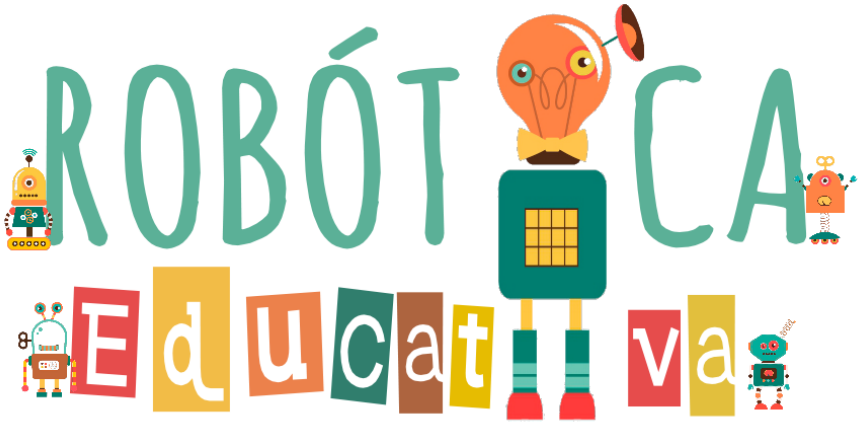
Robótica Educativa



3.5 SENSORES PARTE 1

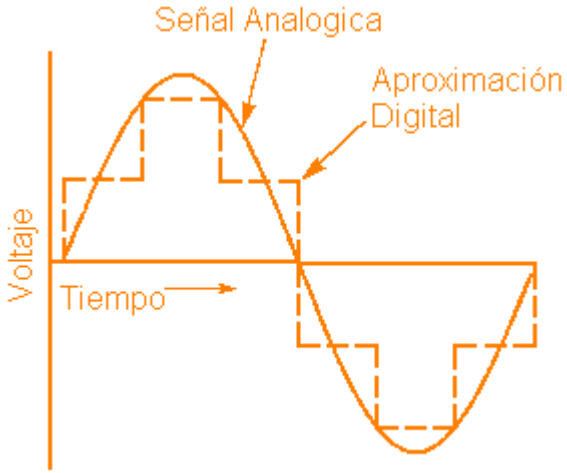
Contenido

Sensores

Entradas Analógicas

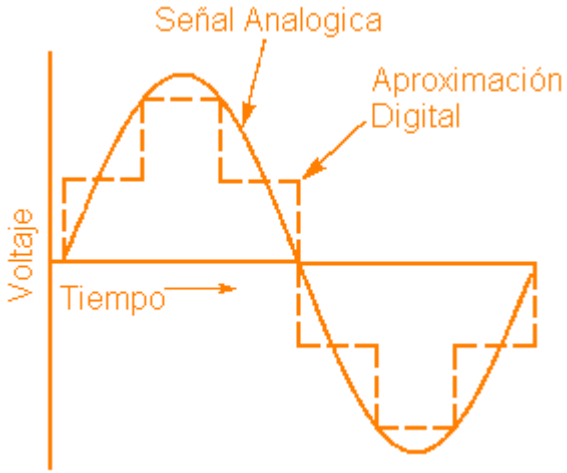
Todas las señales físicas que captamos con nuestros sentidos, al igual que toda variable que existe en el entorno (temperatura, velocidad, peso, caudal, etc) son señales analógicas, es decir señales que varían con el tiempo, y que pueden tomar diferentes valores a diferencia de la señal digital que solo presenta dos valores.

En la siguiente figura podemos ver una señal analógica, que representa una función sinusoidal junto con una aproximación discreta.



Entradas Analógicas

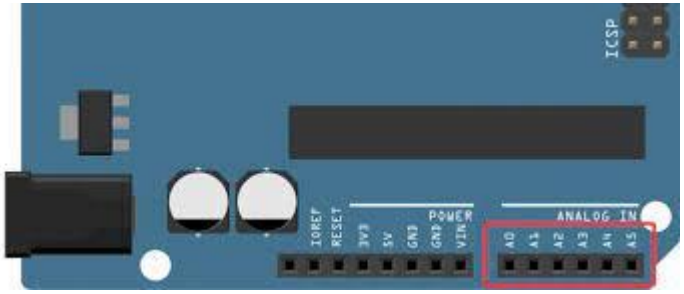
El Arduino tiene la capacidad de **leer señales analógicas** realizando **aproximaciones discretas** a través de pequeños rectángulos digitales como fue observado en la figura, donde podemos ver fácilmente, que entre más pequeños sean los rectángulos, más parecido será la aproximación digital con relación a la señal analógica.



Entradas Analógicas

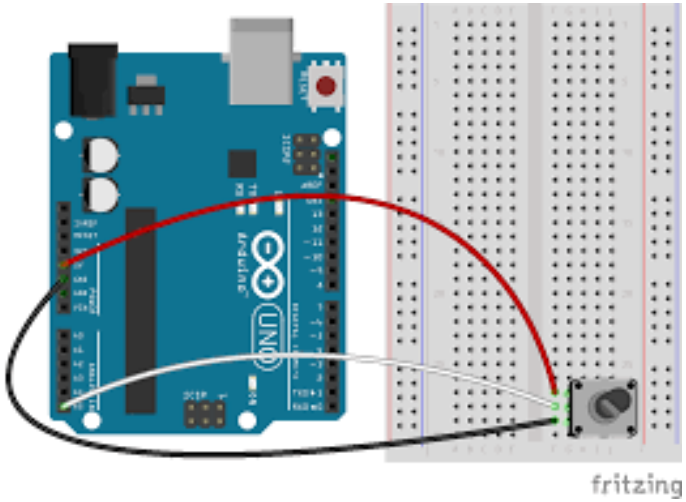
Que significa adc en arduino?

ADC en Arduino es la abreviación del Conversor Analógico Digital Arduino o con sus siglas en Ingles *Analog Digital Conver*ter



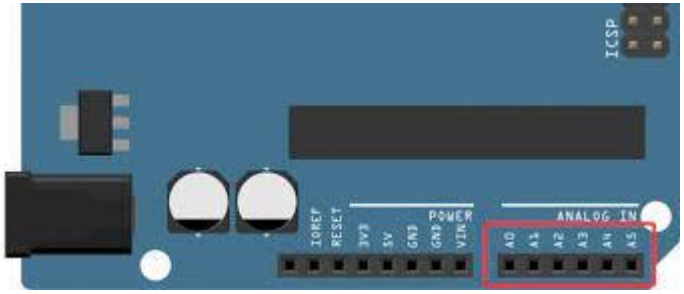
Que es adc en arduino?

El conversor **ADC Arduino** es el encargado de la conversión analógica digital empleada en la placa para poder leer todos los sensores disponibles en el mercado para nuestro proyecto de automatización con Arduino.

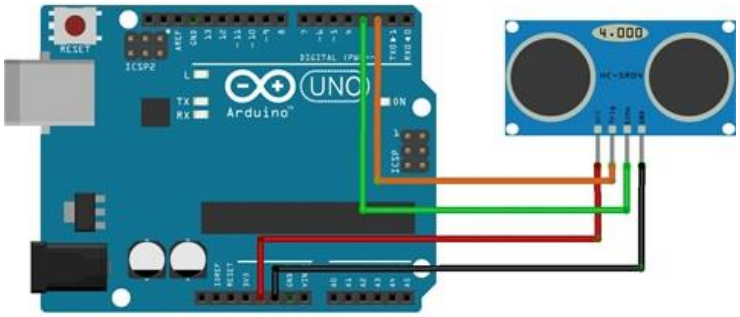


ADC Arduino

Cuando se trabaja con el ADC Arduino (conversor análogo digital) se debe tener en consideración que solo podemos colocar voltajes de **máximo 5v o de 3.3v** si trabajamos con un Arduino con este voltaje como el Mini. De lo contrario podremos quemar nuestra placa.



A continuación se muestra una tabla con los voltajes máximos de operación y la máxima resolución de cada placa Arduino. Aquí podrás ver el convertidor análogo digital – ADC de Arduino Mega 2560, UNO, NANO, etc.



ADC Arduino

Placa	Voltaje Operación	PINES	MAX RESOLUTION
Uno	5 Volts	A0 to A5	10 bits
Mini, Nano	5 Volts	A0 to A7	10 bits
Mega, Mega2560, MegaADK	5 Volts	A0 to A14	10 bits
Micro	5 Volts	A0 to A11*	10 bits
Leonardo	5 Volts	A0 to A11*	10 bits
Zero	3.3 Volts	A0 to A5	12 bits**
Due	3.3 Volts	A0 to A11	12 bits**
MKR Family boards	3.3 Volts	A0 to A6	12 bits**

ADC Arduino

Como podemos ver en la tabla anterior, la lectura que haga el Arduino en la entrada analoga va a depender de la resolución ADC del Arduino que tenga. Por ejemplo, si estamos leyendo voltajes en un Arduino UNO, como es de 10 bits en realidad el Arduino va a ver una variación de un **entero** entre **0 a 1023**. ¿Pero por qué?

ADC Arduino

Un bit es un binario que puede tomar **dos** valores (**0** o **1**), o sea que si tiene una resolución de 10 bits indica que (2^10 = 1024), pero como empezamos desde 0, se dice que el valor entero toma valores entre **0 a 1023.** Esto quiere decir que si Arduino mide:

- el máximo voltaje (5v) va a almacenar un valor entero de **1023**.
- Voltaje intermedio (2.5v) va a almacenar un valor entero de **512**.
- Voltaje mínimo (0v) va a almacenar un entero de **0**.

Sensores

analogRead() Arduino

Entendamos rápidamente que hace la función **analogRead()** en Arduino para comenzar a usar el convertidor Analógico Digital de la placa.

Trabajar con las entradas Análogas de Arduino es bastante sencillo, para eso usamos la siguiente sintaxis:

```
int Value = analogRead (pin);
```

Sensores

analogRead() Arduino

```
int Value = analogRead (pin);
```

parámetros

(A0 a A5 en la mayoría de las placas, A0 a A6 en placas MKR, A0 a A7 en el Mini y Nano, A0 a A15 en el Mega).

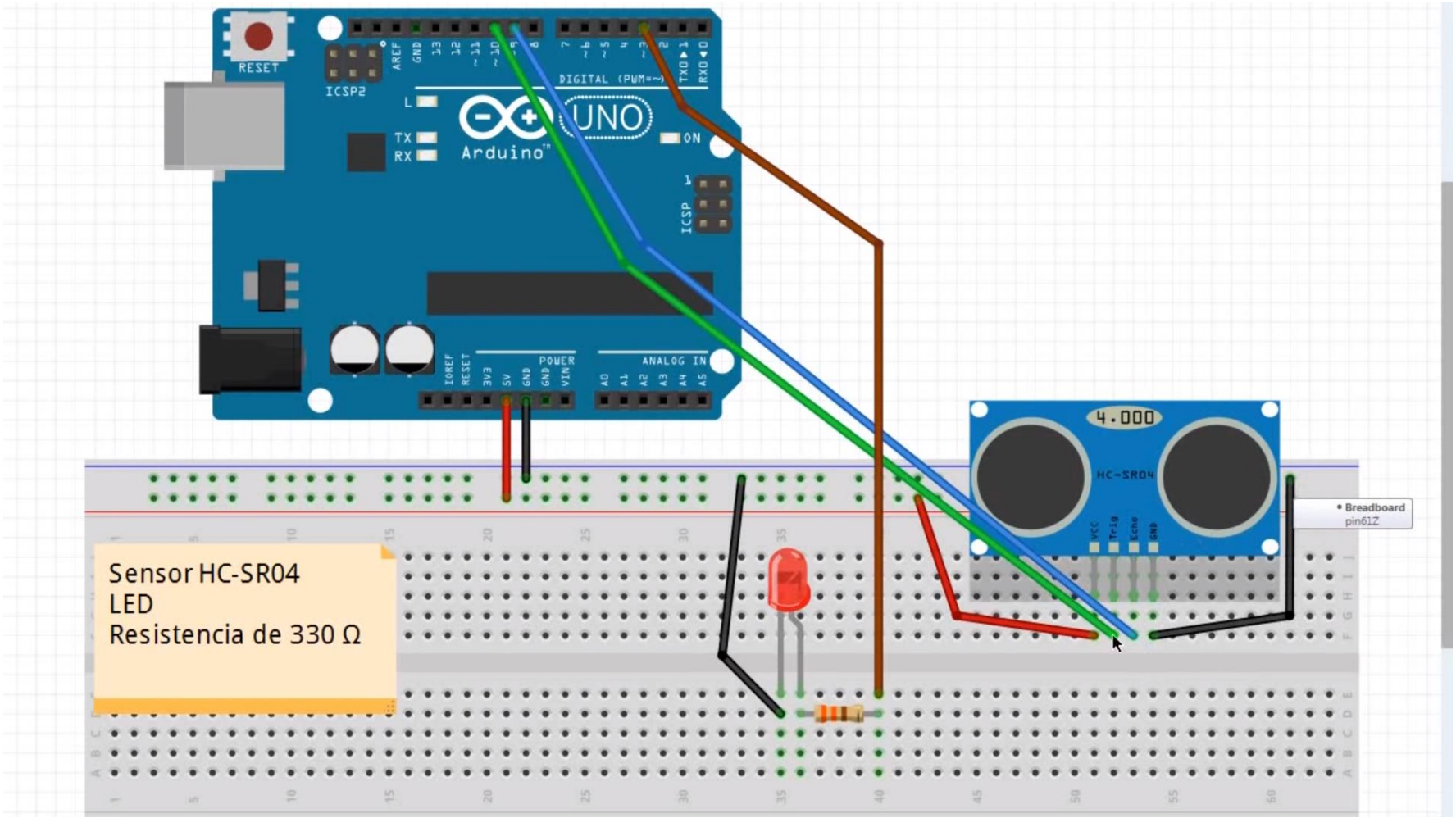
Retornos

La **lectura analógica es retornada como una variable entera**, que está limita a la resolución del convertidor analógico digital Arduino (0-1023 a 10 bits o 0-4095 a 12 bits).

SENSORES - ULTRASONICO

Frecuencia: 40 Khz
Rango: 2 a 400 cms

SENSORES - ULTRASONICO



SENSORES – ULTRASONICO – PROGRAMA 1

Lectura en cm por Puerto serial

```
int TRIG = 10;       // trigger en pin 10
int ECO = 9;        // echo en pin 9
int LED = 3;        // LED en pin 3
int DURACION;
int DISTANCIA;

void setup()
{
  pinMode(TRIG, OUTPUT);  // trigger como salida
  pinMode(ECO, INPUT);    // echo como entrada
  pinMode(LED, OUTPUT);   // LED como salida
  Serial.begin(9600);     // inicializacion de comunicacion serial a 9600 bps

}
```

SENSORES – ULTRASONICO – PROGRAMA 1

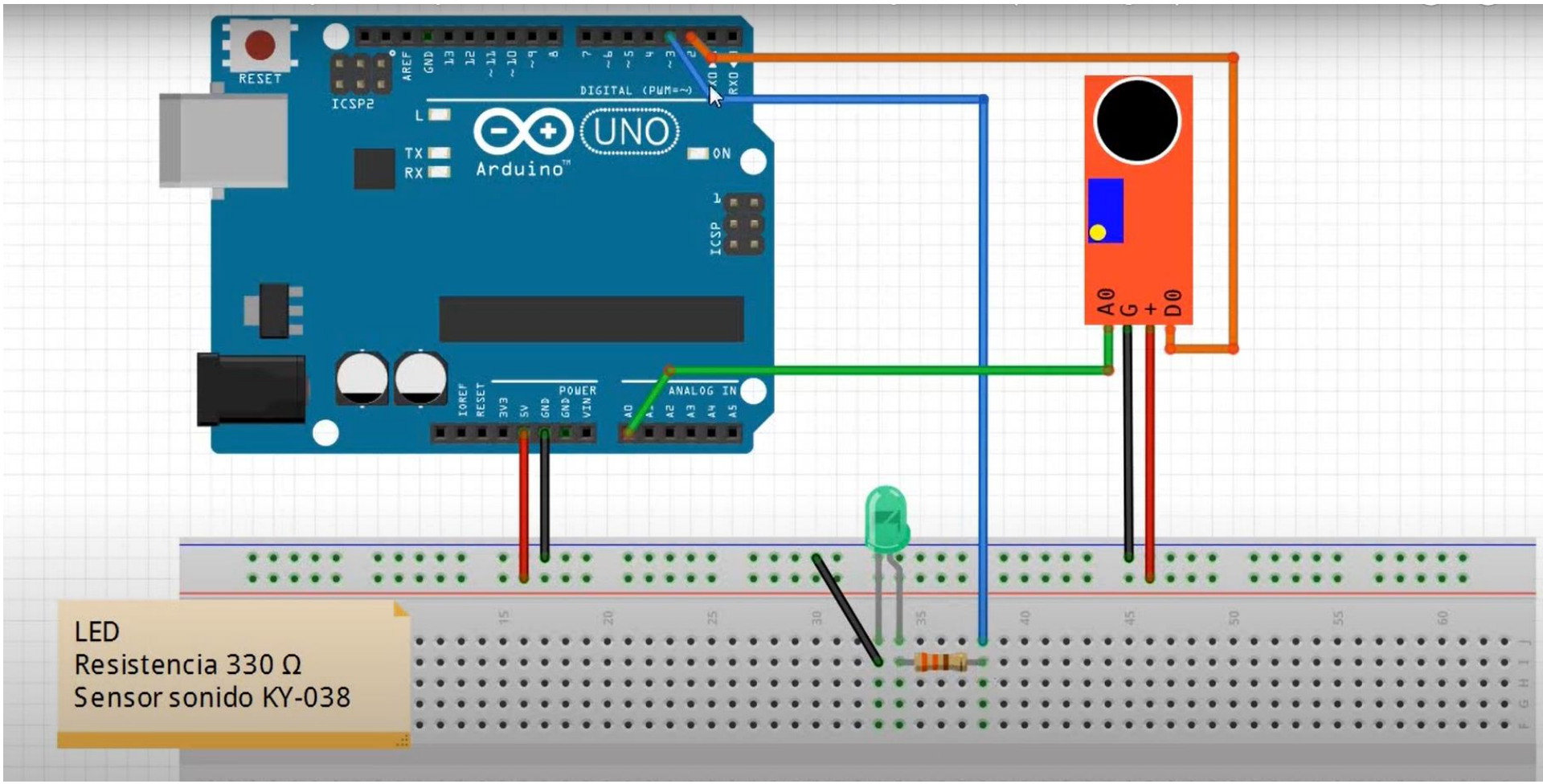
Lectura en cm por Puerto serial

```
void loop()
{

  digitalWrite(TRIG, HIGH);     // generacion del pulso a enviar
  delay(1);        // al pin conectado al trigger
  digitalWrite(TRIG, LOW);    // del sensor

  DURACION = pulseIn(ECO, HIGH);  // con funcion pulseIn se espera un pulso
           // alto en Echo
  DISTANCIA = DURACION / 58.2;    // distancia medida en centimetros
  Serial.println(DISTANCIA);    // envio de valor de distancia por monitor serial
  delay(200);        // demora entre datos
}
```

SENSORES – DE SONIDO



Señales analógicas– Arduino

Conversión de escala

La conversión de una variable de entrada análoga a un valor de medida real, se lo hace mediante una conversión de escala.

Toda entrada análoga tiene un valor de lectura mínimo en bits y un lectura máxima en bits.

Lectura mínima : 0 bits

Lectura máxima : 1023 bits

Señales analógicas – Arduino

Conversión de escala - **Función map**

La función map() de Arduino permite transformar un valor entero de un rango de entrada al valor correspondiente a otro rango de salida.

Los 5 parámetros de entrada son valores enteros:

valor de entrada

inicio rango de entrada

final rango de entrada

inicio rango de salida

final rango de salida

Y la función devuelve el valor entero de salida una vez realizado el «mapeo».

Señales analógicas – Arduino

Conversión de escala - **Función map**

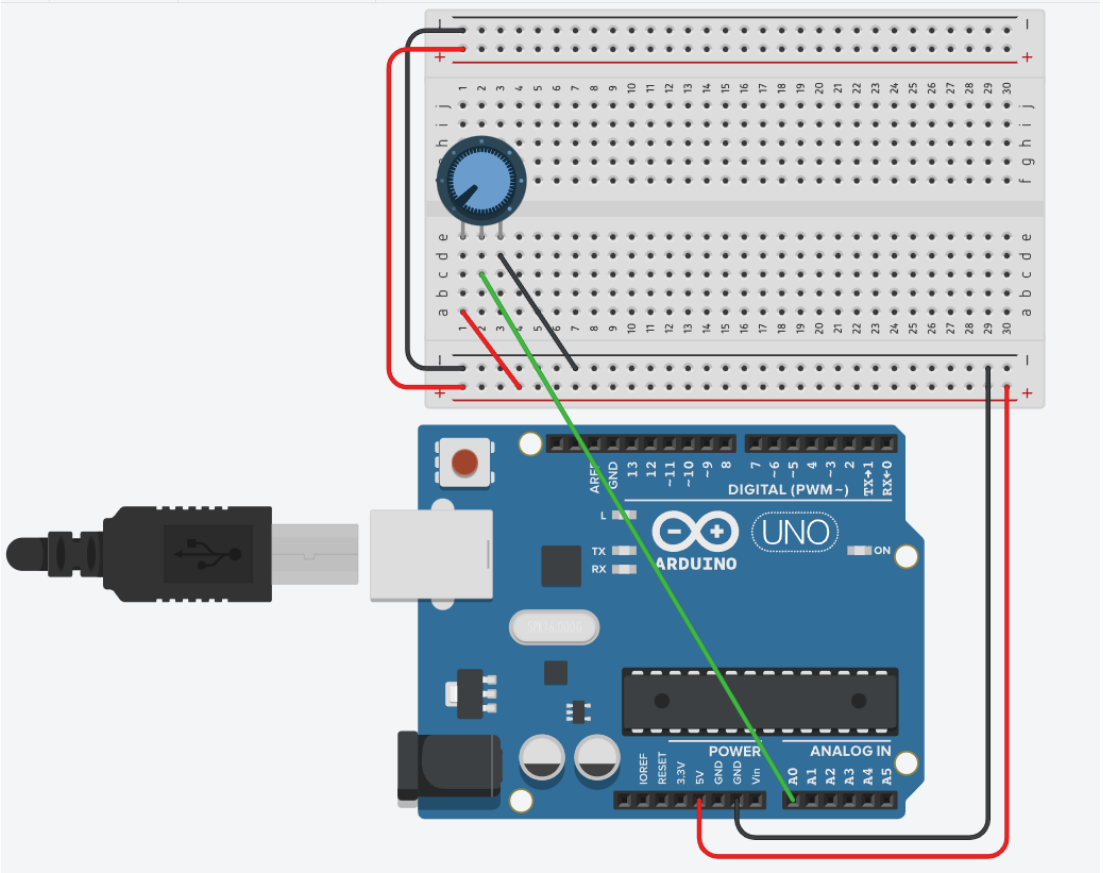
```
valor_led = map(valor_ldr, 0,320, 0,255);
```

El «**valor-ldr**» corresponde al valor leído por un sensor de luz, cuyo rango puede variar de **0** a **320**, valores que ponemos en el segundo y tercer lugar; los valores de salida deben oscilar entre **0** y **255** (valores cuarto y quinto) porque servirán para iluminar un led, este valor se almacena en la variable «**valor_led**» como valor de salida de la función map().

Potenciómetro como señal analógica– Arduino

Conversión de escala - **Función map**

Ejemplo control de entrada analógica



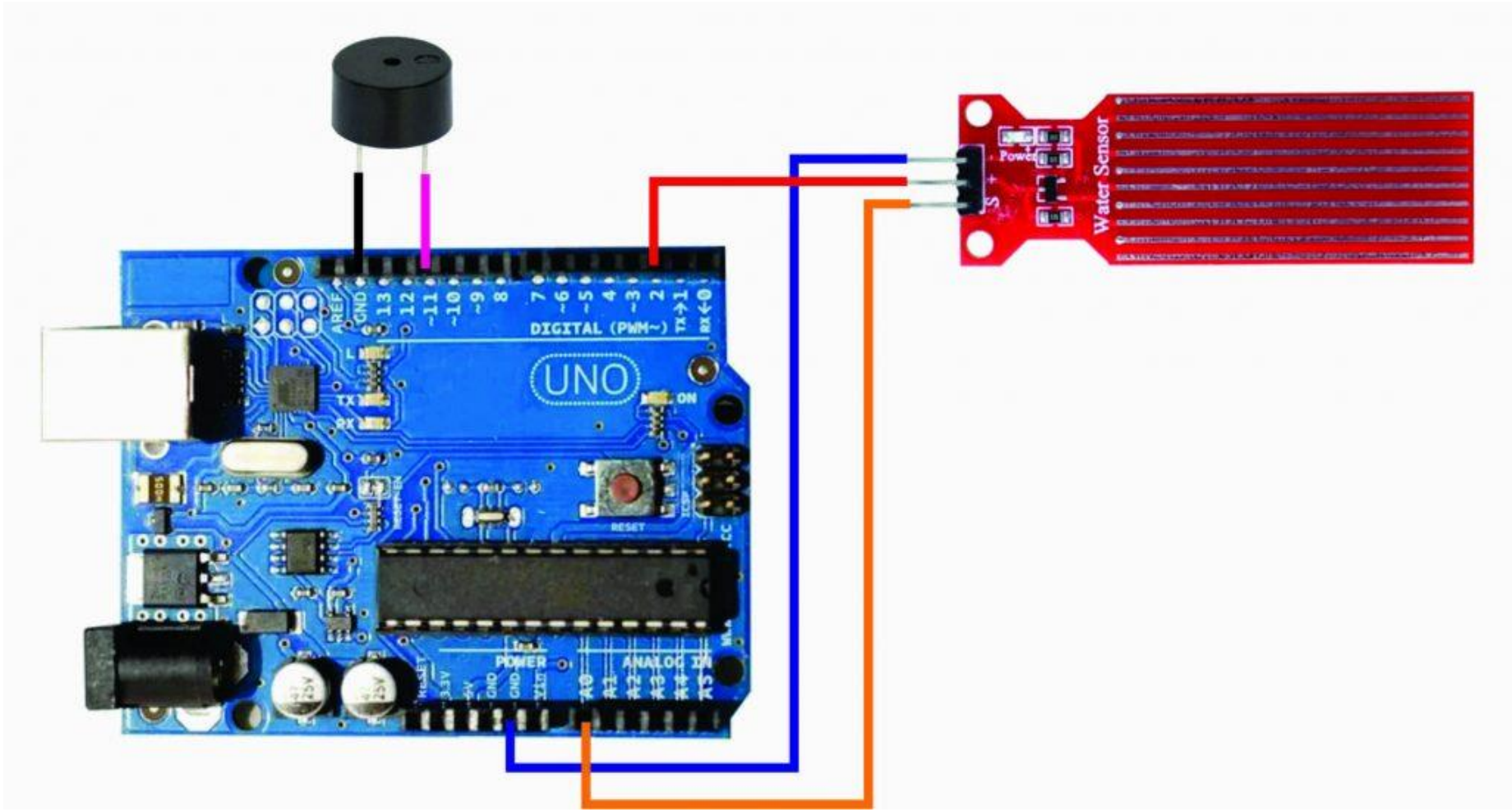
Potenciómetro como señal analógica– Arduino

Conversión de escala - Función map

Ejemplo control de entrada analógica

```
int Potenciometro = A0;
int Lectura;
int Porcentaje;
void setup() {
 Serial.begin(9600);
 pinMode(Potenciometro,INPUT);
}
void loop() {
  Lectura = analogRead (Potenciometro);
  Porcentaje= map(Lectura,0,1023,0,100);
  Serial.print("Lectura:");
  Serial.print(Lectura);
  Serial.print("   Porcentaje:");
  Serial.print(Porcentaje);
  Serial.print("%");
  Serial.println();
}
```

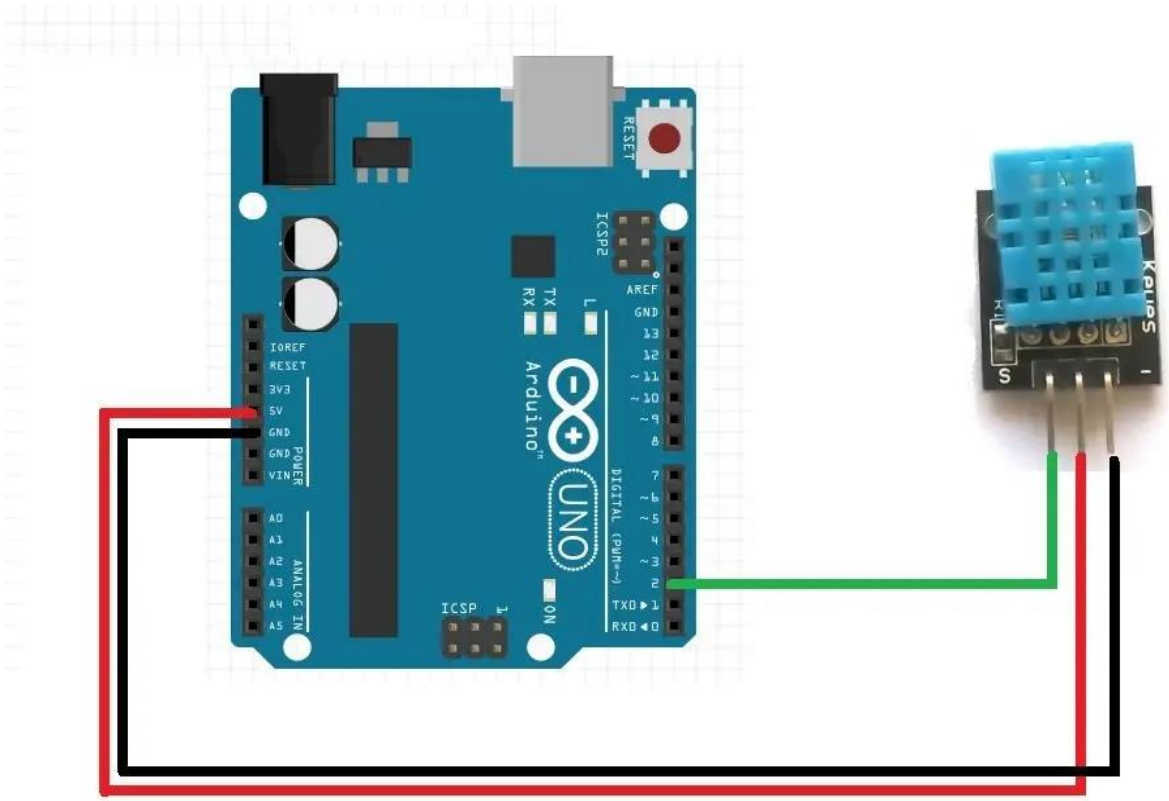
Sensor de lluvia – Arduino



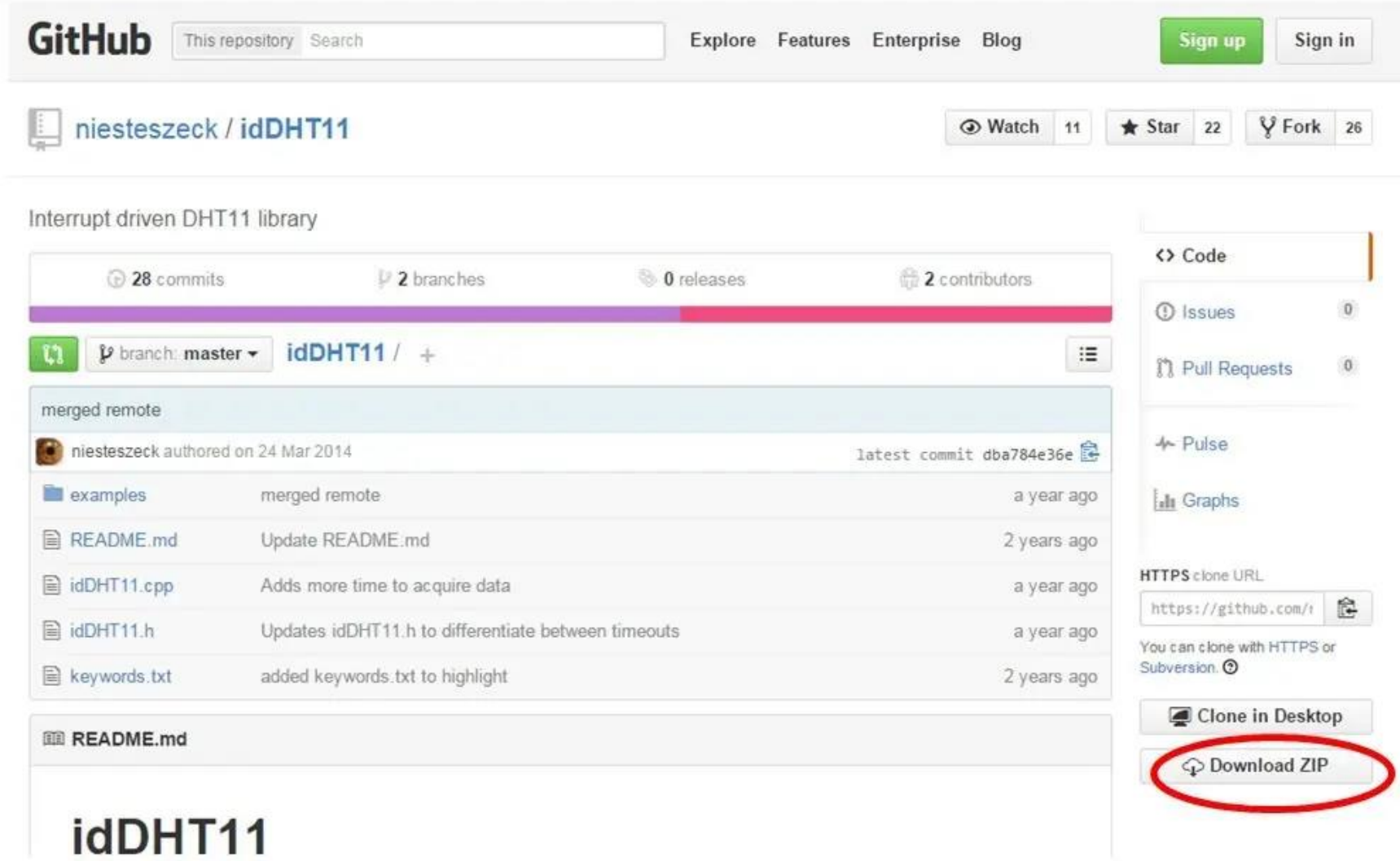
Sensor de lluvia – Arduino

```
#define PIN_ENCENDIDO  2
#define PIN_SENSOR A0
#define BUZZER 11
int valor_sensor = 0; // variable para almacenar el valor del sensor
void setup() {
  Serial.begin(9600);
  pinMode(PIN_ENCENDIDO, OUTPUT);   // configurar el pin D2 como SALIDA
  pinMode(BUZZER, OUTPUT);         //configurar el pin D11 como SALIDA
  digitalWrite(PIN_ENCENDIDO, LOW); // Apaga el sensor
}
void loop() {
  digitalWrite(PIN_ENCENDIDO, HIGH);  // Enciende el sensor
  delay(10);                    // espera 10 milisegundos
  valor_sensor = analogRead(PIN_SENSOR); // Leer el valor analógico del sensor
  digitalWrite(PIN_ENCENDIDO, LOW);   // Apaga el sensor
//Imprime el valor obtenido del sensor
  Serial.print("Sensor VALOR: ");
  Serial.println(valor_sensor);
  //Si el valor en mayor a 100 comienza sonar el buzzer
if(valor_sensor > 100){
digitalWrite(BUZZER, HIGH);
delay(100);
digitalWrite(BUZZER, LOW);
}
  delay(1000);
}
```

Sensor de temperatura y humedad dth11– Arduino



Sensor de temperatura y humedad dth11–



https://hetpro-store.com/TUTORIALES/wp-content/uploads/2016/08/DHT11.zip

Sensor de temperatura y humedad dth11– Arduino

```
#include <DHT11.h> //cargamos la librería DHT
int pin=2; //Seleccionamos el pin en el que se //conectará el sensor
DHT11 dht11(pin); //Se selecciona el DHT11 (hay //otros DHT)

void setup() {
Serial.begin(9600); //Se inicia la comunicación serial
}
void loop() {
float temp, hum;
int err;
if((err=dht11.read(hum, temp))==0)
 {
   Serial.print("Temperatura: ");
   Serial.print(temp);
   Serial.print(" Humedad: ");
   Serial.print(hum);
   Serial.println();
 }
     else
       {
         Serial.println();
         Serial.print("Error Num :");
         Serial.print(err);
         Serial.println();
       }
       delay(1000);
}
```