Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

PATRISIO, EL PERRO ROBOT

Autora:

Carmen Guidet Gómez

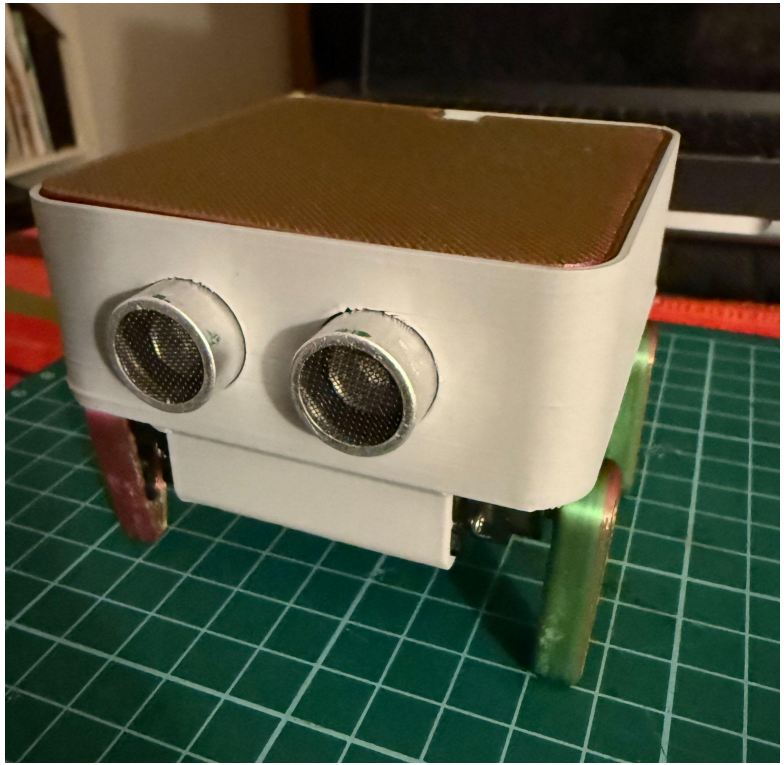
CÍVICA PEOPLE BEYOND TECH · CGI · GPU Solutions · rti · UNIT4 · ZENZORRITO TECNOLOGÍAS SL · amazon · Si2 Soluciones · Oficina de Software Libre · MONTAJES ELECTRICOS JIMÉNEZ · ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Inspiración

- Todos hemos visto robots en películas o videojuegos… pero ¿alguna vez has pensado en construir uno que camine, escuche su entorno y reaccione por sí mismo?
- En este proyecto vamos a crear nuestro propio PATRISIO, un pequeño robot con cuatro patas que será capaz de moverse, detectar obstáculos, emitir sonidos y obedecer órdenes con un mando.
- ¡vamos a programar su comportamiento como si fuera un pequeño ser robótico!

Presentación

Patrisio, el perro robot



- Cada proyecto consistirá en construir nuestro propio robot PATRISIO, un pequeño “compañero” con cuatro patas capaz de moverse e interactuar con su entorno.
- Tendremos un sistema de control programado en Arduino que actuará como el “cerebro” del robot, permitiéndole decidir cómo moverse, cuándo detenerse, cómo reaccionar ante obstáculos .
- También puedes personalizar su apariencia, su forma de moverse o añadir nuevas ideas. No hay un único resultado correcto… ¡cada Patrisio será único!

https://x.com/sciencegirl/status/2039953915207200928

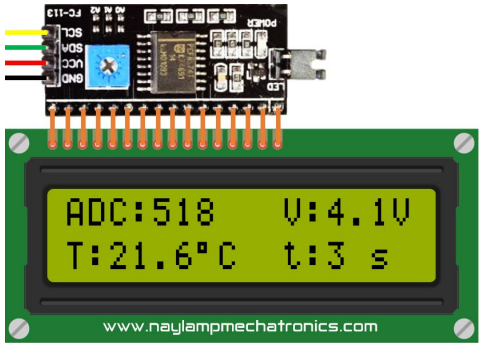
Oficina de Software Libre Universidad de Granada

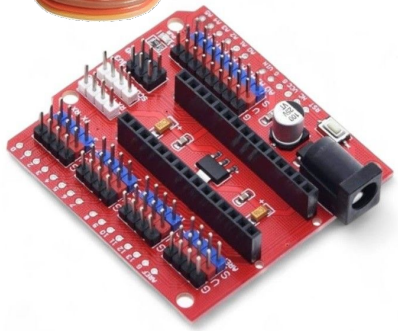
UNIVERSIDAD DE GRANADA

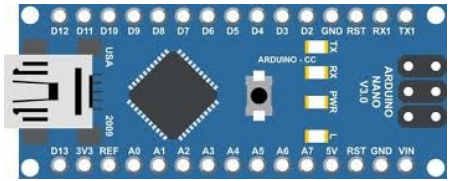
Materiales Necesarios

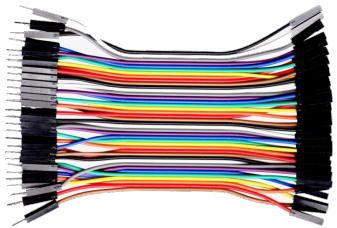
Materiales

- 1 Arduino Nano
- Arduino Nano Shield para servos + 9 Voltios
- Sensor de Distancia HC-05
- 4 servomotores de 180º
- 13 cables hembra-hembra
- Para ampliación:
  - Servomotor de 180º
  - Zumbador
  - Pantalla LED-I2C
  - Sensor de Infrarrojos IR + Mando









Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Fases del proyecto

Objetivo

El objetivo es llegar a un proyecto que funcione pronto y luego añadir funciones a medida que nos de tiempo.

¡Este método se usa mucho en ingeniería y es muy útil para todos los proyectos que hagáis!

Fases del proyecto

Fases

1. **Fase 1: Nace Patrisio**
   - Creación de Patrisio y calibración de los servos
2. **Fase 2: Sus primeros movimientos**
   - Sienta, de pie y tumba
3. **Fase 3: Patrisio aprende a caminar**
   - Andando y detección de obstáculos

Ampliaciones

4. **Fase Extra 1: Patrisio expresa emociones**
   - Movimiento de colita
5. **Fase Extra 2: Patrisio nos habla**
   - Pantalla LED que nos dice que hace Patrisio.
6. **Fase Extra 3: Patrisio se comunica**
   - Perrito ladrador (Zumbador + IR)

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

¡EMPEZAMOS!

Fase 1: Nace Patrisio

- Aprenderemos a:
  - Creamos a Patrisio
  - Usar el servo

Fase 1.1 Creando a Patrisio

Reto 1.1 - ¿Seríais capaces de crear el cuerpo de Patricio de cartón, si os doy las dimensiones? **La base de las patas debe de ser la misma, pero el resto puede ser como queráis** ¿Serias capaces de crear vuestro propio Patricio inspirandoos en lo que más os guste?

Fase 1.2 Montaje de Patrisio.

Reto 1.2 - Montando el cuerpo.

TIPS:

- Fijaos muy bien en la orientación de los servos.
- Para montarlo es necesario meter los cables por dentro.
- Cuidado con los tornillos. Los largos van a los lados y reservaremos el pequeño para después.

Fase 1.2 Montaje de Patrisio.

Reto 1.2 - Montando las patitas. Pegamos la aspa a la patita. Lo repetimos con las cuatro patas.

NO UTILICES PEGAMENTO, SI NO TENÉIS CLARA LA ORIENTACIÓN.

Servomotores

- **¿Qué es y cómo funciona un servomotor?**

  Un servo es un tipo de motor en el que indicamos directamente el ángulo que queremos y el servo se encarga de posicionarse en este ángulo.

- Típicamente los servos disponen de un rango de movimiento de entre 0 a 180º.

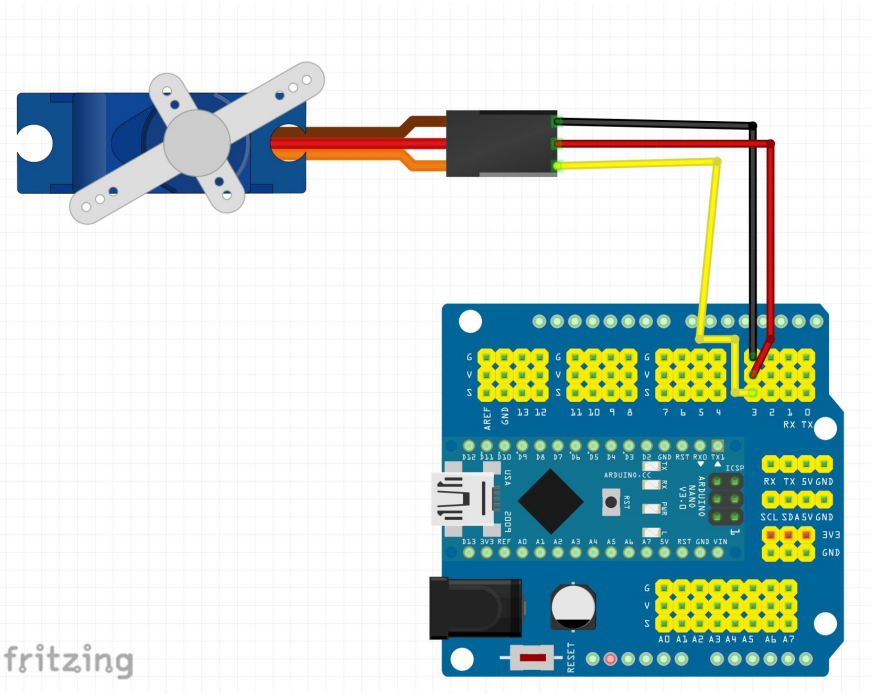
Cada servo motor será una de las patitas de nuestro robot

https://www.luisllamas.es/controlar-un-servo-con-arduino/

Servomotores

- **Servo motor 180º.**
  - Tienen 3 pines:
    - **DIN** pin de entrada **PWM,** tienen un **~**
    - **+5V** proporciona energía al servo. **VCC**
    - **GND** Toma de tierra.**GND**

¡Ojo! no te equivoques al conectarlo que puedes quemar el motor.



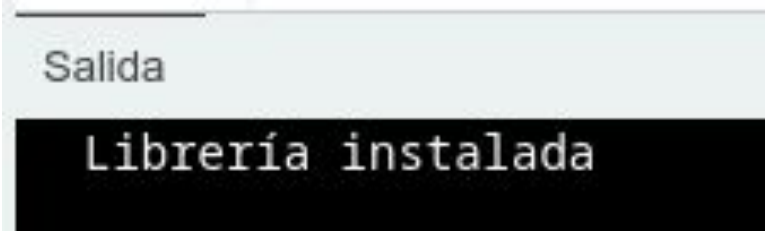
SERVOMOTORES

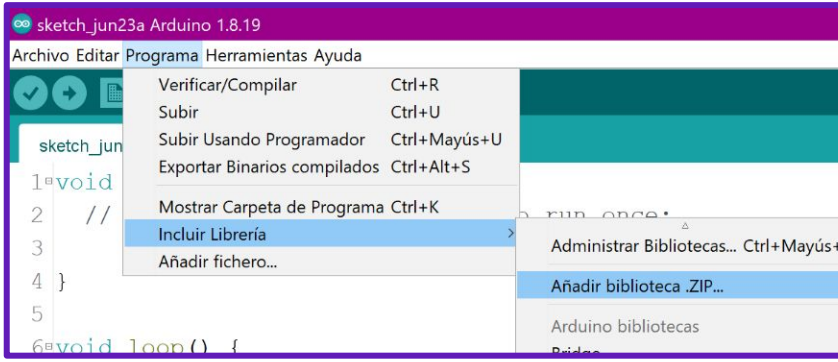
1. Descarga la biblioteca y después la importamos en el arduino:

Está en el enlace:

https://downloads.arduino.cc/libraries/github.com/arduino-libraries/Servo-1.2.2.zip

Sketch > Incluir biblioteca > Añadir biblioteca ZIP





Luego vamos a Sketch > Incluir biblioteca > Servo y nos aparece esto arriba del código

```
#include <Servo.h>
```

ServoMotores

- Las bibliotecas/librerías son código que otras personas crean para facilitar el uso de los componentes. Cuando llamamos a funciones de la biblioteca tenemos que tener cuidado en ver cómo se escriben y qué nos piden

```
Servo motor;      // crear un servo llamado motor
motor.attach(3); // decirle que trabaje en el pin 3
motor.write(90); // mover servo
```

Por ejemplo, aquí creamos un servo llamado motor.
Le decimos que trabajará con el pin 3
Y luego le decimos que se mueva a 90º

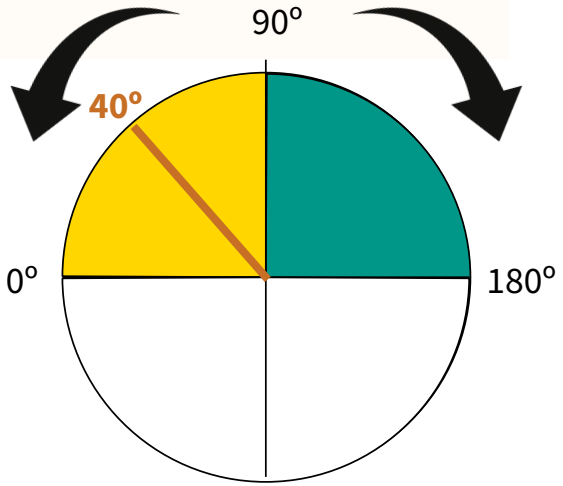
No os preocupéis si no lo entendéis del todo aún, ¡es un ejemplo de función de biblioteca!

ServoMotores

- Hay una función importante que debemos aprender antes de usar nuestro servomotor:

```
motor.write(dirección)
```

  - Cuando escribamos la dirección, el servo se moverá en ese sentido dando vueltas sin parar.
    - 180 es la velocidad máxima hacia la derecha
    - 0 es la velocidad máxima a la izquierda
    - 90 es un servomotor parado.
  - Si queremos que se mueva más lento, usamos grados más cercanos a 90 (por ejemplo, **40** iría girando más lentamente hacia la izquierda).



Servomotores

Este es un pequeño ejemplo de cómo funciona:

Los bucles for son para que gire cada vez más rápido/lento.

Dentro del bucle, motor.write(pos) es para que cambie el servo a esa velocidad.

```
for (int pos = 90; pos >=0; pos = pos - 1){
  motor.write(pos)
  delay(15);
}
```

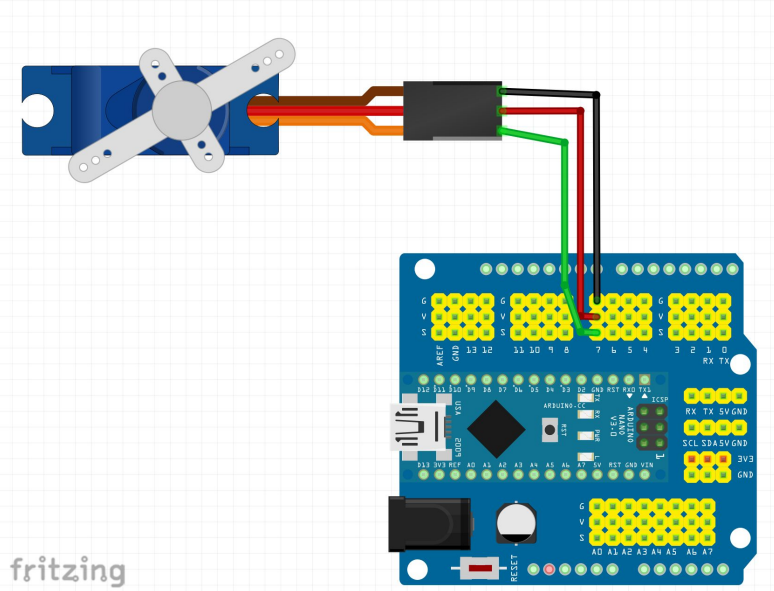
NOTA 1: Aquí empieza parado y va girando cada vez más rápido a la izquierda

```
for (int pos = 0; pos <=90; pos = pos + 1){
  motor.write(pos)
  delay(15);
}
```

NOTA 2: Aquí va reduciendo la velocidad hasta volver a 90 (parado).

Fase 1.3 - Servomotores

Reto 3 - Añadir los cuatro servomotores y ponerlos a 90 grados. Después de eso montar las patitas de Patrisio en 90º(para que quede de pie.)




```
calibracion §
#include <Servo.h>

const int pinDelanteraDerecha = 7;

Servo DelanteraDerecha;

void setup()
{
  DelanteraDerecha.attach(pinDelanteraDerecha);
  DelanteraDerecha.write(90);   // Mover el servo a 90 grados
}

void loop() {

}
```

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Montaje de patitas

Parece que el código del Reto 2 no nos ha funcionado para nada,¡PORQUE PARECE QUE NO HACE NADA!.

Pero de esta forma el Arduino pone los servos a 90º, es decir Patrisio está de pie.

De esta forma ya podemos tener control total de los ángulos de las patitas de Patrisio y no tener cada patita descontrolada.

FOTO DE PATRICIO DE PIE

Fase 2: primeros movimientos

- Aprenderemos a:
  - Hacer que Patrisio se siente, se tumbe y se ponga de pie.
  - Hacer funciones

Fase 2 - Patrisio se mueve

Reto 4 - ¿Seríais capaces de hacer un programa que sea capaz de una vez que Patricio este de pie sea capaz de sentarse? ¿Y tumbarse? Completa el codigo para sentarse y añade el codigo para estar tumbado.

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

```
#include <Servo.h>

const int pinDelanteraDerecha = 7;
const int pinDelanteraIzquierda = 6;
const int pinTraseraDerecha = 5;
const int pinTraseraIzquierda = 4;

Servo DelanteraDerecha;
Servo TraseraDerecha;
Servo DelanteraIzquierda;
Servo TraseraIzquierda;

void setando() {

}

void setup()
{
  DelanteraDerecha.attach(pinDelanteraDerecha);
  TraseraDerecha.attach(pinTraseraDerecha);
  DelanteraIzquierda.attach(pinDelanteraIzquierda);
  TraseraIzquierda.attach(pinTraseraIzquierda);

  // Posición inicial: de pie
  DelanteraDerecha.write(90);
  DelanteraIzquierda.write(90);
  TraseraDerecha.write(90);
  TraseraIzquierda.write(90);

  delay(1000);

  // Sentado
  TraseraDerecha.write(   );
  TraseraIzquierda.write(   );
  DelanteraDerecha.write(   );
  DelanteraIzquierda.write(   );

  delay(1000);
}

void loop() {
}
```

FUNCIONES

Ahora mismo tenemos código que nos hace cosas muy especificas, como que Patrisio se ponga de pie, se siente o se tumbe. **¿Y SI PUDIERAMOS PONERLE UN NOMBRE A TODAS ESAS ACCIONES?**

¡PODEMOS HACERLO Y LA FORMA DE HACERLO ES CON FUNCIONES!

Pero ¿Qué es una función?

Una función es un trozo de código que hace una tarea específica, le ponemos un nombre, y podemos llamarlo desde cualquier sitio de nuestro programa, además de que nos va a permitir tener el código más ordenado y facil de entender.

Imaginad una función como una receta de cocina a la que le ponéis un nombre.En lugar de explicar paso a paso cómo hacer una tortilla cada vez que tenéis hambre (“romper huevos”, “batir”, “calentar aceite”...), simplemente decís: Eh tú, haz una tortilla.

Funciones

De hecho, ¡ya estais usado funciones!

- **setup()** y **loop()** son funciones.
- **digitalWrite()** y **delay()** son funciones.

Estas son funciones que alguien escribió. Ahora vamos a aprender a escribir las nuestras.

La estructura básica para definir una función es:

```
tipoDevuelto nombreFuncion(parámetros) {
  // El código de la receta
  return datoDevuelto;
}
```

Normalmente usaremos funciones tipo void, es decir que no devuelven nada, pero también puede haber funciones que devuelvan datos.

Funciones

Se que es mucha información, pero vamos a verlo con un ejemplo con el programa para Patrisio, vamos a hacerle una función que sea que se ponga de pie:

```
#include <Servo.h>

const int pinDelanteraDerecha = 7;
const int pinDelanteraIzquierda = 6;
const int pinTraseraDerecha = 5;
const int pinTraseraIzquierda = 4;

Servo DelanteraDerecha;
Servo TraseraDerecha;
Servo DelanteraIzquierda;
Servo TraseraIzquierda;

// Escribimos nuestra funcion para que Patrisio se siente
void PatrisioDePie(){
  DelanteraDerecha.write(90);
  DelanteraIzquierda.write(90);
  TraseraDerecha.write(90);
  TraseraIzquierda.write(90);

  delay(1000);
}

void setup()
{
  DelanteraDerecha.attach(pinDelanteraDerecha);
  TraseraDerecha.attach(pinTraseraDerecha);
  DelanteraIzquierda.attach(pinDelanteraIzquierda);
  TraseraIzquierda.attach(pinTraseraIzquierda);

  // Llamamos a la funcion que hace que Patrisio se ponga de pie.
  PatrisioDePie();
```

El código que teníamos antes para que se pusiera de pie lo hemos pasado a una función.

¡Ahora sabemos al leerlo que hace ese código!

Llamamos a la función para que sepa el programa el trozo de código que tiene que ejecutar.

Fase 2 - Patrisio FUNCIONa

Reto 5 - Partiendo del ejemplo de funcion de PatrisioDePie(), ¿seriais capaces de hacer vuestras propias funciones de PatrisioSienta() y PatrisioTumba()? ¿Serias capaz de hacer que patricio haga de forma continua el patron DePie-Sienta-Tumba-Sienta?

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

```
#include <Servo.h>

const int pi
const int pi
const int pi
const int pi

Servo
Servo
Servo
Servo

// Escribimos nuestra funcion para que Patrisio se siente
void PatrisioDePie(){
  delay(1000);
}
void PatrisioSienta(){
  delay(1000);
}

void PatrisioTumba(){
  delay(1000);
}
```

Inicialización de Pines Patitas

Objetos Servos

Posición de Servos para estar de Pie

Posición de Servos para estar Sentado

Posición de Servos para estar Tumbado

Se os debería de haber quedado algo asi el codigo completo:

```
void setup()
{
}

void loop() {
}
```

Inicialización de los ServoMotores con su correspondiente PIn

Patrón De Pie- Sienta- Tumba- Sienta

Fase 3: primeros movimientos

- Aprenderemos a:
  - Hacer que Patrisio ande
  - Se detiene al ver obstaculos

Fase 3 - Patrisio aprende a andar

Podemos hacer que Patrisio ande de muchas formas: Moviendo las patitas delanteras y traseras contrarias a la vez, dando pequeños saltitos (más asociado a correr)...

¿SABÍAS QUE...?

Lo que estamos haciendo con Patricio forma parte de la **robótica bioinspirada**, una rama de la tecnología que estudia cómo se mueven los seres vivos para copiar esas ideas en robots.

Por ejemplo:

- Los perros caminan moviendo patas opuestas (como vamos a probar).
- Algunos insectos usan movimientos muy rápidos y coordinados.
- Otros animales se desplazan dando pequeños saltos.

Igual que la naturaleza ha perfeccionado estos movimientos durante millones de años, ¡nosotras vamos a imitarlos para que Patricio aprenda a andar!

Fase 3 - Patrisio aprende a andar

Reto 5 - Con todo lo que sabemos de mover servomotores, ¿seríais capaces de hacer que Patricio ande, moviendo las patas opuestas ? Como ayuda os dejo un pequeño código que tenéis que completar (debéis de mantener todo el código que ya tenemos) :

```
void loop() {

  // Fase 1
  DelanteraDerecha.write(      );
  TraseraIzquierda.write(60);
  DelanteraIzquierda.write(    );
  TraseraDerecha.write(60);
  delay(200);
  // Fase 2 (cambio directo, sin centro)
  DelanteraDerecha.write(80);
  TraseraIzquierda.write(      );
  DelanteraIzquierda.write(80);
  TraseraDerecha.write(      );
  delay(200);

}
```

Completa el código para que las patas opuestas se muevan a la vez. Sino te gusta como se mueve prueba distintas configuraciones, como moviendo todas las patas a la vez o dando saltitos.

¡ES HORA DE EXPERIMENTAR!

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Sensor de ULTRASONIDO

Un sensor ultrasónico mide la distancia a un objeto utilizando **ondas sonoras.**

Componentes Principales:

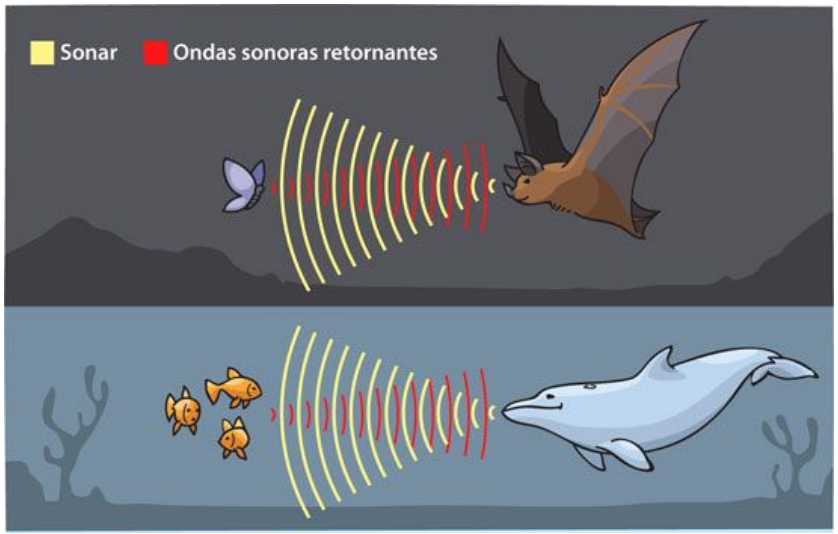
- Emisor: Envía un pulso ultrasónico.
- Receptor: Recibe el pulso reflejado desde el objeto.

El sensor envía un pulso ultrasónico y mide el tiempo que tarda en regresar después de rebotar en un objeto.

La distancia se calcula con la fórmula: ***Distancia = Tiempo * Velocidad del Sonido / 2.***

SENSOR ULTRASONIDO

Algunos animales, como los murciélagos y los delfines, utilizan una técnica muy similar al funcionamiento de este sensor, llamada **ecolocalización**.



Sensor ultrasonido

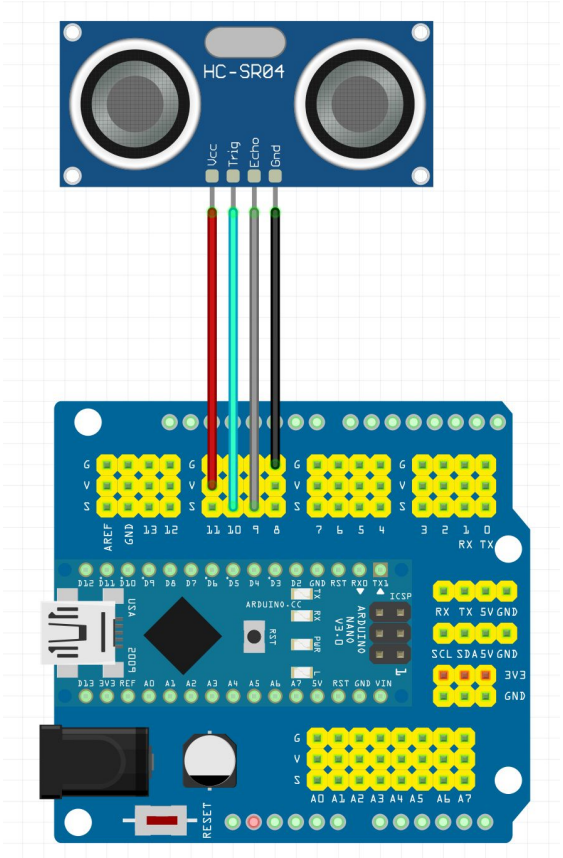
Conectamos el sensor al arduino:

- VCC al **pin de 5V**, positivo
- GND al **pin GND**, negativo
- TRIG al **pin 10.**
- ECHO al **pin 9.**

TRIG se utiliza para enviar un pulso ultrasónico.

ECHO se utiliza para recibir el pulso reflejado y medir el tiempo transcurrido.

NO DESCONECTES EL RESTO DE LAS COSAS



Sensor ultrasonido

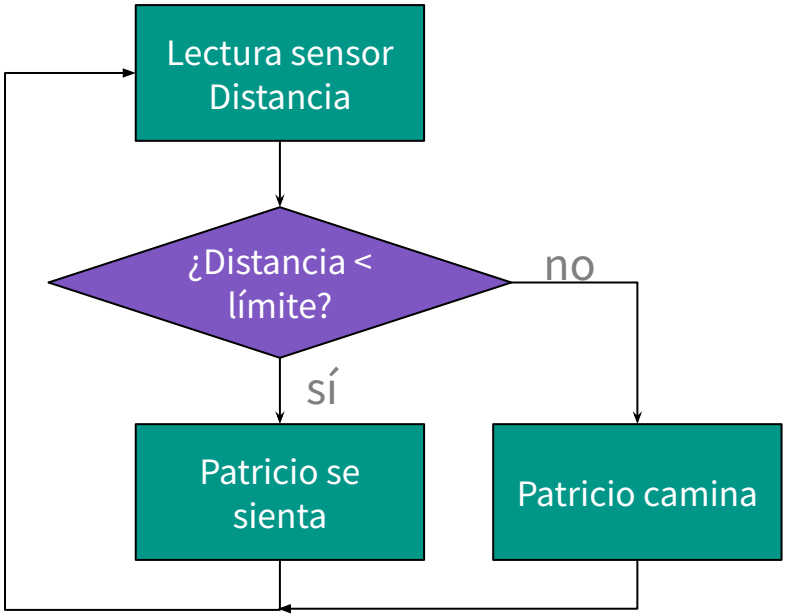
Este programa nos va a permitir hacer que patricio “vea”.
Abrid un nuevo proyecto en Arduino IDE y probad este código.
Tenéis que comprobar por Monitor Serie que el sensor es capaz de medir las distancias correctamente.


```
const int trigPin = 10;
const int echoPin = 9;

void setup() {

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  Serial.begin(9600);
}

void loop() {

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duracion = pulseIn(echoPin, HIGH);

  float distancia = duracion * 0.0343 / 2;

  Serial.print("Distancia: ");
  Serial.print(distancia);
  Serial.println(" cm");

  delay(500);
}
```

Fase 3 - Patricio aprende a evitar objetos

- **Parte 2**: Ahora lo que queremos hacer es que Patricio camine y cuando detecte un obstáculo a unos 10-15 cm se siente. Partiremos del programa del reto 5.

Lo que tenemos a la derecha se llama **Diagrama de Flujo**, es muy usado en programación y nos sirve para entender el programa.



Fase 3 - Patrisio aprende a evitar objetos

Reto 6 - Partiendo tanto del diagrama de flujo, como del pseudocódigo, ¿serias capaces de completar en Arduino el código para que Patrisio sea capaz de sentarse cada vez que detecte un obstáculo?

```
lectura de la distancia con el sensor
si (distancia > 0 y distancia < distanciaLimite){
        Patricio se sienta
}

sino{
        Patricio camina
}
```

PISTA: Tenéis que partir del código creado en el Reto 5, y seguir lo que dice el pseudocódigo, apoyándoos con el código del sensor de ultrasonidos.

Reto 6 Extra: ¿Serias capaz de hacer una función llamada medirDistanciaObjetos y que devuelva la distancia?

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!