Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

PATRISIO, EL PERRO ROBOT

Autora:

Carmen Guidet Gómez

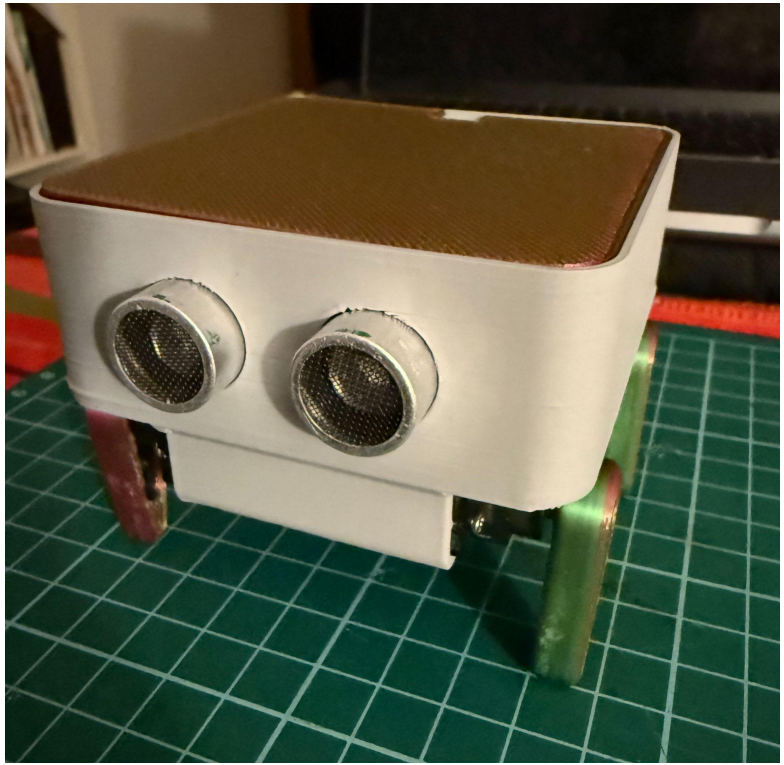
CÍVICA PEOPLE BEYOND TECH · CGI · GPU Solutions · rti · UNIT4 · ZENZORRITO TECNOLOGÍAS SL · amazon · Si2 Soluciones · Oficina de Software Libre · MONTAJES ELECTRICOS JIMÉNEZ · ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

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

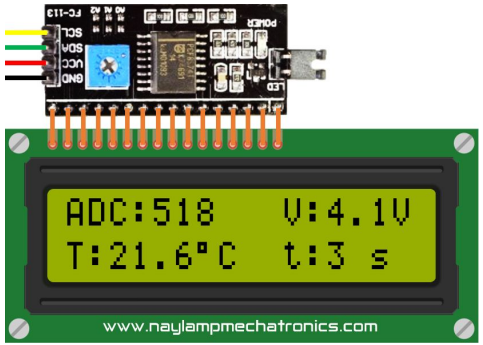
Oficina de Software Libre Universidad de Granada

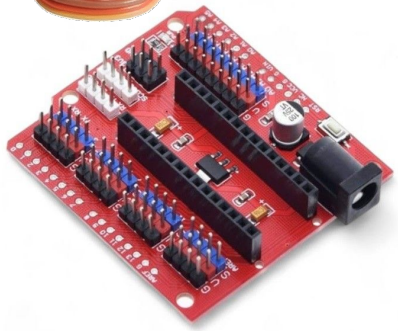
UNIVERSIDAD DE GRANADA

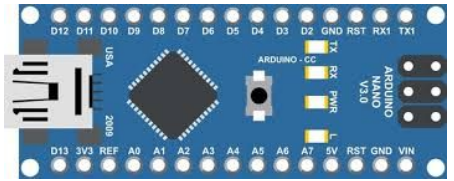
Materiales Necesarios

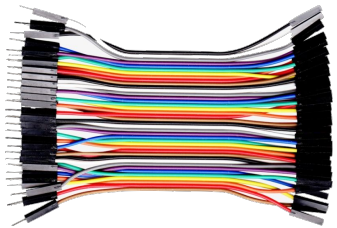
Materiales

- 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.

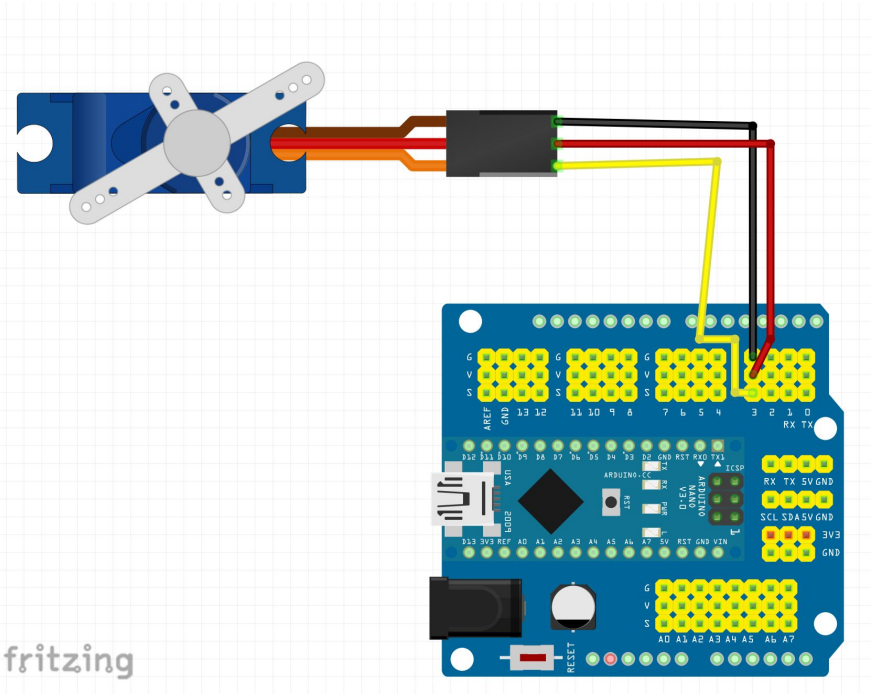
Cada servo motor será una de las patitas de nuestro robot

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.



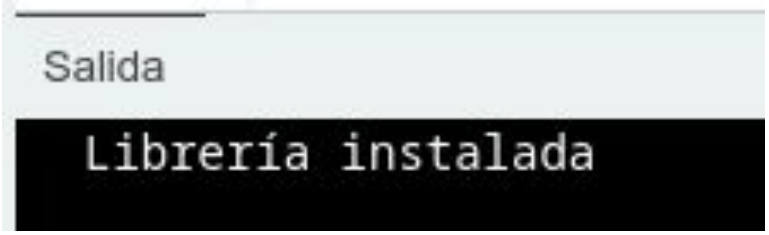
SERVOMOTORES

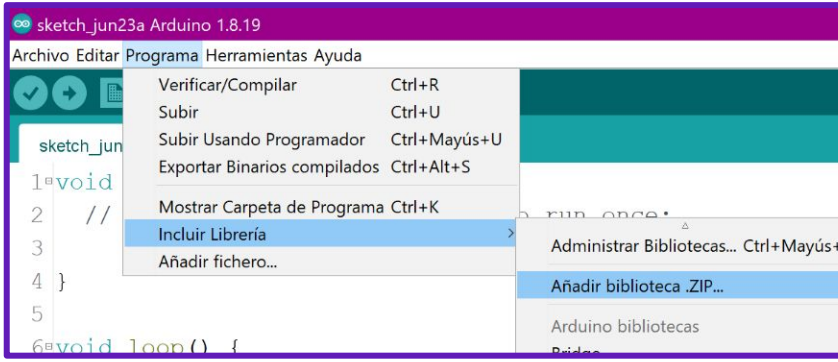
1. Descarga la biblioteca y después la importamos en el arduino:

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

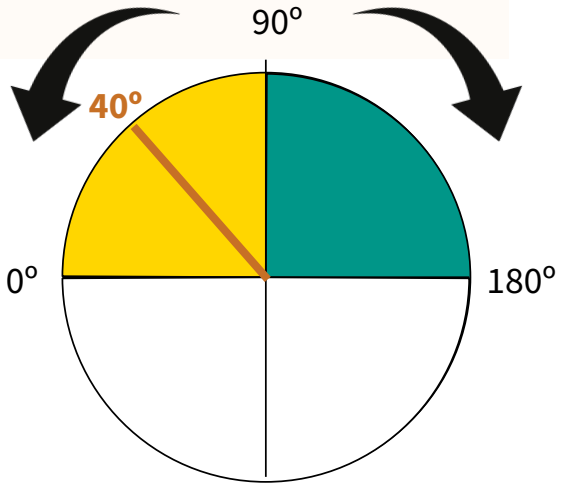
No os preocupéis si no lo entendéis del todo aún, ¡es un ejemplo de función de biblioteca!

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);
}
```

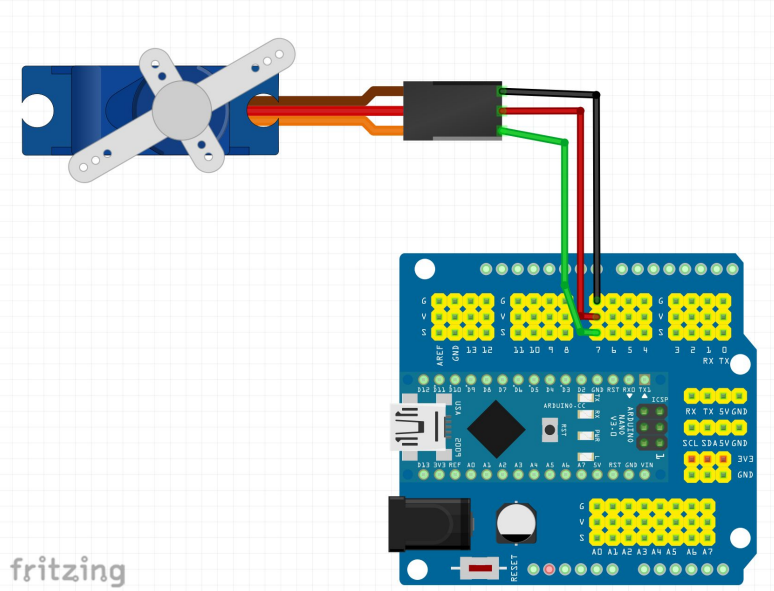
NOTA 1: Aquí empieza parado y va girando cada vez más rápido a la izquierda

```
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:

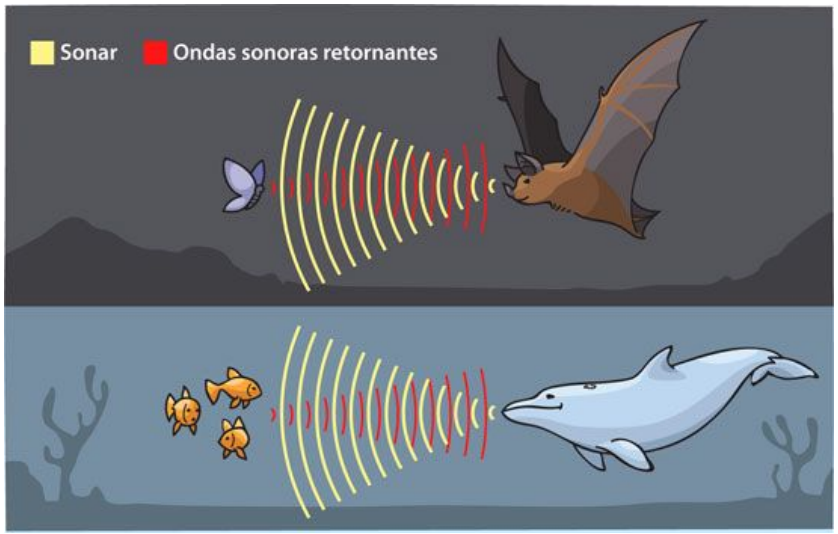
- Emisor: Envía un pulso ultrasónico.
- Receptor: Recibe el pulso reflejado desde el objeto.

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

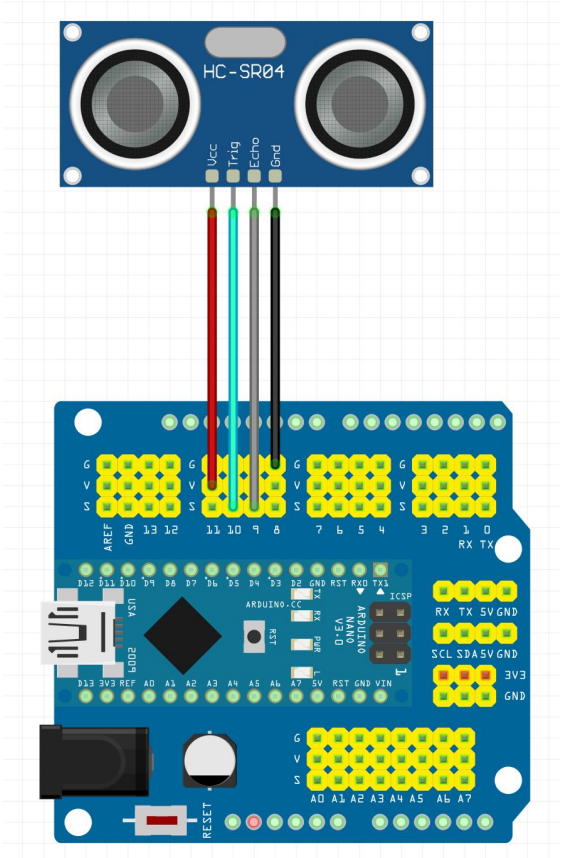
Conectamos el sensor al arduino:

- 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

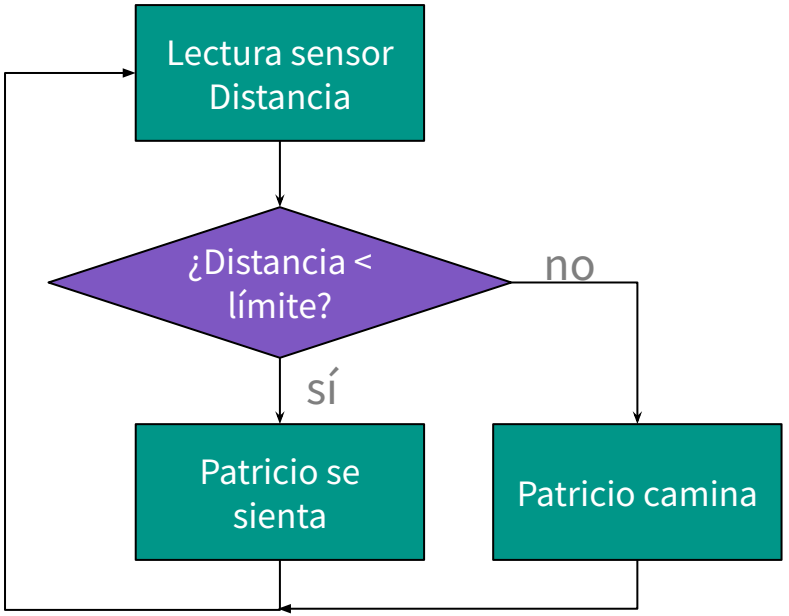
Este programa nos va a permitir hacer que patricio “vea”.
Abrid un nuevo proyecto en Arduino IDE y probad este código.
Tenéis que comprobar por Monitor Serie que el sensor es capaz de medir las distancias correctamente.


```
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!!

Fase Extra 1: Movimiento de cola

- En esta fase extra haremos que patricio mueva la cola mientra camina. ¡También puede ser las orejitas o la lengua!

Fase EXTRA 1 - ¡PATRISIO ESTA FELIZ!

Reto Extra 1 - Hasta ahora Patrisio puede caminar y reaccionar a obstáculos…pero todavía le falta algo importante para parecer un perro de verdad: **¡Mover la cola!** ¿Seríais capaces de añadir un nuevo servomotor para hacer que Patrisio mueva la cola de lado a lado?

Pistas:

- Necesitaréis conectar un nuevo servo
- Tendréis que crear una función para mover la cola
- Podéis alternar distintos ángulos para simular el movimiento

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

Extra extra:
¿Podríais hacer que:

- mueva la cola mientras camina?
- la mueva más rápido cuando “esté feliz”?
- o incluso usarla como orejitas o lengua?

Fase Extra 2: Patrisio se comunica

- En esta fase extra aprenderemos:
  - A usar la LCD
  - Integrar en nuestro proyecto.

Pantalla LCD

- **¿Qué es y cómo funciona una pantalla LCD?**

  Una **pantalla LCD** es una pantalla de retroiluminación LED que permite mostrar dos filas de 16 caracteres con los que podemos escribir texto.

- Realmente no vamos a aprender a conectar la pantalla LCD directamente, ya que tiene muchos pines y si la conectamos nos quedamos sin pines en el Arduino para todo lo demás. Por eso vamos a utilizar un **controlador I2C**.

En esta pantalla mostraremos mensajes de nuestra hucha.

Pantalla LCD

- ¿**Que es I2C y para qué sirve**?

  El controlador I2C nos permite controlar la pantalla utilizando solamente dos cables de control (además del cable de 5v y la toma a tierra GND) a través de un tipo de comunicación entre placas llamada I2C.

- Los dos cables utilizados son SCL, que se conecta al pin analógico A5, y SDA, que se conecta al pin analógico A4:

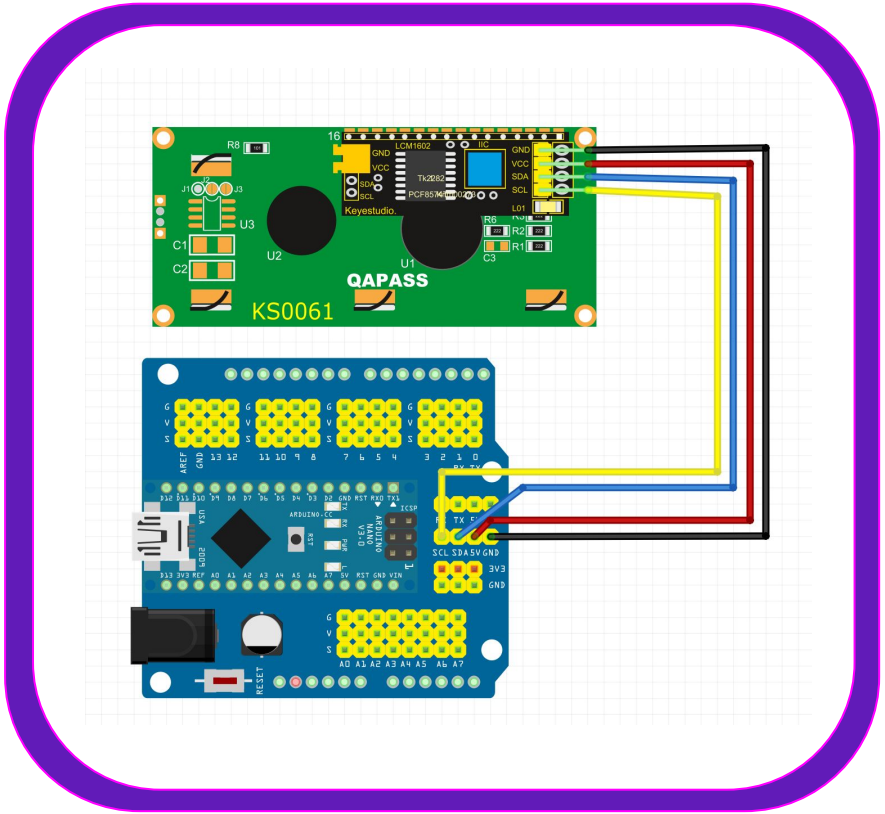
- SCL envía señales de reloj (clock), que se encargan de decidir quién habla en cada momento, para evitar conflictos y SDA envía los datos.

Pantalla LCD

En esta pantalla **mostraremos todo lo que queramos e indicaremos que esta haciendo Patricio.**

Recuerda los pines son:

SCL-> SCL
GND-> GND
SDA-> SDA
VCC-> 5V



Pantalla LCD

1. Incluir en el IDE la biblioteca LiquidCrystal_I2C.h

- Hay **dos formas de incluir bibliotecas** a la biblioteca de nuestro arduino.
  - Si tenemos el archivo comprimido con extensión .zip
  - Buscarlas en el repositorio de Arduino

- Para la pantalla tenemos el archivo comprimido con el nombre **LiquidCrystal_I2C-1.1.2.zip**
- La **descargamos** y la incluimos de la 1º forma que vemos a continuación.

Pantalla LCD

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

Está en el enlace:
http://downloads.arduino.cc/libraries/github.com/marcoschwartz/LiquidCrystal_I2C-1.1.2.zip

Pantalla LCD

- Para poder utilizar la pantalla con I2C además de descargarnos la librería de Arduino llamada **<LiquidCrystal_I2C.h>** , también necesitamos cargar otra llamada **<Wire.h>**.

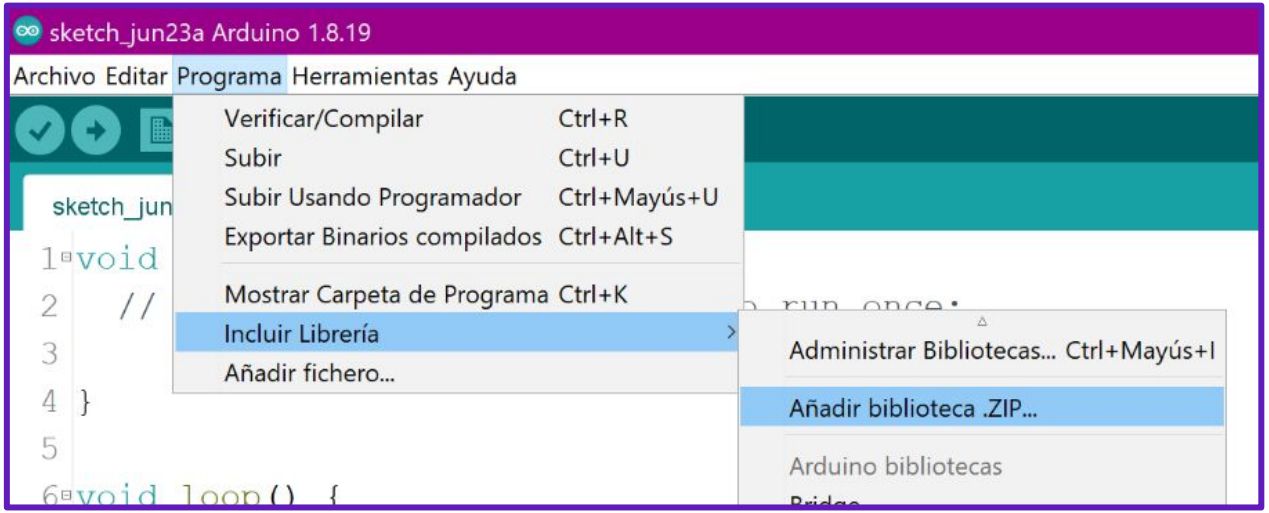
- Al trabajar con funciones de la librería descargada, a las que no les hemos puesto el nombre nosotras, tenemos que acostumbrarnos y entender lo que hacen.

NOTA: Las librerías son como funciones creadas por otras personas y que nosotras podemos aprovechar y utilizar siempre que lo necesitemos.

Pantalla LCD

1. Cargar librerías en el arduino:

En arduino hacemos clic en **Programa**-> **incluir libreria**-> **añadir biblioteca Zip.**

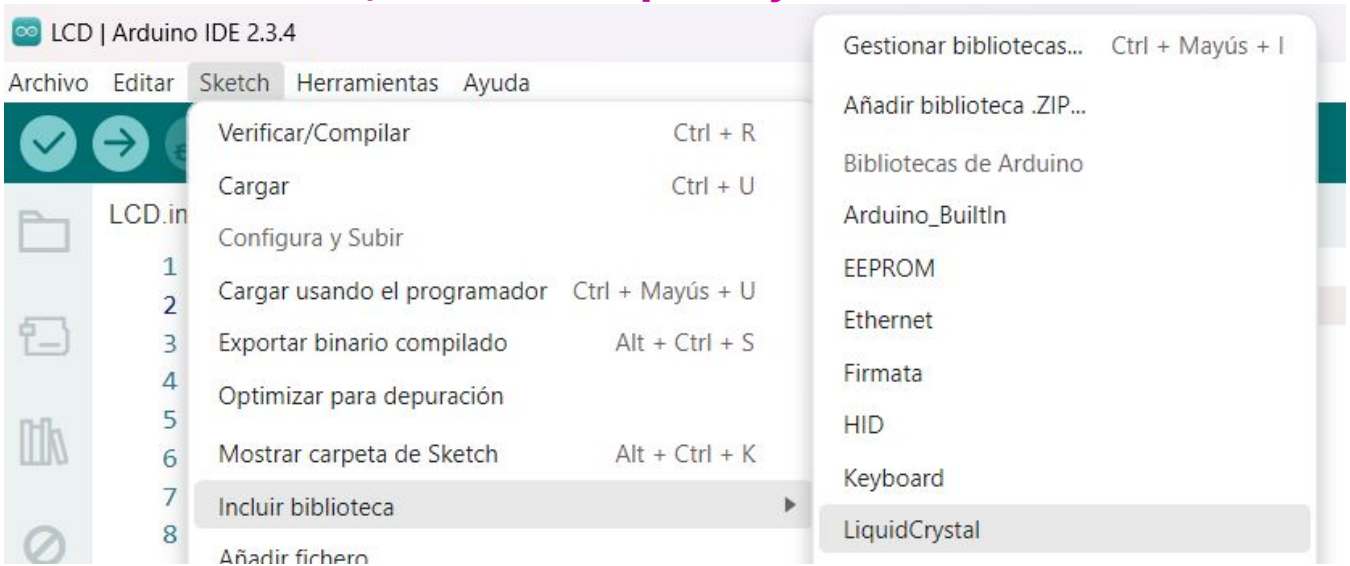


Pantalla LCD

1. Cargar librerías en el arduino:

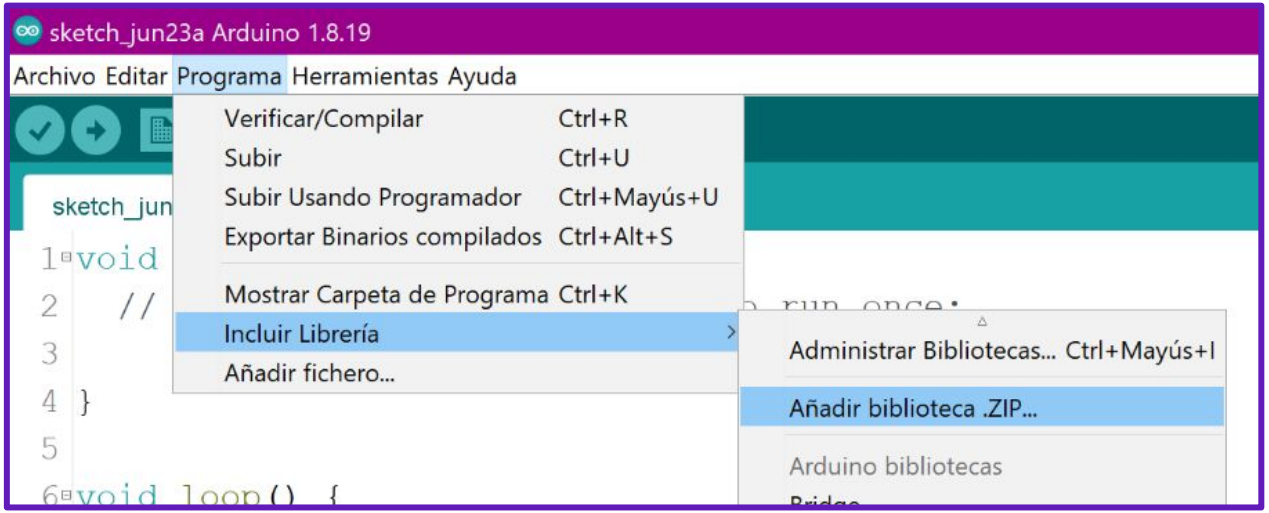
una vez añadida, para poder usarla hay que ir a:

Programa-> **incluir biblioteca/libreria**-> **LiquidCrystal_I2C**



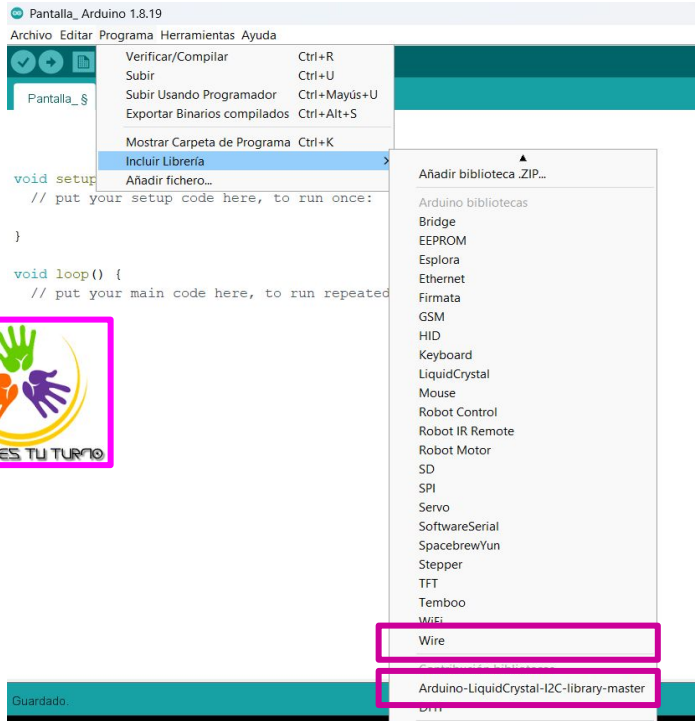
Pantalla LCD

1. Incluir en el IDE la librería **LiquidCrystal_I2C.h** ✓



Pantalla LCD

2. Para poder usar la librería tenemos que añadirla en el archivo. Vamos a añadir también wire.



Pantalla LCD

3. Declaramos un objeto basándonos en nuestra pantalla

- Para ello utilizaremos la siguiente **función de la librería LiquidCrystal_I2C** que acabamos de incluir.
  - **LiquidCrystal_I2C lcd(0x27, 16, 2);**
- Se inicializa con:
  - 0x27 porque se inicializa en esa dirección
  - 16 = número de caracteres por línea
  - 2 = número de líneas

- Vale pero, ¿qué tenemos que hacer y cómo? Poco a poco

Pantalla LDC

4. Aprendemos a usar la librería para mostrar mensajes por pantalla:

- Para poder utilizar nuestra pantalla vamos a tener que aprender las funciones de la librería:
  - **lcd.init()** se utiliza para **inicializar la comunicación con la pantalla** LCD.
  - **lcd.backlight()** **enciende la retroiluminación** de la pantalla LCD.
  - **lcd.setCursor(0, 0)** establece la **posición del cursor en la columna 0 y la fila 0**.
  - **lcd.print("Hola")** muestra el **texto "Hola" en la pantalla** LCD.
  - **lcd.clear()** **borra** el contenido de la pantalla LCD.
  - Esas son las básicas, pero hay alguna más a ver si eres capaz de localizarlas ;)

¿Os acordáis de los vectores?

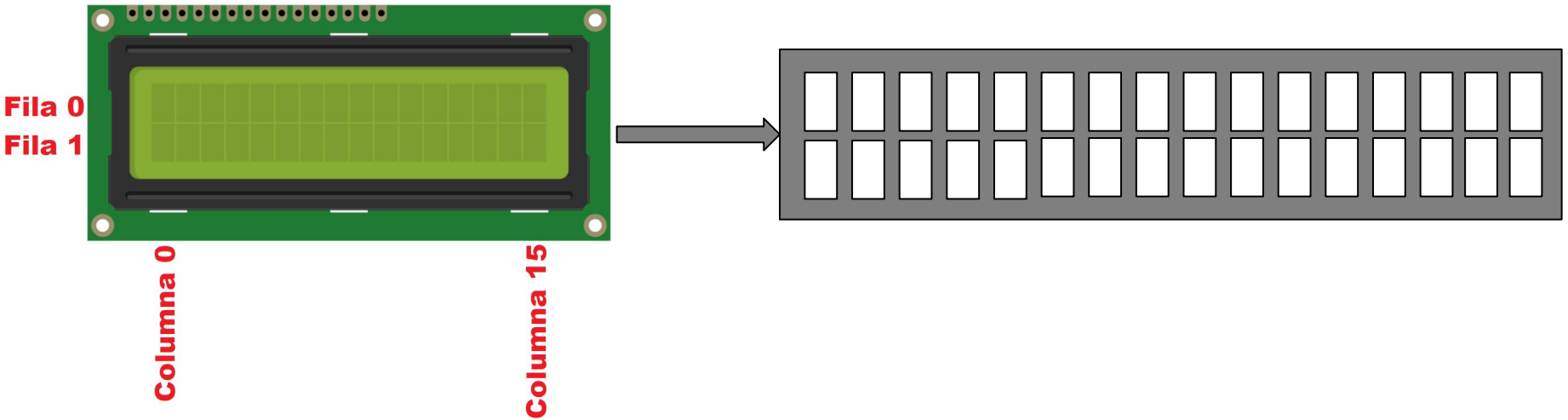
¿Creéis que podríais hacer que la pantalla muestre en la primera línea “Hola”?

Pantalla LCD

4. Aprendemos a usar la librería para mostrar mensajes por pantalla:

EXTRA:

- Vamos a entrar más en profundidad en cómo se muestran las letras en pantalla:
  - Siempre se escribe de la fila 0 posición 0 a la fila 0 posición 15
  - segunda fila es la fila 1 posición 0 a la fila 1 posición 15

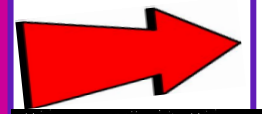


Pantalla LDC

4. Aprendemos a usar las funciones de la librería para mostrar mensajes por pantalla:

- **lcd.init()** inicia la pantalla
- **lcd.backlight()** enciende la luz
- **lcd.setCursor(0, 0)** posiciona el cursor
- **lcd.print("Hola")** escribe en la pantalla
- **lcd.clear()** borra de la pantalla

NOTA: Traduce este pseudocódigo a código real con las funciones que te hemos enseñado



pantalla

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
  // Inicia el LCD en la dirección 0x27, con 16 caracteres y 2 líneas

void setup()
{

      inicio la pantalla
      luz de fondo
 }
 void loop(){
      empiezo a escribir en línea0 pos0
      añado el texto que quiera
      espero 2 segundos
      limpio pantalla
 }
```

Pantalla LCD

Aquí tenéis un ejemplo de cómo usar la pantalla Lcd, como referencia. ¡Probadlo! Os ayudará a entender mejor el lcd.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd ( 0x27, 16, 2 );
void setup() {
  // put your setup code here, to run once:
    lcd.init ( );
}

void loop() {
  // put your main code here, to run repeatedly:
lcd.backlight ( );                  /*~ Encender la luz de fondo. ~*/
    delay ( 1000 );                   /*~ Esperar 1 segundo. ~*/

  lcd.noBacklight ( );                /*~ Apagar la luz de fondo. ~*/
    delay ( 1000 );                   /*~ Esperar 1 segundo. ~*/

  lcd.backlight ( );                  /*~ Encender la luz de fondo. ~*/
  lcd.setCursor ( 0, 0 );             /*~ ( columnas, filas) Ubicamos el cursor en la primera posición(columna:0) de la primera línea(fila:0) ~*/
  lcd.write ( 0 );                    /*~ Mostramos nuestro primer icono o caracter ~*/
    delay ( 1000 );                   /*~ Esperar 1 segundo ~*/
  lcd.clear ( );                      /*~ Limpiar pantalla ~*/

  lcd.setCursor ( 0, 1 );             /*~ ( columnas, filas) Ubicamos el cursor en la primera posición(columna:0) de la segunda línea(fila:1) ~*/
  lcd.write ( 1 );                    /*~ Mostramos nuestro segundo icono o caracter ~*/
    delay ( 1000 );                   /*~ Esperar 1 segundo ~*/
  lcd.clear ( );                      /*~ Limpiar pantalla ~*/

  lcd.setCursor ( 0, 0 );             /*~ ( columnas, filas) Ubicamos el cursor en la primera posición(columna:0) de la primera línea(fila:0) ~*/
  lcd.print ( "Bienvenido" );         /*~ Mostrar una cadena de texto (no exceder 16 caracteres por línea)~*/
   delay ( 1000 );                    /*~ Esperar 1 segundo ~*/

}
```

Fase EXTRA 2 - ¿Qué está pensando Patrisio?

Reto Extra 2 - Ahora que Patrisio puede:caminar, detectar obstáculos, y sentarse cuando "ve" algo… **¡vamos a hacer que también pueda comunicar lo que está haciendo usando la pantalla LCD!** ¿Seríais capaces de combinar el programa de la Fase 3 con la pantalla LCD para mostrar mensajes según el comportamiento de Patrisio?

Pistas:

- Usad lcd.print() para escribir mensajes
- Usad lcd.clear() para borrar la pantalla
- Pensad en qué parte del programa debe cambiar el mensaje

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

Extra extra:
¿Podríais:

- añadir mensajes divertidos?
- mostrar la distancia detectada?
- o hacer una animación cambiando el texto?

Fase Extra 3: Patricio Obediente

- En esta fase extra aprenderemos:
  - Usar el control IR
  - Usar el Buzer

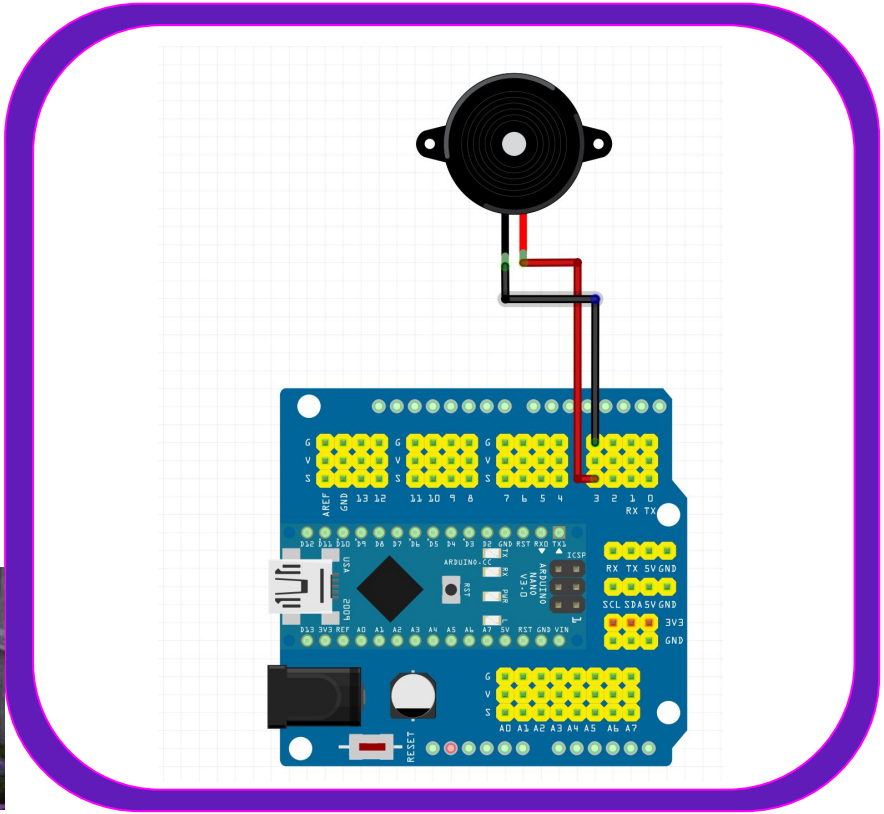
Buzzer

- Usaremos ahora el zumbador(*buzzer*) para crear una melodía:

★ Esto simulara que nuestro patricio ladra o habla.

★ También puede modificar la melodía cuando cambie de acción.

El buzzer se conecta al pin 3



Buzzer

- Los buzzer tienen dos funciones principales y una de ellas tiene 2 versiones:

```
tone(pin, frecuencia); //activa un tono de frecuencia determinada en un pin dado
tone(pin, frequencia, duracion);  //activa un tono de frecuencia y duracion
                                  //determinados en un pin dado
noTone(pin);  //detiene el tono en el pin
```

Es importante entender que como la función tone usa el Timer 2 de arduino, **los pines 3 y 11 no funcionarán mientras suene el buzzer** (para que lo tengamos en cuenta con nuestro motor)

Ahora después os enseñaré un código de ejemplo de cómo funcionan estas funciones

Buzzer

Vamos a ver un ejemplo.

En este código, ponemos el pin del buzzer en el pin 3

Le pedimos que genere tonos de diferentes frecuencias usando delay y noTone

Y usamos la versión alternativa de la función tone para que tenga en cuenta la duración:

```
#define BUZZER 5

void setup()
{
}

void loop()
{
  //generar tono de 440Hz durante 1000 ms
  tone(BUZZER, 440);
  delay(1000);

  //detener tono durante 500ms
  noTone(BUZZER);
  delay(500);

  //generar tono de 523Hz durante 500ms, y detenerlo durante 500ms.
  tone(BUZZER, 523, 300);
  delay(500);
}
```

¿Qué pasa si quiero hacer una melodía?

Cuando queremos hacer una melodía necesitamos decir las notas específicas y su duración. **¿Cómo hacemos eso?**

Uno de ellos tendrá nuestras **notas** y otro las **duraciones**, y luego los recorreremos con un for.

Lo primero que veremos son las notas y cómo definirlas.

¿Qué pasa si quiero hacer una melodía?

Primero vamos a ver algunos conceptos básicos.

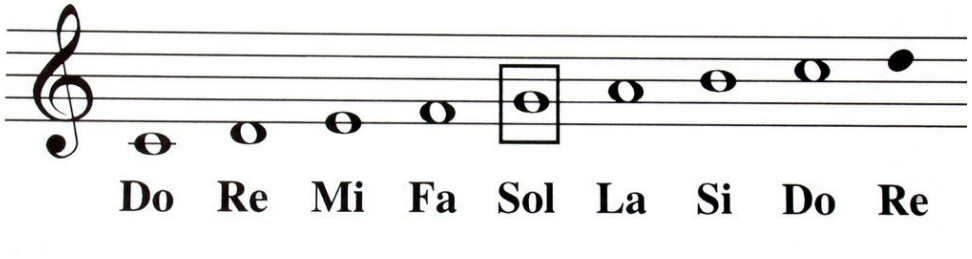
Las notas musicales pueden denominarse de dos formas:

- Sistema latín: Do, re, mi…
- Sistema anglosajón: A, B, C…

Os dejo en la foto una referencia de cómo es cada nota en los dos

Esto es útil para saber qué notas estamos tocando en las tablas de referencia.

```
// C++ code
//
#define BUZZER 5

/*
A - LA
B - SI
C - Do
D - Re
E - Mi
F - Fa
G - Sol
*/
```



¿Qué pasa si quiero hacer una melodía?

Con tablas como estas podemos traducir las notas de cualquier melodía a frecuencias, que será lo que necesitaremos para nuestro buzzer:

(podéis ignorar los decimales o redondearlos)

Music Note To Frequency Chart

MixButton

NOTE	OCTAVE 0	OCTAVE 1	OCTAVE 2	OCTAVE 3	OCTAVE 4	OCTAVE 5	OCTAVE 6	OCTAVE 7	OCTAVE 8
C	16.35 Hz	32.70 Hz	65.41 Hz	130.81 Hz	261.63 Hz (A piano middle C)	523.25 Hz	1046.50 Hz	2093.00 Hz	4186.01 Hz (A piano's highest note)
C#/D♭	17.32 Hz	34.65 Hz	69.30 Hz	138.59 Hz	277.18 Hz	554.37 Hz	1108.73 Hz	2217.46 Hz	4434.92 Hz
D	18.35 Hz	36.71 Hz	73.42 Hz	146.83 Hz	293.66 Hz	587.33 Hz	1174.66 Hz	2349.32 Hz	4698.63 Hz
D#/E♭	19.45 Hz	38.89 Hz	77.78 Hz	155.56 Hz	311.13 Hz	622.25 Hz	1244.51 Hz	2489.02 Hz	4978.03 Hz
E	20.60 Hz	41.20 Hz (A bass's lowest note)	82.41 Hz (A guitar's lowest note)	164.81 Hz	329.63 Hz	659.25 Hz	1318.51 Hz	2637.02 Hz	5274.04 Hz
F	21.83 Hz	43.65 Hz	87.31 Hz	174.61 Hz	349.23 Hz	698.46 Hz	1396.91 Hz	2793.83 Hz	5587.65 Hz
F#/G♭	23.12 Hz	46.25 Hz	92.50 Hz	185.00 Hz	369.99 Hz	739.99 Hz	1479.98 Hz	2959.96 Hz	5919.91 Hz
G	24.50 Hz	49.00 Hz	98.00 Hz	196.00 Hz (A violin's lowest note)	392.00 Hz	783.99 Hz	1567.98 Hz	3135.96 Hz	6271.93 Hz
G#/A♭	25.96 Hz	51.91 Hz	103.83 Hz	207.65 Hz	415.30 Hz	830.61 Hz	1661.22 Hz	3322.44 Hz	6644.88 Hz
A	27.50 Hz (A piano's lowest note)	55.00 Hz	110.00 Hz	220.00 Hz	440.00 Hz	880.00 Hz	1760.00 Hz	3520.00 Hz	7040.00 Hz
A#/B♭	29.14 Hz	58.27 Hz	116.54 Hz	233.08 Hz	466.16 Hz	932.33 Hz	1864.66 Hz	3729.31 Hz	7458.62 Hz
B	30.87 Hz (A 5 string bass's lowest note)	61.74 Hz	123.47 Hz	246.94 Hz	493.88 Hz	987.77 Hz	1975.53 Hz	3951.07 Hz	7902.13 Hz

¿Qué pasa si quiero hacer una melodía?

Ahoravoy a hacer mi canción.

Primero defino mis notas

- NOTE_C5 es la octava 5 de Do (C)
- NOTE_E5 es la octava 5 de Mi (E)

Y luego defino un vector con mi melodía (DO DO MI)

Solo necesito definir cada nota una vez, pero puedo ponerlas varias veces en mi melodía.

Recordad las comas en el vector!!

```
#define NOTE_C5  523
#define NOTE_E5  659

int melodia[] = {
  NOTE_C5, NOTE_C5, NOTE_E5
};
```

Music Note To Frequency Chart

NOTE	OCTAVE 0	OCTAVE 1	OCTAVE 2	OCTAVE 3	OCTAVE 4	OCTAVE 5	O
C	16.35 Hz	32.70 Hz	65.41 Hz	130.81 Hz	261.63 Hz (A piano middle C)	523.25 Hz	10
C#/D♭	17.32 Hz	34.65 Hz	69.30 Hz	138.59 Hz	277.18 Hz	554.37 Hz	11
D	18.35 Hz	36.71 Hz	73.42 Hz	146.83 Hz	293.66 Hz	587.33 Hz	11
D#/E♭	19.45 Hz	38.89 Hz	77.78 Hz	155.56 Hz	311.13 Hz	622.25 Hz	12
E	20.60 Hz	41.20 Hz (A bass's lowest note)	82.41 Hz (A guitar's lowest note)	164.81 Hz	329.63 Hz	659.25 Hz	13
F	21.83 Hz	43.65 Hz	87.31 Hz	174.61 Hz	349.23 Hz	698.46 Hz	13

¿Qué pasa si quiero hacer una melodía?

Y nos faltan las duraciones, quiero que unas notas duren más que otras.

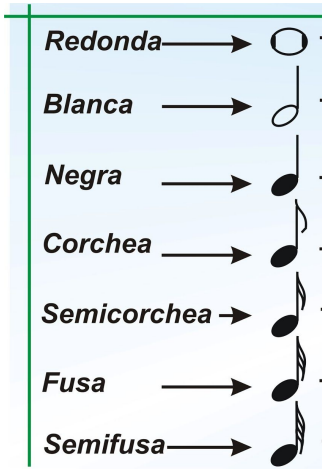
Vamos a definir 3 duraciones.

- La primera va a ser una corchea (8),
- la segunda una negra (4)
- y la tercera una blanca (2).

Estos números se sacan de cuántas veces dividimos un segundo (cuanto más grande, más dividimos y menos dura)

Yo voy a hacer corchea, negra, negra

```
/*
 8 corta
 4 normal
 2 larga
*/
int duraciones[] = {
  8, 4, 4
};
```



¿Qué pasa si quiero hacer una melodía?

En el setup vamos a definir nuestro buzzer como output (puesto que le mandamos información para que suene)

```
void setup()
{
  pinMode(BUZZER, OUTPUT);
}
```

¿Qué pasa si quiero hacer una melodía?

Ahora vamos a poner nuestro código.

Primero vamos a ver cuántas notas tenemos viendo el tamaño del vector de duraciones

Luego vamos a ir nota por nota viendo su duración y su melodía

Por último paramos después de cada nota para pasar a la siguiente

```
void loop()
{
  int tam = sizeof(duraciones) / sizeof(int);

  for (int nota = 0; nota < tam; nota++) {
    // Para calcular la duración de la nota, tomamos un segundo
    //dividido por el tipo de nota (corta larga etc)
    //ej. un cuarto = 1000 / 4, un octavo = 1000/8, etc.
    int duracion = 1000 / duraciones[nota];
    tone(BUZZER, melodia[nota], duracion);

    //Para distinguir las notas entre sí, tenemos que poner un
    //tiempo entre ellas
    //la duración de la nota + 30% funciona bien:
    int pausa = duracion * 1.30;
    delay(pausa);

    //stop the tone playing:
    noTone(BUZZER);
  }
}
```

Fase EXTRA 3.1 - Patrisio ladrador

Reto Extra 3.1 - Los perros no solo se mueven… **¡también hacen ruido!** En este reto vamos a utilizar un buzzer para que Patrisio pueda emitir sonidos y comportarse como un auténtico perro robot. ¿Seríais capaces de hacer que Patrisio ladre usando el buzzer?

Pistas:

- hacer un pitido corto
- combinar varios sonidos
- crear un “guau guau” robótico

Extra extra:
¿Podríais:

- hacer que ladre cuando detecte un obstáculo?
- hacer distintos sonidos para cada acción?
- o crear vuestra propia melodía?

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

CONTROL IR

- El control IR permite enviar órdenes inalámbricas usando luz infrarroja, una luz que no podemos ver con nuestros ojos.
- Los mandos a distancia utilizan esta tecnología para comunicarse con otros dispositivos, como:
  - 📺 Televisores
  - 🤖 Robots
  - 💡 Sistemas domóticos
  - 🎮 Proyectos con Arduino

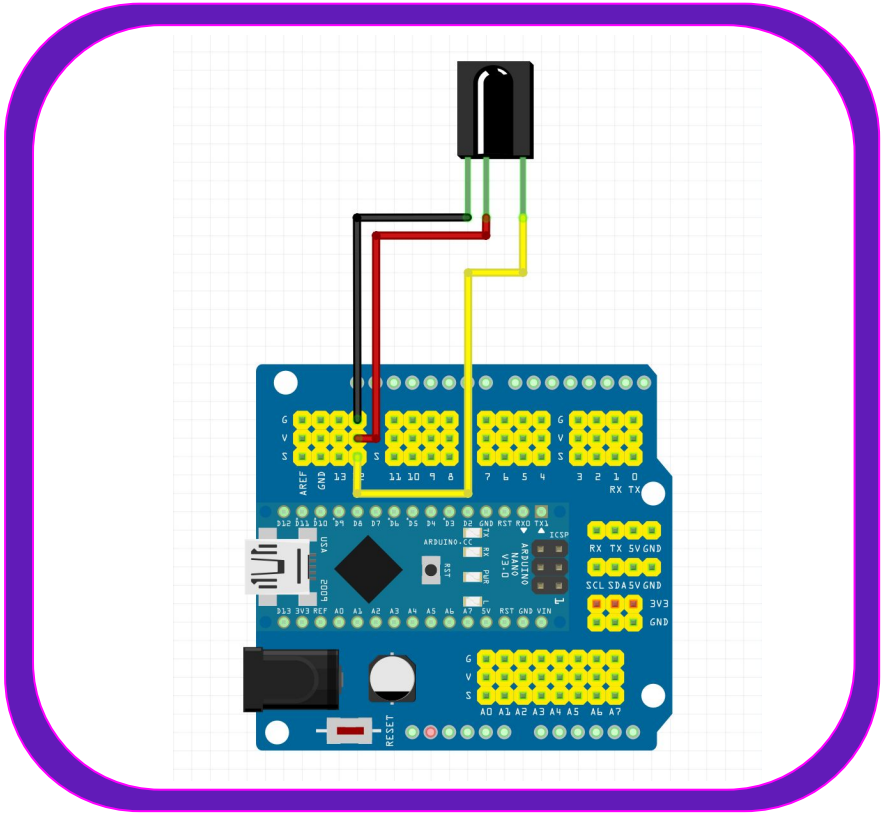
Estos son el mando y el sensor infrarrojo que usaremos para controlar a Patrisio

CONTROL IR

En esta pantalla **mostraremos todo lo que queramos e indicaremos que esta haciendo Patricio.**

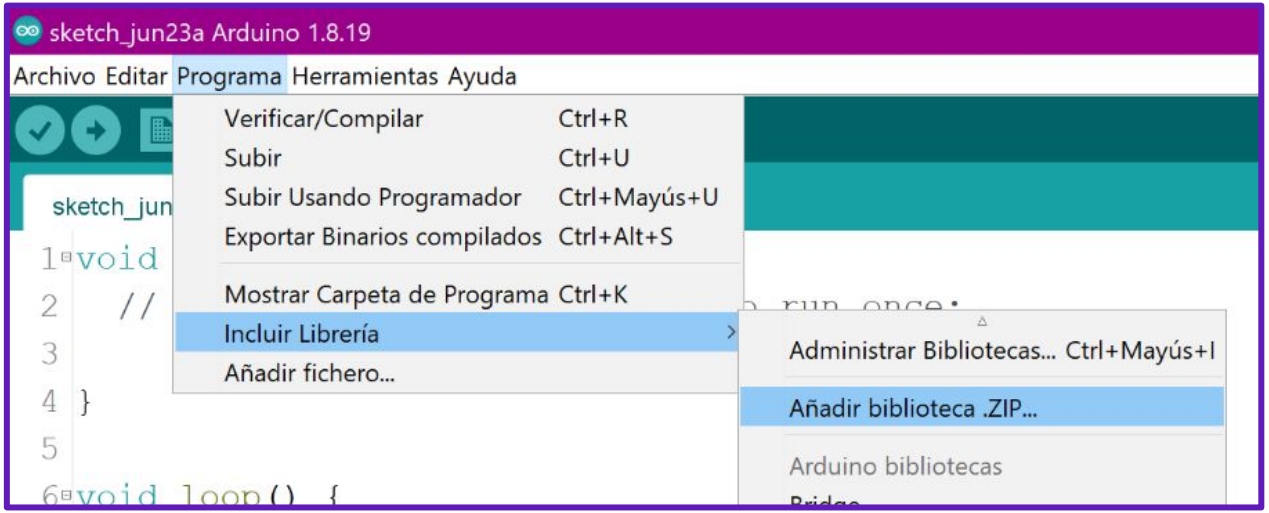
Recuerda los pines son:

GND-> GND
VCC-> 5V
DATA -> 12



CONTROL IR

1. Incluir en el IDE la librería
   https://github.com/Arduino-IRremote/Arduino-IRremote



CONTROL IR

- ¿Qué es un código IR?
- Cada vez que pulsamos un botón del mando, se envía un número único.
- Ese número es el código IR del botón.
- Arduino lo recibe y lo usa para saber qué acción debe realizar.
- A la derecha puedes ver un ejemplo de los códigos de mi mando.
- **¡CADA MANDO ES UNICO ASI QUE DEBEREIS DE DECODIFICAR VUESTRO MANDO!**

Botón	Código	Código hexadecimal
0	22	0x16
1	12	0x0C
2	24	0x18
3	94	0x5E

DECODIFICANDO NUESTRO MANDO

- Crear un nuevo proyecto de arduino y copia este código.
- **Haced una tabla con los botones que queréis utilizar(y algunos más)** y sus códigos, como la que os mostré en la diapositiva anterior.
- Estos códigos guardadlos bien, nos servirán más adelante.

```
#include <IRremote.h>

const int pinIR = 12;

void setup() {
  Serial.begin(9600);

  IrReceiver.begin(pinIR, ENABLE_LED_FEEDBACK);

  Serial.println("Lector IR iniciado");
  Serial.println("Pulsa un boton del mando...");
}

void loop() {
  if (IrReceiver.decode()) {

    Serial.print("Codigo recibido: ");
    Serial.println(IrReceiver.decodedIRData.command);

    Serial.print("Codigo en HEX: 0x");
    Serial.println(IrReceiver.decodedIRData.command, HEX);

    Serial.println("--------------------");

    IrReceiver.resume();
  }
}
```

Fase EXTRA 3.2 - Controlando a Patrisio

Reto Extra 3.2 - Hasta ahora Patrisio actuaba automáticamente… **¡pero ahora vamos a poder controlarlo a distancia usando un mando IR!** ¿Seríais capaces de hacer que Patrisio responda a distintos botones del mando?

Pistas:

- Primero debéis descubrir los códigos IR de vuestro mando
- Usad if para comprobar qué botón se ha pulsado
- Llamad a las funciones que ya habéis creado anteriormente

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

Extra extra:
¿Podríais:

- mover la cola al pulsar un botón?
- mostrar mensajes en la LCD?
- o combinar varias acciones a la vez?