Laboratorio Avanzado

Universidad Autonoma del Estado de Mexico

Facultad de Ciencias

José Alfredo Sepúlveda Garía
Dr. Juan Carlos Corona Oran
26 de marzo de 2021

Índice general

Antecedentes

Cuando surge una hipótesis la manera de comprobar si esta concuerda con la realidad, es, precisamente, obteniendo datos experimentales por medio de mediciones. Muchas veces estas mediciones deben constar de un gran volumen de datos con la finalidad de asegurar la precisión del experimento. Más aún, el experimento debe ser desarrollado bajo condiciones que garanticen que las mediciones realizadas cuenten con un alto nivel de exactitud. Es por eso que la estadística es la herramienta más importante en cuanto al desarrollo y análisis de los datos obtenidos experimentalmente.
Además de explotar la estadística como herramienta base, es importante considerar que cuando se está realizando un experimento la mayoría de las veces las mediciones serán indirectas, es decir, las variables que se encuentran no son los valores que se buscan fundamentalmente, sino que son valores a sustituir para encontrar los fundamentales; y aquí es donde se debe encontrar la forma matemática que permita obtener el valor deseado a partir de valores medibles experimentalmente.
Esta práctica consistió en realizar un ohmetro integrando conocimientos estadísticos y de circuitos electrónicos. El intervalo de medición abarcó valores de resistencia en el intervalo de 220 a 2200 ohms.

Materiales y métodos

Medición de voltajes de salida del microcontrolador Arduino Uno

Se utilizó una placa de microcontrolador Arduino Uno, mostrado en la figura 1, que cuenta con un microprocesador ATMega 328P, equipada con un conjunto de pines de E/S digitales y analógicas. Para medir los voltajes de salida de los pines A0, A1, A2, A3, A4 y A5 del microcontrolador cada 3 segundos se utilizó el código 1. Una vez medido el voltaje de cada salida, se realizó el histograma 1 en el software libre de análisis estadístico R. Estos histogramas aseguran que las salidas de voltaje se encuentran funcionando correctamente, pues estadísticamente los voltajes medidos representan una distribución Gaussiana. Ahora que se conoce la precisión del funcionamiento del microcontrolador se puede proceder a realizar más mediciones con él.

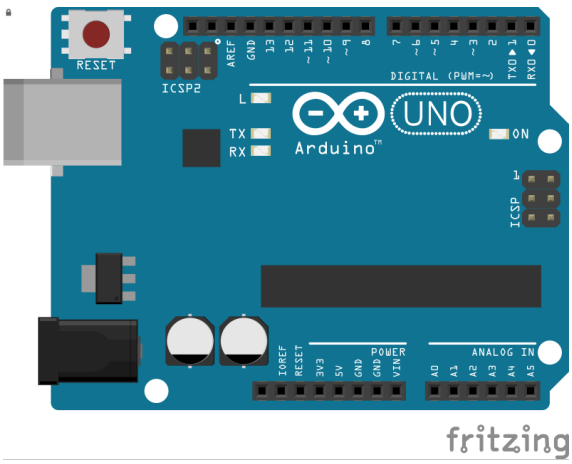


Figura 1: Microcontrolador Arduino Uno

Calibración del Ohmetro

La segunda parte de la práctica consistió en armar un divisor de voltaje como se muestra en la figura 2, con un voltaje de entrada V_e de 5 Volts (que proporciona el microcontrolador), una resistencia de referencia R_1 de 5,6$kOhms$ y se varió otra resistencia R_2 desde 220$Ohms$ hasta 2200$Ohms$ en intervalos de 220$Ohms$ para medir el voltaje que se encontraba entre Gnd y la resistencia de referencia R_1 y la resistencia variable R_2 con el fin de realizar la gráfica 1, y encontrar el polinomio que mejor ajustara a las mediciones tomadas. Los datos obtenidos se encuentran en la tabla 1.

Para encontrar el polinomio que ajustaba mejor a la gráfica 1, se utilizó el software libre de grafiación GNUPlot, con el código 2

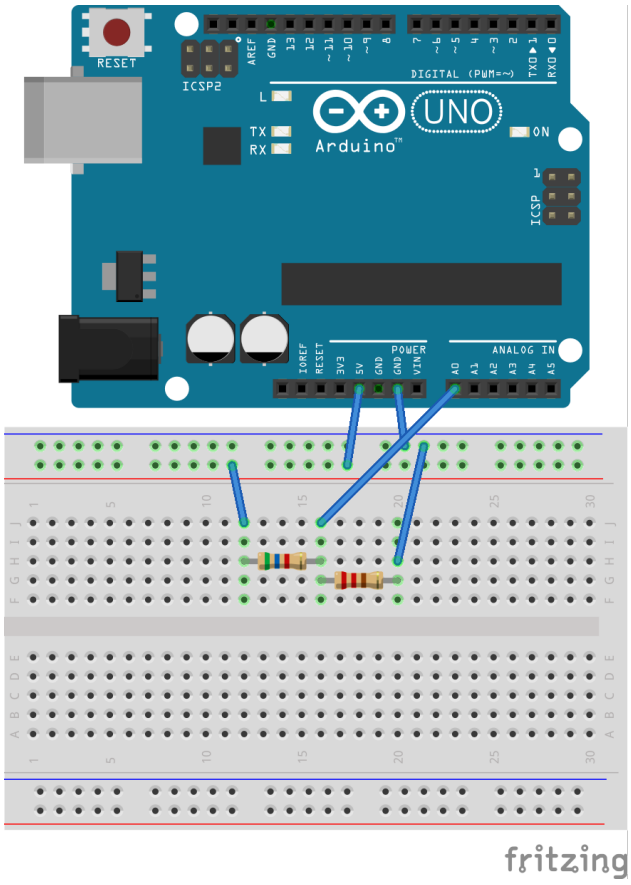
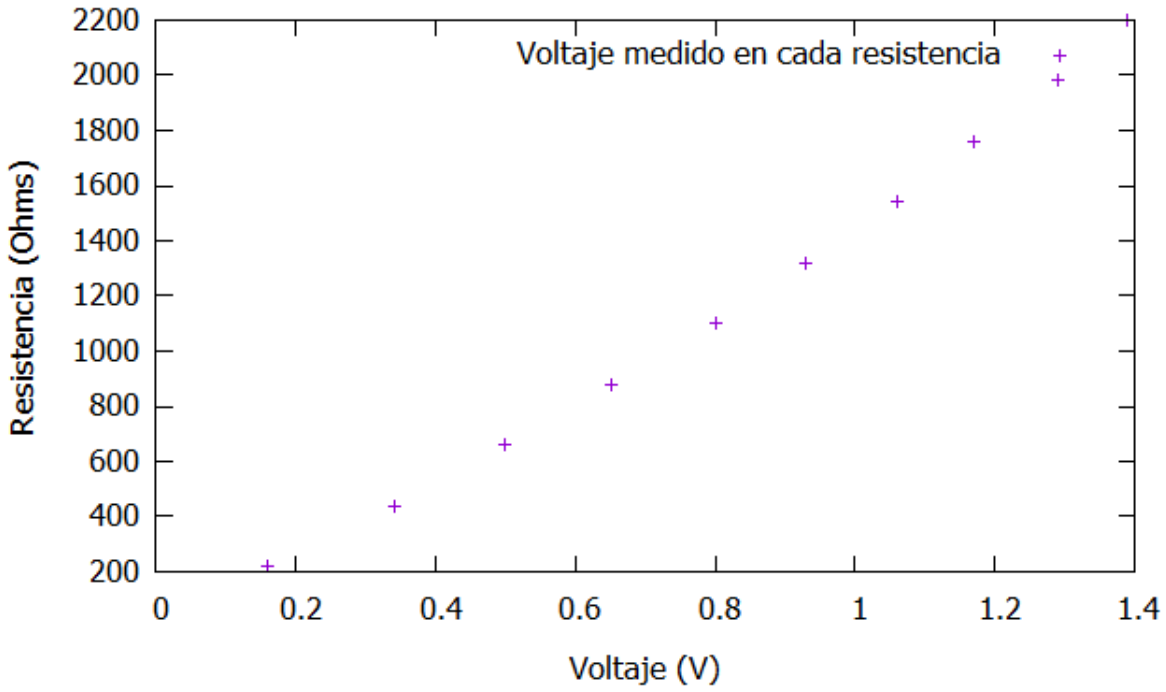


Figura 2: Divisor de voltaje.



Gráfica 1. Medición de voltaje contra resistencia

Medición de resistencias con el Ohmetro

Finalmente, en la plataforma de Arduino se introdujo el código 3, donde se establecieron los parámetros del ajuste a la gráfica 1. El circuito listo para medir resistencias se encuentra en la figura 3.

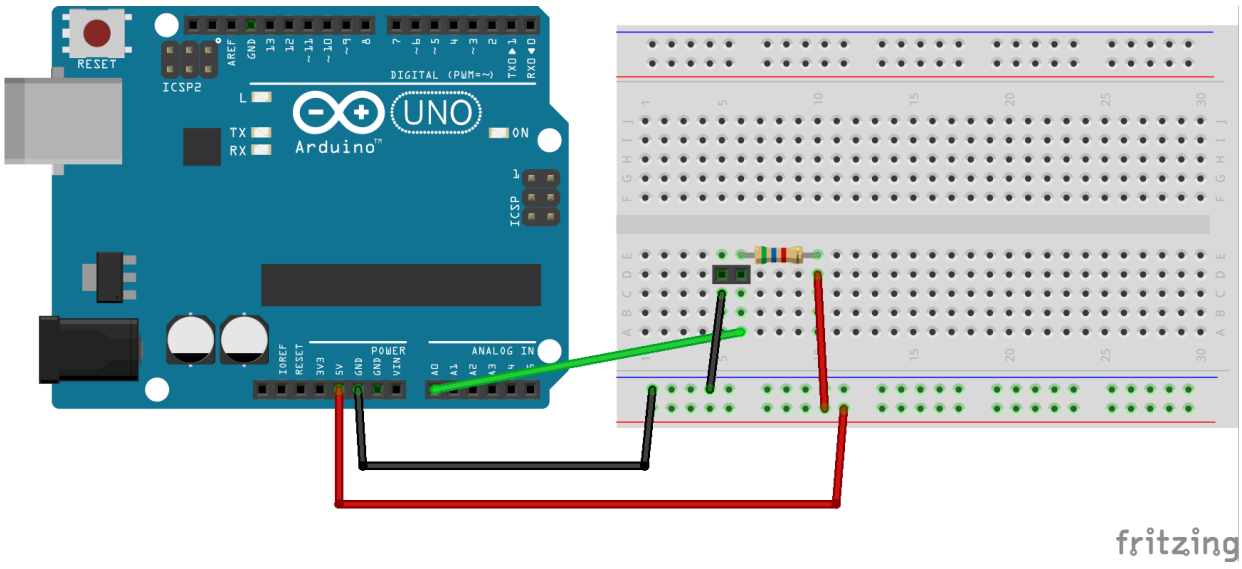
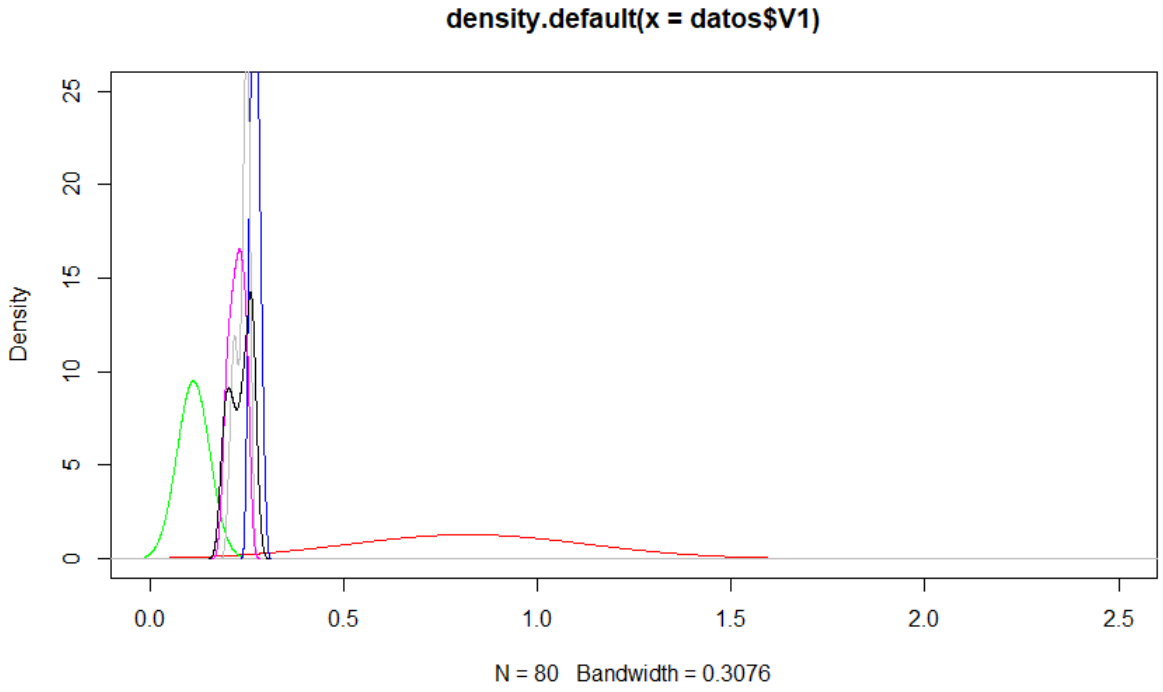


Figura 3. La resistencia a medir debe fue colocada en los pines de color negro ubicados en la protoboard

Resultados

Analisis de voltajes de interferencia

A continuación se muestra el histograma de frencuencias de los voltajes de salida del microcontrolador Arduino. Este resultado permitió asegurar que, dado que la distribución de voltajes en una medición de 800 datos era una distribución gaussiana, la precisión del microcontrolador en la salida de voltajes indicaba el correcto funcionamiento del dispositivo.



Histograma 1. Frecuencias de voltajes medidos en las terminales del microcontrolador

Calibración

El ajuste realizado mediante el código 2 en GNUPlot arrojó los parámetros mostrados en la tabla 2 para el polinomio de segundo orden, y su respectivo error

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a                  = 378.508       +/- 19.08        (5.041%)
b                  = 1012.01       +/- 30.73        (3.037%)
c                  = 52.818        +/- 10.72        (20.3%)
```

Tabla 2. Valores de los parámetros a, b y c encontrados con el uso del software GNUPlot

Medición de resistencias

Con los parámetros hallados integrados al código 3 se procedió a medir resistencias de valores conocidos. La comparación del valor de cada resistencia con el valor medido con el microcontrolador se muestra en la tabla 3

Valor de resistencia (Ohms)	Valor obtenido con el medidor (Ohms)
220	223
440	443
660	665
880	880
1100	1099
1320	1317
1540	1538
1760	1759
1980	1983
2200	2198

Tabla 3. Comparación del valor de la resistencia según el fabricante contra el valor obtenido por el medidor

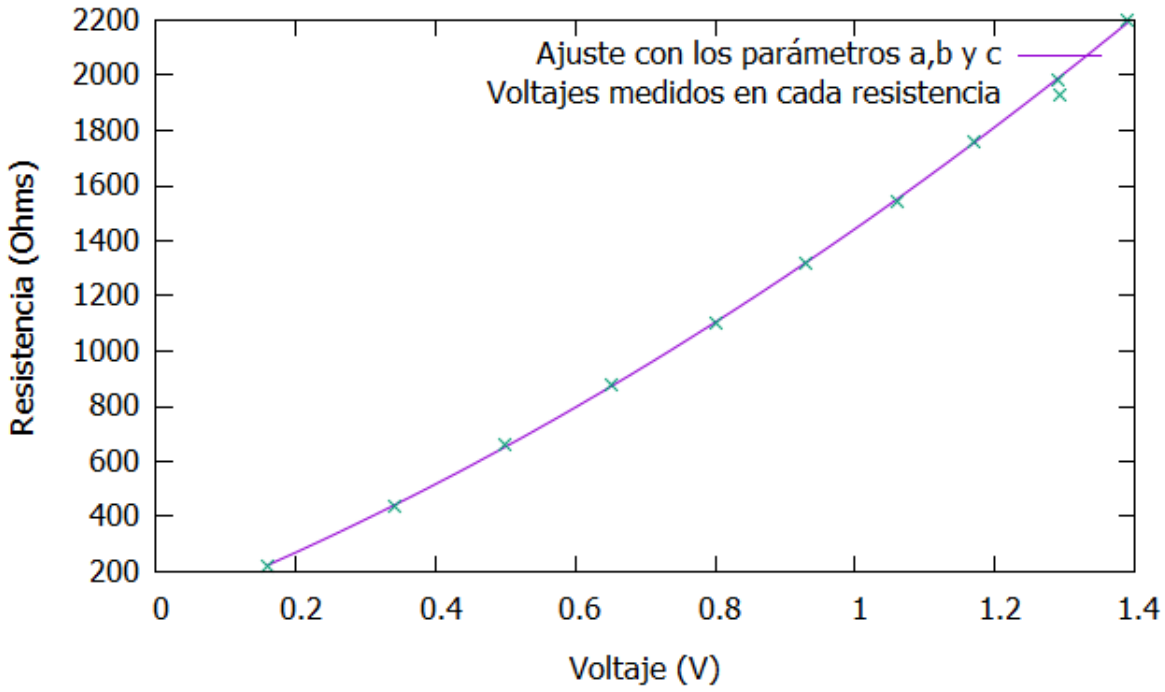
Así, se encontró que la diferencia mayor entre el valor medido y el valor de la resistencia fue de 5 ohms, que representa el 0.75 % del valor de la resistencia de 660 ohms.

Conclusiones

La importancia de contar con el modelo matemático que permitió hallar los valores (resistencia) de manera indirecta a partir de valores medibles (voltaje) es tal que de otra forma, el experimento no podría haber sido realizado de una forma tan sencilla.

Además, es imprtante recalcar que los rangos de medición de la resistencia no deben ser muy pequeños ni muy grandes en relación a la resistencia de referencia, pero ¿qué tan pequeño es muy pequeño y qué tan grande es muy grande? A este concepto se le conoce como intervalo de confianza, es decir, el interválo en el que estadísticamente se puede asegurar la exactitud del experimento. El cálculo de este intervalo se mostrará en otra práctica.

Finalmente, en la Gráfica 2 se muestra que el ajuste de un polinomio de segundo grado fue suficiente para hallar el modelo matemático que fuera capáz de medir valores de resistencias en el rango indicado.



Gráfica 2. Ajuste a los voltajes medidos

Bibliografía

[1] D. C. Baird. *Experimentación: Introducción a la teoría de mediciones y al diseño de experimentos.* Prentice-Hall Hispanoamericana, 2Ed.

[2] D.C. Montgomery. *Design and analisys of experiments.* John-Wiley. 9Ed.

[3] Arduino web site
http://www.arduino.cc

[4] D. Lindsay. *Guía de reacción científica.* Trillas.

[5] Barbara Gastell. *How to write and publish a scientific paper.* 8 Ed. Greenwood.

[6] D.Lindsay. *Scientific writing: thinking in words.* Ed. Csiro.

Apéndice. Códigos, otras mediciones, datos

Voltaje(V)	Resistencia(Ohms)
0.16	220
0.34	440
0.50	660
0.65	880
0.80	1100
0.93	1320
1.06	1540
1.17	1760
1.29	1980
1.39	2200

Tabla 1. Voltaje medido en cada resistencia.

```
1  ------------------------------------------------------------
2    Código 1. Medición de voltajes de las terminales
3  ------------------------------------------------------------
4  unsigned int NSample=126;
5  const double Ratio = (double)5.0/(1023.0*NSample);
6  unsigned int PinSensor0=A0;
7  unsigned int PinSensor1=A1;
8  unsigned int PinSensor2=A2;
9  unsigned int PinSensor3=A3;
10 unsigned int PinSensor4=A4;
11 unsigned int PinSensor5=A5;
12
13 unsigned long int SensorValue0;
14 unsigned long int SensorValue1;
15 unsigned long int SensorValue2;
16 unsigned long int SensorValue3;
17 unsigned long int SensorValue4;
18 unsigned long int SensorValue5;
19
20
21 void setup() {
22   delay(1000);
23   Serial.begin (9600);
24
25   pinMode(PinSensor0,INPUT);
26   pinMode(PinSensor1,INPUT);
27   pinMode(PinSensor2,INPUT);
28   pinMode(PinSensor3,INPUT);
29   pinMode(PinSensor4,INPUT);
30   pinMode(PinSensor5,INPUT);
31
32   delay(1000);
33 }
34
35 void loop() {
36   int i;
```

```
37    double V0, V1, V2, V3, V4, V5;
38    SensorValue0=0;
39    SensorValue1=1;
40    SensorValue2=2;
41    SensorValue3=3;
42    SensorValue4=4;
43    SensorValue5=5;
44
45
46    for (i=1;i<=NSample;i++)SensorValue0=SensorValue0+analogRead(
          PinSensor0);
47    for (i=1;i<=NSample;i++)SensorValue1=SensorValue1+analogRead(
          PinSensor1);
48    for (i=1;i<=NSample;i++)SensorValue2=SensorValue2+analogRead(
          PinSensor2);
49    for (i=1;i<=NSample;i++)SensorValue3=SensorValue3+analogRead(
          PinSensor3);
50    for (i=1;i<=NSample;i++)SensorValue4=SensorValue4+analogRead(
          PinSensor4);
51    for (i=1;i<=NSample;i++)SensorValue5=SensorValue5+analogRead(
          PinSensor5);
52
53    V0=(double)SensorValue0*Ratio;
54    V1=(double)SensorValue1*Ratio;
55    V2=(double)SensorValue2*Ratio;
56    V3=(double)SensorValue3*Ratio;
57    V4=(double)SensorValue4*Ratio;
58    V5=(double)SensorValue5*Ratio;
59
60
61    Serial.print(millis()/1000.0,3);
62    Serial.print(",");
63    Serial.print(V0,3);
64    Serial.print(",");
65    Serial.print(V1,3);
66    Serial.print(",");
67    Serial.print(V2,3);
68    Serial.print(",");
69    Serial.print(V3,3);
70    Serial.print(",");
71    Serial.print(V4,3);
72    Serial.print(",");
73    Serial.println(V5,3);
74
75
76  }
```

```
-----------------------------------------------------------------------

Código 2. Ajuste de la función f(x) a los datos medidos en la Tabla
    1
-----------------------------------------------------------------------

gnuplot> f(x)=a*x**2+b*x+c
gnuplot> fit f(x) '5.6kohms-220ohms.dat' via a,b,c
```

```
-----------------------------------------------------------------------
Código 3. Los parámetros a, b y c del polinomio de ajuste fueron
   introducidos al código de manera que el medidor estaba calibrado
-----------------------------------------------------------------------
 float a = 378.502;
  float b = 1012.01;
  float c = 52.818;
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);

}

// the loop routine runs over and over again forever:
void loop() {
  long resistencia;

  // read the input on analog pin 0:
  int sensorValue = analogRead(A4);
  // Convert the analog reading (which goes from 0 - 1023) to a
     voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:

  resistencia = a*voltage*voltage+b*voltage+c;
  Serial.println(resistencia);
  delay(2000);
}
```