ROBÓTICA Y PROGRAMACIÓN

Quiero fabricar un robot

Para el cerebro del robot puedes optar por 2 tecnologías:

1. Basadas en **microcontroladores**: placas con circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Podemos programar los puertos de E/S y las interrupciones directamente, sin necesidad de un sistema operativo. A diferencia de los PC's la unidad central, la memoria y periféricos de entrada/salida están en un único circuito integrado. Destacan la placa Arduino y NodeMCU
2. Basadas en **microprocesadores** tipo PC: su funcionamiento, aunque más desarrollado, necesita de un sistema operativo, encontrándose la CPU, la memoria central y los periféricos totalmente deslocalizados. Destacan Raspberry y BeagleBone Black

Por qué arduino

Arduino es una placa de prototipado que contiene:

- un microcontrolador AVR de Atmel: CPU+MEMORIA+PERIFÉRICOS E/S, todo en uno
- Un regulador de tensión para alimentar la placa y poco más
- Un convertidor serie-USB: se usa para grabar programas usando un ordenador, pero también se usa para comunicación serie placa-pc.
- Un puerto serie nativo o UART: para conectar dispositivos serie, como bluetooth o algunos sensores.
- Un puerto I2C: para conectar sensores y periféricos
- Un puerto SPI: para conectar sensores y periféricos

La memoria en arduino

- **SRAM**: Es dónde se almacenan las variables del programa. Esta se puede llenar y provocar "cuelgues". Es volátil.
- **EEPROM**: es no volátil. En ella se guardan datos que no se pueden cuando se resetea la placa, como las constantes de programa. A veces da problemas su borrado.
- **Flash**: dónde se almacena el programa que pasamos con el cable usb. También se almacena del bootloader. Usa la misma tecnología que las tarjetas SD, los pen drives o los discos duros SSD

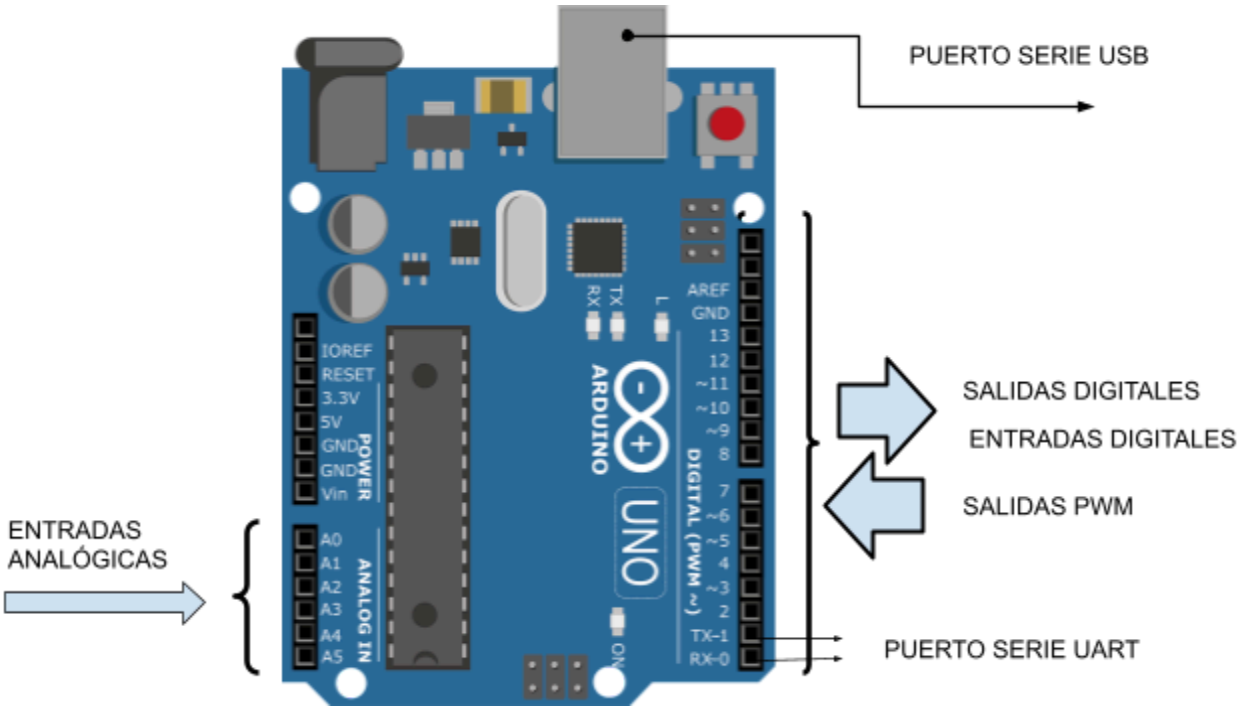
La gestión del tiempo

En un microcontrolador, la gestión de las interrupciones es en tiempo real, ya sean las interrupciones del “timer” o las de los sensores. No se suele colgar, a no ser que falle la memoria.

En un PC el sistema operativo gestiona el tiempo que la CPU dedica a cada aplicación. Puede darse el caso, y de hecho, pasa, que un programa consuma muchos recursos de la CPU o de la memoria y el PC “se cuelgue”. Por eso, en aplicaciones de mucha responsabilidad no se usan PC, o en aquellas con motores paso a paso.

1.- Sistema robótico

2.- Distribución de puertos en la placa Arduino



3.- Tipos de señales: analógicas y digitales

3.1 Señales digitales

- Aquella que posee un número finito de estados entre dos límites
- Arduino maneja señales digitales binarias, las cuales solo tiene dos estados entre dos límites: on/off, 0/1, 0V/5V, Verdadero/Falso
- Ejemplos: pulsadores, parado/en marcha

3.2.- Señales analógicas:

- Aquella que posee infinitos estados entre 2 límites.
- Ejemplo: claridad de día, el volumen de una conversación, la velocidad de un objeto . . .
- Arduino no puede procesar una señal analógica pura, por lo que la “simula”. El grado de perfección con la que simula una señal analógica o el número de estados intermedios entre dos límites se denomina resolución. Cuanto más resolución más se parece a la señal real o analógica.
    - R. de entrada: 10 bits (1024 estados entre 0V y 5V)
    - R. de salida: 8 bits (256 estados entre 0V y 5V)

4.- Variables: tipos, nombre y asignación de un valor

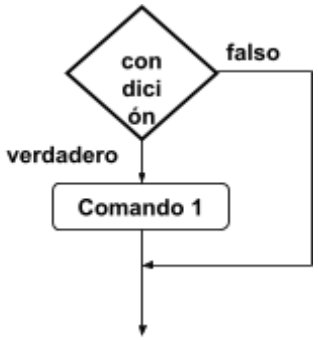
Una variable queda definida con tres parámetros:

1. Tipo variable
    a. **int** → almacena datos enteros
    b. **float** → almacena números con decimales
    c. **char** → almacena letras o caracteres
    d. **String** → almacena cadenas de caracteres (texto)
    e. **boolean** → almacena dos estados: true/false
2. Nombre de la variable
    No sirven palabras clave como time, delay, millis … Distingue entre mayúsculas y minúsculas.
3. Asignar un valor inicial a la variable (se puede quedar sin asignar)

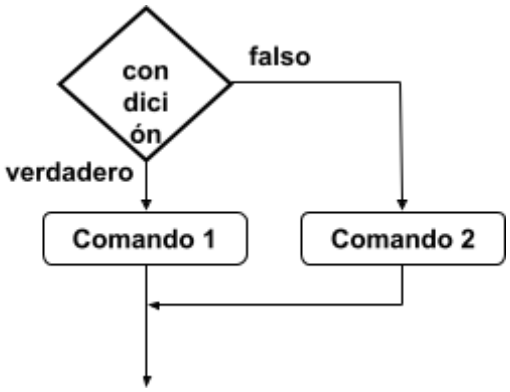
Ejemplos:

- int a = 0;
- int dato = 2;
- float un_numero = 7;
- char mi_letra = ‘q’;
- String dato = ‘no me gustan los lunes’;

5.- Sentencias de control




```
if (condición) {
    comando 1
            }
```




```
if (condición) {
 comando1
 } else {
  comando2
            }
```

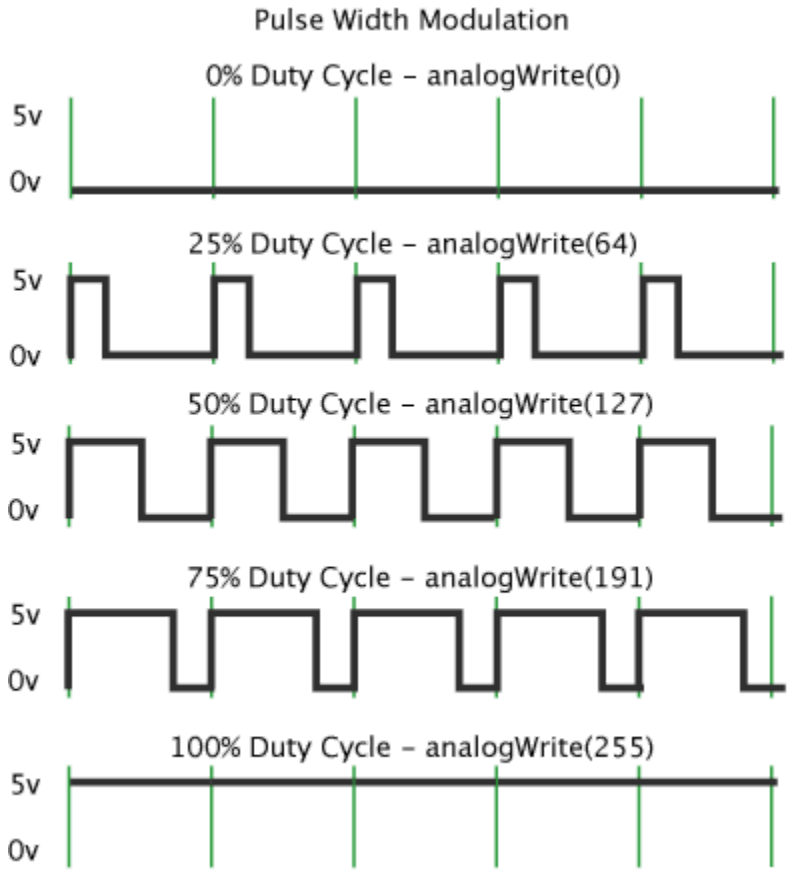
Dentro de las condiciones podemos usar los siguientes operadores lógicos:

- Operador AND: **if(condición1 && condición2 && condición3)**
- Operador OR: **if(condición1 || condición2 || condición3)**
- Operador IGUALDAD: **if(a==10)**
- Operador MAYOR/MENOR: **if(a>10)**
- Operador DIFERENTE: **if(a != 0)**

nota: si a es tipo “char”, la igualdad se hace: **if(a=='10')** *(notar que son comillas simples)*

6.- Salidas PWM, simulación señal analógica.

Arduino simula la intensidad de la señal analógica mediante una onda cuadrada modulada en amplitud denominada PWM. Esta onda cuadrada, tiene 2 valores, 0V y 5V. Su periodo es de 20ms. Dependiendo el ancho de pulso, varía la tensión efectiva, ver tabla:



6.1.- PWM como regulador de potencia

Podemos controlar la potencia de un motor o la intensidad de un led. No todos los pines admiten salida analógica, solo los pines marcados con ~

- configurar el pin como salida: **pinMode(nºpin, OUTPUT);**
- en el bucle ppal loop: **analogWrite(potencia);** 0-255 de resolución

6.2.- PWM control de servo

- *1.- Importamos la librería Servo.h*
- *2.- Nos creamos una variable de tipo Servo*
- *3.- Indicamos el nº pin al que conectamos la señal*
- *4.- Movemos el servo al ángulo deseado*

```
#include <Servo.h>
Servo mi_servo;
void setup() {
   mi_sevo.attach(9);
}
void loop() {
   mi_servo.write(180);
}
```

6.3.- PWM generadora de tonos

```
tone(nºpin, nota expresada en Hz, tiempo en milisegundos);
```

7.- Comunicación serie: Arduino-PC

• Abrir el puerto serie y sincronizar la velocidad del emisor con la del receptor	`Serial.begin(9600);`

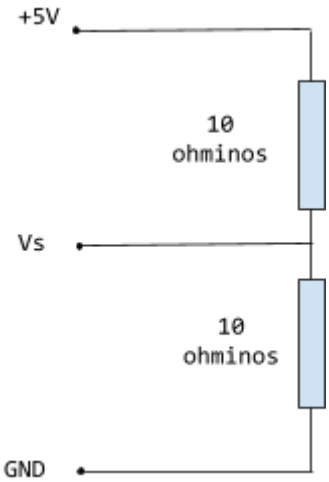
• Detectar si ha llegado algún dato al buffer	`if (Serial.available()) {`
	leo buffer
	`}`

• Leer el buffer y guardar el dato en una variable (dato es char)	`dato = Serial.read();`

• Enviar un dato por el puerto	`Serial.print()`
	`Serial.println()`
	`Serial.write`

8.- Entradas analógicas

8.1.-El divisor de tensión



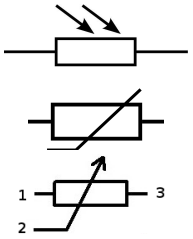
Para estudiar este apartado es necesario saber que es un divisor de tensión.

Divisor de tensión: la medida de la tensión (voltios) entre dos resistencias en serie.

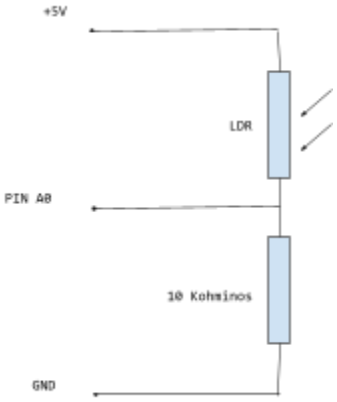
1. Calculamos la resistencia equivalente y con ella la corriente del circuito
2. Con la corriente cálculo la tensión Vs en la resistencia inferior.

Los sensores analógicos que se ven en la clase de tecnología se basan en el divisor de tensión compuestos por:

1. Una resistencia fija, de unos 10KΩ
2. Una resistencia variable cuyo valor depende de una magnitud física.
    - Resistencia LDR: su valor depende de la intensidad luminosa
    - Resistencia PTC o NTC: su valor depende de la temperatura
    - Potenciómetros: su valor depende de un giro



8.1.-Sensor de luz pull-down (resistencia fija abajo)



8.2.- Lectura de una entrada analógica

Los pines que admiten entrada analógica son A0, A1, A2, A3, A4 y A5. No es necesario configurar el setup, pues estos pines no sirven para otra cosa, que como entradas analógicas. Su resolución es de 1024.

```
analogRead(A0);
```

9.- Entradas digitales

Aquellas que solo pueden proporcionar dos estados: conectado/desconectado, 0V/5V, pulsado/sin pulsar, 0/1. A nivel de programación:

en setup: **pinMode(nºpin, INPUT);**
en loop: **digitalRead(nºpin);**

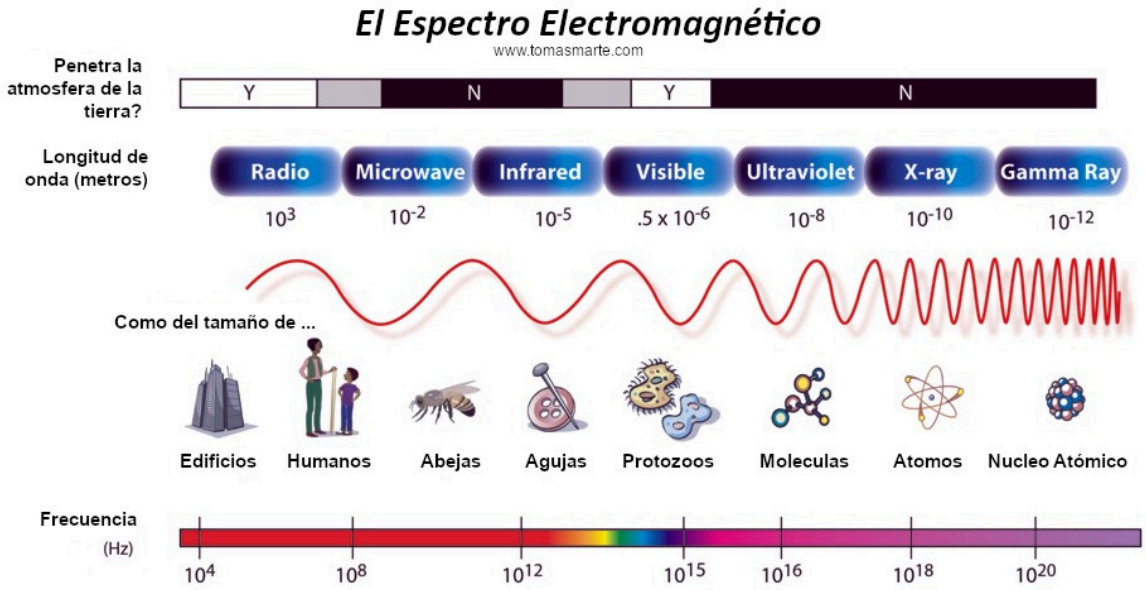
10.- Salidas digitales:

Aquellas que solo proporcionan dos estados: encendido/apagado, 0V/5V, verdadero/falso, 0/1. A nivel de programación:

en setup: **pinMode(nºpin, OUTPUT);**
en loop: **digitalWrite(nºpin, HIGH/LOW);**

11.- SENSOR INFRARROJOS IR

Trabajan en el rango del espectro electromagnético por debajo de la luz visible:

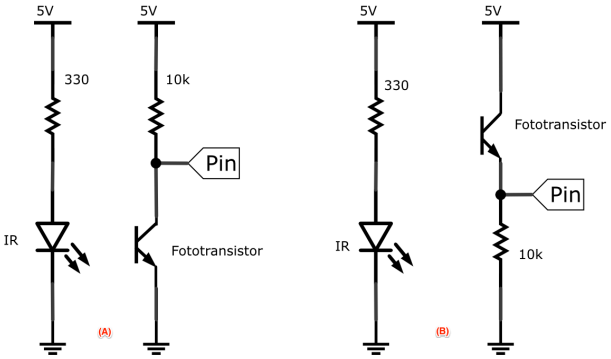


Características:

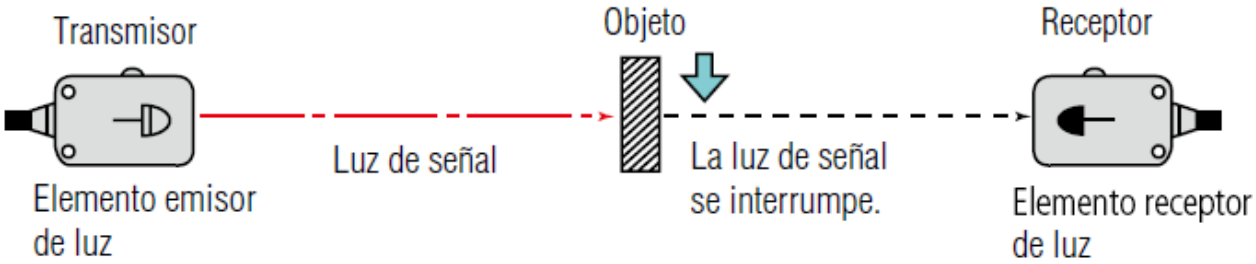
1. Son muy empleados en robótica porque son baratos
2. Son muy versátiles, sirven para: detectar obstáculos, medir distancias y detectar agún color, como el blanco/negro. Mandos a distancia.
3. No funcionan bien si les da el sol.

11.1.- Sensor de corte

Definición: Detectan si pasa un objeto o una persona por un determinado lugar al cortar un haz de luz. Es un sensor digital.



Funcionamiento: el emisor IR y el receptor se colocan enfrentados de forma que el rayo emitido inicie directamente en el receptor. El emisor IR suele ser un diodo IR, y el receptor un fototransistor o fotodiodo. El fototransistor funciona como una resistencia variable según la luz, es decir, como un LDR pudiendo tener el sensor una configuración pullup o pulldown. La señal generada en el PIN, es analógica pero se

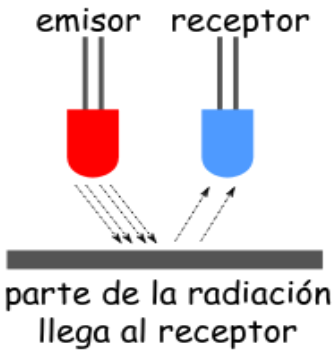


trata como digital: hay luz o no la hay.

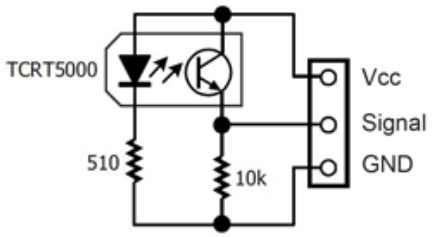
Usos: en barreras y en líneas de montaje.

11.2.- Sensor reflexivo

Definición: en este caso el emisor IR, y el receptor se colocan en el mismo lado.que se detecta es la reflexión del rayo infrarrojo.



Funcionamiento: dependiendo de cuánta luz reciba el receptor sabremos si el color del objeto en el que se refleja la luz es más claro o más oscuro. Vamos a detectar una señal analógica que será proporcional a la claridad del color del objeto

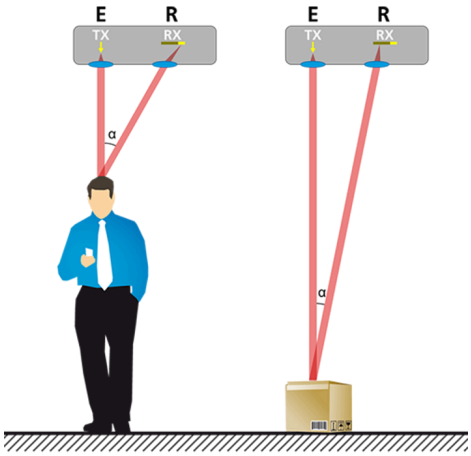


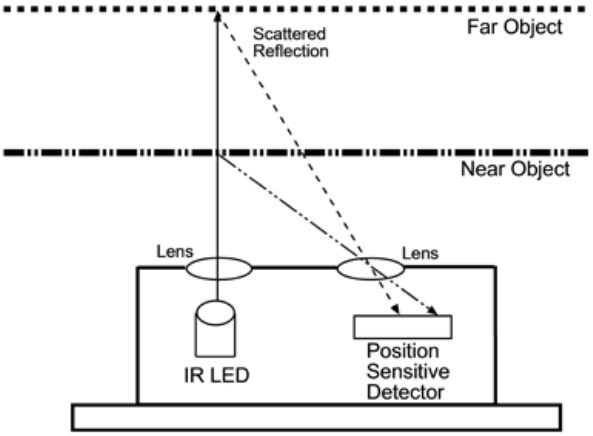
Usos: dependiendo de la cantidad de luz recibida optenemos una senal analógica, que puede servirnos para detectar el tono de gris de un objeto (blanco o negro). Su uso principal es para hacer seguidores de línea negra. También puede ser usado para medir distancias y detectar objetos

11.3.- Sensor reflexivo SHARP

Definición: sensor óptico infrarrojo capaz de medir la distancia entre él y un objeto de forma muy precisa.

Funcionamiento: el sensor con la ayuda de unas lentes puede conocer el ángulo de reflexión y calcular la distancia usando triangulación. Si el ángulo es grande, el objeto está cerca. Si el ángulo de reflexión es pequeño, el objeto está lejos





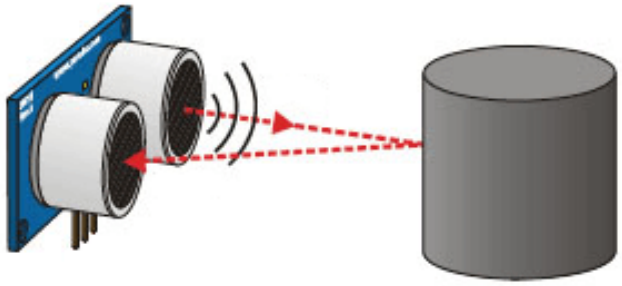
Usos: para medir distancias ya que es más preciso que el sensor de ultrasonidos en medias y largas distancias (20cm)

11.4.- Receptores de infrarrojos

Estos reciben señales codificadas y moduladas que contienen información. Se usa para mandos a distancia TV y juguetes RC. No se estudiarán en este apartado.

12.- SENSOR ULTRASONIDOS

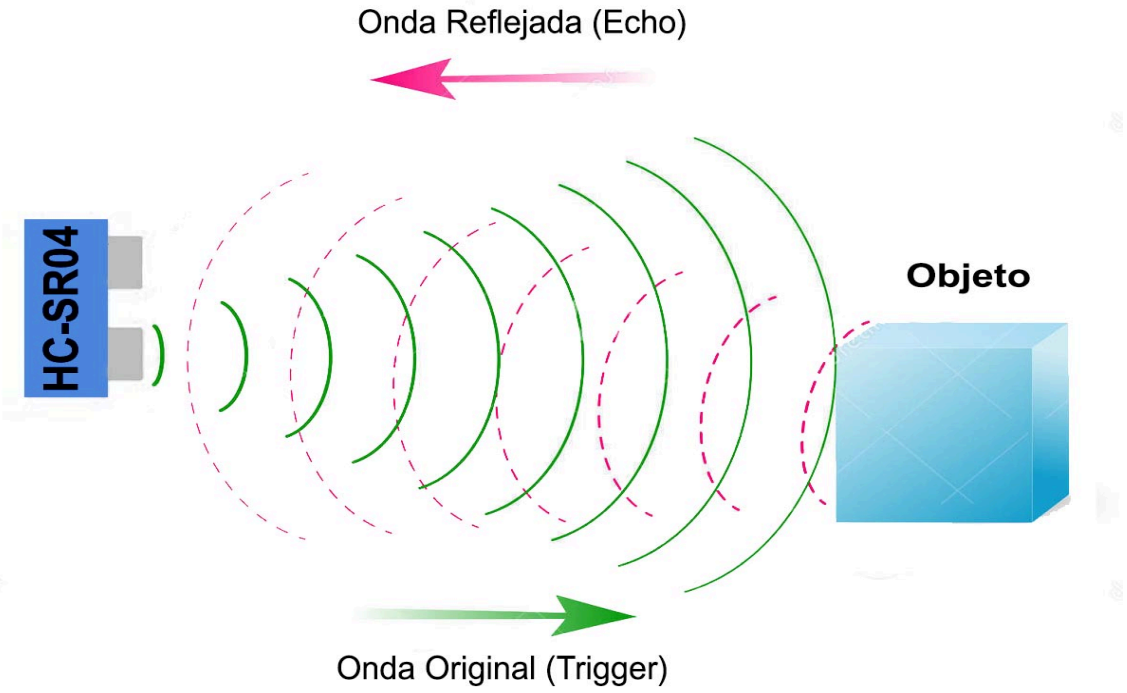
Definición y funcionamiento: sensor de distancia que consta de un emisor de ultrasonidos (no audible por el oído humano) y un receptor (micrófono). El sensor mide el tiempo que pasa desde que emite una onda, denominada trigger, hasta que la recibe (echo). Sabiendo que velocidad del sonido en la atmósfera es aprox. de 343 m/s podemos calcular la distancia con la siguiente fórmula;



Tiempo = 2 * (Distancia / Velocidad)

Distancia = Tiempo · Velocidad / 2

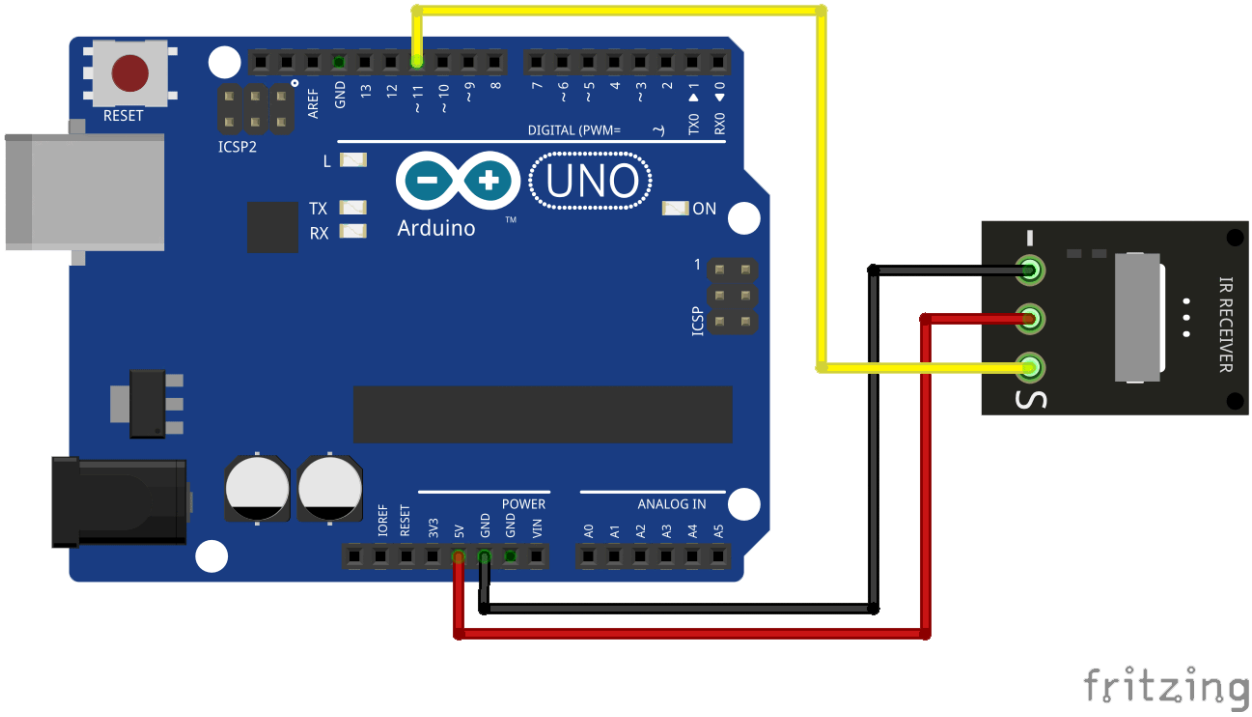
Usos: para medir distancias cortas, en las que los sensores SHARP no funcionan. Son poco precisos y no siempre funcionan bien, debido a que el rebote del sonido depende de la forma y el material de cada objeto. La velocidad del sonido en la atmósfera depende de la temperatura, humedad y presión.



13.- COMUNICACIÓN INFRARROJOS

Un mando a distancia transmite un cierto mensaje al receptor empleando luz infrarroja. Como en cualquier transmisión el mensaje tiene que seguir unas determinadas normas, el protocolo, que deben ser conocidas tanto por el emisor como por el receptor para que la comunicación sea correcta.

No existe un único protocolo adoptado como estándar. Tenemos el RC-5 y RC-6 de Philips, el SIRC de Sony, el protocolo NEC. Por otro lado, el mensaje nunca se envía directamente como un pulso, si no que se envía modulado, es decir, encapsulado sobre una onda portadora. Se mejoran las interferencias por ruido y a la luz ambiental.



Es muy complicado decodificar una onda de un mando por lo que usamos librerías, la más popular es IRreceiver.

PROGRAMA DESACTUALIZAD A ACTUAL LIBRERÍA

```
#include <IRremote.h>

int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
        {
                Serial.begin(9600);
                irrecv.enableIRIn();    // Start the receiver
        }
void loop() {
        if (irrecv.decode(&results)) {      //convierto los datos a hexadecimal
                Serial.println(results.value);
                irrecv.resume();                    // Receive the next value
        }
        delay(100);
}
```

Ejercicios para clase

1. Encender apagar led:
    a. Un led y variar frecuencia delay()
    b. Varios led's: semáforo, secuencias
2. Encender apagar led:
    a. Variable tiempo
    b. Variable tiempo va disminuyendo: tiempo = tiempo -1
    c. if tiempo == 0 { tiempo = 1000;} //reseteo
3. Controlar potencia de un led (usar led gordo, no olvidar que no todos los pines son PWM):
    a. Poner un led al mínimo de potencia sin que se apague
    b. Aumentar la luminosidad de un led a saltos (5 saltos, para que se noten)
    c. Aumentar la luminosidad de forma gradual brillo = brillo+1
    d. Aumentar y resetear luminosidad: if brillo = 255 {brillo =0}
    e. Simular latido corazón: if (brillo == 0 || brillo == 255) { sentido = sentido*-1;}
4. Prácticas de salida analógica con tip122 y motor
5. Control PWM servo
    a. Poner servo a 0º, 90º y 180º con delay( )
    b. Hacer segundero con servo.
    c. Hacer segundero con servo en sentido horario y anti-horario
6. Puerto serie
    a. Leer puerto serie: leer un 'cero' o 'uno' en monitor serie y encender led en consecuencia
    b. Leer puerto serie: leer un ángulo en monitor serial y girarlo en un servo
    c. Leer la resolución de un potenciómetro en el puerto serie
    d. Leer la resolución de un LDR → Ampliación: encender led si se hace de noche
7. Entradas analógicas
    a. Potenciómetro: regular la luz: función map()
    b. Potenciómetro: regular el ángulo servo

PROGRAMAS DE EJEMPLO

6 PUERTO SERIE: monitor serie → arduino

```
char dato;
void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop() {

  if (Serial.available()) {
    dato = Serial.read();
  }
    if(dato=='1') digitalWrite(13, HIGH);
    if(dato=='0') digitalWrite(13, LOW);

}
```

2 FUNCIONES BÁSICAS

```
int t=0;
void setup() {
  pinMode(13,OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(t);
  digitalWrite(13, LOW);
  delay(t);
  t=t+1;
  Serial.println(t);
}
```

```
int t=0;
void setup() {
  pinMode(13,OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(t);
  digitalWrite(13, LOW);
  delay(t);
  t=t+1;
  if(t==50){
    t=0;
    Serial.println("RESETEO");
    }
}
```

3 analogWrite( ) pin 11

```
int brillo = 1;
void setup() {
  pinMode(11, OUTPUT);
}

void loop() {
analogWrite(11, brillo);
brillo = brillo +1;
delay(100);
}
```

```
int brillo = 1;
int sentido =1;
void setup() {
  pinMode(11, OUTPUT);
}

void loop() {
analogWrite(11, brillo);
brillo = brillo +1*sentido;
delay(5);
 if (brillo == 0 || brillo == 255) {
     sentido = sentido*-1;
     }
}
```