

Actividad 1 - Introducción Entradas Digitales y Analógicas – ARDUINO

Por:

Juan Felipe Higuera Perez

1032011172

Miguel Angel Uribe Zuluaga

1022142422

David Alejandro Suarez Varon

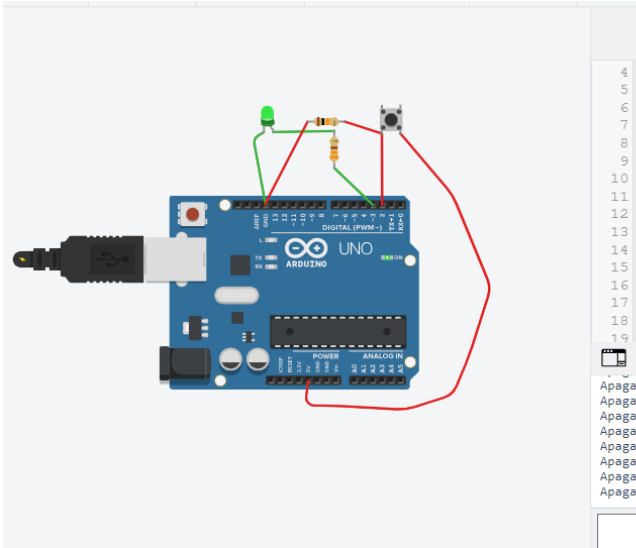
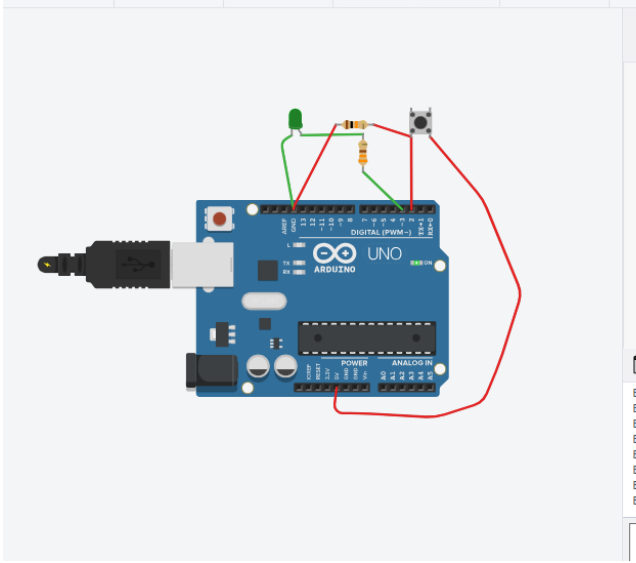
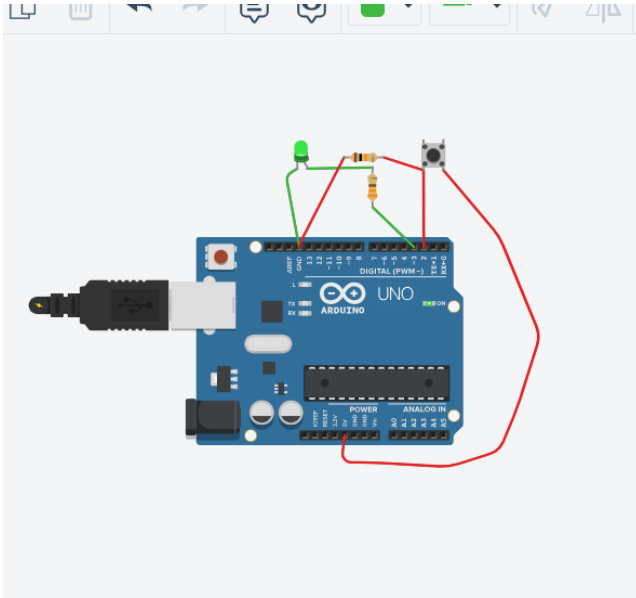
1091969718

UDEA 2023

Retos de entradas digitales

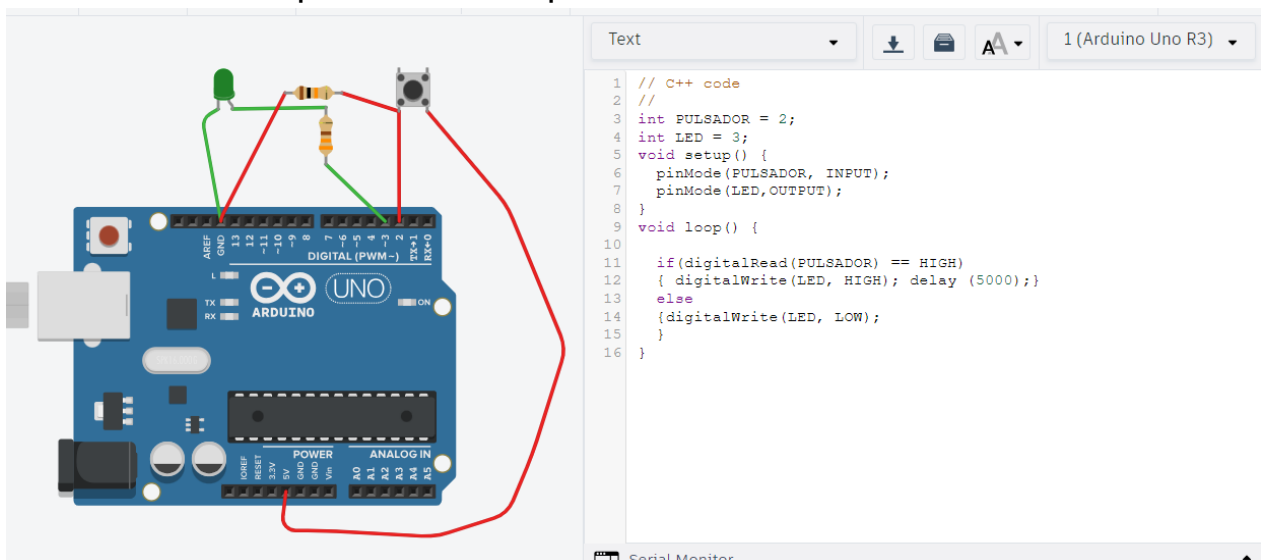
- 1) Este reto nos hablaba de que investigáramos sobre las resistencias pull-up y pull-down, para así aplicarlas en el código que habíamos hecho de ejemplo anteriormente, entonces mi compañero y yo nos pusimos a consultar sobre estas resistencias dándonos a entender que pull-down es usado normalmente ya que esta indica que cuando el circuito está en reposo o apagado, pasará 0V mientras que cuando está activo al pulsar el pulsador dejará pasar la corriente de 5V; siendo que 0V significa un estado LOW y 5V un estado HIGH. Ahora la pull-up es todo lo contrario ya que si tenemos el circuito en reposo la corriente pasará como 5V en un estado HIGH aunque si decidimos pulsar el pulsador dejará de pasar corriente pasando a un estado LOW de 0V.

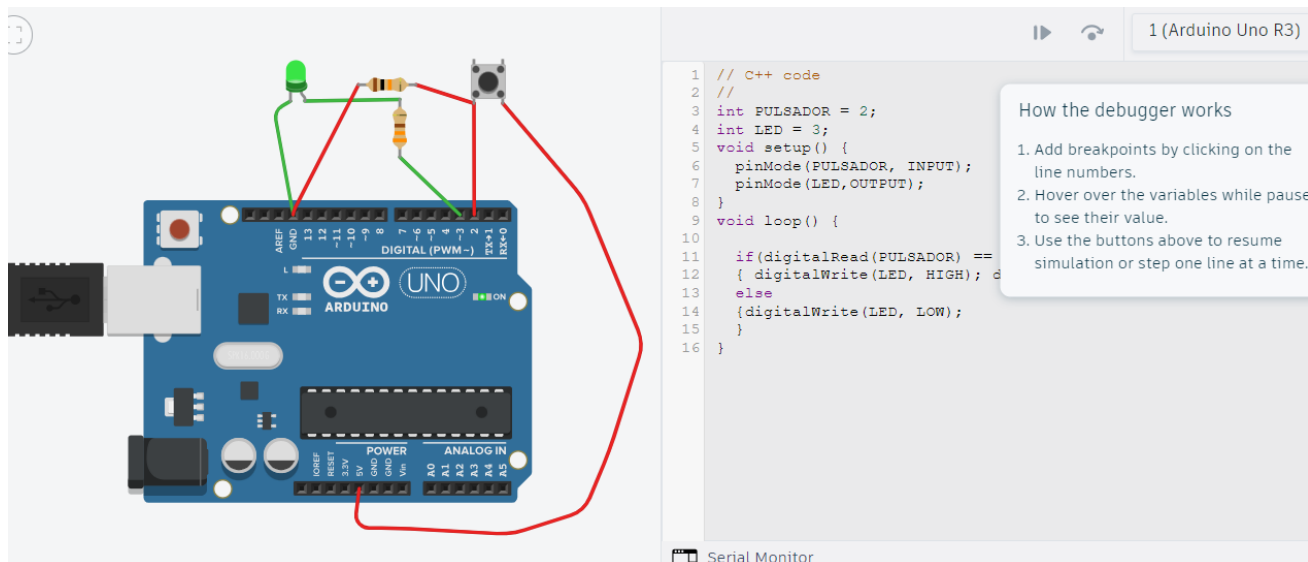
Teniendo eso en cuenta, modificamos el ejemplo que teníamos como nos decía el ejercicio para que pudiéramos aplicar la resistencia pull-up, así que cada vez que una vez arranquemos la simulación el sistema estará pasando corriente y como consecuencia encenderá el LED que tenemos conectado al arduino, también, para que sea más notorio esto hicimos que en el Serial monitor se imprima cuando esta apagado y encendido el pulsador. Ahora a continuación dejamos las capturas de pantalla del arduino y el resultado



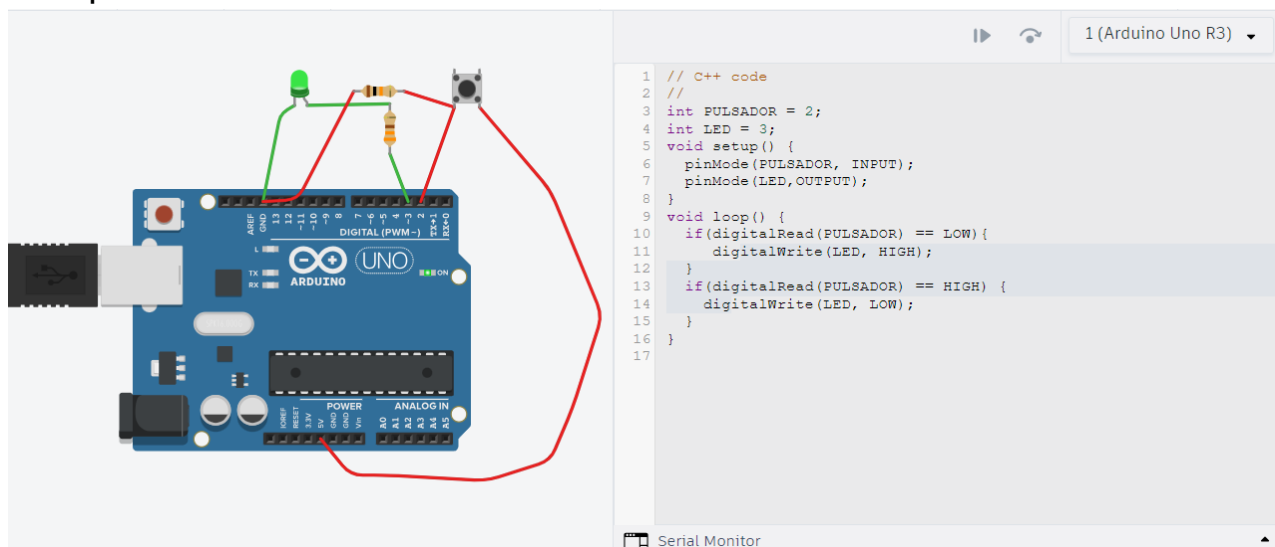
Se puede ver claramente que cuando el Serial monitor imprime “Apagado”, es porque el pulsador está en un estado LOW y eso causara que el LED esté encendido (HIGH), aunque si decidimos presionar el pulsador para HIGH hará que el LED se apague mientras esté presionado (LOW), como se puede ver en el Serial monitor imprime “Encendido”.

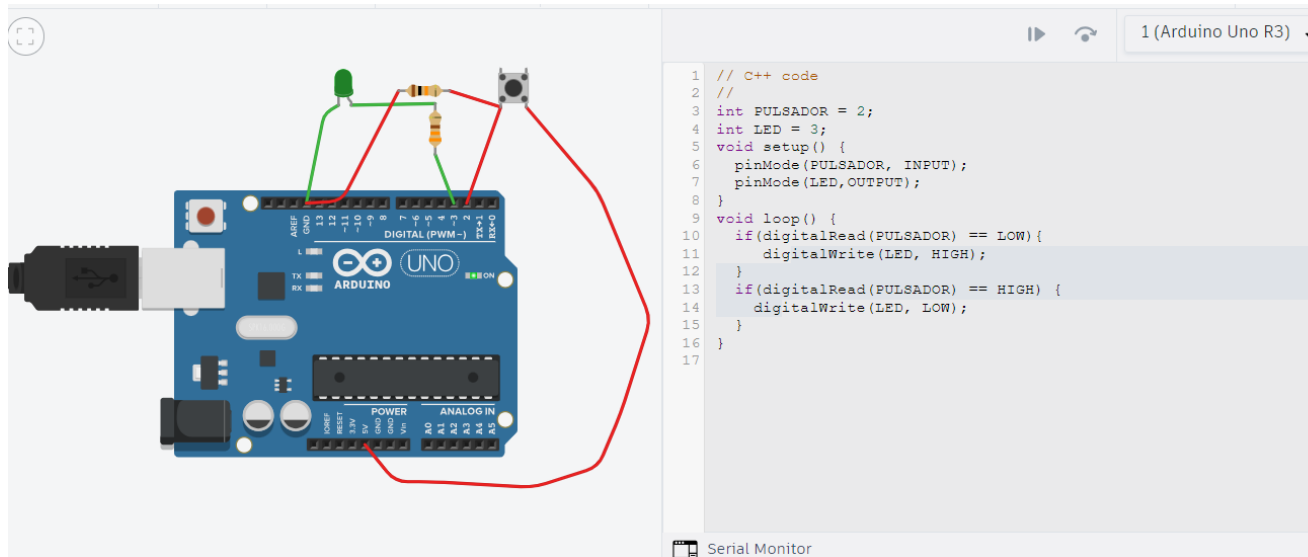
- 2) Este reto nos indicaba que modificáramos el código del ejemplo para que el LED se encendiera por 5 segundos cada vez que presionamos el pulsador, después de investigar un poco más de código arduino pudimos entender como hacer esto y es muy simple, la única modificación que hay que hacer es que después de la condición de que si el pulsador estaba en HIGH hacía que el LED pasara a HIGH también, debíamos colocar un comando llamado “delay()” que significa que nos pausara el código en el tiempo que decidamos, pero el tiempo debe ser colocado en milisegundos así que debemos que como es 5 segundos es una conversión muy fácil al multiplicar 5×1000 . Si quisiéramos hacerlo en minutos habría que multiplicar el número de minutos por 60 y eso multiplicarlo por 1000, una vez aclarado esto pudimos hacer el código sin ningún problema y que funcionará correctamente. A continuación compartimos las capturas del resultado.





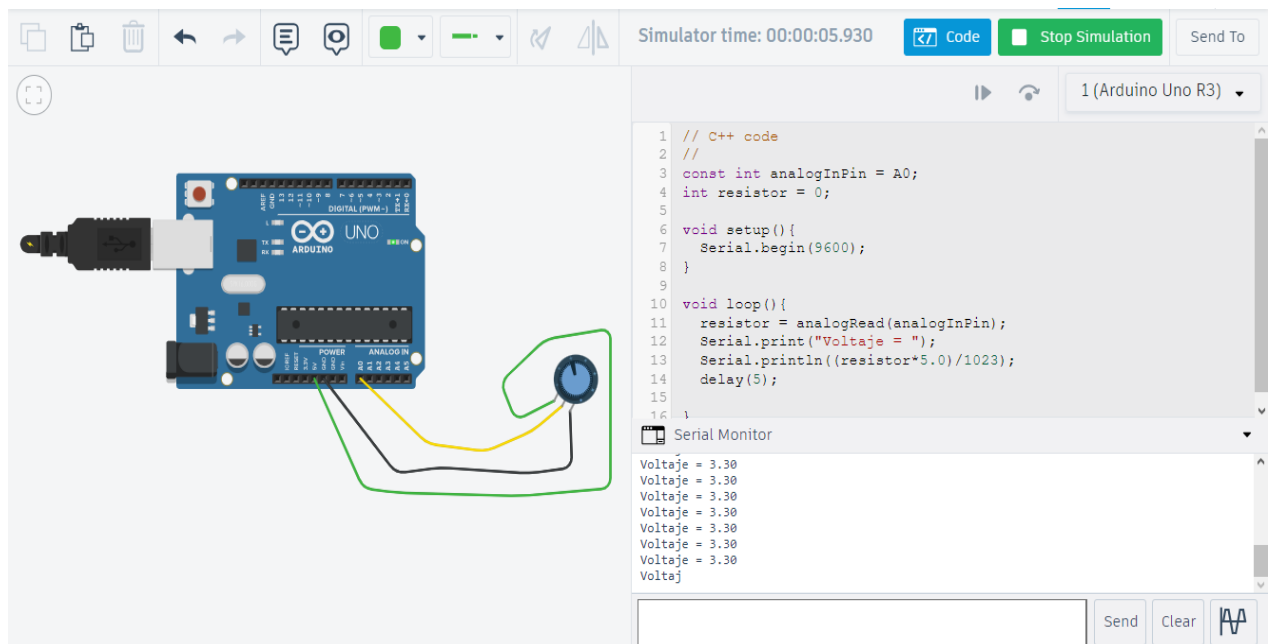
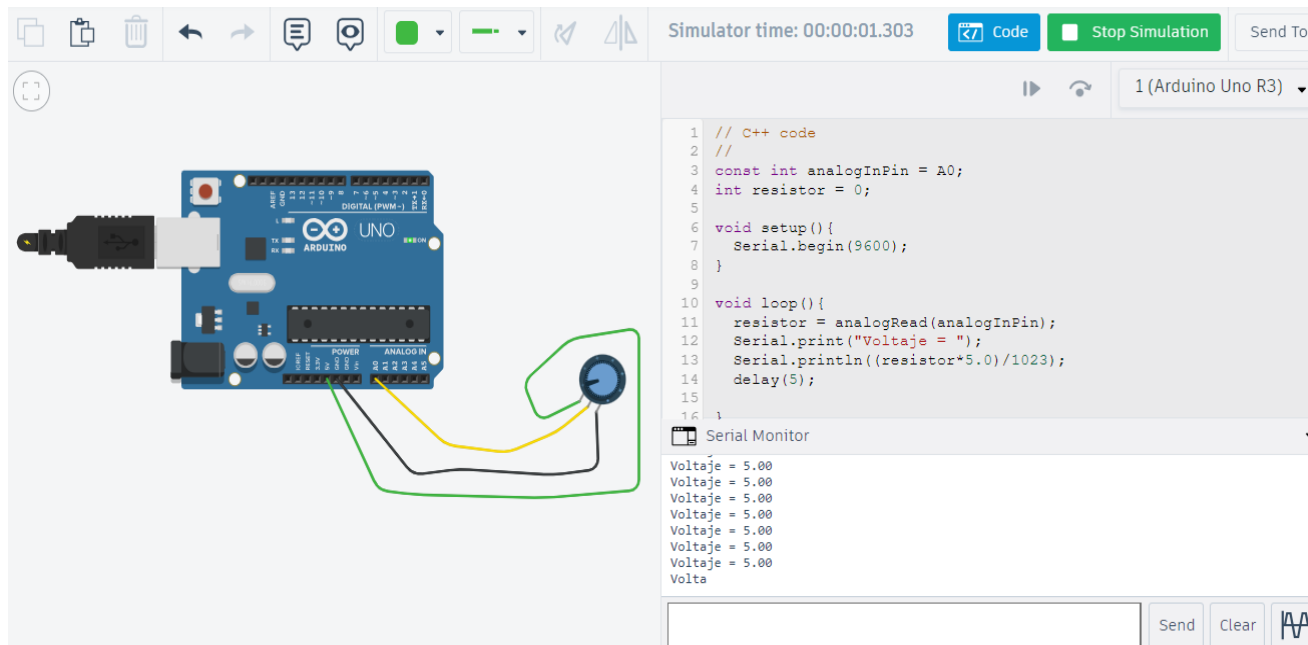
- 3) Este reto nos indicaba que invirtieramos el código del ejemplo, para que en vez de ser Apagado —> Encendido, Encendido—>Apagado, debíamos hacerlo de Encendido—>Apagado, Apagado—> Encendido. Así que lo único que había que modificar es que cuando el pulsador esté en HIGH, el LED muestre un estado LOW significando que si este no es el caso, el LED estará en estado HIGH, así que no hay mucho que discutir en este punto.





Retos de entradas analógicas

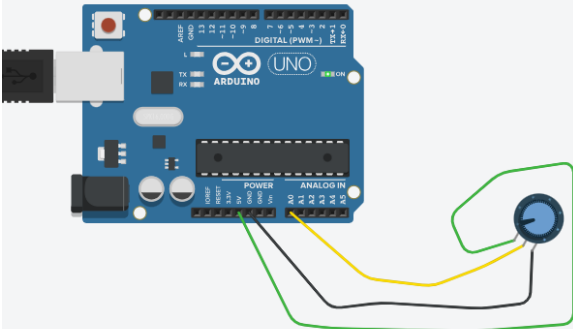
- 1) A) Ok, consideramos mientras realizabamos este reto que era el más complicado de entender al principio, de pronto ahora que lo terminamos de hacer no se ve tan difícil, pero igualmente eso no quita que nos costó hacerlo funcionar correctamente. En este reto primero nos decían que modificáramos el ejemplo que nos da el taller para que nos den el valor del Voltaje en el Serial monitor, así que hicimos un loop en el cual todo el rato mientras esté activa la simulación y encendido el circuito nos va a dar este valor gracias a la operación que tenemos escrita, ya que tenemos que encontrar el voltaje nos toca despejar diciendo que el Voltaje va a ser igual a la resistencia por el voltaje máximo que es 5 y dividido con 1023, cómo se va a mostrar con la siguiente imagen.



Como se ve, dependiendo de donde gire el potenciómetro, me va dar un valor de Voltaje diferente cada vez yendo desde 0 siendo el mínimo hasta 5 que es el máximo que puede llegar.

- 1) B) Ahora este reto fue más fácil de realizar, ya que solo nos decía que en vez de dar el valor del voltaje, nos diera en el Serial Monitor el valor de la resistencia en R2 pero en kΩ que nos muestra en la figura 6 de la guía, lo que hay que agregar al anterior programa es realizar la ley ohm para encontrar a R2

haciendo que el voltaje mínimo que es 5V se reste con el voltaje previamente calculado y sea dividido entre este voltaje menos R1 que se asume que es 10, así en el Serial Monitor nos imprimirá R2 dependiendo de donde giramos el potenciómetro. A continuación compartimos las capturas de los resultados.

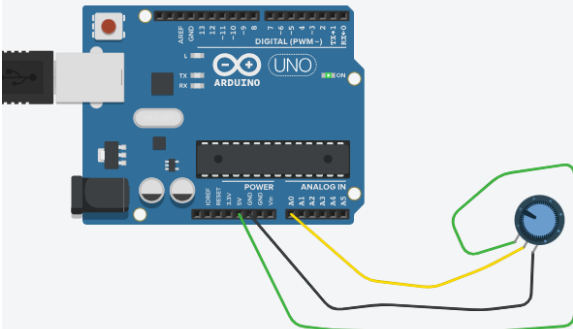


Simulator time: 00:00:58

```
1 // C++ code
2 //
3 const int analogInPin = A0;
4
5 void setup() {
6   Serial.begin(9600);
7 }
8
9 void loop() {
10  int sensorValue = analogRead(analogInPin);
11  float voltage = (sensorValue * 5.0) / 1023.0;
12  float resistanceR2 = (5.0 - voltage) / (voltage / 10.0);
13
14  Serial.print("Resistencia de R2 = ");
15  Serial.print(resistanceR2);
16  Serial.println(" kΩ");
17 }
```

Serial Monitor

Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ



Simulator time: 00:01:22

```
1 // C++ code
2 //
3 const int analogInPin = A0;
4
5 void setup() {
6   Serial.begin(9600);
7 }
8
9 void loop() {
10  int sensorValue = analogRead(analogInPin);
11  float voltage = (sensorValue * 5.0) / 1023.0;
12  float resistanceR2 = (5.0 - voltage) / (voltage / 10.0);
13
14  Serial.print("Resistencia de R2 = ");
15  Serial.print(resistanceR2);
16  Serial.println(" kΩ");
17 }
```

Serial Monitor

Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 0.00 kΩ
Resistencia de R2 = 1.37 kΩ
Resistencia de R2 = 1.91 kΩ
Resistencia de R2 = 1.91 kΩ

Simulator time: 00:01:39

Code Stop Simulation Send To

1 (Arduino Uno R3)

```
1 // C++ code
2 //
3 const int analogInPin = A0;
4
5 void setup() {
6   Serial.begin(9600);
7 }
8
9 void loop() {
10  int sensorValue = analogRead(analogInPin);
11  float voltage = (sensorValue * 5.0) / 1023.0;
12  float resistanceR2 = (5.0 - voltage) / (voltage / 10.0);
13
14  Serial.print("Resistencia de R2 = ");
15  Serial.print(resistanceR2);
16  Serial.println(" kΩ");
17 }
```

Serial Monitor

Resistencia de R2 = 1.91 kΩ
Resistencia de R2 = 1.91 kΩ
Resistencia de R2 = 1.91 kΩ
Resistencia de R2 = 16.30 kΩ
Resistencia de R2 = 23.32 kΩ
Resistencia de R2 = 23.32 kΩ
Resistencia de R2 = 23.32 kΩ
Resistencia de R2 = 23.32 kΩ

Send Clear

