Building a combination create machine for four probe measurements

S W D K R M Manamendra

12th August 2024

Abstract

This project outlines the design and implementation of an automated four-probe measurement device to streamline sheet resistance evaluations for thin conductive materials. Leveraging an ESP32 microcontroller, a 16-channel relay module, and a Wi-Fi-enabled web interface, the system automates probe reconfiguration to minimize manual intervention. It combines robust hardware integration with a user-friendly software interface, supporting reliable and scalable measurements. The device integrates the Van der Pauw and four-probe methods, offering precise, repeatable results. Addressing challenges like probe failure and time inefficiencies, this innovation enhances high-throughput material characterization, vital for advancements in semiconductors, photovoltaics, and flexible electronics.

1 Introduction

The measurement of electrical properties of thin films and conductive materials is fundamental in the characterization and development of modern electronic devices. Among various methods, the **Van der Pauw method** and the **four-probe technique** stand out as essential tools for accurately determining the sheet resistance of conductive layers. This report focuses on the design and development of a four-probe measurement device that is crucial for sheet resistance calculations, integrating principles of both methods to enhance accuracy and reliability.

Importance of Sheet Resistance Measurements

Sheet resistance (R_s) is a critical parameter for evaluating the electrical performance of thin films used in applications such as semiconductors, photovoltaics, and flexible electronics. It describes the resistance of a square of the material and is typically expressed in ohms per square ($\Omega/$). Accurate measurement of R_s is essential for ensuring the quality and functionality of materials in electronic components Van der Pauw [1958], Smits [1958].

The Four-Probe Technique

The **four-probe method** is widely regarded as a reliable technique for measuring the electrical resistivity and sheet resistance of materials. By using four aligned probes in contact with the sample surface:

1. A current is passed through the outer probes, while the voltage is measured across the inner probes.

2. The setup minimizes the influence of contact resistance, yielding precise measurements Schroder [2006], ast [2004].

The resistivity (ρ) is then calculated as:

$$\rho = \frac{\pi s}{\ln(2)} \left(\frac{V}{I} \right) F$$

where:

- s is the spacing between adjacent probes,
- V/I is the measured resistance,
- F is a correction factor dependent on the sample geometry and probe placement Kelley et al. [1984], Hoffmann [1996].

The Van der Pauw Method

For samples with arbitrary shapes, the **Van der Pauw method** offers a complementary approach to sheet resistance measurement. It requires placing four small ohmic contacts on the sample's edge and measuring resistances (R_A and R_B) in two orthogonal configurations. The sheet resistance is calculated using the Van der Pauw equation:

$$e^{-\pi R_A/R_s} + e^{-\pi R_B/R_s} = 1$$

This method assumes that the sample is homogeneous, isotropic, and thin, with uniform thickness Van der Pauw [1958], Gupta [2003].

Research Problem

While the four-probe method is highly effective, achieving accurate measurements often requires performing multiple configurations by re-wiring or swapping probes. This process can be time-consuming, especially when dealing with large numbers of samples. Additionally, frequent re-wiring introduces significant risks of probe failure and contact degradation, further complicating the measurement process. These challenges highlight the need for an automated system that can handle probe reconfiguration efficiently, without manual intervention.

To address these limitations, this project aims to design and develop a four-probe measurement device with automated re-wiring capabilities. The proposed system seeks to enhance the accuracy, reliability, and scalability of sheet resistance measurements, particularly for high-throughput applications Hoffmann [1996], Schroder [2006].

Integration and Objectives

The project presented here combines the robustness of the four-probe setup with the flexibility of the Van der Pauw method to develop a measurement device capable of delivering accurate sheet resistance values across a variety of samples. This device employs precise alignment of probes, robust contact mechanisms, and advanced data acquisition systems to ensure reliability and repeatability in measurements Hoffmann [1996], Gupta [2003].

Through the integration of these methodologies and the incorporation of automation, the developed system provides a versatile platform for characterizing the electrical properties of diverse materials, contributing to advancements in material science and electronic device fabrication Schroder [2006], Kelley et al. [1984].

2 Methodology

This section describes the methodology employed to develop an automated four-probe measurement system using an ESP32 microcontroller and a 12VDC 16-channel relay module. The goal of this project was to automate the process of switching probe patterns in a four-probe measurement setup for efficient and accurate sheet resistance calculations. The system is controlled through a Wi-Fi-enabled web interface, providing remote operation and flexibility in managing different switching configurations.

2.1 Hardware Components

The primary components of the system include:

- **ESP32 Microcontroller:** This is used to handle the logic for switching between probe configurations and controlling relay modules. It is chosen for its robust Wi-Fi capabilities, which allow for remote control via a web interface.
- **12VDC 16-Channel Relay Module:** This module is used to control the switching of probes. Each channel on the relay corresponds to a specific output pin on the ESP32, enabling or disabling different probes.
- **Probes:** The system uses a set of 16 probes, each corresponding to a GPIO pin on the ESP32. These probes are controlled via the relay module to establish different measurement configurations.
- **Wi-Fi Network:** The ESP32 is connected to two Wi-Fi networks, enabling stable connectivity and ensuring that the system can operate even if one network fails.

2.2 System Design

The system was designed to handle multiple switching patterns for the four-probe measurement technique. It offers two primary modes of operation: manual control through physical switches and automated switching via the ESP32's web interface. The switching configurations were predefined in the system, and these configurations can be activated remotely.

2.2.1 Wiring and Relay Configuration

The ESP32's GPIO pins are connected to a 16-channel relay module. Each GPIO pin controls one of the relays, which in turn controls the corresponding probe. The 16 probes are distributed across the relay channels in a manner that allows for flexible switching between different combinations.

2.2.2 Wi-Fi Communication and Web Interface

The ESP32 connects to a Wi-Fi network using the `WiFi.h` library, and it hosts a local web server on port 80. The web server serves a simple HTML interface that displays buttons for controlling probe configurations (e.g., "Reading 1", "Reading 2", etc.). Users can select the configuration through the web interface, which sends HTTP GET requests to the ESP32. Upon receiving these requests, the ESP32 interprets the commands and switches the appropriate probes on or off.

The ESP32 also provides the functionality to connect to two Wi-Fi networks, ensuring a stable connection. If the first network fails, it attempts to connect to a second network, ensuring that the system remains operational even in varying network conditions.

2.2.3 Code Explanation

The core of the system is the following code, which facilitates the switching of probes and the web interface.

- **Wi-Fi Setup:** The system attempts to connect to two Wi-Fi networks (`ssid1` and `ssid2`) with a timeout of 10 seconds per network.

```
WiFi.begin(ssid1, password1);
WiFi.begin(ssid2, password2);
```

- **Web Server:** Once connected to Wi-Fi, the system initializes a web server to listen for incoming HTTP requests. The ESP32 then serves an HTML page with control buttons for selecting different probe configurations.

```
server.begin();
```

- **Switching Logic:** The function `handleReading()` is responsible for switching specific probes based on user selection. Each probe is controlled by setting its corresponding GPIO pin to LOW (active) or HIGH (inactive).

```
pinMode(2, OUTPUT); digitalWrite(2, LOW); outputState[0] = "on";
```

- **Web Interface:** The HTML interface includes buttons for each probe configuration. When a user clicks a button, it sends a corresponding HTTP request to the ESP32, which triggers the appropriate probe switching.

```
client.println("<a href=\"/reading1\"><button class=\"button\">Reading 1</button></a>'
```

2.3 Automation and Switching Patterns

The system allows users to select from predefined switching patterns, such as Reading 1, Reading 2, etc., through the web interface. Each reading corresponds to a unique combination of probes that are activated (set to LOW) to create a specific measurement configuration.

The flexibility of the ESP32 allows for the expansion of this system to accommodate any desired combination of probe switching. This feature was further developed to support multiple Wi-Fi networks and ensure seamless communication and control across different network environments.

2.4 Challenges and Solutions

- **Probe Failures:** During the development, several issues were encountered with probe failure due to frequent re-wiring. To address this, the system was designed to automate the process of switching probe configurations, eliminating the need for manual re-wiring.
- **Manual Control:** Although the system was designed for automation, switches were incorporated to allow manual control for debugging and troubleshooting purposes.

2.5 System Workflow

1. **Initial Setup:** Upon powering on, the ESP32 connects to the designated Wi-Fi network.
2. **Web Interface:** The user accesses the web interface through a browser and selects the desired probe configuration.
3. **Probe Switching:** Based on the selected configuration, the ESP32 activates the relevant relays, adjusting the probe connections.
4. **Measurement:** Once the probes are in the correct configuration, sheet resistance measurements are performed.

2.6 Images and Diagrams

To enhance the understanding of the system design, here are some sample images:

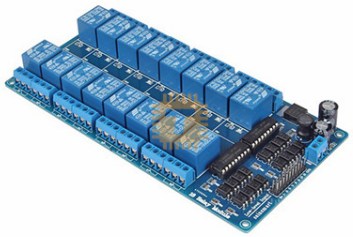

Figure 1: 16-Channel Relay Module

Figure 2: ESP32 Microcontroller

By following this methodology, the automated four-probe measurement system provides an efficient solution to the problem of manual switching, enabling precise, high-throughput measurements with minimal human intervention.

The code written for this task is given below.

```
1
2  // Load Wi-Fi library
3  #include <WiFi.h>
4
5  // Replace with your network credentials for both networks
6  const char* ssid1 = "Rangana Note 20";
7  const char* password1 = "12345678";
8
9  const char* ssid2 = "UOC_Staff";
10 const char* password2 = "admin106";
11
12 // Set web server port number to 80
13 WiFiServer server(80);
14
15 // Variable to store the HTTP request
16 String header;
17
18 // Auxiliar variables to store the current output state
19 String outputState[16] = { "off", "off", "off", "off", "off", "off", "off"
       , "off", "off", "off", "off", "off", "off", "off", "off", "off" };
20
21 // Assign output variables to GPIO pins
22 const int outputPins[16] = { 2, 4, 5, 12, 13, 14, 15, 16, 17, 18, 21, 22,
       25, 32, 26, 27 };
23
24 // Current time
25 unsigned long currentTime = millis();
26 // Previous time
27 unsigned long previousTime = 0;
28 // Define timeout time in milliseconds (example: 2000ms = 2s)
29 const long timeoutTime = 2000;
30
31 void connectWiFi() {
32   Serial.println("Trying to connect to the first network...");
33   WiFi.begin(ssid1, password1);
34
35   // Try to connect to the first network for 10 seconds
36   unsigned long startAttemptTime = millis();
37   while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime <
         10000) {
38     Serial.print(".");
39     delay(500);
40   }
41
42   if (WiFi.status() == WL_CONNECTED) {
43     Serial.println("\nConnected to the first network!");
44     Serial.print("IP address: ");
45     Serial.println(WiFi.localIP());
46   } else {
47     Serial.println("\nFailed to connect to the first network. Trying the
           second network...");
48     WiFi.begin(ssid2, password2);
```

```
49
50      // Try to connect to the second network for another 10 seconds
51      startAttemptTime = millis();
52      while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime <
            10000) {
53        Serial.print(".");
54        delay(500);
55      }
56
57      if (WiFi.status() == WL_CONNECTED) {
58        Serial.println("\nConnected to the second network!");
59        Serial.print("IP address: ");
60        Serial.println(WiFi.localIP());
61      } else {
62        Serial.println("\nFailed to connect to both networks.");
63      }
64    }
65  }
66
67  void setup() {
68    Serial.begin(115200);
69
70    // Set all pins to high impedance mode (input)
71    for (int i = 0; i < 16; i++) {
72      pinMode(outputPins[i], INPUT);
73    }
74
75    // Call the Wi-Fi connection function
76    connectWiFi();
77
78    // Start web server if connected to Wi-Fi
79    if (WiFi.status() == WL_CONNECTED) {
80      server.begin();
81    }
82  }
83
84  void handleReading(int reading) {
85    // Set all pins to high impedance mode
86    for (int i = 0; i < 16; i++) {
87      pinMode(outputPins[i], INPUT);
88      outputState[i] = "off";
89    }
90
91    // Set specific pins to LOW (as output) based on the reading
92    switch (reading) {
93      case 1:
94        pinMode(2, OUTPUT); digitalWrite(2, LOW); outputState[0] = "on";
95        pinMode(14, OUTPUT); digitalWrite(14, LOW); outputState[5] = "on";
96        pinMode(21, OUTPUT); digitalWrite(21, LOW); outputState[10] = "on";
97        pinMode(27, OUTPUT); digitalWrite(27, LOW); outputState[15] = "on";
98        break;
99
100     case 2:
101       pinMode(5, OUTPUT); digitalWrite(5, LOW); outputState[2] = "on";
```

```
      pinMode(16, OUTPUT); digitalWrite(16, LOW); outputState[7] = "on";
      pinMode(17, OUTPUT); digitalWrite(17, LOW); outputState[8] = "on";
      pinMode(32, OUTPUT); digitalWrite(32, LOW); outputState[13] = "on";
      break;

    case 3:
      pinMode(2, OUTPUT); digitalWrite(2, LOW); outputState[0] = "on";
      pinMode(15, OUTPUT); digitalWrite(15, LOW); outputState[6] = "on";
      pinMode(18, OUTPUT); digitalWrite(18, LOW); outputState[9] = "on";
      pinMode(27, OUTPUT); digitalWrite(27, LOW); outputState[15] = "on";
      break;

    case 4:
      pinMode(4, OUTPUT); digitalWrite(4, LOW); outputState[1] = "on";
      pinMode(16, OUTPUT); digitalWrite(16, LOW); outputState[7] = "on";
      pinMode(17, OUTPUT); digitalWrite(17, LOW); outputState[8] = "on";
      pinMode(26, OUTPUT); digitalWrite(26, LOW); outputState[14] = "on";
      break;
  }
}

void loop() {
  WiFiClient client = server.available();  // Listen for incoming clients

  if (client) {  // If a new client connects
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client."); // print a message out in the serial
       port
    String currentLine = ""; // make a String to hold incoming data from
       the client
    while (client.connected() && currentTime - previousTime <= timeoutTime
       ) {  // loop while the client's connected
      currentTime = millis();
      if (client.available()) {  // if there's bytes to read from the
         client,
        char c = client.read();  // read a byte, then
        Serial.write(c);         // print it out the serial monitor
        header += c;
        if (c == '\n') {  // if the byte is a newline character
          if (currentLine.length() == 0) {  // if the current line is
             blank, you've reached the end of the request
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Handle "Reading" buttons
            if (header.indexOf("GET /reading1") >= 0) {
              Serial.println("Reading 1 pressed");
              handleReading(1);
            } else if (header.indexOf("GET /reading2") >= 0) {
              Serial.println("Reading 2 pressed");
              handleReading(2);
```

```
            } else if (header.indexOf("GET /reading3") >= 0) {
              Serial.println("Reading 3 pressed");
              handleReading(3);
            } else if (header.indexOf("GET /reading4") >= 0) {
              Serial.println("Reading 4 pressed");
              handleReading(4);
            }

            // Generate HTML page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=
               device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            client.println("<style>html { font-family: Helvetica; display:
                inline-block; margin: 0px auto; text-align: center;}");
            client.println(".button { background-color: #4CAF50; border:
               none; color: white; padding: 16px 40px;");
            client.println("text-decoration: none; font-size: 30px; margin
               : 2px; cursor: pointer;}");
            client.println(".button2 {background-color: #555555;}");
            client.println("</style></head>");

            client.println("<body><h1>ESP32 GPIO Control</h1>");
            client.println("<p>");
            client.println("<a href=\"/reading1\"><button class=\"button
               \">Reading 1</button></a>");
            client.println("<a href=\"/reading2\"><button class=\"button
               \">Reading 2</button></a>");
            client.println("<a href=\"/reading3\"><button class=\"button
               \">Reading 3</button></a>");
            client.println("<a href=\"/reading4\"><button class=\"button
               \">Reading 4</button></a>");
            client.println("</p>");

            for (int i = 0; i < 16; i++) {
              client.println("<p>GPIO " + String(outputPins[i]) + " -
                 State " + outputState[i] + "</p>");
              if (outputState[i] == "off") {
                client.println("<p><a href=\"/" + String(outputPins[i]) +
                   "/on\"><button class=\"button\">ON</button></a></p>");
              } else {
                client.println("<p><a href=\"/" + String(outputPins[i]) +
                   "/off\"><button class=\"button button2\">OFF</button></
                   a></p>");
              }
            }

            client.println("</body></html>");
            client.println();
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
```

```
            currentLine += c;
          }
        }
      }
      header = "";
      client.stop();
      Serial.println("Client disconnected.");
      Serial.println("");
    }
}
```

Listing 1: ESP32 Code for GPIO Control and Web Server

3 Result

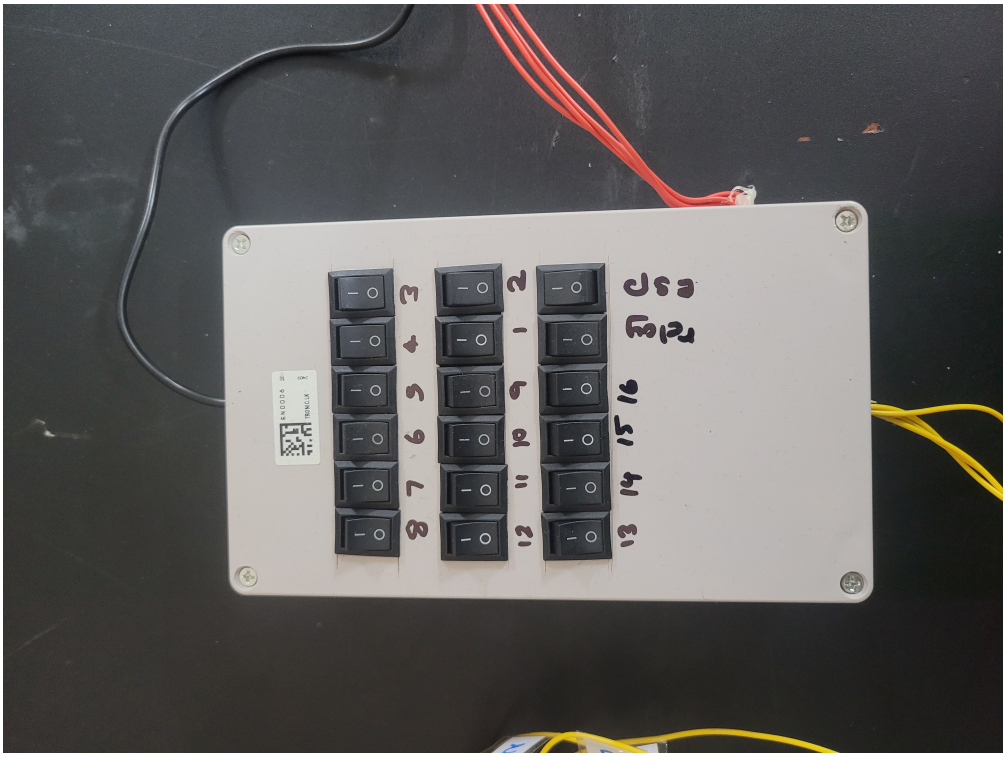

Figure 3: Controller

References

L.J. Van der Pauw. A method of measuring specific resistivity and hall effect of discs of arbitrary shape. *Philips Technical Review*, 20(8):220–224, 1958.

F.M. Smits. Measurement of sheet resistivities with the four-point probe. *Bell System Technical Journal*, 37(3):711–718, 1958.

D.K. Schroder. *Semiconductor Material and Device Characterization*. John Wiley & Sons, 2006.

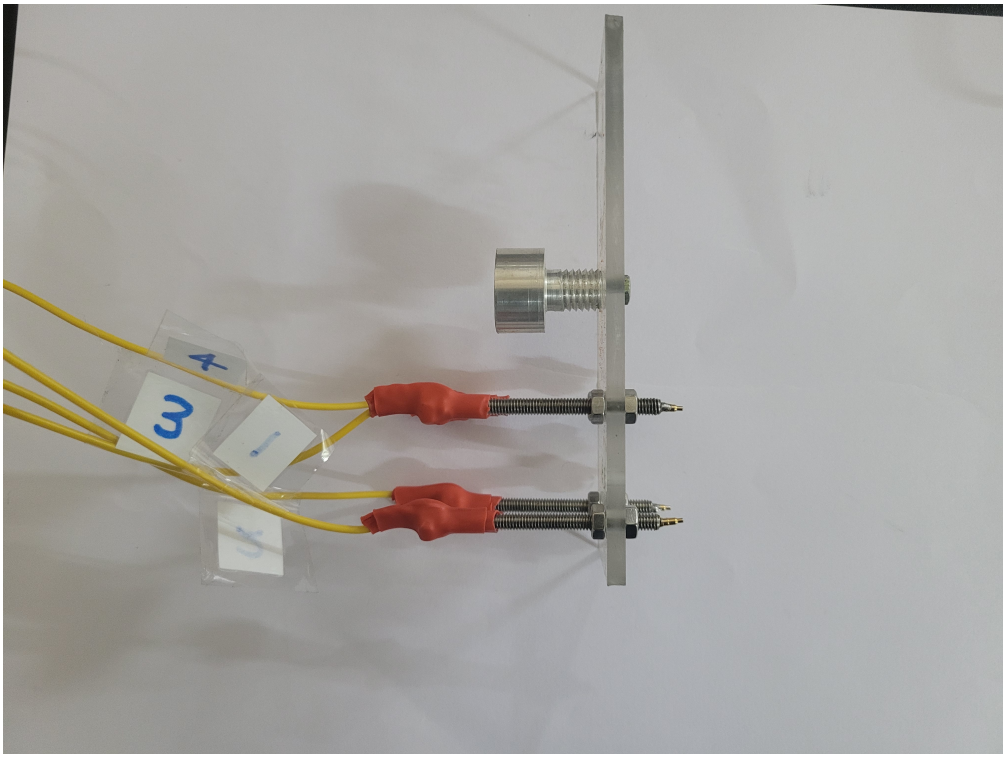

Figure 4: Four probes for surface resistance measurements

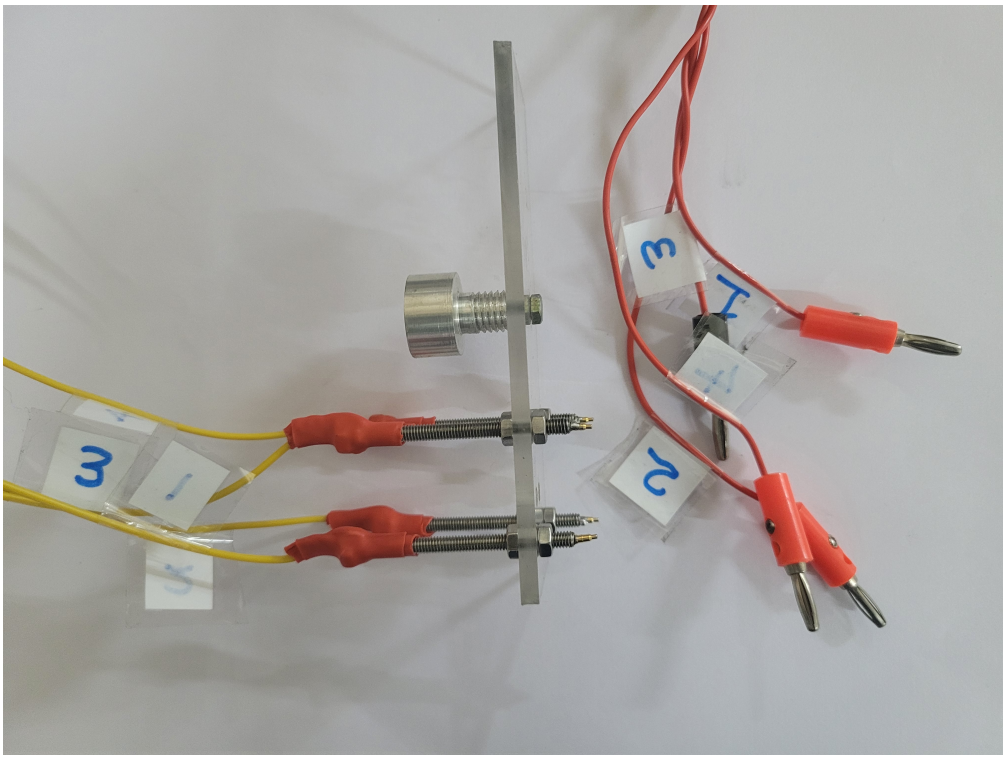

Figure 5: Input and output probes

Standard test methods for measuring resistivity and hall coefficient of materials. (F76-04), 2004.

J.E. Kelley et al. Accuracy of four-probe resistivity measurements on thin films. *Journal of Applied Physics*, 55(7):2853–2856, 1984.

R. Hoffmann. Four-point probe measurements. *Solid State Technology*, 39(2):213–218, 1996.

R.P. Gupta. Sheet resistance calculations using the van der pauw technique. *IEEE Transactions on Electron Devices*, 50(9):1906–1912, 2003.