8 Programación con código

- Ojo: Tres cosas
- Entorno de programación de Arduino
- Semáforo
- Semáforo sonido
- Pulsador y leds
- Señales PWM
- Intensidad del led verde según joystick
- Intensidad del led verde según la luz en LDR
- Pitido
- Servomotores
- Ángulo del servo según Joystick
- ¿Y Makey Makey? Estoy harto que me roben las naranjas

Ojo: Tres cosas

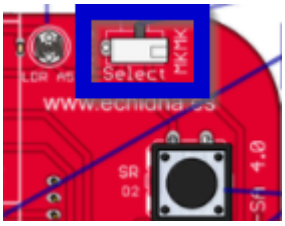
Te cargas el firmware

Si te pasas a la programación por código **te cargas el firmware** pues grabas tu programar en el Arduino del Echidna.

- Si quieres volver a programar con Echidna Scratch tienes que cargar el FIRMATA ver Por si te pasa, PROBLEMA: EchidnaScratch no detecta Echidna: Instalar Firmata
- Si quieres volver a programar con mBlock tienes que instalar el firmware de mBlock ver Por si te pasa, PROBLEMA: mBlock no detecta Echidna: Instalar Firmware mBlock

Ponlo en modo sensor

Nota: Acuérdate de poner la Echidna en modo Sensor, es decir Echidna no trabaja en modo MkyMky



Tienes que saber algo de código

Este curso no pretende ser una formación en código Arduino IDE, sino que tienes que saber ya los conceptos básicos. Sólo pretende que sepas como utilizar tus conocimientos de código en el Echidna

Entorno de programación de Arduino

Necesitarás el **entorno de desarrollo Arduino IDE** (IDE, Integrated development environment) (aquí https://www.arduino.cc/en/Main/Software para descargártelo)

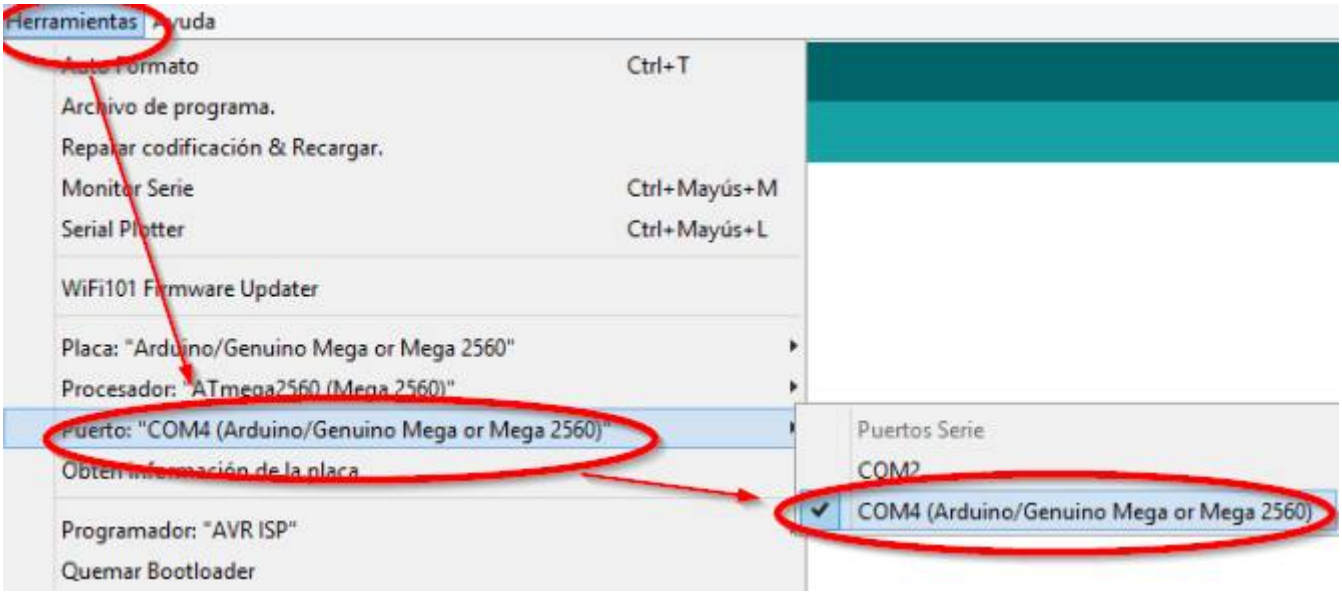
> OJO, existe **la versión online** del editor https://create.arduino.cc/editor.
> Es una buena solución si trabajas en varios equipos y quieres que tus proyectos estén disponibles en cualquier equipo.
> **ATENCION** para usar la versión online, tienes que instalar en tu ordenador el software **AGENT**
> https://create.arduino.cc/getting-started/plugin/welcome

> En Linux puede salir este mensaje "can't open device "/dev/ttyUSB0": Permission denied" donde 0 puede ser otro número, la solución aquí

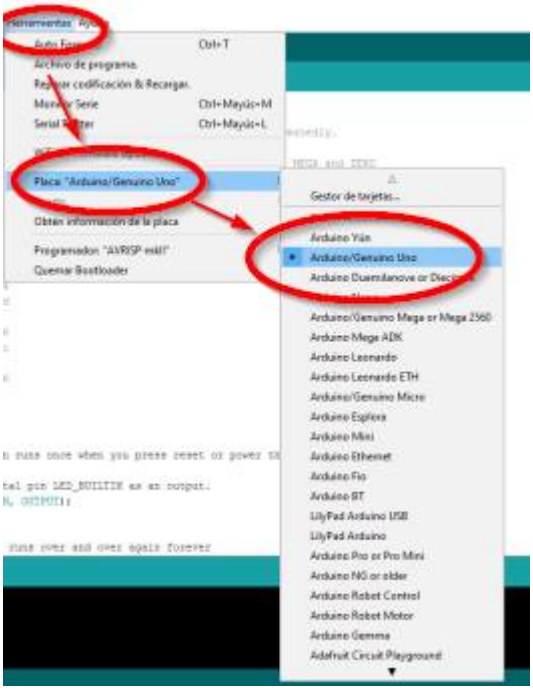
Está constituido por un **editor de texto** para escribir el código, un **área de mensajes**, una barra de herramientas con botones para las funciones comunes, y una serie de menús.

Arduino utiliza para escribir el código fuente o programa de aplicación lo que denomina "sketch" (programa). Estos programas son escritos en el editor de texto. Existe la posibilidad de cortar/pegar y buscar/remplazar texto.

Permite la conexión, por USB, con el hardware de Arduino para cargar los programas y comunicarse con ellos.



Y permite varias placas, tenemos que elegir la nuestra, en el KIT de CATEDU es Arduino UNO pero si tienes otro modelo este curso seguro que puede ser válido:

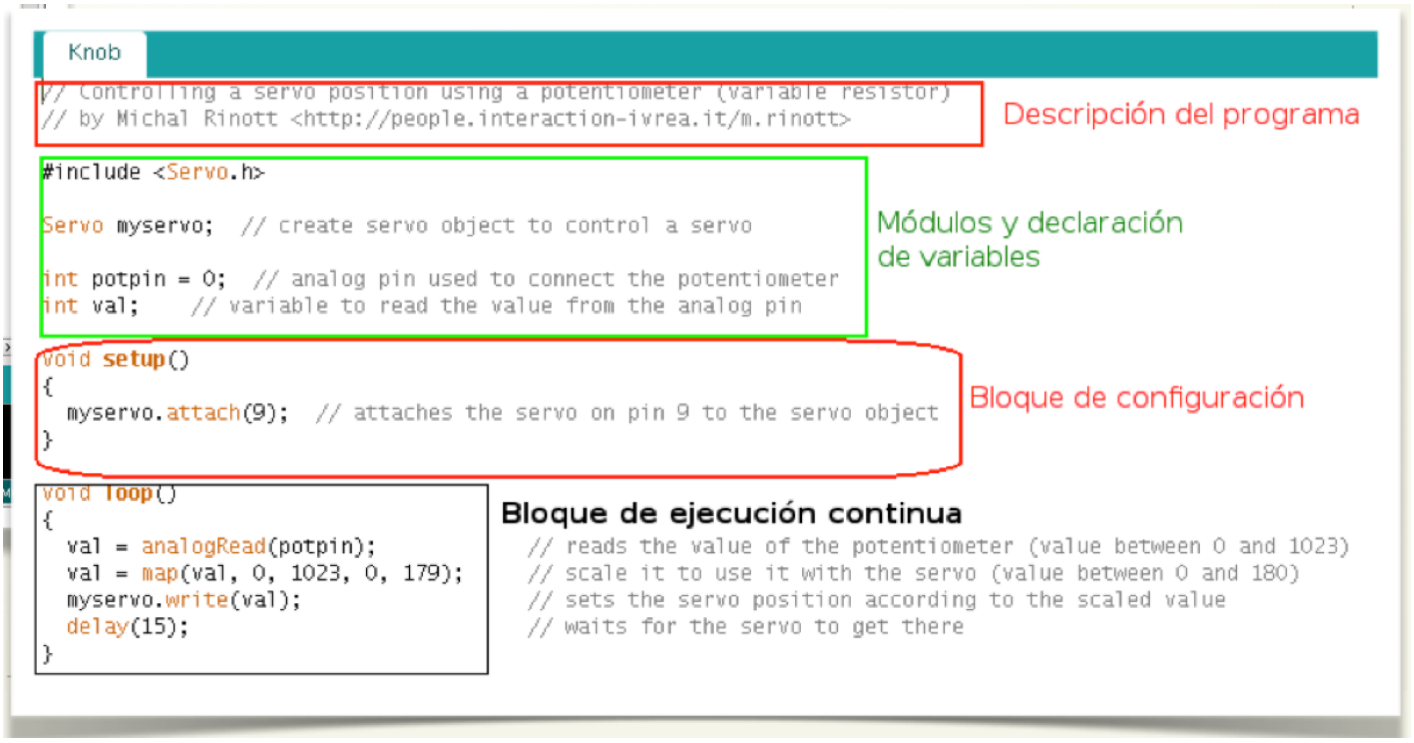


En el área de mensajes se muestra información mientras se cargan los programas y también muestra errores.

Lo importante es cuando pinchemos en la flecha de subir nuestro programa, no salga ningún error, sino simplemente "Subido".

¿Cómo se programa Arduino?

Las partes principales de un programa hecho en Arduino son: Bloque de inclusión de módulos y declaración de variables, bloque de configuración **void setup()** donde se indica el modo de funcionamiento de los pines (entrada y salida), comunicación serie, etc... y bloque de ejecución continua **void loop()**, en este bloque se incluyen las acciones que queremos que realice el programa. Se ejecutará línea a línea de forma secuencial y continua. Cuando llegue a la última instrucción incluída en la función **loop()** volverá a ejecutar la primera y continuará en un bucle infinito.



¿Arduino tiene que estar continuamente conectada a un ordenador?

Sólo es necesario que esté conectado al ordenador mediante el USB para cargar los programas o para visualizar en tiempo de ejecución datos del programa mediante **la consola serie**. El ordenador proporciona la energía eléctrica suficiente para que funcionen los programas, pero una vez cargado el programa en la memoria del microcontrolador de Arduino se puede desconectar del USB y alimentar a la tarjeta mediante una fuente externa mediante el jack de alimentación con un margen de (5 a 20 Voltios). El programa cargado en Arduino queda grabado permanentemente aunque cese el suministro eléctrico.

Para una mayor información y manejo de la instalación del entorno de programación, lenguaje de programación y librerías se encuentra en la página web de la comunidad Arduino:

- www.arduino.cc (portal en inglés, más actualizada).
- www.arduino.es (portal en español).

Semáforo

Tal y como hemos comentado, tienes dos formas de ejecutar el programa Arduino IDE, o descargándote el programa o online

La placa que tienes que seleccionar es **Arduino UNO** aunque si estas utilizando Echidna Black puedes seleccionar Arduino nano, no obstante no pasa nada si seleccionas Arduino Uno

> Vamos a realizar un programa que sea un simple semáforo (secuencias de luces rojo-amarillo-verde en bucle cada 2seg) y además que muestre por el puerto serie (para practicar esta opción) el estado del semáforo.

El programa es sencillo, lo tienes desarrollado aquí

https://app.arduino.cc/sketches/6705568b-32f2-43ca-a7f8-25d0fcd72bf8?view-mode=preview

https://app.arduino.cc/sketches/6705568b-32f2-43ca-a7f8-25d0fcd72bf8?view-mode=preview?embed

Y puedes ver que además de encender las luces, por el **puerto serie** salen los mensajes correspondientes

https://www.youtube.com/embed/UfgTnBMUfOI

Semáforo sonido

Vamos a practicar con el semáforo y con la lectura de una variable analógica como es el micrófono conectado a A7 cuyos valores van de 0-1023.

> Realizar un programa que :
> - si el valor del sonido es menor de 2 ninguna luz esta encendida
> - si es entre 2-200 solo el verde
> - si es entre 200-400 entonces verde y amarillo
> - si es más de 400 entonces todos

Solución aquí https://app.arduino.cc/sketches/8ceb4e52-0447-4968-b2bb-8720de847248?view-mode=preview

https://app.arduino.cc/sketches/8ceb4e52-0447-4968-b2bb-8720de847248?view-mode=preview?embed

https://www.youtube.com/embed/bi17ruyNxSg

> **¿Te atreves a ...?**
> Realizar un programa que haga lo mismo pero con la temperatura (los valores límites de encendido y apagado de las luces dependen de la temperatura de trabajo)

Pulsador y leds

Vamos ahora a realizar un código que si pulso el botón D2 entonces las tres luces del semáforo se encienden

El código lo tienes aquí

https://app.arduino.cc/sketches/16b618d5-af1f-425c-8279-7be1b98d8a0f?view-mode=preview

https://app.arduino.cc/sketches/16b618d5-af1f-425c-8279-7be1b98d8a0f?view-mode=embed

Como puedes ver **no es necesario tener el ordenador conectado** puedes conectarlo a un PowerBank pues no hay comunicación con el puerto serie y el programa **se carga** en la placa

https://www.youtube.com/embed/0S3sqvE-eWQ

Señales PWM

Arduino tiene entradas analógicas y digitales. Pero salidas **sólo digitales.**

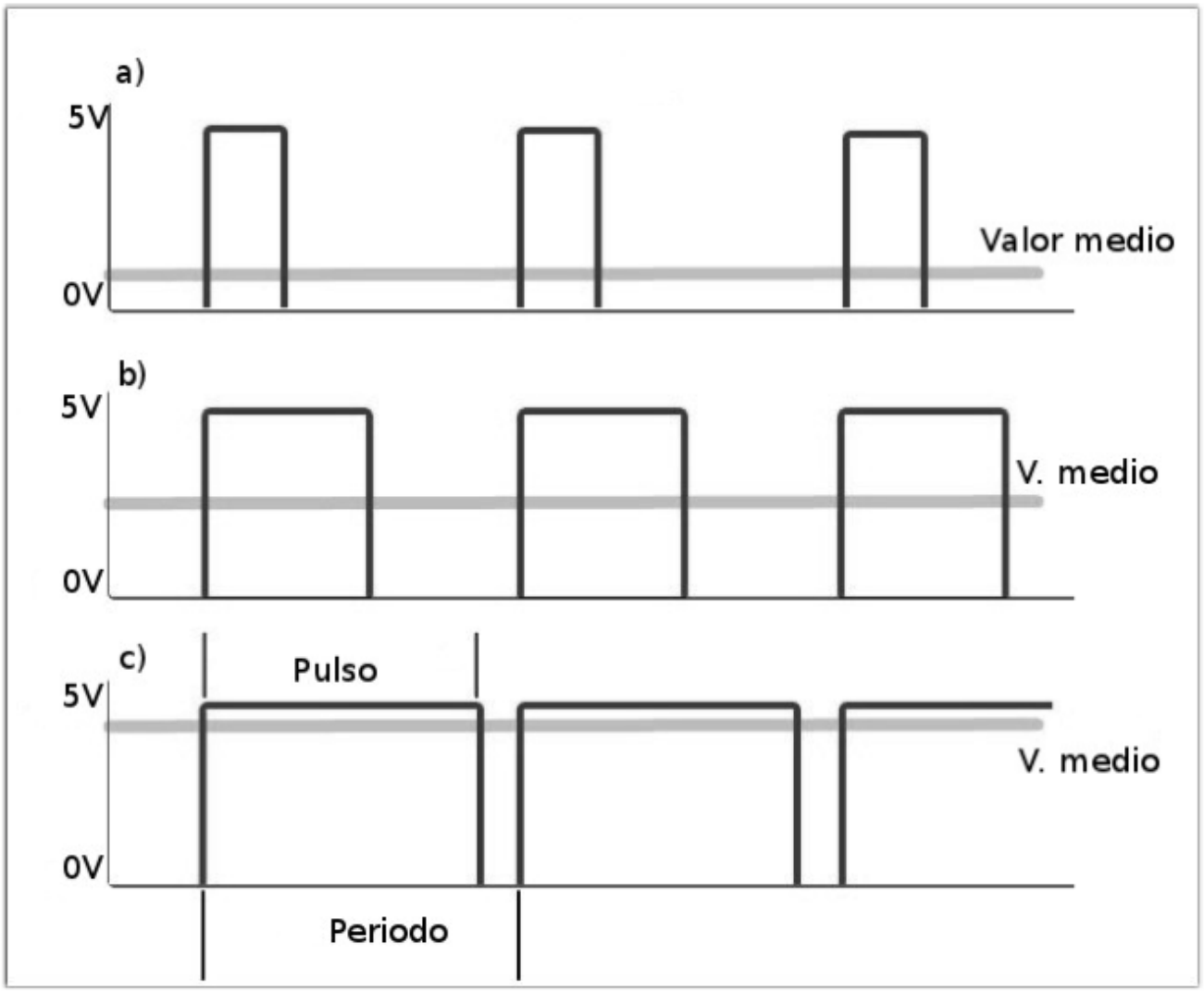
Para **simular** una salida **analógica** entre 0V y 5V se utilizan señales digitales PWM. En Arduino sólo tiene 6 salidas pseudo-analógicas. En los pines digitales 3, 5, 6, 8, 10 y 11 son PWM

¿Qué es PWM?

La señal PWM *(Pulse Width Modulation, Modulación de Ancho de Pulso*) es una señal que utiliza el microcontrolador para generar una señal continua sobre el proceso a controlar. Por ejemplo, la variación de la intensidad luminosa de un led, el control de velocidad de un motor de corriente continua,...

Para que un dispositivo digital, microcontrolador de la placa Arduino, genere una señal continua lo que hace es emitir una señal cuadrada con pulsos de frecuencia constante y tensión de 5V. A continuación, variando la duración activa del pulso (ciclo de trabajo) se obtiene a la salida una señal continua variable desde 0V a 5V.

Veamos gráficamente la señal PWM:



Los pines digitales de la placa Arduino que se utilizan como salida de señal PWM generan una señal cuadrada de frecuencia constante (490Hz), sobre esta señal periódica por programación podemos variar la duración del pulso como vemos en estos 3 casos:

- La duración del pulso es pequeña y la salida va a tener un valor medio de tensión bajo, próximo a 0V.
- La duración del pulso es casi la mitad del período de la señal, por tanto, la salida va a tener un valor medio de tensión próximo a 2,5V.
- La duración del pulso se aproxima al tiempo del período y el valor medio de tensión de salida se aproxima a 5V.

Para ejecutar una señal PWM, es simplemente **analogWrite(analogOutPin, outputValor);** donde

- analogOutPin es el número del Pin PWM, acuérdate que sólo puede ser uno de estos 6 : **3, 5, 6, 8, 10 y 11**
- outpuValor es el valor de la señal PWM pero **ojo desde 0 a 255** es decir
  - si quieres el valor de 0V tienes que poner 0

- si quieres el valor de 5V tienes que poner 255
- si quieres poner un valor medio, haz una regla de tres, por ejemplo 2.5V tienes que poner 255/2=127 o 128 da igual

Intensidad del led verde según joystick

Vamos a utilizar la salida D11 que es PWM (led verde) para modular una señal PWM, para ello vamos a utilizar la entrada analógica del eje x del Joystick, en A0 para modularlo

Mapear un valor

Como uno lee 0 a 1024 y el otro necesita valores de 0 a 255 necesitamos un traductor o mapeo con la instrucción **map(value, fromLow, fromHigh, toLow, toHigh)** esta instrucción la colocaremos, como suele estar los intérpretes en medio, o sea, entre la instrucción analogRead y la analogWrite

Solución con map

https://app.arduino.cc/sketches/491021f3-fbd8-498f-99a1-1a5ff02a441d?view-mode=preview

https://app.arduino.cc/sketches/491021f3-fbd8-498f-99a1-1a5ff02a441d?view-mode=preview?embed

Como puedes ver, en el puerto serie, los valores del potenciómetro del Joystick ejeX van desde 0 a 1024 y gracias al mapeo la señal PWM va desde 0 a 255

https://www.youtube.com/embed/urGlfsvxi2w

¿Se puede hacer sin la instrucción map?

Sí. Para ello el valor que lee 0-1024 lo convertimos a 0-255 que necesita la señal PWM que enviamos al LED simplemente dividiéndolo entre 4. (1024/4 = 256 aproximadamente 255)

El bucle loop() quedaría :

```
void loop() {
  // lee el valor de la entrada analogica:
```

```
  potValor = analogRead(analogInPin);
  // mapea el rango para la señal de salida PWM:
  outputValor = potValor/4;
  // asigna el valor cambiado a pin 3 PWM:
  analogWrite(analogOutPin, outputValor);

  // escribe el resultado en el monitor serie:
  Serial.print("Potenciometro = " );
  Serial.print(potValor);
  Serial.print("\t PWM = ");
  Serial.println(outputValor);

  // espera 1 segundo cada bucle para una visualizacion aceptable
  // conviene tener un valor aunque sea pequeño (10ms)
  // por el proceso de conversion de A/D
  delay(10);
}
```

Intensidad del led verde según la luz en LDR

Ahora para practicar más los conceptos anteriores

> Vamos a modificar la luz del led verde según la luz recibida en el LDR **pero al revés** cuanto más luz reciba el LDR más se apaga el led verde y al revés cuanto menos luz reciba el led LDR más brilla el led verde

Aquí vamos a tomar como valores mínimos y máximos del LDR los valores 250 de mínimo y 1.000 de máximo. Esto lo puedes comprobar en los valores del puerto serie que se visualizan en el siguiente programa. El por qué más adelante.

Como va al revés, la instrucción map será así **luz = map(analogRead(ldr), 250, 1024, 255, 0);**

https://app.arduino.cc/sketches/adb0c4b8-c7b5-478f-b902-392674372159?view-mode=preview

https://app.arduino.cc/sketches/adb0c4b8-c7b5-478f-b902-392674372159?view-mode=preview?embed

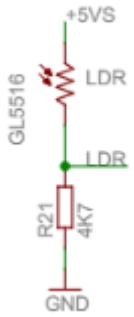
Como puedes ver el brillo de la luz verde va al revés de la luz recibida en el LDR

https://www.youtube.com/embed/0CctOVw-Uw4

¿Por qué el LDR va de 250 a 1.024 en vez de 0 a 1.024?

Esto es debido al que el LDR no está directamente conectado a masa, sino a través de un divisor de tensión con una resistencia de 4.7kOhm que se queda algo de tensión. Lo puedes comprobar en los planos aquí

https://github.com/EchidnaShield/Recursos/blob/master/electronica/Black/EchidnaBlack_0_ESQ.pdf


+5VS
GL5516
LDR
LDR
R21
4K7
GND

Pitido

Vamos a experimentar con el buzzer, con un simple enunciado

> Realizar un programa que suene una "alarma" es decir, que haga un pitido intermitente de 1 segundo

La solución la tienes aquí: https://app.arduino.cc/sketches/4c7a18bb-5012-499e-a353-6916182e728d?view-mode=preview

https://app.arduino.cc/sketches/4c7a18bb-5012-499e-a353-6916182e728d?view-mode=preview?embed

> Acuérdate de subir el volumen, en el potenciómetro !!

https://www.youtube.com/embed/cUUkWGCcz3Q

> **¿Te atreves,,,?**
> A acompañar el pitido con intermitencias de luces, por ejemplo el RGB que pase de luz máxima (255,255,255) a apagado al compás del pitido, para dar el efecto real de alarma
>
> **Pista**: La tienes en ¿Y Makey Makey? Estoy harto que me roben las naranjas

Servomotores

Una de las aplicaciones más utilizadas de los sistemas de control por ordenador y en la robótica están asociados con los motores, que permiten accionar o mover otros componentes, como puertas, barreras, válvulas, ruedas, etc. Uno de los tipos que vamos a ver en este capítulo son lo servos, hay de dos tipos:

- El **servomotor** o **servos convencionales** que posee la capacidad de posicionar su eje en un ángulo determinado entre 0 y 180 grados en función de una determinada señal.
- **Servo de rotación continua** Son servos por fuera igual que los anteriores, pero pueden girar 360º y se controlan por tiempo

> Por defecto cuando se dice **servo**, es un **servomotor o servo convenciona**

Servos de rotación continua

Para controlar un servo de rotación continua, las instrucciones a realizar son :

- Incluyes la librería de servos **#include <Servo.h>**
- Declaras una variable servo **Servo myservo;** //puedes poner el nombre que quieras p.e. miservo
- En *setup()* tienes que decir a qué pin está conectado **myservo.attach(9);** //por ejemplo pin 9
- Y en *loop()*
  - **myservo.write(90)**; //significa servo parado
  - **myservo.write(180);** //significa servo funcionando al 100% en el sentido de las agujas del reloj
  - **myservo.write(0);** //significa servo funcionando al 100% en el sentido contrario de las agujas del reloj

Mira el vídeo, esta realizado con otra shield ECHIDNA y con bloques mBlock (curso Echidna https://libros.catedu.es/books/echidna/) fíjate como:

- Los extremos 0º y 180º es a máxima velocidad, pero un sentido u otro.
- 90º es parado.
- Un valor intermedio es menos velocidad (se ve el ejemplo 80º y 100º) -
- Si tiene deriva, (cosa frecuente) tienen un potenciómetro para ajustar.

https://www.youtube.com/embed/Z-5SerXmRY0

> Si quieres saber más sobre servomotores te recomendamos estas paginas del Zaragozano Luis LLamas: Servomotores convencionales y Servomotores de rotación continua

Servomotores o servos convencionales

Los servos son un tipo especial de motor en el que se añade una circuito lógico electrónico que permite un control mucho más preciso que a un motor normal de corriente continua. Esto les permite posicionar el eje en un ángulo determinado.

El hardware interno se compone de un potenciómetro y un circuito integrado que controlan en todo momento los grados que gira el motor. De este modo, en nuestro caso, desde Arduino, usando las salidas digitales PWM podremos controlar fácilmente un servo. Lo ideal es conectarlo a 6V pero trabajan bien en los 5V del Arduino.

Hay muchos modelos, en robótica educativa cuestan entre 1-5€, el más común es el SG90, muy barato, pero tiene muy poca fuerza, el MG90S tiene algo más, si queremos algo más, ya tiene que ser el MG996R pero ya este modelo **NO se puede conectar directamente al Arduino,** el pico de energía que necesita, provoca el reinicio de la placa. Incluso varios pequeños SG90.

Las instrucciones son las mismas que los servos de rotación continua, pero los valores que se proporcionan son los grados que se desean.

- Incluyes la librería de servos **#include <Servo.h>**
- Declaras una variable servo **Servo myservo;** //puedes poner el nombre que quieras p.e. miservo
- En *setup*() tienes que decir a qué pin está conectado **myservo.attach(9);** //por ejemplo pin 9

- Y en *loop*()
  - **myservo.write(90)**; //Posición 90º (posición por defecto)
  - **myservo.write(180);** //Posición 180º
  - **myservo.write(0);** // Posición 0º

La instrucción *myservo.write(angulo)* envía por el pin digital declarado en *myservo.attach()* pulsos cuadrados de 50Hz y de anchura el estado alto proporcional al ángulo que se desea.

- Un pulso de 0.5-1ms es 0º
- Un pulso de 1.5 ms es 90º
- Un pulso de 2-2.5ms es 180º

Si quieres saber más, te recomendamos https://www.luisllamas.es/controlar-un-servo-con-arduino/

Ángulo del servo según Joystick

Para practicar un poco los servos, vamos a realizar el siguiente enunciado

> Mover el servomotor un ángulo entre 0º y 180º según los valores del Joystick en el ejeY, 0º abajo del todo, 180º arriba del todo

Aquí hay que tener claro que los valores de entrada es el Joystick eje Y por lo tanto es la señal analógica A1 y sus valores van de 0 a 1023, y al servo hay que indicarle los valores en grados de 0º a 180º luego la función de mapeo es: ***val = map(val, 0, 1023, 0, 180);*** donde val va a ser una variable que ha guardado el valor del Joystick (0-1023) y que con la instrucción map lo ha traducido a 0-180.

El programa es https://app.arduino.cc/sketches/29ac0e0b-8da8-482b-bf62-6f90a58f2459?view-mode=preview

https://app.arduino.cc/sketches/29ac0e0b-8da8-482b-bf62-6f90a58f2459?view-mode=preview?embed

https://www.youtube.com/embed/A-wCDePVppI

Si no tienes servo, puedes simularlo. En la siguiente simulación, puedes mover el potenciómetro y ver el resultado

https://www.tinkercad.com/embed/gl6syqapJHe?editbtn=1

> **¿Te atreves...?**
> A realizar un programa que mueva el servo **según la inclinación de la placa**

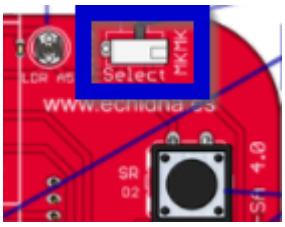
https://www.youtube.com/embed/IkzXSBXz6aw

¿Y Makey Makey? Estoy harto que me roben las naranjas

¿Podemos hacer proyectos Makey Makey? Por supuesto, pero....

Ponlo en modo Makey Makey

Nota: Acuérdate de poner la Echidna en modo Makey Makey, y luego acuérdate de cambiarlo si vuelves a utilizar los sensores



Estoy harto que me roben las naranjas

> Vamos a hacer un programa que suene una alarma (tanto sonora como luminosa en RGB) si tocamos una naranja, conectada en el terminal Makey Makey A0

La solución la tienes aquí : https://app.arduino.cc/sketches/b043d527-bf50-476a-ac61-c537d820f261?view-mode=preview

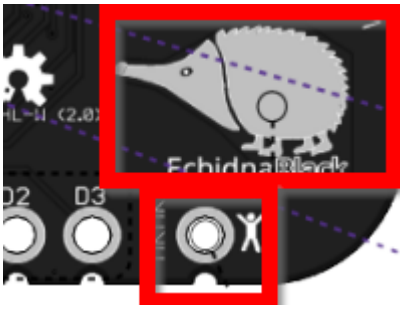
https://app.arduino.cc/sketches/b043d527-bf50-476a-ac61-c537d820f261?view-mode=preview?embed

https://www.youtube.com/embed/DoOtR7O7Pug

.

¿Por qué no va 100% bien?

Porque **no estamos tocando la masa del Echidna**. Si con la otra mano estuviésemos tocando con un cable la masa del Echidna, entonces iría perfecto:



P: ¿Por qué funciona? Funciona en el primer toque y luego no ¿Por qué?
R: La primera vez que tocamos la naranja se descarga nuestra electricidad estática y lo lee el terminal A0 del Echina, por lo que sube su valor, luego ya baja.

Esto lo puedes comprobar leyendo los valores del puerto serie, si te fijas en el siguiente vídeo, al tocar con el dedo, sube a 195, 158 pero luego cuando se ha descargado la electricidad, ya baja él sólo.

Si con otra mano tocase la masa, no pasaría eso.

https://www.youtube.com/embed/EDyZYWEptk0

> **¿Te atreverías a hacer un piano con bananas ?**