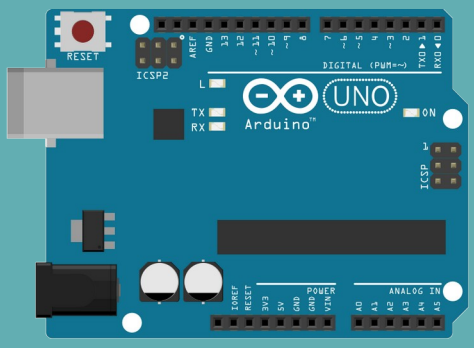


Introducción a Arduino con steamakers blocks

Lecciones con videotutoriales

Prácticas

Tareas

Francisco Trigueros

creative commons
BY SA

Todo el contenido del sitio web STEAMakersBlocks.com, tanto el software como la parte artística (logotipos, iconos, etc), son propiedad intelectual de Juan José López Almendros, responsable del sitio web.

Imagen Arduino logo de Autor desconocido protegida como Marca Registrada.

Esta licencia permite a los reutilizadores distribuir, remezclar, adaptar y crear a partir del material en cualquier medio o formato, siempre que se cite al creador. La licencia permite el uso comercial. Si remezclas, adaptas o construyes a partir del material, debes licenciar el material modificado bajo idénticos términos. CC BY-SA incluye los siguientes elementos:

BY: debe darse crédito al creador.

SA: Las adaptaciones deben compartirse bajo los mismos términos.

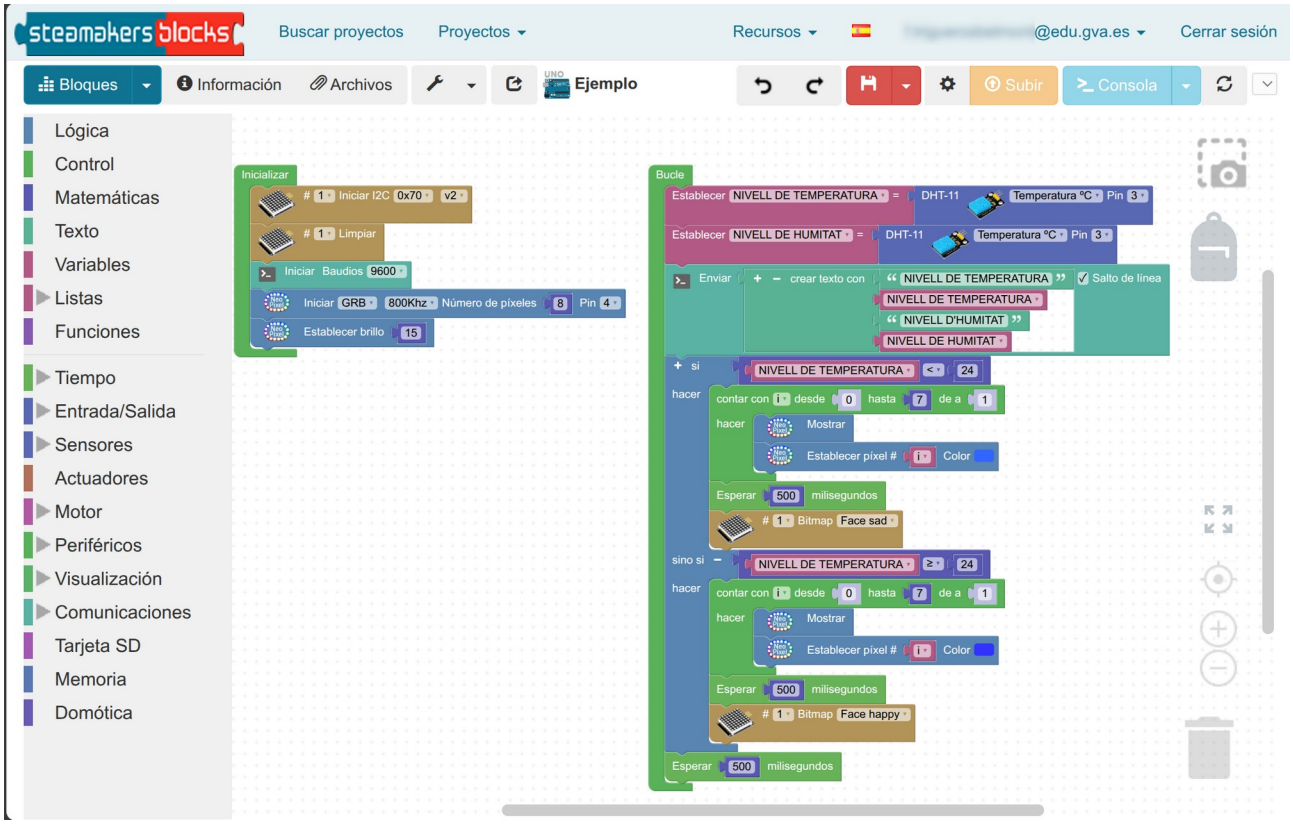
Índice de contenidos

1. Plataforma educativa STEAMakersBlocks

STEAMakersBlocks es la evolución de ArduinoBlocks, una plataforma online orientada a educación creada por el profesor Juanjo López. Con ella podemos programar por bloques placas de desarrollo, como la Arduino UNO, sin necesidad de escribir código en C++ con Arduino IDE (su entorno de programación oficial).

A diferencia de TinkerCAD, una herramienta con la que podemos realizar una simulación online de circuitos con Arduino y programarlo por bloques, **STEAMakersBlocks no es un simulador**, sino que está pensado para programar desde un primer momento el hardware real (Arduino UNO, ESP32, etc) y para poder comunicarnos con la placa mediante la consola serie.

Está diseñado para proyectos reales y dispone de bloques para programar una gran cantidad de sensores y actuadores: sensores de distancia, de humedad y temperatura, motores, servomotores, pantallas LCD y OLED, matrices y tiras de LED, módulo de MP3, comunicaciones por Bluetooth y WIFI, etc.



Plataforma STEAMakersBlocks

Al registrarnos en la plataforma, podemos aprovechar sus posibilidades:

- Guardar tus proyectos en la nube de STEAMakersBlocks.

- Añadir información al proyecto: descripción, componentes utilizados, imágenes, etc.
- Añadir archivos adjuntos relacionados con el proyecto: esquemas, fotos, archivos para impresión 3D, aplicaciones, etc.
- Compartir proyectos con el resto del mundo.
- Importar proyectos compartidos por otros usuarios.
- Valorar y comentar proyectos

Además, nos permite crear un proyecto para un grupo o clase, pudiendo invitar a los alumnos a unirse con su propia cuenta o con cuentas gestionadas por el profesor. De esta manera, el profesor puede ver el trabajo de cada estudiante, corregirlo y hacer comentarios dentro del sistema. Podemos ver información sobre la gestión de usuarios en el siguiente enlace:

- **Alumnos en ArduinoBlocks, Gestión de Usuarios y Revisión de Ejercicios**

Página de proyecto con alumnos en STEAMakersBlocks

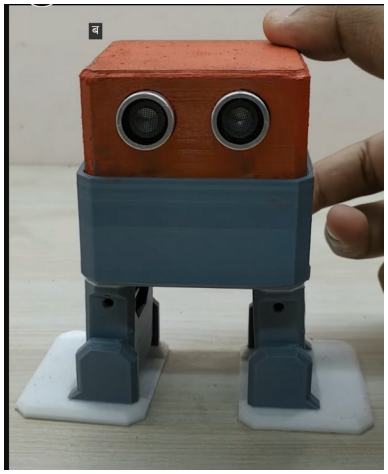
1.1. Placas de desarrollo soportadas por STEAMakersBlocks

Una **placa de desarrollo** como Arduino UNO incorpora un microcontrolador y permite leer sensores, controlar actuadores y crear prototipos electrónicos de forma sencilla.

STEAMakerBlocks nos permite crear proyectos con el siguiente hardware:

- UNO
- NANO / ATmega328
- MEGA / 2580
- Leonardo
- UNO + Imagina TIRSTEAM
- UNO + Imagina 3DBot
- Keyestudio EasyPlug
- Keyestudio KeyBot
- ESP32 micro:STEAMakers
- ESP32 STEAMakers

- ESP32 STEAMakers + Imagina TAR STEAM
- ESP32 STEAMakers + Imagina 3DBot
- Keyestudio KidsloT
- ESP32 / WROOM
- ESP8266 / NodeMCU v2
- ESP8266 / WeMos D1
- Otto DIY / Nano (clic en la imagen para verlo en movimiento):



Vídeo de Otto, robot imprimible en 3D

📝 **Una vez que conozcamos los fundamentos de un tipo de placa y su programación, migrar hacia otra nos resultará muy sencillo.**

Describimos a continuación cuatro de las placas que podemos usar en nuestros proyectos:

1.2. Arduino UNO

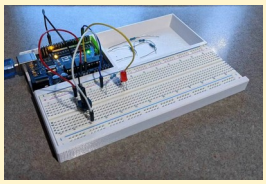
Una placa **Arduino UNO** es una placa de hardware libre. Su principal objetivo es facilitar el aprendizaje y el desarrollo de proyectos de robótica y electrónica, especialmente para estudiantes y principiantes, gracias a su facilidad de uso y a un enorme ecosistema de recursos y ejemplos.

Arduino UNO se lanzó en 2010, evolucionó y se expandió rápidamente gracias a su filosofía open-source, su facilidad de uso y una comunidad global que impulsó su adopción en educación, arte, ingeniería y el movimiento maker.

Arduino UNO. SparkFunElectronics.Lisens: CC BY 2.0

Se pueden encontrar clones de Arduino UNO R3 en tiendas españolas desde unos 6€.

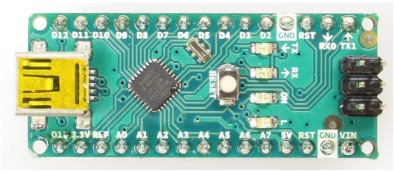
📝 **Existen modelos 3D de bases para imprimir y acoplar la placa Arduino UNO y la protoboard, muy cómodas para la realización de prácticas. Haz clic en la imagen para descargar el modelo de la foto.**



1.3. Arduino Nano

La placa **Arduino Nano** ofrece las mismas funcionalidades esenciales del Arduino Uno, pero en un formato mucho más pequeño, lo que la hace ideal para proyectos portátiles, wearables y sistemas embebidos donde la miniaturización es clave.

Se pueden encontrar clones de Arduino Nano desde unos 6€ en tienda española.

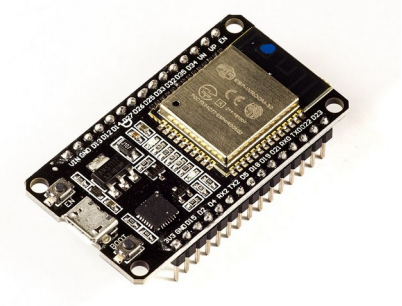


Arduino Nano. By MakeMagazinDE - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=68035748

1.4. ESP32

ESP32: desarrollada por Espressif Systems (no Arduino) y es conocida por integrar **Wi-Fi y Bluetooth**, lo que lo convierte en una **opción ideal si necesitamos proyectos de IoT, domótica y como alternativa a Arduino UNO en cualquier automatización**. Su capacidad de trabajar en modos de bajo consumo, su versatilidad en programación (Arduino IDE, MicroPython, STEAMakersBlocks) y su gran variedad de módulos disponibles lo han convertido en uno de los microcontroladores más populares entre makers y profesionales.

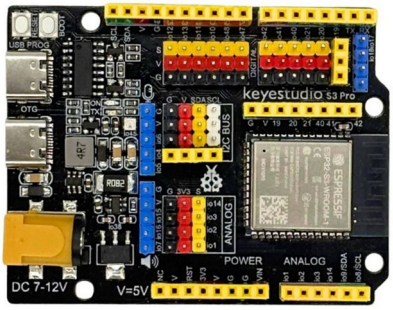
La encontramos por unos 6,80€ en tienda española.



ESP32. Creative Commons CC0 1.0 Universal Public Domain Dedication.

1.5. Keyestudio ESP32 STEAMakers AI

Si disponemos de más presupuesto, podemos optar por un kit con distintos sensores que incorporan conectores de 3 pines y la Placa ESP32 STEAMakers AI, que ofrece sobre la Arduino o ESP32 la posibilidad de realizar prácticas con



Keyestudio Placa ESP32 STEAMakers AI. Fuente: Innova Didactic

conexiones directas a la placa reduciendo el cableado y el uso de protoboard, resultando ser las prácticas más sencillas y fiables.

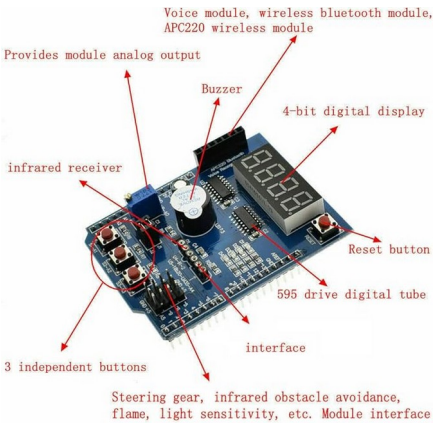
La placa cuesta unos **35,90€ en INNOVA DIDACTIC,** los sensores de 3 pines también salen más caros que los tenéricos, pero también debemos saber que esta tienda es la **colaboradora oficial de STEAMakersBlocks**, **con lo que apoyaremos al mantenimiento y desarrollo de esta plataforma**.

1.6. Tabla comparativa entre placas

Placa	Nivel educativo / uso	Programación gráfica	Conectividad	Potencia y recursos	Facilidad para principiantes
Arduino Uno R3	Desde principiante / proyectos sin conexión WIFI	STEAMakersBlocks	Bluetooth (conectando módulo)	Básica	⭐⭐⭐⭐⭐
ESP32 genérico	Desde nivel medio / IoT	STEAMakersBlocks	Wi-Fi/Bluetooth	Alta	⭐⭐⭐
ESP32 STEAMakers Ai	Desde principiante / IoT	Optimizado para STEAMakersBlocks	Wi-Fi/Bluetooth + opciones Ai	Alta con extras	⭐⭐⭐⭐

2. Placas de expansión.

Hay varios shields (placas de expansión) y kits para usar Arduino sin necesidad de cableado complejo ni conocimientos previos de electrónica.

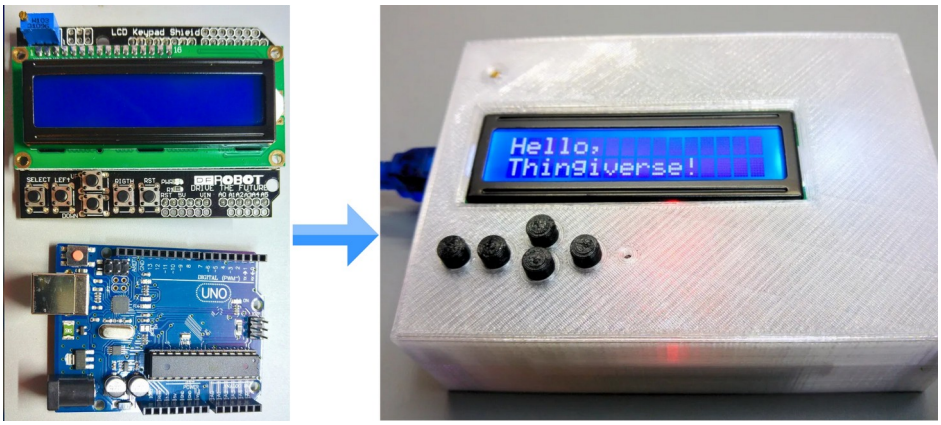


Vemos un ejemplo de placa que se puede encontrar por unos 6€ en tienda española, características:

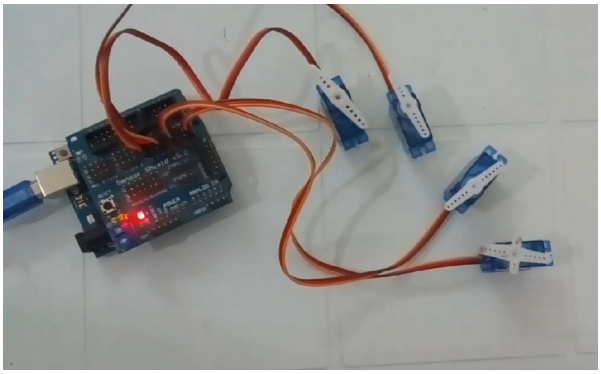
- Display de 4 dígitos de 7 segmentos controlado por un 74HC595
- 4 leds
- Zumbador
- Potenciómetro de 10K
- 3 pulsadores
- Pulsador de reset
- Interface para el controlador APC220 bluetooth o reconocimiento de voz
- Interface para sensor de infrarojos

- interface para el sensor de temperatura LM35
- Interface para motores servo

Otras placas de expansión:

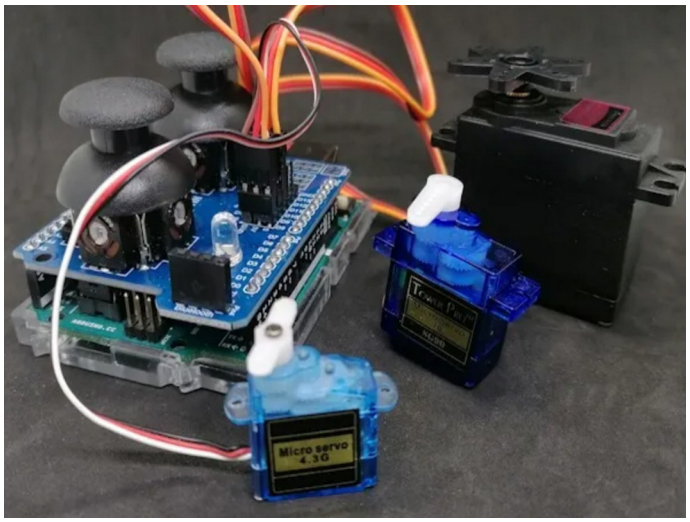


Placa de expansión con LCD y pulsadores



Placa de expansión para servomotores

Placa tipo gamepad



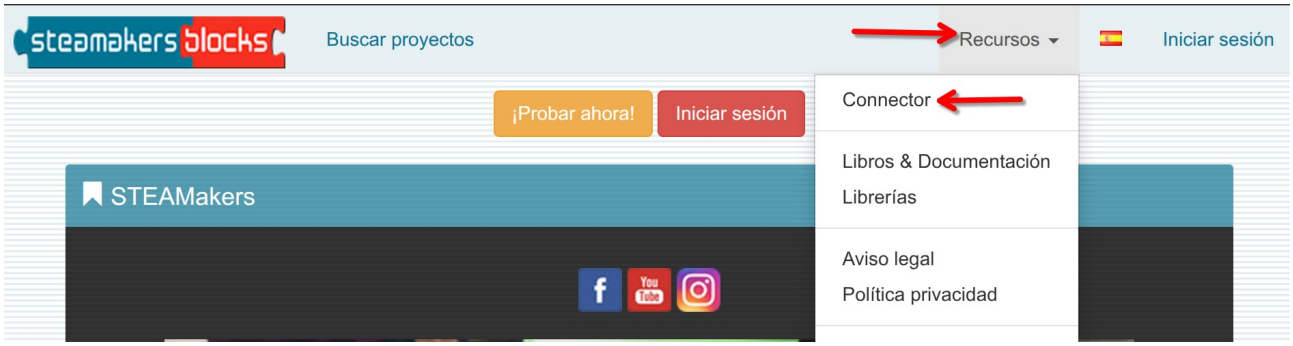
Placa con joystick, módulo bluetooth y servomotores

3. Instalación de STEAMakersBlocks Connector

Connector de STEAMakersBlocks es un pequeño programa que actúa como puente entre el entorno STEAMakersBlocks y la placa física (Arduino, ESP32, etc.).

Cuando programas en STEAMakersBlocks desde el navegador, este no puede comunicarse directamente con el puerto USB del ordenador, por lo que ***Connector* debe estar siempre abierto mientras transfieres el programa o te comuniques con la placa por USB.**

Tiene versiones para Windows, Linux, Mac y Chromebook. Está disponible en su **web oficial (Connector v6)**:



Descarga desde STEAMakersBlocks

Si tenemos problemas porque nuestro equipo no reconoce la placa, podemos ayudarnos con este vídeo:

- **Connector setup (videos)**

> 📝 **En Lliurex lo podemos instalar desde el Zero-Center, lo encontramos dentro de "Aplicaciones Tecnológicas"**

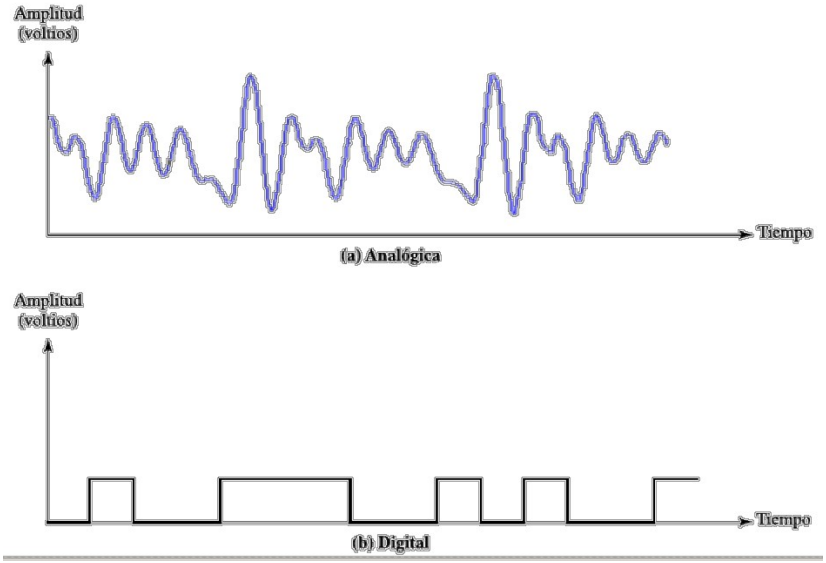
4. Señales analógicas y digitales.

> ▶️ **Vídeo 1: Arduino con STEAMakersBlocks - Primeros pasos**

En Arduino nos podemos encontrar con dos tipos de señales:

- **Señal analógica**: no tienen saltos (cambian poco a poco), tienen infinitas posibilidades de valores y son más sensibles al ruido e interferencias.

- **Señal digital**: es una señal que solo tiene dos estados, que podemos expresar de varias maneras:
  - 0 - 1
  - LOW - HIGH
  - OFF – ON
  - Desactivado - Activado

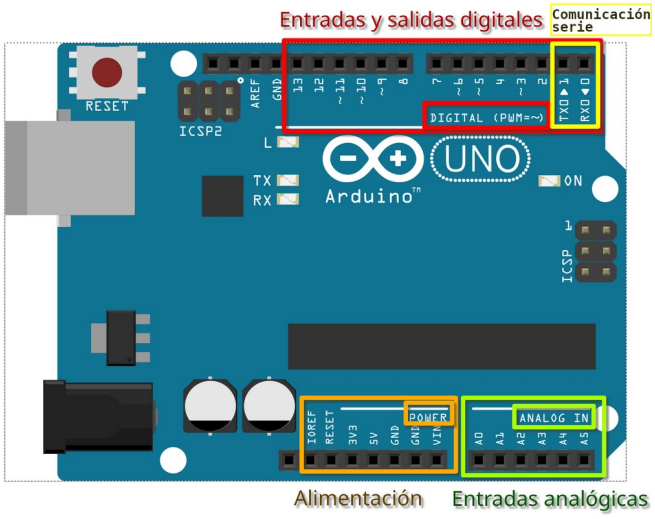


Señal analógica y digital. ***Wikimedia Commons***

5. Pines digitales en Arduino UNO.

Los pines digitales, numerados de 0 a 13 (marcados en rojo en la imagen siguiente), permiten trabajar con dos estados: ON/OFF, Activado/Desactivado, 1/0.

Los pines se pueden configurar como entrada o como salida según se necesite conectar un sensor o un actuador.



Pines en Arduino UNO

Si los pines digitales están configurados como **salida**, suministran el siguiente voltaje:

Estado digital del PIN	Tensión de salida
OFF	0 V (sin corriente)
ON	5 V

Si los pines están configurados como **entrada**, según el voltaje se comportan así:

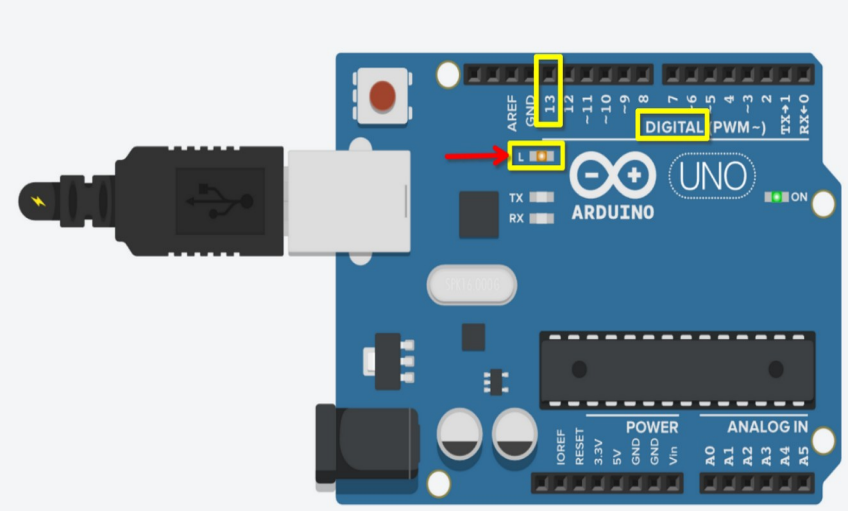
Tensión de entrada	Estado digital del PIN
≥ 3.0 V	ON / HIGH / 1
≤ 1.5 V	OFF / LOW / 0

📝 **Usaremos únicamente los pines del 2 al 13 como entrada/salida. STEAMakersBlocks no utiliza los pines 0 y 1 como pines de E/S, los reserva para la conexión serie y la programación desde el PC.**

Práctica 1. Nuestro primer programa: parpadeo del LED integrado

Vamos a comenzar a programar el comportamiento de nuestra placa. Para ello, nada mejor que nuestro primer programa clásico: hacer parpadear el LED integrado.

Todas las placas Arduino UNO incluyen un pequeño LED (marcado con una "L") conectado internamente al pin digital 13, lo podemos ver en la imagen.



LED integrado conectado al pin 13

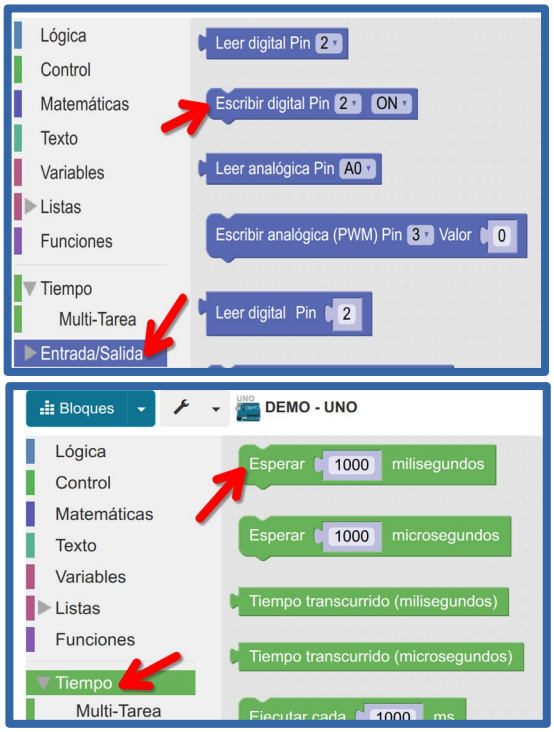
Es nuestro "¡Hola mundo!" en Arduino, y marca el comienzo de todo lo que podremos crear a partir de ahora.

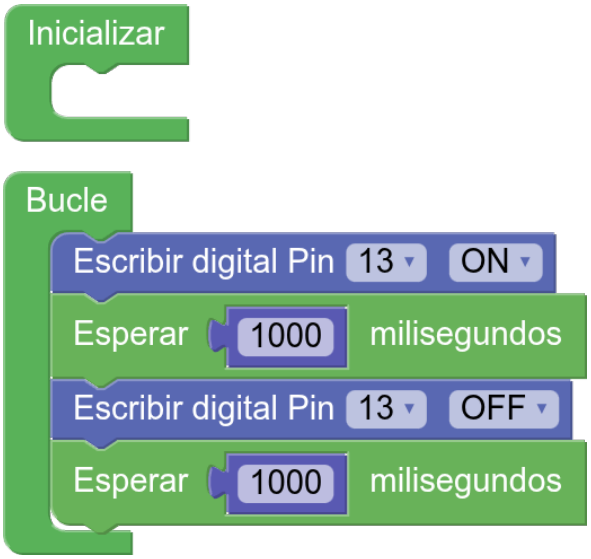
Haremos el programa siguiente:

1. Encender el LED conectado al pin 13
2. Mantenerlo encendido 1 segundo (1000 milisegundos)
3. Apagar el LED conectado al pin 13

4. Mantenerlo apagado 1 segundo (1000 milisegundos)

Con STEAMakersBlocks, el programa quedaría así:





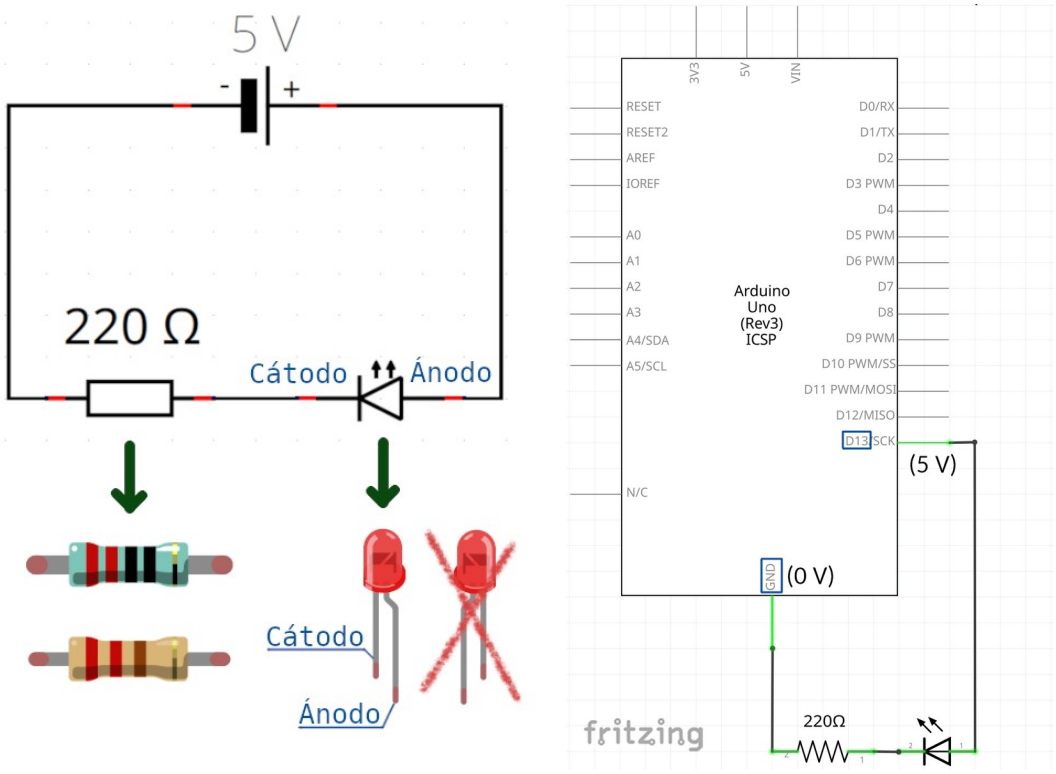
Podemos ir cambiando los tiempos de encendido y apagado, comprobando el resultado.

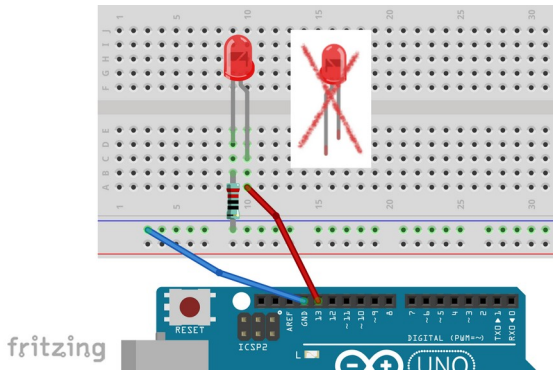
Veamos cómo sería el mismo programa con Arduino IDE:

```
void setup() {
   pinMode(13, OUTPUT);
}

void loop() {
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13, LOW);
   delay(1000);
}
```

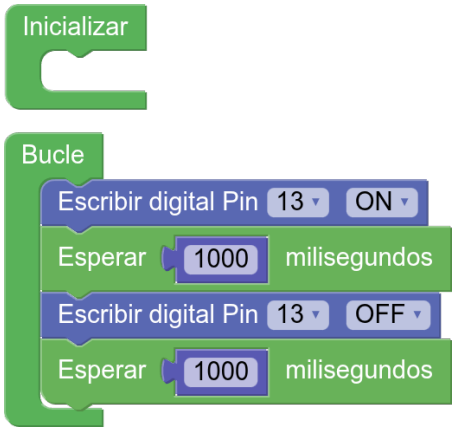
6. Protoboard y conexión de componentes

▶️ **Vídeo 2: Semáforo con Arduino y STEAMakersBlocks**



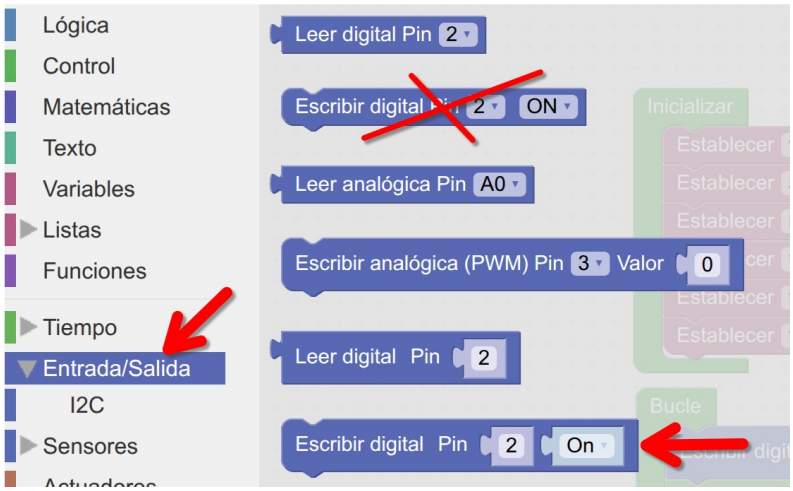


Monta el circuito de la figura anterior con la protoboard y con el mismo programa del apartado anterior (ver figura) observa cómo parpadea tanto el LED integrado en la placa y el LED conectado al pin 13.



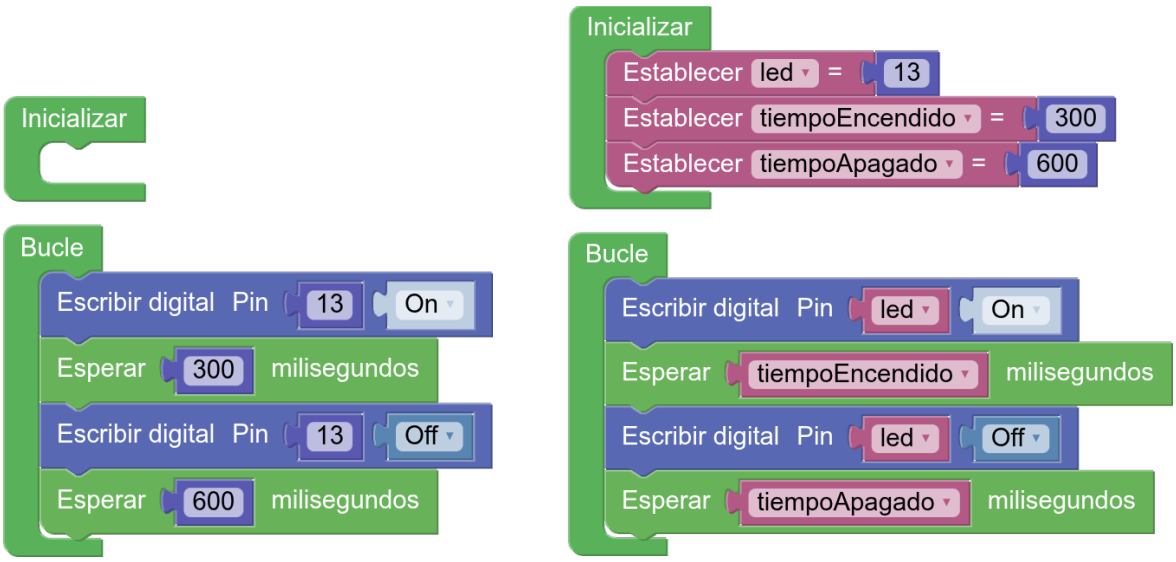
7. Variables.

Una **variable** es un espacio de memoria de la placa identificado por un nombre, que se utiliza para almacenar un valor (número, texto...). Su valor puede cambiar (variar) durante la ejecución de un programa.



Bloque para usar variables con los pines digitales, ***los valores tienen forma de puzzle***

Vemos la comparativa del mismo programa sin usar variables y después usándolas (**tendrá más sentido en programas más complejos**):



El mismo programa con Arduino IDE:

```
int led = 13 ;
int tiempoEncendido = 300;
int tiempoApagado = 600 ;

void setup() {
   pinMode(led, OUTPUT);
}

void loop() {
   digitalWrite(led, HIGH);
   delay(tiempoApagado);
   digitalWrite(led, LOW);
   delay(tiempoApagado);
}
```

Vemos algunos de los usos y ventajas de las variables:

- **Guardar información**: Una variable es como una "caja" donde guardamos datos (números, texto, etc.) para usarlos más tarde. Así no tenemos que escribir el mismo valor muchas veces.
- **Hacer programas más flexibles**: Con variables, el programa puede funcionar con diferentes datos sin cambiar el código. Por ejemplo, si cambiamos la conexión del LED a otro pin, no hay que reescribir todo.
- **Organizar mejor el código:** Usar variables hace que el programa sea más ordenado y fácil de entender, porque damos nombres a los datos en lugar de usar números o textos sueltos.

8. Relé: encendido o apagado de dispositivos de más potencia

▶️ Vídeo: ¿Qué es un relé (relay)?

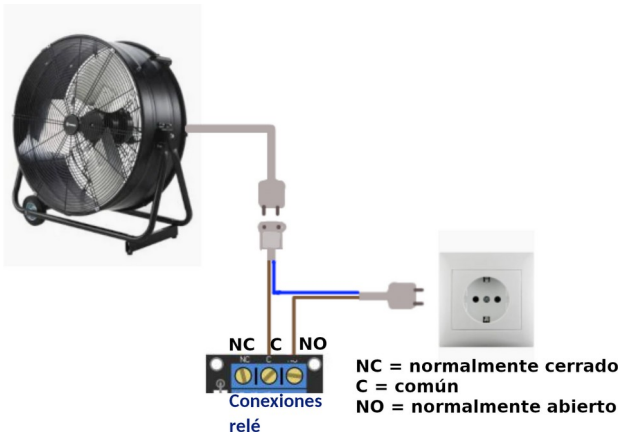
Los pines de Arduino tienen poca potencia de salida, no se recomienda pasar de 20 mA (0,1 W) para uso continuado.

Imaginemos que en vez de querer encender o apagar un LED, queremos encender o apagar una lámpara o ventilador conectados a la red eléctrica de 230V.

Para controlar estos dispositivos externos, podemos usar **un relé,** que funcionará como un interruptor controlado por la señal digital de Arduino.

El segundo circuito puede ser de corriente continua o de corriente alterna a 230V, pudiendo conectar una lámpara, ventilador, etc.

Conexiones a Arduino.



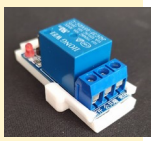
Ejemplo de conexiones a un circuito actuador

Ejemplo de programa básico

Activa y desactiva el aparato conectado al relé intermitentemente cada 3 segundos:

```
const int pinRele = 13;
int retardo = 3000;

void setup() {
pinMode(pinRele,OUTPUT);
}


void loop() {
digitalWrite(pinRele,HIGH);
delay(retardo);
digitalWrite(pinRele,LOW);
delay(retardo);
}
```

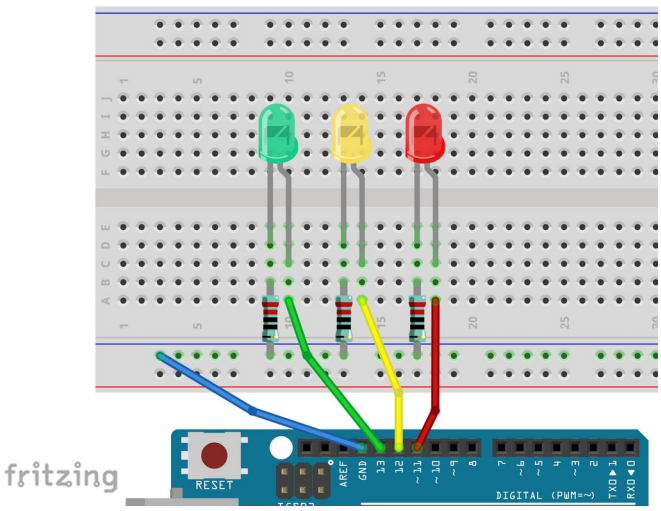
📝 **Precaución:** si se conecta a 230V, conviene evitar el peligro por contacto en la parte inferior de la placa. **Recurso: base para imprimir en 3D y acoplar el módulo con el relé**



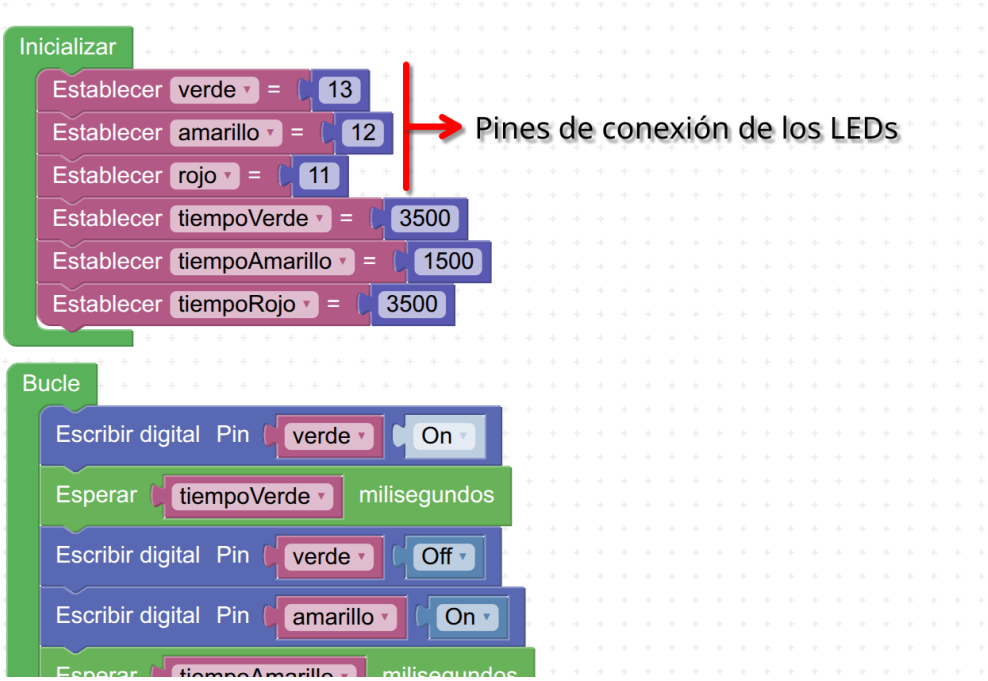
☑️ TAREA 1.1. Semáforo de coches

Necesitamos hacer una maqueta en la que haya un semáforo de coches. Vamos a realizar el montaje sobre protoboard y programarlo con STEAMakersBlocks.

Realiza el siguiente montaje:



Aquí tienes el **comienzo del programa**, **está por terminar,** puedes usarlo como guía y personalizar las variables o los tiempos:



El programa completo del semáforo con código sería:

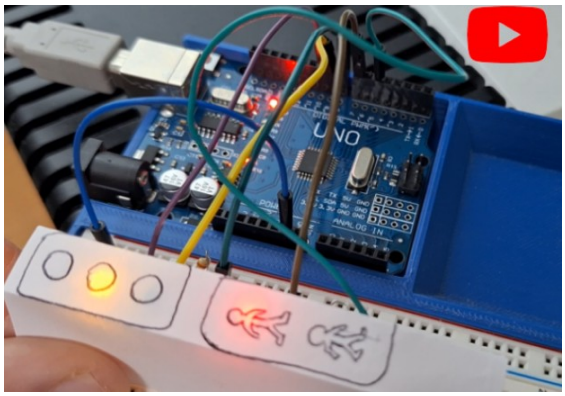
```
int verde = 13 ;
int amarillo = 12 ;
```

```
int rojo = 11 ;
int tiempoVerde = 3500;
int tiempoAmarillo = 1500 ;
int tiempoRojo = 3500 ;

void setup() {
  pinMode(verde, OUTPUT);
  pinMode(amarillo, OUTPUT);
  pinMode(rojo, OUTPUT);
}

void loop() {
  digitalWrite(verde, HIGH);
  delay(tiempoVerde);
  digitalWrite(verde, LOW);
  digitalWrite(amarillo, HIGH);
  delay(tiempoAmarillo);
  digitalWrite(amarillo, LOW);
  digitalWrite(rojo, HIGH);
  delay(tiempoRojo);
  digitalWrite(rojo, LOW);
}
```

☑️ TAREA 1.2. Semáforo de coches y de peatones.

Vamos a hacer una maqueta en el que hay un semáforo de coches y otro de peatones.

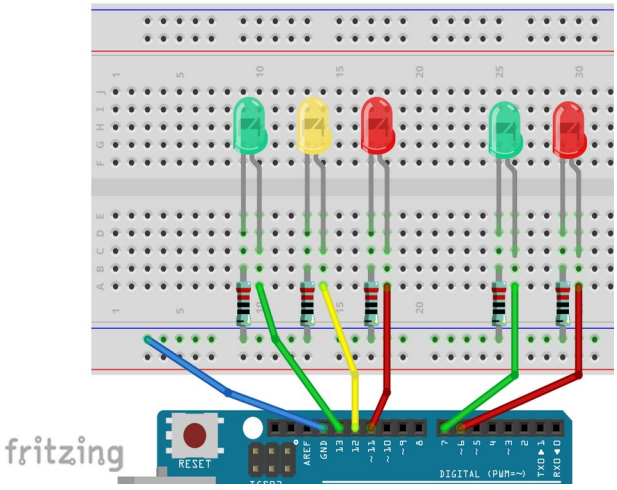
Debes programarlo para que cuando el semáforo de coches esté en rojo se ponga en verde el de peatones y viceversa.



Por seguridad, habrá un pequeño tiempo de espera entre esos cambios (entre el rojo de los coches y el verde de los peatones, así como entre el rojo de los peatones y el verde de los coches).

Puedes un ejemplo en **vídeo con el resultado**.

Puedes ver una sugerencia de montaje eléctrico en la imagen.



¿Necesitas ayuda con el programa? Aquí tienes dos pistas por si las necesitas:

- A continuación puedes ver una imagen con una sugerencia para comenzarlo, **está por terminar** (los tiempos son muy cortos para visualizarlo más rápido, los puedes cambiar):

- Así sería el programa completo con Arduino IDE:

```
int verdeCoches = 8;
int amarilloCoches = 9;
int rojoCoches = 10;
int rojoPeatones = 6;
int verdePeatones = 7;
int tiempoVerdeCoches = 7000;
int tiempoAmarilloCoches = 2000;
int tiempoRojoCoches = 5000;
```

```ino
int tiempoSeguridad = 1000;
int tiempoVerdePeatones = 3000;

void setup() {
  pinMode(verdeCoches, OUTPUT);
  pinMode(amarilloCoches, OUTPUT);
  pinMode(rojoCoches, OUTPUT);
  pinMode(verdePeatones, OUTPUT);
  pinMode(rojoPeatones, OUTPUT);
}

void loop() {
  digitalWrite(rojoPeatones, HIGH);
  digitalWrite(verdeCoches, HIGH);
  delay(tiempoVerdeCoches);
  digitalWrite(verdeCoches, LOW);
  digitalWrite(amarilloCoches, HIGH);
  delay(tiempoAmarilloCoches);
  digitalWrite(amarilloCoches, LOW);
  digitalWrite(rojoCoches, HIGH);
  delay(tiempoSeguridad);
  digitalWrite(rojoPeatones, LOW);
  digitalWrite(verdePeatones, HIGH);
  delay(tiempoVerdePeatones);
  digitalWrite(verdePeatones, LOW);
  delay(tiempoSeguridad);
  digitalWrite(rojoCoches, LOW);
}
```

9. Alimentación de la placa Arduino UNO

Una vez que cargamos el programa a la placa, éste se queda en la memoria interna y puede seguir funcionando sin depender de la conexión al ordenador.

Veamos diferentes formas de alimentar la placa sin necesidad de estar conectada al ordenador por USB.

Con un cargador de móvil.

Un cargador de móvil entrega normalmente 5V por USB, igual que un powerbank o un ordenador.

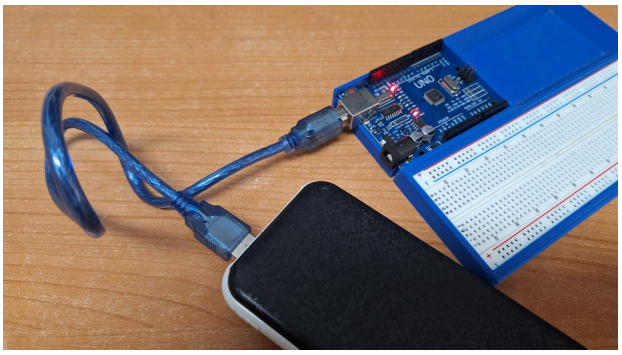
👉 **¿Cómo conectarlo?**

- Usas un cable USB (tipo A a USB-B) y lo enchufas al Arduino.
- El Arduino recibe 5V estables a través de su puerto USB.

🟢 **Son muy seguros,** tienen protecciones internas contra:

- Sobrecorriente
- Cortocircuitos
- Temperatura
- Picos de tensión

Alimentación mediante puerto USB desde un “power bank”



Esta es una forma muy práctica y cada vez más usada, especialmente para proyectos móviles, prototipos escolares.

Una batería externa (power bank) entrega 5V regulados a través del USB, igual que si conectaras el Arduino al ordenador. Simplemente conectas el cable USB normal.

✔️ **Ventajas:**

- **Muy estable**: la mayoría de power banks ofrecen 5V regulados.
- **Protección integrada**: control de sobrecorriente, cortocircuito, temperatura.
- **Gran capacidad**: 5.000–20.000 mAh → muchas horas de uso.
- **No requiere electrónica adicional.**
- **Seguridad energética muy alta.**

🔋 **No apto para alimentar cargas grandes del Arduino.**

Si conectas motores, servos o tiras LED directamente, el consumo puede subir a niveles que el power bank no está diseñado para entregar por mucho tiempo.

En esos casos: usar una fuente de alimentación independiente.

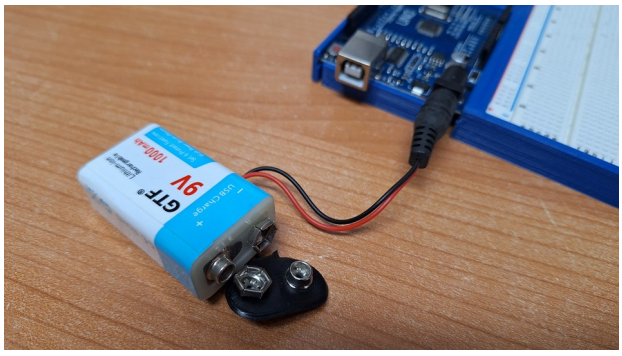
❗ **Posible problema y solución: Power banks que se apagan solos**

Algunos power banks se apagan si el consumo es bajo (< 80–100 mA). El Arduino UNO suele consumir entre 50–70 mA, dependiendo de los periféricos.

Soluciones:

- Añadir una carga pequeña (por ejemplo, un LED con resistencia).
- Usar un power bank para “dispositivos pequeños” o “modo carga baja”.
- Elegir un power bank que soporte low current mode o trickle mode.

Alimentación mediante el conector de alimentación (DC jack)



- 🔋 Voltaje recomendado: 7–12 V DC
- Conector: **2.1 mm centro positivo**

El Arduino incorpora un **regulador interno**, por lo que admite tensiones mayores que los 5 V operativos.

✔️ **Ventajas:**

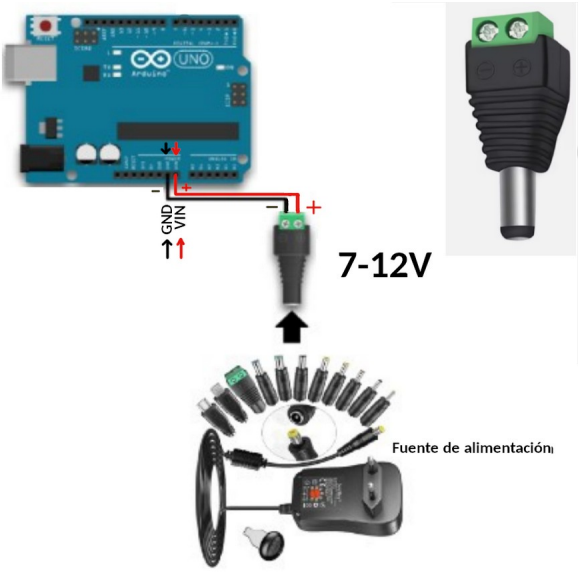
- Muy sencillo de usar.
- Protección y regulación integradas.
- Ideal para adaptadores y baterías externas de 9V.

❗ **Desventajas:**

- No usar más de 12V: el regulador se calienta y pierde eficiencia.
- Las pilas de 9V duran poco con cargas medianas.

Alimentación por el pin VIN del conector de pines

- 🔋 Voltaje recomendado: 7–12 V DC
- Funciona de forma similar al jack: pasa por el mismo regulador de 5V.



✔️ **Ventajas:**

- Permite alimentar desde baterías recargables, packs de pilas, etc.
- Muy útil en prototipos sin usar el jack.

❗ **Precauciones:**

- No conectar 5V a VIN (tensión insuficiente → resets).
- No superar los 12V.

Alimentación directa por el pin 5V

⚠️ Voltaje requerido: **5 V EXACTOS,** aquí te saltas el regulador interno y alimentas directamente la placa.

✔️ **Ventajas:**

- Muy eficiente: ideal para baterías LiPo + regulador externo.

- Menos calor

❗ **Riesgos:**

- Si introduces más de 5V → daño inmediato.
- No tiene protección de polaridad.
- Necesitas una fuente muy estable y regulada.

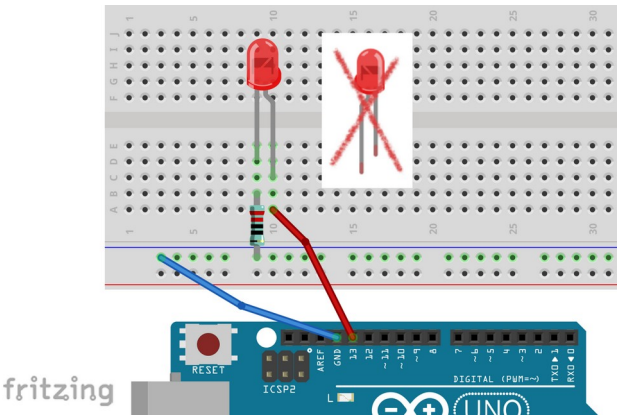
💡 **Resumen:**

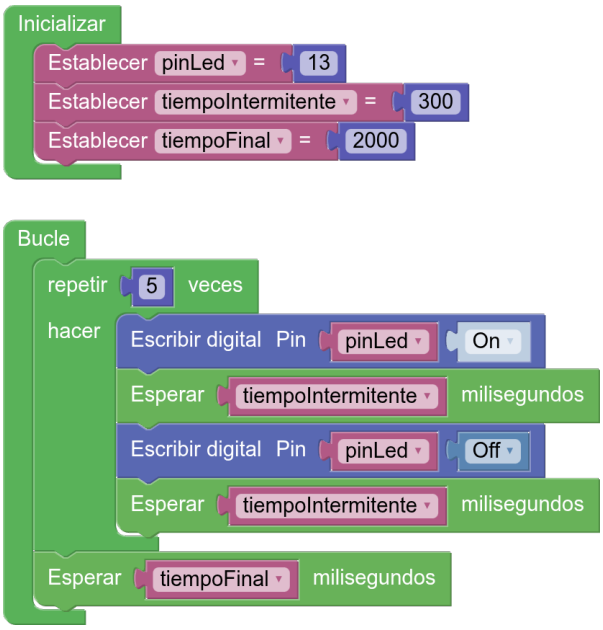
Método	Voltaje	Pasa por regulador	Seguro	Comentario
Power Bank USB	5V	No (regulado por el power bank)	⭐ Muy seguro	Práctica y estable.
DC Jack	7–12V	Sí	⭐ Muy seguro	Perfecto para adaptadores y pilas.
VIN	7–12V	Sí	⭐ Seguro	Ideal para baterías externas.
Pin 5V	5V exactos	No	⚠️ Con cuidado	Requiere regulador externo.
Pin 3.3V	3.3V	No	❌ No usar	No alimenta la placa.

10. Bloque repetir

▶️ **Vídeo 3: Repetir y contar en Arduino con STEAMakersBlocks**

En el programa de la imagen puedes ver cómo ahorrarnos trabajo con tareas repetitivas. Fíjate cómo podríamos hacer que un LED conectado al pin 13 parpadee 5 veces y después se quede apagado durante un tiempo final de 2 segundos, para después volver a repetir el proceso:



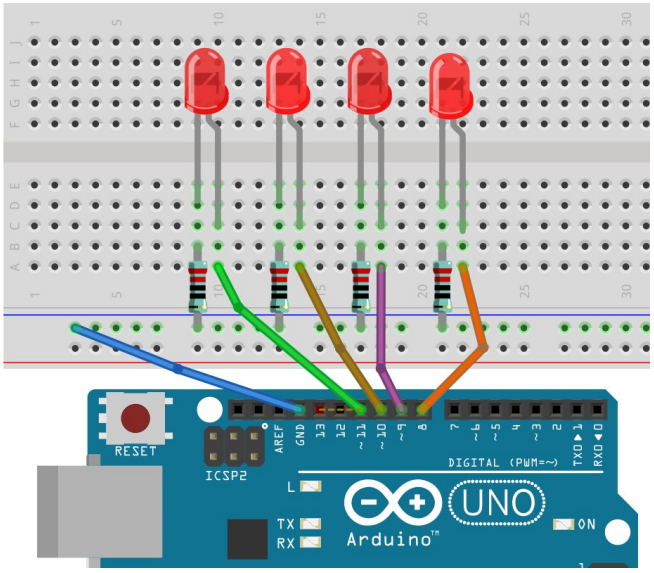


Práctica 2. Repetir un parpadeo.

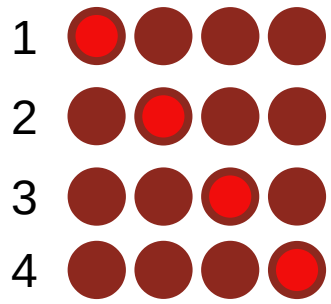
Con el montaje anterior, realiza un programa para que un LED parpadee 4 veces y después se quede 3 segundos apagado antes de repetir el ciclo.

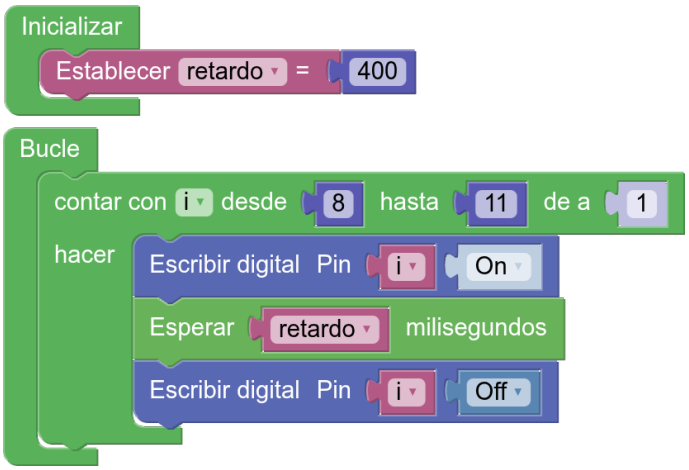
11. Bloque “contar”. Prácticas con secuencias de luces.

Vamos a realizar el siguiente montaje para hacer unas prácticas con secuencias de luces:



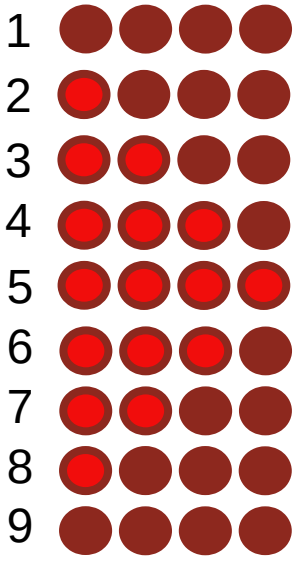
Práctica 3. Secuencia 1, luz que avanza

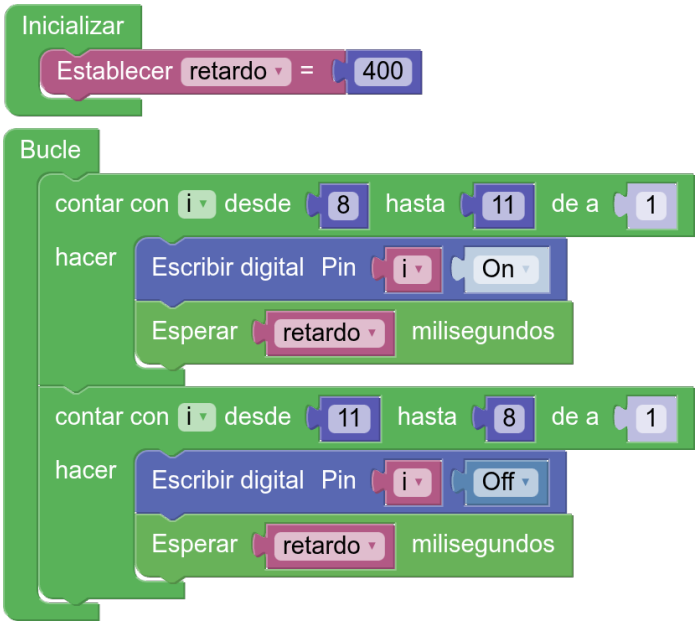





```
int retardo = 400;
void setup() {
for (int i = 8; i <= 11; i++) {
pinMode(i, OUTPUT);
}
}

void loop() {
for (int i = 8; i <= 11; i++) {
digitalWrite(i, HIGH);
delay(retardo);
digitalWrite(i, LOW);
}
}
```

Práctica 4. Secuencia 2, luces con efecto ecualizador gráfico

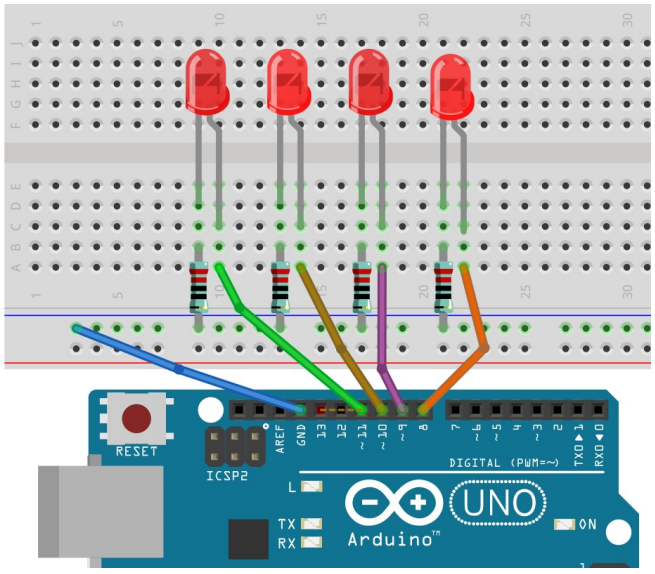




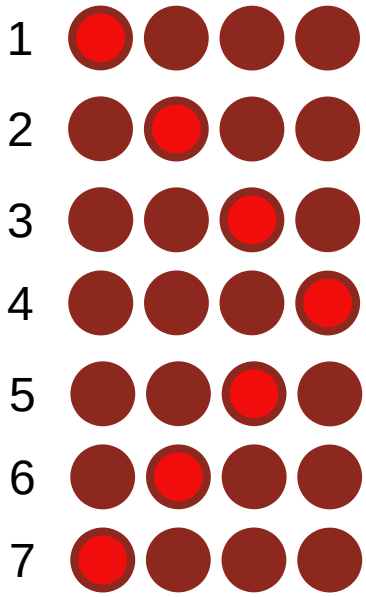
```
int retardo = 400;
void setup() {
for (int i = 8; i <= 11; i++) {
pinMode(i, OUTPUT);
}
}

void loop() {
for (int i = 8; i <= 11; i++) {
digitalWrite(i, HIGH);
delay(retardo);
}
for (int i = 11; i >= 8; i--) {
digitalWrite(i, LOW);
delay(retardo);
}
}
```

☑️ TAREA 2. Secuencia de luces de vaivén al estilo “coche fantástico”

La empresa TechnoCars busca estudiantes capaces de recrear un sistema de luces dinámicas para un prototipo de coche autónomo. Tu misión es construir con Arduino una barra de LEDs que simule la secuencia que lucía el **coche fantástico en la mítica serie (clic para ver en vídeo)**. Deberás montar el circuito y programar la secuencia para presentarlo al “equipo de ingeniería”.



Montaje



Secuencia

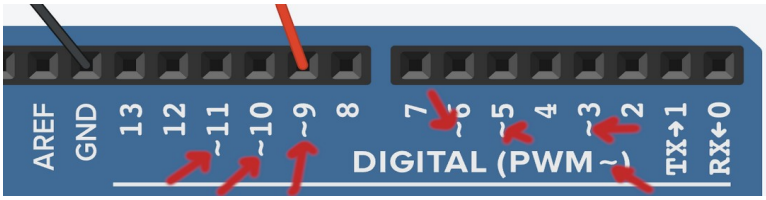
¿Necesitas una pista? Así sería el código del programa:

```
int retardo = 400;
void setup() {
  for (int i = 8; i <= 11; i++) {
    pinMode(i, OUTPUT);
  }
}

void loop() {
  for (int i = 8; i <= 11; i++) {
    digitalWrite(i, HIGH);
    delay(retardo);
    digitalWrite(i, LOW);
  }
  for (int i = 10; i >= 9; i--) {
    digitalWrite(i, HIGH);
    delay(retardo);
    digitalWrite(i, LOW);
  }
}
```

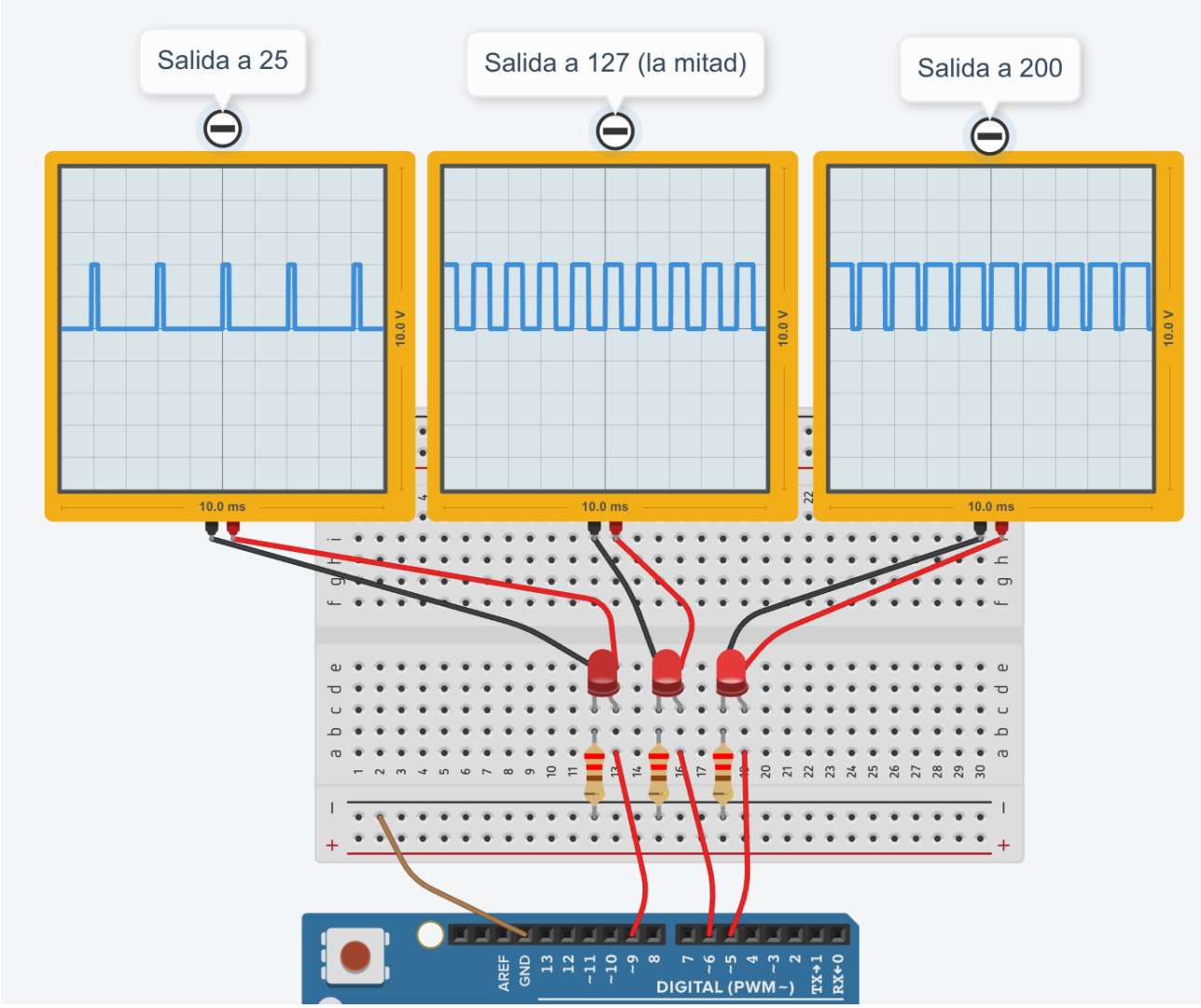
12. Graduar la intensidad de luz de un LED. Salidas PWM.

> ▶️ **Vídeo 4: PWM: Cambiamos la intensidad de brillo de LEDs con STEAMakersBlocks**

Nota: no todos los pines digitales se pueden usar con PWM, solamente los que se indican con la “onda”:

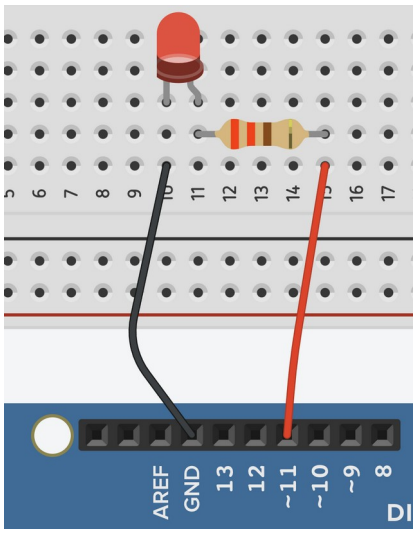


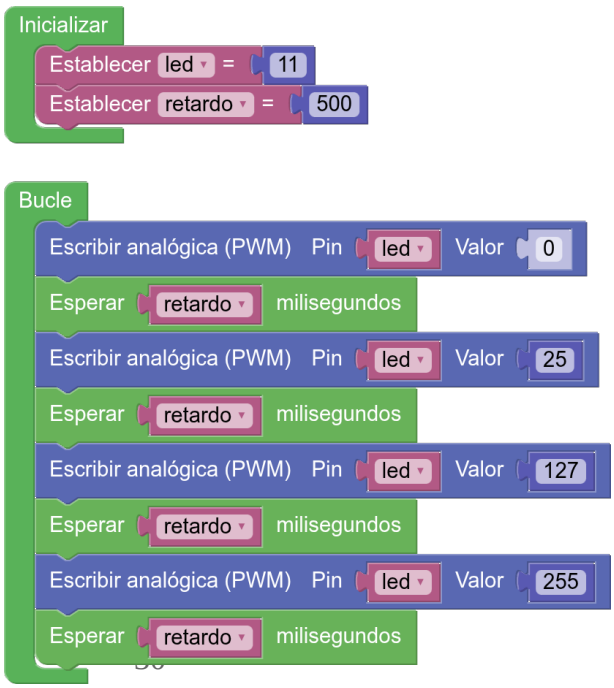
PWM: Pulse Wide Modulation (modulación de ancho de pulso). Se utiliza para simular valores de distinta intensidad analógicos mediante pulsos o “parpadeos” de la señal:



Práctica 5. Cambiamos la intensidad de brillo

Monta el circuito de la figura y el siguiente programa, observa cómo cambia el brillo del LED:





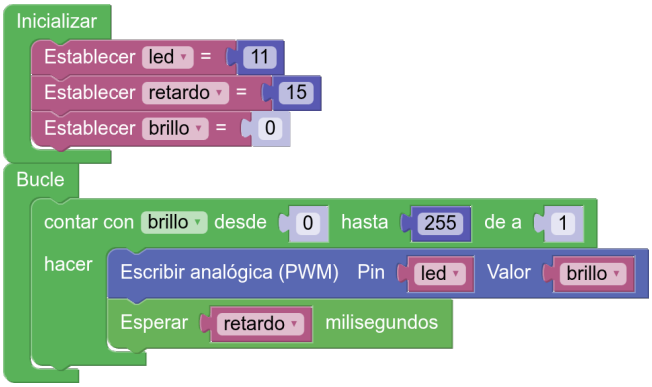
El mismo programa en Arduino IDE sería.

```
int led = 11;
int retardo = 500;

void setup() {
  pinMode(led,OUTPUT);
}

void loop() {
  analogWrite(led,0);
  delay(retardo);
  analogWrite(led,25);
  delay(retardo);
  analogWrite(led,127);
  delay(retardo);
  analogWrite(led,255);
  delay(retardo);
}
```

Práctica 6. Incrementos de brillo gradual con el bucle “contar”

Usando el bucle “contar”, programa el LED para que se encienda progresivamente de 0 a 255 con un retardo de 20 ms entre cambios de intensidad.

El programa sería:

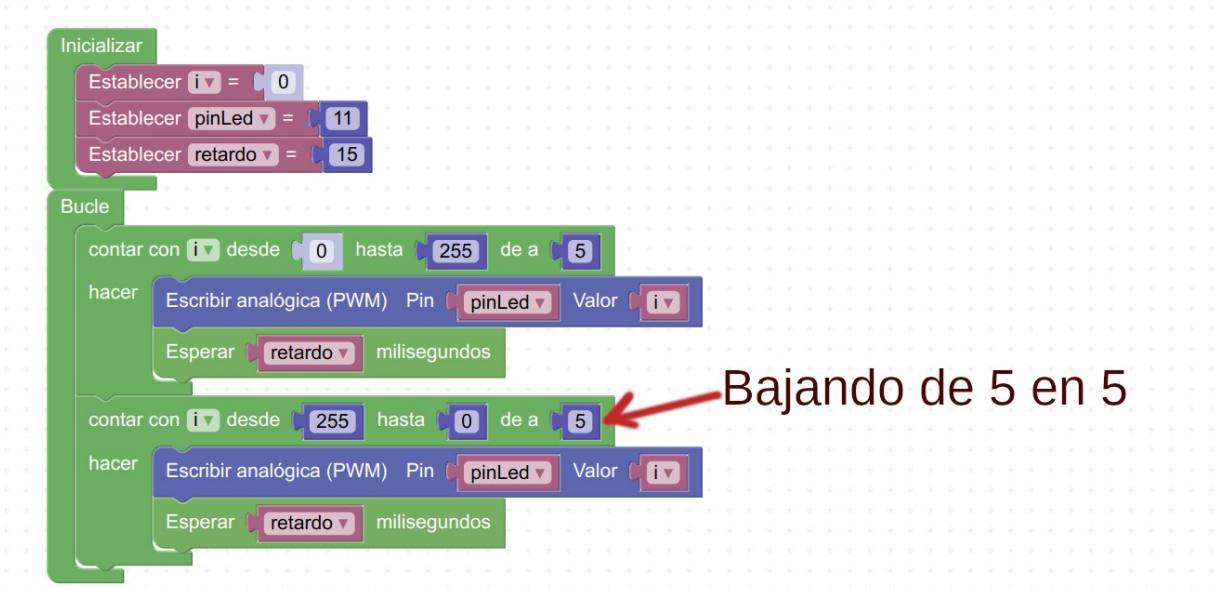


El código sería:

```
int led = 11;
int retardo = 15;
int brillo = 0;

void setup() {
  pinMode(led,OUTPUT);
}

void loop() {
  for (brillo = 0 ; brillo <= 255 ; brillo++){;
  analogWrite(led,brillo);
  delay(retardo);
  }
}
```

Práctica 7. Brillo con que sube y baja

Usando el bucle “contar”, programa el LED para que se encienda progresivamente de 0 a 255 de 5 en 5 y luego otro bucle “contar” para que descienda el brillo de 255 a 0 de 5 en 5, con un retardo de 15 ms entre cambios de intensidad. Fíjate en la imagen, está por completar, te servirá de guía:

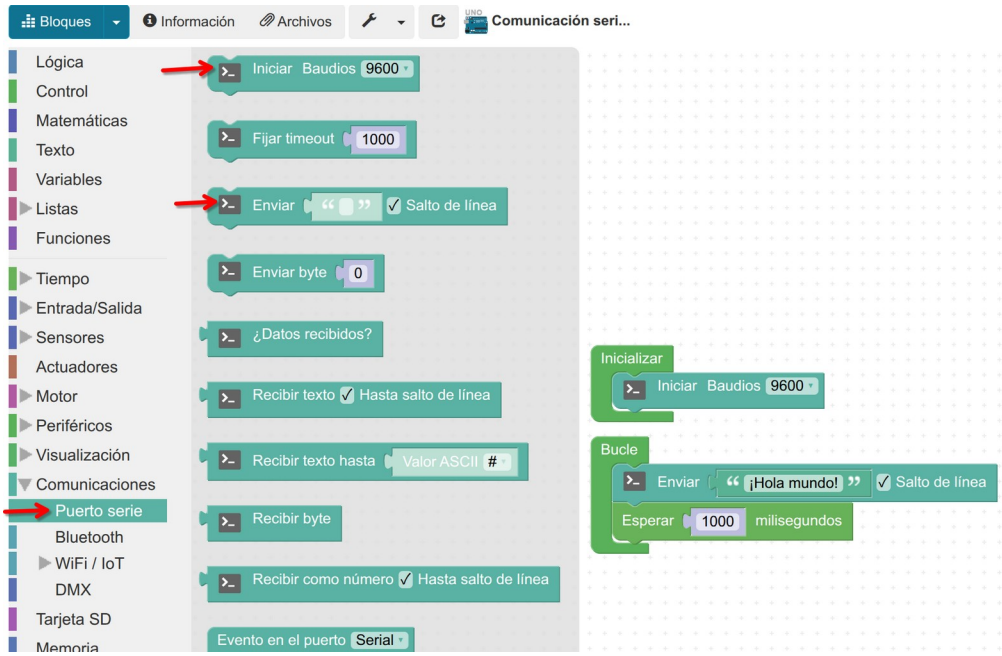


13. Consola serie. Recibiendo información de Arduino.

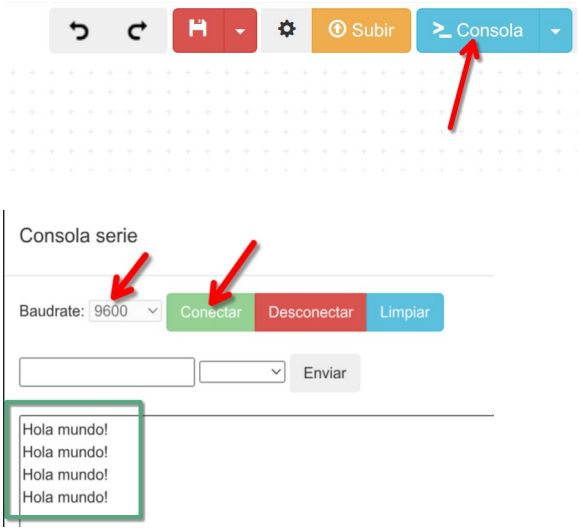
> ▶️ **Vídeo 5: Recibiendo información por consola serie en STEAMakersBlocks**

Programa básico Arduino nos manda el mensaje “¡Hola mundo!” cada segundo por la consola serie.

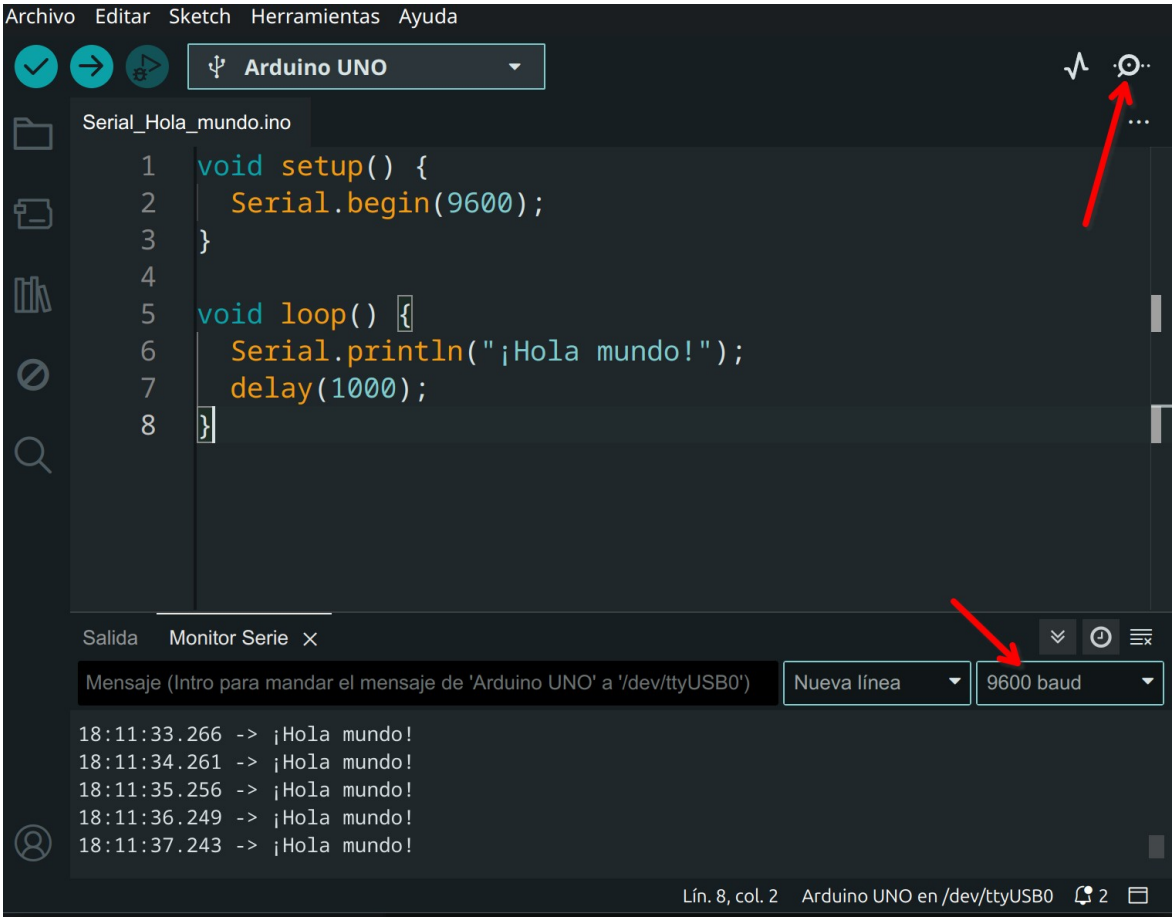
El programa sería:



El resultado lo podemos ver abriendo y conectando la consola serie:

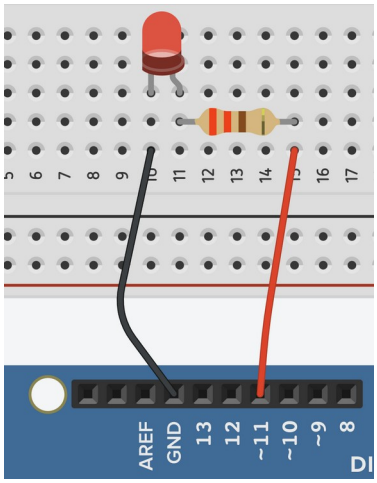


El mismo programa con Arduino IDE sería:

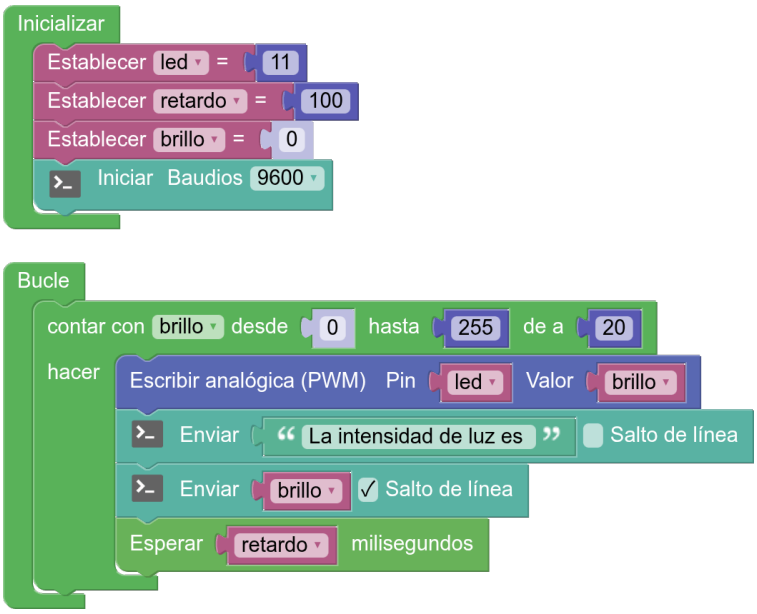


Práctica 8. Visualizando el nivel de brillo

Con el mismo montaje del ejercicio anterior:



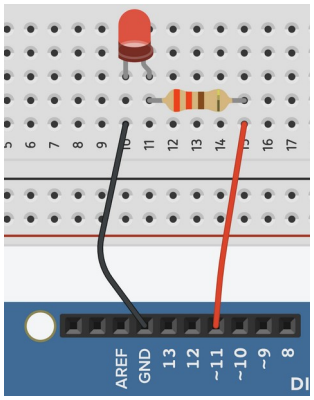
Realizamos un programa que hace que se incremente el nivel de brillo de 20 en 20 y **Arduino nos informa por comunicación serial del valor en cada momento**:



Resultado mostrado en la consola serie

☑️ TAREA 3: brillo que sube y baja, visualizando el valor por consola serie.

Queremos hacer un regalo por San Valentín que consiste en un corazón que hemos modelado e impreso en 3D y que tendrá un efecto de luz roja en su interior, de tal manera que suba y baje la intensidad de brillo dando una sensación de latidos al corazón.



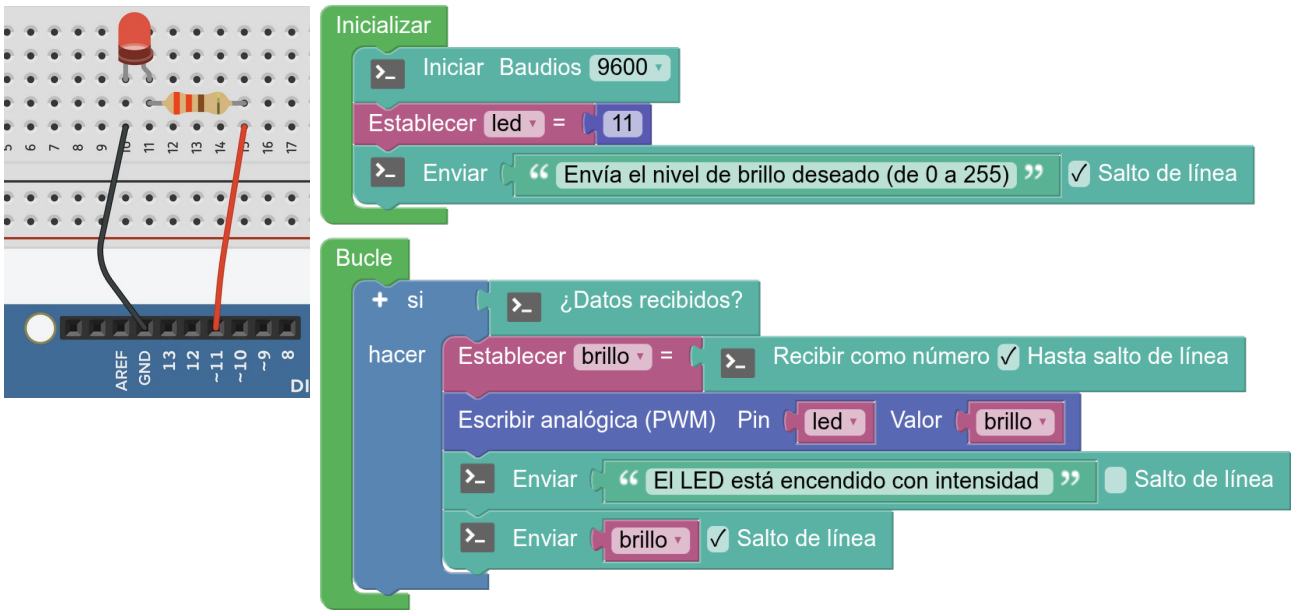
Usando el bucle “contar”, programa el LED para que se encienda progresivamente de 0 a 255 de 20 en 20 y luego otro bucle “contar” para que descienda el brillo de 255 a 0 de 20 en 20, con un retardo de 120 ms entre cambios de intensidad. En la consola serie debemos poder visualizar la intensidad de brillo en tiempo real.

14. Comunicación serial. Enviando información a Arduino.

> ▶️ **Vídeo 6. Enviando información serial con STEAMakersBlocks**

Con el circuito de la imagen probamos los siguientes programas:

Arduino pide el nivel de brillo deseado en la consola serie, escribimos y enviamos un número de 0 a 255 para que el LED se encienda con esa intensidad lumínica:



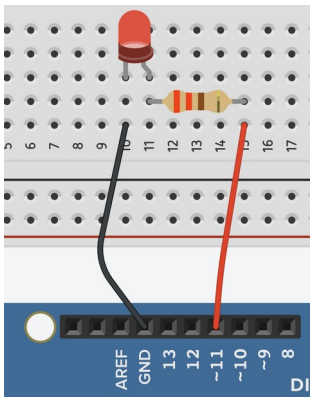
☑️ TAREA 4: Arduino nos pregunta si queremos encender o apagar un LED

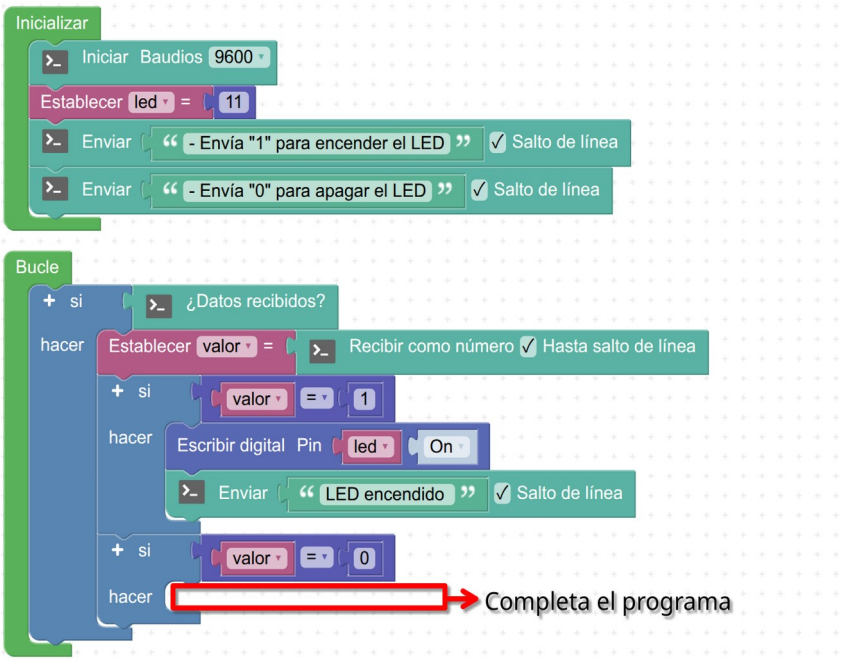
Queremos domotizar la luz de una habitación, de tal manera que la podamos encender o apagar desde un teléfono móvil.

En esta tarea realizaremos la primera fase del proyecto:

- Arduino nos informa por consola serie:
  - Envía “1” para encender el LED.
  - Envía “0” para apagar el LED.
- Si tecleamos “1” y enviamos, enciende el LED, mostrando la frase “LED encendido”
- Si se envía un “0” se apagará, mostrando la frase “LED apagado”

En la siguiente figura tienes un modelo de montaje eléctrico y modelo de programa que deberás completar:



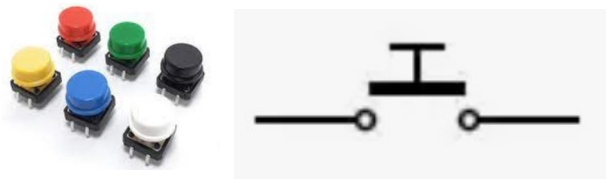


15. Entradas digitales, pulsadores, funciones, operadores lógicos “Y” y “O”

> ▶️ Vídeo 7. Pulsadores. Entradas digitales con STEAMakersBlocks

15.1. Pulsadores

Un **pulsador** es un sensor que sólo nos da dos estados, “pulsado o no pulsado”, por tanto, una señal digital. Conectado a la placa Arduino debe generar 0 V en reposo y 5 V al pulsarlo. De esta forma desde el programa de Arduino podremos leer el estado del botón.

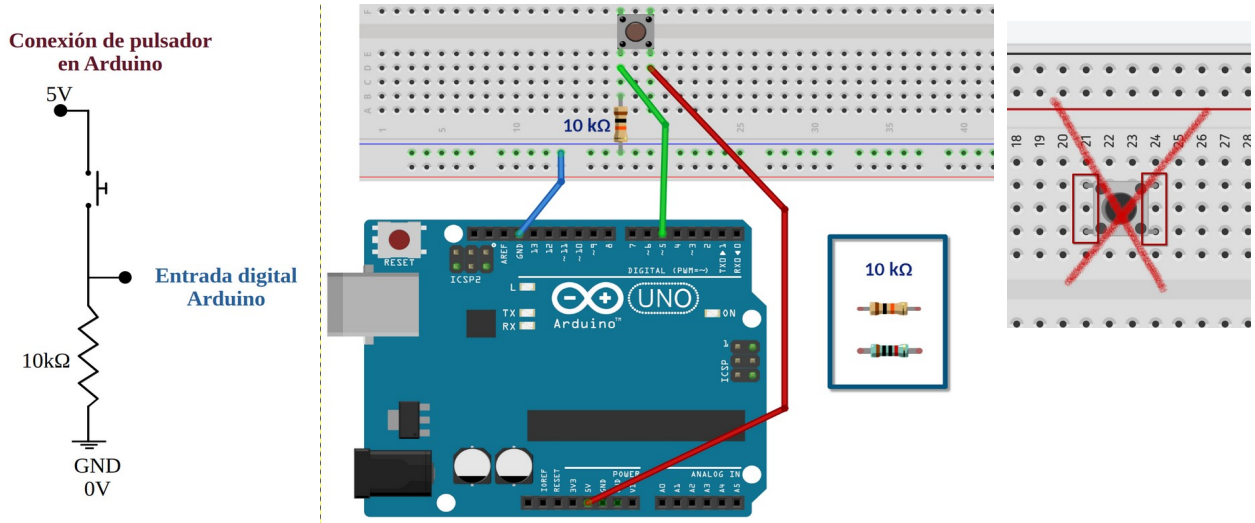


Pulsadores. Fotografía y símbolo

Práctica 9. Programa que muestra en la consola serie el estado del pulsador

> **Bandas de color de la resistencia de 10 kΩ que colocaremos con los pulsadores:**
>
> **De 3 bandas: marrón – negro - naranja (más la banda de tolerancia)**
>
> **De 4 bandas: Marrón – negro – negro – rojo (más la banda de tolerancia)**

Circuito:



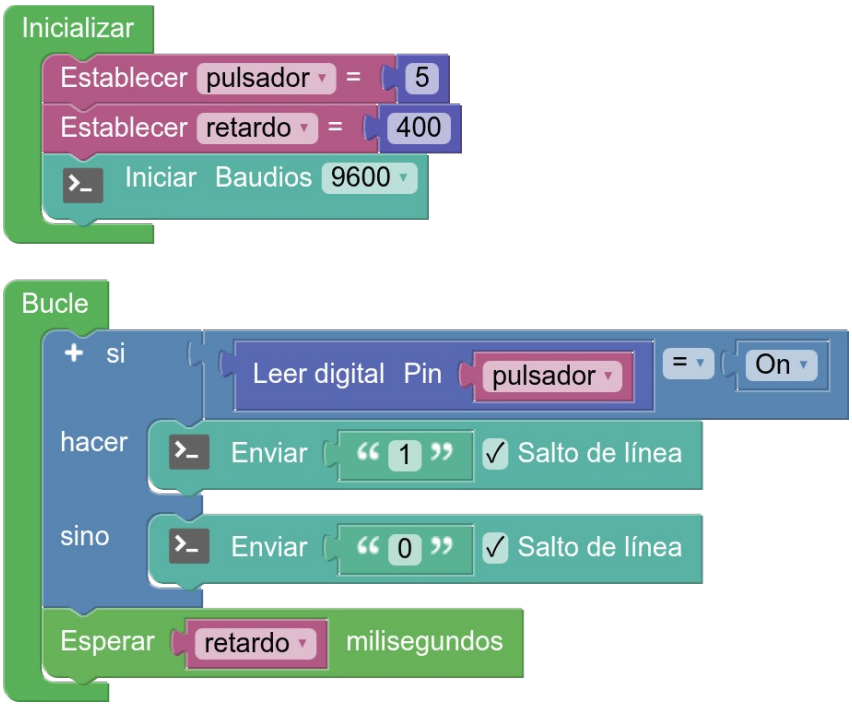
Se mostrará en la consola serie:

- 1 mientras pulsamos el pulsador
- 0 si no lo pulsamos

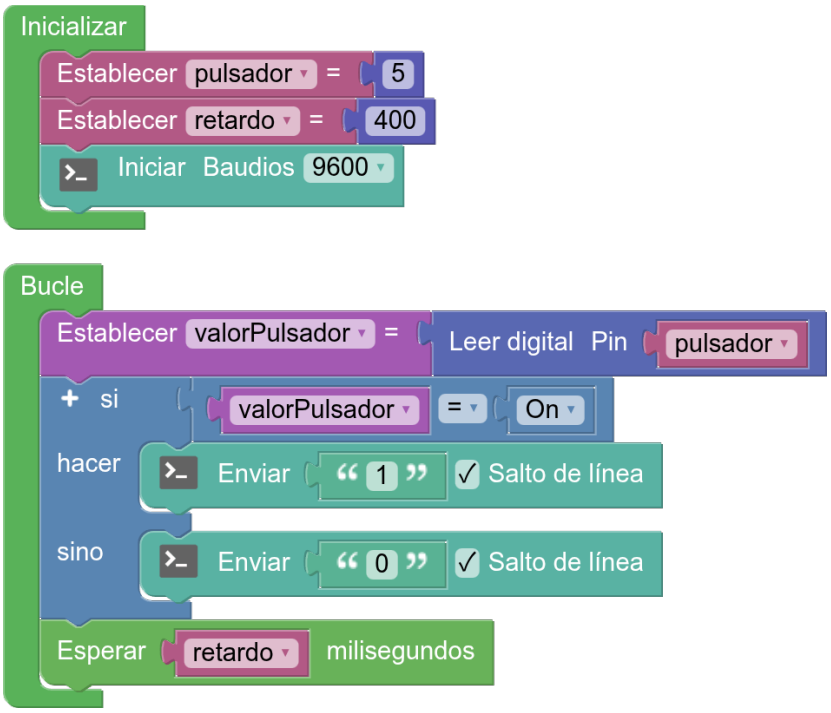

```
int pulsador = 5;
int retardo = 400;

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

void loop() {

  if (digitalRead(pulsador)) {
    Serial.println("1");
  } else {
    Serial.println("0");
  }
  delay(retardo);
}
```

Otras formas de hacer el programa más fáciles de entender, aunque un poco más largas:

- Usando la igualdad en la condición sí:

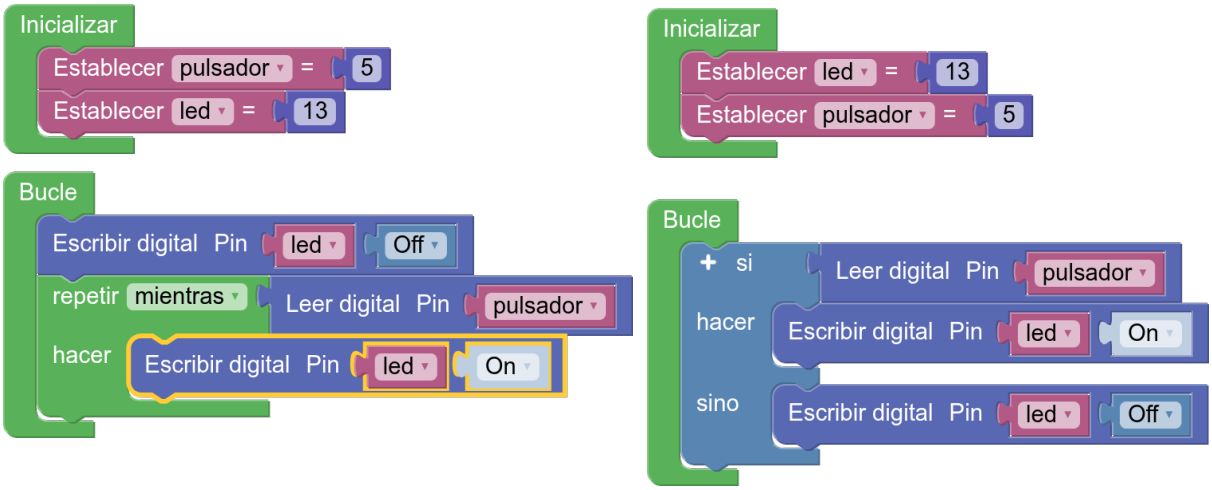


- Usando además una variable valorPulsador de tipo “Boolean”, es decir, solamente puede tener dos valores (On-Off):



Práctica 11. Enciende el LED integrado mientras accionamos el pulsador.

- Dos programas equivalentes para encender el LED integrado (conectado al pin 13) mientras accionamos el pulsador:



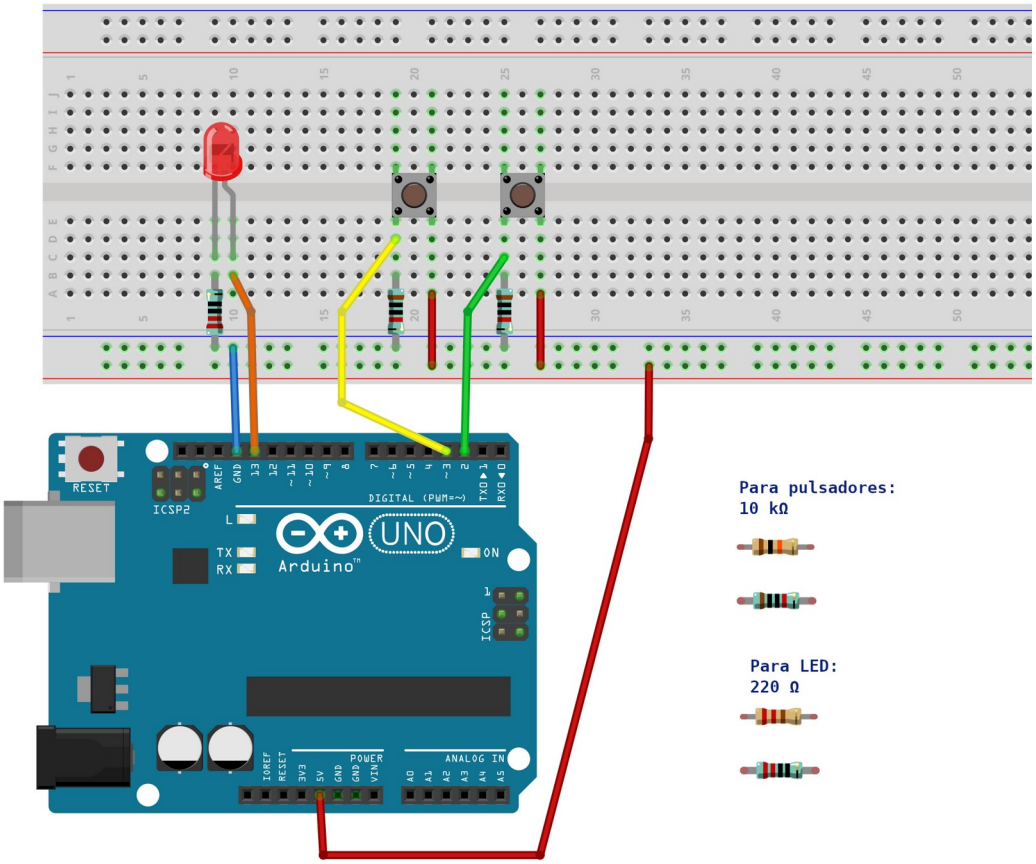
15.2. Pulsadores y operadores lógicos Y (and), O (or)

Práctica 12. Pulsadores y operador lógico “O”.

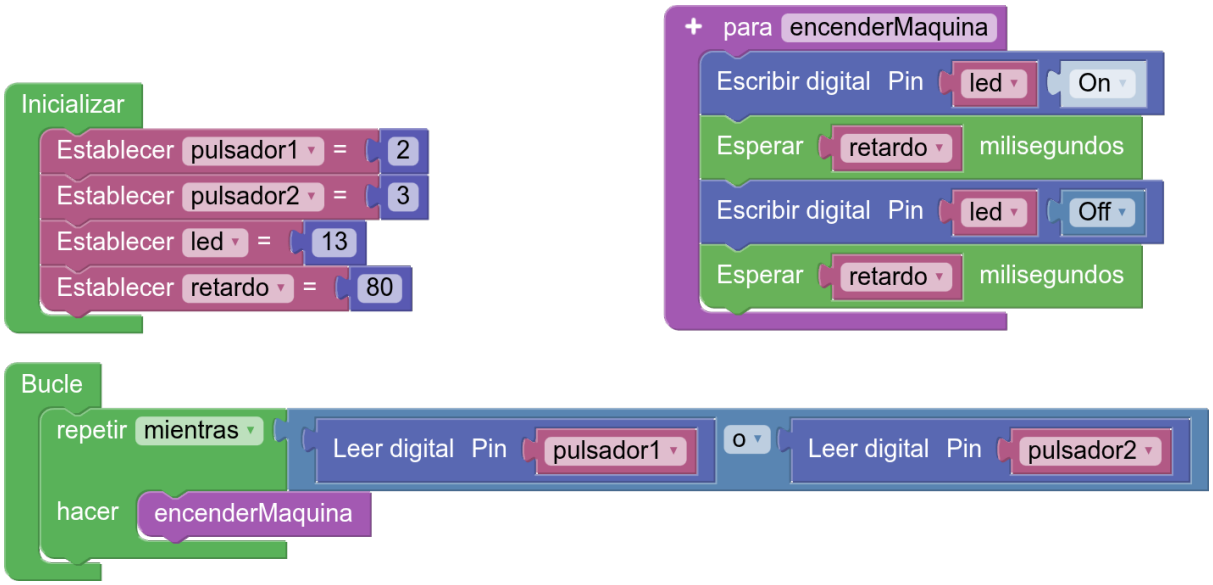
La empresa para la que trabajamos nos ha solicitado diseñar el sistema de control básico para una máquina auxiliar. En este caso, el operario debe poder activar la máquina desde dos posiciones diferentes que se encuentran distantes entre sí, por lo que el sistema debe arrancar mientras se pulsa cualquiera de los dos botones disponibles. El operario puede utilizar **uno u otro pulsador indistintamente**, y si pulsa ambos a la vez, la máquina también debe ponerse en funcionamiento sin ningún problema.

Para simular el funcionamiento de la máquina en nuestro prototipo utilizaremos un LED, que se parpadeará únicamente cuando ambos pulsadores se activen al mismo tiempo.

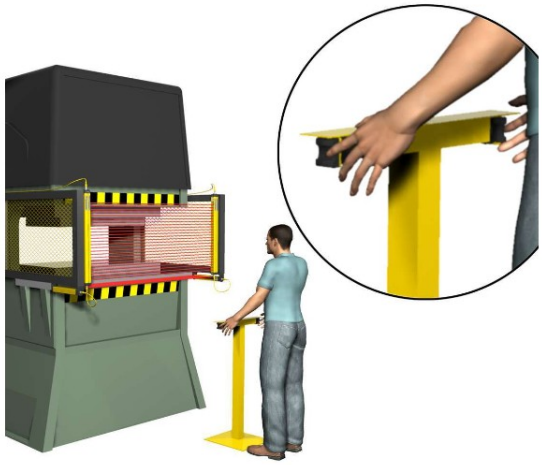
Sugerencia de montaje eléctrico:



Puedes usar un programa como el siguiente:



☑️ TAREA 5. Pulsadores y operador lógico “Y”



From www.bannerengineering.com

La empresa para la que trabajamos nos ha encargado el diseño de un sistema de seguridad para una máquina que puede resultar peligrosa si el operario introduce accidentalmente una mano mientras está en funcionamiento.

Para evitarlo, debemos implementar un mecanismo de activación segura basado en doble pulsación: la máquina sólo podrá ponerse en marcha cuando el operario pulse simultáneamente dos pulsadores, uno con cada mano, evitando así que le quede alguna libre que pueda introducir por error en la zona de riesgo.

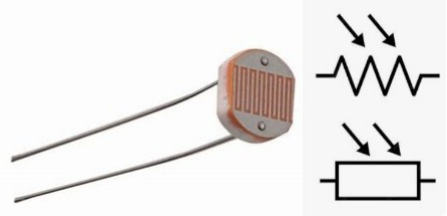
Para simular el funcionamiento de la máquina en nuestro prototipo utilizaremos un LED, que se parpadeará únicamente cuando ambos pulsadores se activen al mismo tiempo.

Se valorará el uso de alguna función, por ejemplo, una función “encenderMaquina”.

16. Entradas analógicas. Fotorresistencias o LDR

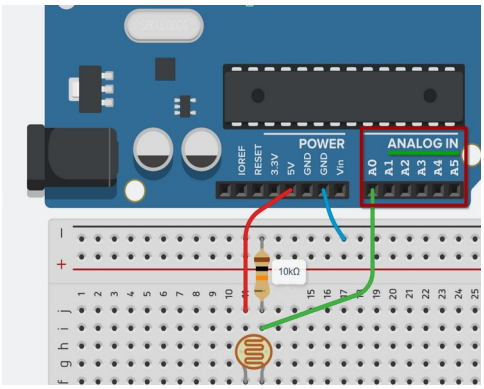
▶️ Vídeo 8. LDR. Entradas analógicas con STEAMakersBlocks

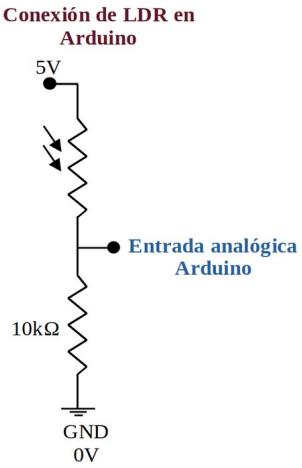
Fotorresistencia o LDR (Light Depending Resistor): resistencia que varía con la luz. A más luz, deja pasar más corriente. Por lo tanto, a más luz tiene menos resistencia.

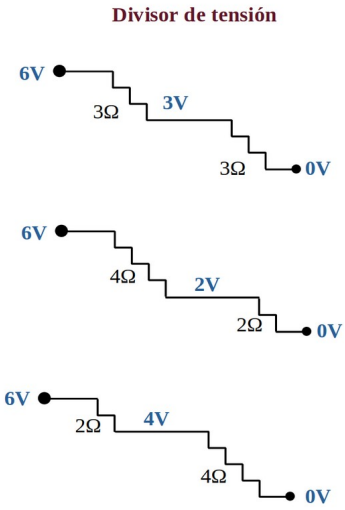


LDR. Fotografía y símbolo

La conexión eléctrica forma un divisor de tensión:





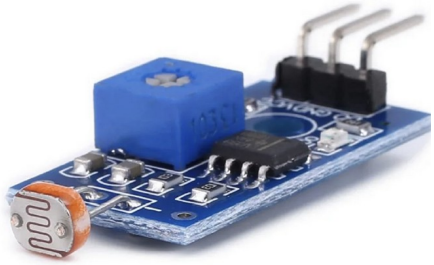


Bandas de color de la resistencia de 10 kΩ que colocaremos con los pulsadores:

- **De 3 bandas**: marrón-negro-naranja
- **De 4 bandas**: Marrón – negro – negro – rojo

Instrucciones básicas para una LDR (fotorresistencia):

Módulos LDR de 3 pines con potenciómetro:

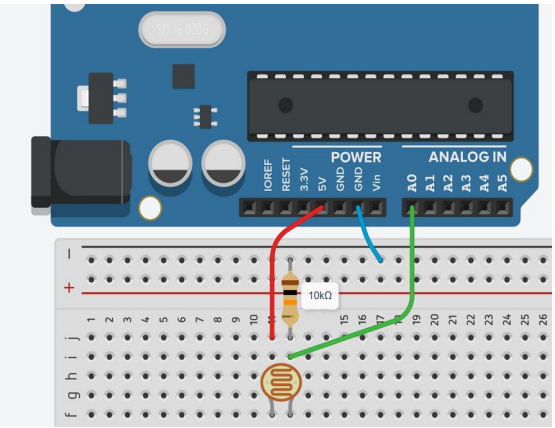


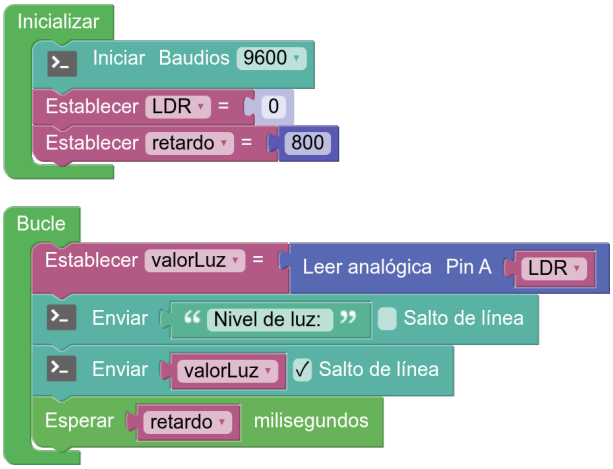
Hay módulos LDR con 3 pines en los que **no hace falta resistencia**, la lleva incorporada además de un potenciómetro para calibrarla.

Pines y su conexión a la placa Arduino:

- **Vcc:** a 5V de Arduino
- GND: a GND de Arduino
- DO (data output): a entrada analógica de Arduino

Práctica 13. Mostrar el valor de luz ambiental en la consola serie





Resultado. Si abrimos y conectamos la consola serie obtenemos la lectura:

El programa en Arduino IDE sería:

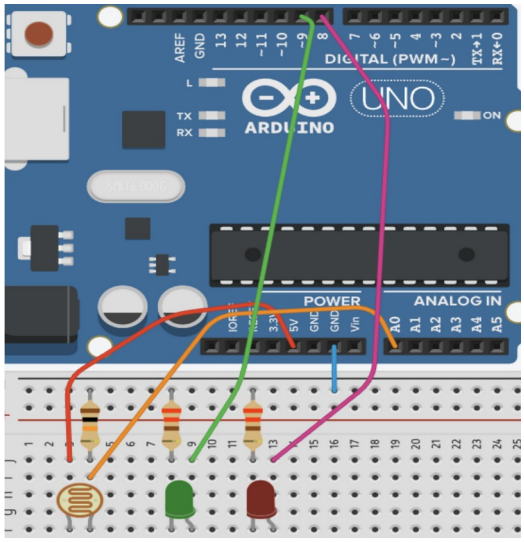
```
int pinLdr =A0 ;
int valorLuz ;
int retardo = 800 ;

void setup(){
  Serial.begin(9600);
  pinMode(pinLdr,INPUT);
}

void loop() {
  valorLuz = analogRead(pinLdr);
  Serial.println(valorLuz);
  delay(retardo);
}
```

Práctica 14. Información de nivel de luz con LEDs

La empresa SmartEnvironments está desarrollando un sistema básico de monitorización ambiental para distintos espacios de trabajo. Como parte del equipo de prototipado, se te ha asignado la tarea de diseñar un dispositivo capaz de medir el nivel de luz ambiental y proporcionar alertas visuales en función de la intensidad detectada.

Para ello, deberás montar el circuito mostrado en la imagen y programar un sistema basado en Arduino y un sensor LDR (resistencia dependiente de la luz). El comportamiento del prototipo será el siguiente:

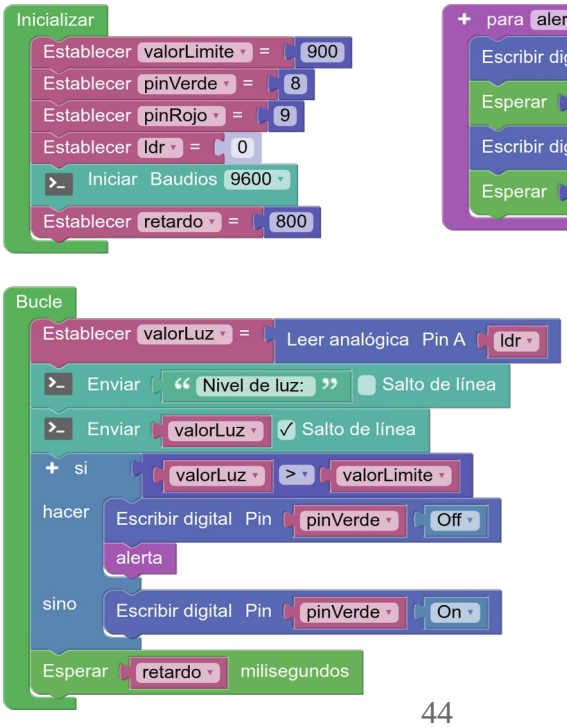
- Se debe visualizar el nivel de luz en la consola serie.
- Se visualiza una luz verde si hay un nivel normal de luz.
- Parpadea una luz roja de alarma en el caso de una luz muy brillante como sería la luz directa sobre la LDR de la linterna del móvil.



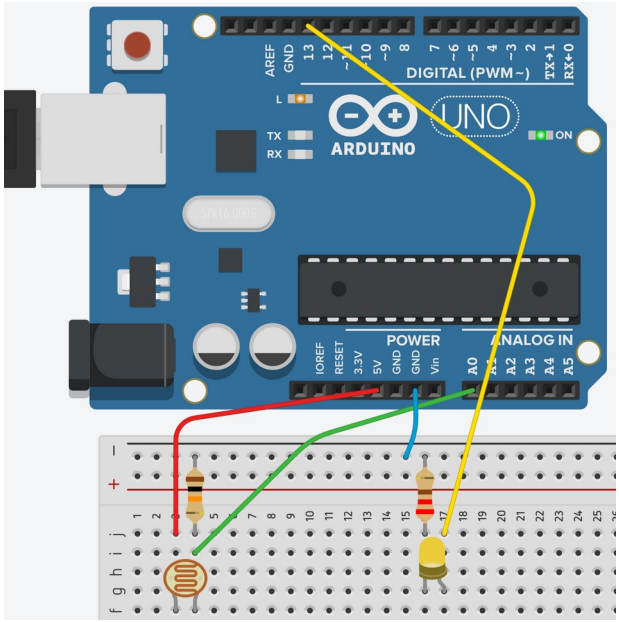
Necesitarás dos resistencias de 220Ω para los LEDs y otra de 10kΩ que va junto a la LDR. el valor de luz ambiental en la consola serie.

Se valorará el **uso de alguna función,** como puede ser una función llamada “alertaLuz” por la que el LED rojo parpadeará**.**

El programa podría ser como este:



☑️ TAREA 6. Sistema automático de iluminación.



La empresa EcoHome Solutions está desarrollando un sistema automático de iluminación para mejorar la eficiencia energética en hogares inteligentes. Como parte del equipo de prototipado, **debes diseñar un dispositivo capaz de encender una lámpara cuando el nivel de luz ambiental sea bajo.**

En esta primera fase, realizaremos el prototipo donde usaremos un LED conectado al pin 13 para visualizar el funcionamiento. En un futuro se conectará el sistema a una lámpara conectada a la red eléctrica mediante módulo relé.

Tu tarea consiste en:

- Montar el circuito indicado, que incluye una fotorresistencia (LDR) para captar el nivel ambiental.
- Programar Arduino para crear un sistema que controle un LED conectado al pin 13, encendiéndolo automáticamente cuando haya poca luz en el entorno.
- Deberemos poder visualizar el nivel de luz ambiental en la consola serie.

Necesitarás una resistencia de 220Ω para el LED y otra de 10kΩ para colocar junto a la fotorresistencia.

17. Sensor de temperatura y humedad DHT11. Tablas y gráficas con los valores de los sensores.

▶️ **Vídeo 9. Sensor de temperatura y humedad. Tablas y gráficas de datos.**

El sensor DHT11 permite detectar valores de temperatura y humedad.

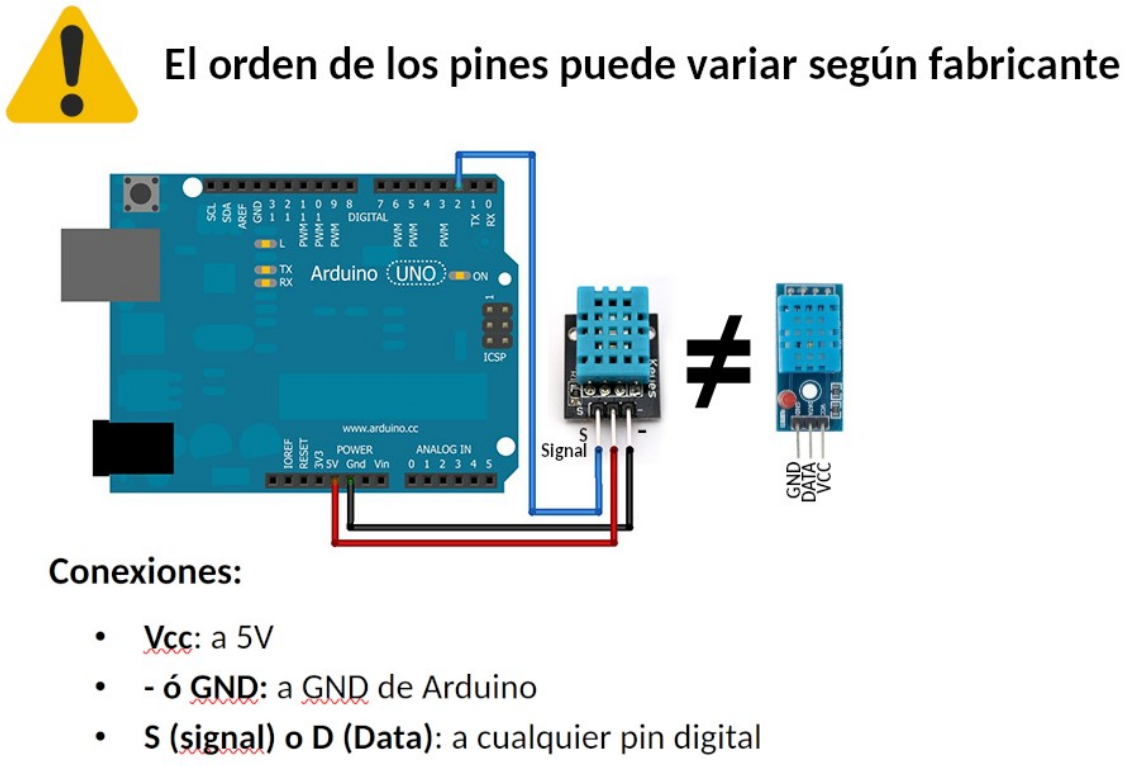
Especificaciones:

- Alimentación: 3.3 ó 5V.
- Rango de temperatura: 0 - 50°C. (5% de precisión)
- Rango de humedad: 20 - 80% (5% de precisión)
- 1 muestra por segundo.

Conexiones:

- **Vcc**: a 5V
- **GND:** a GND de Arduino
- **S (signal) o D (Data)**: a cualquier pin digital

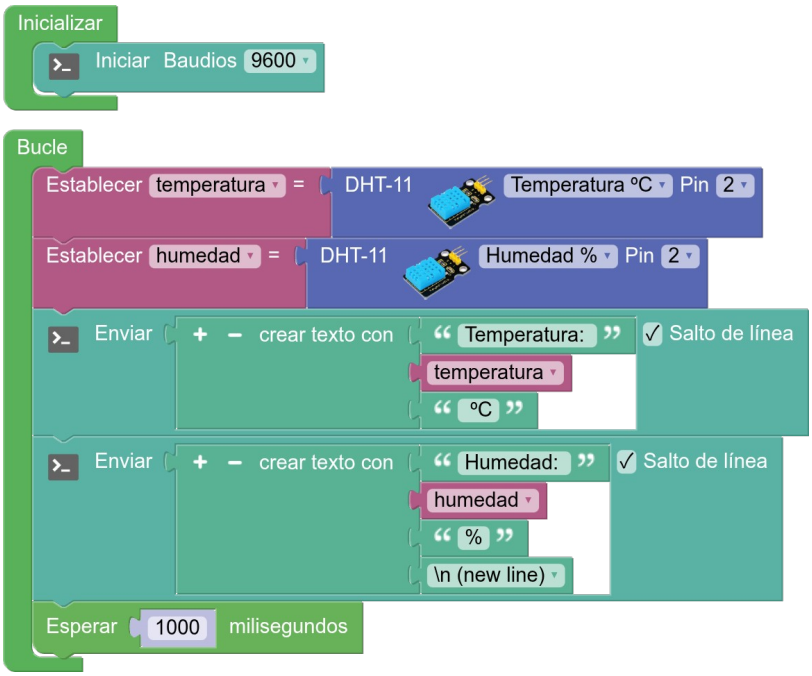
📝 **Para no dañar el sensor: hay que tener cuidado con el orden de los pines ya que distintos fabricantes pueden situar los pines en orden distinto. Es necesario mirar las inscripciones del módulo para asegurarnos de seleccionar los pines correctos.**



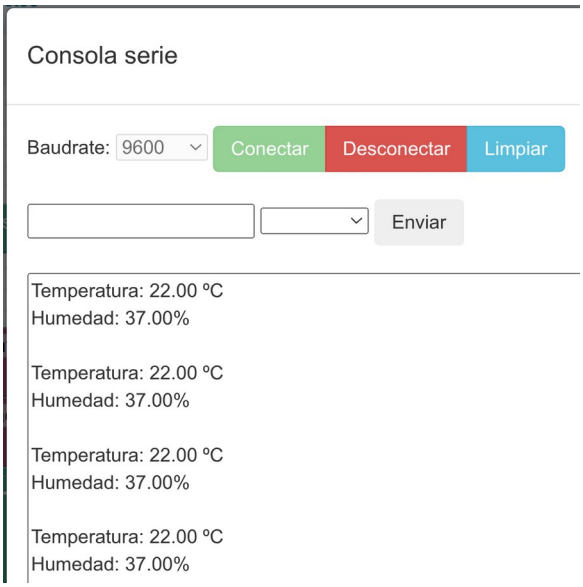
Conexiones:

- **Vcc**: a 5V
- **- ó GND:** a GND de Arduino
- **S (signal) o D (Data)**: a cualquier pin digital

Práctica 15: Mostrar en consola serie la temperatura y humedad.



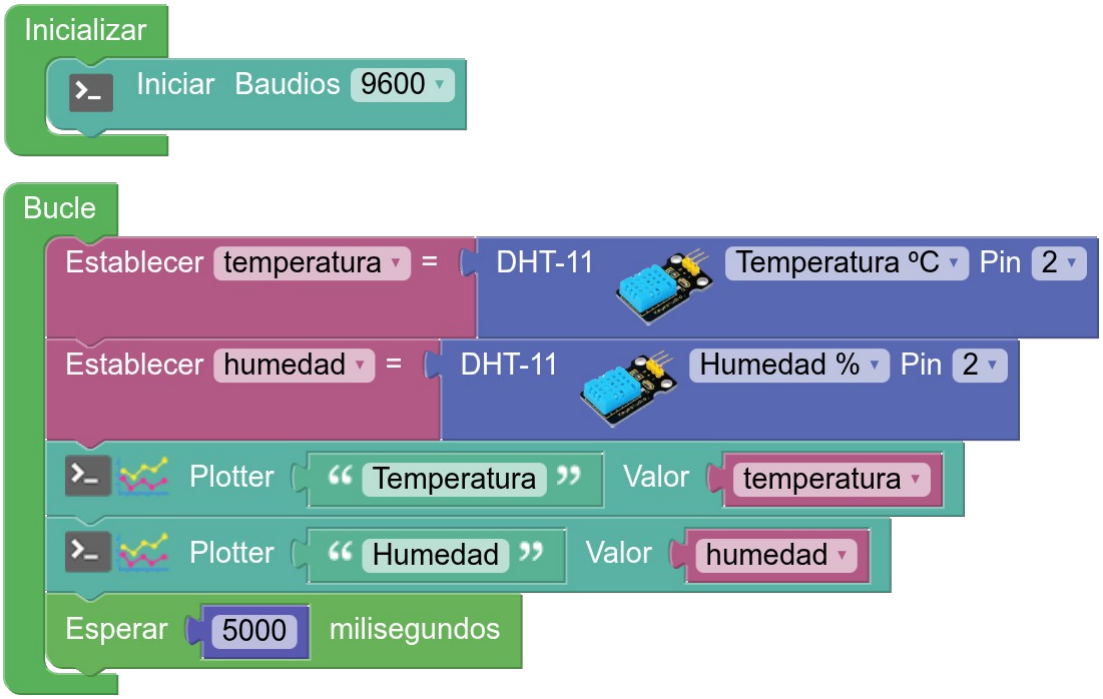
Resultado mostrado en la consola serie:



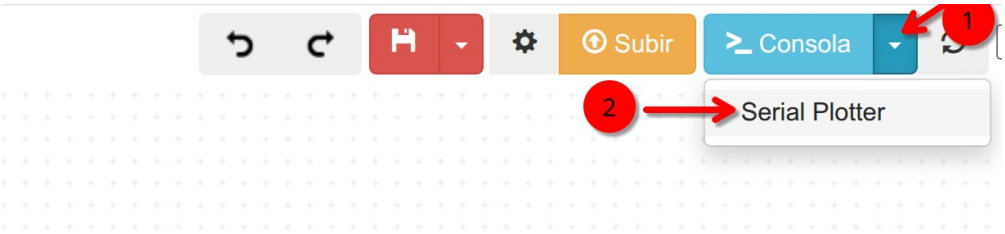
Práctica 16: Generar CSV y gráfica de datos.

Vamos a generar un archivo CSV que abriremos con LibreOffice y crearemos una gráfica de la evolución de la temperatura y humedad a lo largo del tiempo.

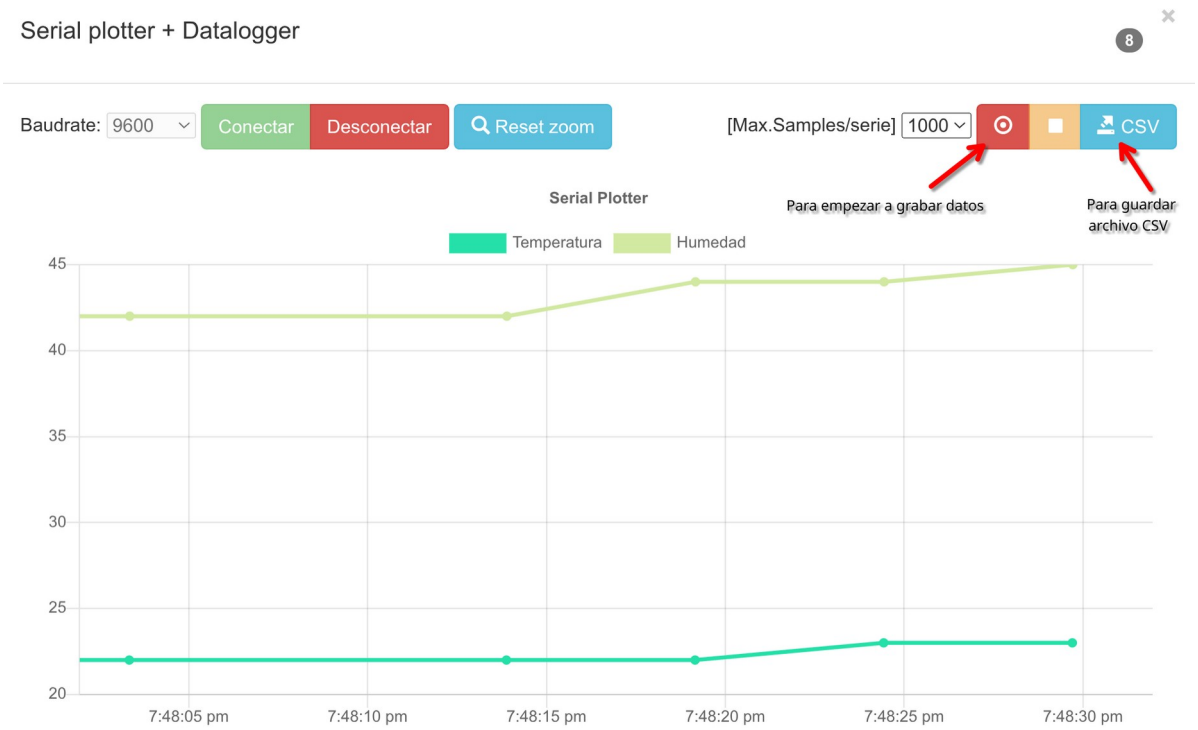
El programa sería:



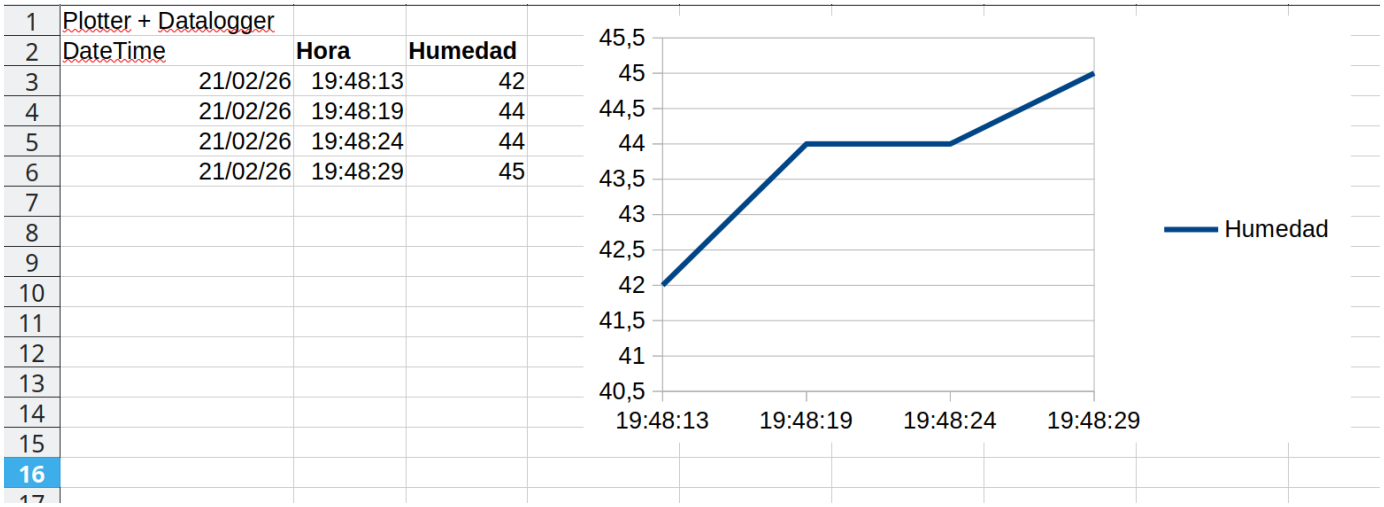
Abrimos el Serial Plotter para visualizar la gráfica en tiempo real:



Haciendo clic en Conectar, veríamos la gráfica con los valores:



El archivo CSV lo podemos abrir con un programa de hoja de cálculo y visualizar los datos y gráficas:



1	Plotter + Datalogger		
2	DateTime	**Hora**	**Humedad**
3	21/02/26	19:48:13	42
4	21/02/26	19:48:19	44
5	21/02/26	19:48:24	44
6	21/02/26	19:48:29	45
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			

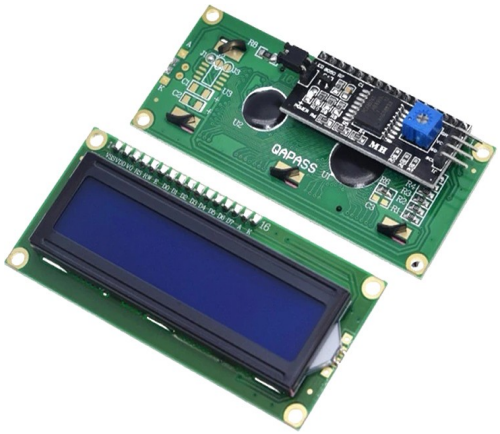
18. Pantalla LCD1602 I2C

> ▶️ **Vídeo 10. Pantalla LCD1602 con STEAMakersBlocks**

La pantalla LCD1602 tiene 16 caracteres en 2 filas y permite mostrar visualmente información, como pueden ser los datos de los sensores.

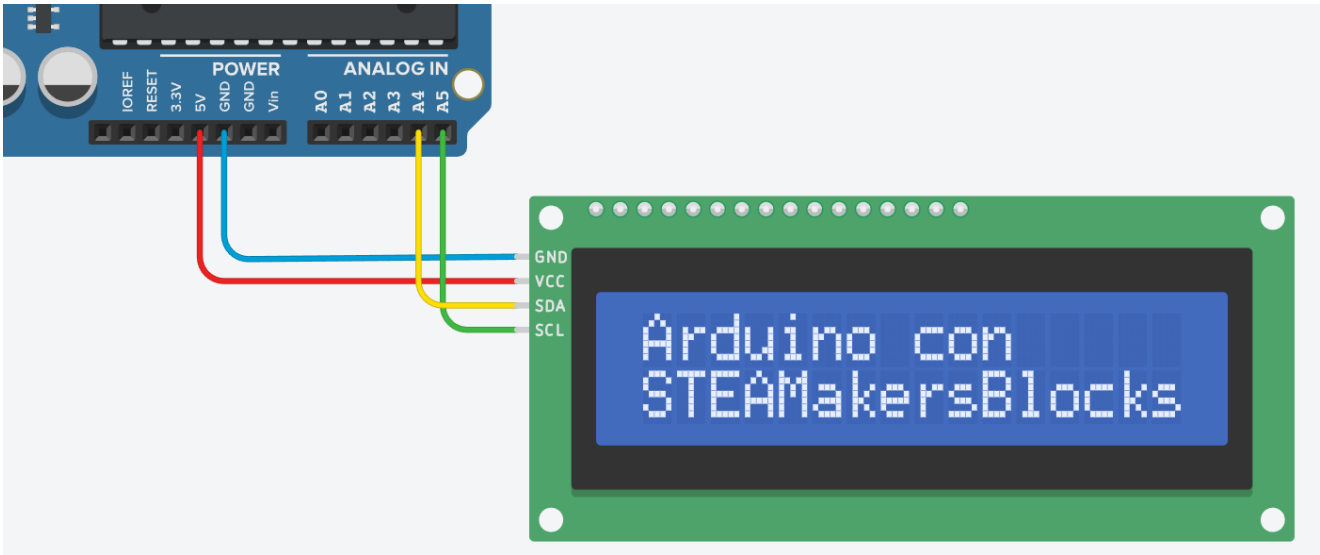
Nosotros vamos a usar la que tiene un módulo I2C que reduce las conexiones a 4 cables:

- GND
- Vcc (5V)
- SDA: **s**erial **da**ta, lo conectamos **siempre a A4 en Arduino UNO.**
- SCL: **s**erial **cl**ock lo conectamos **siempre a A5 en Arduino UNO**.

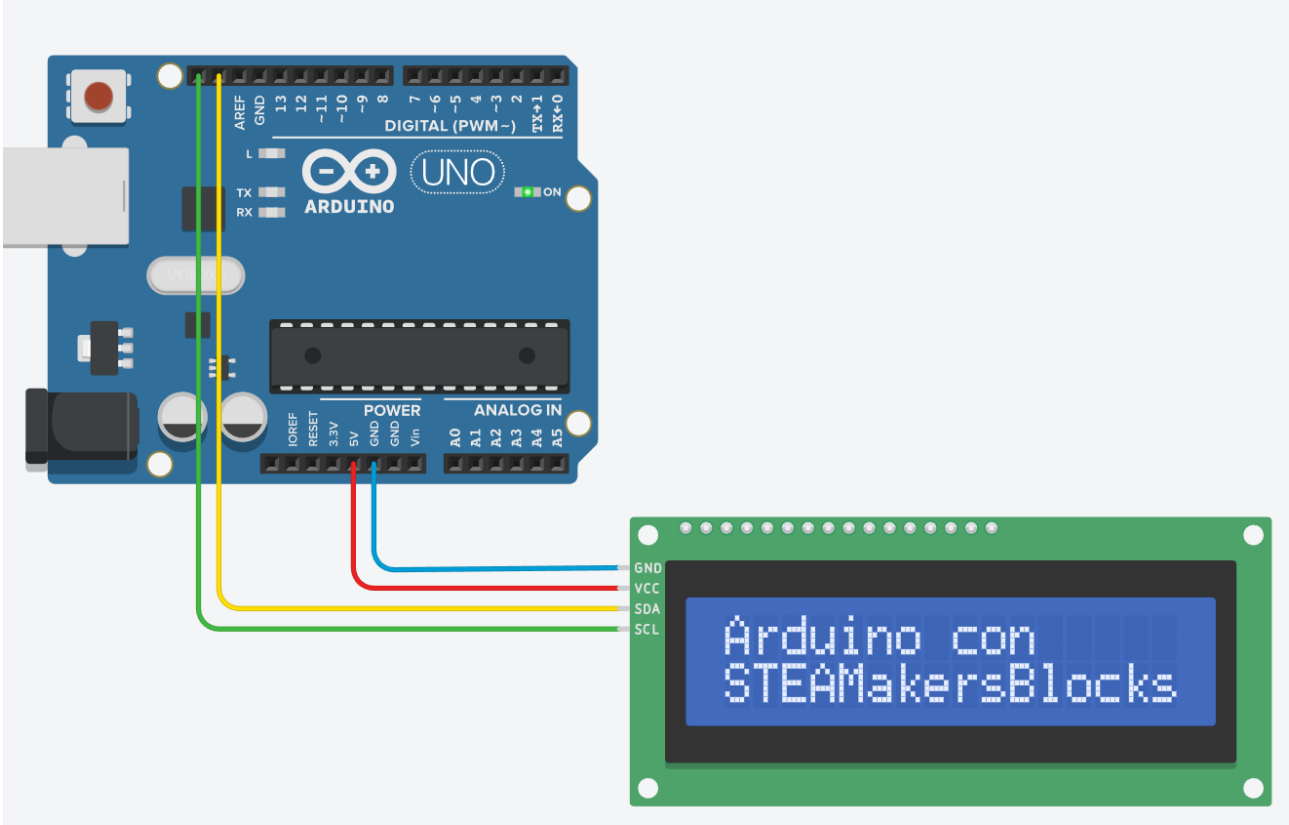


LCD 1602 I2C. Vista frontal y posterior

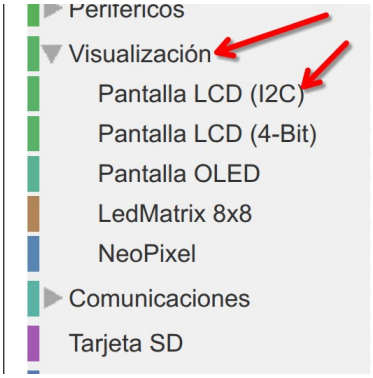
Conexiones:

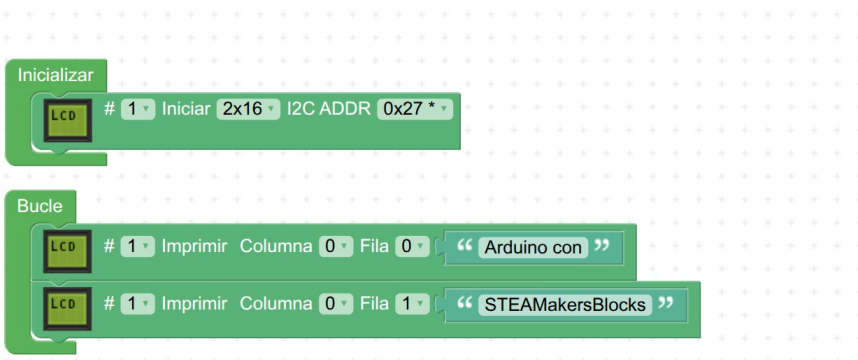


Otra forma de conexión (los dos primeros pines superiores están conectados internamente a los A4 y A5):



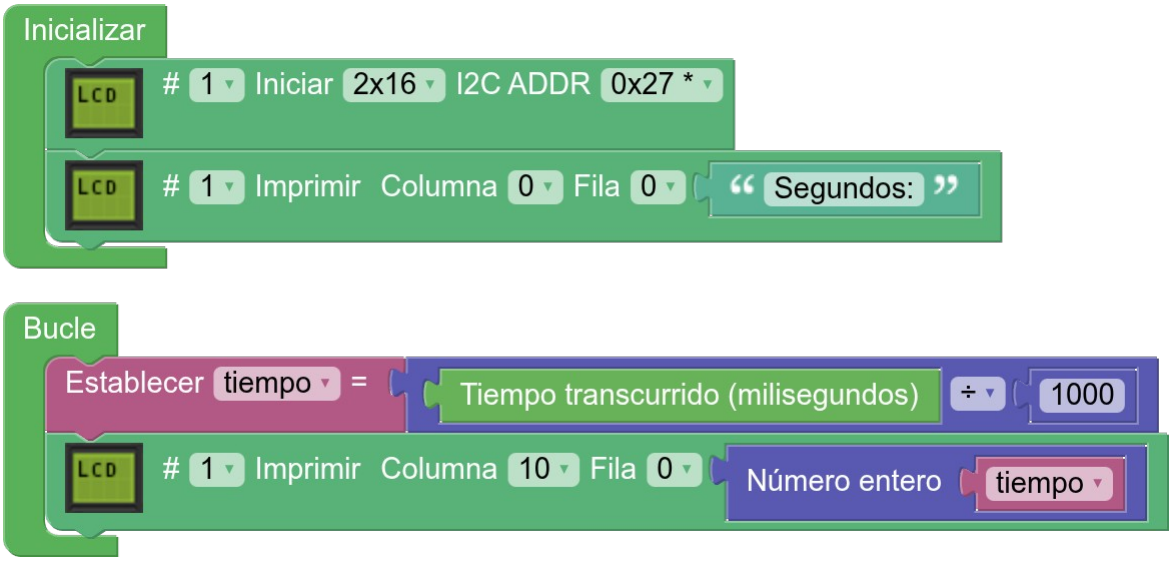
Práctica 17. Muestra el mensaje de la imagen superior:





>>> Si no se ven o se ven mal los caracteres, ajustamos el contraste de la pantalla girando el potenciómetro azul de la parte de atrás:

Práctica 18. Contador de segundos



☑️ TAREA 7. Estación meteorológica.

Realiza una estación meteorológica en la que se pueda visualizar por pantalla la información de la temperatura y humedad, similar a la imagen.

19. Servomotores

▶️ Vídeo 11. Servomotores con STEAMakersBlocks

Un servomotor es un tipo de motor que puede moverse hasta un ángulo exacto y mantenerlo con precisión. A partir través del programa que realizaremos con STEAMakersBlocks, recibirá una señal de Arduino que le indicará a qué ángulo debe girar (por ejemplo, 0°, 90°, 180°).

Servomotor SG90

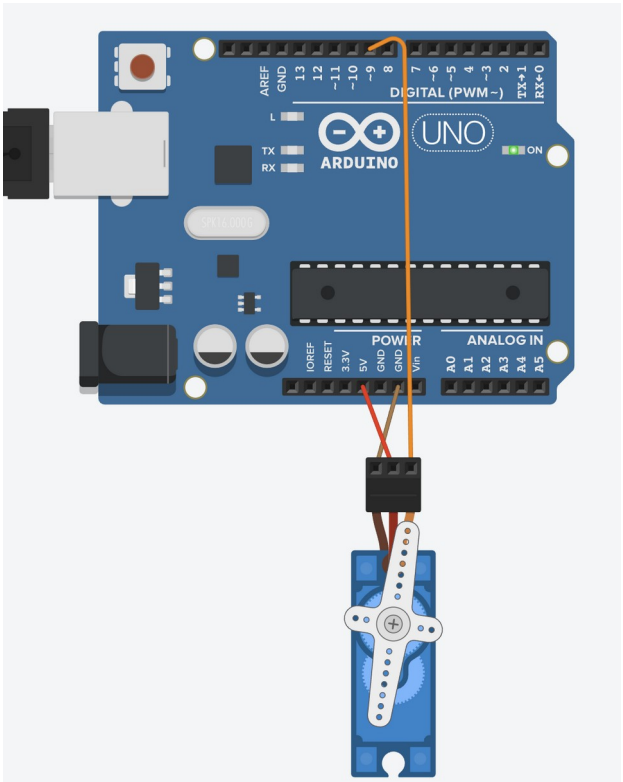
Dentro lleva un motor, un sensor que detecta la posición y un circuito de control. Gracias a eso puede ajustar su movimiento y parar justo donde debe.

El modelo **SG90** (el de la figura) es muy utilizado porque es el más económico, con engranajes de plástico, que no puede transmitir mucha potencia. Puede girar de 0° a 180°, aunque este modelo económico no suele llegar a esos valores extremos, así que, para evitar forzarlo, mejor usarlo en un rango entre 5 y 175 grados.

Servo MG996R

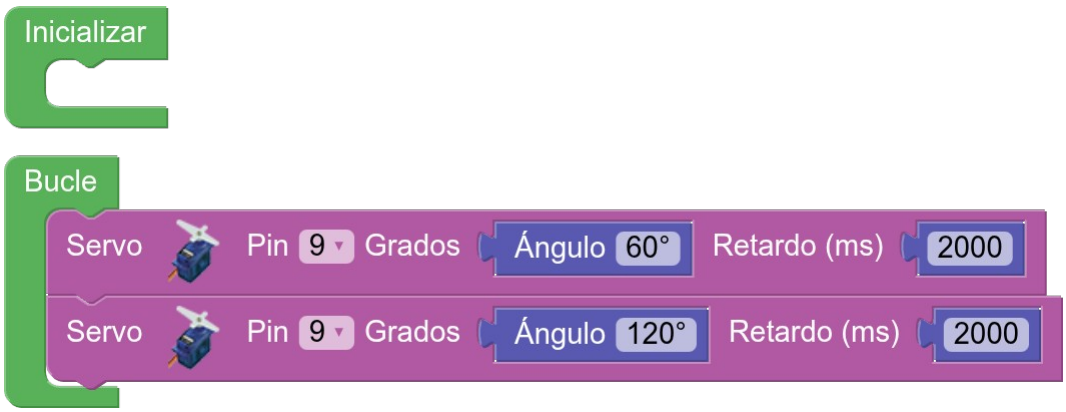
Si necesitamos más torque o par motor y con engranajes metálicos, tenemos otras opciones, como el **MG996R**.

La conexión básica sería como en la imagen, teniendo que conectar el cable naranja a cualquier **salida digital**.

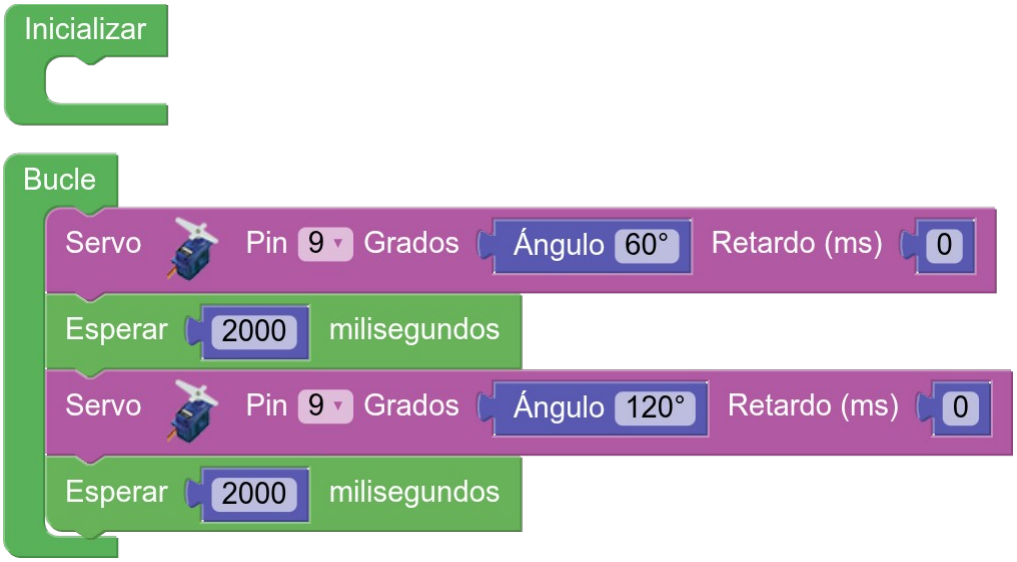


Programa básico en STEAMakersBlocks para un servo conectado al pin 9:

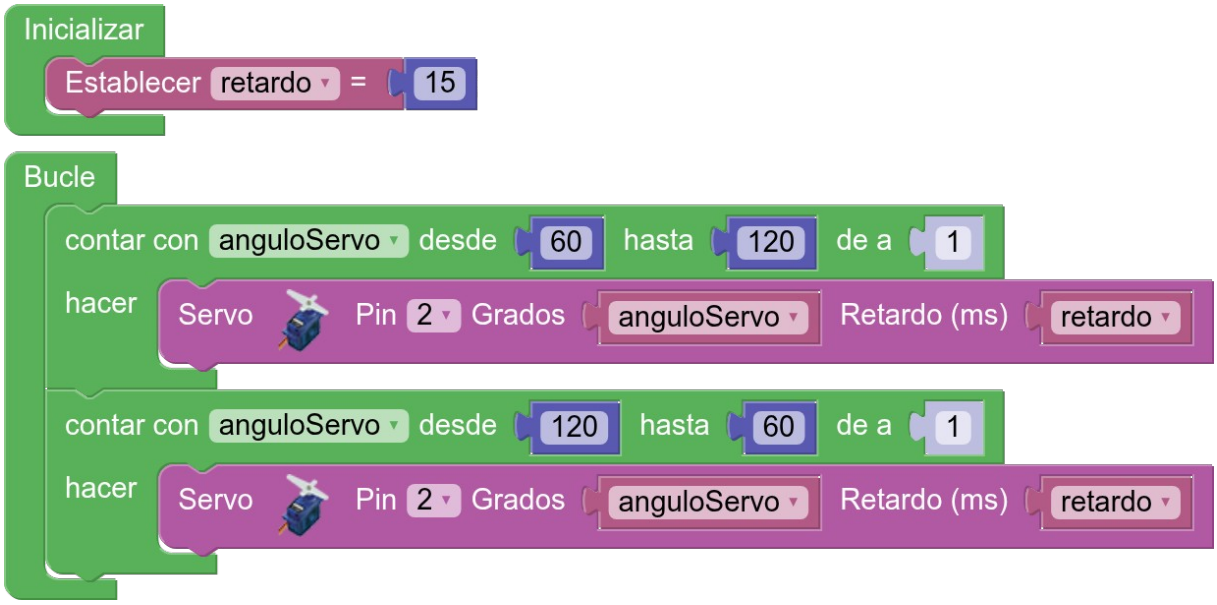
- Coloca el servo en la posición de 60º
- Espera 2 segundos
- Coloca el servo en la posición de 120º
- Espera 2 segundos



El retardo que incorpora el bloque de los servos, se aplicaría después del giro. El programa anterior sería equivalente al siguiente:



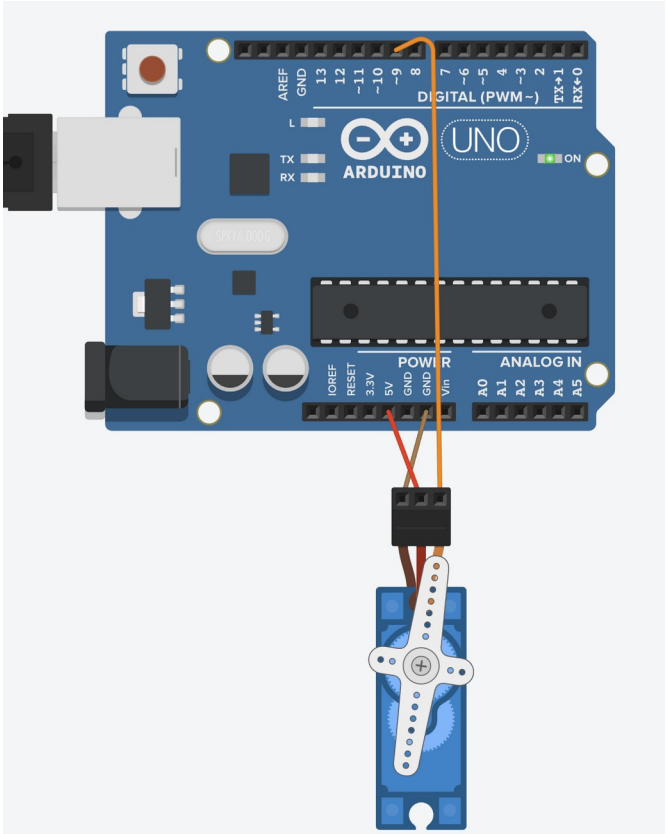
Para una **velocidad de giro más lenta**, podemos usar el bloque contar:



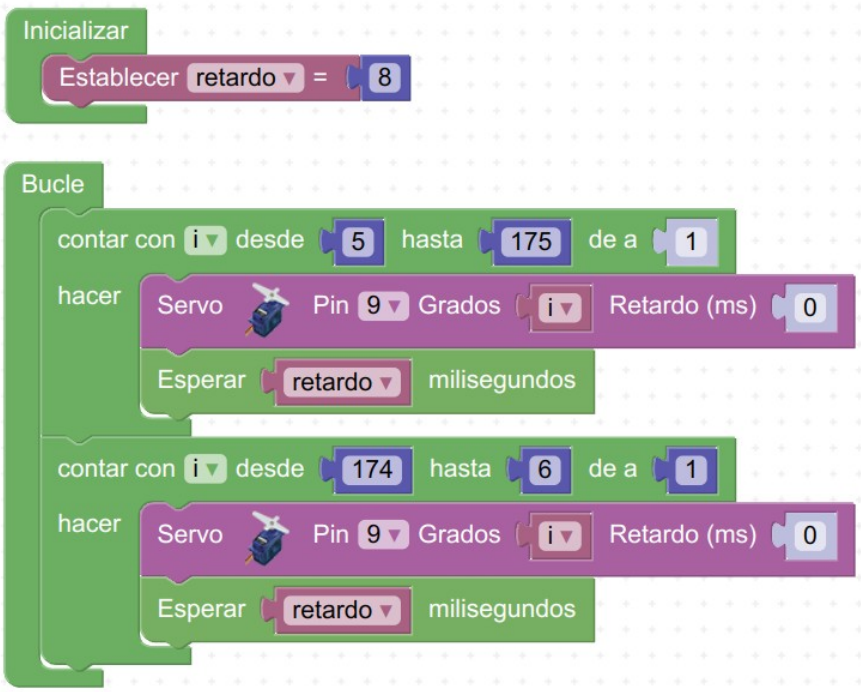
Práctica 19. El limpiaparabrisas

Con el montaje básico de la figura, programa el servo usando el bloque contar para que haga un movimiento de vaivén como el de un limpiaparabrisas de un coche:

- Que vaya aumentando el ángulo de 1 en 1 desde 5º hasta 175º, con una espera de 8 ms entre una posición y la siguiente.
- Que vaya disminuyendo el ángulo de 1 en 1 desde 174º hasta 6º con una espera de 8 ms entre una posición y la siguiente.



El programa sería:



19.2. Servomotores de rotación continua.

> 📝 **Presta atención al comprar un servomotor. Fíjate en la descripción, ya que venden servomotores modificados, indicados con 360°, que funcionan como una motorreductora con giro continuo.**

Un ejemplo de los dos tipos de servomotor a elegir en la misma página del producto:

- **180 grados** indica el rango del servo estándar.
- **360 grados** indica que está modificado para rotación continua.

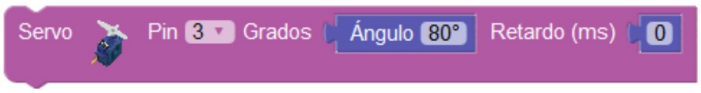
Servomotor estándar a la izquierda y, a la derecha, el mismo servo modificado para funcionar con rotación continua

En este caso, usaremos el mismo bloque de programación, pero la reacción del servo será la siguiente:

0º	Servo Pin 3 Grados Ángulo 0° Retardo (ms) 0	Giro en un sentido (máxima velocidad)
90º	Servo Pin 3 Grados Ángulo 90° Retardo (ms) 0	Parado
180º	Servo Pin 3 Grados Ángulo 180° Retardo (ms) 0	Giro en sentido contrario (máxima velocidad)

Fuente: Juan José López. ArduinoBlocks. Programación visual con bloques para Arduino (2ª ed.)

Si utilizamos valores cercanos a 90° el motor girará a una velocidad más lenta en cada uno de los sentidos.

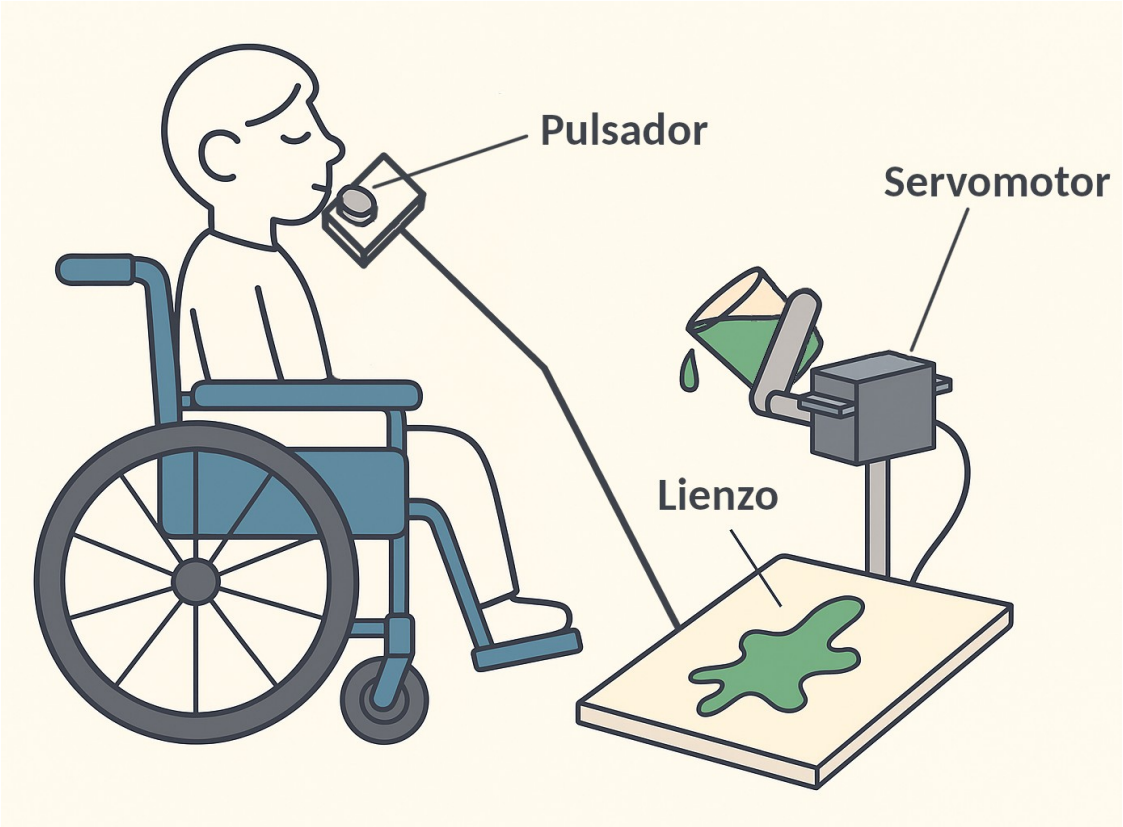
80º		Giro en un sentido (velocidad lenta)
100º		Giro en sentido contrario (velocidad lenta)

Fuente: Juan José López. ArduinoBlocks. Programación visual con bloques para Arduino (2ª ed.)

☑️ TAREA 8. Dispositivo adaptado para pintar

El Departamento de Tecnología de nuestro instituto participa en varios proyectos de aprendizaje-servicio con un colegio de educación especial cercano.

Entre el listado de necesidades planteadas que hemos seleccionado, está la de un dispositivo adaptado que permita a sus alumnos pintar utilizando un pulsador accionado con la barbilla. Al activarlo, un vaso girará para verter la pintura sobre el lienzo.



A modo de ejemplo, nos han mostrado un vídeo de un dispositivo de este tipo de la cuenta de Instagram de “International Academy of Hope” (**ver publicación**).

Debes crear un prototipo del sistema de control para que funcione de la siguiente manera:

- La posición inicial del servo es de 5 grados

- Mientras se presiona el pulsador, el servomotor avanza lentamente hasta un máximo de 175 grados.
- En el momento que se deje de pulsar, cambiará la dirección de giro del servo.

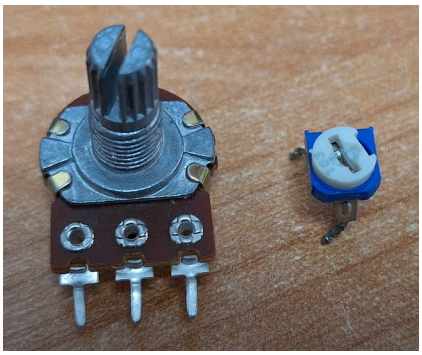
En el siguiente vídeo puedes ver la explicación de cómo hacer el programa:

> ▶️ **Vídeo 11.2. Dispositivo adaptado para pintar. Explicación del programa**

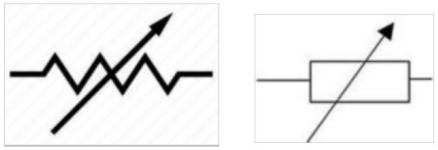
20. Potenciómetro y joystick

> ▶️ **Vídeo 12. Potenciómetro y joystick con STEAMakersBlocks**

20.1. Potenciómetro



Potenciómetro de usuario y técnico



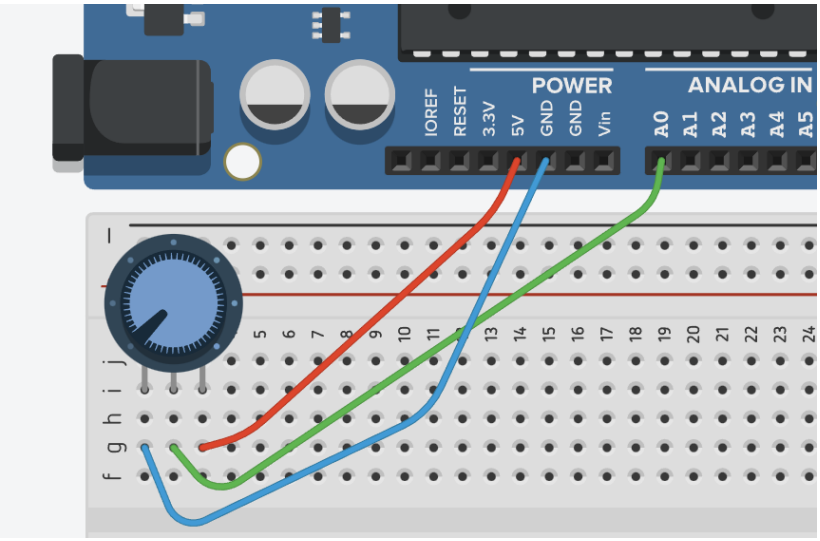
Símbolo del potenciómetro

Los **potenciómetros** son resistencias variables, las ajustamos al hacer girar su mando, aunque también las hay deslizables.

Conexión básica a Arduino:

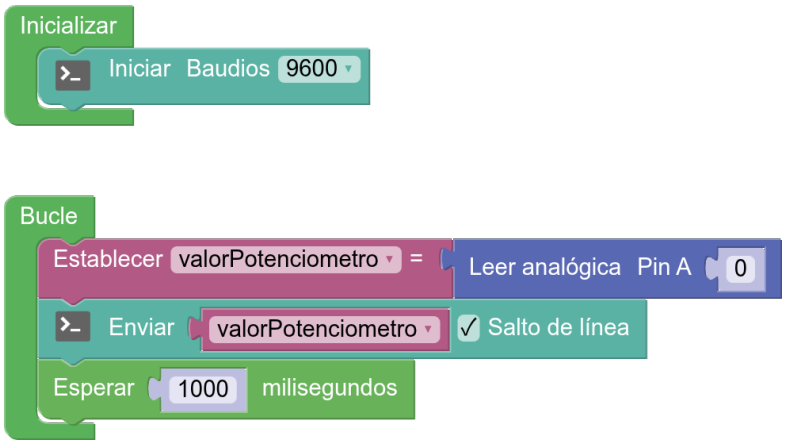
- Los pines de los extremos a GND y 5V de Arduino.

- El pin central a entrada analógica de Arduino, que registra valores entre 0 y 1023



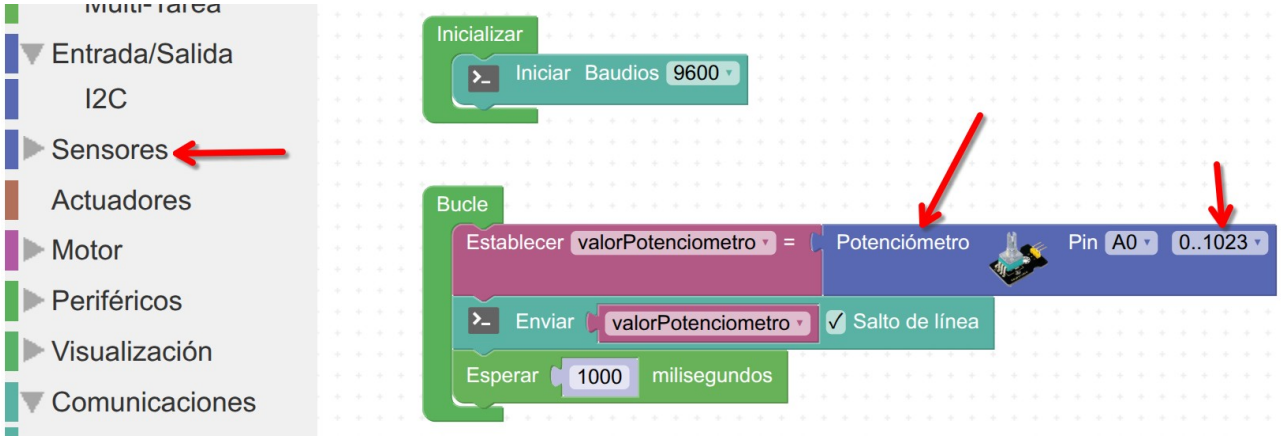
Práctica 20. Leemos el valor de entrada analógica de un potenciómetro.

La instrucción para leer la entrada analógica será igual que la estándar:

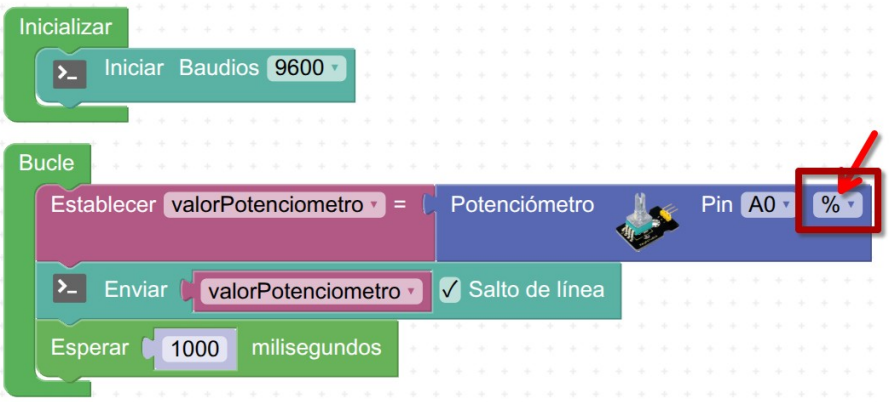


Obteniendo en la consola serie el valor de la entrada analógica entre 0 y 1023.

Pero STEAMakerBlocks tiene bloque específico, con lo que podríamos usar el siguiente programa:



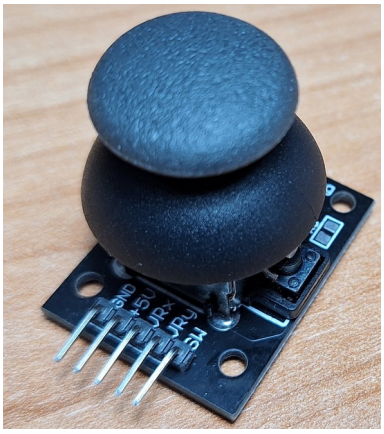
Y nos permite elegir el rango de entrada entre 0 y 1023 ó entre 0 y 100:



20.2. Joystick

El **joystick** integra:

- **Dos potenciómetros** internos colocados a 90°, uno para el eje X y otro para el eje Y.
- **Un pulsador**, presionando la palanca se escucha el "clic".



Joystick

Descripción de los pines:

- **GND:** Se conecta al pin GND del Arduino.
- **5V:** Se conecta al pin de 5V del Arduino.
- **VRx:** Es como el pin central del potenciómetro, entrega un voltaje proporcional al movimiento en el **eje X** (izquierda–derecha). Se conecta a una entrada analógica del Arduino (A0, A1, etc.).
- **VRy:** Entrega un voltaje proporcional al movimiento en el **eje Y** (arriba–abajo). También se conecta a otra entrada analógica.
- **SW:** Es la salida del **interruptor** que se activa al presionar el joystick hacia abajo. Funciona como una entrada digital (HIGH/LOW).

Por tanto, se programa igual que los potenciómetros y el pulsador, pero STEAMakersBlocks nos incluye un bloque específico también:

21. Mapear rangos de valores en STEAMakersBlocks

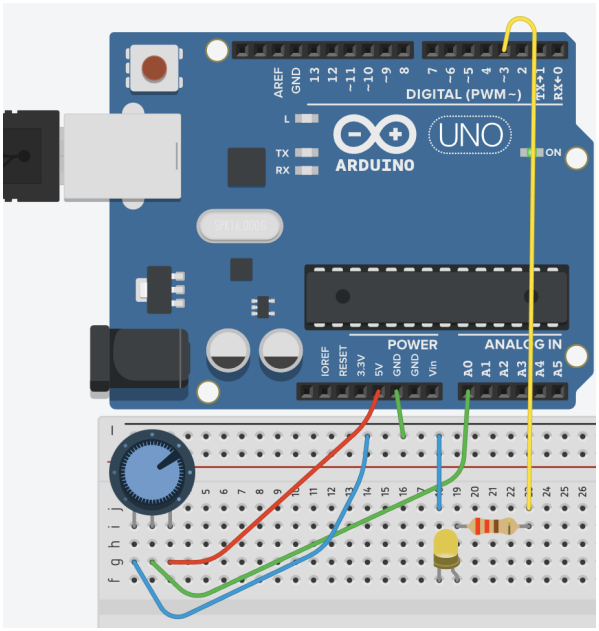
▶️ Vídeo 13. Mapear rangos de valores en STEAMakersBlocks

El bloque **mapear**, dentro de la categoría Matemáticas, permite remapear un número desde un rango a otro. Es útil, por ejemplo, para remapear una lectura analógica que está en un rango de 0-1023 a un nivel de salida PWM que está en un rango 0-255.

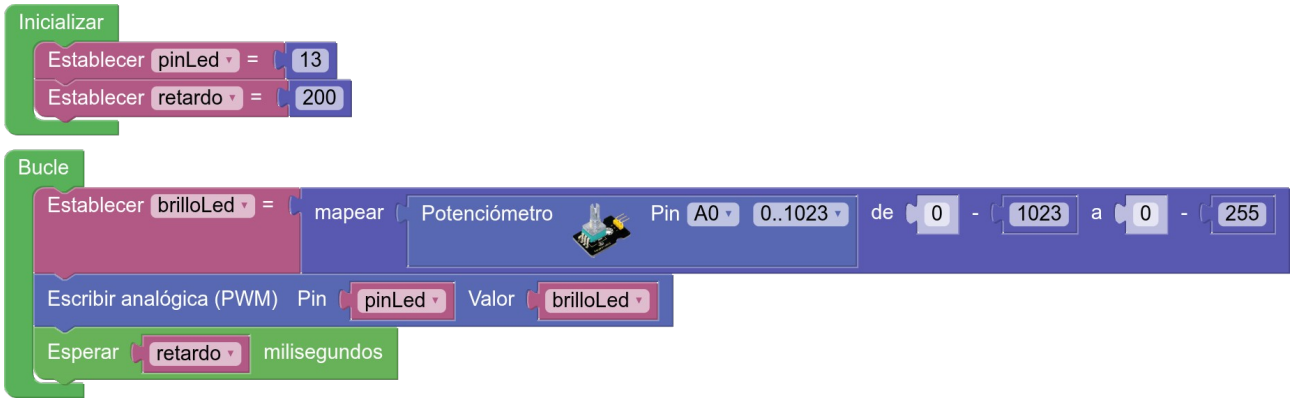
Lo vemos en el siguiente ejemplo:

Práctica 21. Regulador de luz con un potenciómetro.

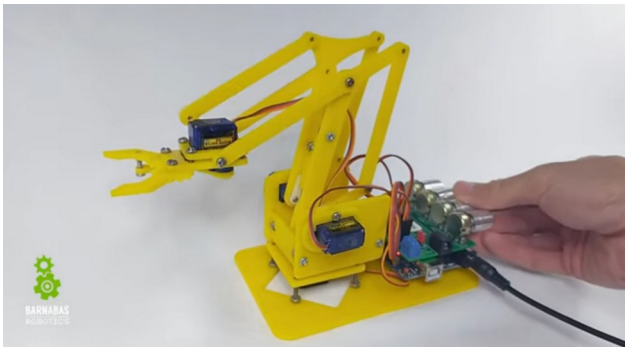
Remapeamos el valor de entrada del potenciómetro (rango de 0 a 1023) al rango de salida PWM (de 0 a 255) para regular el nivel de brillo del LED con el potenciómetro.



El programa sería:

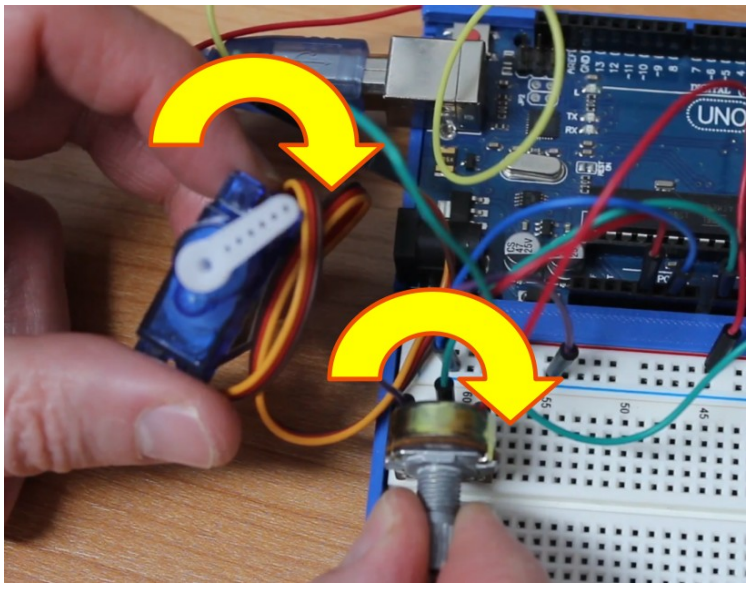


☑️ TAREA 9. Programar un brazo robótico.



La finalidad de esta tarea es programar el control básico entre un potenciómetro y un servomotor, de manera que, en una fase posterior, puedas replicar el mismo procedimiento para cada uno de los cuatro servos y sus correspondientes potenciómetros que conforman un brazo robótico como el que se muestra a continuación (haz clic en la imagen para ver el vídeo):

Deberás realizar el prototipo en el que un potenciómetro se usa para dirigir un servomotor:



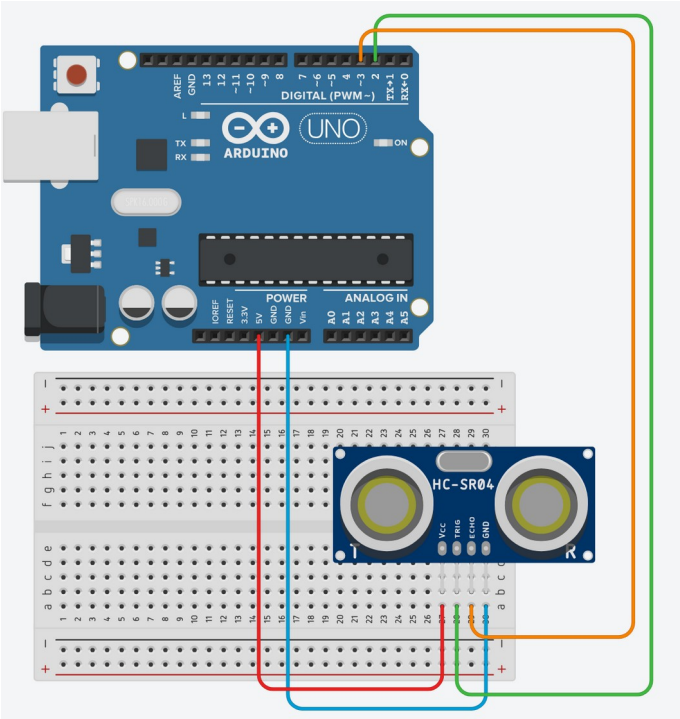
22. Sensor de distancia ultrasónico HC-SR04

▶️ Vídeo 14. Sensor distancia en Arduino con STEAMakersBlocks

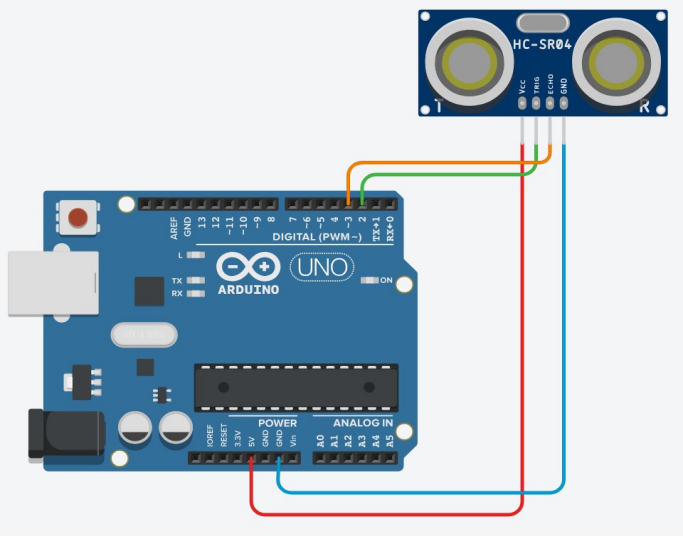
Los sensores **HC-SR04** utilizan ondas ultrasónicas para determinar la distancia entre el sensor y un objeto. Mide con una precisión aceptable entre 2 y 400 cm

Las patillas TRIG (trigguer o disparador) y ECHO (eco) pueden ir a **cualquier pin digital** de Arduino.

Dos formas básicas de montaje:

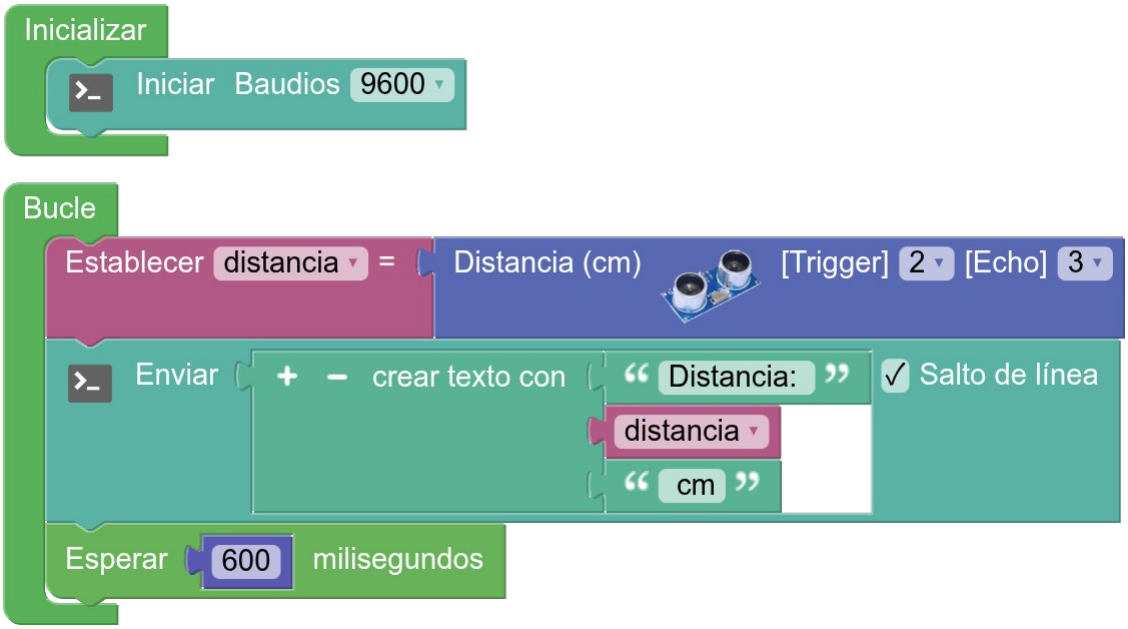


Con protoboard y cables macho-macho



Directamente con 4 cables macho-hembra

Práctica 22. Mostrar el valor de la distancia en la consola serie.



El **resultado** mostrado en la consola serie sería algo así.

☑️ Tarea 10. Proyecto ganador Premios Sapiència 2024

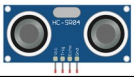
Con lo que hemos visto a lo largo de este curso, vamos a poder realizar **el prototipo del proyecto que ganó el Primer Premio Sapiència 2024.** Este importante primer premio otorga:

- **4.000 € para el alumnado** ganador.
- **1.000 € para el tutor o tutora** del proyecto.

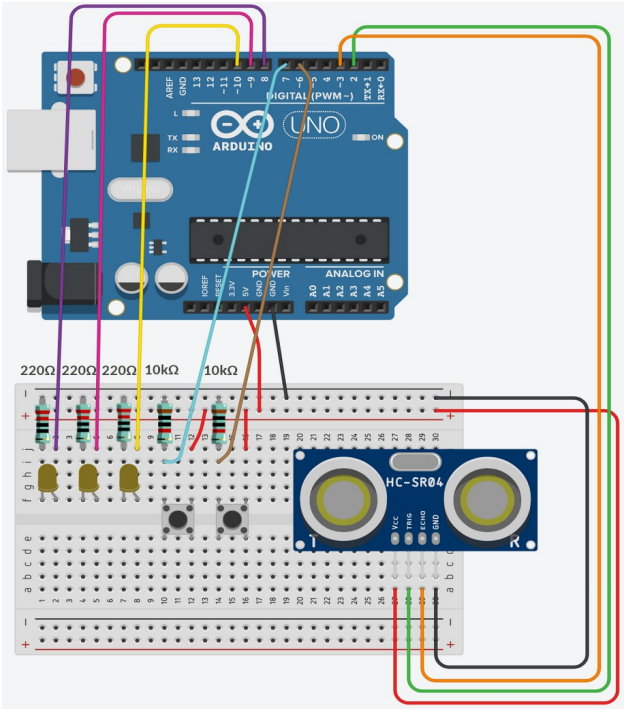
El proyecto, que realizó un equipo del **IES NIT DE L'ALBA de Elche,** consiste en un **chaleco inteligente salvavidas** en seguridad vial con intermitentes integrados y luces de emergencia, para así evitar accidentes con vehículos de movilidad personal (VMP), como son los patinetes eléctricos, bicicletas, u otros vehículos que carecen de sistema de aviso de cambio de dirección (fuente: **memoria del proyecto**).

▶️ Vídeo de la presentación del proyecto

Para el prototipo de esta tarea, usaremos los siguientes componentes:

	3 LEDs para las señales de intermitentes y de alerta de proximidad
	**3 resistencias de 220Ω** para proteger los LEDs
	**2 pulsadores** para accionamiento de los intermitentes.
	**2 resistencias de 10kΩ** para los pulsadores.
	**1 sensor HC-SR04**

El montaje eléctrico del prototipo puede ser como el siguiente:



El sistema deberá funcionar de la siguiente manera:

- **Si presionamos el pulsador izquierdo**, se ejecutará una función "intermitenteIzquierdo" realizando una animación luminosa de intermitente hacia la izquierda.
- **Si presionamos el pulsador derecho**, se ejecutará una función "intermitenteDerecho" realizando una animación luminosa de intermitente hacia la derecha.
- **Si un objeto se acerca** a una distancia de prueba de, por ejemplo, 8 cm, se ejecutará la función "alerta" que activará la intermitencia de los 3 LEDs simultáneamente.

Las instrucciones para realizar el programa las puedes ver en el siguiente videotutorial:

> ▶️ **Vídeo 15. Prototipo de chaleco salvavidas inteligente**

23. Enlaces con recursos y proyectos

En los siguientes enlaces puedes encontrar enlaces con documentación y proyectos:

- **Libros y documentación de STEAMakersBlocks**
- **Guía para el profesor. Usuarios gestionados**
- **Fichas en PDF de Robolot (catalán y castellano)**, disponibles a través de **Innova Didactic**.
- **Proyectos de la comunidad educativa. Innova Didactic**

24. Ideas para posibles proyectos con lo visto en el curso.

24.1. Semáforo de coches y peatones

Vamos a hacer una maqueta en el que hay un semáforo de coches y otro de peatones.

Debes programarlo para que cuando el semáforo de coches esté en rojo se ponga en verde el de peatones y viceversa.

Por seguridad:

- Habrá un pequeño tiempo de espera entre esos cambios (entre el rojo de los coches y el verde de los peatones, así como entre el rojo de los peatones y el verde de los coches).
- El semáforo verde de los peatones parpadeará antes de cambiar a rojo.

Componentes necesarios:

- LEDs
- Resistencias de 220 Ω

24.2. Encendido automático por movimiento.

Diseña y construye un sistema de encendido automático de iluminación que se active de forma autónoma cuando una persona o vehículo pase por delante.

Podría ser de utilidad en el pasillo de un edificio o en las farolas de una calle al pasar por delante vehículos o personas.

Para ello, deberás utilizar un sensor de distancia ultrasónico (HC-SR04) capaz de detectar la presencia mediante la medición del cambio en la distancia frente al sensor.

Componentes necesarios:

- LEDs
- Resistencias de 220 Ω
- Sensor de distancia HC-SR04

24.3. Cronómetro.

Diseña y construye un cronómetro que muestre en pantalla segundos, minutos y horas desde el encendido.

Componentes necesarios:

- LCD 1602

24.4. Alarma por alta temperatura.

Realiza un sistema que nos de la señal de alarma mediante parpadeo de señal luminosa roja si la temperatura supera un valor de temperatura visualizado en un display LCD.

En el display se mostraría:

- Temperatura real
- Temperatura límite.

Componentes necesarios:

- LCD 1602
- Sensor de temperatura y humedad DHT11
- LED y resistencia de 220Ω

24.5. Alarma por alta temperatura con límite regulable.

Realiza un sistema que nos de la señal de alarma mediante parpadeo de señal luminosa roja si la temperatura supera un valor de temperatura ajustado por nosotros con un potenciómetro y visualizado en un display LCD.

En el display se mostraría:

- Temperatura real
- Temperatura límite (que hemos prefijado con el potenciómetro)

Componentes necesarios:

- LCD 1602
- Sensor de temperatura y humedad DHT11
- Potenciómetro
- LED y resistencia de 220Ω

24.6. Medidor de distancia.

Diseña y construye un dispositivo capaz de mostrar en un display los centímetros de distancia hasta un elemento.

Componentes necesarios:

- LCD 1602
- Sensor de distancia HC-SR04

24.7. Sensor de aparcamiento

Diseña un sistema que nos avise de la distancia detectada hasta un objeto que se encuentre delante o detrás de nuestro vehículo.

Debe parpadear más rápido conforme nos acercamos al objeto.

Componentes necesarios:

- LED y resistencia de 220Ω
- Sensor de distancia HC-SR04

24.8. Fotómetro digital

Diseña y construye un fotómetro que muestre el valor de luz ambiental por pantalla.

Componentes necesarios:

- LDR y resistencia de 10kΩ
- Pantalla LCD1602

24.9. Indicador de aparcamiento libre

Diseña y construye un sistema indicador luminoso para un aparcamiento de un centro comercial que muestre si cada plaza está libre u ocupada.

Sobre cada plaza del aparcamiento habrá un indicador luminoso con un led rojo y otro verde:

- Si la plaza está ocupada por un coche, el indicador luminoso se encenderá rojo
- Si la plaza está libre, el indicador se encenderá verde.

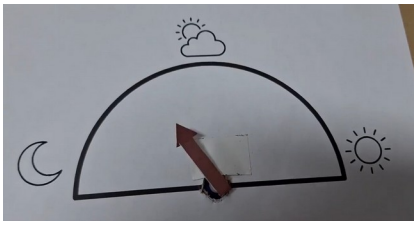
Se diseñará con sensor de luz (LDR) incrustado en el suelo de la plaza de aparcamiento. Al estar ocupada por un coche, registrará menos nivel lumínico.

Componentes necesarios:

- LED rojo, verde y resistencias de 220Ω.
- LDR y resistencia de 10kΩ

24.10. Fotómetro analógico.

Diseña y construye un fotómetro que muestre el valor de luz ambiental de forma analógica.



Clic en la imagen para ver el vídeo del funcionamiento

Componentes necesarios:

- LDR y resistencia de 10kΩ
- Servomotor

24.11. Termómetro analógico.

Diseña y construye un termómetro que muestre el valor de la temperatura ambiental de forma analógica.

Componentes necesarios:

- Sensor de temperatura y humedad DHT11
- Servomotor

24.12. Barrera automática.

Diseña un sistema que levante la barrera de entrada a un aparcamiento cuando detecte vehículo delante.

Componentes necesarios:

- Servomotor
- Sensor de distancia HC-SR04

24.13. Papelera inteligente.

Diseña una papelera que abra automáticamente la tapa cuando alguien se ubica delante.

Componentes necesarios:

- Servomotor
- Sensor de distancia HC-SR04

24.14. Puerta automática.

Diseña una puerta que abra automáticamente cuando alguien se aproxima.

Componentes necesarios:

- Servomotor
- Sensor de distancia HC-SR04

25. Fuentes de información

- López Almendros, J. J. (2017). ArduinoBlocks: Programación visual con bloques para Arduino (2ª ed.). CreateSpace Independent Publishing Platform.
- STEAMakersBlocks. (2026). Documentación oficial. Recuperado el 13 de enero de 2026, de https://www.steamakersblocks.com/web/site/doc
- Arduino Uno. (2026). En Wikipedia, La enciclopedia libre. Recuperado el 20 de diciembre de 2025, de **https://es.wikipedia.org/wiki/Arduino_Uno**.
- Pérez E., J. (s.f.). Johann Perez E – Roboteg [Canal de YouTube]. YouTube. **https://www.youtube.com/c/JohannPerezE**