Programación visual de robots y *gadgets*

PID_00249858

Marco Antonio Rodríguez Fernández

Universitat Oberta de Catalunya

Índice

Introducción

Scratch4Arduino, a partir de ahora S4A (accesible en línea), es un desarrollo del Citilab de Barcelona en colaboración con el Instituto Tecnológico de Massachusetts (MIT). Este software es una modificación de la versión 1.4 del Scratch original, al que se han añadido toda una serie de bloques que permiten que nos conectemos con la placa de Arduino. De esta forma se unen dos proyectos de software y hardware libre con vocación educativa.

¿Qué es Scratch4Arduino y Snap4Arduino?

Véase la página web de Arduino (accesible en línea).

Como ya sabéis, las placas microcontroladoras de Arduino tienen muchas posibilidades. Aun así, el lenguaje de programación que utilizan puede ser un poco complejo si somos nuevos en el ámbito de la programación. S4A es mucho más sencillo y nos ofrece la posibilidad de combinar la programación de juegos con Scratch y la interacción con el exterior utilizando Arduino.

Figura 1
Arduino

Para trabajar con S4A tenemos que cargar un *firmware* desde el programa de Arduino en cualquier placa de Arduino. A partir de ese momento, la placa se comunicará todo el tiempo con el software de S4A. Usaremos el *firmware* específico, que podemos descargar desde s4a.cat. El *firmware* en el caso de Snap4Arduino es StandardFirmata, que viene como ejemplo en el mismo software de Arduino.

Figura 2
Snap4Arduino

Snap4Arduino es la evolución de S4A. Para este proyecto, Bernat Romagosa utiliza como base Snap! –la versión de Scratch que implementó la universidad de Berkeley– para diseñar una versión nueva de S4A mucho más estable y versátil, y que utiliza el *firmware* StanrdardFirmata.

Figura 3
S4A

Durante este curso, tanto los contenidos audiovisuales como las capturas de pantalla realizadas con el software Fritzing harán referencia a Snap4Arduino, aunque son perfectamente compatibles con la versión de S4A anterior.

1. Instalación

En primer lugar vamos a ver cómo instalar S4A, y a continuación veremos cómo se instala Snap4Arduino.

1.1. Cómo instalar S4A

El proceso de instalación de S4A es relativamente fácil. Consta de dos partes:

1) En primer lugar, cargamos el *firmware* en la placa de Arduino. Para ello descargamos el archivo *firmware* que encontramos en la sección «Downloads» de la web de s4a.cat (accesible en línea). Una vez descargado, debemos abrir Arduino –podemos encontrarlo en «Arduino» / «Software» (accesible en línea)–, seleccionar el modelo de placa de Arduino que tengamos y el puerto donde está conectada (COM3, COM4, ...), y pulsar el botón para cargar el *firmware*.

2) El paso siguiente es instalar propiamente el programa S4A, que encontraremos en la misma sección de descarga de s4a.cat (accesible en línea).

Al iniciar S4A tras la instalación, empieza la búsqueda de la placa. En el momento en que conectamos la placa, S4A la reconoce y ya podemos empezar a programar.

Vídeo 01. Conexión de la placa con Snap4Arduino

Descripción del procedimiento de conexión y comunicación de una placa de Arduino –o compatible– con el software Snap4Arduino.

1.2. Cómo instalar Snap4Arduino

La instalación de Snap4Arduino es idéntica a la de S4A. La única diferencia es que el *firmware* que cargamos en la placa es StandardFirmata. Podemos encontrar este *firmware* directamente en los ejemplos que se preinstalan al instalar el software de Arduino.

Para cargar el *firmware*, abrimos la aplicación de Arduino y pulsamos «Archivo» / «Ejemplos» / «Firmata» / «StandardFirmata». Seleccionamos el modelo de placa y el puerto, y pulsamos en el botón de cargar en la placa.

Una vez cargado el *firmware*, ya podemos descargar Snap4Arduino de la sección «Download» de la página web de Snap4Arduino (accesible en línea). Al ejecutarlo tras la instalación, buscará la placa de manera automática.

Vídeo 02. Conexión de la placa con S4A

Descripción del procedimiento de conexión y comunicación de una placa de Arduino –o compatible– con el software S4A.

2. Recursos

En los recursos (REC) vamos a aprender todo lo que necesitamos saber para poder realizar todos los proyectos que queramos, desde la conexión de los componentes hasta cómo programarlos utilizando Snap4Arduino.

2.1. Conexión de la placa de Arduino

Como ya hemos dicho, en este módulo utilizaremos el software Snap4Arduino para llevar a cabo las prácticas. Todos los ejemplos harán referencia a este software, aunque el modo de conectar los componentes será común para cualquier software, como por ejemplo, arduino.cc.

2.1.1. Conexión de la placa con el software Snap4Arduino

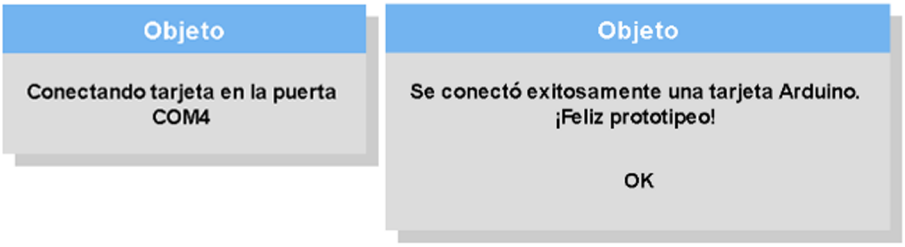
En primer lugar tenemos que asegurarnos de que nuestra placa de Arduino está conectada al PC en alguno de los puertos USB. En Snap4Arduino, la conexión se realiza desde el apartado «Arduino» (figura 4).

Figura 4

Conexión de la placa

Una vez seleccionado el apartado Arduino, hacemos clic sobre la opción Conectar Arduino. El programa se pondrá a buscar nuestra placa automáticamente hasta encontrarla (figura 5).

Figura 5



Es posible que la opción «Conectar Arduino» nos permita escoger entre varios puertos COM. Esto pasará si tenemos más cosas conectadas al ordenador por el puerto USB. Para saber qué puerto COM elegir, tenemos que fijarnos simplemente en el puerto que aparece al conectar la placa.

Una función muy interesante de Snap4Arduino es el bloque *conectar a puerto*. Si sabemos cuál es el puerto COM de la placa, podemos hacer un pequeño programa en el que la placa se nos conecte de manera automática al pulsar una tecla concreta. Esto es especialmente interesante si la placa se desconecta con frecuencia o si planeamos conectar y desconectar la placa de modo recurrente.

Vídeo 03. Conexión de la placa con mBlock

Descripción del procedimiento de conexión y comunicación de una placa de Arduino –o compatible– con el software mBlock.

2.1.2. Conexión de la placa con el software S4A

El proceso en S4A es muy parecido. Simplemente, la búsqueda de la placa es automática. Al encender el programa, se mostrará una ventana con el mensaje «Buscando placa», que desaparecerá una vez la encuentre. La búsqueda es automática gracias al *firmware* para S4A que instalamos en la placa, aunque dejamos de tener tanto control a la hora de conectar y desconectar las placas de Arduino.

2.2. Introducción a la programación por bloques

En Snap4Arduino se programa de manera muy parecida a la de Scratch. Si estamos familiarizados con los bloques de Scratch nos será muy fácil acostumbrarnos. La única diferencia es que se añade la compatibilidad con Arduino y permite la interacción con elementos físicos.

Vamos a hacer un repaso de los diferentes bloques.

2.2.1. Eventos

En el apartado de «Control» encontramos todos los detectores de eventos (figura 6). Estos son los bloques que nos permiten empezar nuestros programas.

Figura 6

Pueden detectarse varios tipos de eventos (figura 7):

Figura 7

- Cuando se presiona , el icono de inicio de programa.
- Cuando se presiona una tecla en concreto. Esto sirve, por ejemplo, para hacer que algo se mueva hacia la izquierda o hacia la derecha con las flechas del teclado.
- Cuando se hace clic sobre un personaje.

2.2.2. Bucles

Podemos usar los bucles si queremos que nuestro programa o alguna parte de nuestro programa se repita. Estos bloques (figura 8) nos permiten hacer que el programa repita todo el rato aquello que le pedimos, en vez de hacerlo una sola vez.

Figura 8

Con el bloque *por siempre* podemos hacer que el bucle se repita hasta que decidamos detener el programa.

Con el bloque *repetir_* podemos indicar cuántas veces queremos que se repita el bucle.

También podemos usar el bloque *repetir hasta que* para hacer que el bucle se repita hasta que se cumpla una condición. Por ejemplo, hasta que se acabe el tiempo o se llegue a una cantidad de puntos determinada.

2.2.3. Condicionales

Los condicionales (figura 9) sirven para que un bloque o un conjunto de bloques se ejecuten solo si pasa alguna cosa concreta, como por ejemplo, que un personaje desaparezca cuando un enemigo le toca (figura 10), o que gire a la derecha cuando el sensor de distancia detecte algo.

Figura 9

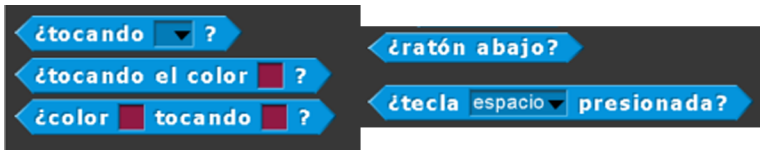
Hay dos versiones de condicionales: el *si* y el *si... si no*. Con el *si*, podemos hacer que se ejecute alguna acción cuando la condición es cierta. Con el *si... si no* podemos hacer que una acción se ejecute cuando la condición sea cierta, y otra acción cuando la condición sea falsa.

Figura 10

2.2.4. Sensores

Podemos usar sensores como condiciones y ponerlos dentro de las instrucciones condicionales. Tenemos los sensores típicos de Scratch, que detectan si se tocan objetos o colores, y en la sección «Arduino» tenemos también el sensor lectura digital, que será cierto cuando lea un valor positivo en la entrada que le indiquemos.

Figura 11



Snap! añade a estos sensores la posibilidad de consultar sitios web con http:// snap.berkeley.edu o consultar datos sobre la fecha y la hora con current date.

2.2.5. Variables y listas

Las variables y las listas se utilizan para almacenar datos en memoria. Esto nos sirve, por ejemplo, cuando queremos guardar la puntuación de un juego o el valor de un sensor. Las listas pueden almacenar muchos valores, en vez de almacenar uno solo.

En esta sección encontraremos algunas particularidades de Snap! que no se encuentran en Scratch, como variables de programa a, que sirve para crear variables en tiempo de ejecución.

Las particularidades de Snap!

Estas mejoras de Snap! logran una implementación orientada a objetos mucho más completa que en Scratch.

2.2.6. Operaciones

Los operadores son el conjunto de instrucciones que nos permiten trabajar con diferentes elementos matemáticos, lógicos y de texto.

Figura 12. Diversos grupos de operadores

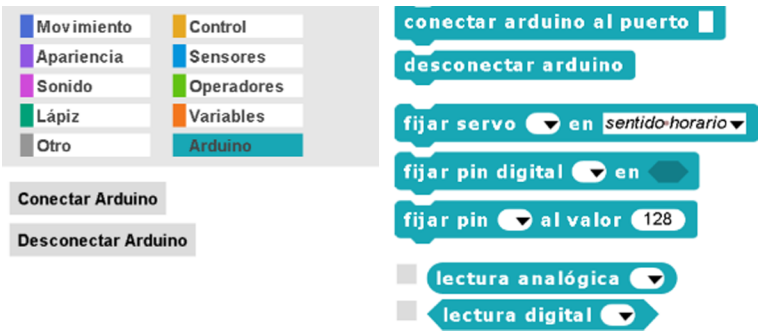
2.3. Entradas y salidas (I/O)

Las placas **Arduino** nos permiten elaborar proyectos muy interesantes utilizando unos conceptos muy sencillos: el de **entrada** y el de **salida**.

> Las **entradas** se pueden «leer» para obtener información de sensores, como pulsadores o sensores de luz.
>
> Las **salidas** pueden «escribir/fijar» valores para activar actuadores, como motores o luces.

En **Snap4Arduino** encontraremos todos los bloques necesarios para controlar los elementos de la placa de Arduino. Estos bloques se encuentran en el apartado Arduino.

Figura 13



Hay dos tipos de valores con los que podemos trabajar en las entradas y salidas de Arduino; **digitales** y **analógicos**:

- Las **entradas/salidas digitales** permiten tener la información de **encendido** ( ) o **apagado** ( ), sin niveles de intensidad.

- En cambio, las **entradas/salidas analógicas** (0 , 128 , 1024) nos dejan elegir diferentes niveles de intensidad (de 0 a 1.023 para las entradas, y de 0 a 255 para las salidas). Así tenemos una mejor precisión a la hora de controlar cosas como la intensidad de una luz o la velocidad de un motor.

2.3.1. Salidas digitales y analógicas

Usaremos **salidas digitales** para, por ejemplo, **encender** y **apagar** un led (figura 14).

Figura 14

Podemos elegir como **salida digital** cualquiera de los pines, **desde el 2 hasta el 13** de nuestro Arduino (figura 15), y darle un valor de «Encendido» o «Apagado» usando la instrucción: «Fijar pin digital». No es recomendable seleccionar las salidas digitales 0 y 1 porque son las que utiliza el puerto serie para que Snap! se comunique con la placa.

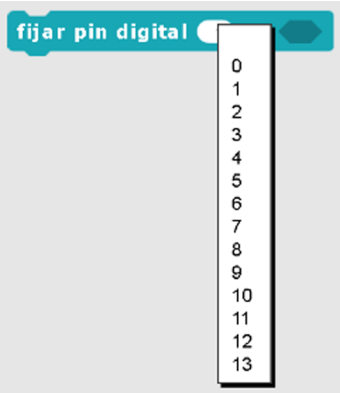


Figura 15

Cuando queremos **decidir la intensidad** de la luz del led, podemos usar las **salidas analógicas**. Con un valor del 0 al 255, podremos elegir si queremos que brille más o menos:

- Cuanto **menor** sea el valor que pongamos, **menos luz** dará el led.

- Cuanto **mayor** sea el valor que pongamos, **más luz** dará el led.

Podemos usar salidas analógicas para todo tipo de actuadores; pero ¡cuidado!, algunos usan rangos distintos.

Solo podemos usar como salidas analógicas los **pines 3, 5, 6, 9, 10 y 11** de Arduino (figura 16). Tendremos que asegurarnos de que usamos uno de estos pines.

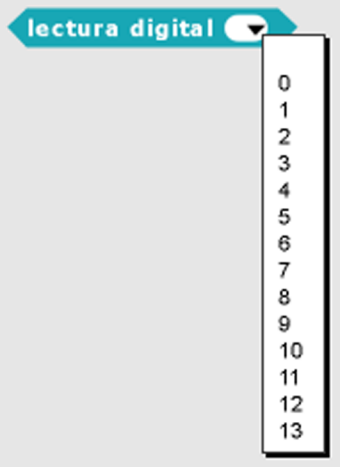
Figura 16

2.3.2. Leer un sensor digital

Si conectamos sensores a la placa, podemos leer sus valores utilizando los bloques de lectura.

Usamos la **lectura digital** para **sensores digitales**, los que detectan si algo está «Encendido» o «Apagado». Podemos hacer la lectura digital de cualquiera de los pines **desde el 0 hasta el 13** (figura 17).

Figura 17



El resultado de esta lectura será «cierto» o «falso». Por lo tanto, podremos hacer construcciones condicionales usando un bloque *si* _ para comprobar si un sensor está activado o no, como mostramos a continuación:

- Vamos a conectar un sensor. Por ejemplo, un **pulsador** en el **pin número 3** de la placa de Arduino (figura 18).

Figura 18

- Cuando se pulsa el **pulsador**, se ejecutará el código que esté dentro del condicional ***si*** _. Pero, ¡ojo!, para que un programa detecte en cualquier momento si el botón se ha pulsado, tendremos que hacer una **estructura de bucle** para que esté comprobando constantemente si el botón se ha pulsado (figura 19).

Figura 19

¡Truco!

También se puede usar una estructura como esta, en la que se registra el evento *cuando ocurra* _, que esperará hasta que el sensor se active.

Figura 20

2.3.3. Leer un sensor analógico

Utilizaremos la lectura analógica si queremos alcanzar una mayor precisión en los valores de un sensor (figura 21). Esta lectura nos retornará un valor de 0 a 1.023.

Figura 21

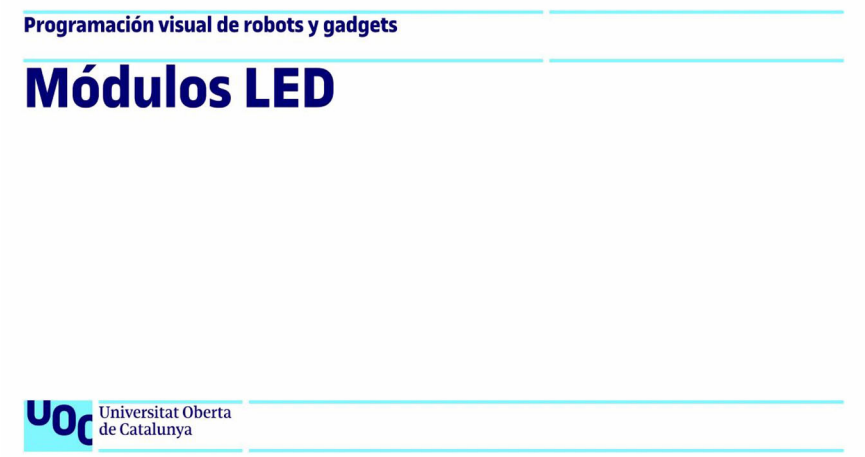
2.4. Actuadores

En este subapartado echamos un vistazo a diversos tipos de actuadores:

- Leds, módulos LED y diodos láser
- Motores de tipo servo
- Motores de tipo CC
- Zumbadores
- Indicadores de siete segmentos

2.4.1. Leds, módulos LED y diodos láser

Vídeo 04. Módulos LED

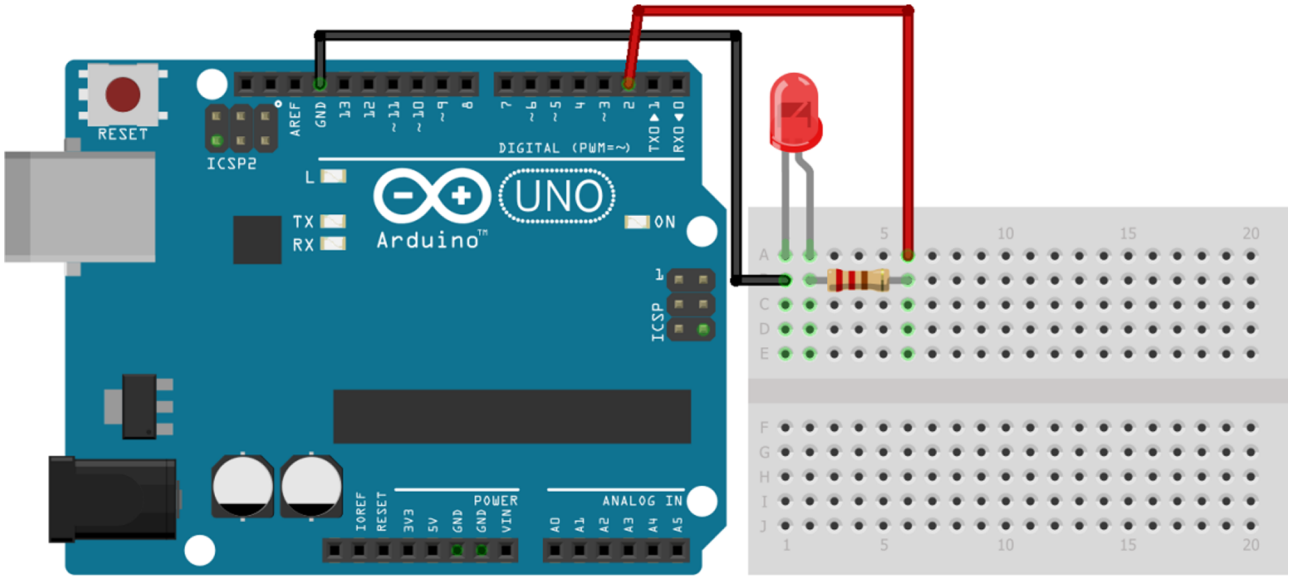


Descripción del procedimiento de conexión de diferentes módulos LED a la placa de Arduino, y de cómo controlarlos utilizando el software de Snap4Arduino.

Leds

Normalmente conectaremos la pata más corta del led a GND y la pata más larga a alguna de las salidas de Arduino. Pero, ¡ojo!, la salida de Arduino funciona a 5 V y los leds acostumbran a funcionar entre 2,5 V ~ 3,5 V, así que necesitaremos poner delante una resistencia de unos 200-300 Ω.

Figura 22



Módulos LED

Otra cosa que podemos hacer es usar módulos LED con la resistencia ya incluida (figura 23).

Figura 23
Módulo LED RGB.

Para conectar un módulo LED y controlarlo desde Snap4Arduino, primero nos tenemos que asegurar de que la placa esté conectada al ordenador por un cable USB, y también dentro del programa.

A continuación, conectamos el módulo LED al pin de la placa usando cables. Tenemos que asegurarnos de que la pata GND, G, o –, está conectada a tierra. Y conectamos la otra pata en la fila S, al pin que queramos asignarle; por ejemplo, el pin 9.

En Snap4Arduino podemos controlar estos leds con la instrucción «fijar pin digital» seguido del pin en el que hemos conectado el módulo LED. O usaremos «fijar pin» seguido del valor que queramos para la salida, si queremos una señal analógica (figura 24).

Figura 24

Módulos LED RGB

Vídeo 05. Módulos LED RGB

Descripción del procedimiento de conexión de diferentes módulos LED RGB a la placa de Arduino, y de cómo controlarlos utilizando el software de Snap4Arduino.

Los **módulos LED tricolor**, o **módulos LED RGB**, tienen tres pequeños leds dentro:

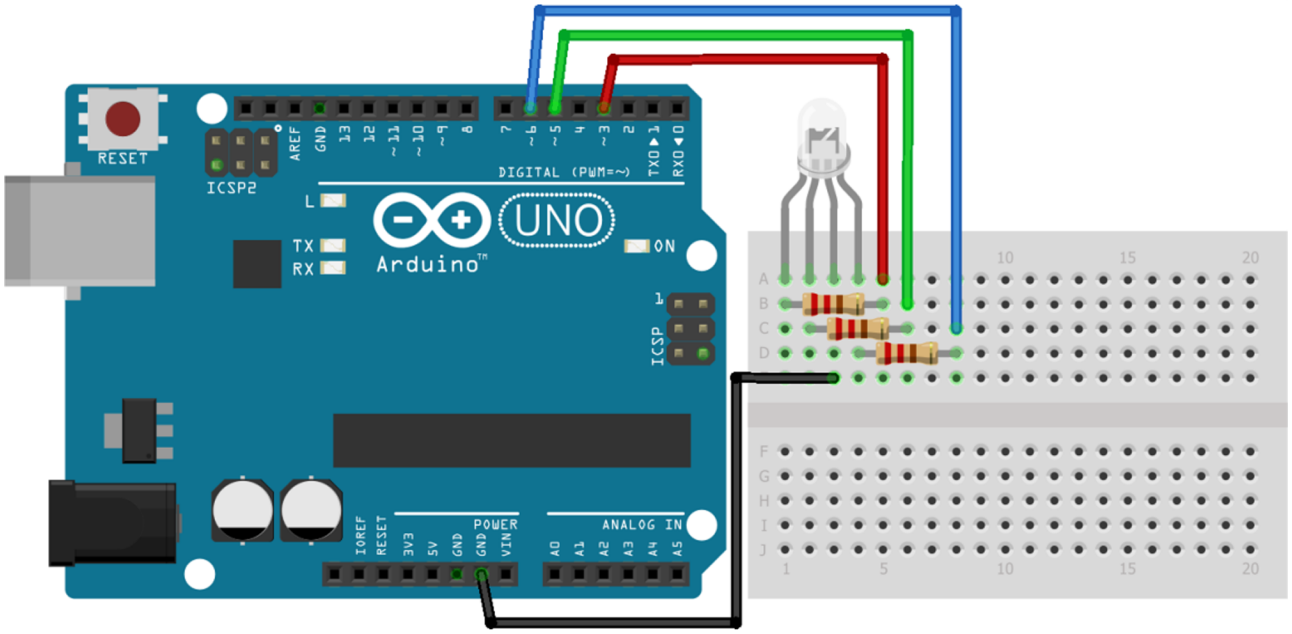
- Uno rojo (*red*)
- Uno verde (*green*)
- Uno azul (*blue*)

Estos tres leds se pueden controlar uno por uno para conseguir un montón de colores diferentes.

La conexión de este tipo de leds es muy sencilla. Vamos a conectar la pata más larga a tierra. El resto de patas, que servirán para indicar la intensidad de los tres colores diferentes, se conectan a tres pines diferentes (3, 5 y 6), con una resistencia de 220 Ω.

También podemos utilizar un módulo que ya tenga una resistencia integrada. Si este es el caso, conectaremos el pin marcado con una G o con un – a GND, y los otros tres pines indicarán el color.

Figura 25

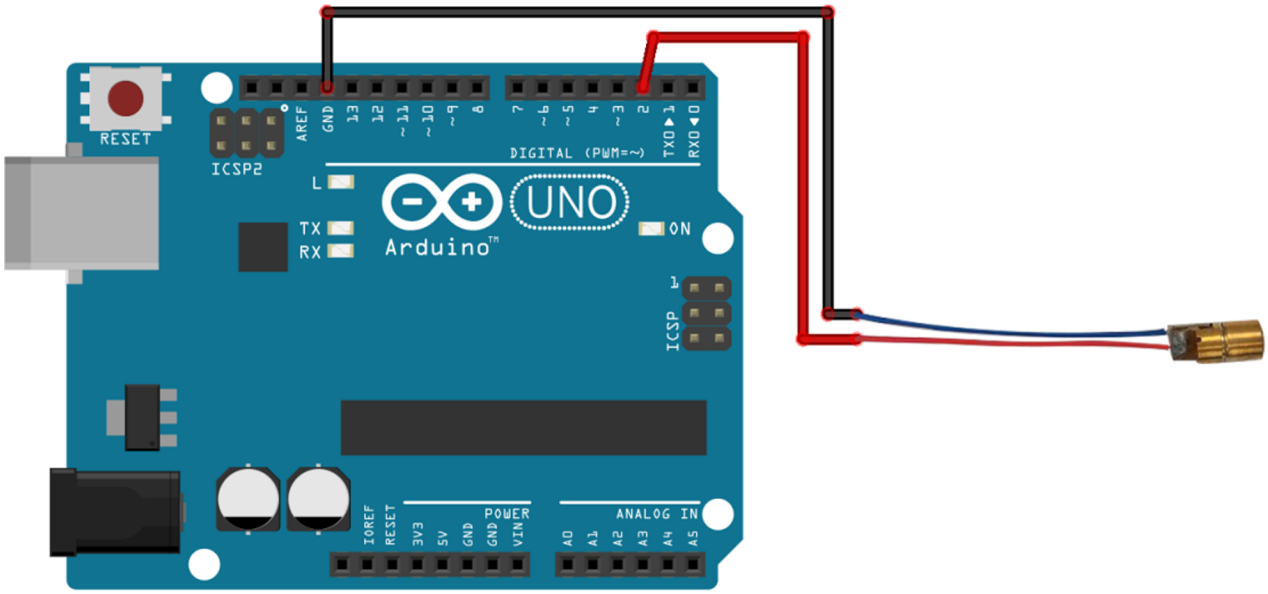


Utilizamos los pines 3, 5 y 6. Así podremos emplear instrucciones analógicas para variar la intensidad poco a poco.

Diodos láser

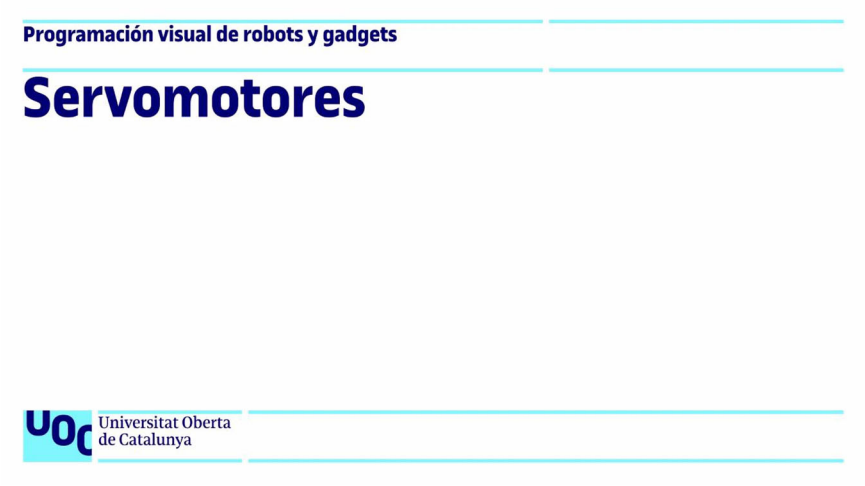
El diodo láser funciona exactamente igual que el módulo LED (figura 26), pero hay que ir con cuidado con él. Al ser una fuente de luz muy concentrada, puede quemar o hacer daño si se utiliza durante demasiado tiempo o sobre elementos sensibles.

Figura 26



2.4.2. Motores de tipo servo

Vídeo 06. Servomotores



Descripción del procedimiento de conexión de los motores de tipo servo a la placa de Arduino, y de cómo controlarlos utilizando el software de Snap4Arduino.

Los **motores de tipo servo**, o **servomotores**, son pequeños motores que nos permiten mover y hacer girar objetos, como por ejemplo unas ruedas (figura 27). Hay dos tipos de servomotores, los servomotores estándar y los de rotación continua:

Figura 27

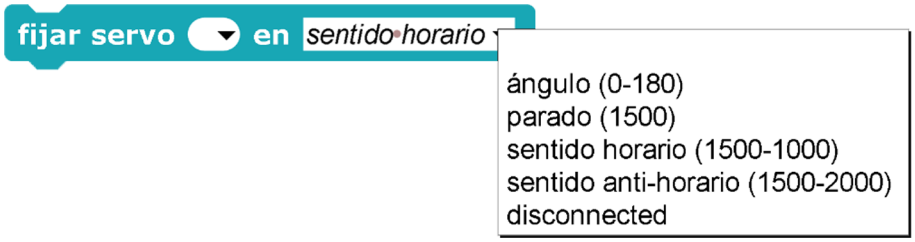
- Los **servomotores estándar** permiten girar un ángulo concreto con mucha precisión, pero solo permiten girar 180º, es decir, media vuelta.

- Los **servomotores continuos**, en cambio, permiten dar todas las vueltas que queramos, e incluso controlar la velocidad a la que se gira.

Para controlar un servomotor en Snap4Arduino, tendremos que conectar el cable negro del servomotor al GND de nuestra placa, el cable rojo a 5V, y el cable blanco (o amarillo) al pin que queramos usar para programar el robot.

Para controlar el servomotor, usaremos el bloque *fijar servo _ en _*, en que habrá que indicar en qué pin está conectado el servo y cómo queremos que se mueva.

Figura 28



Tenemos las opciones siguientes:

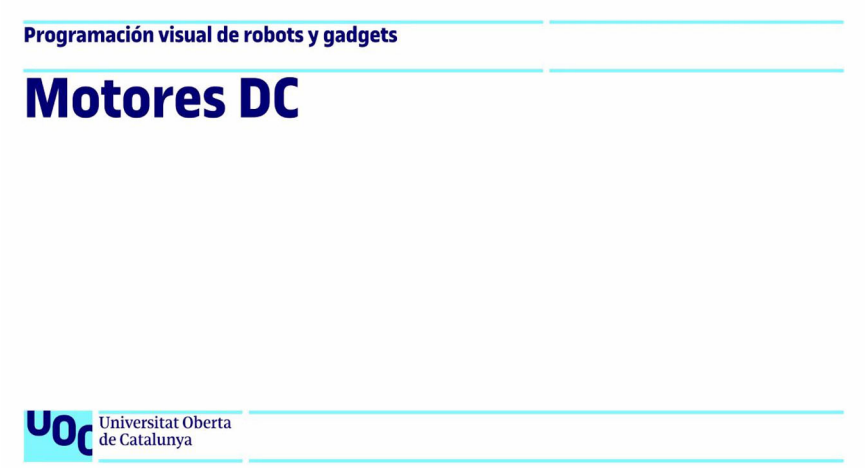
- «ángulo», para indicar a qué ángulo queremos que apunte si se trata de un servo estándar.

- «parado», para que el servomotor se pare.
- «sentido horario», para que el servomotor gire en el sentido de las agujas del reloj.
- «sentido antihorario», para que el servomotor gire en el sentido contrario al de las agujas del reloj.

Podemos crear un sencillo programa para controlar un robot con dos motores que asigne la pulsación de una tecla al sentido de rotación de los motores y así controlar su movimiento.

2.4.3. Motores de tipo CC

Vídeo 07. Motores de tipo CC



Descripción del procedimiento de conexión de los motores de tipo CC sencillos a la placa de Arduino mediante un controlador, y control de estos mediante el software de Snap4Arduino.

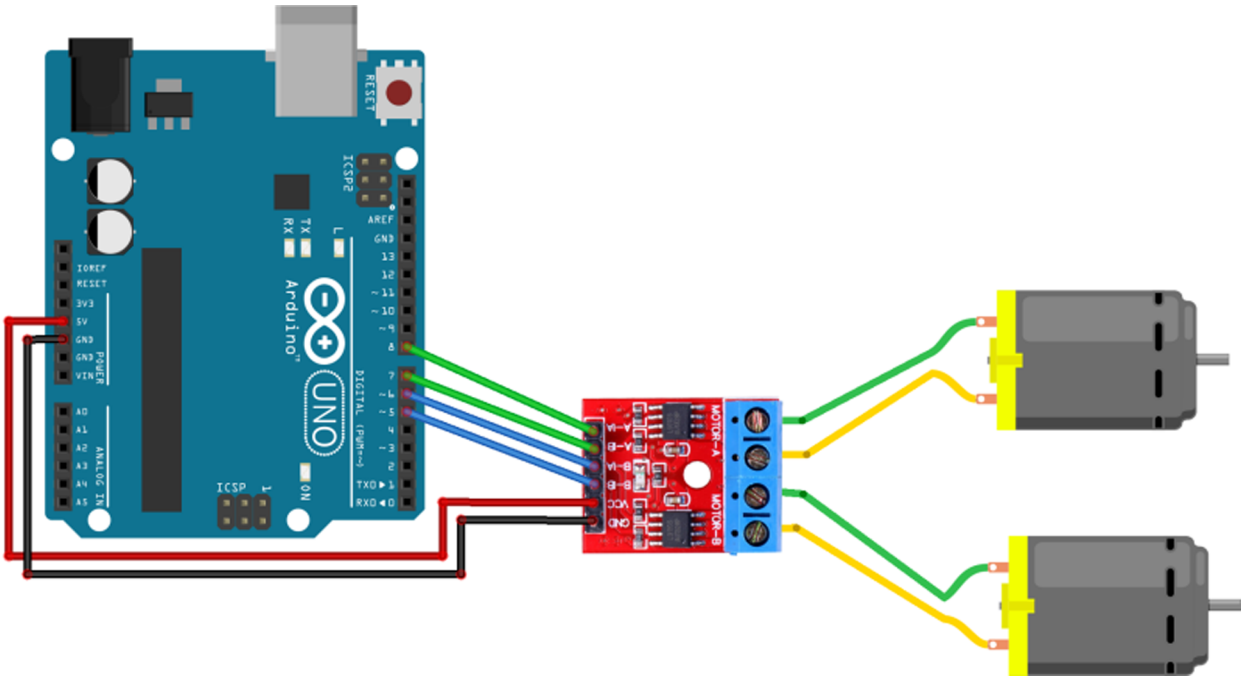
Los **motores de corriente continua**, **motores de tipo CC**, o **motores CC**, son un componente muy parecido a los servomotores continuos, pero tienen un par de diferencias clave:

- Por un lado, son más económicos, cosa que nos puede ayudar a ahorrar un poco de dinero en componentes.

- Por otro lado, no se pueden controlar directamente desde la placa de Arduino, sino que necesitan un controlador[1] de motores para que les proporcione alimentación externa (figura 29).

(1) En inglés, *driver*.

Figura 29



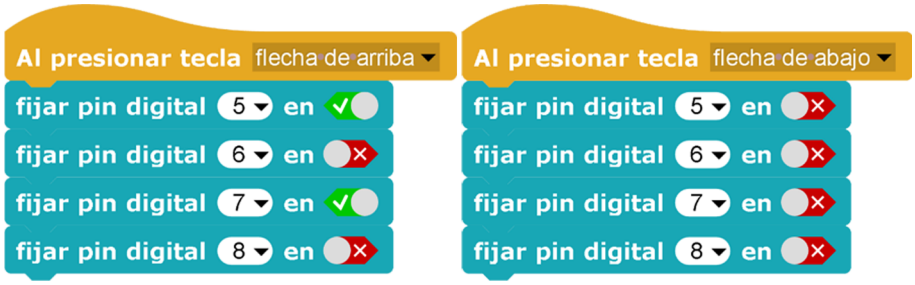
Para utilizar este **controlador de motores** (podemos utilizar el L9110, el L298 o el L293), por un lado debemos conectar los cables que alimentan el motor, y por otro lado, los cables que Arduino utiliza para enviar la señal. En nuestro caso vamos a utilizar el controlador L9110 y extraeremos la alimentación del pin VIN de Arduino.

El pin VIN de Arduino

El pin VIN de Arduino envía el mismo voltaje con el que se alimenta Arduino, sin regular, es decir, si alimentamos Arduino con una pila de 9 V, recibiremos 9 V en este pin.

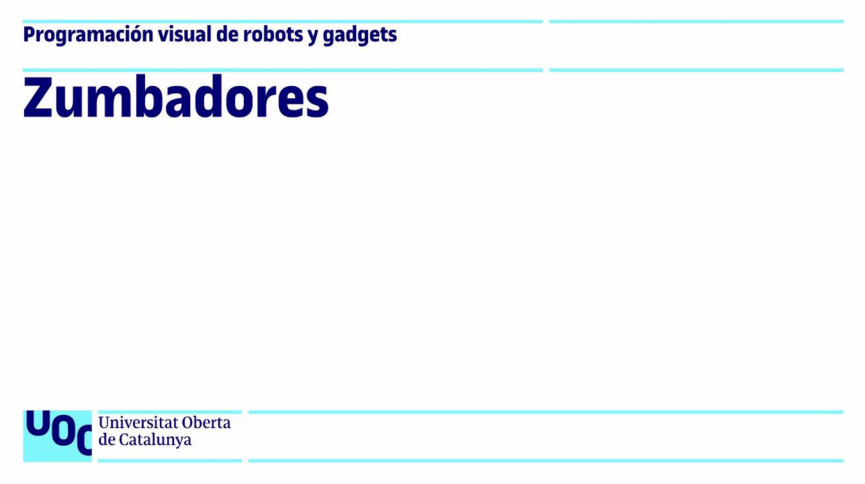
Utilizando dos pines digitales para cada motor, podemos controlar la dirección de rotación del motor. En nuestro caso hemos conectado el motor A a los pines 5 y 6, y el motor B a los pines 7 y 8. Si encendemos uno de los pines y apagamos el otro, el motor girará en un sentido, y si invertimos estos pines, girará hacia el lado contrario. Si paramos todos los pines, el motor estará parado.

Figura 30



2.4.4. Zumbadores

Vídeo 08. Zumbadores



Funcionamiento de diferentes módulos de altavoz sencillos y control de estos mediante el software Snap4Arduino.

Los zumbadores son pequeños altavoces muy sencillos que nos permiten generar sonido. Hay dos tipos de zumbadores, los activos y los pasivos:

Figura 31

- Los **zumbadores activos** emiten un pitido muy fuerte, pero no dejan reproducir frecuencias diferentes.
- Los **zumbadores pasivos** suenan algo más suave, pero permiten modular la frecuencia a la que suenan.

Podemos programar los zumbadores directamente para que se enciendan y apaguen como si fueran un led (*pin digital n encendido*), pero es mucho más interesante usarlos junto a una librería que nos permita decidir directamente qué tono queremos que el zumbador reproduzca. Así podemos componer canciones o melodías sencillas para que se reproduzcan gracias al zumbador (*reproducir tono m en pin n*).

Para abrir esta librería:

- Primero tenemos que descargar el *firmware* para la placa [«Snap4Arduino-Firmata» / «BuzzerFirmata»] (accesible en línea).
- Instalaremos el *firmware* BuzzerFirmata en nuestra placa usando el programa de Arduino.
- A continuación, tenemos que incluir los bloques de Snap4Arduino para poder usar esa funcionalidad en nuestros programas (figura 32).

Figura 32

¡Y ya podemos componer nuestras canciones!

2.4.5. Indicadores de siete segmentos

Vídeo 09. Indicadores de siete segmentos

Descripción del procedimiento conexión de indicadores del tipo de siete segmentos a la placa de Arduino, y de cómo controlarlos utilizando el software de Snap4Arduino.

Los indicadores[2] de siete segmentos son una manera fácil de mostrar números y letras (figura 33). Internamente funcionan igual que siete módulos LED con la conexión de tierra común.

(2)En inglés, *displays*.

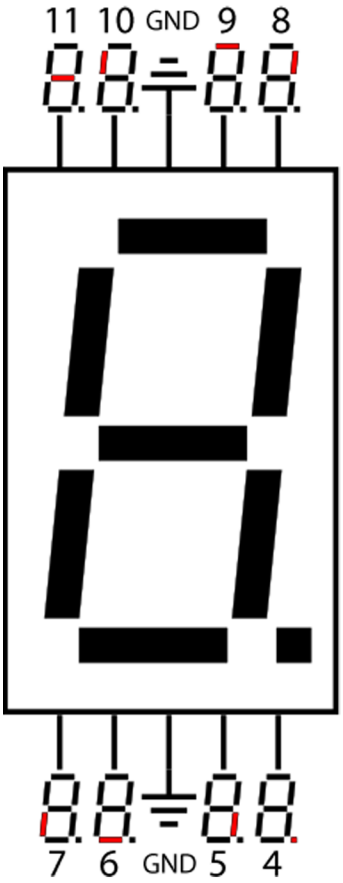
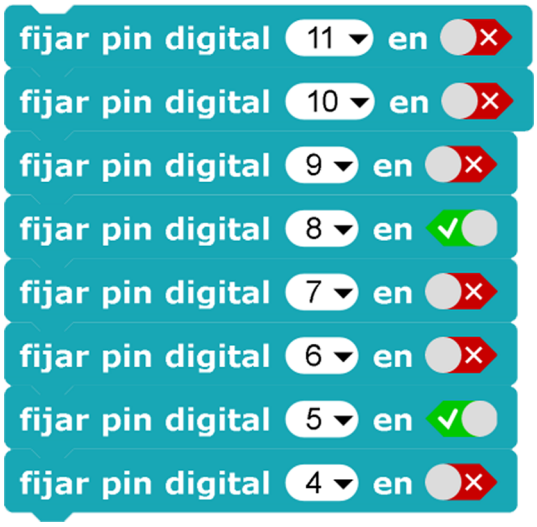


Figura 33

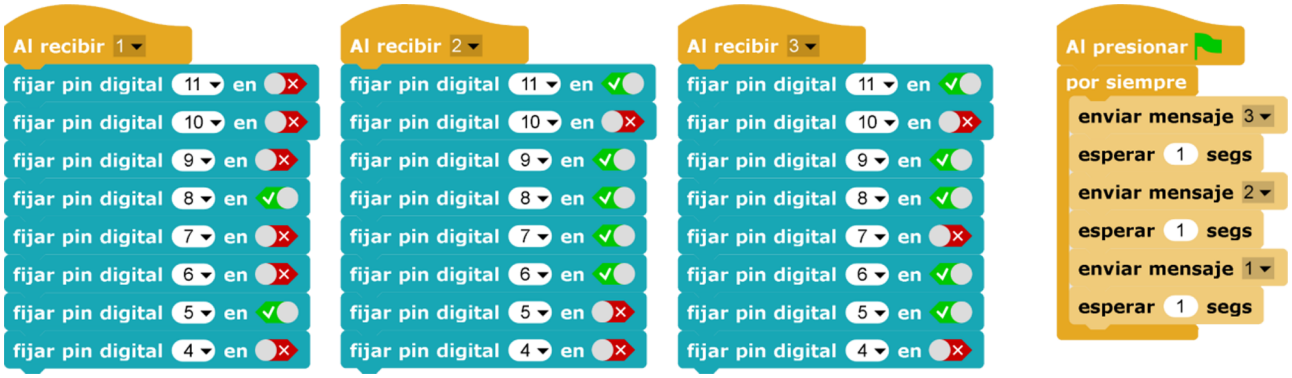
Los dos pines centrales se interconectan a una resistencia de 220 Ω y se conectan a tierra. El resto de pines se conectan a diferentes salidas de Arduino, que encenderán una línea u otra al activarlas (figura 34).

Figura 34



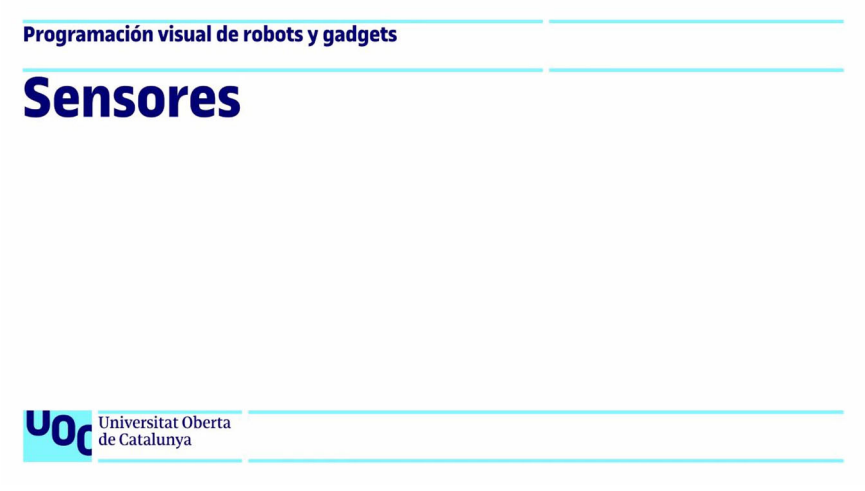
Diferentes combinaciones harán aparecer distintos números o letras (figura 35).

Figura 35



2.5. Sensores

Vídeo 10. Sensores



Descripción de la conexión de diferentes tipos de sensores a la placa de Arduino mediante divisores de tensión o utilizando módulos específicos para ello.

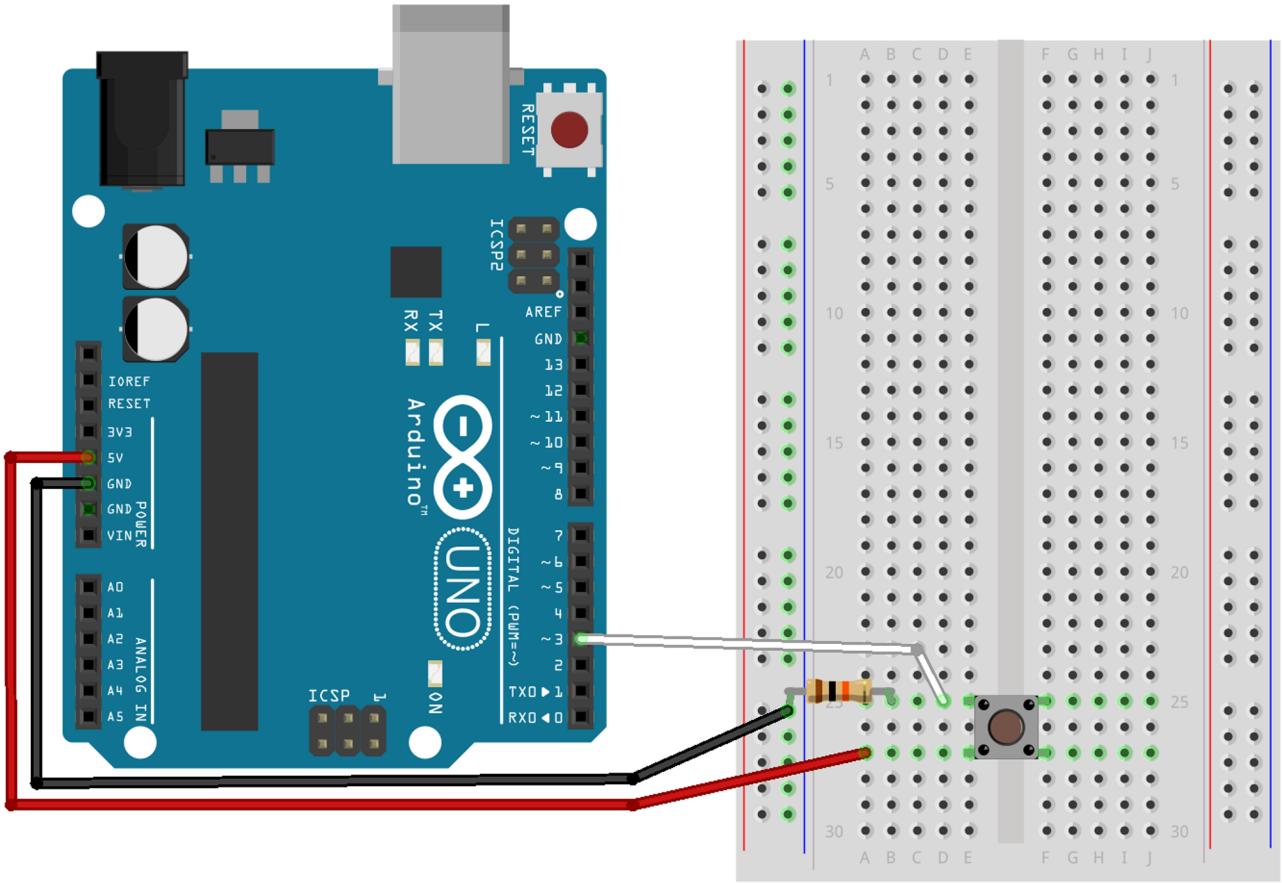
Los sensores son componentes muy útiles que nos permiten detectar cambios en el mundo real. Con ellos vamos a activar y desactivar elementos de nuestros programas y robots.

Muchos de los sensores que usamos son «resistivos». Esto significa que generan más o menos resistencia al paso de la electricidad según la luz, el contacto o la temperatura.

Para hacer funcionar estos sensores utilizamos un montaje que se denomina **divisor de tensión**. Este montaje consiste en conectar una resistencia de 10 kΩ en serie con el sensor. El voltaje que «cae» por la resistencia va a ser mayor o menor según si el sensor recibe mucha luz/temperatura/contacto.

El esquema de conexión es muy sencillo. Tenemos la resistencia y el sensor conectados en serie. Conectamos una de las puntas a GND, la otra a 5V, y justo en medio tiramos un cable hacia la entrada para leer el sensor (figura 36).

Figura 36



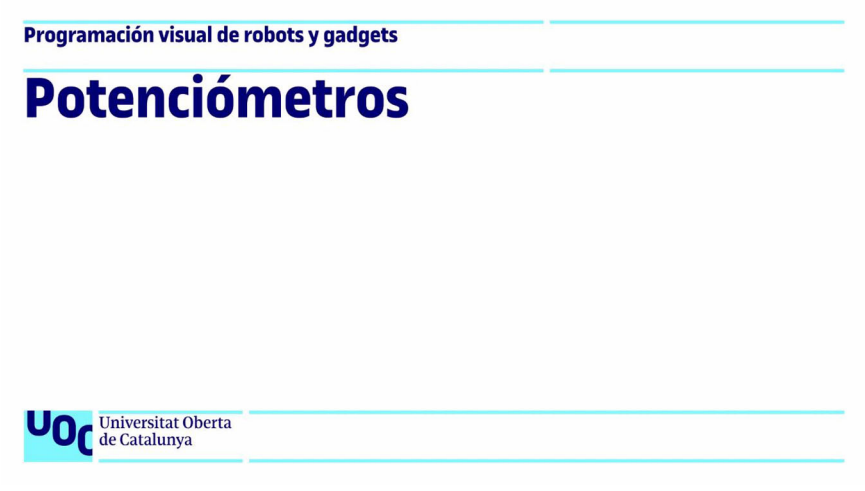
Para programar los sensores utilizamos los bloques de Snap4Arduino *lectura analógica* o *lectura digital*, dependiendo de si queremos un valor digital o analógico.

Figura 37

Conectaremos los sensores digitales en los pines 2 a 13, pero debemos conectar los sensores analógicos en los pines analógicos A0 a A5.

2.5.1. Potenciómetros

Vídeo 11. Potenciómetros

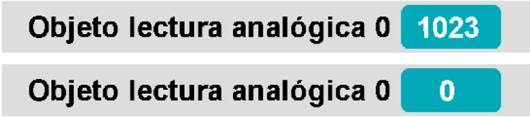


Descripción del procedimiento de conexión de potenciómetros a la placa de Arduino, y de cómo controlarlos utilizando el software de Snap4Arduino.

Para utilizar un potenciómetro, conectaremos la pata izquierda del potenciómetro a GND, la pata derecha del potenciómetro a 5V, y la pata central del potenciómetro al pin de entrada analógica que queramos (entre A0 y A5).

Al usar el bloque *lectura analógica*, los potenciómetros generan una señal que va entre 0 y 1.023 (figura 38). Si giramos el potenciómetro en una dirección u otra, obtendremos valores diferentes.

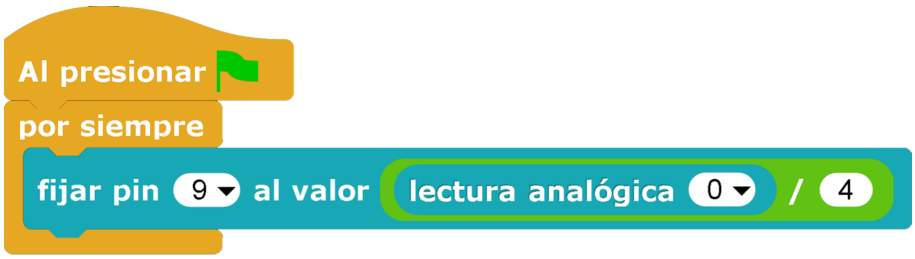
Figura 38



Para probarlo, podemos crear un programa que controle la intensidad de un led conectado al pin 9.

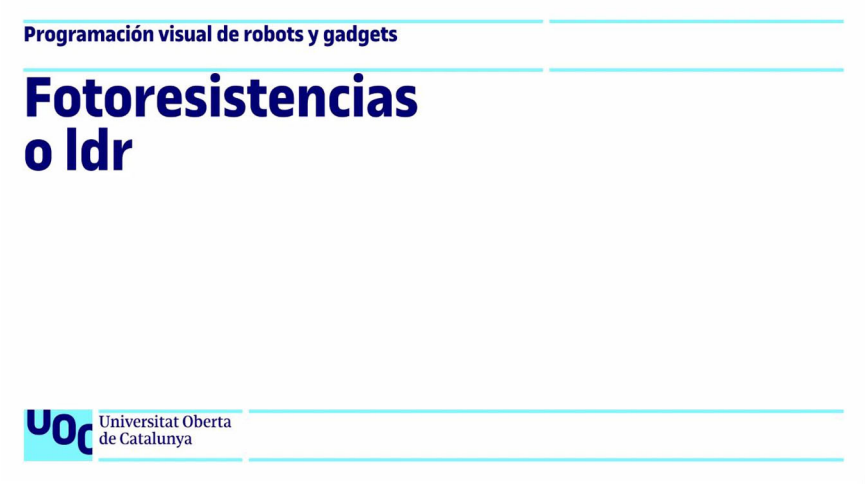
La salida analógica del led (pin 9) necesita un valor de 0 a 255. Como la lectura analógica del sensor será de 0 a 1.023, vamos a tener que dividirlo por cuatro para que le envíe un valor de 0 a 255 (figura 39).

Figura 39



2.5.2. Sensores de luz

Vídeo 12. Fotorresistencias, o módulos LDR



Descripción del procedimiento de conexión de resistencias sensibles a la luz (LDR), o fotorresistencias, a la placa de Arduino, y de cómo controlarlos utilizando el software de Snap4Arduino.

Los **sensores de luz**, o **fotorresistencias**, funcionan de manera similar al resto de sensores resistivos. Si disponemos de un módulo con el sensor integrado, simplemente conectamos el pin marcado con el – a tierra (GND), el pin central a 5V, y el pin marcado con S al pin de entrada analógica que elijamos.

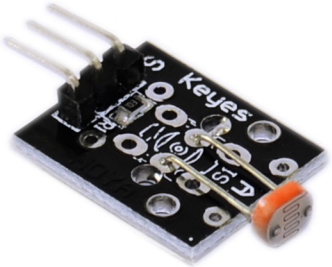
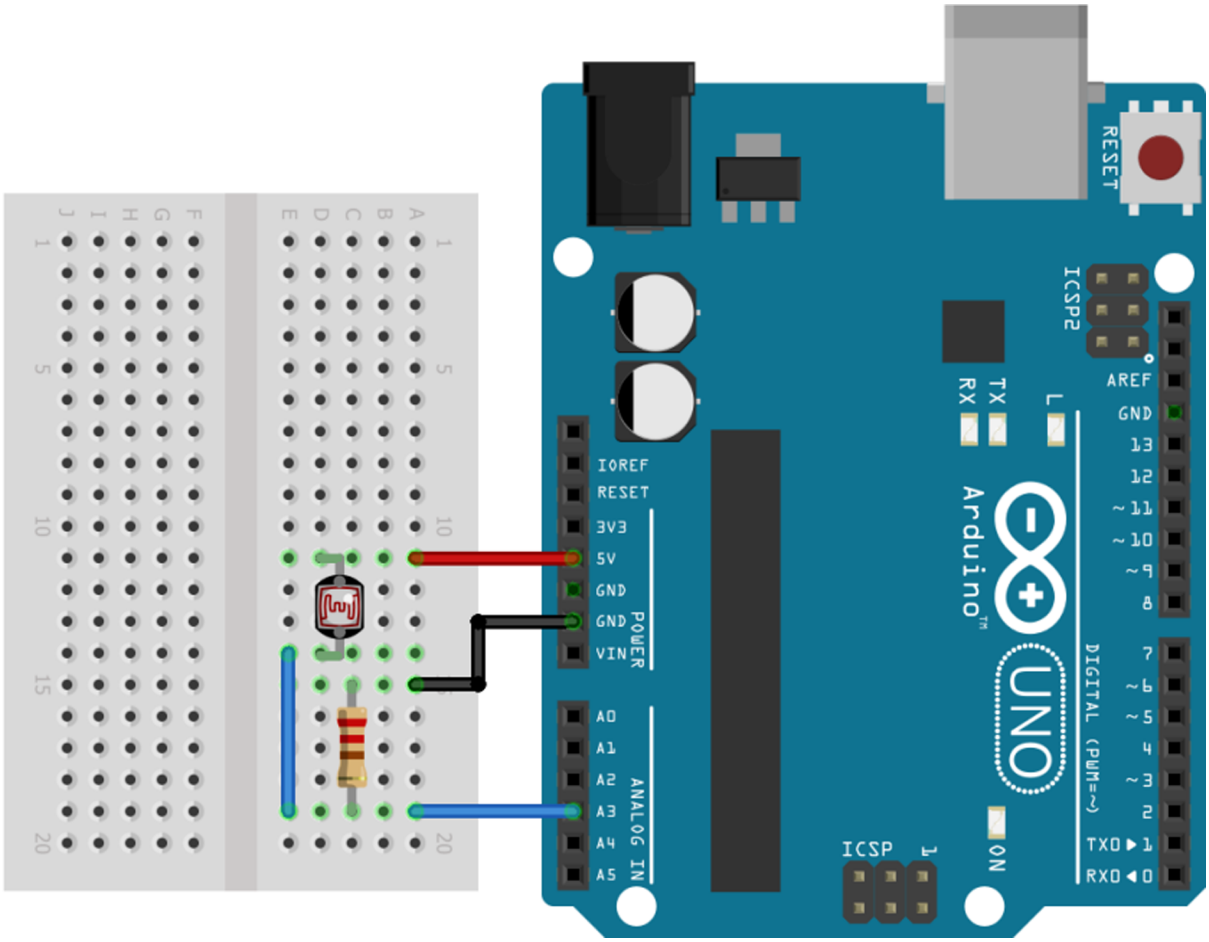


Figura 40
Sensor LDR o fotoresistencia.

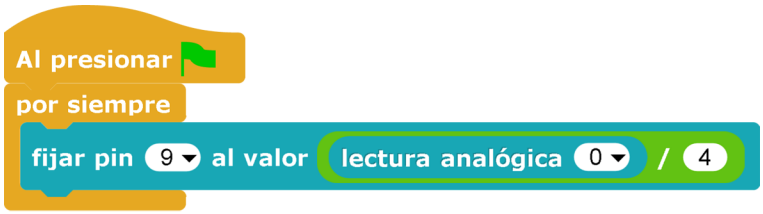
Si disponemos del sensor sin el módulo, conectamos el sensor en serie con una resistencia de 10 kΩ. Entonces conectamos el vértice que interconecta la resistencia con el sensor al pin de lectura analógica (A0), la otra pata del sensor a 5V, y la pata de la resistencia restante a GND (figura 41).

Figura 41



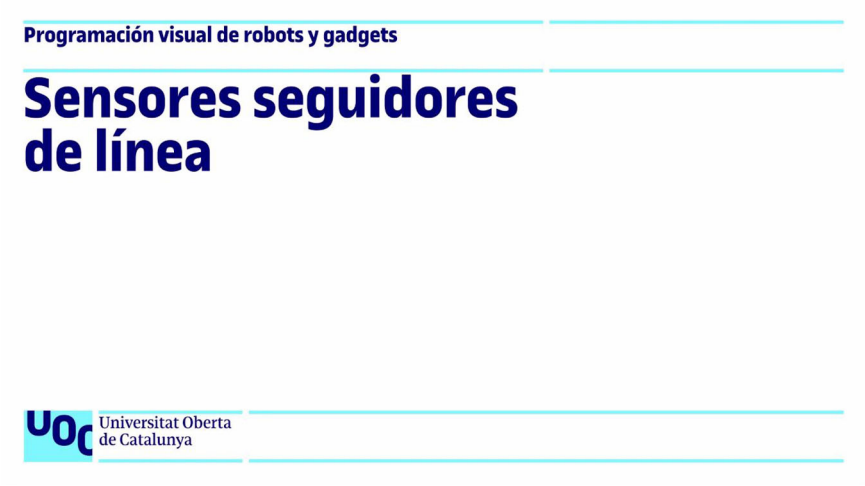
Podemos usar el mismo programa que con el potenciómetro para probar su uso controlando la intensidad de un led. El led perderá intensidad cuando el sensor reciba luz y se iluminará más intensamente cuando el sensor deje de recibir luz.

Figura 42



2.5.3. Seguidores de línea

Vídeo 13. Sensores seguidores de línea



Descripción del procedimiento de conexión de módulos seguidores de línea a la placa de Arduino, y de cómo controlarlos utilizando el software de Snap4Arduino.

Los **sensores seguidores de línea**, también denominados **sensores sigue-líneas**, utilizan tecnología infrarroja para detectar si el color que hay debajo es oscuro o es claro (figura 43). En el caso de los robots seguidores de línea, se utilizan para detectar si lo que tienen debajo es un fondo blanco o una línea negra. Muchos de estos sensores tienen un potenciómetro para ajustar la sensibilidad del sensor, y un pequeño led que indica si se está viendo blanco o negro.

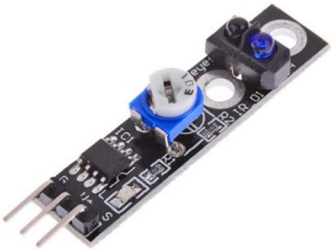
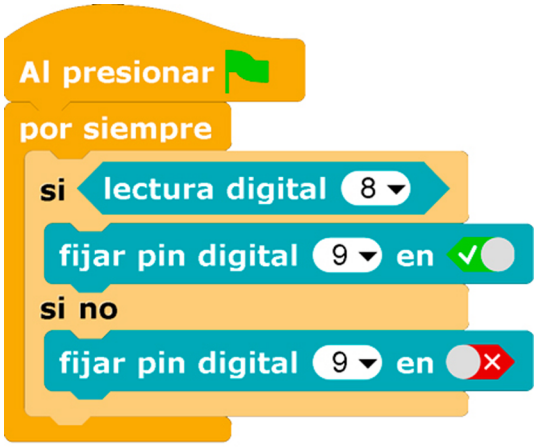


Figura 43
Sensor de infrarrojos para sigue-líneas.

El modo de conectarlo es muy sencillo. Conectamos la pata G a tierra (GND), la pata V^+ a 5V, y la pata S a una entrada digital, por ejemplo, al pin 8.

Para programarlo, simplemente tenemos que leer el valor del pin 8 (figura 44).

Figura 44



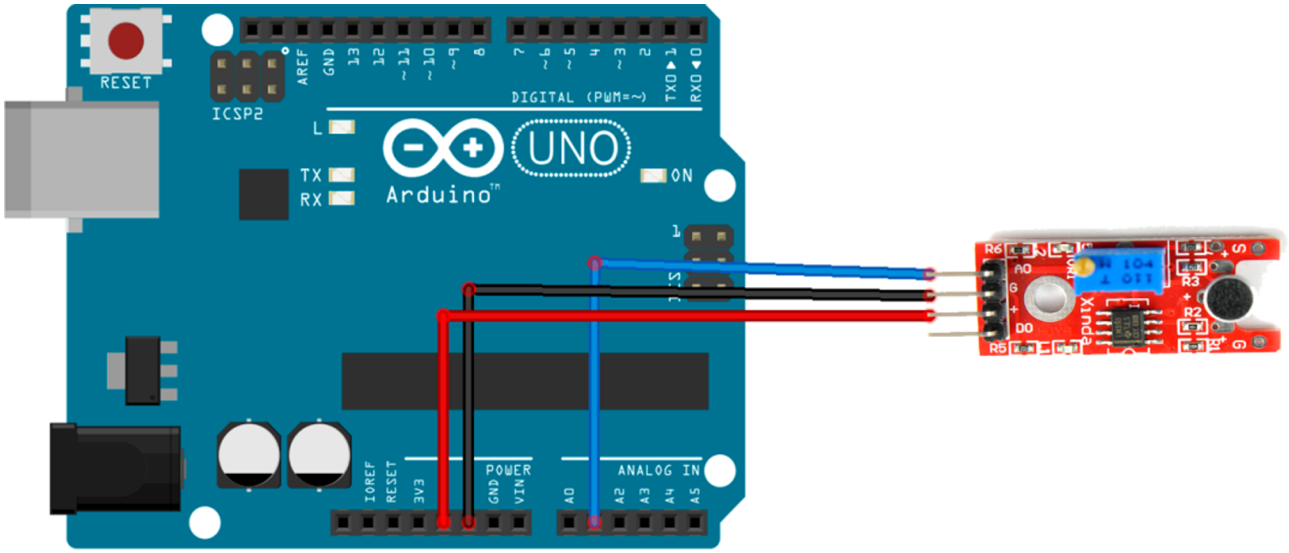
2.5.4. Sensores de sonido

Vídeo 14. Sensores de sonido

Descripción del procedimiento de conexión de los módulos sensores de sonido a la placa de Arduino, y de su calibración.

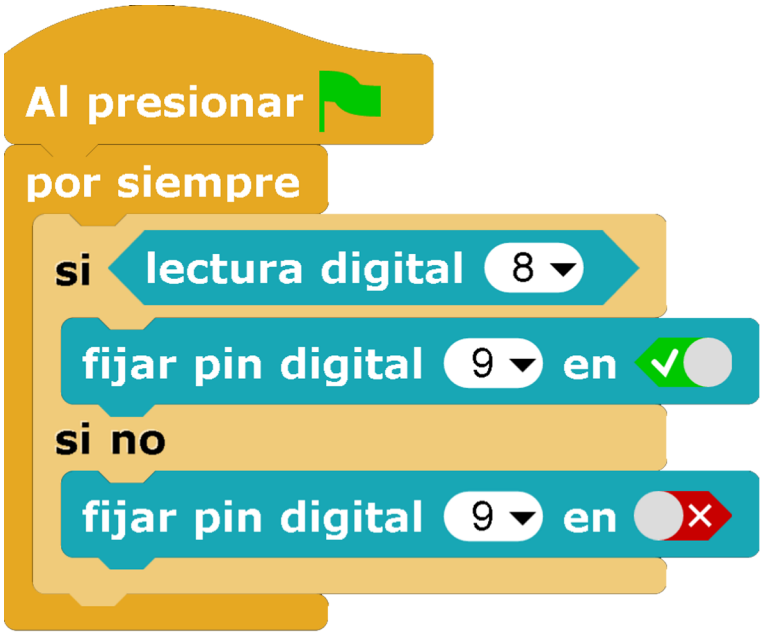
Los sensores de sonido funcionan de manera similar a un potenciómetro. Para utilizar el sensor de sonido, simplemente conectamos el pin marcado con GND a tierra (GND), el marcado con +5V a 5V y el pin marcado con OUT al pin de entrada analógica que elijamos (figura 45).

Figura 45



El led se apagará cuando el sensor detecte sonido y se iluminará cuando el sensor deje de detectar sonido.

Figura 46



2.5.5. Sensores de proximidad

Vídeo 15. Sensores de proximidad, o de ultrasonidos

Descripción del procedimiento de conexión de los módulos de ultrasonidos HC-SR04 para efectuar lecturas de distancias mediante una placa de Arduino, y de cómo controlarlos utilizando el software de Snap4Arduino.

Un sensor de proximidad es un sensor un poco más complejo y necesita un *firmware* especial para poder usarse en Snap4Arduino. Su instalación es muy sencilla, exactamente igual que cuando instalamos Firmata por primera vez:

- Primero deberemos descargar el archivo siguiente, que contiene el *firmware* para la placa de Arduino y los bloques que deberemos importar a Snap4Arduino: Snap4Arduino-Firmata/Ultrasound_HC_SR04Firmata/ (accesible en línea)
- Descomprimimos el archivo descargado y abrimos el archivo «Ultrasound_HC_SR04Firmata.ino» con el programa Arduino.
- Antes de cargarlo en la placa, debemos importar la librería «Ultrasonic_HC-SR04.zip». Para incluir la librería, debemos ir a «Progra-

ma» > «Include library» > «Add.zip library», y de ahí buscamos y seleccionamos la librería que queremos importar.

- Igual que como cuando instalamos el Firmata original, pulsamos en «cargar» y subimos el programa a la placa de Arduino.
- En Snap4Arduino, debemos pulsar el icono con la hoja en blanco que hay en la parte superior izquierda de la pantalla y seleccionar la opción «importar», e importamos el archivo «Bloque Ultrasonido Snap4Arduino». Esto nos creará un bloque nuevo: el *lectura distancia ultrasonido*, que podemos usar para leer el valor que nos da el sensor (figura 47).

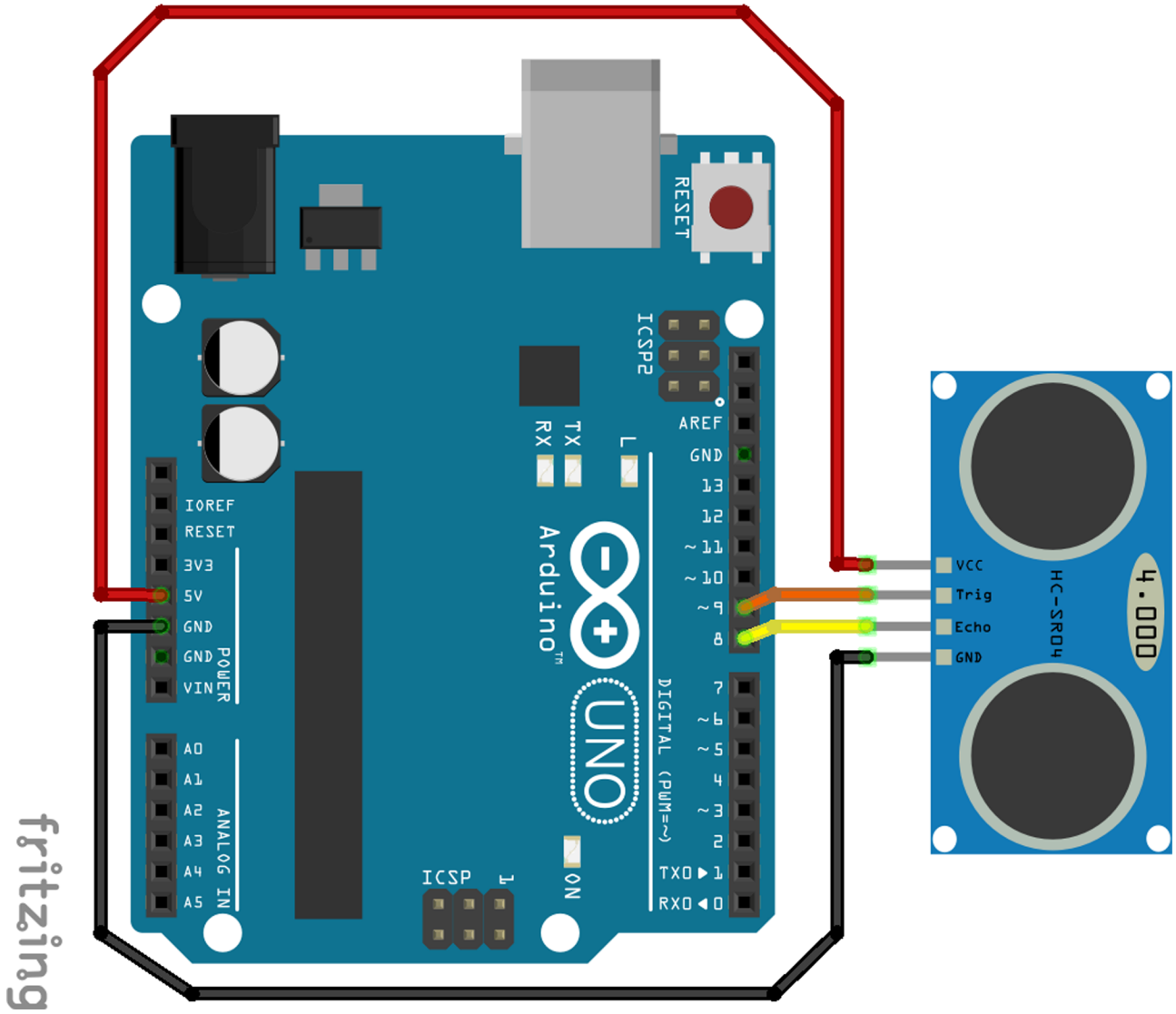
Figura 47

El montaje del sensor es sencillo, pero debemos hacerlo correctamente para que funcione.

a) Conectamos el pin GND a tierra y el pin VCC a 5V.

b) A continuación es importante que conectemos el pin Trig al pin digital 9, y el pin Echo al pin 8 (figura 48).

Figura 48

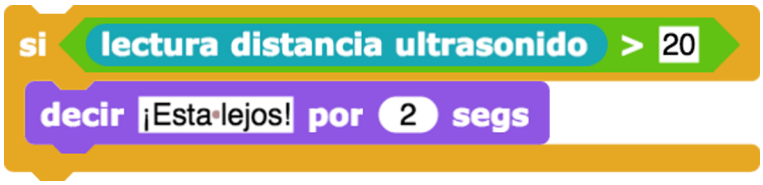


En Snap4Arduino, simplemente tenemos que usar el bloque que hemos importado para leer el valor de distancia que nos da el sensor (figura 49).

Figura 49

Podemos guardar este valor en una variable –por ejemplo– o como una condición de una estructura condicional (figura 50).

Figura 50



Si somos creativos, los sensores nos ofrecen posibilidades infinitas.

2.6. Conectividad

Por último, tratamos la conectividad de la placa de Arduino mediante módulos de comunicación. En concreto, nos vamos a centrar en los módulos de comunicación Bluetooth.

2.6.1. Módulos de comunicación Bluetooth

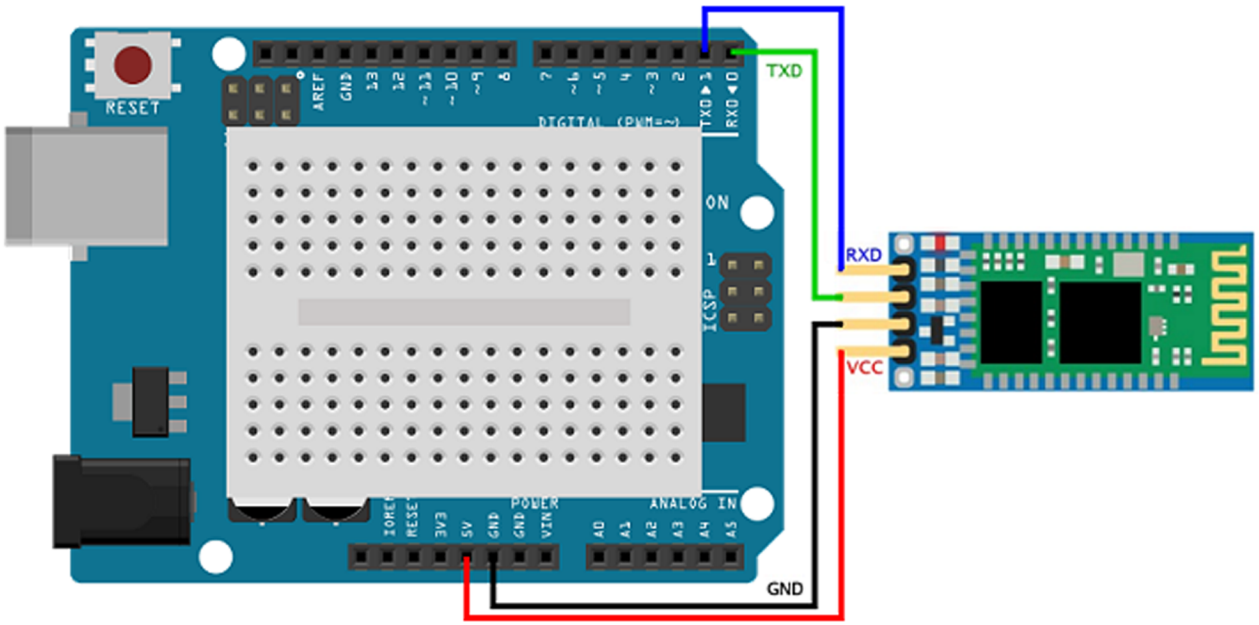
Vídeo 16. Módulos de comunicación Bluetooth

Análisis de diferentes módulos de comunicación Bluetooth. Configuración del *firmware* de estos. Emparejamiento y comunicación con Snap4Arduino.

Para realizar la conexión por Bluetooth vamos a utilizar el módulo HC-06, configurado a 38.400 bps. Como es posible que este módulo no tenga esta velocidad (38.400 bps) configurada por defecto, vamos a tener que efectuar unos pasos previos con Arduino.

En primer lugar conectamos la pata más larga de un led al PIN13, y la más corta a GND (figura 51). Abrimos el software de Arduino y cargamos el software «[REC12a] Conexión Bluetooth.zip» con el módulo HC-06 desconectado. Cuando el led comience a parpadear cada medio segundo, conectamos el módulo y grabamos el *firmware* de S4A sin quitar el cable.

Figura 51



A continuación debemos sincronizar el módulo de Bluetooth con nuestro PC. Lo haremos del mismo modo que sincronizamos cualquier dispositivo Bluetooth, utilizando la contraseña «1234» cuando nos lo pida.

3. Prácticas de dificultad baja

A continuación presentamos algunas aplicaciones simples para practicar con Arduino:

1) Semáforo

2) Semáforo con botón para peatones

3) Juegos del tipo «Clicker»

4) Juego de reflejos

5) Coche con motores de tipo CC

6) Casa inteligente - Luces inteligentes

7) Luz de fiesta con un módulo LED RGB

8) Alarma antirrobo

3.1. Semáforo

En esta práctica simularemos un semáforo a partir de tres leds de colores distintos: rojo, amarillo y verde, conectados a nuestra placa de Arduino.

Los semáforos tienen unos tiempos establecidos y un orden en el que se enciende la luz de cada color. Vamos a programar nuestro semáforo para que cumpla los tiempos de un semáforo de verdad.

Para esta práctica solamente necesitaremos (figura 52):

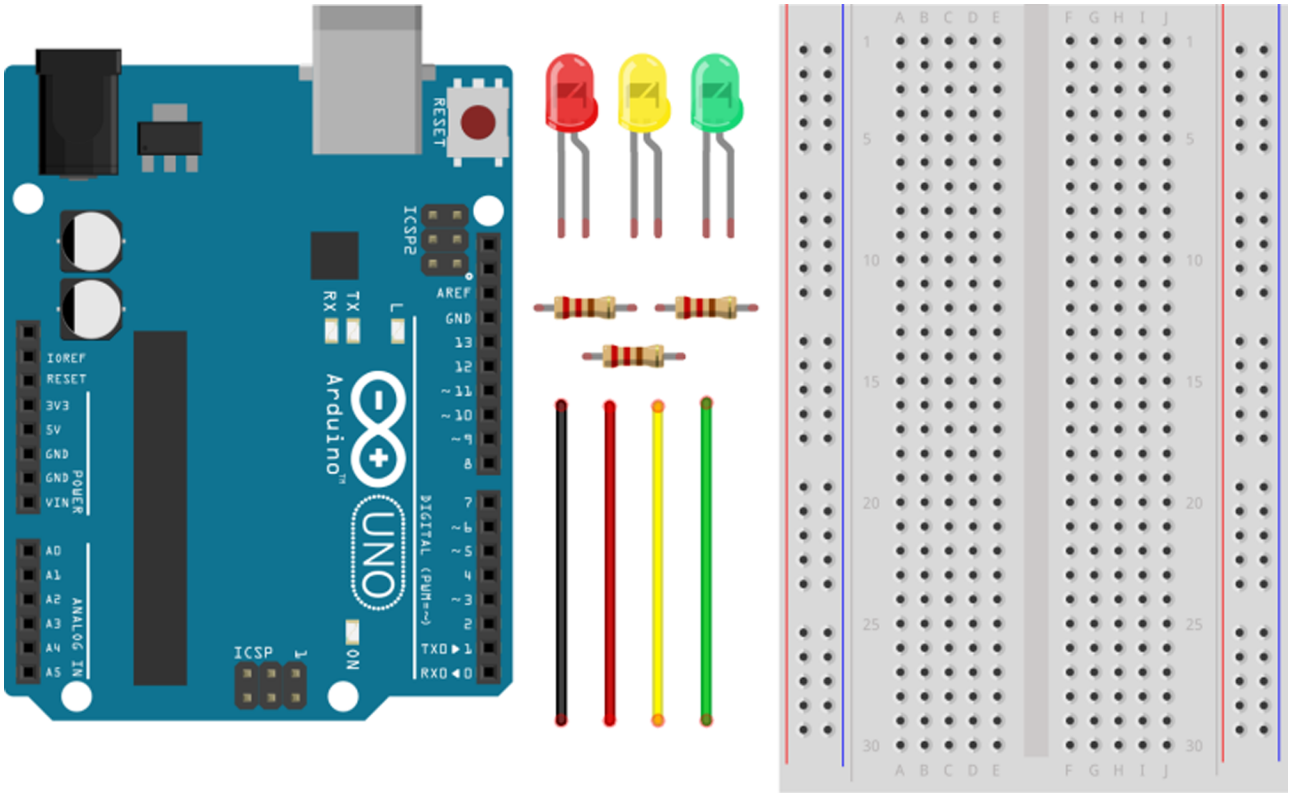
- La placa de Arduino
- 3 leds: rojo, amarillo y verde
- 3 resistencias de 220 Ω
- 1 placa de prototipado[3]
- Cables

Observación

También podemos usar los módulos LED en lugar de los leds y las resistencias.

(3)En inglés, *breadboard*.

Figura 52



1) El montaje es muy sencillo (figura 53):

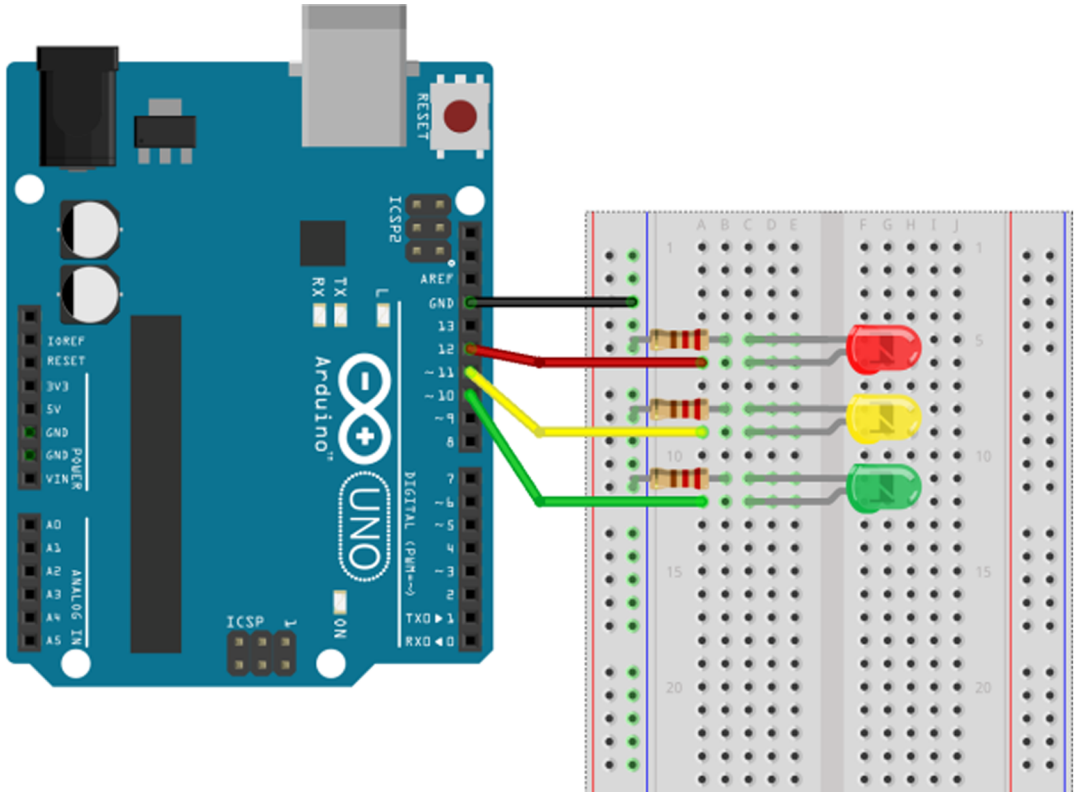
a) Primero, colocaremos los tres leds en la placa de prototipado siguiendo los colores rojo, amarillo y verde. Los colocamos con la pata larga a la izquierda.

b) A continuación, conectamos las resistencias entre las patas largas y la fila que tiene un signo – de nuestra placa de prototipado, para conectarlos a tierra. Conectamos a tierra esta fila usando un cable que conectamos a GND en la placa de Arduino.

c) Por último, conectamos las patas cortas de los leds a los pines 9, 10 y 11 de la placa de Arduino, para poder controlar cuándo se apaga o se enciende cada led.

También podemos usar módulos LED en vez de leds y resistencias, en cuyo caso el montaje es mucho más sencillo, ya que simplemente conectamos cada led al pin que toca y a tierra.

Figura 53

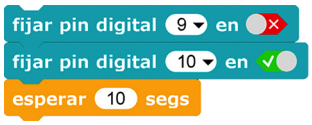


2) Para programarlos, hacemos un programa en Snap4Arduino que nos permita encender el led verde y que se mantenga encendido unos segundos (figura 54).

Figura 54

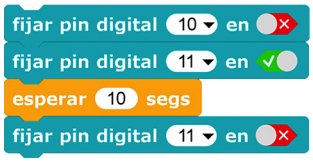
A continuación, hacemos que se apague el led verde y se encienda el led amarillo (figura 55).

Figura 55



Hacemos lo mismo para el led rojo, y hacemos que vuelva a empezar (figura 56).

Figura 56

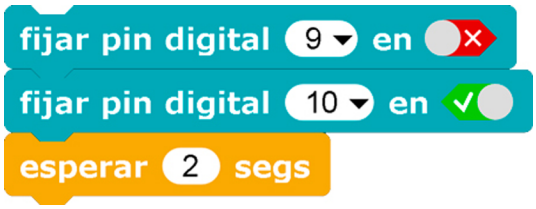


Para que vuelva a empezar el ciclo y que sea un bucle, acordaos de que tenemos que usar una estructura de bucle como esta (figura 57):

Figura 57

Para añadir más realismo podemos hacer que los tiempos en lo que se enciende cada luz se correspondan con los de un semáforo real. Podemos hacer que la luz amarilla esté encendida poco tiempo, ya que solo se usa para avisar de que el semáforo va a cambiar a rojo.

Figura 58



¡Ya tenemos nuestro semáforo!

Cuestiones

¿Podríamos añadir dos luces más para que también esté el semáforo de peatones?

3.2. Semáforo con botón para peatones

En esta práctica seguiremos con la idea del semáforo, pero esta vez vamos a hacer un semáforo que cambie de color cuando un peatón pulse el botón.

En vez de tener un ciclo que se repite automáticamente, vamos a usar el botón para que las luces cambien de color.

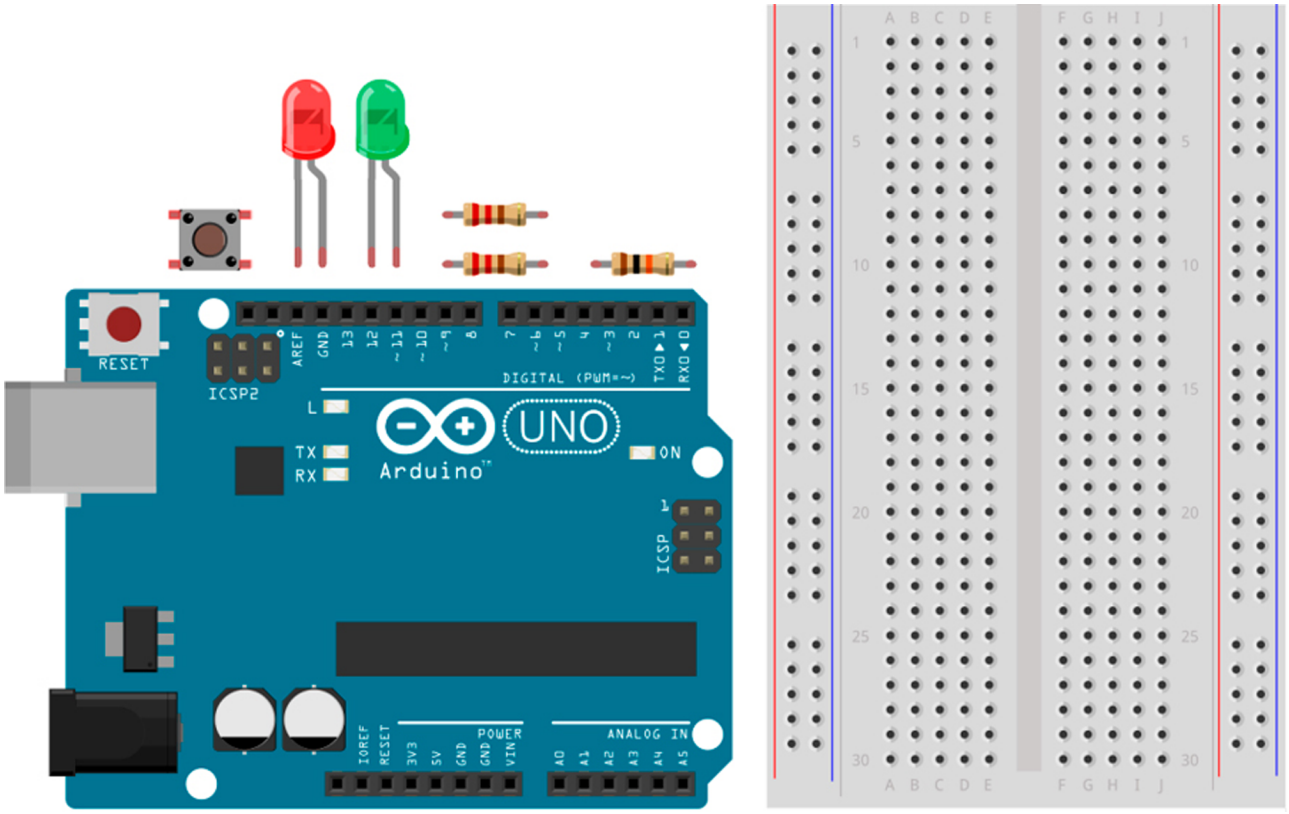
Para esta práctica solamente necesitaremos (figura 59):

- La placa de Arduino
- 1 botón / pulsador
- 2 leds: rojo y verde
- 2 resistencias de 220 Ω
- 1 resistencia de 10 kΩ
- 1 placa de prototipado
- Cables

Observación

También podemos usar los módulos LED en lugar de los led y las resistencias.

Figura 59

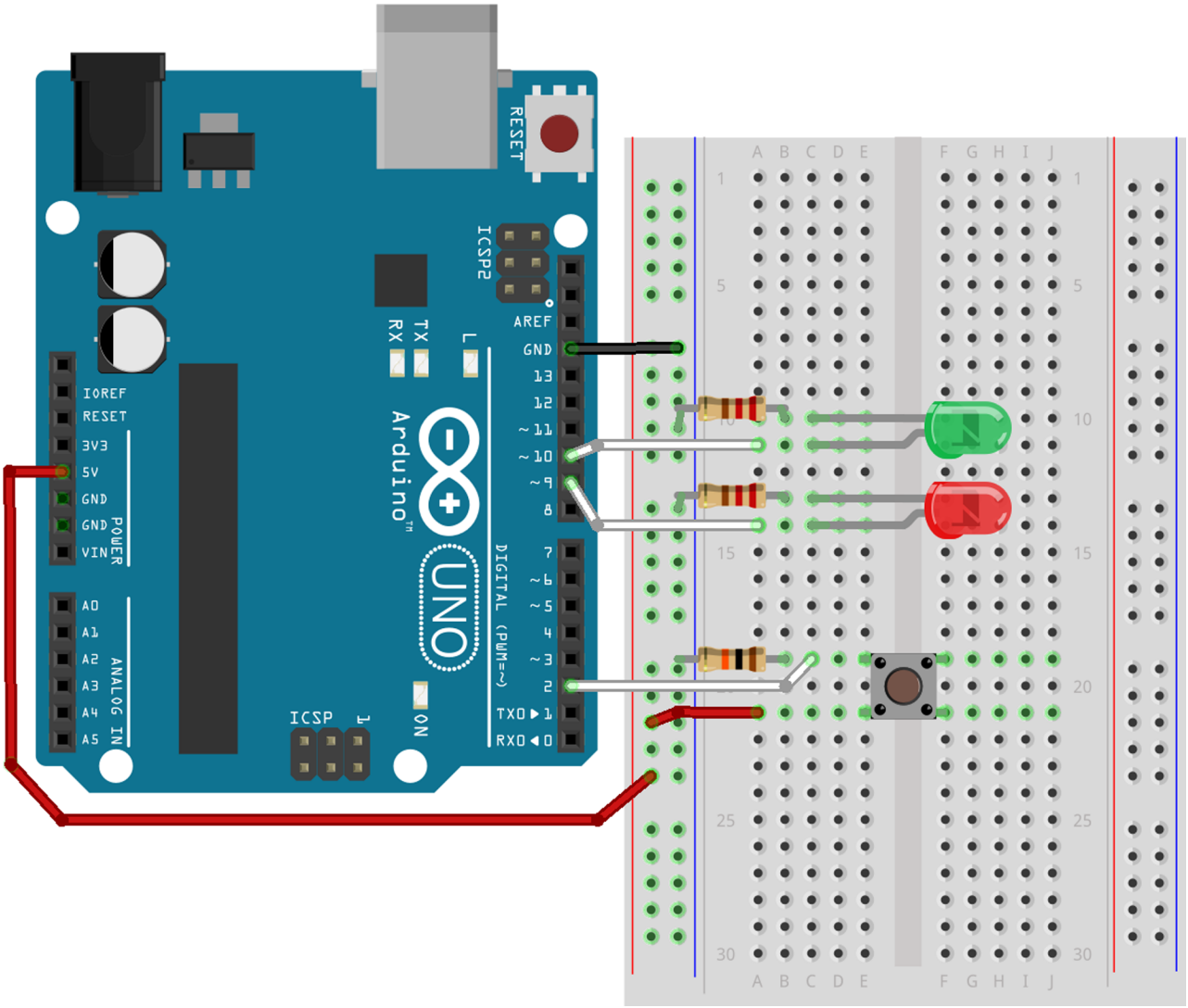


1) El montaje consta de dos partes (figura 60):

a) Por un lado, tenemos que montar los leds. El montaje es igual que en la práctica anterior. Conectamos el led rojo al pin 9 y el led verde al pin 10.

b) Por el otro lado, tenemos que montar un pulsador. Para ello tendremos que colocar el pulsador en la placa de prototipado y montar un circuito divisor de tensión. Conectamos el botón al pin 3.

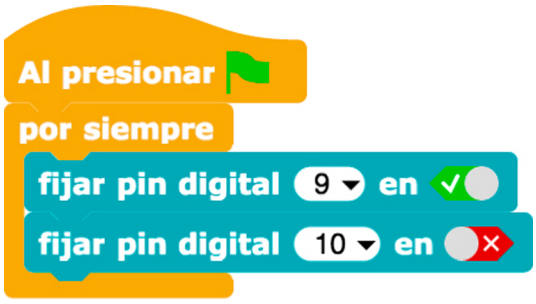
Figura 60



2) Para programarlo, vamos a hacer que el semáforo de peatones esté siempre en rojo y cambie de color tras unos segundos si alguien pulsa el botón. Pasado el tiempo correspondiente para que los peatones crucen, la luz verde parpadea y el semáforo vuelve a ponerse en rojo.

Primero encenderemos el led rojo. También es recomendable asegurarnos de que el led verde está apagado (figura 61).

Figura 61



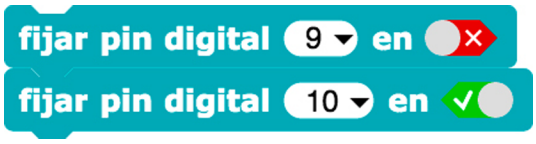
Podemos hacer un bucle infinito y poner un bloque para que espere hasta que alguien pulse el botón (figura 62).

Figura 62

Ahora hacemos que el programa espere un par de segundos más (figura 63) y las luces cambien; por lo tanto, que se apague el led rojo que está en el pin 9 y se encienda el led verde que está en el pin 10 (figura 64).

Figura 63

Figura 64



Por último, solo nos queda esperar el tiempo que queramos que el semáforo esté en verde antes de completar el bucle (figura 65).

Figura 65

¡Ya tenemos nuestro semáforo de peatones con botón!

Cuestiones

¿Podríamos hacer que la luz verde parpadee antes de cambiar a rojo?

¿Podríamos combinar el semáforo de peatones con el semáforo de los coches y que vayan sincronizados?

3.3. Juegos del tipo «Clicker»

En esta práctica vamos a hacer un sencillo pulsador con el que podremos controlar una gran cantidad de juegos del tipo «Clicker».

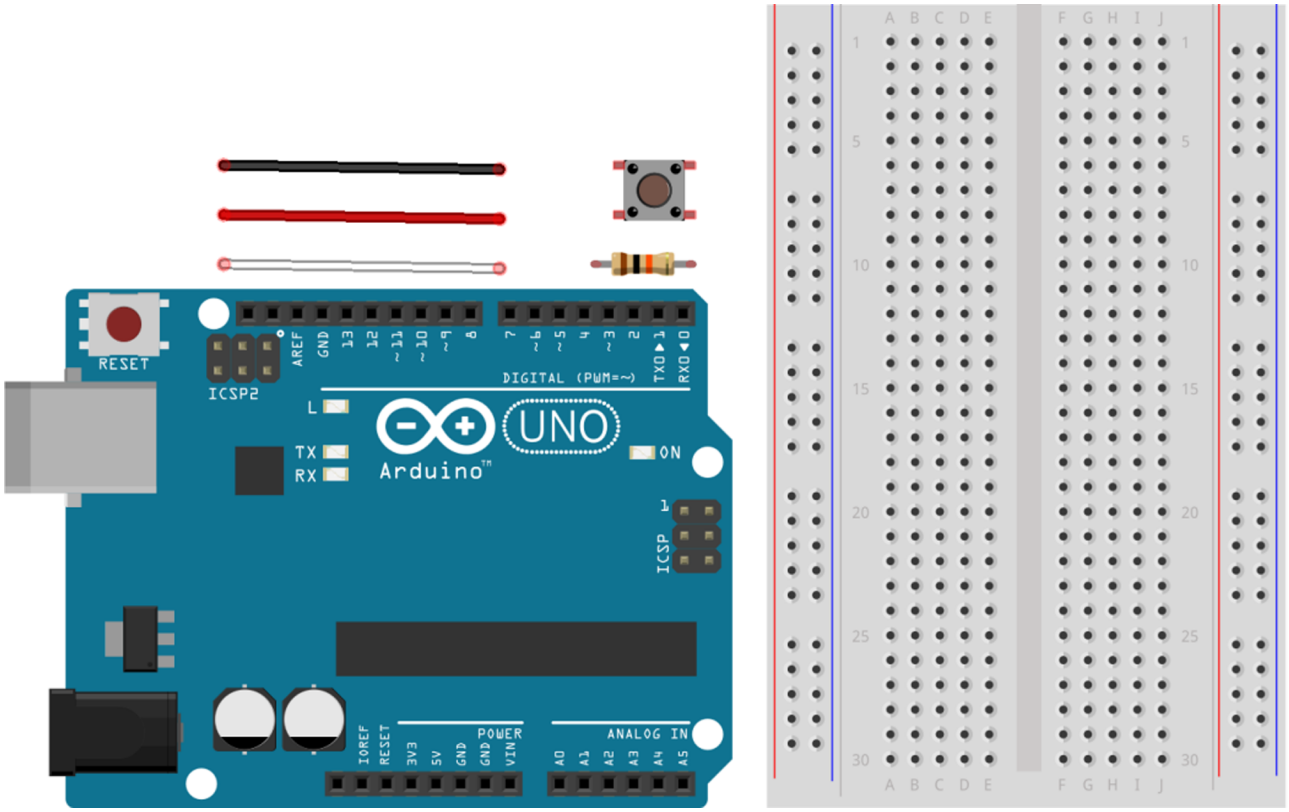
El mecanismo es muy sencillo, solamente necesitaremos (figura 66):

- La placa de Arduino
- 1 botón / pulsador
- 1 resistencia de 10 kΩ
- 1 placa de prototipado
- Cables

Observación

También podemos usar un módulo de botón en lugar del pulsador y la resistencia.

Figura 66

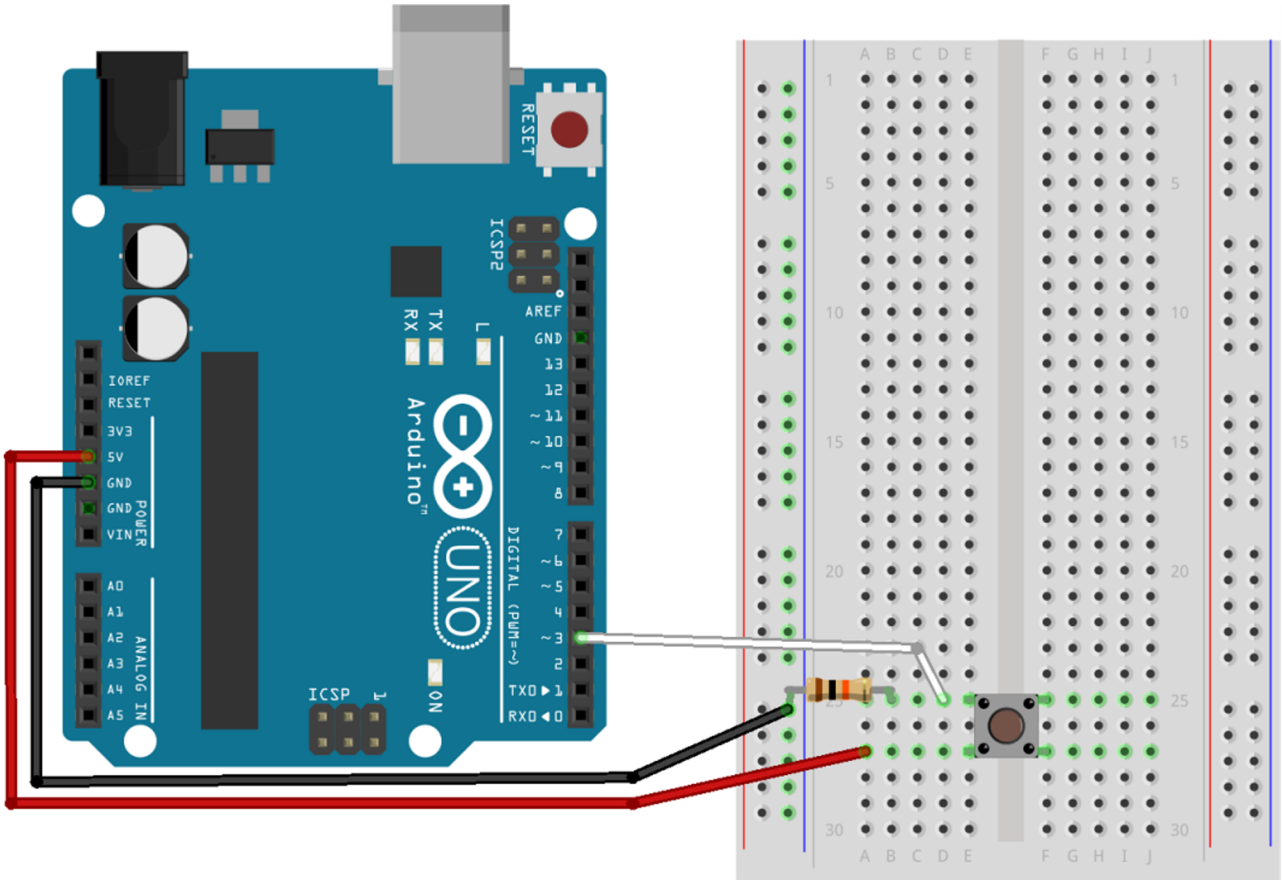


1) El montaje es muy sencillo (figura 67). De hecho, es igual que para el pulsador de la práctica anterior.

a) Solamente tendremos que colocar el pulsador en la placa de prototipado y montar un circuito divisor de tensión.

b) Conectamos el botón al pin 3 de la placa de Arduino.

Figura 67



2) Para este proyecto haremos un pequeño juego que consiste en un contador de diez segundos que guarde cuántos clics podemos hacer. Para ello necesitaremos usar el bloque *cronómetro* que se encuentra en el apartado de sensores. Es un bloque muy sencillo; tan solo tenemos que *reiniciar contador* al empezar el programa, y al ejecutarse el programa va contando el paso del tiempo automáticamente (figura 68).

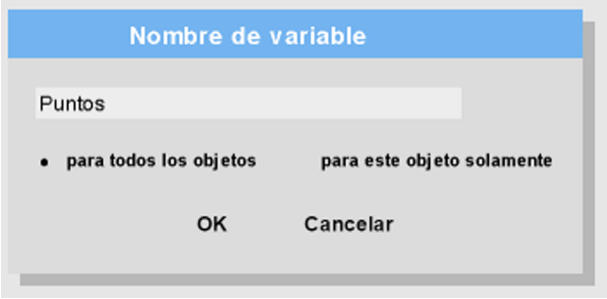
Figura 68

Para hacer que el programa cuente diez segundos, podemos usar el bloque *repetir hasta que...* con un condicional que establezca que el cronómetro sea igual a 10 (figura 69).

Figura 69

Ahora crearemos la variable *Puntos*, que contará los clics que hacemos (figura 70).

Figura 70



¡Recordad que tenemos que inicializar la variable a 0 al principio del programa (figura 71)!

Figura 71

A continuación queremos que el programa sume un punto cada vez que pulsemos el botón durante los diez segundos que el programa va a contar.

Para ello podríamos hacer que el programa sume 1 a *Puntos* cada vez que detecte la lectura digital del pin 3 (figura 72).

Figura 72

El problema al hacer esto es que el programa detectará que cada segundo pulsamos muchas veces el botón. Para solucionar este problema podemos esperar hasta que el botón deje de estar pulsado antes de continuar con el programa; así, cada clic se contará solamente una vez (figura 73).

Figura 73

¡Truco!

Si usamos el bloque *no* en estructura condicional, podemos ahorrar bloques.

Figura 74

Por último, podemos hacer que el programa nos diga cuántos puntos hemos conseguido usando el bloque *decir... por ... segundos* (figura 75). En Snap! hay un bloque muy útil que se llama *unir*. Este bloque nos permite unir diferentes textos para decirlos todos a la vez. Así, podemos enlazar el valor de una variable dentro del texto que queramos decir:

Figura 75

¡Ya tenemos un juego simple de tipo «Clicker»!

Cuestiones

¿Podríamos hacer que se vea la cuenta atrás en la pantalla mientras se cuenta hasta 10?

¿Podríamos guardar las mejores puntuaciones que hemos hecho?

¡Inténtalo!

3.4. Juego de reflejos

En este juego vamos a crear un medidor de reflejos en que debemos pulsar el botón lo más rápidamente posible justo en el momento en que un objeto cambia de color. Para esta práctica aprovecharemos el montaje del botón que hemos creado en la práctica anterior.

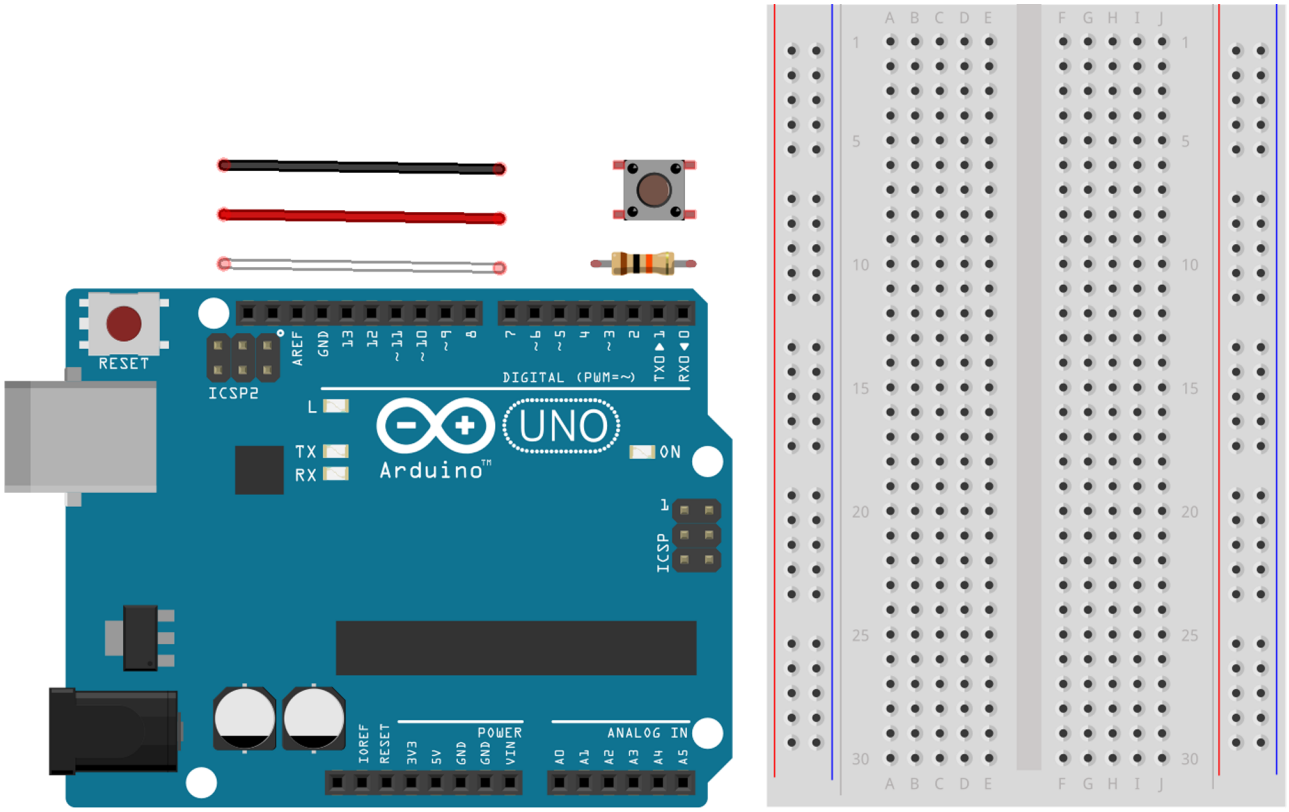
Observación

También podemos usar un módulo de botón en lugar del pulsador y la resistencia.

Necesitamos (figura 76):

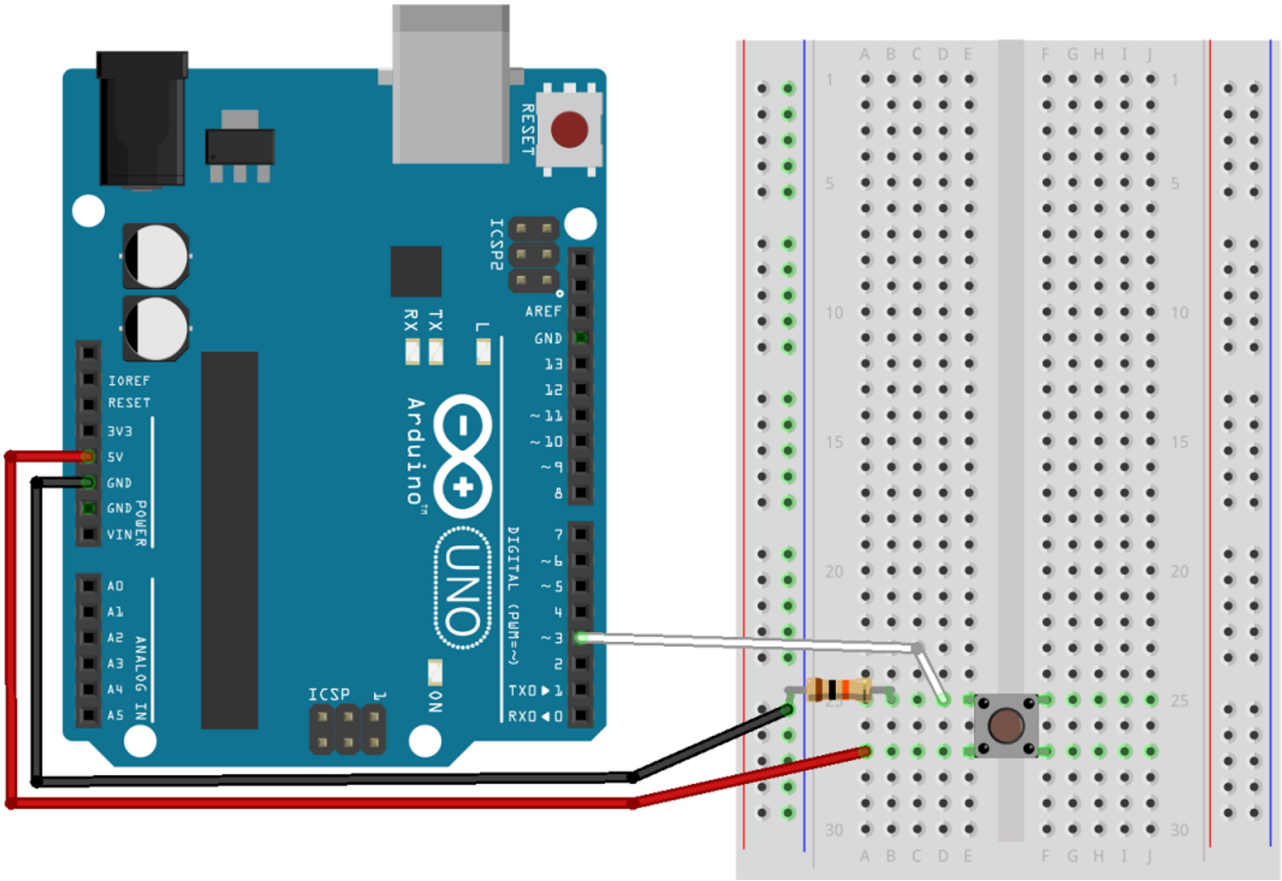
- La placa de Arduino
- 1 botón / pulsador
- 1 resistencia de 10 kΩ
- 1 placa de prototipado
- Cables

Figura 76



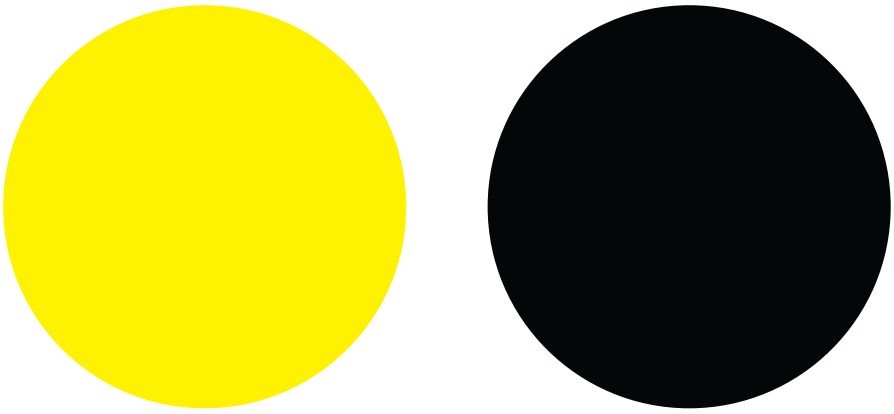
1) Colocamos el pulsador en la placa de prototipado y montamos un circuito divisor de tensión. Conectamos el botón al pin 3 de la placa de Arduino (figura 77).

Figura 77



2) Para programar el juego con Snap!, debemos calcular el tiempo que tardamos en pulsar el botón cada vez que hay un cambio. Crearemos dos «disfraces», uno de cada color –aunque también podemos hacerlo con el efecto «color», en vez de utilizar dos disfraces– (figura 78).

Figura 78



a) Empezamos con un bucle infinito (figura 79).

Figura 79

b) Creamos la variable *tiempo*, que nos va a contar el tiempo que tardamos en pulsar el botón (figura 80) y la inicializamos a 0 al principio del bucle (figura 81). Nos aseguramos de que está el disfraz correcto usando *cambiar el disfraz a ...* (figura 82).

Figura 80

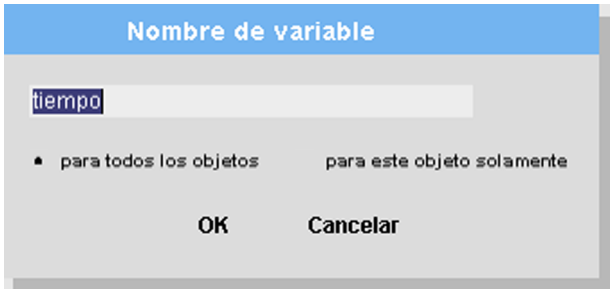
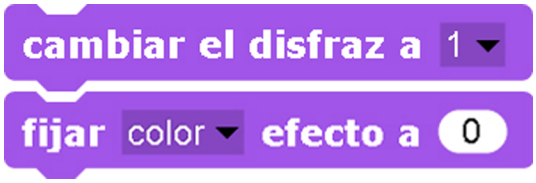


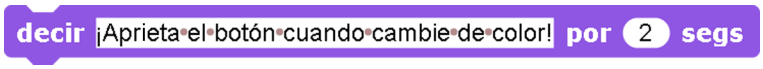
Figura 81

Figura 82



c) Ahora ya podemos explicar al jugador qué tiene que hacer (figura 83).

Figura 83



Para que sea más emocionante, vamos a hacer que el tiempo de espera hasta que el color cambie sea aleatorio. Para ello combinaremos los bloques *esperar* con *número al azar* (figura 84).

Figura 84

d) Después de esperar el tiempo al azar, cambiamos el disfraz para cambiar el color y reiniciamos el cronómetro (figura 85).

Figura 85

Al reiniciar el cronómetro, nos aseguramos de que empezamos a contar desde el momento en el que el color cambia.

Esperamos hasta que se pulse el botón; mientras tanto, el cronómetro continúa contando (figura 86).

Figura 86

e) Una vez que se haya pulsado el botón, guardamos el tiempo que ha contado el cronómetro con el bloque *fijar variable* (figura 87).

Figura 87

f) Por último, solo nos queda informar al jugador del tiempo que ha tardado en apretar el botón; para esto usaremos el bloque *decir*. Si queremos decir: «¡Has tardado (el tiempo que haya tardado) segundos!», lo haremos así (figura 88):

Figura 88

Si queremos que el jugador pulse una tecla –como la tecla de espacio, por ejemplo– antes de volver a empezar el juego, podemos hacer lo siguiente:

- Primero usamos el bloque *decir* para avisar al jugador de que debe pulsar la tecla de espacio para continuar (figura 89):

Figura 89

- Y esperamos hasta que el jugador pulse la tecla de espacio (figura 90):

Figura 90

- Para que desaparezca el mensaje anterior, volvemos a *decir*, pero sin decir nada.

Ya tenemos el funcionamiento general del juego, pero tiene un pequeño fallo... ¡Se puede hacer trampa!

Si el jugador mantiene el botón pulsado, o si lo presiona repetidas veces antes de que cambie de color, el programa entiende que tarda «0 segundos», y no hay ningún efecto negativo. Si queremos evitar que se pueda hacer esta trampa, podemos crear una variable, que vamos a denominar *click*, y controlar con ella si el botón se ha pulsado o no.

Creamos otro programa que simplemente pone la variable *click* en el estado «cierto» al pulsar el botón (figura 91) y en el bucle principal hacemos que la variable *click* se vuelva a fijar en «falso» cada vez que el bucle empiece (figura 92).

Figura 91

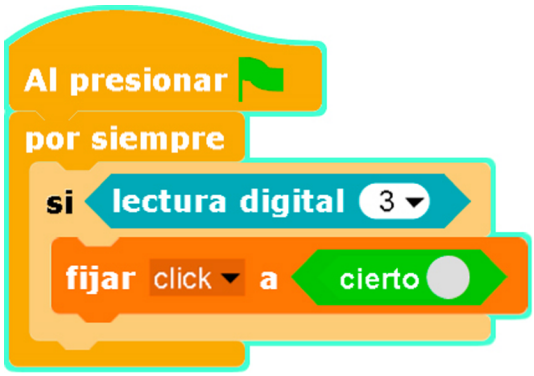


Figura 92

Por último, después de 

creamos una estructura condicional *si... si no*. Si se ha pulsado el botón –o sea, si la variable *click* está en el valor «cierto»–, avisamos al jugador de que el botón se tiene que pulsar solamente cuando cambie el color. Para ello, usamos el bloque *decir*. Y, de lo contrario (*si no*), que siga con el programa como hasta ahora. Pero dentro del bloque *esperar hasta que*, cambiamos la *lectura digital* (figura 93) por la variable *click* (figura 94).

Figura 93

Figura 94

¡Ya tenemos nuestro juego de reflejos totalmente funcional y a prueba de tramposos!

¡Truco!

Al tratarse de una variable booleana, que tiene como valores «cierto» o «falso», podemos usarlo directamente como condición de una estructura condicional, y así ahorrar bloques:

Figura 95

Otro juego que utiliza la misma mecánica del botón

¿Podemos pensar algún juego diferente que utilice la misma mecánica del botón?

Figura 96

Versión a prueba de trampas

Si pulsamos antes de tiempo, nos avisa de que tenemos que pulsar cuando cambie de color, y no antes.

Figura 97

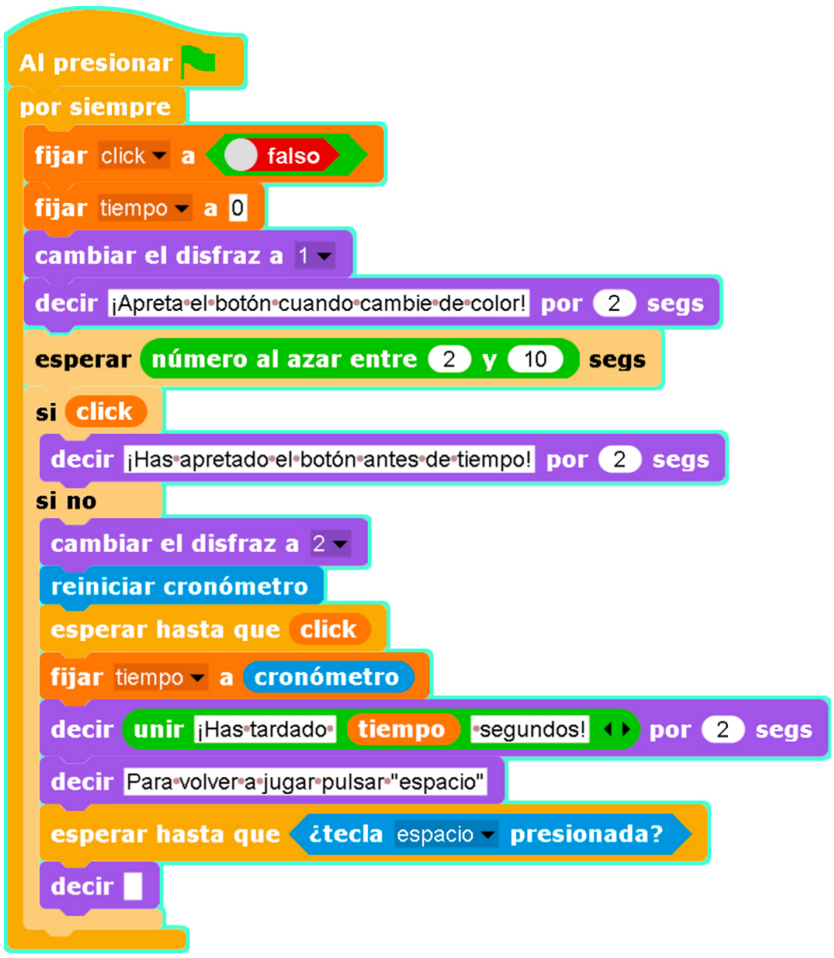
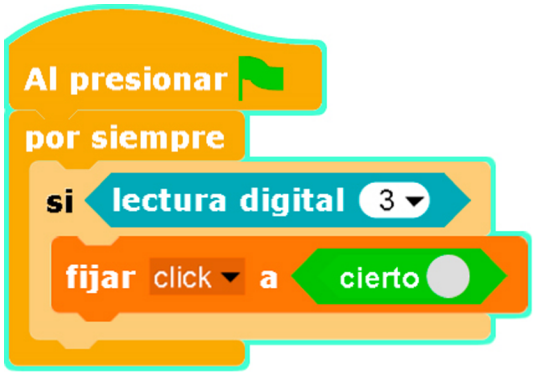


Figura 98



3.5. Coche con motores de tipo CC

En esta práctica vamos a hacer un coche que podamos mover usando dos motores de corriente continua. Los motores de corriente continua, o motores CC, son más económicos que los servomotores, pero necesitan un controlador de motores (o *driver* de motores).

1) El montaje es muy sencillo, simplemente necesitaremos (figura 99):

- Un chasis donde montar todos los elementos
- Una placa de Arduino
- Una placa controladora de motores (podemos utilizar la placa L9110, la L298 o la L293)
- Dos motores CC
- Una fuente de alimentación (pilas o batería)
- Cables
- Dos ruedas

¿Por qué necesitamos pilas o batería?

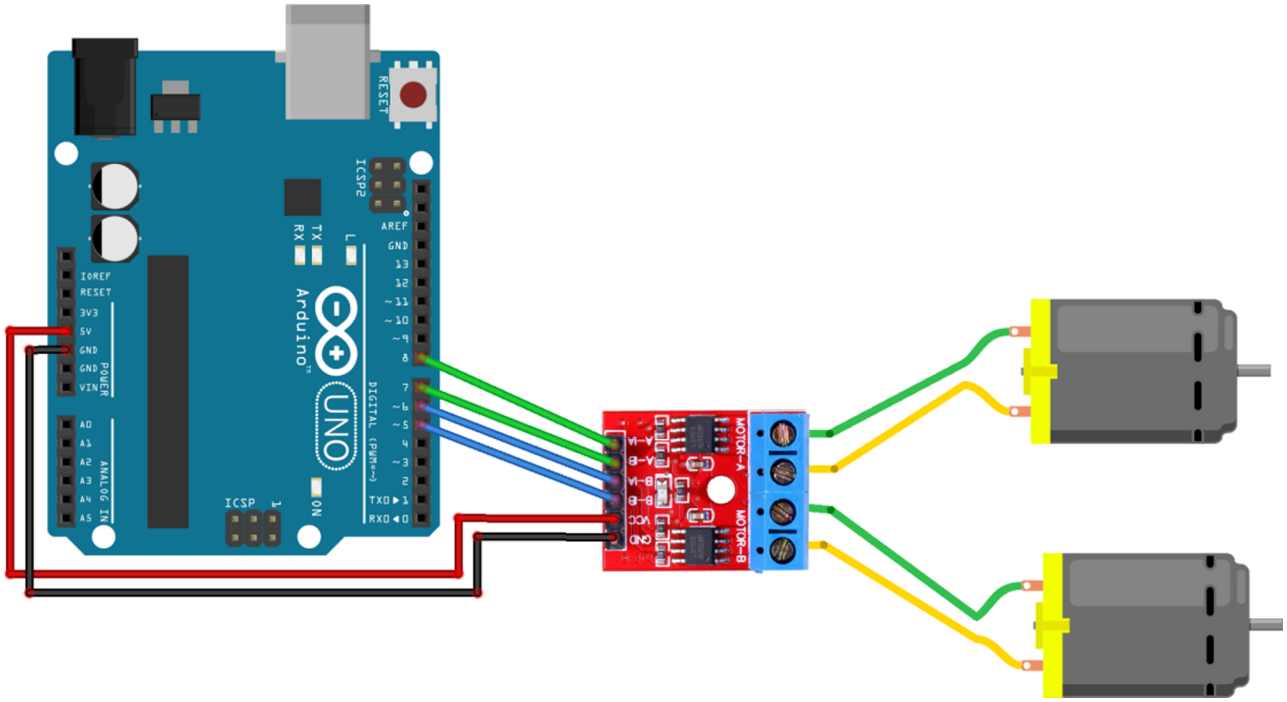
Este proyecto necesita alimentación externa porque es un robot que se mueve y no puede ir conectado al PC (a no ser que tengamos un cable muy largo). La conexión de la alimentación externa va en el conector de alimentación de la placa (el cuadradito negro que hay en la parte superior izquierda de la figura 99).

Para conectarlo todo, simplemente conectamos los pines de la placa controladora de motores a la placa de Arduino, de la manera siguiente:

- GND a tierra
- VCC a 5V
- A-A y A-B a los pines 8 y 7
- B-A y B-B a los pines 5 y 6

Por otro lado, conectamos el motor A y el motor B y aseguramos los cables usando los tornillos con un destornillador plano. Conectamos el motor A en las dos ranuras marcadas con MOTOR-A, y el motor B en las dos ranuras marcadas con MOTOR-B.

Figura 99



2) Una vez tenemos todo montado, podemos conectar la placa de Arduino al ordenador y empezar a programar.

En Snap4Arduino podemos empezar por programar el movimiento del robot usando las teclas de «flecha arriba» (↑), «flecha abajo» (↓), «flecha izquierda» (←) y «flecha derecha» (→) para controlar si se mueve hacia adelante, hacia atrás o hacia los lados, y la tecla espacio para que se quede quieto.

Como vimos, usamos dos pines digitales para controlar los motores CC; el motor girará en un sentido si encendemos uno de los pines y apagamos el otro, y girará hacia el lado contrario si invertimos estos pines (figura 100). Si paramos todos los pines, el motor estará parado.

Figura 100

Si quisiéramos usar servomotores en vez de hacer esta práctica con motores CC, simplemente deberíamos conectar los servomotores a la placa de Arduino y usar el bloque *fijar servo* (figura 101). Con este bloque podemos decidir el sentido (horario o antihorario), podemos parar el motor, e incluso podemos decidir la velocidad a la que queremos que se mueva (usando los ángulos).

Figura 101

3.6. Casa inteligente - Luces inteligentes

Esta es la primera de una serie de prácticas que tratan la domótica, que es la automatización de los elementos del hogar. Podríamos decir que se trata de crear «casas inteligentes».

En esta práctica vamos a controlar unas luces con un sensor de luz (fotorresistencia, o LDR) que medirá la luz del exterior, de modo que las luces se enciendan automáticamente al bajar el sol, y se apaguen durante el día.

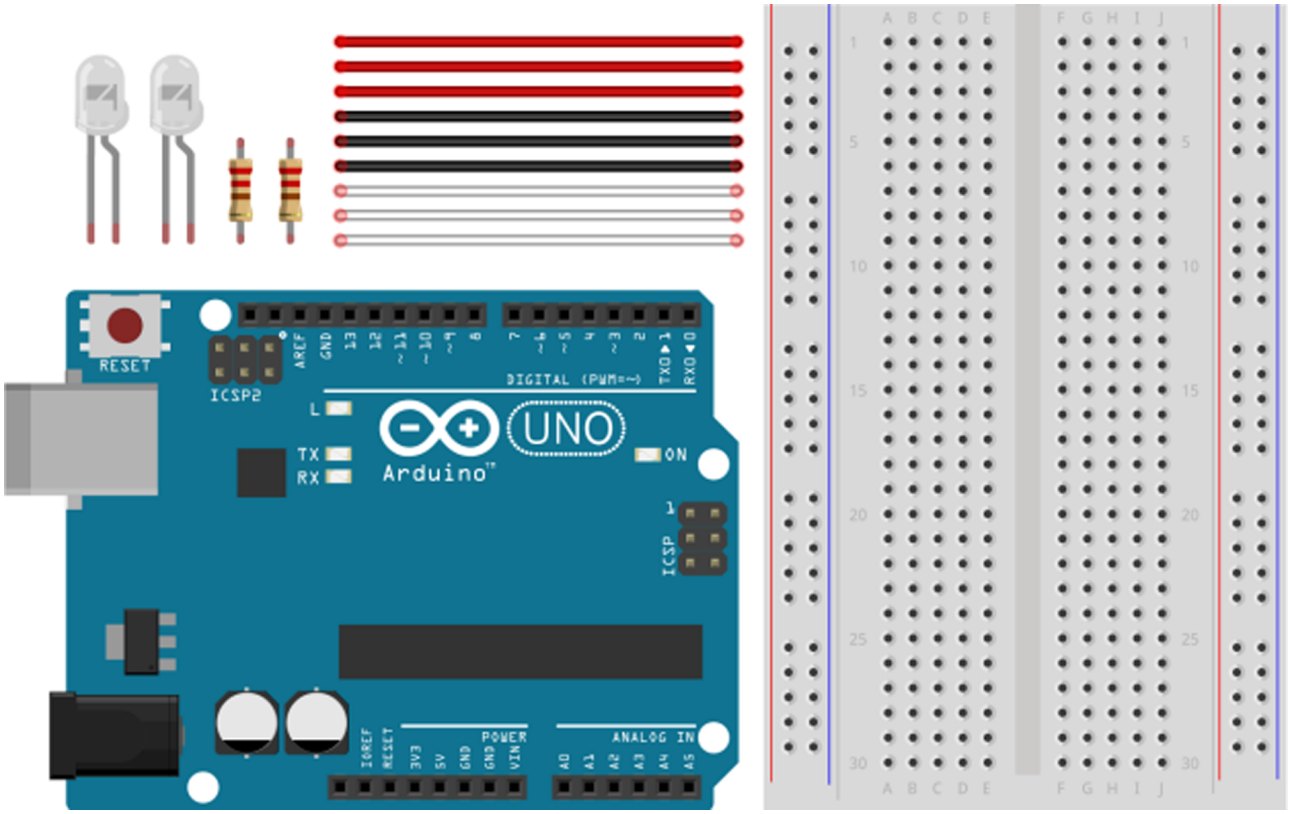
1) El material que usaremos es (figura 102):

- La placa de Arduino
- 2 leds de luz blanca
- 2 resistencias de 220 Ω
- 1 placa de prototipado
- Cables
- 1 fotorresistencia (módulo LDR)

Observación

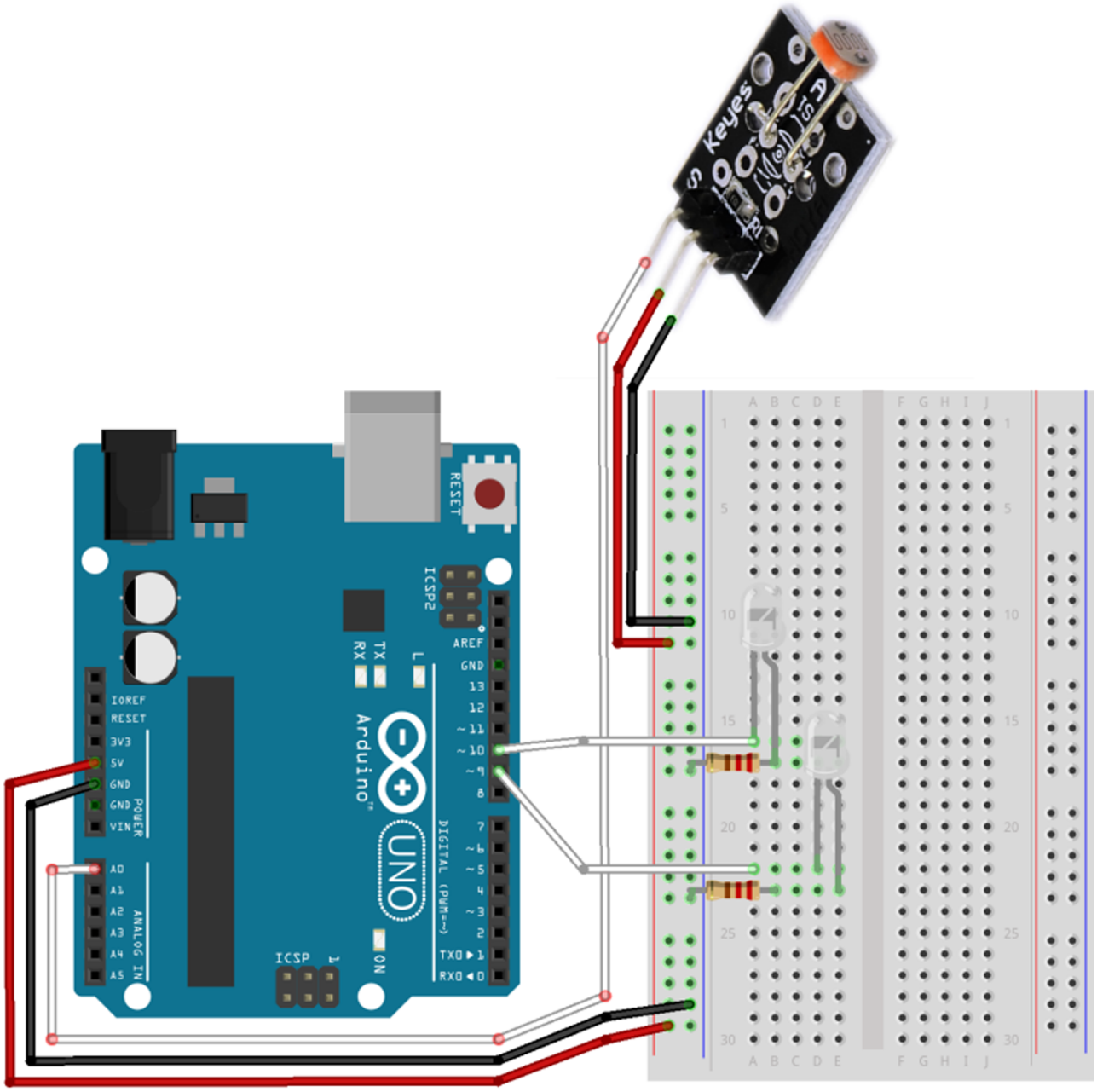
También podemos usar módulos LED en lugar de los leds y las resistencias.

Figura 102



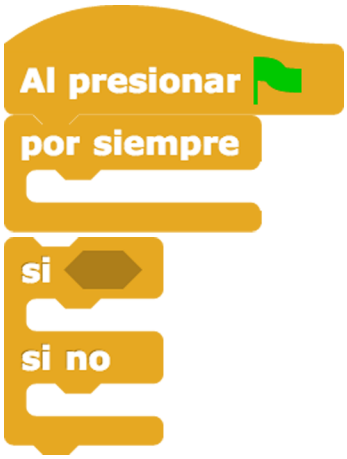
1) El montaje es muy simple (figura 103). Conectamos los leds a los pines 9 y 10, y el módulo LDR al pin analógico 0. Para conectar el módulo LDR, debemos conectar el pin marcado con la letra S al pin analógico que queramos usar; en nuestro caso, el A0. El pin del módulo marcado con un – va conectado a Tierra, y el pin central va conectado a 5V.

Figura 103



2) El programa debe consistir en un bucle infinito que compruebe si el sensor recibe luz, o no. Si recibe luz, le diremos que apague las luces; y si no, que las encienda (figura 104).

Figura 104



Podemos dar un valor de sensibilidad a la lectura analógica para establecer a partir de qué valor se activarán las luces. En nuestro caso empezaremos probando con 500 y viendo si funciona bien; o, si necesitamos más o menos sensibilidad, simplemente cambiamos este valor (figura 105).

¡Ojo!

El sensor LDR tiene un valor mayor cuando hay menos luz. Hay que tener esto en cuenta para que las luces se enciendan cuando el valor de la lectura analógica sea mayor.

Figura 105

Cuestiones

¿Qué otros usos podríamos dar al sensor LDR?

Por último, encendemos y apagamos los leds conectados en los pines digitales 9 y 10 (figura 106).

Figura 106

Ya tenemos una luz automática, el primer paso para crear nuestra casa inteligente.

3.7. Luz de fiesta con un módulo LED RGB

Este es un proyecto muy simple. Programaremos un módulo de leds tricolor para que las luces de una sala de fiestas cambien aleatoriamente cada segundo.

Para ello necesitaremos:

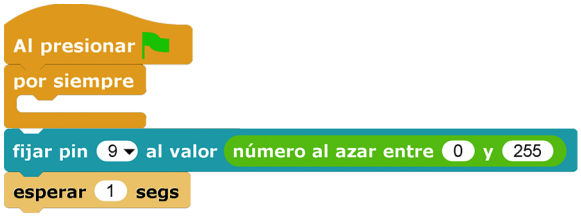
- La placa de Arduino
- 1 módulo LED RGB (figura 107)
- Cables

Figura 107

1) La conexión es muy sencilla. Conectamos los pines del módulo LED RGB de la manera siguiente: el pin marcado con –, a GND de nuestra placa de Arduino; y los pines R, G y B, a las conexiones 9, 10 y 11 de la placa de Arduino.

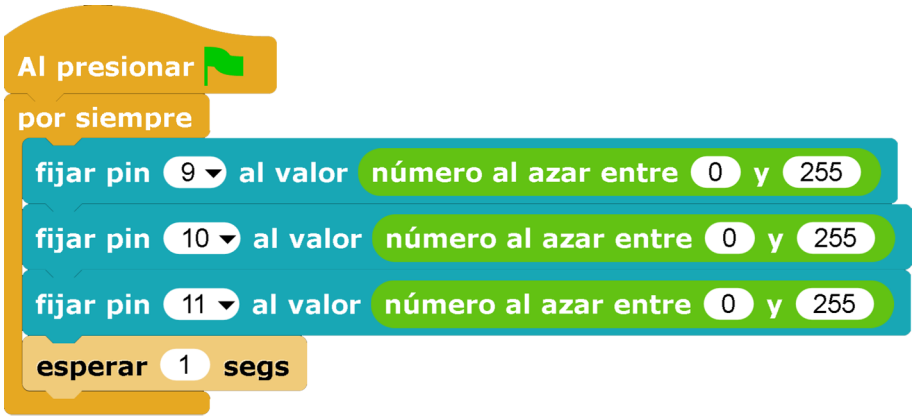
2) En Snap4Arduino crearemos un bucle *por siempre* y asignaremos un valor al azar entre 0 y 255 al pin 9 (figura 108).

Figura 108



Por último, duplicamos el bloque *fijar pin* para los pines 10 y 11 (figura 109).

Figura 109



¡Y ya tenemos nuestra luz de discoteca!

Cuestiones

¿Podríamos hacer que la luz cambiara de color al pulsar un botón, en vez de hacerlo cada segundo?

3.8. Alarma antirrobo

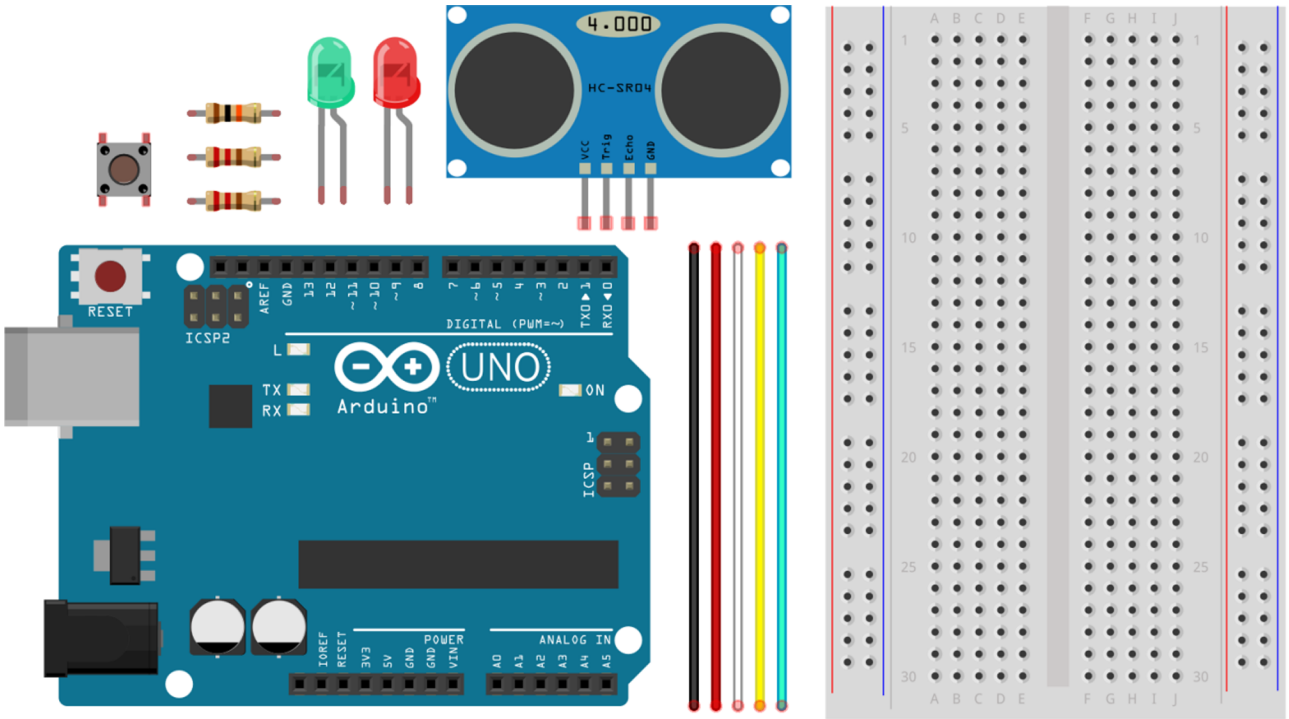
En esta práctica crearemos una alarma que nos avisará si alguien mueve nuestros objetos más preciados. Vamos a aprender cómo usar los sensores de distancia para crear un indicador con leds que nos avisará si alguien toca nuestras pertenencias.

El material que vamos a usar es el siguiente (figura 110):

- 1 sensor de distancia
- La placa de Arduino
- 2 leds: uno rojo y uno verde

- 2 resistencias de 220 Ω
- 1 botón
- 1 resistencia de 10 kΩ
- 1 placa de prototipado
- Cables

Figura 110



Como para esta práctica vamos a usar el sensor de ultrasonidos, debemos instalar el *firmware* adecuado en la placa de Arduino e importar los bloques adecuados en Snap4Arduino.

Ved también

Consultad la instalación del *firmware* adecuado en la placa de Arduino que se hace en el subapartado 2.5.5 de este módulo didáctico.

Vamos a poner la alarma cerca de nuestros objetos más preciados. Al pulsar un botón, activamos la alarma y se enciende el led verde. A partir de ese momento, si alguien mueve nuestro objeto, ¡la alarma se activará y se encenderá el led rojo!

1) Para el montaje, daremos los pasos siguientes (figura 111):

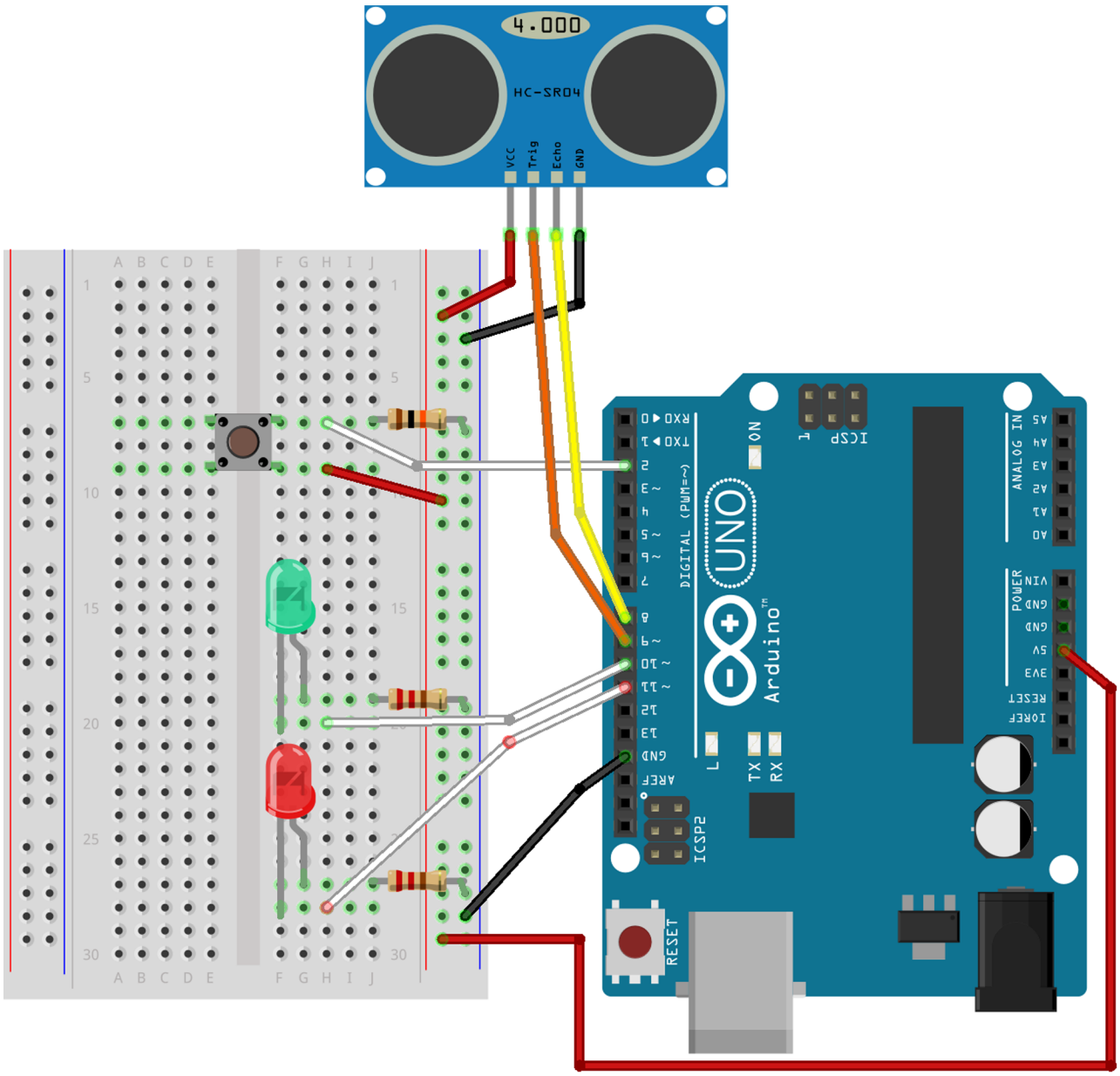
a) Conectar los leds a los pines 10 y 11 usando una resistencia de 220 Ω para controlar el voltaje.

b) Conectar el botón al pin 2 usando la resistencia de 10 kΩ para crear un circuito divisor de tensión.

c) El sensor de ultrasonidos tiene cuatro pines. Conectamos el pin VCC del sensor a 5V, el pin GND del sensor a tierra, el pin Trig del sensor al pin 9 de Arduino, y el pin Echo del sensor al pin 8.

d) Por último, conectamos la placa al ordenador, y ya podemos empezar a programar.

Figura 111



2) Empezamos el programa con un bucle principal con el bloque *repite por siempre*. Esto hace que nuestro programa se ejecute todo el tiempo (figura 112).

Figura 112

Ahora vamos a hacer dos cosas.

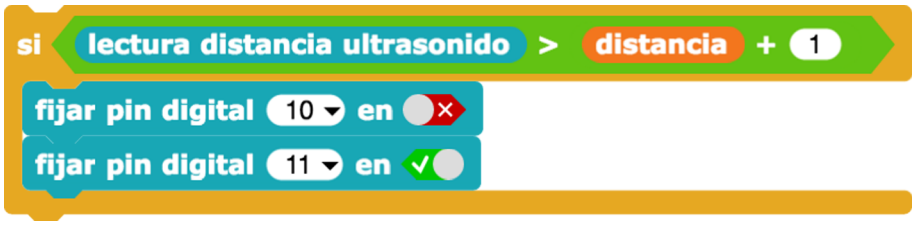
a) En primer lugar, programamos que, al pulsar el botón de la placa, se guarde la distancia a la que se encuentra el objeto y se encienda el led verde para mostrar que la alarma está activa. Para esta tarea creamos una variable *distancia* a la que se asigna (=) el valor que recibe el sensor de distancia al apretar el botón conectado al pin número 2 (figura 113).

Figura 113

b) En segundo lugar tenemos que hacer que la alarma se active si el objeto se mueve. Para ello crearemos dos condiciones: una para ver si el objeto se ha acercado, y otra para ver si se ha alejado.

La primera condición tiene que ser: si la distancia que recibe el sensor conectado al pin número 8 de la placa es mayor que la variable distancia más 1 (sumamos 1 a la variable para que tenga un poco de margen, ya que el sensor es tan preciso que, si no se le suma nada, podría activar la alarma sin querer). Entonces encendemos el led rojo con el bloque *fijar pin digital* en el pin número 11, donde hemos colocado el led rojo (figura 114).

Figura 114



¡Ya tenemos nuestra propia alarma de seguridad!

4. Prácticas de dificultad media

Continuamos con algunas aplicaciones de Arduino algo más complejas:

1) Dado digital

2) Juego del pimpón

3) Piano con zumbadores

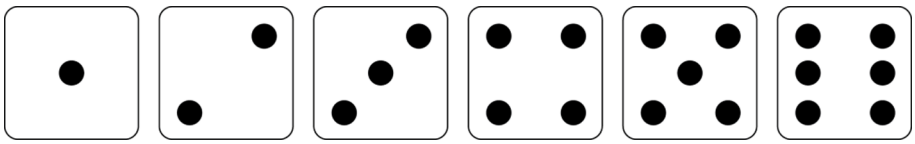
4) Pizarra con dos potenciómetros

5) Variación difícil de la luz de fiesta con un módulo LED RGB

4.1. Dado digital

En esta práctica crearemos un dado que funcione utilizando leds. Usaremos siete leds para poder crear todas las combinaciones que hay en un dado (figura 115).

Figura 115



Para esta práctica necesitaremos (figura 116):

- Una placa de Arduino
- 1 placa de prototipado
- 7 leds
- 7 resistencias de 220 Ω
- Cables

Debemos colocar los leds de manera que tengan la forma de todas las combinaciones que pueden hacer las caras de los dados. Una vez colocados, hay que conectar las patas cortas de los leds con una resistencia a tierra y las patas largas de los leds a los pines de la placa de Arduino que vamos a usar. Nosotros usaremos los pines del 6 al 12 (figura 117).

Figura 116

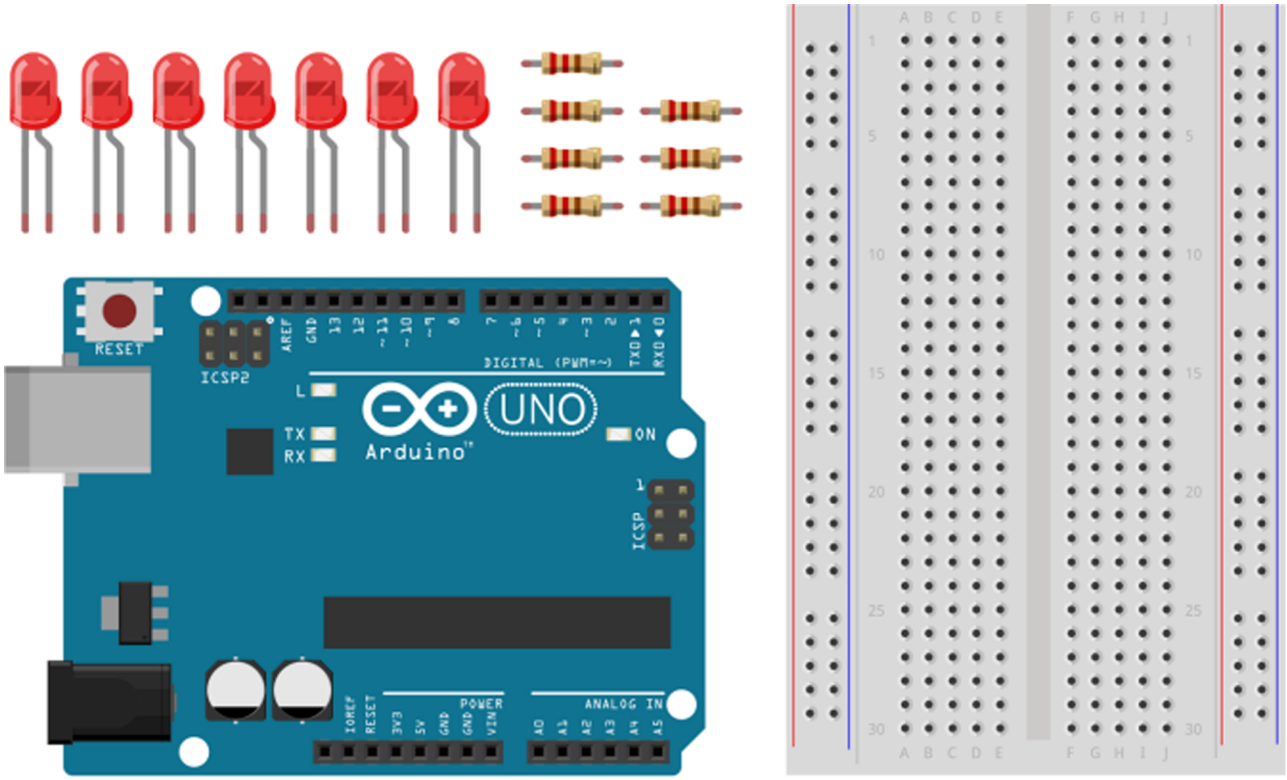
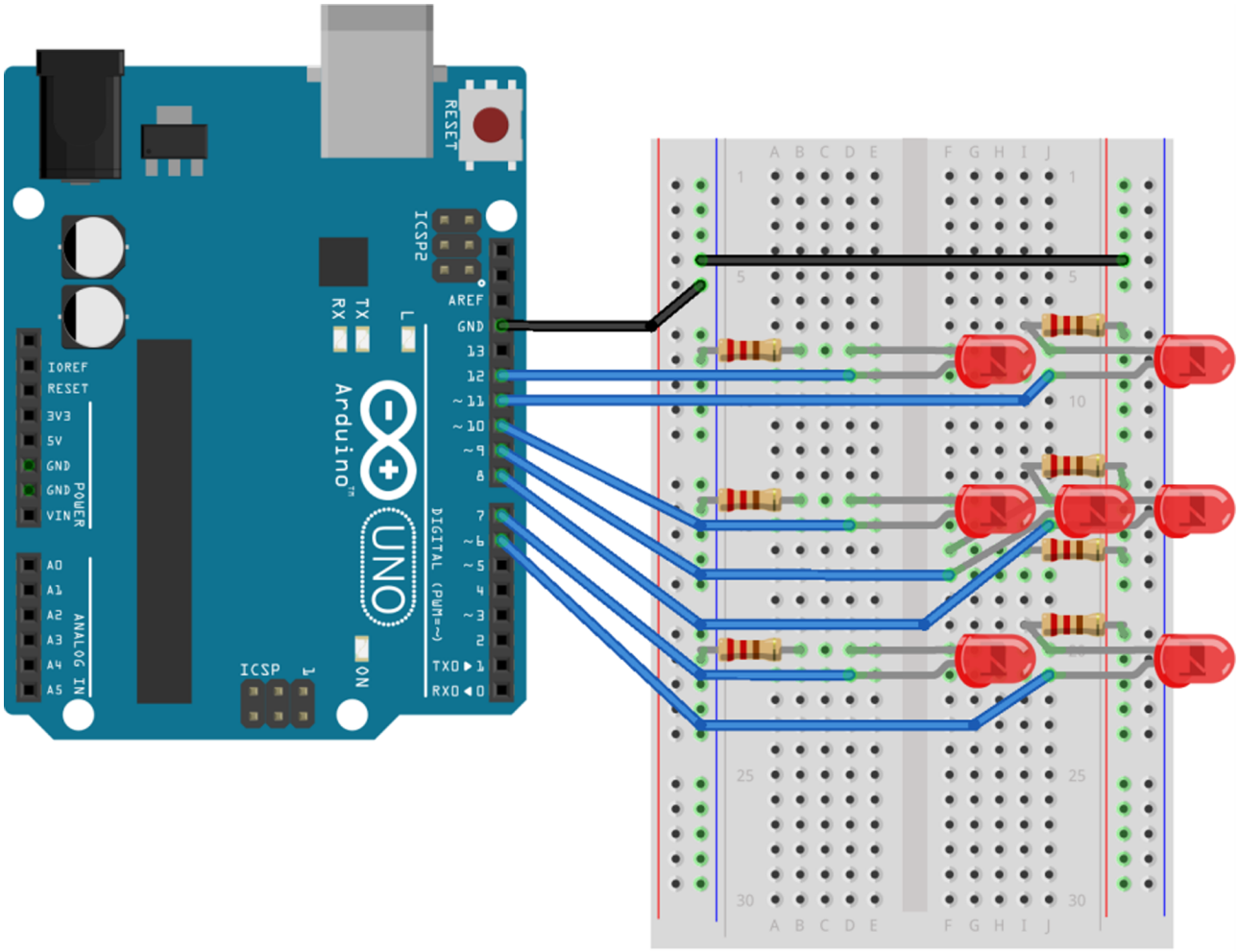

RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN

Figura 117


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX0 ▶1
RX0 ◀0
DIGITAL (PWM=~)
L
TX
RX
Arduino
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
A B C D E
F G H I J
1
5
10
25
30

Una vez acabado el montaje, podemos empezar a programar. La manera más sencilla de programarlo es parecida al modo como se controla el indicador de siete segmentos. Podemos usar seis disfraces –uno para cada cara del dado– para ver en Snap4Arduino el resultado de cada tirada del dado (figura 118).

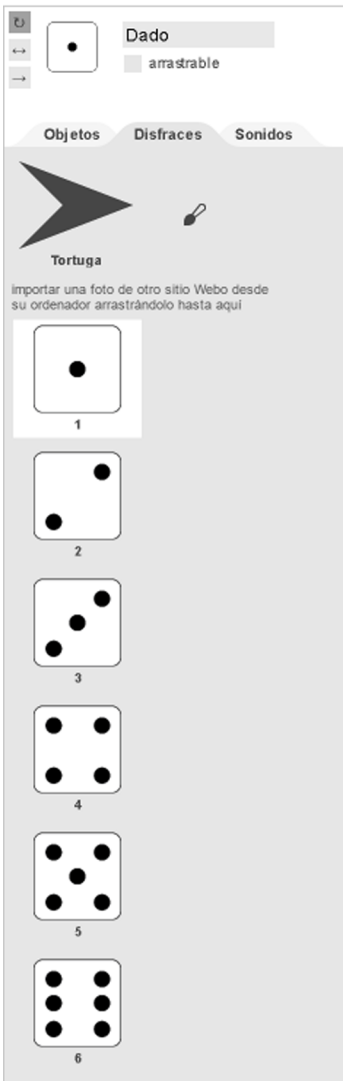
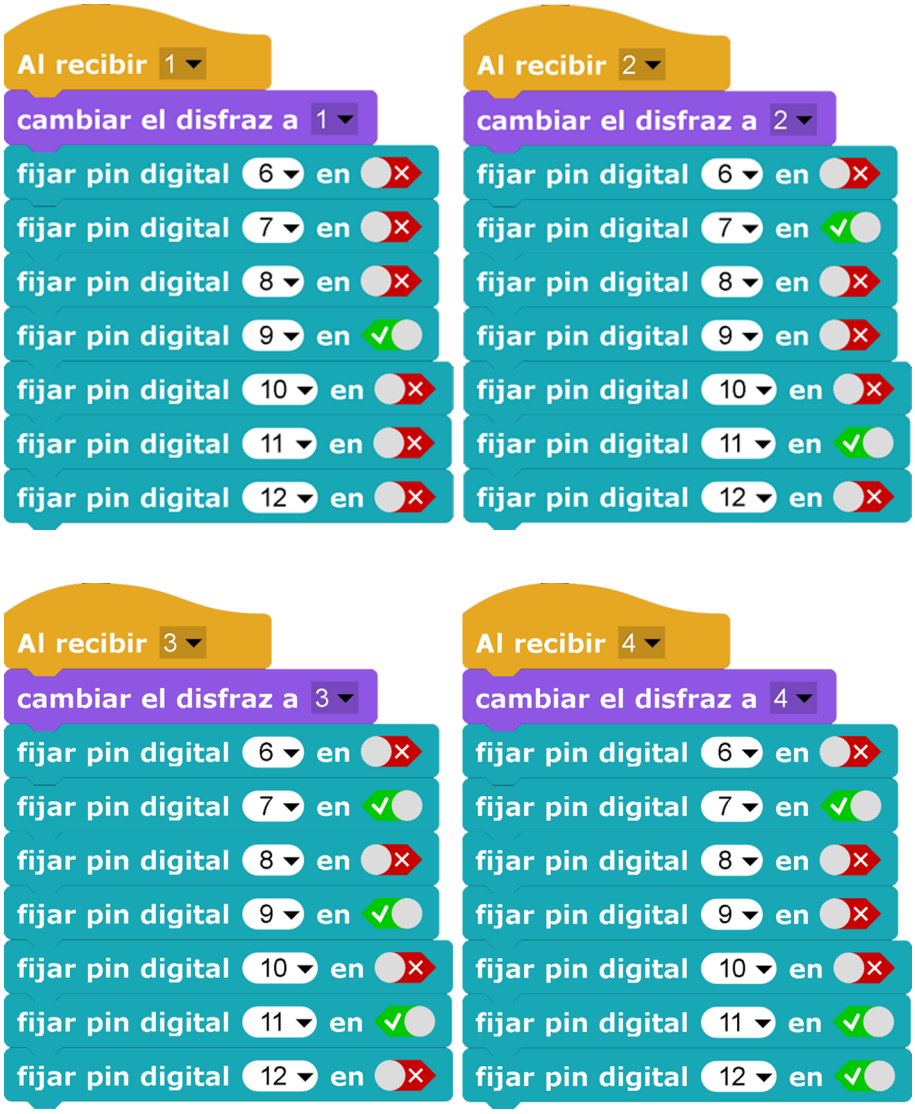
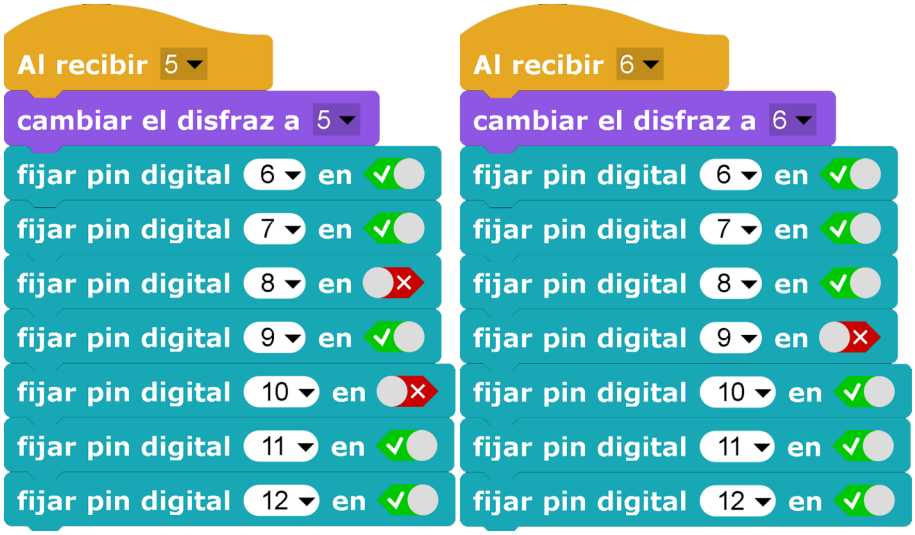


Figura 118

En cada mensaje se hará un cambio de disfraz al disfraz que se corresponda con cada mensaje, y asignamos los valores para las salidas digitales (figura 119).

Figura 119





Por último, hacemos que al pulsar una tecla se envíe un mensaje al azar entre el 1 y el 6 (figura 120).

Figura 120

¡Ya tenemos un dado!

Cuestiones

¿Cómo podríamos usar un botón externo en vez de la tecla de espacio?

¿Podríamos añadir emoción haciendo aparecer una animación aleatoria antes de que aparezca el número?

4.2. Juego del pimpón

En esta práctica crearemos nuestra versión del juego Pong, ¡uno de los primeros juegos de la historia! Usaremos dos potenciómetros para mover las dos raquetas. Guardaremos las puntuaciones de los dos jugadores y daremos un aviso de ganador cuando acabe la partida.

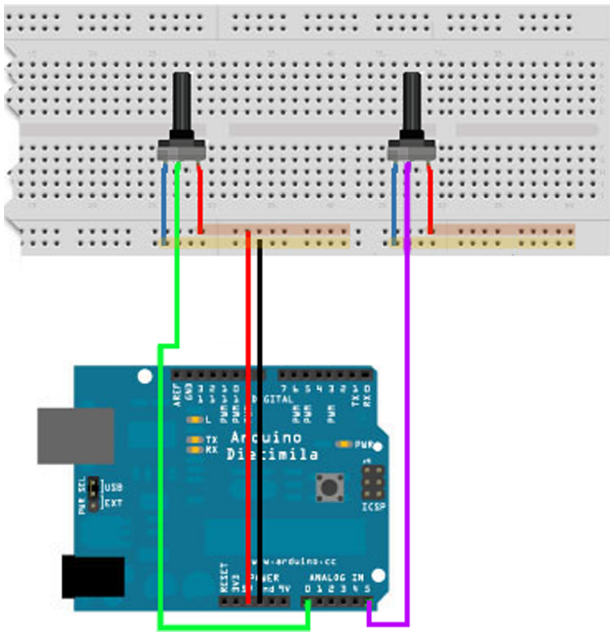
Componentes:

- La placa de Arduino
- 1 placa de prototipado
- 2 potenciómetros

El código de Scratch es un poco más complejo que en otros ejemplos. En este caso hay cinco variables; cuatro objetos con *scripts* (dos raquetas, una pelota y el mensaje ganador); y el fondo, que también tiene un *script*.

La conexión de los potenciómetros es muy simple. Conectamos las patas que van a tierra y a 5V, y la pata central a los pines de entrada analógica que nosotros elijamos, en nuestro caso A0 y A5 (figura 121).

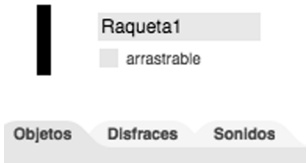
Figura 121



Ahora empezamos a crear el juego en Snap4Arduino. El primer paso será crear el objeto **raqueta**, que vamos a dibujar pulsando en el icono de «dibujar un nuevo objeto», .

Dibujamos nuestra raqueta y renombramos el objeto a *Raqueta1* (figura 122).

Figura 122

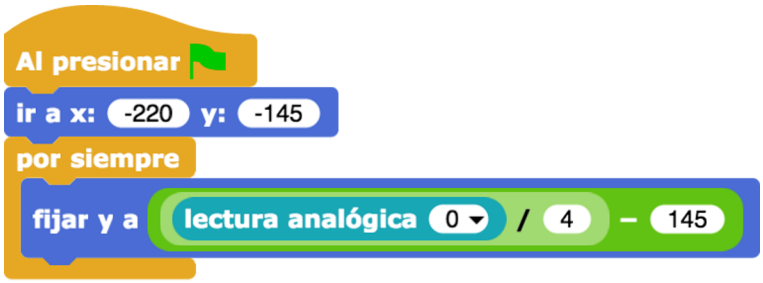


Son dos objetos que utilizan la misma conexión a la placa, es decir los dos potenciómetros están conectados a la misma placa. La imagen de estos objetos es una barra vertical negra que ocupa aproximadamente una quinta parte de la altura del campo de juego. El código es sencillo, ya que solo controla el giro del potenciómetro para mover la raqueta.

Al inicio del programa colocamos la raqueta en la posición inicial con el bloque *ir a...*; la coordenada *x* no cambiará, y la coordenada *y* cambia según se mueve el potenciómetro.

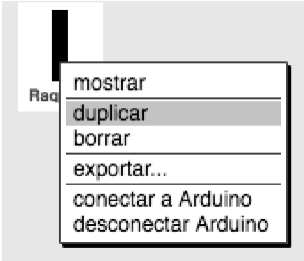
El potenciómetro conectado a la entrada analógica (la 0 para una raqueta y la 5 para la otra) genera valores entre 0 y 1.023, y este valor debe transformarse en el desplazamiento de la raqueta; en este caso, para calcular la relación con respecto del desplazamiento máximo, se divide por 4 y se resta 145 (figura 123).

Figura 123



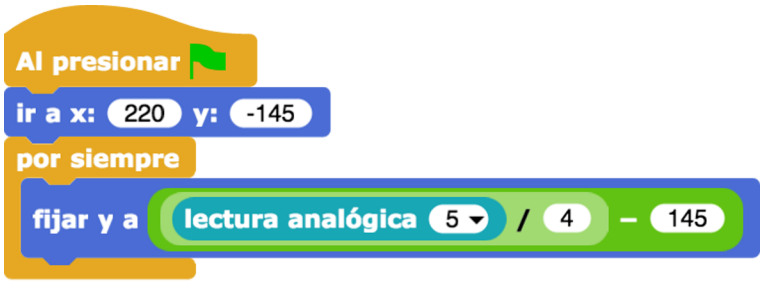
Primero haremos todo el programa de una raqueta y después lo duplicaremos para hacer la segunda (figura 124).

Figura 124



Cambiamos la posición en *x* de la segunda raqueta y la lectura analógica al pin 5 (figura 125), y ¡ya tenemos las dos raquetas que funcionan con su potenciómetro!

Figura 125

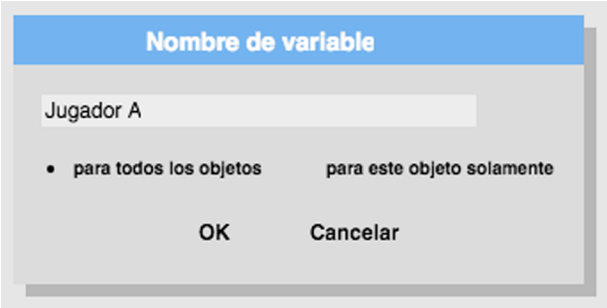


Ahora vamos a hacer el **fondo**. Para ello, pintamos una línea de un color diferente en cada lado, por ejemplo, rojo y azul, que van a ser los colores de cada equipo (figura 126). Estas líneas de colores nos servirán para detectar cuándo se marca un gol, y que hay que sumar un punto al jugador que lo marca.

Figura 126

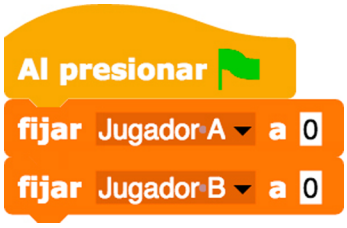
Ahora crearemos dos variables (*Jugador A* y *Jugador B*) con la opción «crear variable» en el apartado «variables» (figura 127).

Figura 127



Empezamos el programa del fondo fijando las variables a 0 (figura 128).

Figura 128



A continuación introducimos un bucle que compruebe si alguno de los dos jugadores tiene más de tres puntos. Si alguno tiene más de tres puntos, tendremos que *enviar mensaje Ganador A* o *Ganador B* para terminar el juego (figura 129).

Figura 129

Si añadimos el bloque *stop... este script*, evitaremos que sigan sumándose puntos por error.

Por último crearemos el objeto **pelota** de la misma manera que creamos las raquetas, pero esta vez dibujaremos un círculo pequeño (figura 130).

Figura 130

Este objeto es un poco más complicado, ya que ejecuta varias acciones a la vez, pero aun así es muy fácil de programar. ¡Solamente tenemos que ir por partes!

Primero, al empezar el programa, hacemos que la pelota se coloque en el centro de la pista usando el bloque *ir a...* y colocándolo en (*x*: 0, *y*: 0) (figura 131).

Figura 131

A continuación le diremos que apunte hacia una dirección aleatoria. Para ello usaremos los bloques *apuntar en dirección...* y *número al azar...*. Como queremos que vaya en cualquier dirección, le diremos que el número al azar sea entre 0 y 360, que son los grados que hay en una circunferencia (figura 132). ¡Así puede empezar apuntando en cualquier dirección!

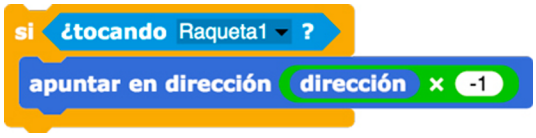
Figura 132

El paso siguiente es hacer que la pelota se mueva. Para ello usaremos el bloque *mover* y le daremos una velocidad que creamos conveniente, por ejemplo, 10. Como ya sabemos, para que se mueva todo el rato, debemos hacer un bucle. Para que rebote al tocar un borde hay un bloque llamado *rebotar si está tocando un borde* (figura 133). Más fácil que eso, ¡imposible!

Figura 133

Ahora vamos a hacer que también rebote al tocar las raquetas. Primero haremos la raqueta número 1 y después duplicaremos el programa y cambiaremos lo necesario para la raqueta número 2. Debemos crear un condicional con el bloque *si...* y la condición *tocando la Raqueta1*. Dentro del condicional haremos que cambie el sentido, para ello multiplicaremos la dirección por –1 (figura 134).

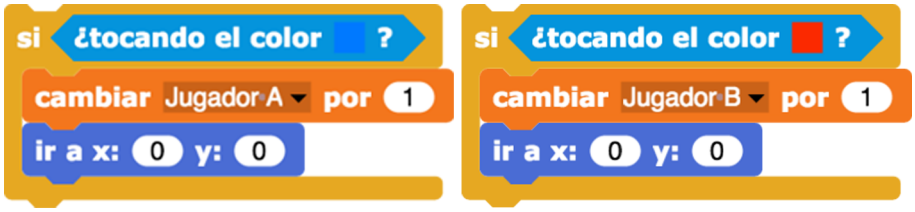
Figura 134



Debemos hacer lo mismo para la *Raqueta2*.

Solo nos queda hacer que la pelota vuelva a aparecer en el centro y el programa sume un punto al jugador que ha marcado cuando la pelota llega a alguna de las dos porterías. Para ello usaremos un condicional que compruebe *si...* la pelota *toca el color...* azul para saber si ha marcado el *Jugador1*, y *si...* la pelota *toca el color...* rojo para saber si ha marcado el *Jugador2* (figura 135).

Figura 135



Por último, solo queda hacer que el juego pare en el momento en que alguno de los jugadores marque el cuarto gol (figura 136).

Figura 136

¡Ya tenemos nuestro juego de pimpón!

Cuestiones

¿Podríamos complicar un poco más el juego haciendo que la pelota cambie de ángulo al azar al rebotar? ¿Y que la velocidad de la pelota aumente con el tiempo?

4.3. Piano con zumbadores

En esta práctica haremos un piano para poder tocar música usando el teclado del ordenador y un zumbador.

Preparamos la placa con el zumbador de modo parecido a como se hizo para la práctica del zumbador.

Ved también

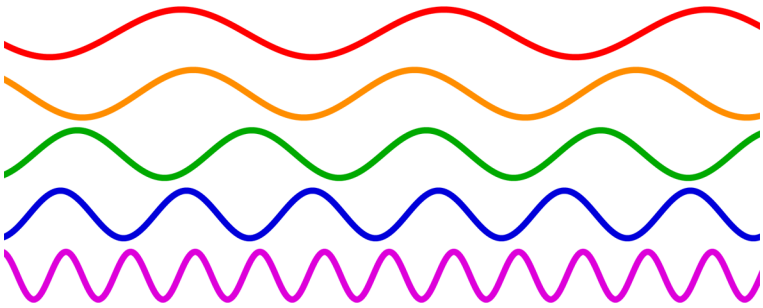
La práctica de los zumbadores, la instalación del *firmware* BuzzerFirmata y la importación de los bloques BuzzerBlocs se explican en el subapartado 2.4.4 de este módulo didáctico.

¡Recordad que para poder usar los zumbadores tenemos que instalar el BuzzerFirmata en la placa de Arduino e importar los bloques BuzzerBlocs!

2) Una vez que hemos preparado la placa, hemos conectado el zumbador y hemos importado los bloques en Snap4Arduino, podemos empezar a crear nuestro programa para controlar el piano.

Para que el zumbador suene, tenemos que decirle qué frecuencia queremos que reproduzca. ¿Qué quiere decir esto? Muy fácil: El sonido está compuesto de ondas de presión que se repiten. Estas ondas de presión pueden repetirse más rápidamente o más lentamente. El sonido que generan es agudo si las ondas se repiten rápidamente, y grave si se repiten lentamente. La cantidad de veces que se repite una onda por cada segundo es lo que llamamos *frecuencia* (figura 137). Los humanos podemos oír en un rango de frecuencias entre 20 Hz y 20.000 Hz. Aunque el rango de frecuencias que podemos oír se va reduciendo cada vez más con la edad.

Figura 137



Podemos buscar cuál es la frecuencia de cada nota para poder recrear con un zumbador las notas musicales de un piano.

Nota	Frecuencia
Do	261.63
Re	293.66
Mi	329.63
Fa	349.23
Sol	392.00
La	440.00
Si	493.88
Do	523.25
Re	587.33
Mi	659.26
Fa	698.46
Sol	783.99
La	880.00
Si	987.77

También podemos usar mensajes para ver qué nota se pulsa con cada tecla, de modo que al pulsar la tecla, el programa envíe un mensaje con la nota que se está tocando, y para cada nota creamos un programa que recree la frecuencia de la nota (figura 138).

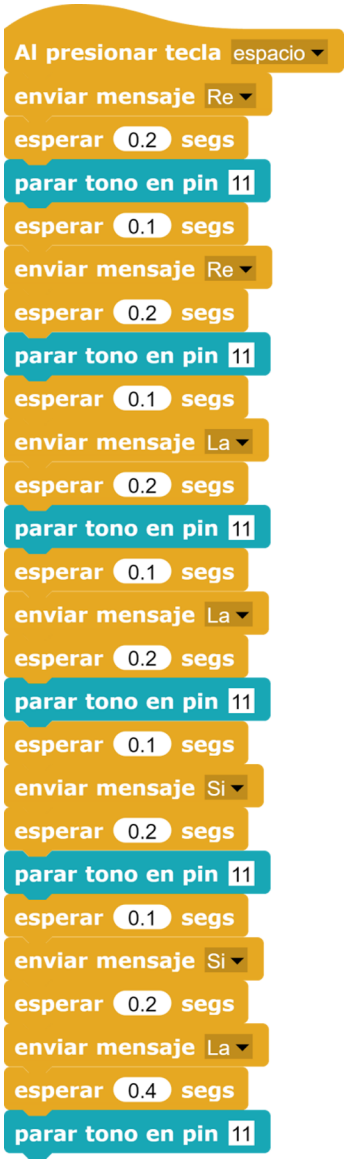
Figura 138

¡Recordad que es muy importante parar los tonos con el bloque *parar tono* (figura 139) porque de otro modo el zumbador no parará nunca de sonar y puede ser molesto mantener un tono demasiado tiempo!

Figura 139

Ahora que dominamos la programación de las notas, podemos hacer todo el teclado y tocar nuestras canciones preferidas o componer canciones nuevas. O incluso hacer un programa que toque canciones automáticamente usando los mensajes de cada nota combinados con las esperas del tiempo que queramos que dure cada nota y los bloques *parar tono* para los silencios (figura 140).

Figura 140



4.4. Pizarra con dos potenciómetros

En esta práctica vamos a recrear una pizarra del tipo «Etch a Scketch», que dibuja líneas moviendo dos potenciómetros (figura 141).

Figura 141
By Etcha (Own work) [CC BY-SA 3.0 or GFDL], via Wikimedia Commons.

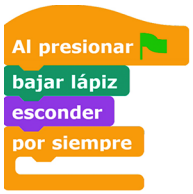
Para ello solamente necesitaremos:

- Una placa de Arduino
- 2 potenciómetros
- Cables

El montaje es muy simple; solamente hace falta conectar los potenciómetros a los pines analógicos de la placa de Arduino. Podemos conectarlos, por ejemplo, a los pines A0 y A5.

Como queremos usar el cursor como lápiz, podemos empezar el programa utilizando el bloque *bajar lápiz* en el apartado «lápiz» para que el cursor siempre dibuje al moverse. Para que no nos moleste, podemos «esconder» el cursor, así podremos ver mejor lo que dibujamos. A continuación empieza el bucle principal (figura 142).

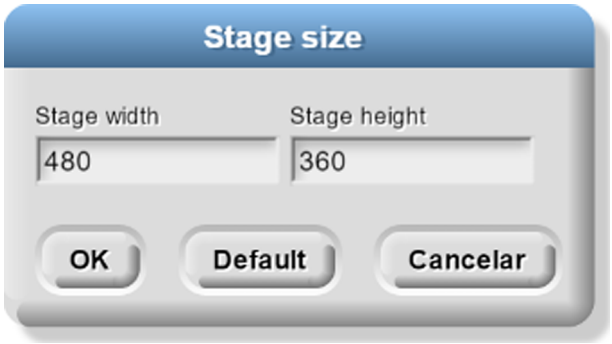
Figura 142



Para poder regular las posiciones en *x* e *y* del lápiz con respecto al alto y el ancho del escenario, tenemos que hacer un par de operaciones matemáticas para «mapear» el valor de cada uno de los potenciómetros.

Podemos consultar las dimensiones del escenario en las opciones de Snap4Arduino pulsando el icono con el engranaje y después en el apartado «Stage Size». Ahí veremos las dimensiones del escenario, que por defecto son 480 de ancho y 360 de alto (figura 143).

Figura 143



Para poder mover el lápiz en el eje de las *x* o a lo ancho del escenario a partir de la lectura de la entrada analógica del potenciómetro, tenemos que dividir el valor de la lectura analógica entre el número que obtenemos al dividir el valor máximo posible de la entrada analógica (1.023) entre el ancho del escenario.

$$\left(\text{Lectura analógica} \;/\; \frac{\text{Máximo de la entrada analógica}}{\text{Ancho del escenario}}\right)$$

Para que no nos quede desplazado, tenemos que restarle la mitad del ancho del escenario (porque en Snap4Arduino, el centro es 0 y tenemos números negativos si vamos a la izquierda, y positivos si vamos a la derecha).

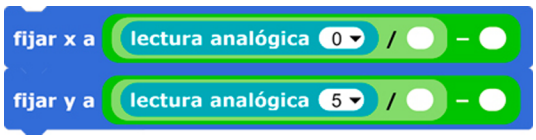
Por lo tanto, la fórmula es:

$$\left(\text{Lectura analógica} \;/\; \frac{\text{Máximo de la entrada analógica}}{\text{Ancho del escenario}}\right) - \frac{\text{Ancho del escenario}}{2}$$

Figura 144

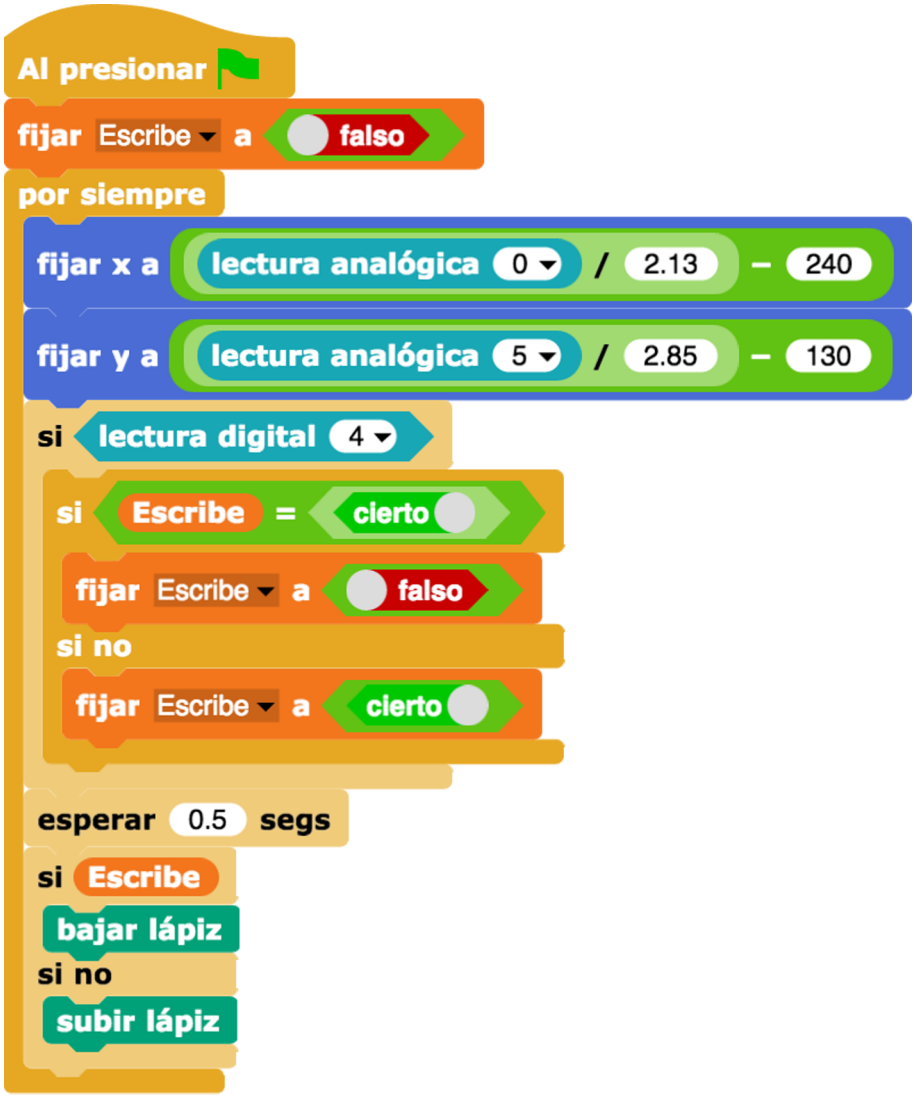
Una vez que hemos mapeado el valor de la lectura analógica, podemos fijar *x* al valor del potenciómetro que controla el ancho y fijar *y* al valor del potenciómetro que controla el alto (figura 145).

Figura 145



Por último, solo nos falta programar una tecla para borrar todo (figura 146). Para ello podemos usar la tecla de espacio y el bloque *borrar* del apartado «lápiz».

Figura 146



4.5. Variación difícil de la luz de fiesta con un módulo LED RGB

Para esta práctica vamos a usar el módulo LED RGB para crear una luz que cambia de color poco a poco de forma controlada.

Necesitaremos:

- La placa de Arduino
- 1 módulo LED RGB (figura 147)
- Cables

Figura 147

La conexión es muy sencilla, conectaremos los pines del módulo LED RGB de la manera siguiente: conectaremos el pin marcado con – a GND de nuestra placa de Arduino, y los pines R, G y B a los pines 9, 10 y 11 de la placa de Arduino.

En Snap4Arduino crearemos tres variables, *R*, *G* y *B*, para almacenar los valores de cada uno de los tres colores que tiene el led, y otras tres variables *R nuevo*, *G nuevo* y *B nuevo*, para almacenar los valores de los colores nuevos que queremos en el led.

Empezamos el programa con un bucle *por siempre* (figura 148) y asignamos un valor al azar entre 0 y 255 a las variables del color nuevo (figura 149).

Figura 148

Figura 149

Ahora añadimos un bucle *repetir hasta que* que pare cuando se pulse la tecla de espacio; así, el bucle principal volverá a empezar cada vez que apretemos la tecla de espacio y buscará un color nuevo (figura 150).

Figura 150

Queremos que en el bucle *repetir hasta que* las variables del color *R*, *G* y *B* vayan cambiando poco a poco hacia el color que tienen guardado las variables *R nuevo*, *G nuevo* y *B nuevo*. Para ello podemos usar la fórmula siguiente en las tres variables (figura 151):

Figura 151

Por último, solo queda asignar el valor de la variable al pin analógico al que está conectado cada color del led (figura 152):

Figura 152

Podemos usar el bloque *esperar* para ajustar el tiempo que tarda en pasar de un color a otro, pero con un valor muy pequeño para que la transición entre colores sea suave; por ejemplo: 0,01 (figura 153).

Figura 153