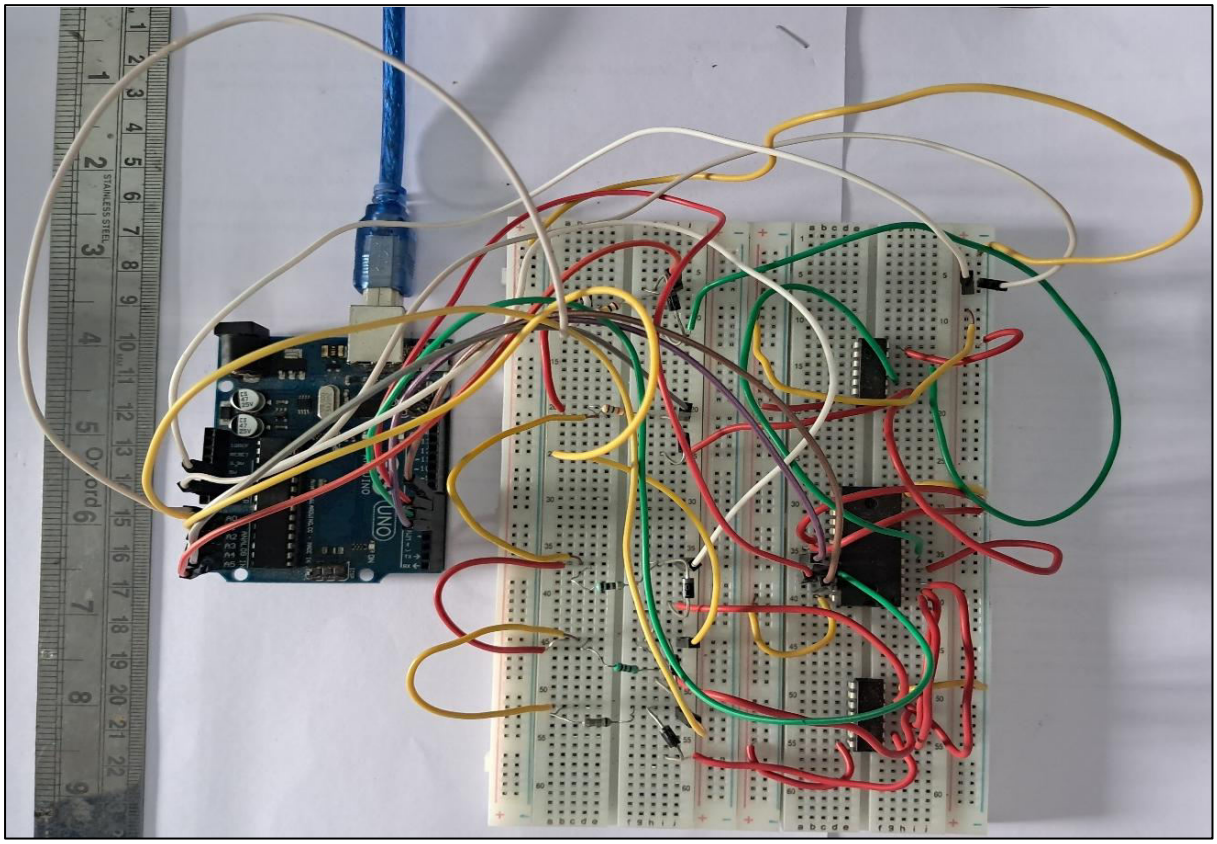


Fig 8: Prototype of the Auto Ranging Ohmmeter

Auto-ranging Algorithm

The auto-ranging mechanism ensures selection of the most appropriate reference resistor based on the measured voltage. We use the potential divider rule to calculate the unknown resistance. The algorithm is implemented as follows:

Vx is the voltage at the junction between reference resistance and unknown resistance. Vr is the voltage after diode or voltage before the reference resistor.

Step 1: BEGIN

Step 2: Declare Res [ ] = {2,177,987,14780,98000}, Vol [ ] = {0,0,0,0,0}, Vol_at_Ref [ ] = {0,0,0,0,0}; Vol_diff [ ] = {0,0,0,0,0}.

Step 3: For S = 1 to 5

I. digitalOut(S);
II. Vol(S) = analogRead(Vx);
III. Vol_at_Ref(S) = analogRead(Vr)
IV. Vol_diff(S) = (Vol_at_Ref(S)/2) - Vol(S);

End For

Step 4: Index = Find_Index_of_Least_Vol_Diff(Vol_diff[ ]);

Step 5: Ref_Res = Res (Index);

Step 6: Rx = (Vol(Index)/(Vol_at_Ref(Index) - Vol(Index)))* Ref_Res;

Step 7: END

Arduino Code:

```
const int C=5;
const int B=6;
const int A=7;
const int D=4;
float CF=0.004887585;
float res[5]={2,177,987,14760,98000};
float vol[5]={0.0,0.0,0.0,0.0,0.0};
float vol_diff[5]={0.0,0.0,0.0,0.0,0.0};
float ref_vol[5]={0.0,0.0,0.0,0.0,0.0};  //voltage coming instead of 5
const int dl =4000;
void setup() {
  pinMode(C,OUTPUT);
  pinMode(B,OUTPUT);
  pinMode(A,OUTPUT);
  pinMode(D,OUTPUT);
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  pinMode(A2,INPUT);
  pinMode(A3,INPUT);
  pinMode(A4,INPUT);
  pinMode(A5,INPUT);


  Serial.begin(9600);


  digitalWrite(D,0);
  digitalWrite(C,0);
  digitalWrite(B,0);
  digitalWrite(A,0);
  vol[0]=(CF*analogRead(A0));
  Serial.println(vol[0]);
  ref_vol[0]=(CF*analogRead(A1));
```

```
 vol_diff[0]=abs((ref_vol[0]*0.5)-vol[0]);
 Serial.println(vol_diff[0]);
 delay(dl);

 digitalWrite(D,0);
 digitalWrite(C,0);
 digitalWrite(B,0);
 digitalWrite(A,1);
 vol[1]=CF*analogRead(A0);
 Serial.println(analogRead(A0)*CF);
 ref_vol[1]=(analogRead(A2)*CF);
 vol_diff[1]=abs((ref_vol[1]*0.5)-vol[1]);
 Serial.println(vol_diff[1]);
 delay(dl);


 digitalWrite(D,0);
 digitalWrite(C,0);
 digitalWrite(B,1);
 digitalWrite(A,0);
 vol[2]=CF*analogRead(A0);
 Serial.println(analogRead(A0)*CF);
 ref_vol[2]=(analogRead(A3)*CF);
 vol_diff[2]=abs((ref_vol[2]*0.5)-vol[2]);
 Serial.println(vol_diff[2]);
 delay(dl);


 digitalWrite(D,0);
 digitalWrite(C,0);
 digitalWrite(B,1);
 digitalWrite(A,1);
 vol[3]=CF*analogRead(A0);
 Serial.println(analogRead(A0)*CF);
 ref_vol[3]=(analogRead(A4)*CF);
 vol_diff[3]=abs((ref_vol[3]*0.5)-vol[3]);
 Serial.println(vol_diff[3]);
 delay(dl);


 digitalWrite(D,0);
 digitalWrite(C,1);
 digitalWrite(B,0);
 digitalWrite(A,0);
 vol[4]=CF*analogRead(A0);
 Serial.println(analogRead(A0)*CF);
 ref_vol[4]=(analogRead(A5)*CF);
 vol_diff[4]=abs((ref_vol[4]*0.5)-vol[4]);
 Serial.println(vol_diff[4]);
 delay(dl);

int index = findIndexOfLowestElement(vol_diff, 5);
float r_ref=res[index];
float v_x=vol[index];
float ref_voltage=ref_vol[index];  // calculate v_x
```

```
89
90  Serial.println(r_ref);
91  Serial.println(v_x);
92
93  float r_x;
94
95  r_x=(v_x*(r_ref))/(ref_voltage-v_x);
96  Serial.print("r_x=");
97  Serial.println(r_x);
98  }
99
100 void loop() {
101 }
102
103 int findIndexOfLowestElement(float arr[], int size) {  // function for finding the index with lowest voltage difference
104   // Finding the index of the lowest element
105   int minIndex = 0;
106   for (int i = 1; i <= size-1; i++) {
107     if (arr[i] < arr[minIndex]) {
108       minIndex = i;
109     }
110   }
111   return minIndex;
112 }
```

PCB Layout

Using Easy EDA software we created a PCB. To build the PCB we have first created to Schematic of our PCB. Then placed the components inside a rectangle of 60.58mm x 35.23mm and routed the connections in both top layer and bottom layer.

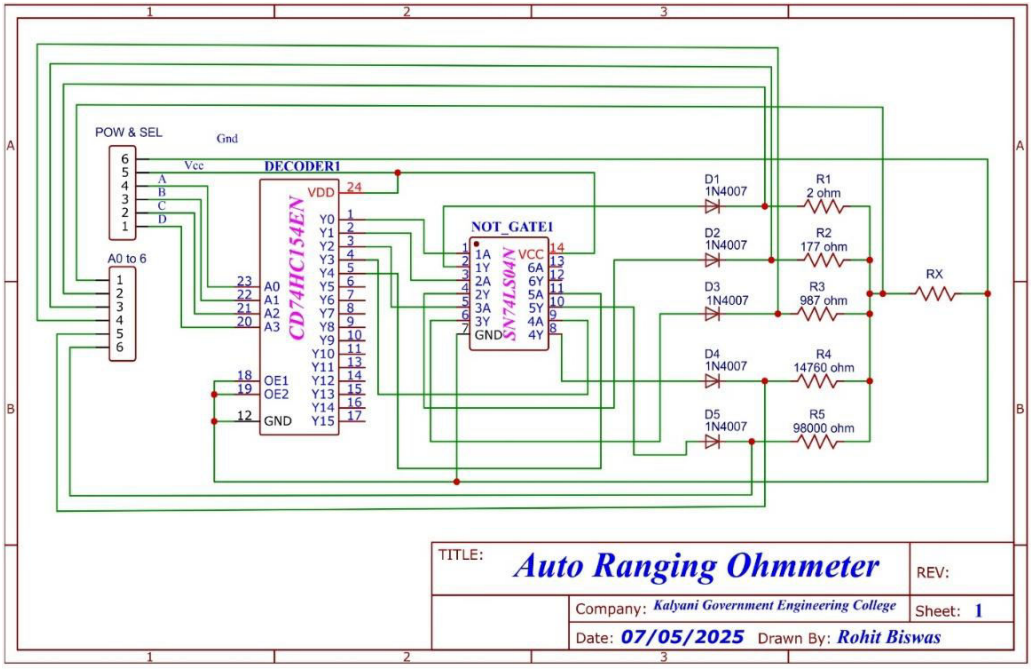


Fig 9: Schematic Diagram of PCB

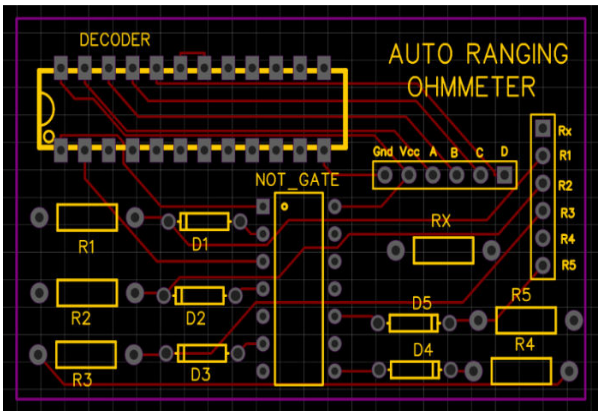


Fig 10.1

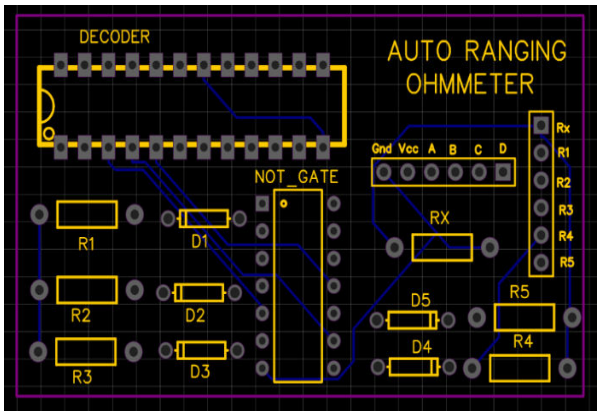


Fig 10.2

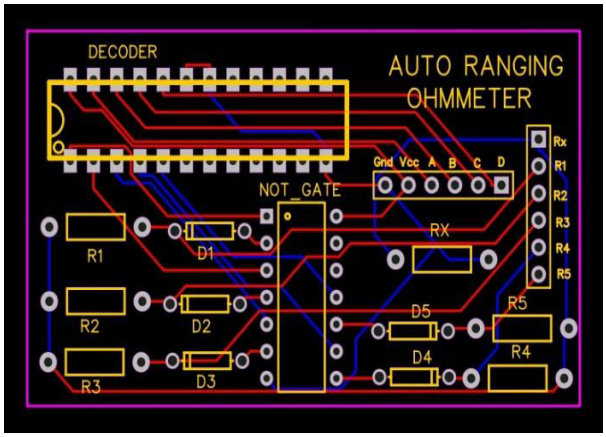


Fig 10.3

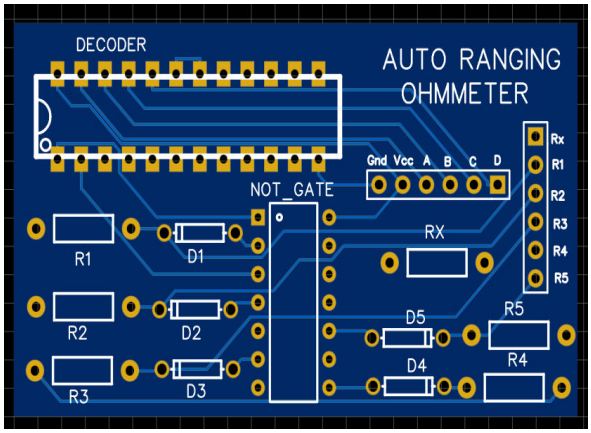


Fig 10.4

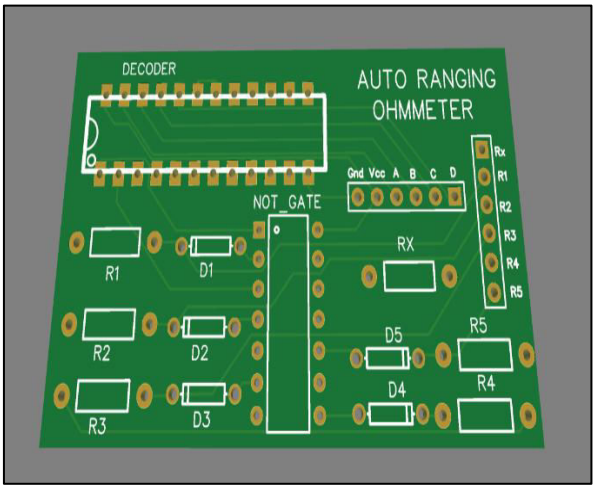


Fig 10.5

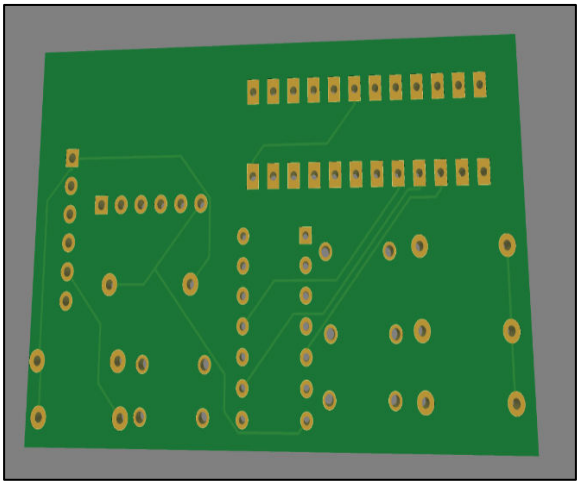


Fig 10.6

Fig 10: (10.1) Top Layer of the PCB, (10.2) Bottom Layer of the PCB, (10.3) Complete PCB after Component placement and Routing, (10.4) 2D view of the PCB, (10.5) 3D view of Top side of the PCB, (10.6) 3D view of Bottom side of the PCB

The information regarding the PCB is following:

- **Size:** 60.58mm x 35.23mm
- **Signal Layers:** 2
- **Components:** 15
- **Pads:** 72
- **Surface Pads:**0
- **Plated Through-hole Pads:** 72
- **None Plated Through-hole Pads:** 0
- **Holes:** 0
- **Vias:** 0
- **Length of Tracks:** 736.25mm
- **Copper Areas:** 0

Results

TABLE 2: Auto-ranging Ohmmeter Measurement of Test Resistances in Hardware

Serial Number	Real Value (Ohm)	Measured Value in Hardware (Ohm)	Error (in %)
1	0.8	0.55	**31.25**
2	23.5	30.74	**30.80851064**
3	147.5	147.43	**0.047457627**
4	263	265.23	**0.847908745**
5	325	327	**0.615384615**
6	466	462.91	**0.663090129**
7	554	550.48	**0.635379061**
8	659	658	**0.151745068**
9	675	675.61	**0.09037037**
10	980	984.77	**0.486734694**
11	1460	1480.5	**1.404109589**
12	2610	2611	**0.038314176**
13	3830	3846.08	**0.419843342**
14	9850	8754.21	**11.12477157**
15	14000	14476.15	**3.401071429**
16	23000	23223.49	**0.971695652**
17	26000	26093.57	**0.359884615**
18	46000	46235.11	**0.511108696**
19	54000	53851.49	**0.275018519**
20	73000	68884	**5.638356164**
21	98000	92735	**5.37244898**
22	266000	257203.09	**3.307109023**
23	503000	480260.96	**4.520683897**
24	510000	480260.96	**5.831184314**
25	666000	625224.87	**6.122391892**
26	804000	751036.31	**6.587523632**
27	1000200	929385.43	**7.080040992**
28	2000000	1824001.62	**8.799919**

TABLE 3: Auto-ranging Ohmmeter Measurement of Test Resistances in Software

Serial Number	Real Value (Ohm)	Measured Value in Software (Ohm)	Error (in %)
1	0.8	0.81	**1.25**
2	23.5	23.13	**1.574468085**
3	147.5	147.28	**0.149152542**
4	263	263.81	**0.307984791**
5	325	324.29	**0.218461538**
6	466	465.37	**0.135193133**
7	554	553.95	**0.009025271**
8	659	659.92	**0.139605463**
9	675	676.13	**0.167407407**
10	980	977.88	**0.216326531**
11	1460	1460.76	**0.052054795**
12	2610	2615.55	**0.212643678**
13	3830	3822.58	**0.193733681**
14	9850	9830.92	**0.193705584**
15	14000	13966.45	**0.239642857**
16	23000	22992.34	**0.033304348**
17	26000	25975.81	**0.093038462**
18	46000	45795.85	**0.443804348**
19	54000	53694.12	**0.566444444**
20	73000	72805.3	**0.266712329**
21	98000	97365.01	**0.64794898**
22	266000	262888.9	**1.169586466**
23	503000	491264.7	**2.33306163**
24	510000	495727.4	**2.79854902**
25	666000	646162.6	**2.978588589**
26	804000	775599.9	**3.532350746**
27	1000200	957470.9	**4.272055589**
28	2000000	1788002	**10.5999**

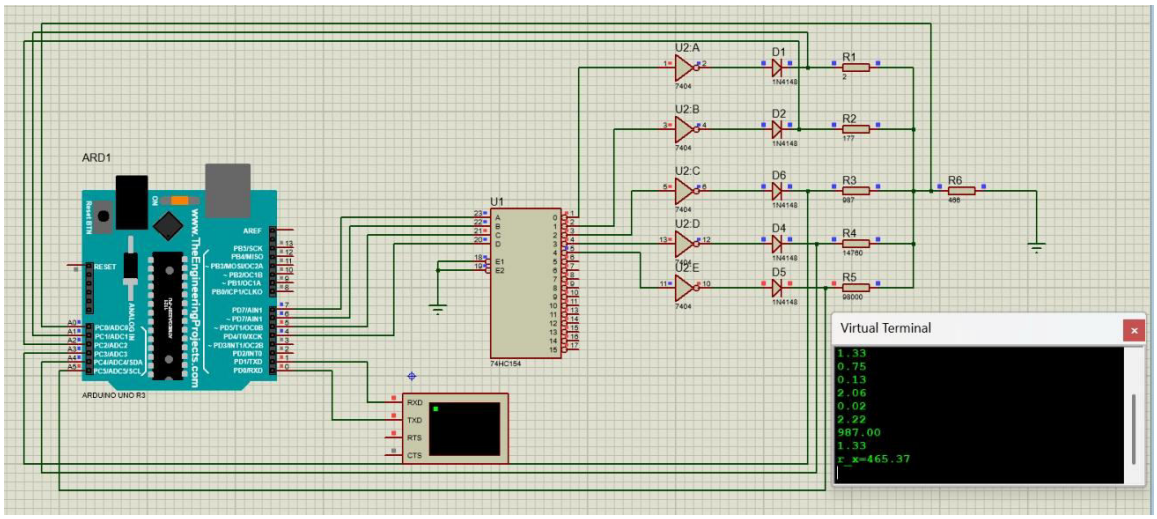


Fig 11.1

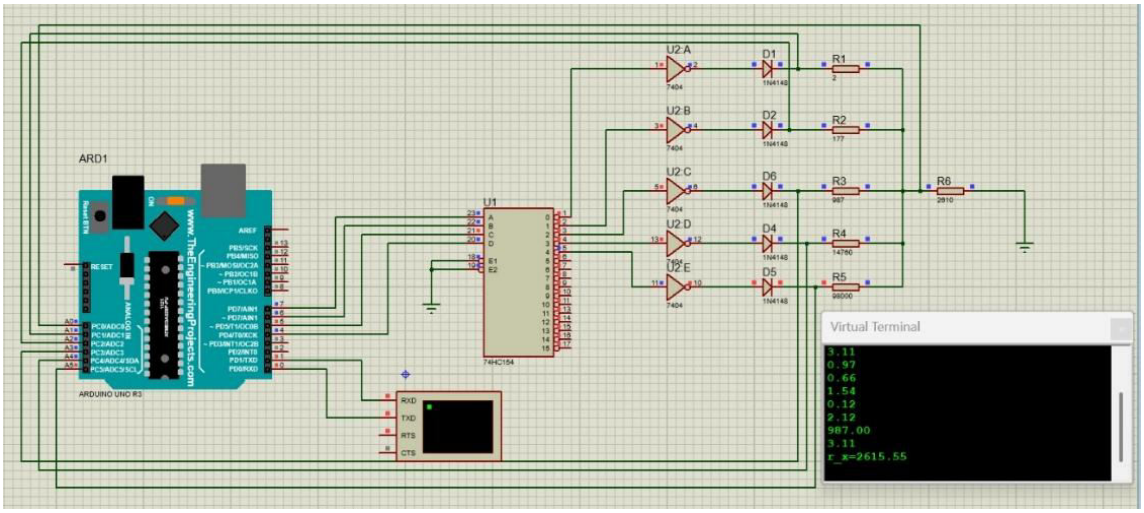


Fig 11.2

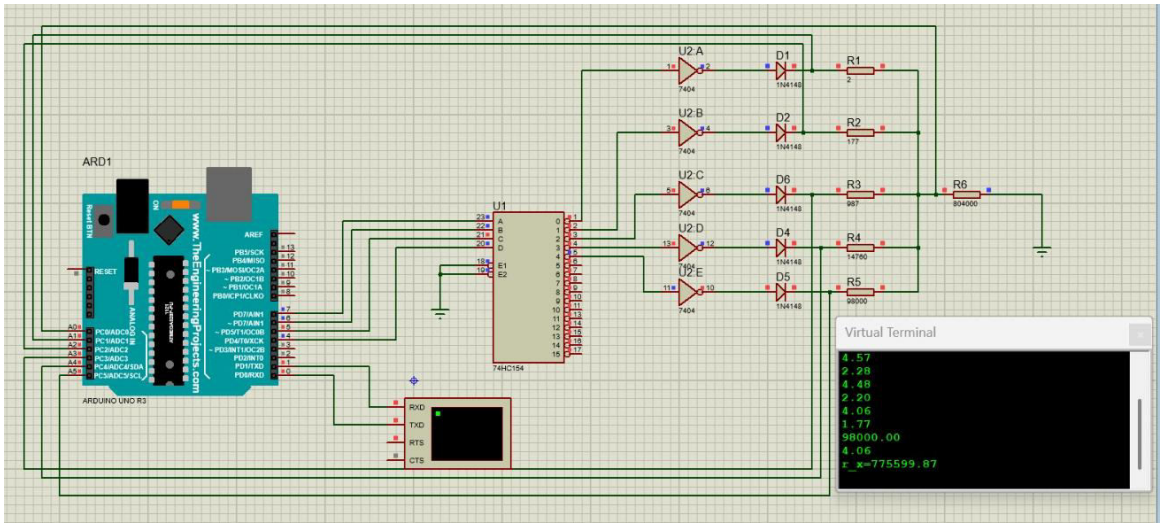


Fig 11.3

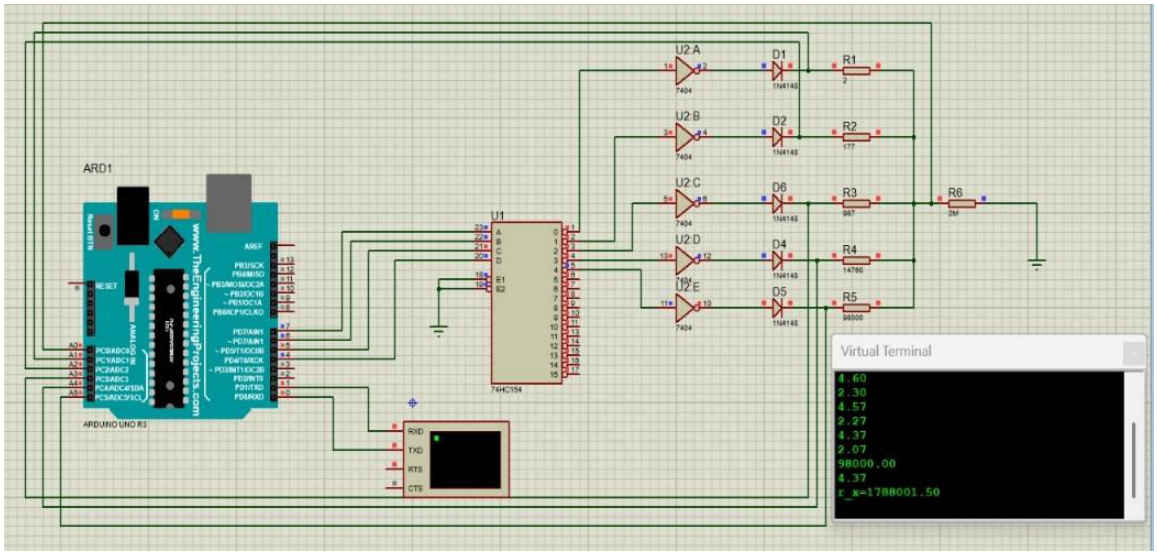


Fig 11.4

Fig 11: Measuring (11.1) 466 ohm, (11.2) 2610 ohm, (11.3) 804k ohm, (11.4) 2M ohm in Proteus 8

DISCUSSION

In this project, the ohmmeter measures the resistances almost correctly in range 140 ohm -54 kiloohm with minimum error. Beyond this range the error increases but remains under 6 percent. But in the measurement of very high resistances (>510k ohm) or very low resistances the error is significant.

Some factors which cause these significant errors can be:

1. In case of the measurement of high resistances (> 510k) the internal resistance of the inbuilt Analog to Digital Converter (ADC) of Arduino Uno comes into play. Thus, error in measurement increases.
2. Due to 10-bit ADC in Arduino we got a resolution of 4.88 mV. But higher resolution is needed for calculating accurate value of unknown resistance.
3. The 5 V source of Arduino Uno does not always provide constant 5 V.
4. The voltage at the output of the PN-junction diode is expected to be constant when the corresponding reference line is selected i.e. the PN diode is on. But we have observed that the output voltage of the diode did not remain at a fix value but varies as the unknown or test resistance values change. Even if the test resistance value is fixed the voltage varies with the reference resistances.
5. In the project, the code uses a delay of 4000 ms i.e. each reference line is ON for 4 s. We have observed that if the delay was given less than 2 s then erroneous values were given by the ohmmeter. Hence, we kept a delay of 4000 ms.

The following two bar graphs shows the error (in percentage) in measured value in hardware and software respectively with original resistance value.

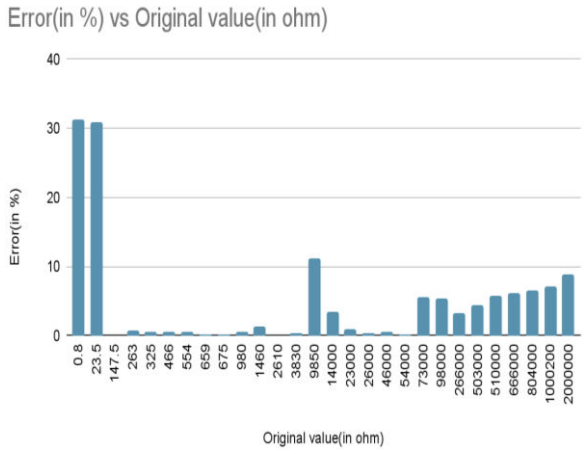


Fig 12.1

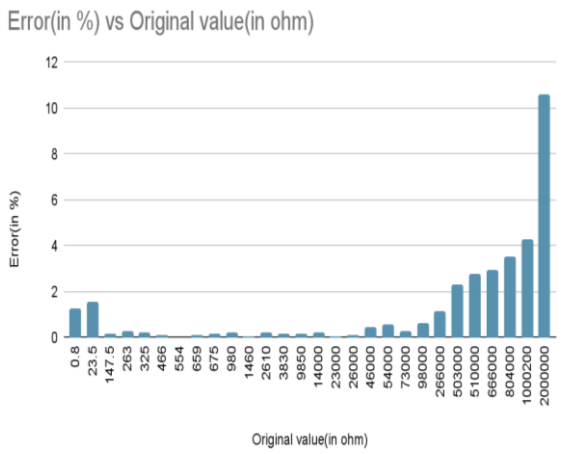


Fig 12.2

Fig 12: Measuring (12.1) Error in measurement of resistance using hardware, (12.2) Error in measurement of resistance using Proteus 8 software

CONCLUSION

In this project, we have built an auto-ranging ohmmeter using Arduino Uno microcontroller. Here we have implemented the concept of voltage division rule to measure the voltage across the unknown resistor and to determine the resistance of it from the measured voltage between the unknown and reference resistor. From the graphs obtained we can say that our auto-ranging ohmmeter measures resistors having resistances in the range of 140 ohm- 54 kiloohm with minimal error. The simulation results obtained are almost same as that of the measured resistances in this range. More accurate result can be obtained if we use ESP32 in place of Arduino Uno because ESP has 12-bit ADC, thus provides a resolution of 1.22 mV. We also need to eliminate the loading effect between the microcontroller's inbuilt ADC and the unknown resistive load which plays a crucial role in measuring high resistance values (> 600k ohm). By solving these two issues, we can implement an auto ranging ohmmeter which can measure a large range of resistances with higher accuracy and less error.

REFERENCES

[1] Texas Instruments, *CD74HC154: 4-line to 16-line decoder/demultiplexer datasheet*. [Online]. Available: https://www.ti.com/lit/ds/symlink/cd74hc154.pdf

[2] Texas Instruments, *SN74LS04N Hex Inverter Datasheet*, [online]. Available: https://www.ti.com/lit/ds/symlink/sn74ls04.pdf

[3] onsemi, *1N4001–1N4007: Axial Lead Standard Recovery Rectifiers*, [online]. Available: https://www.onsemi.com/pdf/datasheet/1n4001-d.pdf