

PROGRAMA ARDUINO MEDIANTE CÓDIGO



CENTRO ARAGONÉS de TECNOLOGÍAS para la EDUCACIÓN

LICENCIA Y AUTORÍA:
VER CRÉDITOS

Table of Contents

Introducción	1.1
--------------	-----

1. Fundamentos del Arduino

1.1 Arduino	2.1
1.2 Conozcámoslo	2.2
1.3 Hardware	2.3
1.3.1 Alimentación eléctrica de Arduino	2.3.1
1.4 Entorno de programación	2.4
1.5 EDUBASICA	2.5
1.6 Simulación	2.6
1.7 Un poco de orden... el pensamiento computacional	2.7
1.8 Kit de préstamo de CATEDU	2.8

2. Montajes básicos con Arduino

2.1 Montajes básicos con Arduino	3.1
2.2 Conexiones digitales	3.2
2.2.1 M1 LED parpadeante	3.2.1
2.2.2 M2: LED parpadeante EDUBASICA	3.2.2
2.2.3 Pull-up y pull-down	3.2.3
2.2.4 M3: SEMÁFORO	3.2.4
2.2.5 M4: Pulsador	3.2.5
2.3 Conexiones analógicas	3.3
2.4 Señales PWM	3.4
2.4.1 M5: Control LED	3.4.1

Entradas y salidas

3.0 Entradas y Salidas	4.1
3.1 M1: Potenciómetro	4.2
3.1.1 M2: Mapeo	4.2.1
3.1.2 Mapeo de valores	4.2.2
3.2 M3: LDR	4.3
3.3 M4 Intensidad LED	4.4
3.4 Bobinas-altavoz	4.5
3.4.1 M5: Sirena	4.5.1
3.4.2 M6: Alarma	4.5.2
3.5 LCD	4.6
3.5.1 Pantalla LCD	4.6.1

3.5.2 I2C	4.6.2
3.5.3 Escaneo	4.6.3
3.5.4 Librería LiquidCrystal_I2C	4.6.4
3.5.5 M7 Texto en LCD	4.6.5
3.4 Sensor de ultrasonidos	4.7
3.4.1 M8: Medición	4.7.1
3.4.2 M9 Medición con LCD	4.7.2
3.5 Temperatura y humedad	4.8
3.5.1 Conexión	4.8.1
3.5.2 Libreria DHT11.h	4.8.2
3.5.3 Libreria DHT12.h	4.8.3
3.5.4 M9 Numéricamente T y H	4.8.4
3.5.5 M10 Por LCD	4.8.5
3.5.6 Processing	4.8.6
3.5.7 M11 Gráficamente T y H	4.8.7
3.5.8 dweet.io	4.8.8
3.5.9 M12 Por Internet T y H	4.8.9
3.6 Sensor de infrarrojos CNY70	4.9
3.6.1 M13 detección linea blanca	4.9.1

4. Comunicaciones

4.0 Introducción	5.1
4.1 Arduino y móvil	5.2
4.1.1 Teoría Bluetooth	5.2.1
4.1.2 JY-MCU HC-06	5.2.2
4.1.3 La APP	5.2.3
4.1.4 Vincular móvil	5.2.4
4.1.5 M1 Encender LEDs	5.2.5
4.1.6 M1bis sin EDUBASICA	5.2.6
4.1.7 Configuración avanzada	5.2.7
4.1.8 Bluetooth maestro	5.2.8
4.2 Arduino-Arduino	5.3
4.2.1 M2: dos Arduinos	5.3.1
4.2.2 Otras conexiones	5.3.2

5. Robótica

5.0 Robótica	6.1
5.1 Control de servomotores	6.2
5.1.1 M1 Test servo	6.2.1
5.1.2 Conexión	6.2.2
5.1.3 M2 Servo	6.2.3

5.1.4 M3 servo paso a paso	6.2.4
5.1.5 M4 servo y potenciómetro	6.2.5
5.2 Motores DC	6.3
5.2.1 Transistor	6.3.1
5.2.2 Conexión	6.3.1.1
5.2.3 M5 Motor-transistor	6.3.1.2
5.2.2 Circuito L293	6.3.2
5.2.2.1 Con Edubásica	6.3.2.1
5.2.2.2 Sin Edubásica	6.3.2.2
5.2.3 Coche	6.3.3
5.2.3.1 Chasis con EDUBASICA	6.3.3.1
5.2.3.2 Chasis sin Edubásica	6.3.3.2
5.2.3.3 M6 Coche loco	6.3.3.3
5.2.3.4 M7 Coche teledirigido	6.3.3.4
5.2.3.5 M8 Coche con voz	6.3.3.5
5.3 Barrera	6.4
5.3.1 Bluetooth	6.4.1
5.3.2 M9 Parking	6.4.2
Muro	6.5
Chat robótica educativa aragón	6.6
Créditos	6.7

Fuera del curso

Fuera del curso	7.1
EA.1 Electrónica analógica	7.1.1
EA.2 Resistencias	7.1.2
EA.2.1 Divisor de tensión	7.1.2.1
EA.2.2 M1: serie	7.1.2.2
EA.2.3 M2: paralelo.	7.1.2.3
EA.3 Diodos	7.1.3
EA 3.1 Diodos M3 Estudio tensión umbral	7.1.3.1
EA.4 Condensadores	7.1.4
EA.4.1 M4: Carga	7.1.4.1
EA.4.2 M5: Descarga	7.1.4.2
EA.4.3 Actividad	7.1.4.3
EA.5 Transistores	7.1.5
EA.5.1 Conexiones	7.1.5.1
EA.5.2 M6: SAT-CORT	7.1.5.2
EA.5.3 M7: activo	7.1.5.3
EA.5.4 M8: Mejor con diodo	7.1.5.4
EA.5.5 M9: hfe	7.1.5.5

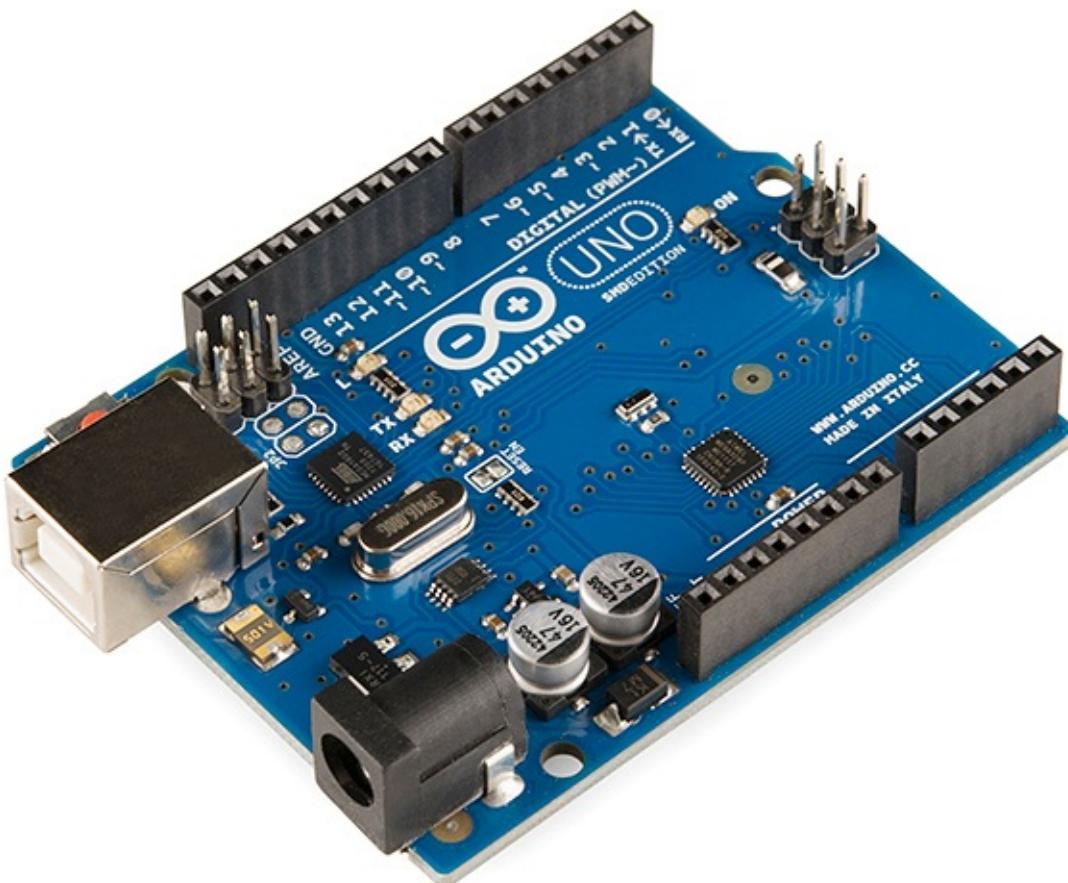
EA.5.7 Recta de carga de un transistor	7.1.5.6
ED.0 Electrónica digital	7.1.6
ED.1 Álgebra de Boole	7.1.7
ED.2 M10 AND sin EDUBASICA	7.1.8
ED.3 M11 AND con EDUBASICA	7.1.9
ED.4 M12 ELEVADOR sin EDUBASICA	7.1.10
ED.5 M13 ELEVADOR con EDUBASICA	7.1.11
ED.6 M14 ALARMA	7.1.12

Introducción

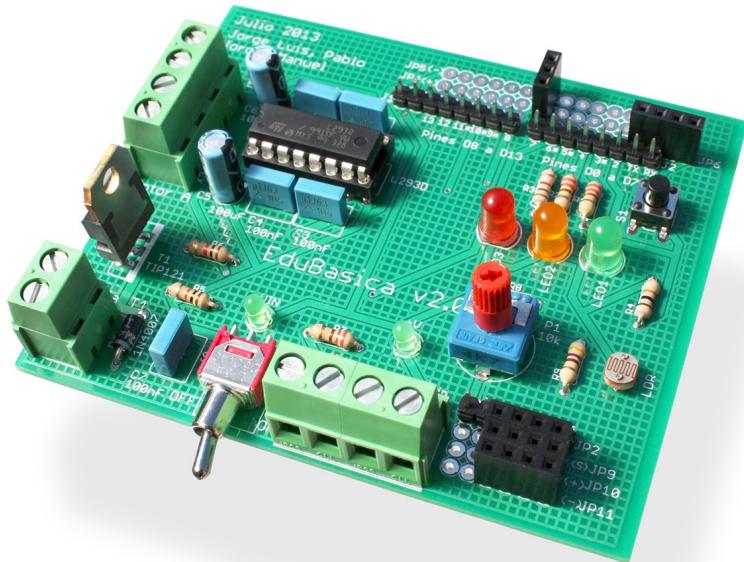
En este curso intenta hacer un barrido por los fundamentos de la electrónica y del Arduino los dos de la mano.

1. El primer tema es una breve descripción del **ARDUINO**, siempre desde el punto de vista de la enseñanza del pensamiento computacional: su programación, simulación y comparación con otra opciones en robótica educativa.
2. Un primer contacto práctico con las **PROPUESTAS MÁS SENCILLAS** y siempre sin perder la posibilidad de aprender los fundamentos de la electrónica.
3. Dentro de esta línea entraremos en el mundo de la **ELECTRÓNICA ANALÓGICA**
4. Un breve contacto con los principios de la **ELECTRÓNICA DIGITAL**
5. **LAS COMUNICACIONES**, en especial con un Smartphone, tienen un buen componente innovador y motivador.
6. **CONTROL Y ROBÓTICA** donde dejaremos la puerta abierta al mundo de posibilidades a los pequeños inventores.

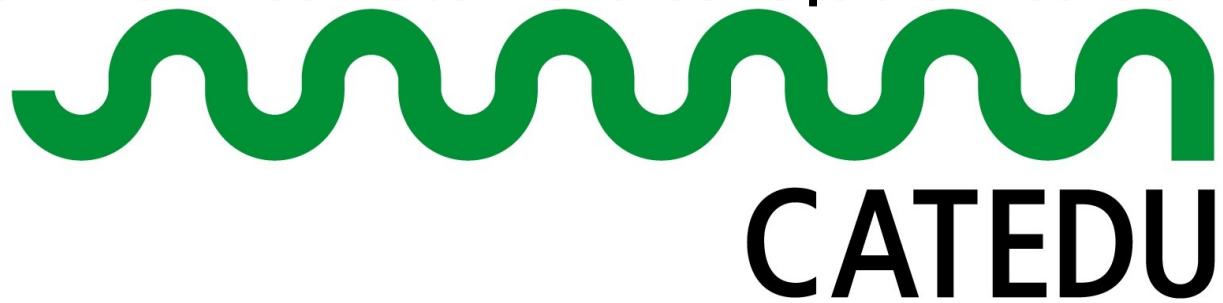
En este curso se utiliza una Shield llamada **EDUBASICA** pero sin esta Shield el curso se puede hacer, aunque con más cableado y electrónica.



Se busca una metodología **totalmente práctica** sin conocimientos previos, pero para simplificar el cableado engoroso de circuitos complejos se utilizará un escudo: la **edubásica**, aunque se puede prescindir de ella en prácticas sencillas.

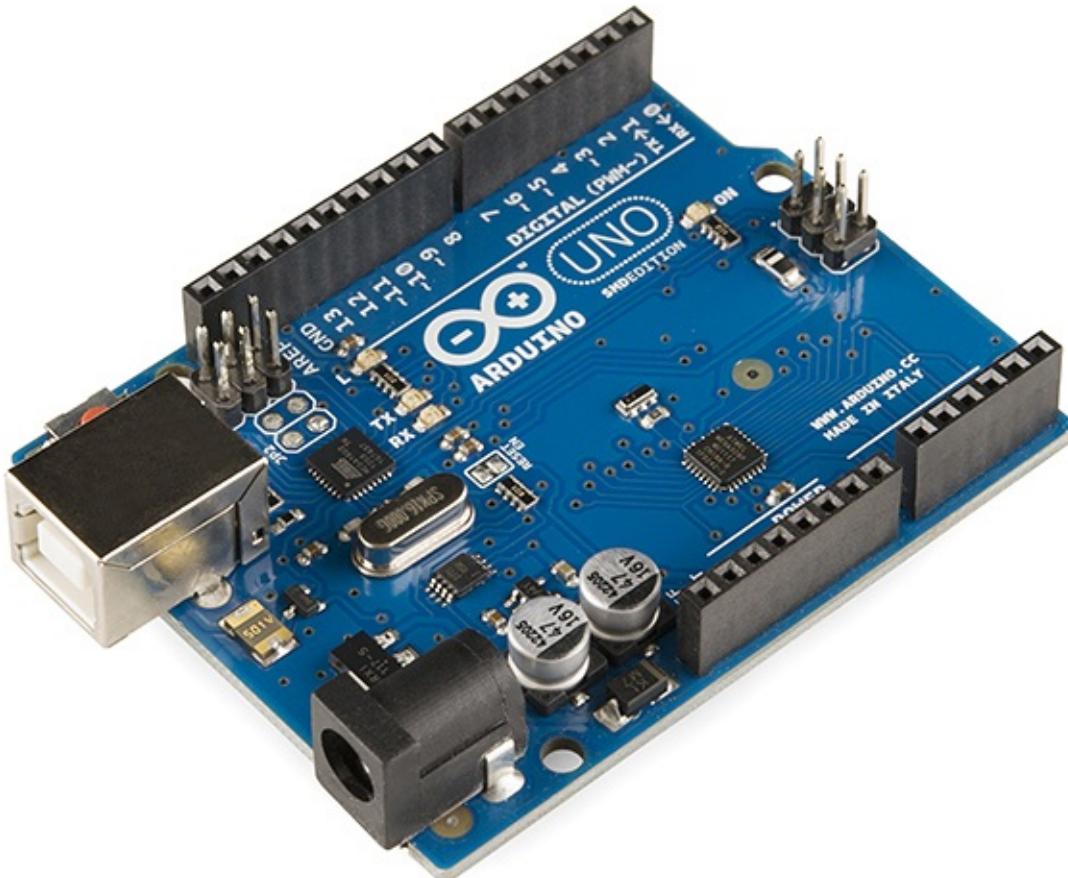


CENTRO ARAGONÉS de TECNOLOGÍAS para la EDUCACIÓN



Fundamentos de Arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. En este capítulo ofrecemos una guía de referencia rápida que siempre puede ser ampliada accediendo a la página oficial: www.arduino.cc



Información básica sobre Arduino

¿Qué es Arduino?

Arduino es una tarjeta electrónica que integra básicamente a un microcontrolador y un conjunto de pines de conexión de entradas y salidas que permiten, mediante un determinado programa, interaccionar con el medio físico mediante sensores y actuadores electrónicos. De esta forma podrás crear tus propios proyectos tecnológicos, dotarlos de sensores que detecten magnitudes físicas como luz, calor, fuerza, etc... y en base a esa información, escribiendo un programa, activar otros dispositivos (actuadores) como pequeñas bombillas, ledes, servomotores, pequeños motores DC, relés, etc... Los sensores se conectan a los pines de entrada y los actuadores a los de salida.

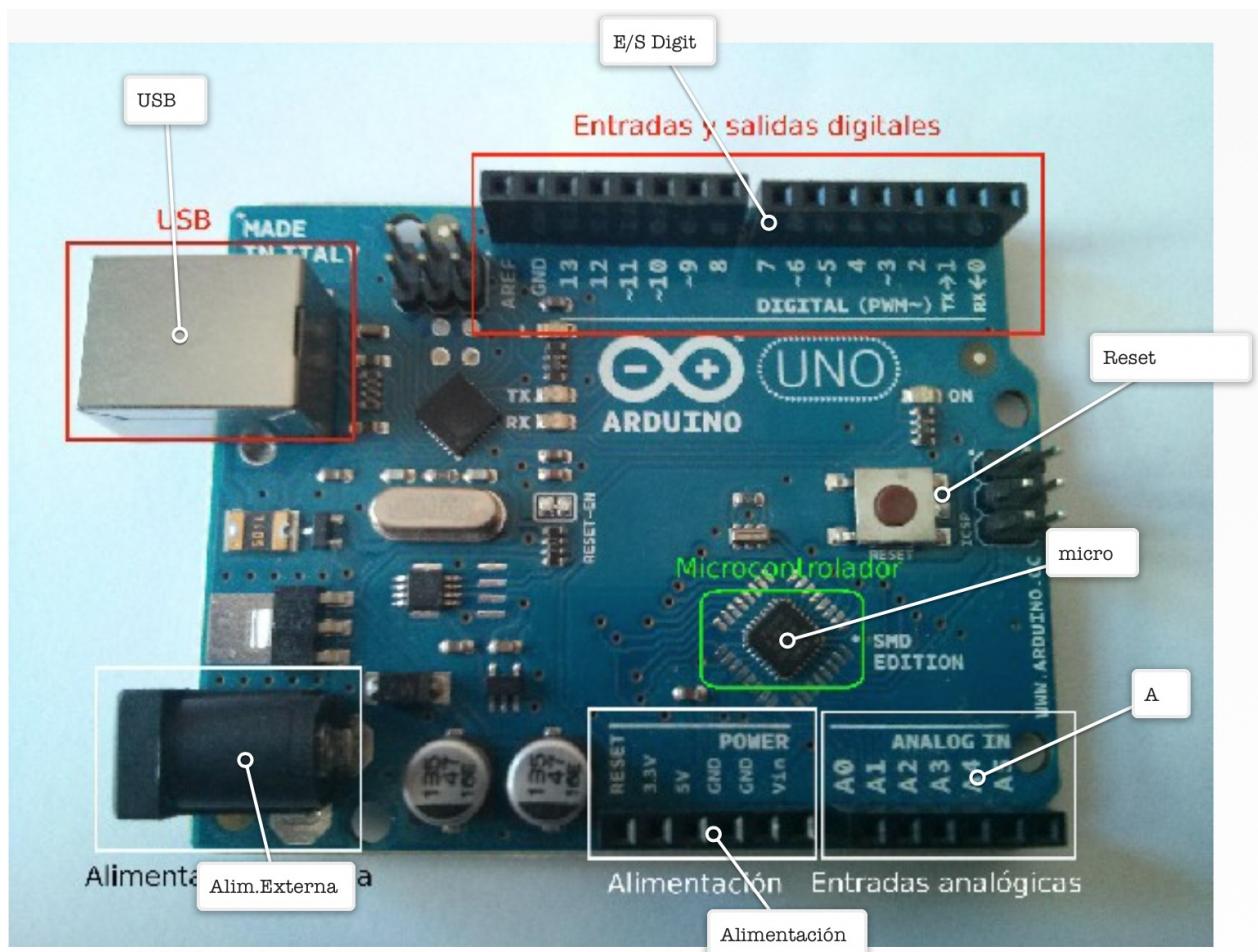
¿Sabías que.... ? Uno de los co-creadores de Arduino es Español, de Zaragoza: David Cuartielles [+info](#)

¿Qué es un microcontrolador?

Es un circuito integrado que se puede programar, o sea que se puede ejecutar las órdenes que tenga almacenadas en su memoria. Tiene las tres funciones principales de un computador: la unidad central de proceso, memoria y entradas y salidas.

Arduino utiliza la marca ATMEL, y el modelo de microcontrolador depende del tipo de tarjeta, por ejemplo la tarjeta Arduino Uno utiliza el micro ATMEL MEGA 328P.

¿Qué se puede hacer con Arduino, algún ejemplo?



Realmente el límite lo marca tu imaginación pero por dar alguna pista, podrías diseñar un sistema para la apertura y cierre de la puerta de un garaje, hacer un robot móvil que detecte objetos o que siga una línea negra, crear un detector de luz y oscuridad, implementar un termómetro, controlar un cilindro neumático, etc...

En este manual tienes múltiples ejemplos de pequeños proyectos para el aula, aunque Arduino es una herramienta que también se utiliza en el ámbito profesional para monitorización de sensores y automatización a pequeña escala por su flexibilidad, fiabilidad y precio.

¿Qué son las entradas y salidas?

Mediante los conectores de Arduino correspondientes a las entradas y salidas podemos comunicar nuestros programas con el “mundo exterior”. Si queremos leer el valor de la magnitud física medida por un sensor, por ejemplo una LDR que detecta el nivel de luminosidad, lo tendremos que hacer conectando el sensor a uno de los pines de entrada (en este caso analógicas) de la tarjeta.

De esta forma con una simple instrucción de lectura en el programa, podremos obtener el valor de la magnitud física. Si nuestra intención es actuar o “hacer algo” una vez leído el valor del sensor, por ejemplo encender un led si el sensor de luminosidad detecta oscuridad, tendremos que conectar el actuador (en este caso el led) a un pin de salida que proporcionará la corriente necesaria para activarlo.

En Arduino las entradas pueden ser analógicas o digitales y las salidas sólo digitales. Cada pin digital tiene doble función entrada o salida. En la zona de configuración del programa hay que indicar explícitamente mediante una instrucción cuál es función desempeña un determinado pin.

¿Dónde se conectan los sensores a las entradas analógicas o digitales?

Los sensores utilizados en los proyectos que vamos a utilizar son de salida analógica, es decir proporcionan una variación de voltaje dentro de un rango (normalmente de 0 a +5V) dependiendo de lo que varíe la magnitud física medida. Muchos sensores son resisitivos (luz, temperatura, humedad,...), es decir que varían su resistencia eléctrica con la magnitud física, pero mediante un sencillo montaje de divisor de tensión conseguimos una variación de voltaje apta para Arduino. Estos montajes los veremos en las prácticas del manual.

Una vez realizadas las conexiones, si midiéramos la salida del sensor con un voltímetro nos daría un valor decimal, por ejemplo un nivel de luz “intermedio” (rango de 0 a 5V) de un sensor de luz podría dar 3,3 voltios. Este tipo de información el microcontrolador no la entiende tal cual, sólo es capaz de interpretar números binarios (“0” ó “1”) por lo que para traducir los valores analógicos dispone internamente de un conversor analógico – digital que hará la conversión entre los dos sistemas, de forma que podremos tener valores discretos de la medida de los sensores analógicos.

Las entradas analógicas leen valores analógicos entre 0V y la alimentación (normalmente 0-5V) y lo convierte en números entre **0 y 1024**.

Entonces, ¿qué utilidad tienen las entradas digitales?

Las entradas digitales son útiles cuando las señales a leer son valores discretos. Por ejemplo queremos poner un pulsador o un interruptor que encienda un led. Hacemos un montaje que cuando se pulse, entren 5 voltios en el pin digital de entrada y cuando no se pulse que “entren” 0 voltios. De esta manera la lectura del pin digital de entrada será “HIGH” con 5 voltios o “LOW” con 0 voltios.

¿Qué son las salidas digitales etiquetadas con PWM (~)?

Son salidas digitales que simulan una salida analógica. Las siglas significan Modulación por Ancho de Pulso (Pulse Width Modulation) o proporcionan una onda cuadrada con un nivel alto (+5V) de “cierta” duración.

Los valores PWM que podemos proporcionar pueden ir desde **0 a 255** que corresponderían a un 0V analógico y a la máxima tensión (la de alimentación, normalmente 5V).

Es muy útil para activar servomotores y llevarlos a una posición determinada o variar la luminosidad de un led. Lo puedes ver más explicado en la siguiente sección.

¿Puedo accionar motores DC con Arduino?

Si son motores muy pequeños sí sería posible aunque no es recomendable. Los motores necesitan un consumo alto de corriente, sobre todo si tienen que mover cierta carga, por lo que se recomienda o bien utilizar una tarjeta Shield o extensión de Arduino que dispone de circuitería apta para proporcionar dicha corriente (transistores). En este manual utilizamos una Shield bautizada como Edubásica de elaboración propia que dispone de un transistor y un circuito integrado LM293 para realizar esta función, además de otras ventajas para el aprendizaje de Arduino.

Para saber más ...

Te recomendamos esta página donde indica la historia y la electrónica al detalle de esta placa. <https://programarfacil.com/blog/arduino-blog/arduino-uno-r3/>

Hardware

Placa Arduino

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing).

Entradas y salidas

La placa Arduino Duemilanove o UNO consta de:

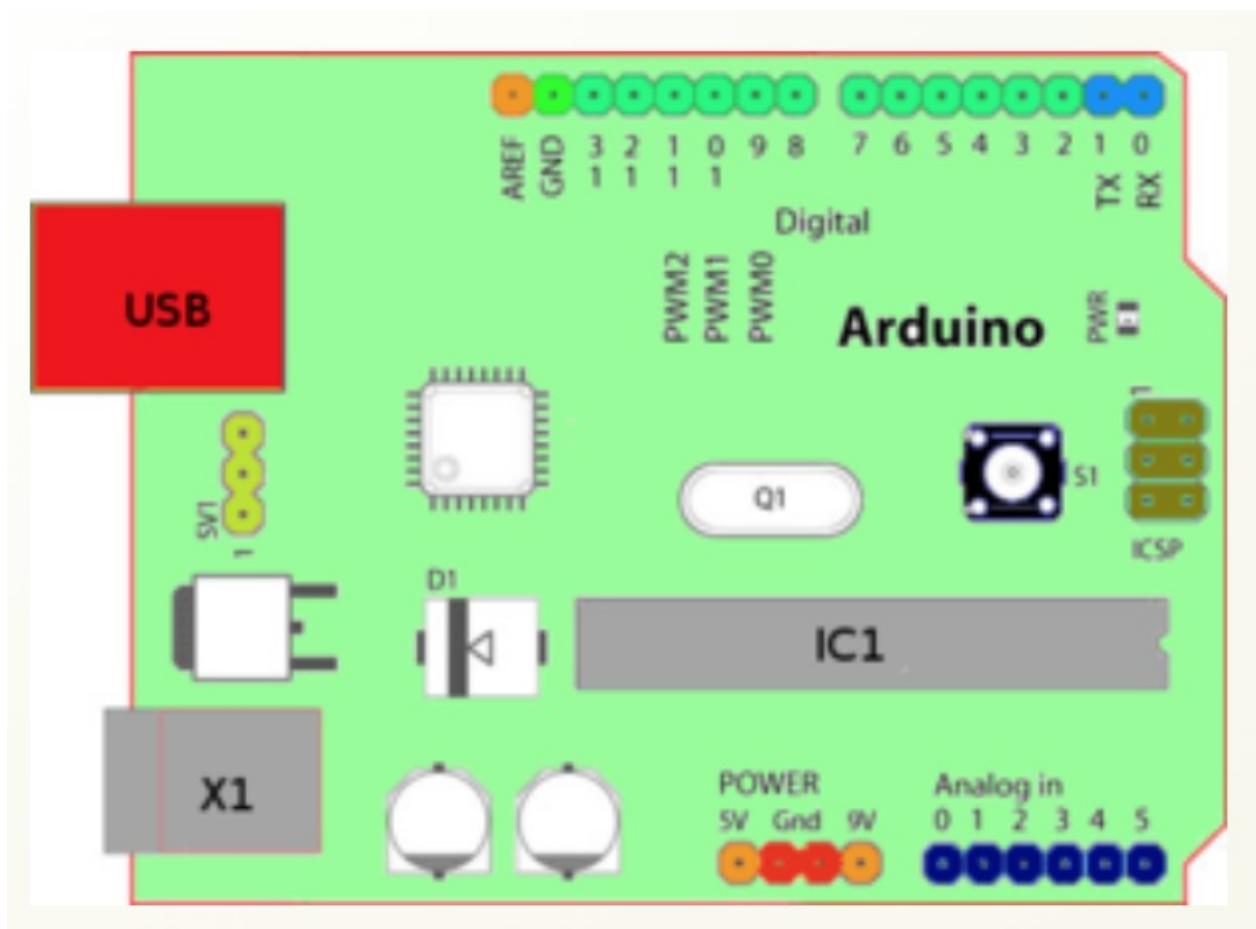
14 entradas digitales configurables Entrada/Salidas que operan a 5 ó 0 voltios. Cada pin puede proporcionar o recibir como máximo 40 mA.

Los pines 3, 5, 6, 8, 10 y 11 pueden proporcionar una salida PWM (Pulse Width Modulation). Si se conecta cualquier dispositivo a los pines 0 y 1, eso interferirá con la comunicación USB.

6 entradas analógicas con una resolución de 10 bits que proporcionan un número entero de 0 a 1023. Por defecto miden de 0 voltios (masa) hasta 5 voltios.

Pines de la placa

Elementos con los que podemos interactuar: (tomando como ejemplo la placa USB). Empezando en el sentido de las agujas del reloj desde el centro de la parte superior:



- Pin de referencia analógica (naranja).
- Señal de tierra digital (verde claro).

- Pines digitales 2-13 (verde).
- Pines digitales 0-1 / entrada y salida del puerto serie: TX/RX (azul) (estándar de comunicación serie IC2).
- Botón de reset (negro).
- Entrada del circuito del programador serie (marrón).
- Pines de entrada analógica 0-5 (azul oscuro).
- Pines de alimentación y tierra (naranja y naranja claro).
- Entrada de la fuente de alimentación externa (9-12V DC) – X1 (gris).
- Comutación entre fuente de alimentación externa o alimentación a través del puerto USB – SV1. En las placas más recientes la conmutación de la alimentación se realiza con un MOSFET.
- Puerto USB (rojo).

Las placas: Arduino Diecimila, Arduino Duemilanove o UNO y Arduino Mega están basados en los microcontroladores Atmega168, Atmega 328 y Atmega1280 respectivamente.

Las especificaciones de cada uno de los microcontroladores se exponen en la tabla siguiente:

	Atmega168	Atmega328	Atmega1280
Voltaje operativo	5 V	5 V	5 V
Voltaje de entrada recomendado	7-12 V	7-12 V	7-12 V
Voltaje de entrada límite	6-20 V	6-20 V	6-20 V
Pines de entrada y salida digital	14 (6 proporcionan PWM)	14 (6 proporcionan PWM)	54 (14 proporcionan PWM)
Pines de entrada analógica	6	6	16
Intensidad de corriente	40 mA	40 mA	40 mA
Memoria Flash	16KB (2KB reservados para el bootloader)	32KB (2KB reservados para el bootloader)	128KB (4KB reservados para el bootloader)
SRAM	1 KB	2 KB	8 KB
EEPROM	512 bytes	1 KB	4 KB
Frecuencia de reloj	16 MHz	16 MHz	16 MHz

SHIELDS para Arduino

Las llamadas Shields (escudos) para Arduino son tarjetas que añaden funciones a la placa Arduino.

Como se ha comentado antes, Arduino por sí sola no puede proporcionar la suficiente intensidad para alimentar motores, relés o electroválvulas. El límite de intensidad que proporciona cada una de las salidas digitales es de 40 mA. Para poder activar estos dispositivos tendremos que montar un circuito externo adicional con transistores o circuitos integrados específicos para motores, como es el caso del LM293, que entregan la intensidad suficiente. Para facilitarnos la tarea existen unas placas adaptadas a los pines de Arduino que se ensamblan directamente sobre ella a modo de "escudo" (de ahí su nombre, shields) y nos permiten tener pines adicionales para alimentar las cargas que Arduino por sí solo no es capaz de mover.

Hay también otro tipo de "shields" que proporcionan funciones como conexión Ethernet, WIFI, XBee, GSM, Host USB, etc...

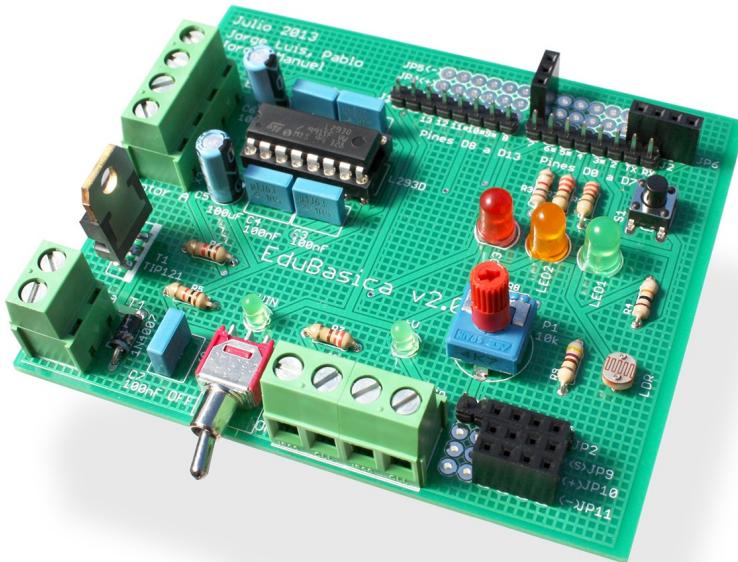
Actualmente hay decenas de ellas en el mercado. Puedes encontrar una larga lista de ellas en

<http://playground.arduino.cc/Main/SimilarBoards#goShie>

Algunos ejemplos:

SHIELDS EDUCATIVAS

La shield de este curso **EDUBASICA**



La shield de [otro curso en CATEDU: ECHIDNA](#)



SHIELD VISUALINO k5864195

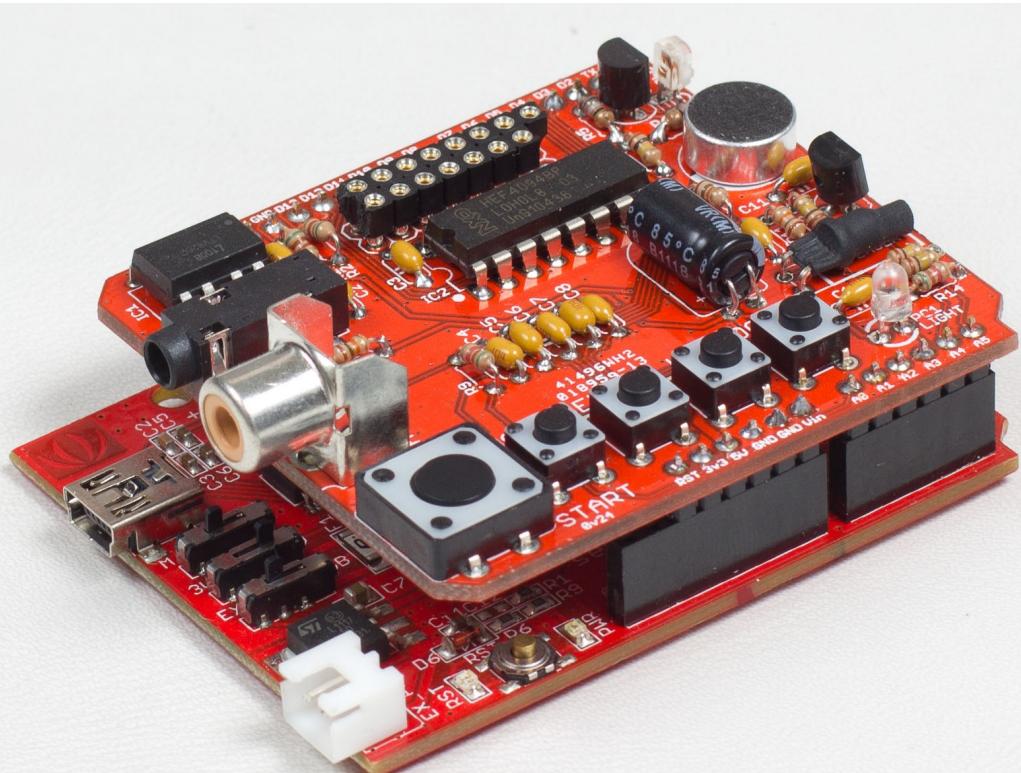
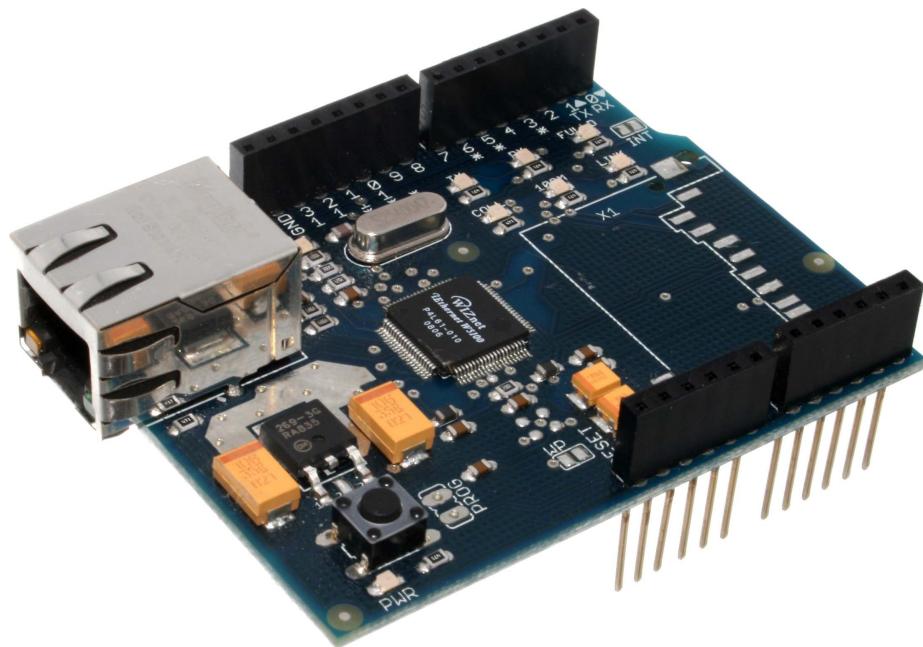
Es una shield bastante barata, por 10€ placa Arduino+Shield Visualino y nos gusta porque tiene :

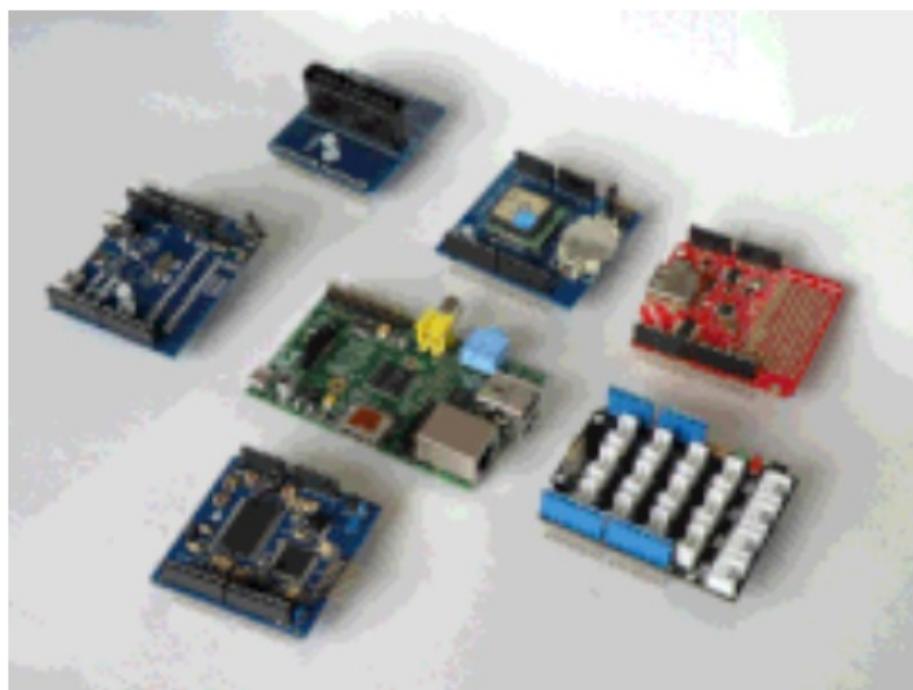
- 4 diodos LED
- 3 pulsadores
- 1 potenciómetro
- 1 zumbador
- 1 Display

O sea buena y barata. ¿Por qué seguimos apostando por Edubásica? Por la posibilidad de Edubásica de poner motores y por lo tanto es más motivador. Echidna nos gusta porque tiene un Joystick, sensor de luz.. que también da juego gamificado.. en fin, Visualino es una buena opción para Arduino básico.

OTRAS SHIELD COMERCIALES







via GIPHY

Alimentación eléctrica de Arduino

Uno de los aspectos claves para el buen funcionamiento de proyectos con Arduino que incluyan elementos que consuman una intensidad superior a 200 mA como motores, relés, electroválvulas, etc... es la alimentación eléctrica de la placa. Normalmente tenemos dos posibilidades para alimentar Arduino:



Mediante el cable USB conectado al ordenador:

- Cada pin proporciona 40 mA.
- El límite proporcionado por el USB es de 500 mA en total.

Utilizando una fuente de alimentación externa conectada al jack de Arduino (fuente de voltaje, adaptador de corriente, batería o portapilas) :

- El voltaje recomendado de la fuente externa está entre 9 y 12 V.
- La intensidad máxima que puede entregar Arduino a los actuadores que queramos controlar (servos, motores, relés,...) es de 1A, aunque una exposición prolongada a esta corriente puede estropear la placa. Lo recomendable son 800 mA.
- El pin serigrafiado con Vin proporciona directamente el voltaje de la fuente conectada al jack de Arduino (menos la caída de tensión del diodo de protección), desde ese pin podemos sacar un cable y alimentar a los actuadores que necesitemos. Por ejemplo, si alimentamos con una pila externa de 9 V conectada al jack, en el pin Vin tendremos aproximadamente 9 V (hay que restar la caída de tensión del diodo de protección). Además en los pines 5V y 3.3V dispondremos también de dichos voltajes aunque la fuente externa sea de 9V.

Si conectamos demasiada carga, la placa Arduino suele tener un comportamiento anómalo pudiéndose se resetear el micro.

Conectando el positivo (+Vcc) de la fuente externa a Vin y el negativo a GND:

Podemos alimentar Arduino externamente si necesitamos conectar Jack a través de Vin y GND el problema es que nos saltamos un diodo de protección que evita que se queme el circuito por un exceso de corriente.

CONCLUSIÓN:

- Si necesitamos hacer funcionar actuadores de bajo consumo (luces, zumbadores, etc...) podremos trabajar directamente con el USB conectado al ordenador.
- Si necesitamos mover cargas, excitar bobinas u otros elementos de mayor consumo lo recomendable es alimentar externamente

Arduino desde el Jack con un rango de 9 a 12 V.

Entorno de programación

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

OJO, existe la versión web del editor <https://create.arduino.cc/editor> para trabajar online, puedes probarlo, pero seguimos recomendando la versión de escritorio por ser más rápida, no obstante es una buena solución si trabajas en varios equipos y quieras que tus proyectos estén disponibles en cualquier equipo.

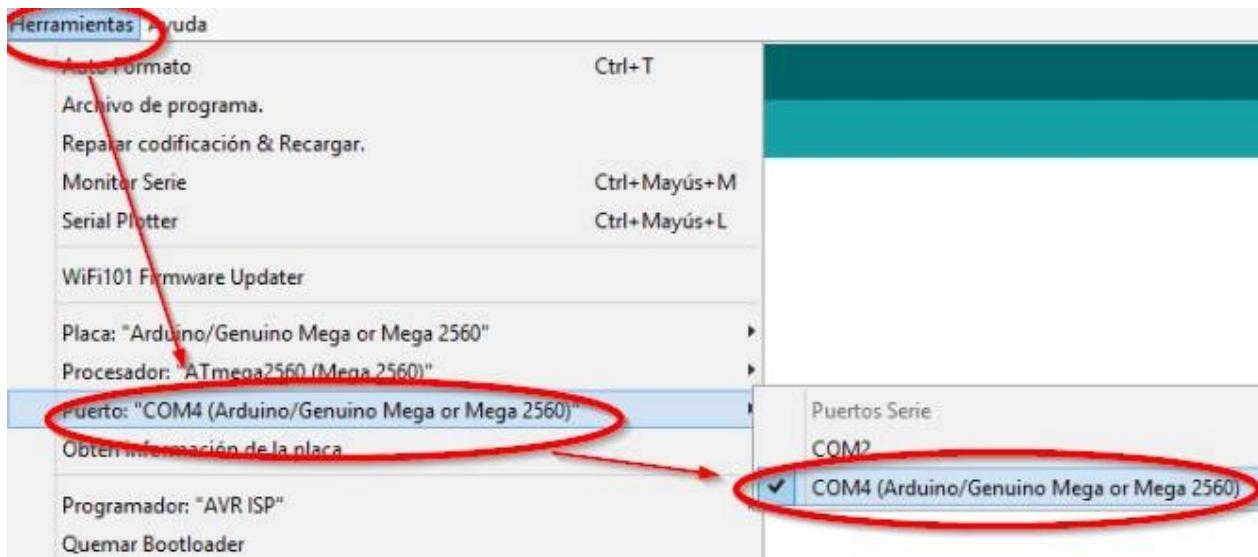
En Linux puede salir este mensaje "can't open device "/dev/ttyUSB0": Permission denied" donde 0 puede ser otro número, la solución [aquí](#)

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

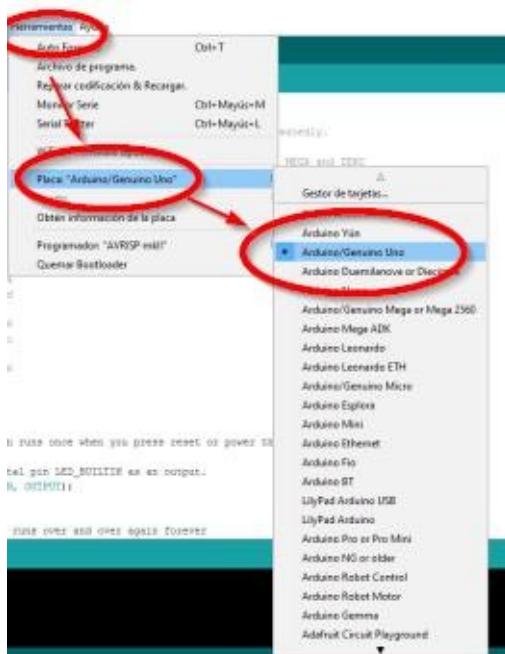
Arduino utiliza para escribir el código fuente o programa de aplicación lo que denomina "sketch" (programa). Estos programas son escritos en el editor de texto. Existe la posibilidad de cortar/pegar y buscar/reemplazar texto.



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



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



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

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



¿Cómo se programa Arduino?

Las partes principales de un programa hecho en Arduino son: Bloque de inclusión de módulos y declaración de variables, bloque de configuración **void setup()** donde se indica el modo de funcionamiento de los pines (entrada y salida), comunicación serie, etc... y bloque de ejecución continua **void loop()**, en este bloque se incluyen las acciones que queremos que realice el programa. Se ejecutará línea a línea

de forma secuencial y continua. Cuando llegue a la última instrucción incluída en la función **loop()** volverá a ejecutar la primera y continuará en un bucle infinito.

```

Knob
// Controlling a servo position using a potentiometer (variable resistor)
// by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

#include <Servo.h>

Servo myservo; // create servo object to control a servo

int potpin = 0; // analog pin used to connect the potentiometer
int val; // variable to read the value from the analog pin

void setup()
{
    myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop()
{
    val = analogRead(potpin);
    val = map(val, 0, 1023, 0, 179);
    myservo.write(val);
    delay(15);
}

```

¿Arduino tiene que estar continuamente conectada a un ordenador?

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

¿Qué voy a aprender con este manual?

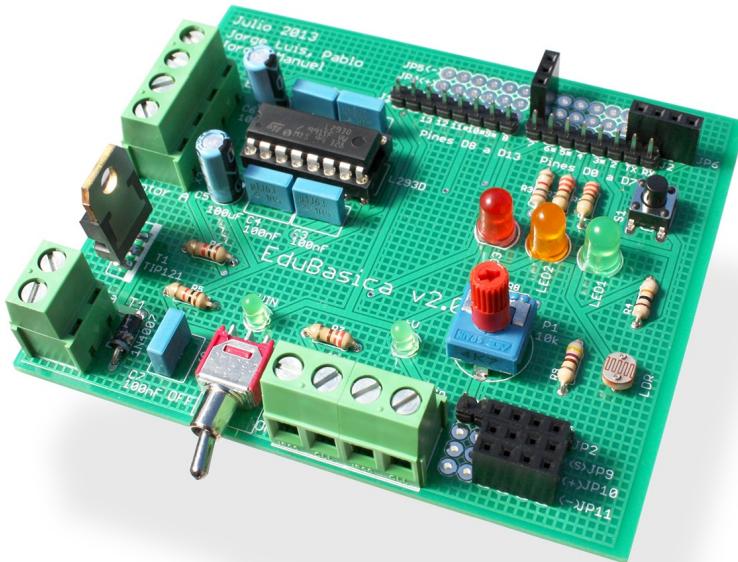
Aprenderás a realizar pequeños proyectos y prácticas cuya base de control es la tarjeta Arduino y en algunos casos la shield “Edubásica”. EDUBÁSICA es una tarjeta que se coloca sobre Arduino y lleva integrados muchos de los componentes básicos para realizar las prácticas de electrónica, y ciertos proyectos tecnológicos de una manera muy sencilla, aunque todas las prácticas se pueden implementar sin el uso de esta tarjeta. La idea del manual es integrar los contenidos del currículo en 4º de Enseñanza Secundaria Obligatoria (alumnos de 15-16 años) con diferentes proyectos para que el alumno consiga un aprendizaje significativo de la Tecnología.

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

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

www.arduino.es (portal en español).

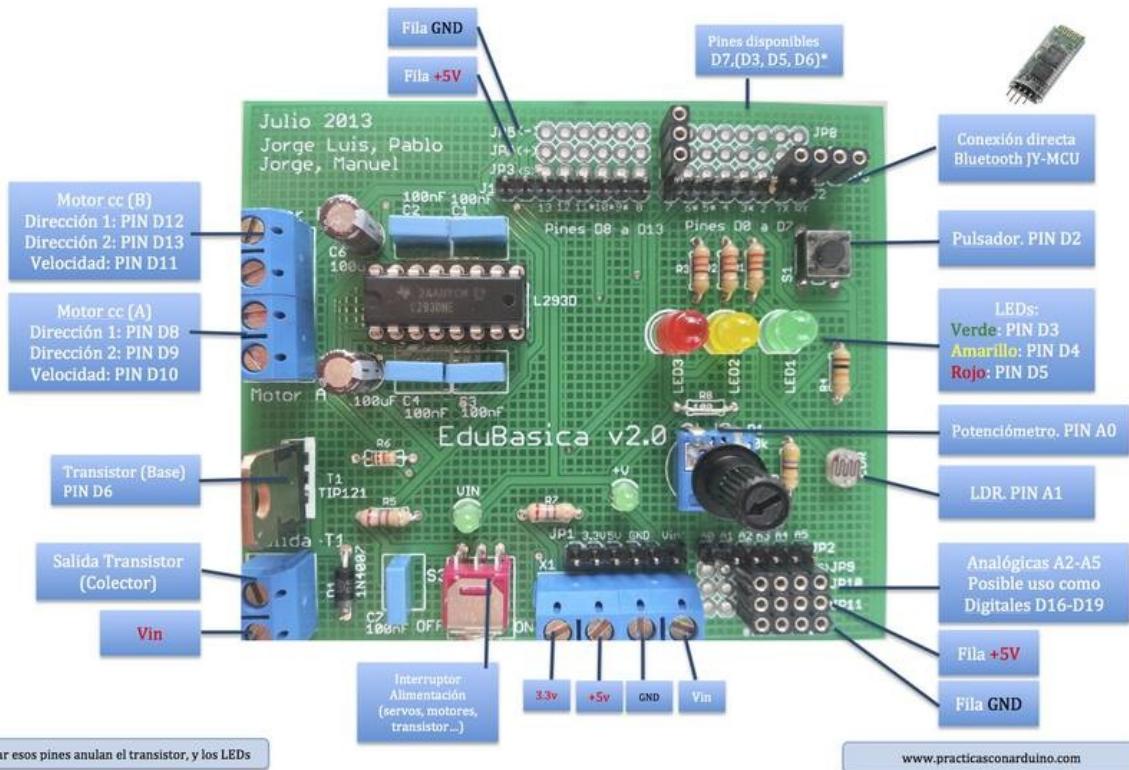
Una placa de apoyo: Edubásica



EDUBASICA es una tarjeta de prototipado rápido para conectar a Arduino. Incluye componentes electrónicos básicos para gran variedad de prácticas y proyectos. Es una tarjeta multipropósito con componentes electrónicos incluidos que puedes usar para fabricar un robot, controlar un sistema de poleas, activar barreras, comunicar dispositivos bluetooth, y todo lo que te puedas imaginar.

- información, conseguir una, etc.. en <http://www.practicasconarduino.com>
- Otras Shields ver [Echidna](#)

EduBasica v2.0 (rev. Nov/2013)



+Grande (jpg - 335,78)

ESQUEMAS Y PCB PARA SU FABRICACIÓN:

Disponemos de un repositorio de github para la descarga de los esquemas de Edubásica <https://github.com/jorgeroden/edubasica>. En él se incluyen las posibles modificaciones de la placa junto a los programas de test. En cualquier caso los esquemas para la última versión, en el momento de redactar este manual, se adjuntan aquí.

- Esquema
- Placa PCB
- Prácticas
- Proyectos

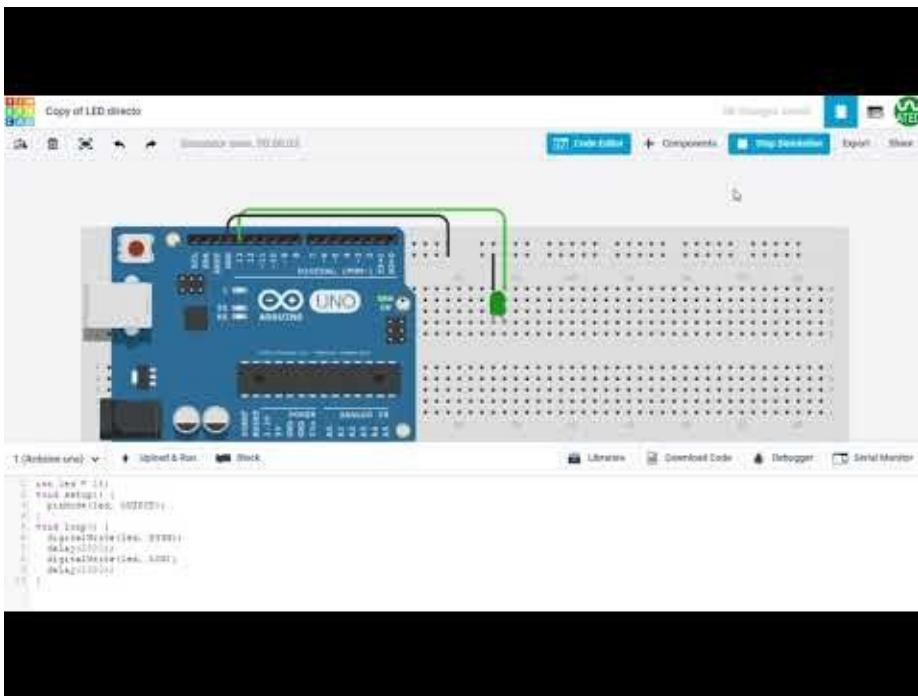
Tweets sobre EdubasicaShield

Tweets by EduBasicaShield

Simulación

Hay aplicaciones que permiten simular el Arduino, en estos apuntes no lo utilizaremos menos en un caso "conexión dos Arduinos" por si no tienes dos placas.

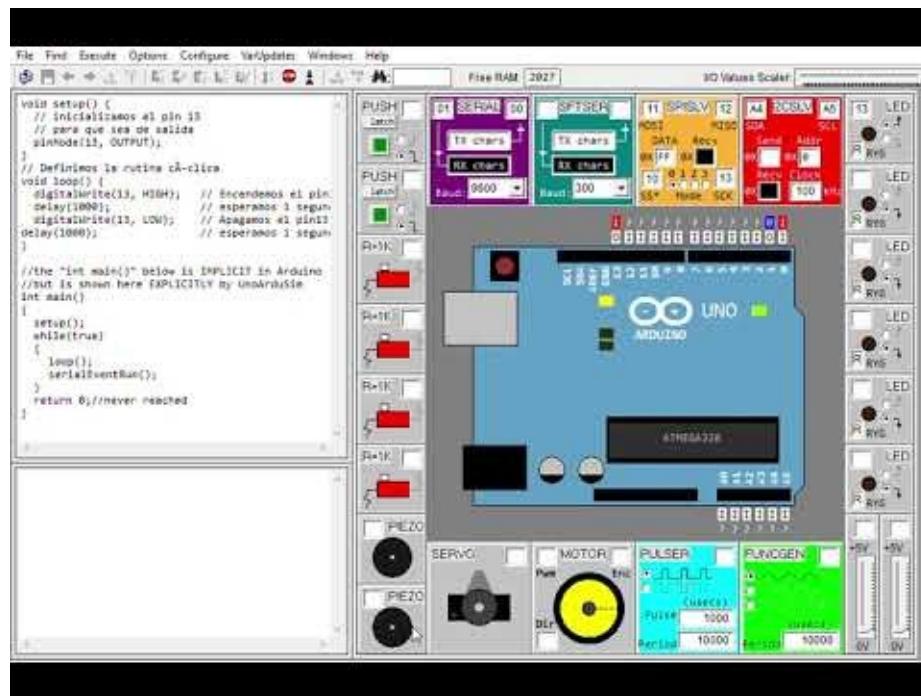
Pueden ser **aplicaciones online** como <https://www.tinkercad.com> pero nuestra experiencia es que es lento y arduo de trabajar pero tiene la ventaja de ser online y muy visual



[Video link](#)

Si quieres trabajar con más velocidad puedes descargar alguna **aplicaciones locales**, como [UnoArduSim](#) además es una aplicación portable fácil de instalar y con los elementos de leds, motores servos ya preparados, ideal para ejemplos sencillos y para examinar señales, pero no es tan versátil como el anterior, por eso te mostramos vídeos de ejemplo para que tú decidas.

En este vídeo se puede ver el mismo ejemplo del led parpadeando:



[Video link](#)

Un poco de orden... el pensamiento computacional

¿Esto es una moda?

No sabemos qué futuro van a encontrar nuestros alumnos, pero sí que sabemos que por ejemplo el **Inglés** será importante en su entorno futuro. Pues igual con las TIC, no es una moda, hace tiempo que está, y seguirá. **El pensamiento computacional es el idioma de los ordenadores.**

Vale, y ... este curso ¿dónde se encuadra? ¿para qué edad es recomendada?

Buena pregunta... para enseñar el pensamiento computacional tenemos dos caminos, totalmente compatibles:

- **La programación**, que sería como enseñar un nuevo idioma.
- **La robótica** que sería como practicar este idioma con un nativo, luego antes hay que saber el idioma.

En CATEDU hemos elaborado esta **hoja de ruta** de herramientas y edades, hay otras herramientas y otros criterios TOTALMENTE VALIDOS, este es el nuestro, lo que hemos elegido en los cursos de [Aularagon](#) y que enseñamos a continuación como orientación, pero no se debe de tomar al pie de la letra.

RoboTICa

Oferta de formación en Pensamiento computacional del Centro Aragonés de Tecnologías para la Educación.



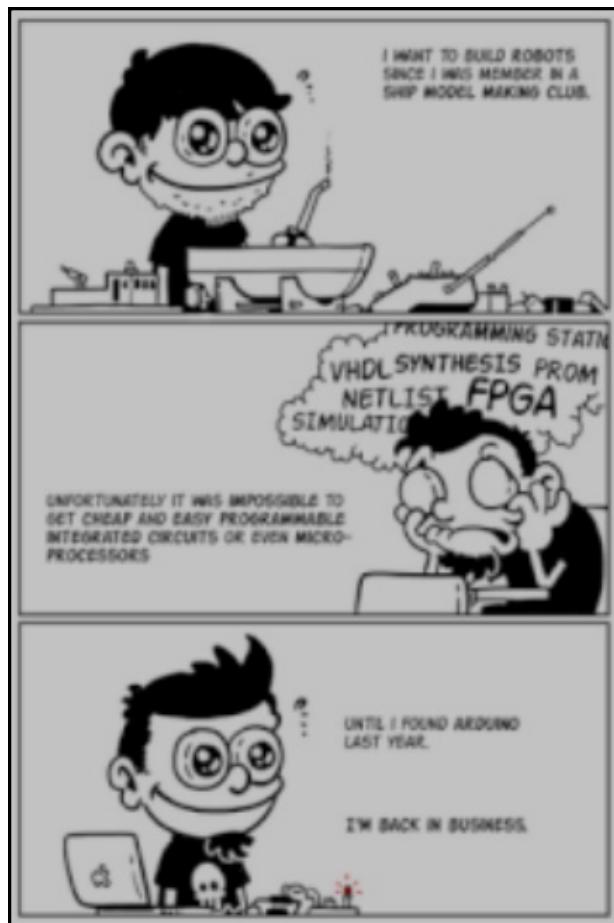
Kit de préstamo de CATEDU

Este es el kit que prestamos para hacer este curso, con la placa de apoyo EDUBASICA. Si no tienes EDUBASICA este curso se puede hacer perfectamente, pero se complica el cableado. **POR FAVOR DEVUELVE TODOS LOS COMPONENTES EN BUEN ESTADO, PIENS A EN TU PRÓXIMO COMPAÑERO QUE VA A HACER EL CURSO**



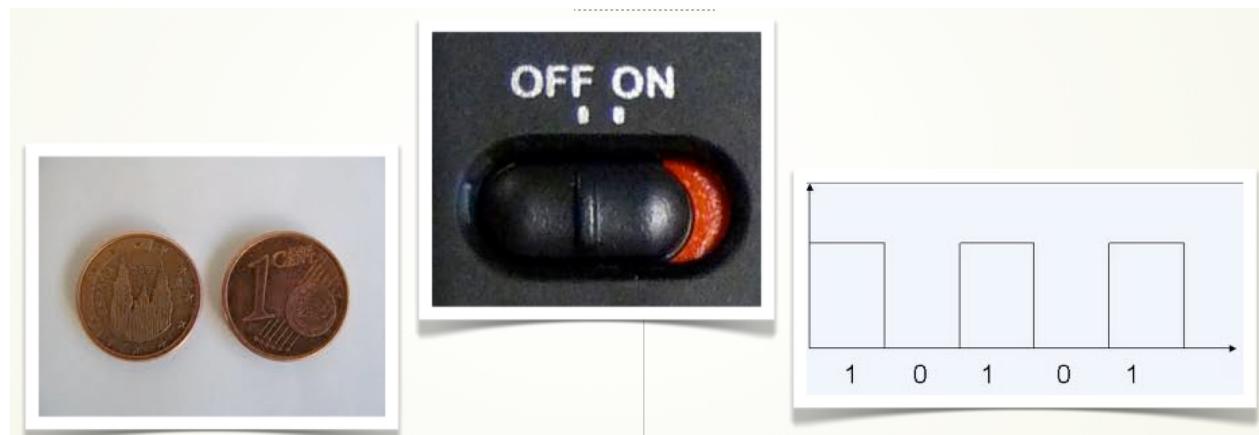
Montajes básicos con Arduino

Antes de empezar a realizar pequeños proyectos, vamos a dar un repaso a la plataforma Arduino, con las prácticas básicas que necesitarás para comprender los conceptos y avanzar en el libro .



via GIPHY

Conexiones digitales



En este apartado aprenderemos el funcionamiento básico de las entradas y salidas digitales de la placa Arduino. Si observamos bien la placa, vemos que hay 13 pines digitales.

En este caso la señal digital no es más que un valor discreto de entre dos posibles, que si en rigor se asocian a tensiones, nosotros por sencillez los asociaremos a dos valores que serán ,apagado o encendido o lo que es lo mismo LOW ó HIGH.

Así si asignamos un 0 al pin digital 4, es lo mismo que decir que ese pin, o mejor dicho, lo que esté conectado a ese pin estará apagado si le asignamos un 1, estamos diciendo que estará encendido.

Entonces, ¿Con los 13 pines digitales de Arduino , podríamos actuar para controlar 13 bombillas? . Si, aunque Arduino es aún más potente ya que aún podemos usar los 5 pines analógicos también como salidas digitales.

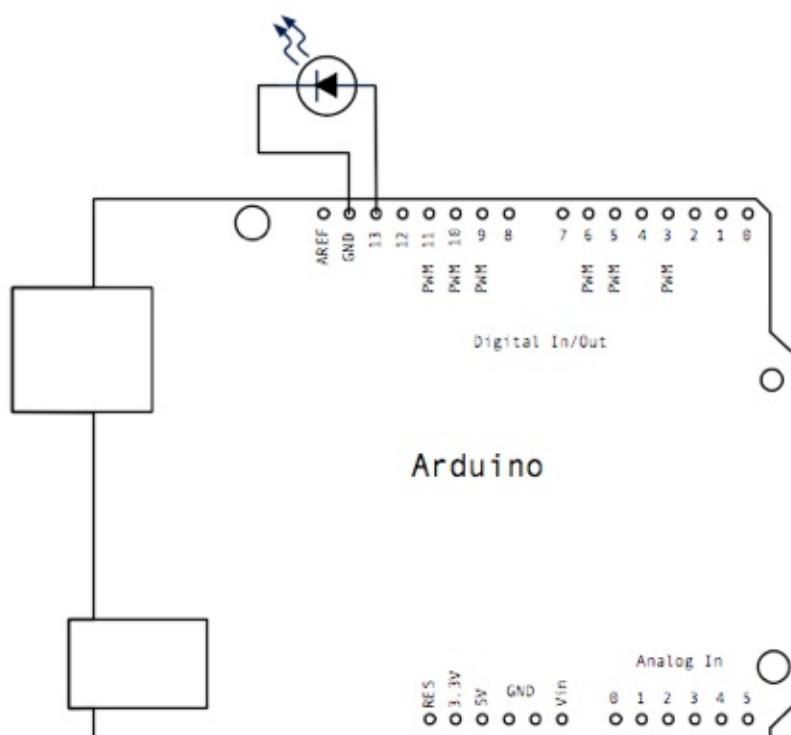
Veamos como.

Montaje 1: LED parpadeante sin EDUBASICA

Vamos a controlar el encendido y apagado de un led conectado al pin13 de Arduino.

¿Por qué el pin13 y no otro? Podríamos hacerlo con otro, pero el pin13 tiene asociado un led en la placa justo debajo de el y así nos evitamos tener que montar. Si pusieramos un pin polarizado correctamente entre el pin13 y GND también funcionaría. El pin13 tiene también una resistencia que hace posible conectarle un led directamente, si hacemos el montaje con otro pin debemos añadir esta resistencia de 10Kohm entre el led y el pin.

Acuérdate: La pata más larga del LED es el (+) por lo tanto en el D13 y el corto (-) en GND.



```

1 void setup() {
2     // inicializamos el pin 13
3     // para que sea de salida
4     pinMode(13, OUTPUT);
5 }
6 // Definimos la rutina cíclica
7 void loop() {
8     digitalWrite(13, HIGH);      // Encendemos el pin13
9     delay(1000);                // esperamos 1 segundo
10    digitalWrite(13, LOW);     // Apagamos el pin13
11    delay(1000);                // esperamos 1 segundo
12 }
```

Todo lo que está entre las llaves de **loop()**, se ejecuta indefinidamente. Así vemos un efecto de led parpadeante ya que si analizamos las líneas del código vemos que el proceso es:

- Encendemos.
- Esperamos un segundo.
- Apagamos.
- Esperamos un segundo.

¡Atrevámonos y cambiemos los tiempos de parada!

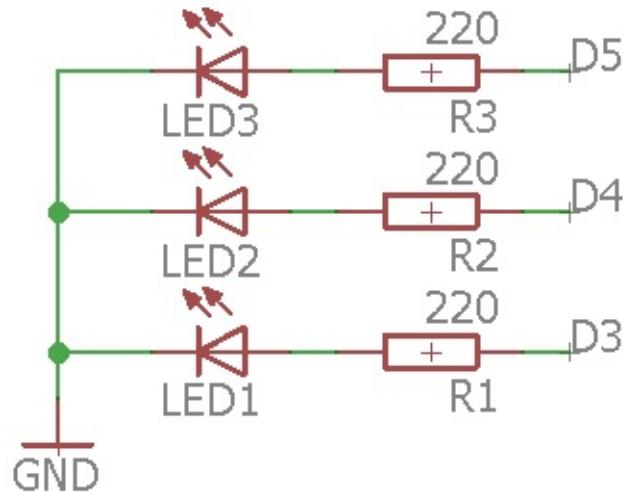


[Video link](#)

Montaje 2: LED EDUBASIC parpadeante

Igual que en el caso anterior, pero vamos a utilizar un LED de la shield de Edubásica, tenemos tres opciones:

- LED VERDE pin 3
- LED AMARILLO pin 4
- LED ROJO pin 5



El programa es igual que el anterior, pero cambiando el número del pin:

```

1 void setup() {
2     // inicializamos el pin 3 en Edubásica tiene el LED VERDE
3     // para que sea de salida
4     pinMode(3, OUTPUT);
5 }
6
7 // Definimos la rutina cíclica
8 void loop() {
9     digitalWrite(3, HIGH); // Encendemos el pin 3
10    delay(1000); // esperamos 1 segundo
11    digitalWrite(3, LOW); // Apagamos el pin 3
12    delay(1000); // esperamos 1 segundo
}

```





[Video link](#)

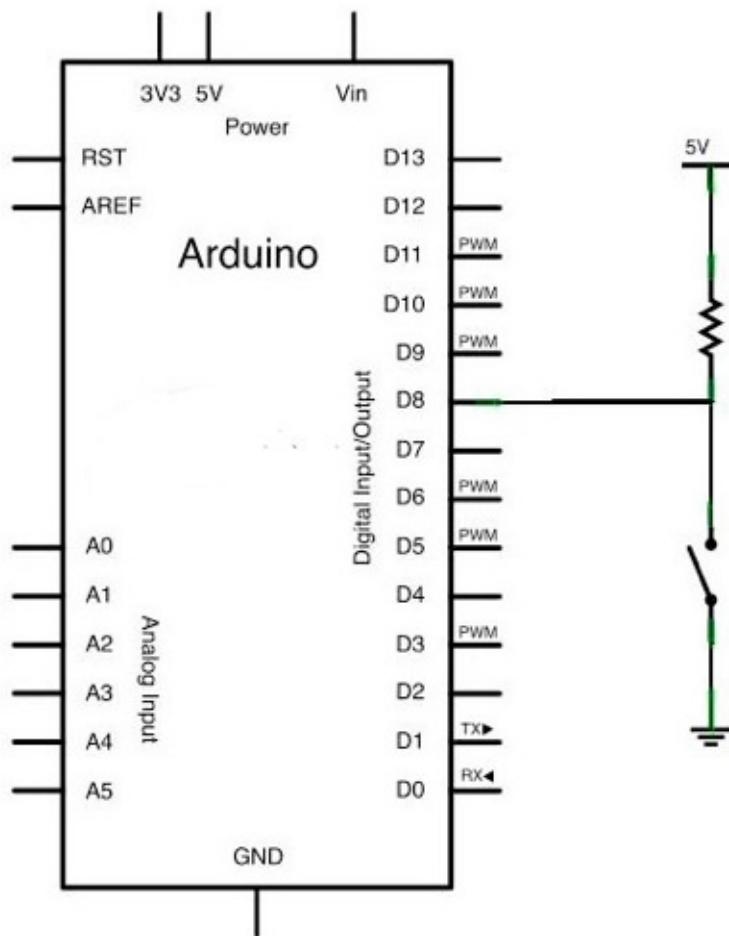
Resistencias pull-up y pull-down

En los proyectos con dispositivos digitales, caso de la placa Arduino, reciben señales de entradas digitales del exterior. Estas señales externas sirven para activar o desactivar un circuito, recibir información del estado de un sensor,...

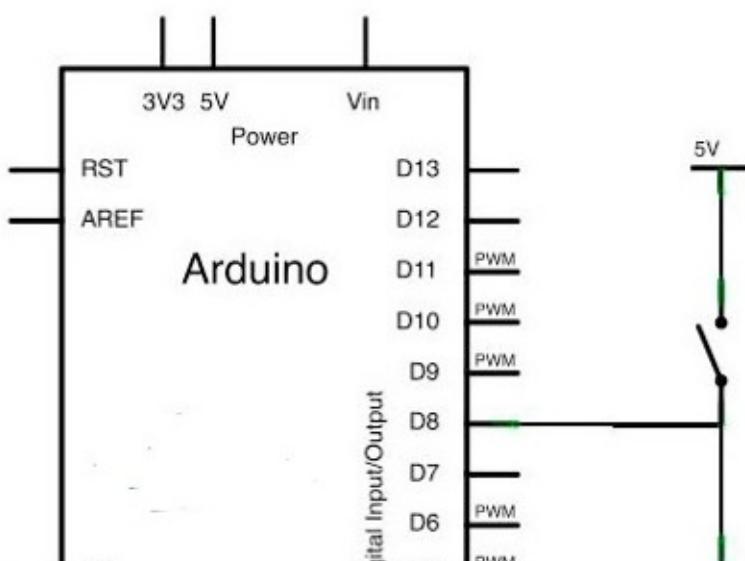
Las resistencias “pull-up” y “pull-down” son resistencias que se ponen en las entradas digitales para fijar un valor por defecto, nivel alto (“1”) o nivel bajo (“0”), cuando no se detecta ningún valor. Esto ocurre cuando la entrada no está conectada a nada.

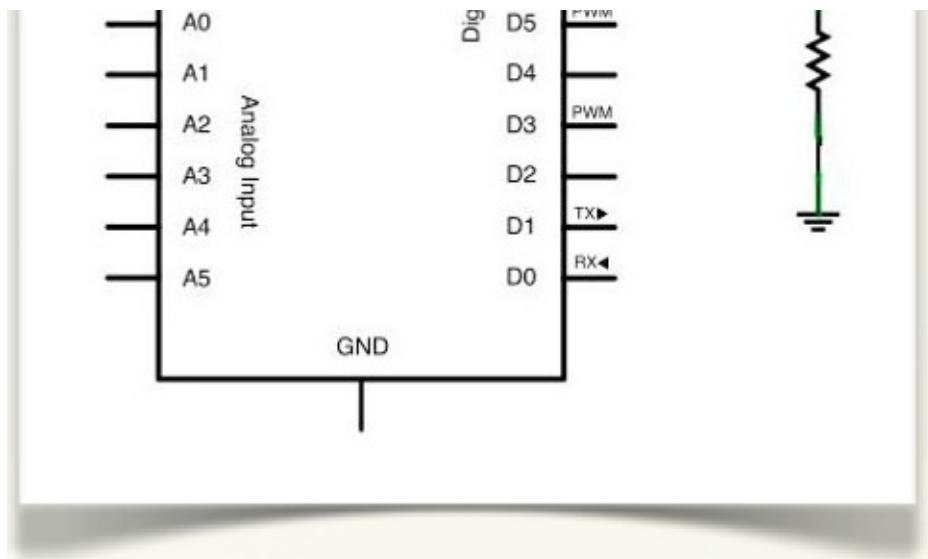
En el caso de tener que conectar un interruptor a una entrada digital de Arduino necesitamos hacerlo a través de una resistencia de este tipo. Veamos un ejemplo:

PULL UP



PULL DOWN

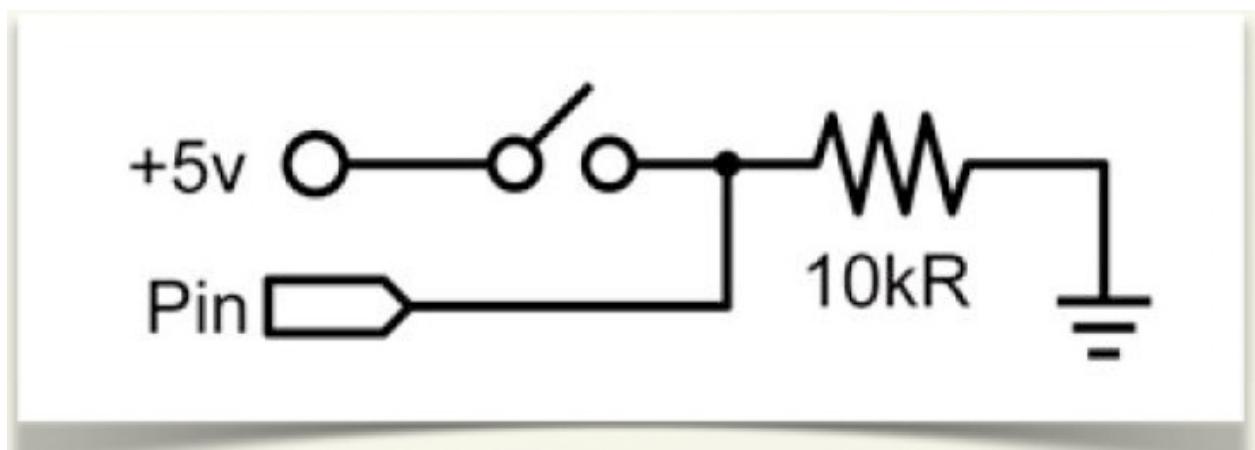




La resistencia “**pull-up**” establece un nivel alto (1) en la entrada digital en el estado de reposo del circuito. Un circuito con una entrada “pull-up” sería el siguiente.

La resistencia “**pull-down**” establece un nivel bajo (0) en la entrada digital en el estado de reposo del circuito. Este tipo de circuito es el más empleado en las entradas digitales para evitar lecturas erróneas debido a ruidos externos y consumo de energía. La resistencia suele ser de 10 kΩ y el circuito con esta configuración sería el siguiente.

Un ejemplo de circuito “**pull-down**” lo tenemos en la placa EduBásica en el pin digital D2, preparado para configurarlo como entrada, tiene conectado un pulsador y una resistencia pull-down. El esquema del circuito es el siguiente



El funcionamiento de este circuito que está conectado al pin digital D2 como entrada es detectar si el pulsador está pulsado o no.

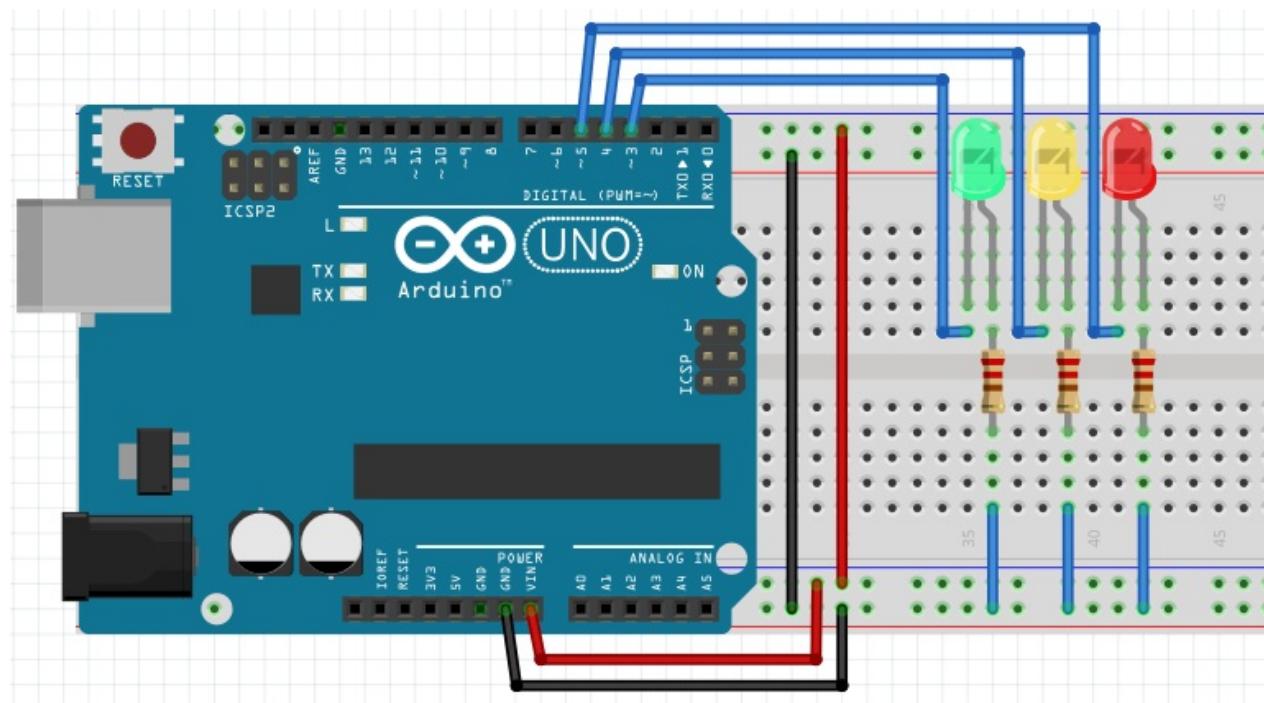
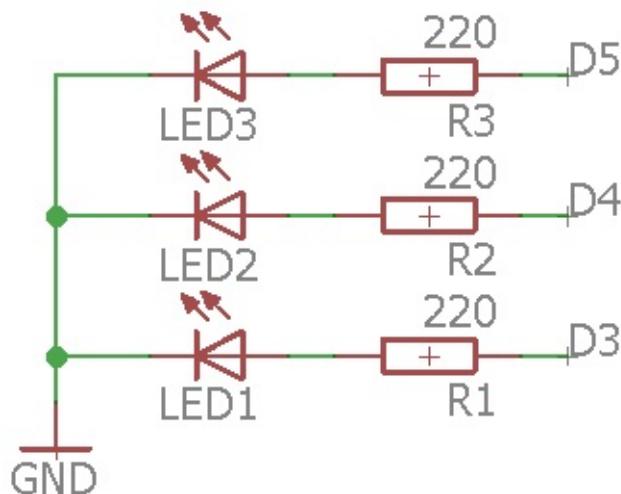
- Si el pulsador no está pulsado en Pin, que está conectado a D2, tenemos 0V por no pasar corriente entre el pin D2 y masa. Por tanto, corresponde a un nivel bajo o “0” lógico.
- Si el pulsador está pulsado en el pin D2 tenemos 5V, que corresponde a un nivel alto o “1” lógico.
- Si en el anterior circuito no se pone la resistencia de 10KΩ y el pulsador está abierto con el propósito de tener un nivel bajo porque no hay tensión, puede ocurrir y de manera aleatoria que el pin D2 lea un nivel alto por alguna interferencia producida por motores eléctricos, bobinas de un relé u otro dispositivo de nuestro proyecto.

Montaje 3: SEMAFORO CON EDUBASICA

Montaremos un semáforo con los tres leds de la EduBásica. La EduBásica es opcional y podemos montar el circuito correspondiente con una protoboard, pero EduBásica nos ahorra trabajo.

Montaje 3: SEMÁFORO SIN EDUBASICA

Necesitamos añadir una resistencia entre el pin y el led, para evitar que el led se funda.



en este caso tienes libertad de utilizar D3 D4 D5 o otros que quieras.

Continuamos ...

Como EduBásica ya lleva estas resistencias integradas las capturas las hacemos con ella, no hay que hacer nada.

Vamos a ver el efecto que queremos realizar:



[Video link](#)

Carga el programa de la página siguiente en Arduino y verás como actúa.

Aparece un nuevo comando: **Serial.print**.

Este comando nos manda un texto al puesto serial por el que nos comunicamos con Arduino. De esta manera podemos depurar un programa sabiendo siempre por qué línea está.

Para que funcione debemos tener en cuenta que:

- Hay que inicializar Serial. Esto se hace poniendo **Serial.begin(9600)** dentro de la rutina de `setup()`. 9600 se refiere a la velocidad que se comunicará.
- `Serial.print("xxx")` escribe lo que ponemos entre comillas tal cual.
- `Serial.print(x)` escribe el valor que contenga la variable `x`.
- `Serial.println()` es similar a lo anterior pero después añade un salto de línea.

Para ver lo que nuestro Arduino nos comunica por Serial, abrimos el monitor Serial que tenemos en el programa Arduino:

sketch_oct08a Arduino 1.8.5

Archivo Editar Programa Herramientas Ayuda

Auto Formato Ctrl+T
 Archivo de programa.
 Reparar codificación & Recargar.

Monitor Serie Ctrl+Mayús+M

Serial Plotter Ctrl+Mayús+L

WiFi101 Firmware Updater

Placa: "Arduino/Genuino Uno" >
 Puerto >
 Obtén información de la placa

Programador: "AVRISP mkII" >
 Quemar Bootloader

```
* Semáforo Arduino
  Leds conectados a :
  int verde = 3;
  int amarillo = 4;
  int rojo = 5;

void setup()
{
  pinMode(verde, OUTP
  pinMode(amarillo, O
  pinMode(rojo, OUTPU
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Semaforo - Inicio"); //Escribe el texto
  digitalWrite(verde, HIGH);
  Serial.println("Semaforo - Verde"); //Escribe el texto
  delay(30000);

  Serial.println("Semaforo - Amarillo");
  Serial.println("Semaforo - Rojo");
  Serial.println("Semaforo - Inicio");
  Serial.println("Semaforo - Verde");
  Serial.println("Semaforo - Amarillo");
  Serial.println("Semaforo - Rojo");
  Serial.println("Semaforo - Inicio");
  Serial.println("Semaforo - Verde");
  Serial.println("Semaforo - Amarillo");
  Serial.println("Semaforo - Rojo");
}
```

/dev/tty.usbmodem411

Enviar

Semáforo - Inicio
 Semaforo - Verde
 Semaforo - Amarillo
 Semaforo - Rojo
 Semaforo - Inicio
 Semaforo - Verde
 Semaforo - Amarillo
 Semaforo - Rojo
 Semaforo - Inicio
 Semaforo - Verde
 Semaforo - Amarillo
 Semaforo - Rojo
 Semaforo - Inicio
 Semaforo - Verde
 Semaforo - Amarillo
 Semaforo - Rojo



Semaforo - Inicio
Semaforo - Verde

Desplazamiento automático Ambos NL

```
1  /* Semáforo Arduino
2   Leds conectados a pines 3, 4 y 5 */
3  int verde = 3;
4  int amarillo = 4;
5  int rojo = 5;
6
7  void setup()
8  {
9    pinMode(verde, OUTPUT);
10   pinMode(amarillo, OUTPUT);
11   pinMode(rojo, OUTPUT);
12   Serial.begin(9600); //inicializa la comunicación Serial
13 }
14
15 void loop()
16 {
17   Serial.println("Semaforo - Inicio"); //Escribe el texto
18   digitalWrite(verde, HIGH);
19   Serial.println("Semaforo - Verde"); //Escribe el texto
20   delay(30000);
21   digitalWrite(verde, LOW);
22   digitalWrite(amarillo, HIGH);
23   Serial.println("Semaforo - Amarillo"); //Escribe texto
24   delay(8000);
25   digitalWrite(amarillo, LOW);
26   digitalWrite(rojo, HIGH);
27   Serial.println("Semaforo - Rojo"); //Escribe el texto
28   delay(20000);
29   digitalWrite(rojo, LOW);
30 }
```

Montaje 4: Pulsador

Hasta ahora hemos visto como programar Arduino para que ejecute repetitivamente acciones, pero este actuaba de manera autónoma y nosotros sólo podíamos observar. Pero podemos interactuar con Arduino, por ejemplo, realizando una acción cuando activamos un pulsador. .

En este ejemplo, vamos a encender un led cuando actuamos sobre el pulsador.

sin EDUBASICA

Utilizamos por ejemplo el pin 2 corresponde al pulsador y el pin 3 al led verde, solo nos queda cargar el programa y probar.

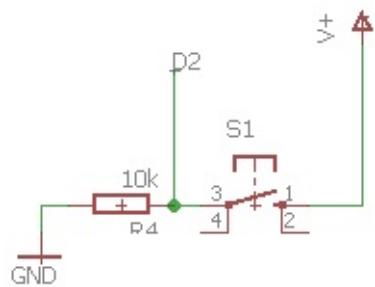


Imagen - Esquema del pulsador en EDUBASICA

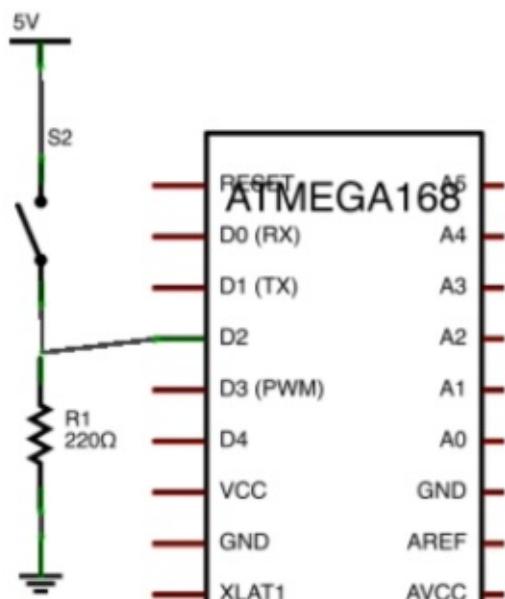


Imagen - Esquema eléctrico del pulsador conectado al micro del Arduino

con EDUBASICA

Con la misma estructura (D2 es donde está el pulsador en EDUBASICA y D3 el LED VERDE) no hace falta realizar ningún cableado.

Por sencillez usaremos EduBásica en las capturas. Vamos a verlo en acción:



[Video link](#)

Continuamos...

Aparece un comando nuevo “**digitalRead(buttonPin)**” . Retorna el valor del pin que se ha configurado como entrada y al igual que en el caso de los pines que se configuran como salida, puede tener dos valores HIGH y LOW.

Si es HIGH significa que este pin está unido a la señal de 5v, si es LOW significa que está unido a 0v.

¿Por qué cuando el pulsador está en OFF D2 está a 0V? espero que esta cuestión la has resuelto [en la sección anterior](#).

```

1 const int buttonPin = 2;      // Pin del pulsador
2 const int ledPin = 3;        // Pin del Led
3
4 void setup() {
5     // pin del led de salida:
6     pinMode(ledPin, OUTPUT);
7     // pin del pulsador de entrada
8     pinMode(buttonPin, INPUT);
9 }
10
11 void loop(){
12
13     // Si el valor del pin del pulsador es HIGH es que esta pulsado
14     if (digitalRead(buttonPin) == HIGH) {
15         // Se enciende el LED:
16         digitalWrite(ledPin, HIGH);
17     }
18     else {
19         // Se apaga el LED:
20         digitalWrite(ledPin, LOW);
21     }
22 }
```

Otra opción es utilizar este programa donde se ve más visual:

```

1 // Indicamos que pin corresponde con cada LED:
2 int ledVerde = 5;
3 int ledAmarillo = 4;
4 int ledRojo = 3;
5 // El pulsador esta conectado al pin 2
6 int pulsa = 2;
7
8 // configuracion de pines
9 void setup() {
10 }
```

```
11 // los pines con leds son de salida
12 pinMode(ledVerde, OUTPUT);
13 pinMode(ledAmarillo, OUTPUT);
14 pinMode(ledRojo, OUTPUT);
15 pinMode(pulsa, INPUT); //pin de entrada
16 }
17
18 void loop() {
19 //Leemos del pin 2 para saber si se esta pulsando el boton
20 if (digitalRead(pulsa) == HIGH) //Si esta pulsado ENCENDEMOS
21 {
22 digitalWrite(ledVerde, HIGH);
23 digitalWrite(ledAmarillo, HIGH);
24 digitalWrite(ledRojo, HIGH);
25 }
26 else
27 {
28 digitalWrite(ledVerde, LOW);
29 digitalWrite(ledAmarillo, LOW);
30 digitalWrite(ledRojo, LOW);
31 }
32 }
```

Conexiones analógicas

Las entradas analógicas se utilizan para leer la información de la magnitud física que nos proporciona los sensores de temperatura, luz, distancia,... La tensión que leemos del sensor no la proporciona un circuito asociado a dicho sensor en un rango de valores de tensión continua entre 0V y 5V.



La placa de Arduino tiene 6 entradas analógicas marcados como “A0”, “A1”,..., “A5” que reciben los valores continuos en un rango de 0V a 5V, pero la placa Arduino trabaja sólo con valores digitales, por lo que es necesario una conversión del valor analógico leído a un valor digital. La conversión la realiza un circuito analógico/digital incorporado en la propia placa.

El conversor A/D de la placa tiene 6 canales con una resolución de 10 bits. Estos bits de resolución son los que marcan la precisión en la conversión de la señal analógica a digital, ya que cuantos más bits tenga más se aproxima al valor analógico leído. En el caso de la placa Arduino el rango de los valores analógicos es de 0 a 5 V y con los 10 bits de resolución se puede obtener de 0 a 1023 valores digitales y se corresponde cada valor binario a $(5V/1024) 5 mV$ en el rango analógico.

En estas condiciones son suficientes para hacer muchos proyectos tecnológicos. En el caso de necesitar mayor resolución y como no podemos aumentar el número de bits de conversor A/D se puede variar el rango analógico utilizando el voltaje de referencia Vref.

Las entradas analógicas tienen también la posible utilización como pines de entrada-salida digitales, siendo su enumeración desde 14 al 19.

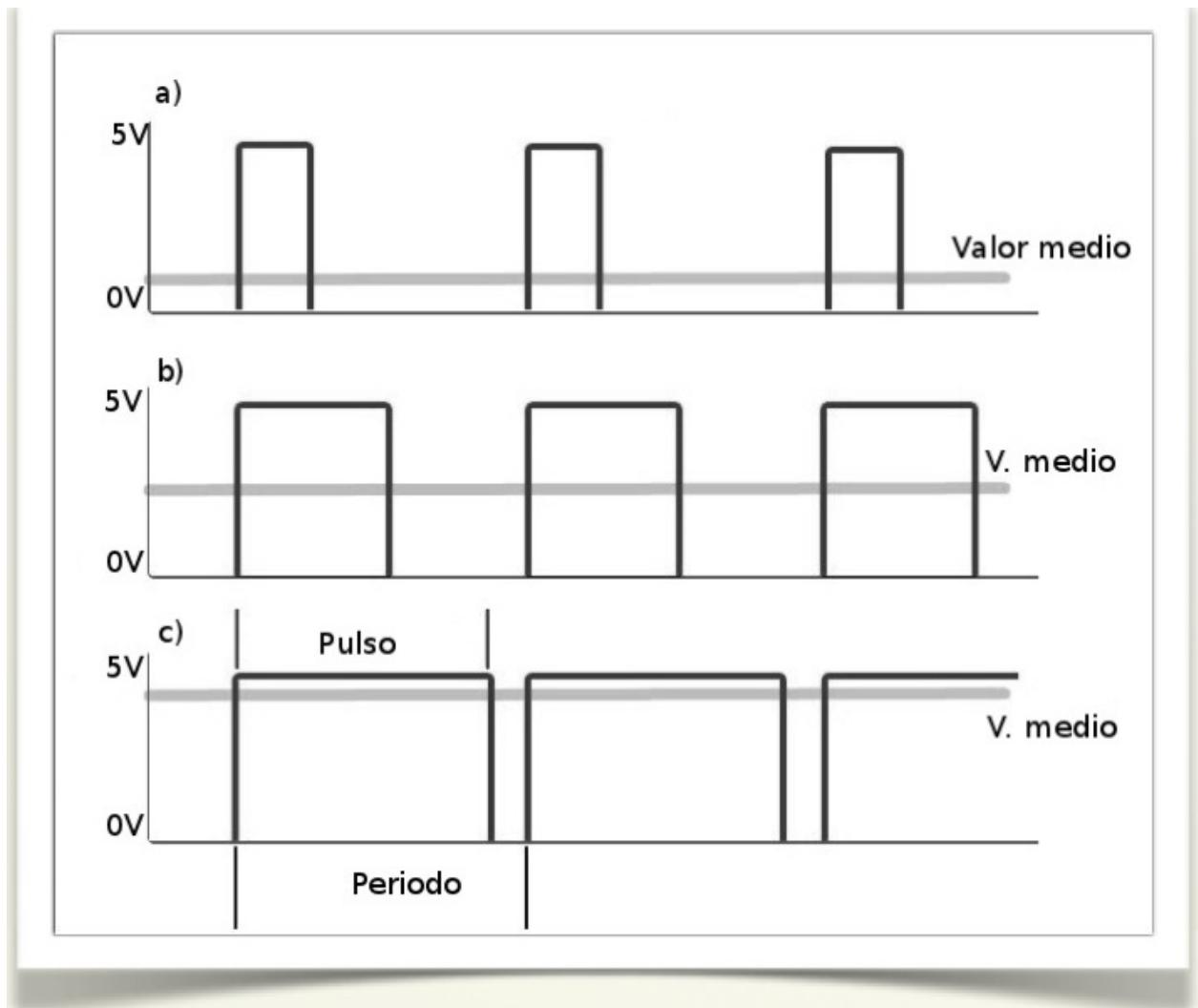
En este manual dedicamos un capítulo completo a analizar la forma en que Arduino lee magnitudes analógicas. Asimismo veremos numerosos montajes en los que se utilizan estos pines para lectura de sensores.

Un caso especial: Señales PWM

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

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

Veamos gráficamente la señal PWM:



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

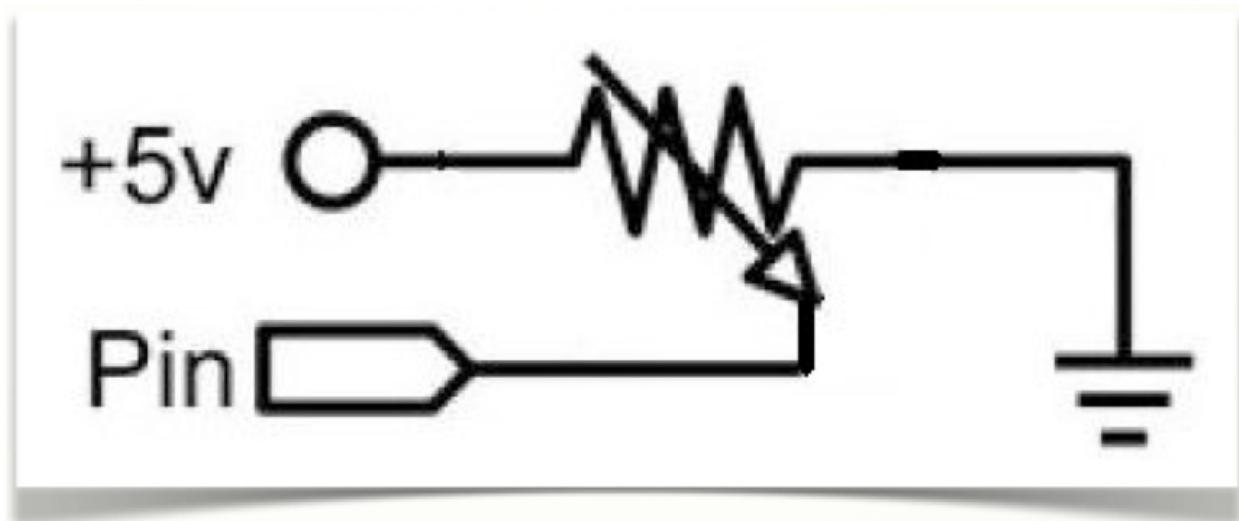
- La duración del pulso es pequeña y la salida va a tener un valor medio de tensión bajo, próximo a 0V.
- La duración del pulso es casi la mitad del período de la señal, por tanto, la salida va a tener un valor medio de tensión próximo a 2,5V.
- La duración del pulso se aproxima al tiempo del período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

Montaje 5: Control de la intensidad de iluminación de un LED

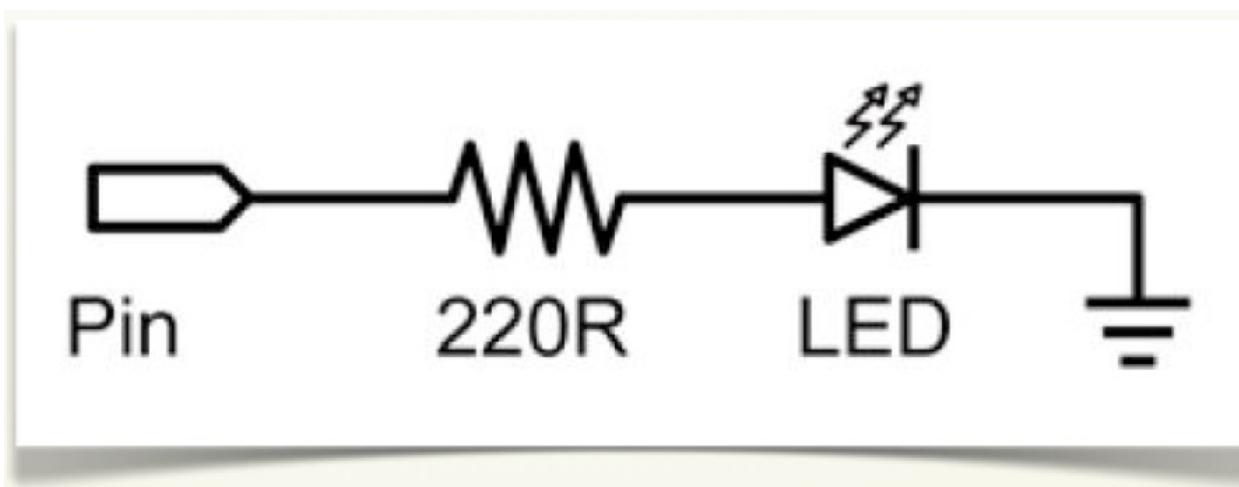
Como ejemplo práctico de la señal PWM vamos a realizar un control de iluminación sobre un diodo led.

Sin EDUBASICA

En una protoboard montamos el circuito formado por el **potenciómetro** conectado a la entrada analógica A0.



y también montamos el circuito conectado al pin digital D3, utilizado como salida PWM, de esta manera nos va a permitir variar la luminosidad del LED.



CON EDUBÁSICA

En este caso ya tiene integrado un potenciómetro conectado a la entrada A0.

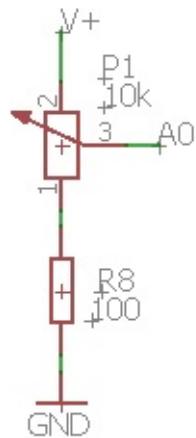


Imagen - Esquema del potenciómetro en EDUBASICa

Y vamos a utilizar de salida el diodo verde conectado a D3 y a una resistencia ya integrado en EDUBASICa:

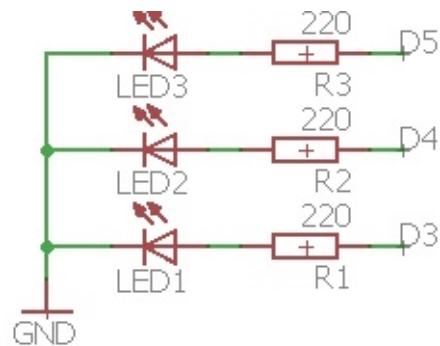
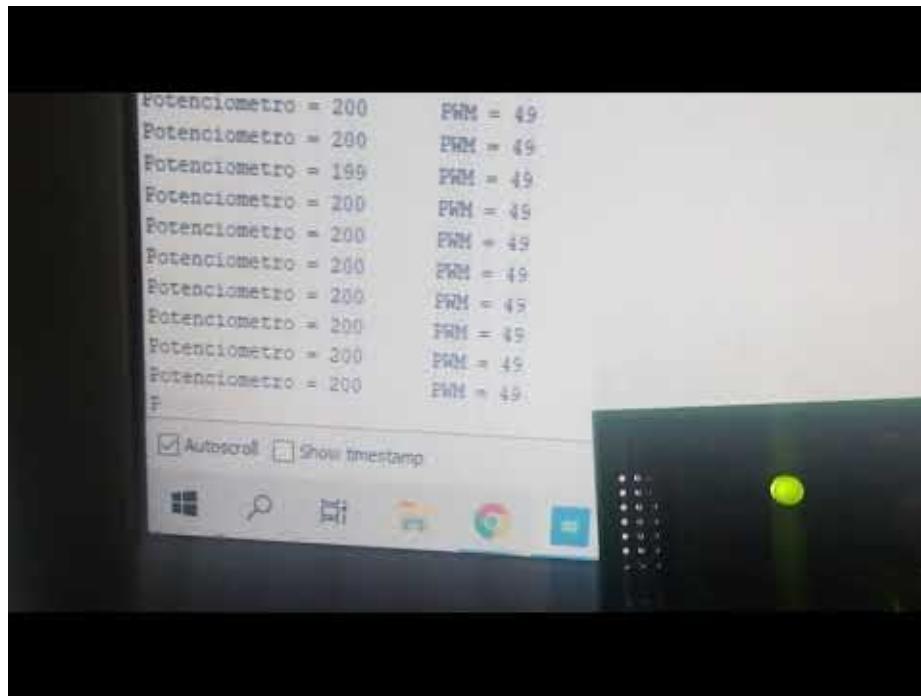


Imagen - Esquema de LEDs en EDUBASICa

Continuamos...

Vamos a ver una pequeña demostración:

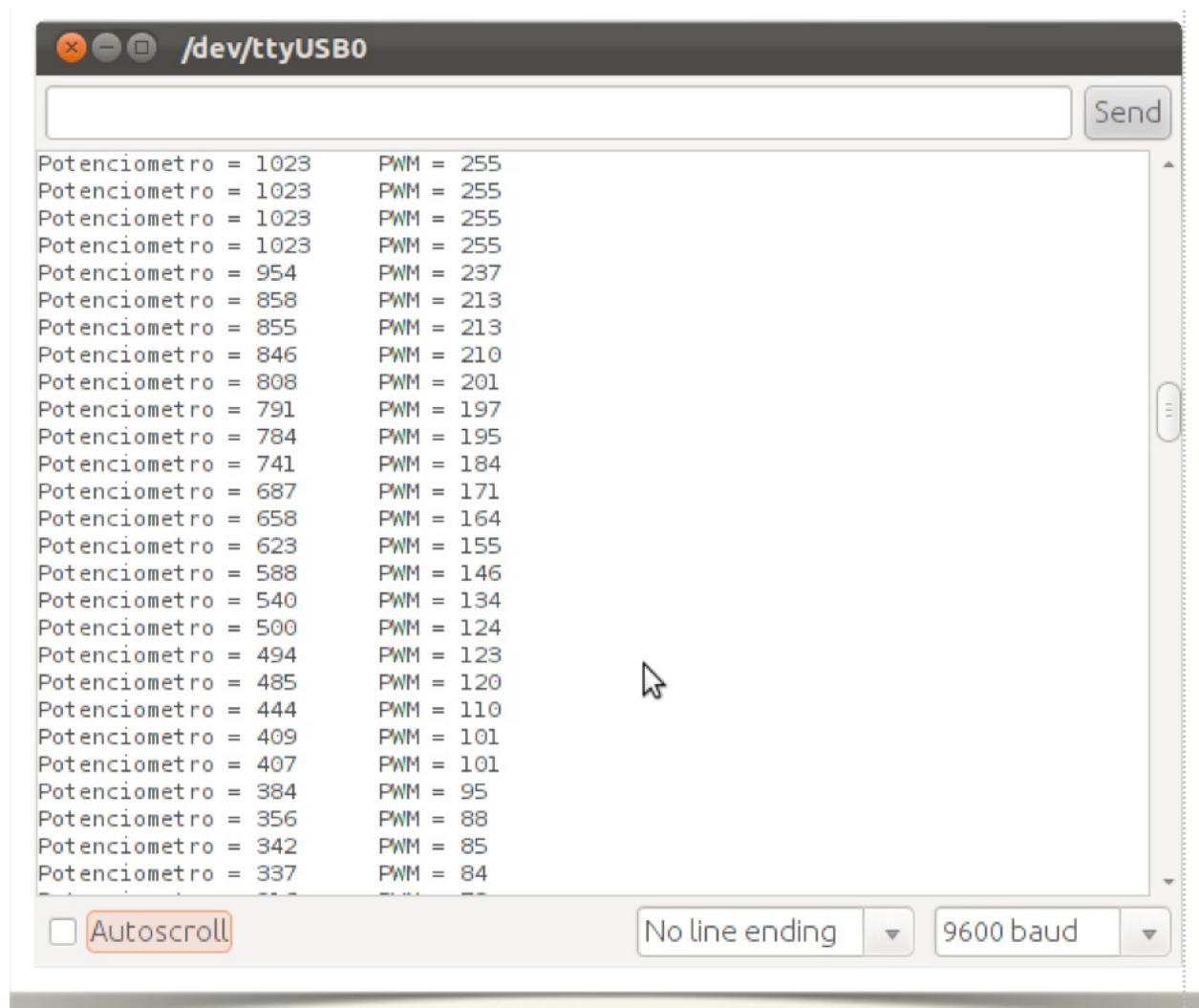


[Video link](#)

Cargamos el programa ejemplo, en la placa Arduino y teniendo acoplada la placa EduBásica o montados los circuitos en una placa protoboard, podemos ver su funcionamiento.

Para ver su funcionamiento activamos el monitor serie del IDE de programación y variamos el potenciómetro. El resultado es una variación de luminosidad y variación de valores en el monitor serie.

El siguiente gráfico es una pantalla del monitor serie con los valores leídos y el valor aplicado a la señal PWM para variar la luminosidad.



```

1  /*Lee la entrada analogica A0, mapea el resultado al rango de 0 a 255
2  y utiliza el resultado para poner la anchura del pulso PWM.
3  Tambien se escribe en el monitor serie el valor binario de A0 y
4  el valor mapeado para la señal PWM.
5  De esta manera controlamos la luminosidad del led verde
6  de la placa Edubasica
7
8  El circuito:
9  * potenciometro conectado a la entrada analogica A0.
10 Terminal central del potenciometro a pin A0.
11 Resto de terminales del potenciometro a +5V y masa
12 * Circuito de LED verde conectado a D3-PWM.
13 */
14
15 // pines usados:
16 const int analogInPin = A0; // Entrada analogica A0 del potenciometro
17 const int analogOutPin = 3; // Salida PWM
18
19 int potValor = 0; // valor de lectura del potenciometro
20 int outputValor = 0; // valor de salida de la señal PWM
21
22 void setup() {
23

```

```
24 // inicializacion del monitor serie a 9600 bps:  
25 Serial.begin(9600);  
26 }  
27  
28 void loop() {  
29 // lee el valor de la entrada analogica:  
30 potValor = analogRead(analogInPin);  
31 // mapea el rango para la señal de salida PWM:  
32 outputValor = map(potValor, 0, 1023, 0, 255);  
33 // asigna el valor cambiado a pin 3 PWM:  
34 analogWrite(analogOutPin, outputValor);  
35  
36 // escribe el resultado en el monitor serie:  
37 Serial.print("Potenciómetro = ");  
38 Serial.print(potValor);  
39 Serial.print("\t PWM = ");  
40 Serial.println(outputValor);  
41  
42 // espera 1 segundo cada bucle para una visualizacion aceptable  
43 // conviene tener un valor aunque sea pequeño (10ms)  
44 // por el proceso de conversion de A/D  
45 delay(10);  
}
```

3 ENTRADAS Y SALIDAS

Vamos a profundizar más en el Arduino y veremos como podemos añadir dispositivos de entrada y salida que nos permitirá la interacción más emocionante



via GIPHY

Si tienes dudas técnicas en este capítulo pon un ticket a <http://soporte.catedu.es/> y te ayudaremos:



Montaje : Potenciómetro

Vamos a realizar un programa para comprobar que al variar el valor de una resistencia mediante un potenciómetro, también variará la cantidad de luz que emite un led. Como se puede ver en el siguiente vídeo, a medida que giramos el potenciómetro el led varía su luminosidad.

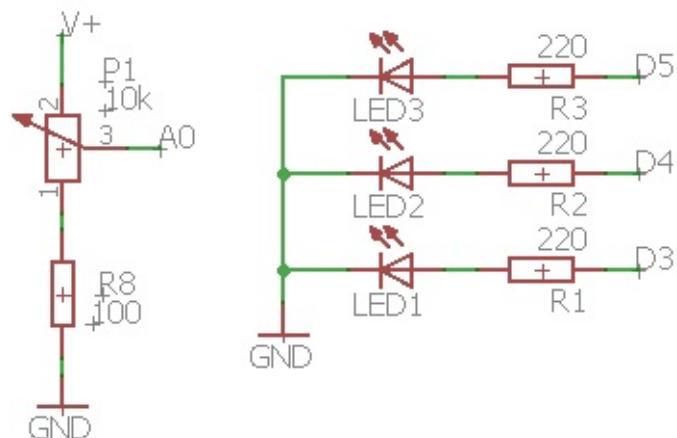


Imagen - YouTube video thumbnail

[Video link](#)

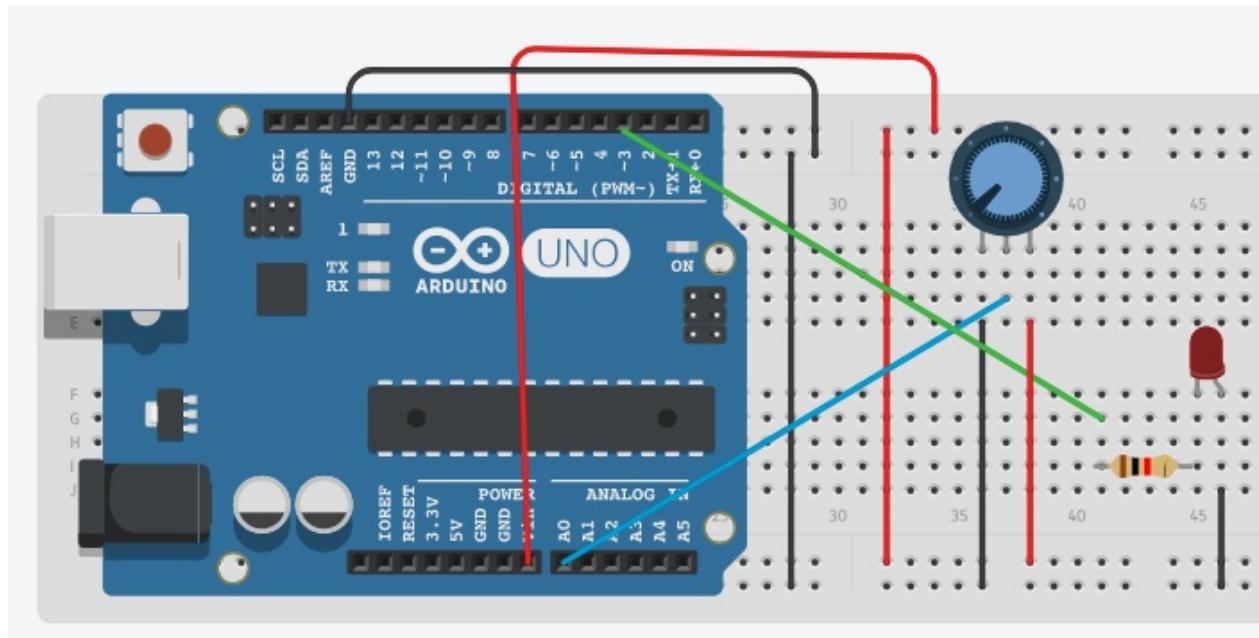
CON EDUBASICA

Vamos a aprovechar el A0 que está conectado al potenciómetro y utilizaremos el D3 que está conectado al LED VERDE



SIN EDUBASICA

Pues se necesita hacer el cableado correspondiente A0 con el potenciómetro y D3 a un led:



Continuamos ...

El programa sería el siguiente

```

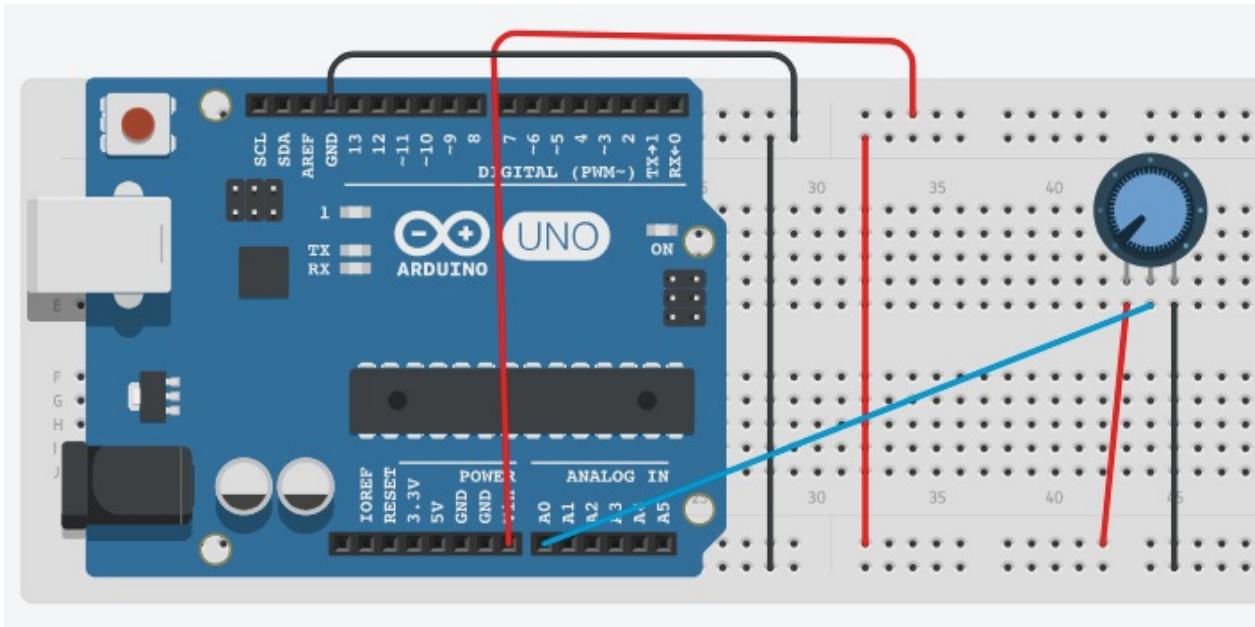
1 //Declaramos una variable para almacenar el valor recibido en pin 0.
2 int val = 0;
3
4 void setup() {
5 //El pin 3 corresponde al LED verde
6 pinMode(3, OUTPUT);
7 }
8
9 void loop() {
10 /*leemos el valor del pin 0 modificado por el potenciómetro que va desde 0
11 a 1023*/
12 val = analogRead(0);
13 /*Escribimos el valor obtenido en el LED verde que puede ser entre 0 y 255.
14 Por eso dividimos val por 4*/
15 analogWrite(3, val/4);
16 }
```

Montaje : Mapeo Potenciómetro

Vamos LEER LOS VALORES DEL POTENCIÓMETRO EN PANTALLA

SIN EDUBÁSICA

Conectamos la salida de un potenciómetro a A0



CON EDUBASICA

No hay que hacer nada, ya está !

Continuamos ...

Fíjate como **dividimos los valores** del potenciómetro A0 (que va desde 0 a 1024) **entre 204.6** para convertirlos dentro del rango de 0-5V ($1024/5=204.6$) :

```

1 float val = 0;
2
3 void setup(){
4 Serial.begin(9600);
5 }
6 void loop(){
7 val = analogRead(A0);
8 //leemos el potenciómetro (0-1024)
9 val = val/204.6;
10 //mapeamos los valores para que sean de 0 a 5V
11 Serial.print (val);
12 //vemos por pantalla el valor en Voltios
13 Serial.println("V");
14 delay(1000);

```

Este es el vídeo:



[Video link](#)

Mapeo de valores

En ocasiones, los valores que obtenemos de una lectura de un pin, como un sensor, pueden estar fuera de una escala determinada, y tenemos que convertirlos a otro rango para poder usarlos.

Por ejemplo:

Supongamos que hacemos una lectura previa de lo que nos devuelve el LDR y nos devuelve unos valores mínimo y máximo por ejemplo: **917, 1024** y queremos traducir estos valores al rango **0-255** pues es lo que podemos darle a un LED. La solución será _mapear _esos valores para que, en caso de obtener el valor 917 (el equivalente a oscuridad), el led verde se apague (0) y la máxima luminosidad (1024) el led se ilumine al máximo (255).

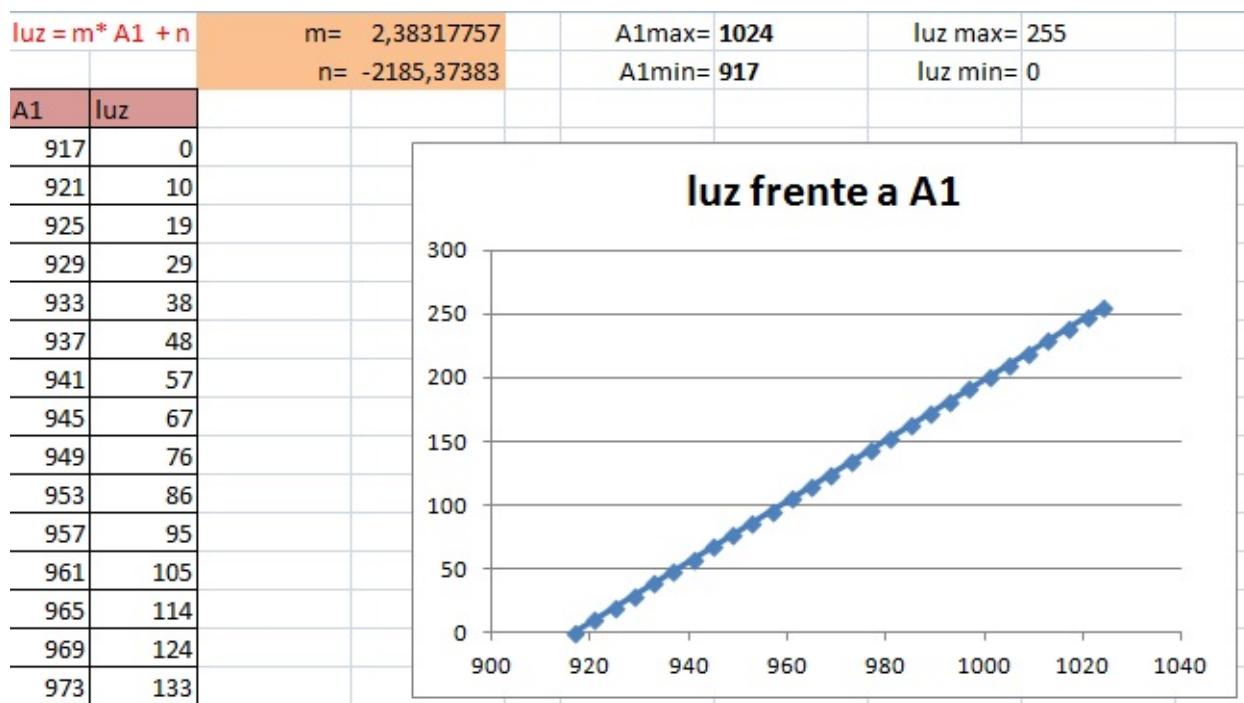
La función “**map**” del programa asigna un valor máximo y un valor mínimo a un rango dado.

_Nota: En el próximo montaje el valor máximo suele estar en 1024, pero el mínimo dependerá de las condiciones de luz en las que realicemos la práctica. Por eso en el código se especifican 2 variables que deberemos colocar a mano:

bajo_LDR y **alto_LDR**.

Observa si tu montaje necesita de algún ajuste utilizando la función **map**.

[Aquí tienes un Excel](#) (xlsx - 12,88 KB) para ver la transformación lineal que hace map con los valores que hemos comentado, es simplemente una demostración de lo que hace internamente la función **map**



Montaje : LDR

Hasta ahora hemos trabajado con resistencias de valor fijo, pero existen una serie de resistencias que varían según distintos parámetros físicos a las que se les somete como presión, luz y temperatura entre otros. Existe una gran variedad que se utilizan para construir lo que llamamos sensores.

En esta práctica vamos a diseñar un circuito que sea sensible a la luz. El objetivo **será regular la intensidad luminosa de un led con una LDR**, una resistencia sensible a la luz.

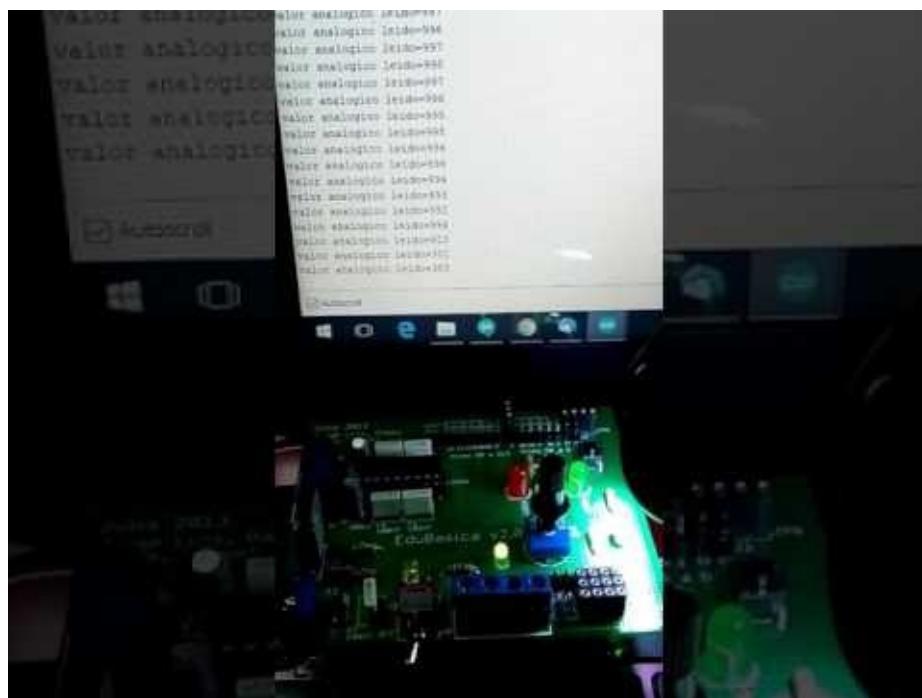
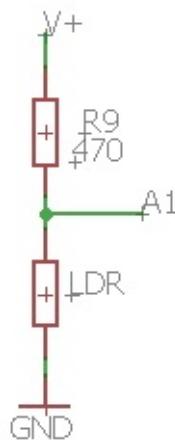


Imagen - YouTube video thumbnail

[Video link](#)

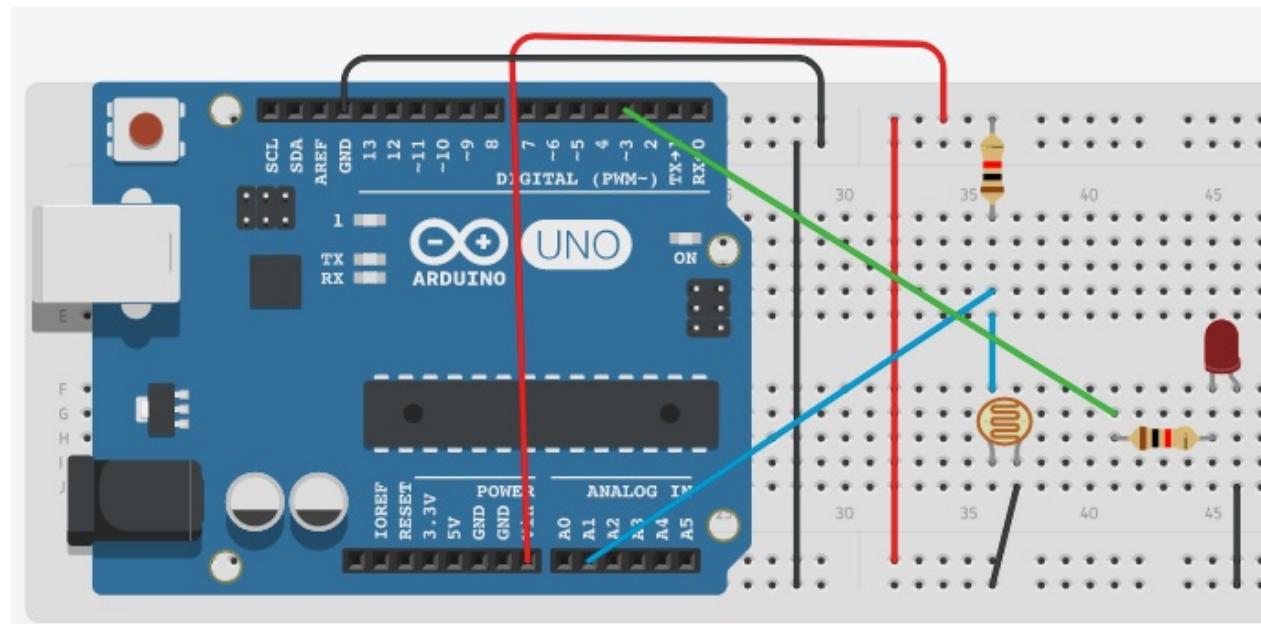
CON EDUBASICA

En este montaje usaremos la resistencia LDR de la placa Edubásica. Como ya hemos comentado, la LDR modifica su resistencia en dependiendo de la cantidad de luz que incida sobre ella. El siguiente programa mostrará por consola (“Monitor Serial”) las variaciones de luminosidad que detecte la LDR simplemente pasando la mano por encima de ella.



SIN EDUBASICA

Hay que conectar en formato pull-down el LDR a A1 y un led verde a D3:



CONTINUAMOS....

Como vemos en el esquema, el LDR está conectado a la entrada A1 del Arduino, por lo tanto la instrucción de lectura será ***analogRead(1)*** lo mapearemos correctamente a una variable llamada ***luz*** y utilizaremos el LED Verde conectado a D3 por lo que la instrucción de salida será ***analogWrite(3,luz)***

Nota: Si no sabes de dónde salen los valores 917,1024, 0,255 es que no has leido la página [Mapeo de valores](#)

```

1 //Detectamos la variación en la LDR
2
3 int luz;
4 void setup() {
5     // Pin 3 tiene el led verde
6     pinMode(3, OUTPUT);
7     Serial.begin(9600);
8 }
```

```
9 void loop() {  
10    luz= map(analogRead(1), 917, 1024, 0, 255);  
11    analogWrite(3,luz);  
12    //escribimos el valor obtenido por la resistencia para ajustarlo  
13    Serial.print("valor analogico leido=");  
14    Serial.println(analogRead(1));  
15    delay (1000);  
16 }
```

Montaje Regular intensidad de un LED

Se trata de obtener el mismo efecto que se consiguió en la práctica correspondiente al potenciómetro, las instrucciones principales eran:

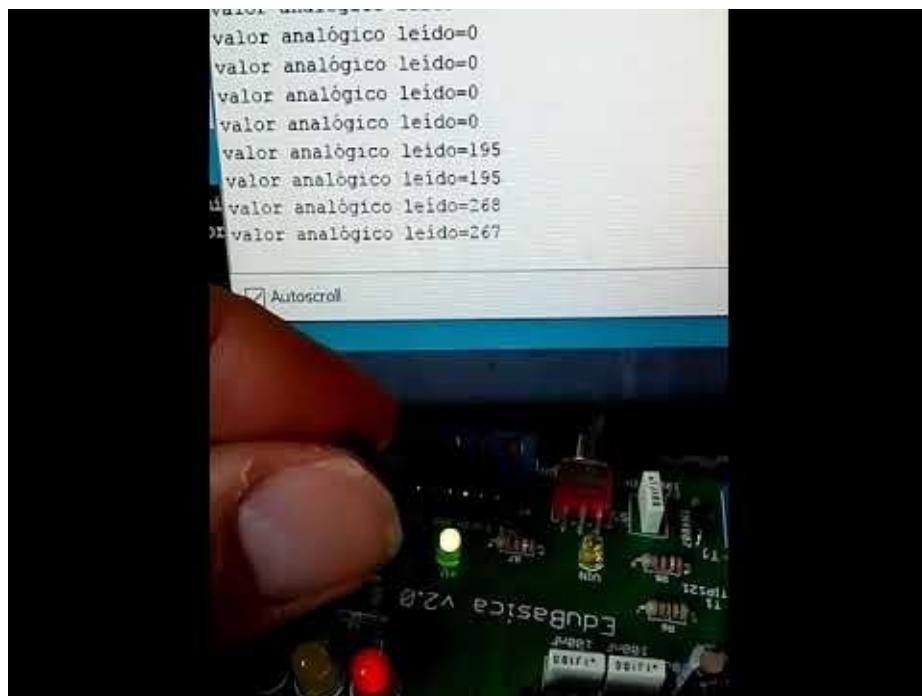
```

1 void loop() {
2     val = analogRead(0); // val tiene el rango 0-1024
3     analogWrite(3, val/4); //dividimos por 4 para entrar en el rango 0-255
4 }
```

Pero en este caso utilizaremos el mapeo.

La descripción de la práctica es la siguiente:

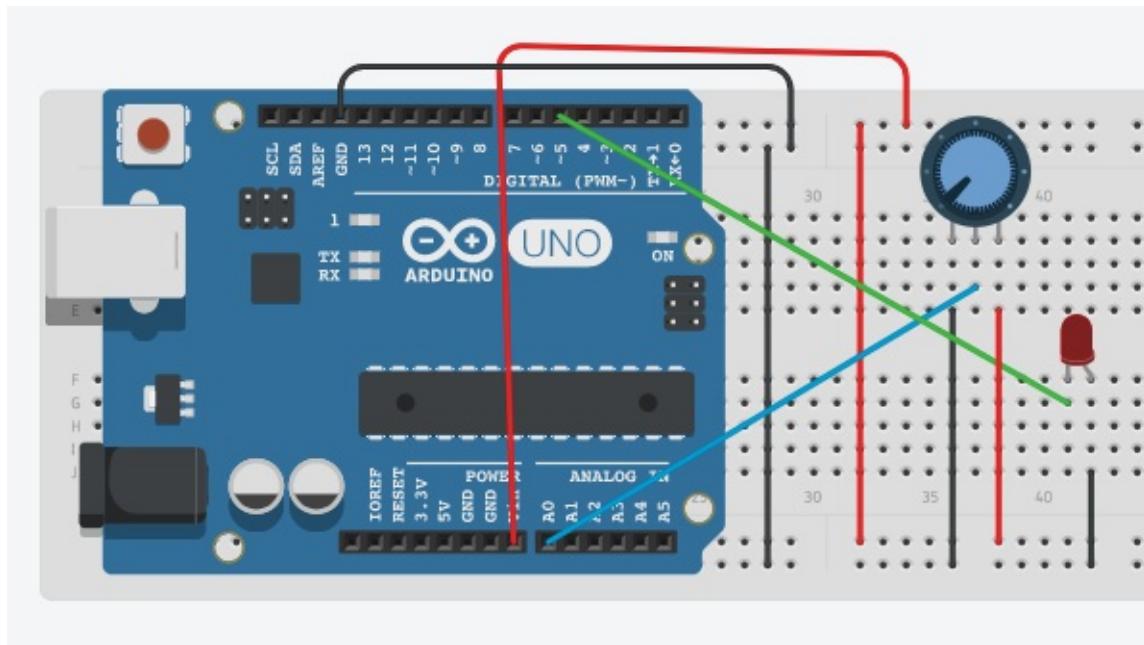
La regulación del potenciómetro provocará una variación de voltaje en el pin de entrada analógico 0 de Arduino. Se realizará una conversión analógica-digital en el que los valores de tensión analógicos entre 0 y 5 V se transforma a un rango discreto de valores de 0 a 1023. Para modificar la intensidad del led rojo le se enviará una señal pseudoanalógica PWM utilizando la salida 5 digital de Arduino. Para ello se enviará un valor de 0 a 255 que marcará el ciclo de trabajo de la onda cuadrada PWM. Previamente habrá que realizar un mapeo (instrucción map) para asignar valores desde el intervalo [0, 1023] al [0, 255].



[Video link](#)

SIN EDUBASICA

Tienes que conectar un led al D5 y un potenciómetro al A0



CON EDUBÁSICA

No tienes que conectar nada ;) ya está !

Continuamos

Este es el **PROGRAMA**:

```

1  /* Regular la luminosidad del LED rojo con el potenciómetro de Edubásica
2   Conexiones:
3   Pin 5 digital Arduino -> LED rojo Edubásica
4   Pin 0 analógico Arduino -> Potenciómetro Edubásica
5   */
6
7   int ledPin = 5;
8   int potenPin = A0;
9   int intensity, valor_poten;
10
11 void setup() {
12   pinMode(ledPin, OUTPUT);
13   Serial.begin(9600);
14 }
15
16 void loop() {
17   valor_poten=analogRead(potenPin);
18   intensity = map(valor_poten, 0, 1024, 0, 255);
19   analogWrite(ledPin,intensity);
20   //Envia una onda PWM especificado en la variable: intensity.
21   // Observamos la lectura analógica para comprobar el fondo de escala (0 ->
22   1024)
23
24   Serial.print("valor analógico leído=");
25 }
```

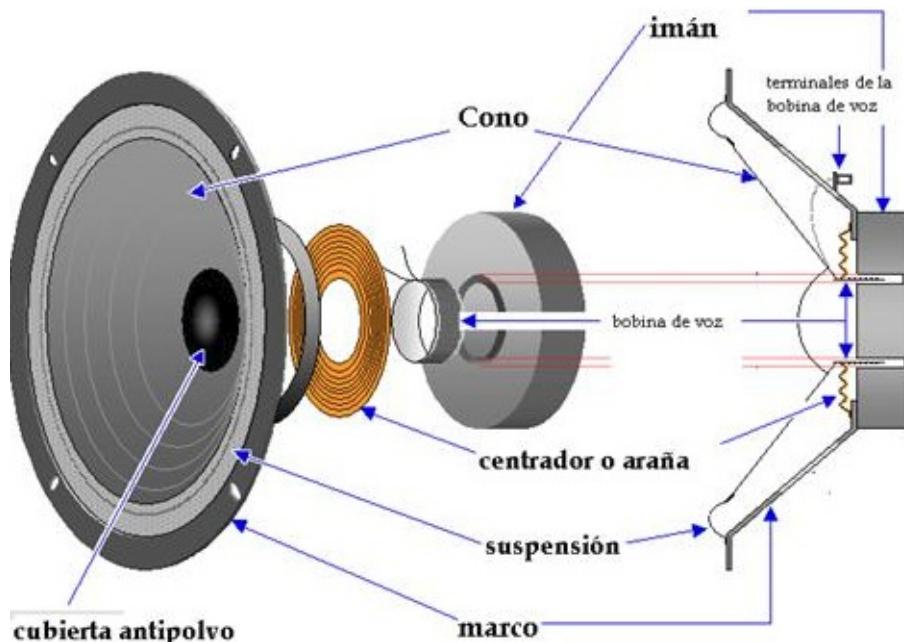
3.3 M

```
26   Serial.println(analogRead(potenPin));
      delay (1000);
}
```

Bobinas-altavoz

El altavoz es una simple bobina o electroimán que mueve una membrana, si la membrana se mueve repetidamente puede producir un sonido.

Este sonido es audible si está dentro de nuestro rango auditivo, suele ser entre 20Hz y 20kHz



¿Sabías que a medida que creces el margen de agudos (20kHz) baja?

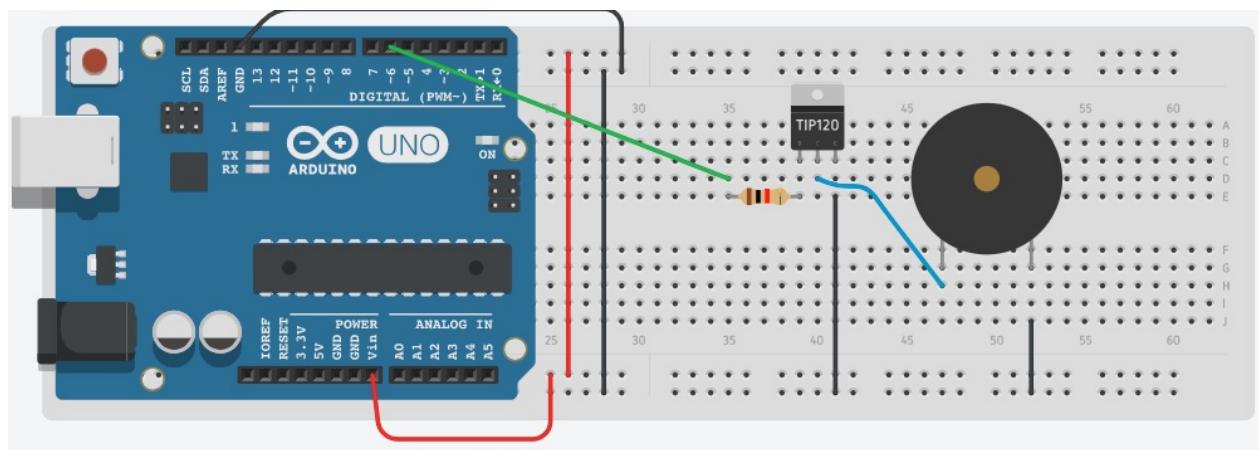


via GIPHY

Montaje : Pitido

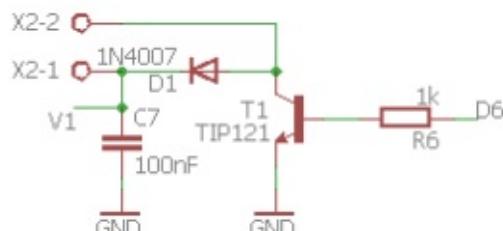
Vamos a incorporar un altavoz y realizar una sirena. Al ser una bobina, es conveniente utilizar una amplificación por medio de un transistor, por D6 enviaremos la señal cuadrada a la base del transistor.

SIN EDUBASICA



Con EDUBÁSICA

Conectaremos el altavoz en el terminal X2 y el interruptor V1 en ON para que esté alimentado



Continuamos ...

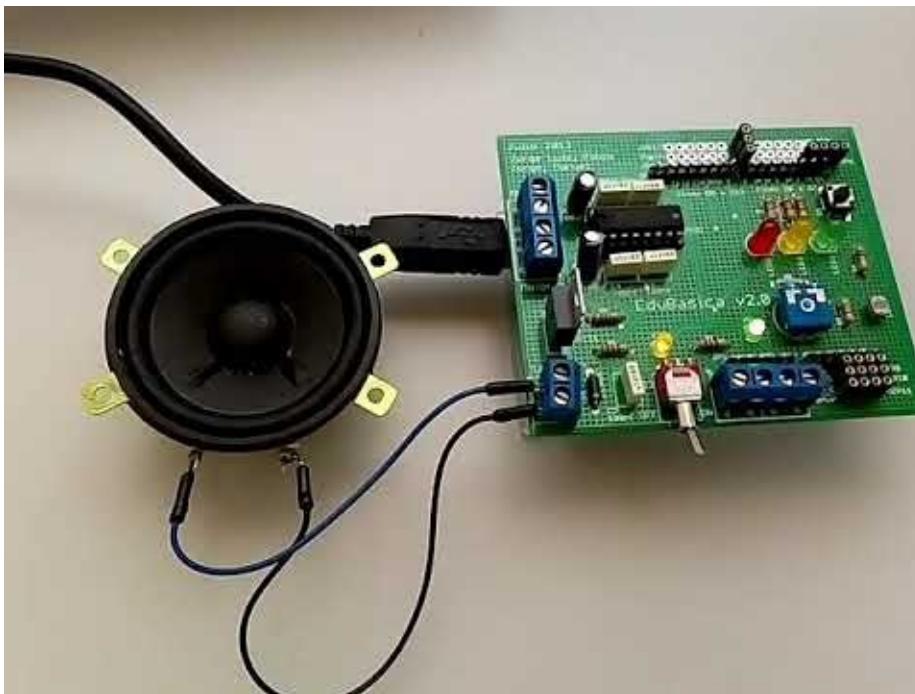
El código es muy sencillo, simplemente es una intermitencia por **D6** que en este caso se ha elegido **1mseg** ¿que pasaría si aumentáramos este valor?

```

1 void setup() {
2     // put your setup code here, to run once:
3     pinMode(6, OUTPUT);
4 }
5
6 void loop() {
7     // put your main code here, to run repeatedly:
8     digitalWrite(6, HIGH);    // Encendemos el pin6
9     delay(1);                // esperamos 1 msegundo
10    digitalWrite(6, LOW);    // Apagamos el pin6
11    delay(1);                // esperamos 1 msegundo
12

```

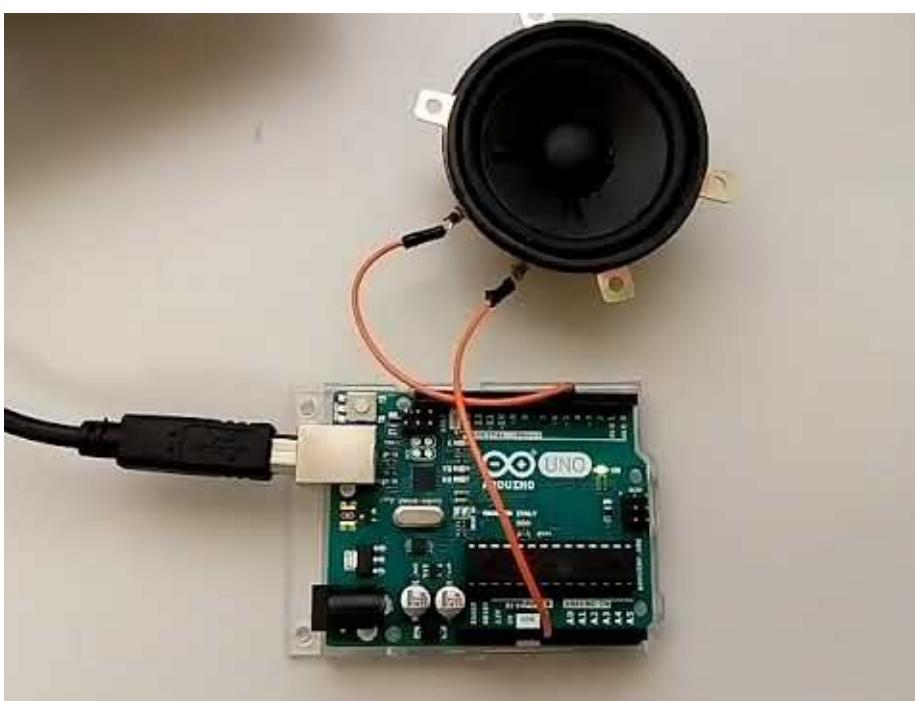
El resultado es :



[Video link](#)

SIN EDUBÁSICA Y SIN TRANSISTOR A LO BRUTO !

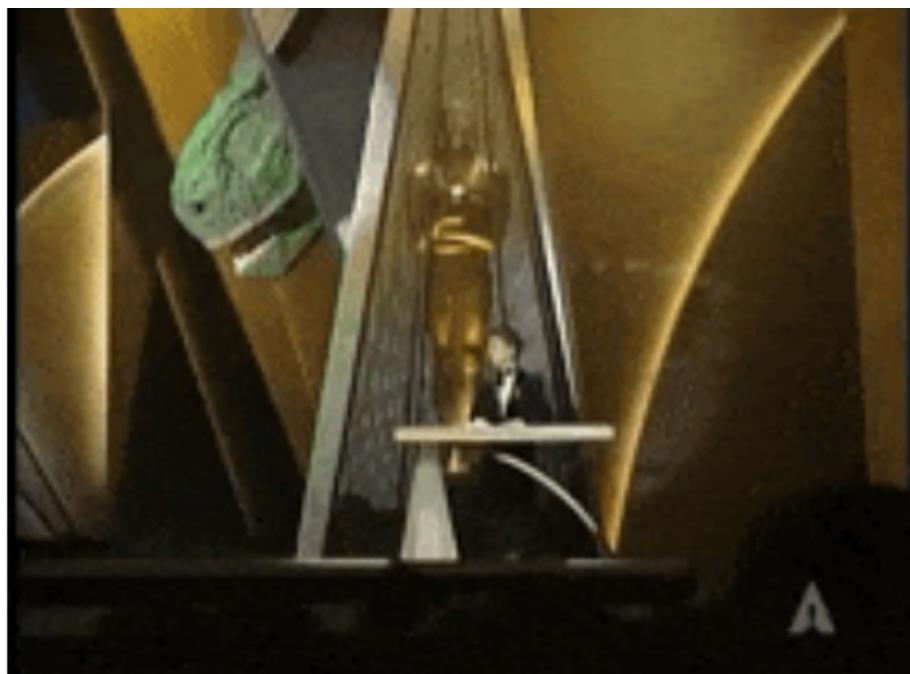
Bueno, vamos a conectarlo DIRECTAMENTE a D6 (el otro extremo a GND) no es muy conveniente pero a ver el resultado (con el mismo código):



[Video link](#)

¿Cuál suena más?

Premio entrada a dinópolis Teruel quien acierte..



via GIPHY

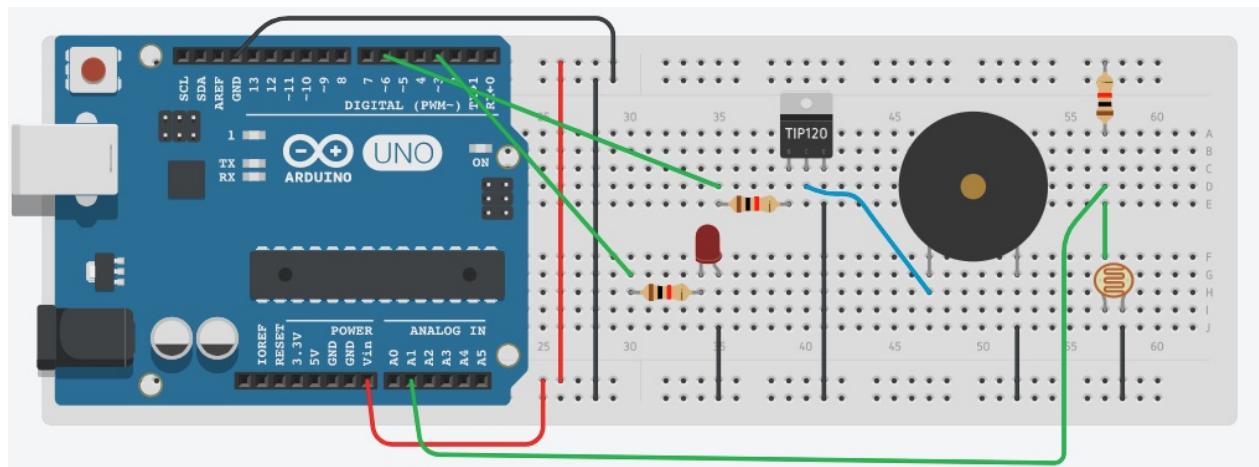
Montaje 15: Alarma

Teniendo en EDUBASICA los LEDs, el LDR que nos puede servir como sensor y el altavoz amplificado con un transistor, y nosotros que somos expertos programadores, NOS ESTÁ PIDIENDO A GRITOS hacer una alarma:

Enunciado: Cuando el LDR esté tapado, tiene que sonar un pitido intermitente de un segundo, con visualización también en los LEDs

SIN EDUBASICA

Pues hay que poner el LDR en A1, las luces (por simplicidad uno), el altavoz y el transistor con la conexión en la base por D5:



CON EDUBASICA

Se simplifica mucho la conexión sólo el altavoz tal y como está conectado en el montaje 14

Programa:

```

1 void setup() {
2     pinMode(6, OUTPUT);
3     pinMode(3, OUTPUT);
4     pinMode(4, OUTPUT);
5     pinMode(5, OUTPUT);
6     Serial.begin(9600);
7
8 }
9
10 void loop() {
11     Serial.print("valor analogico leido=");Serial.println(analogRead(1)); // así visualizamos los valores y determinamos cuando es oscuro o no
12     /////////////////////////////////
13     /////////////////////////////////
14     if (analogRead(1)<950 ){    // LDR NO TAPADO: NO SALTA LA ALARMA
15         digitalWrite(3, LOW);  digitalWrite(4, LOW); digitalWrite(5, LOW); // Se apagan los TRES LEDs
16     }
17 }
18

```

```
19     digitalWrite(6, LOW); //Se apaga el altavoz
20 }else{ // LDR TAPADO, SALTA LA ALARMA
21     digitalWrite(3, HIGH); digitalWrite(4, HIGH); digitalWrite(5, HIGH); /
22 / Se enciende los TRES LEDS:
23     ////////////////////////////////SUENA LA ALARMA durante un segundo 1ms x 1000 = 1 seg /////////////////////
24     for (int j=0;j<1000;j++){
25         digitalWrite(6, HIGH); // Encendemos el
26         delay(1); // esperamos 1 segundo
27         digitalWrite(6, LOW); // Apagamos el pin13
28         delay(1);
29     }
30     digitalWrite(3, LOW); digitalWrite(4, LOW); digitalWrite(5, LOW); // Se apagan los TRES LEDS
31     digitalWrite(6, LOW); //Se apaga el altavoz
32     delay(1000);
33 }
```

Resultado:



[Video link](#)

3 LCD

Las pantallas Liquid Cristal Display es la forma más sencilla de poner una interfáz de texto a nuestro Arduino.

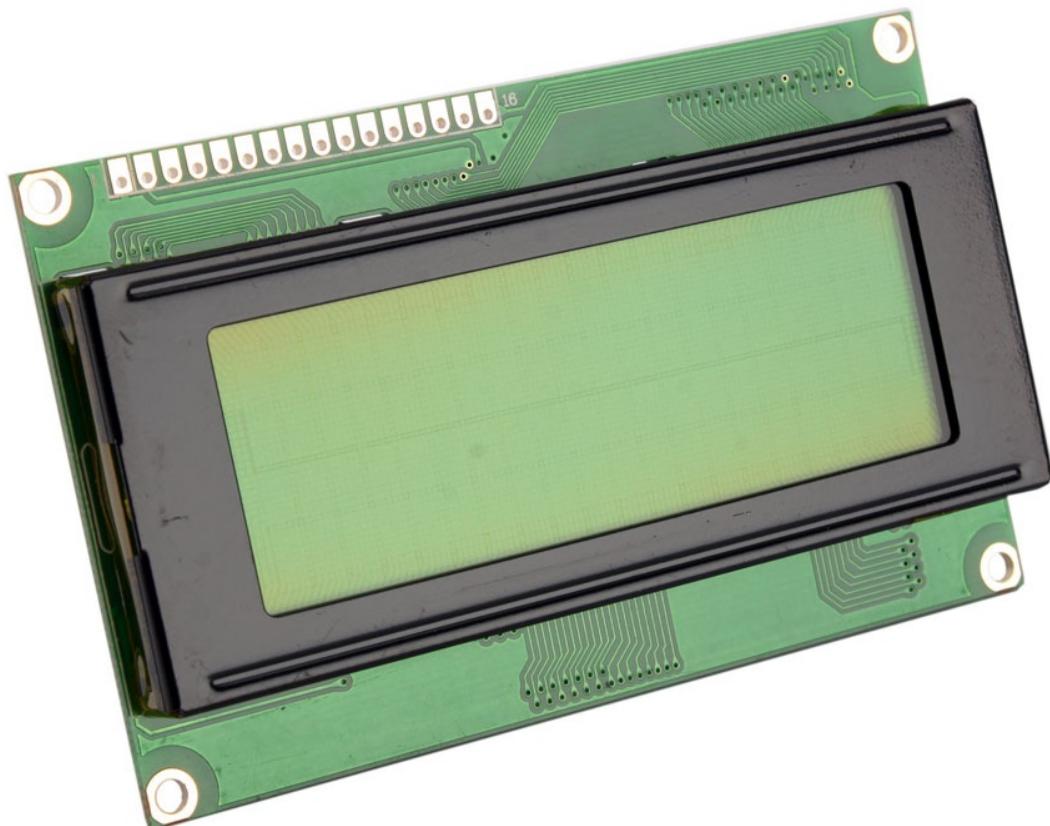


via GIPHY

La pantalla LCD

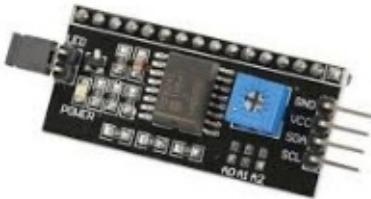
La pantalla LCD tiene un precio muy bajo, el más común es el Hitachi HD44780 monocromo con configuración 16 caracteres y 2 líneas (16x2) pero también se venden 20×02, 20×04 y 40×02.

Su conexión **DIRECTA** con el Arduino no es recomendable por la cantidad de cables que se necesitan y el código elaborado, pero tiene la ventaja de tener total libertad en creación de caracteres y control. Si quieres ver esta opción puedes ver [la página de Luis LLamas](#)



I2C

Como hemos dicho en [La pantalla LCD](#) no es recomendable su conexión directa con Arduino, para ello está este controlador que permite su conexión **utilizando sólo dos cables**. Los dos componentes pueden salir por menos de 5€.



Su conexión con el LCD tiene que ser tal y com indica la imagen, y soldando los terminales con cuidado:

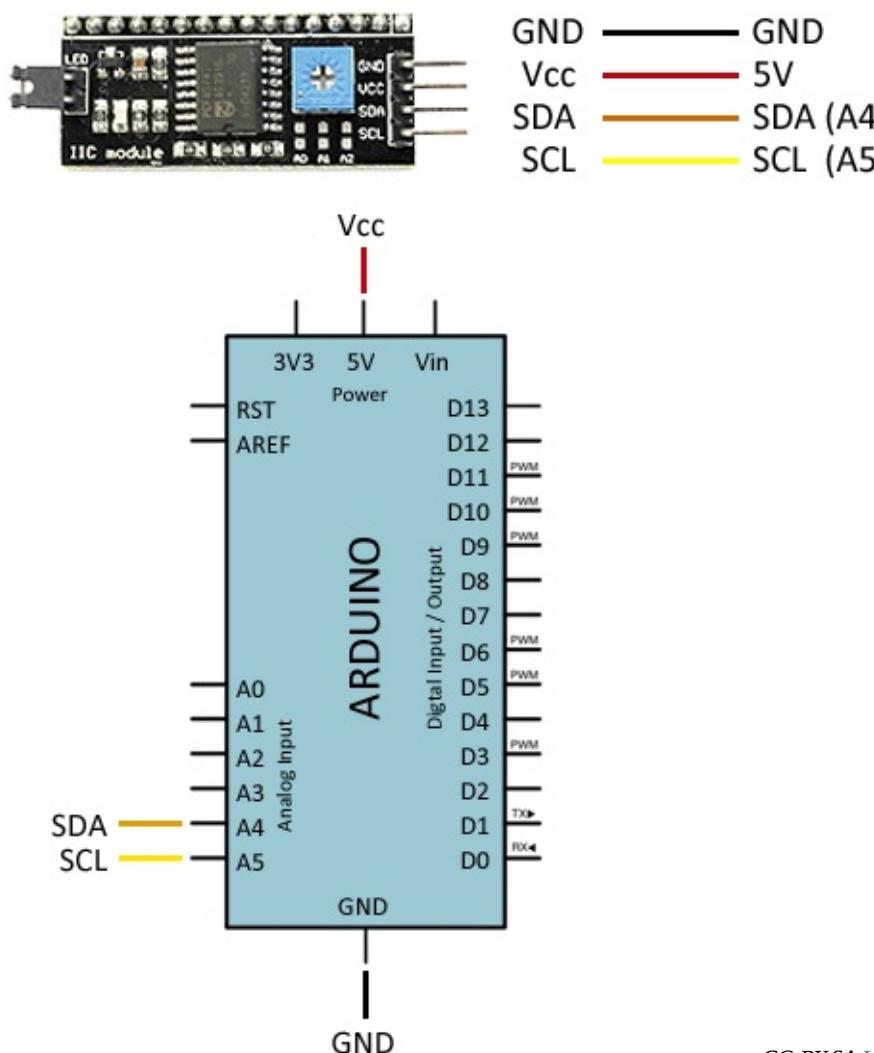


CC-BY-SA [Luis Llamas](#)

Conexión con el Arduino

La conexión tiene que ser en los pines A4 y A5 exclusivamente en el Arduino Uno pues son los dedicados para el protocolo serie I2C que veremos más adelante. Para otras placas ver [la página de Luis Llamas](#).

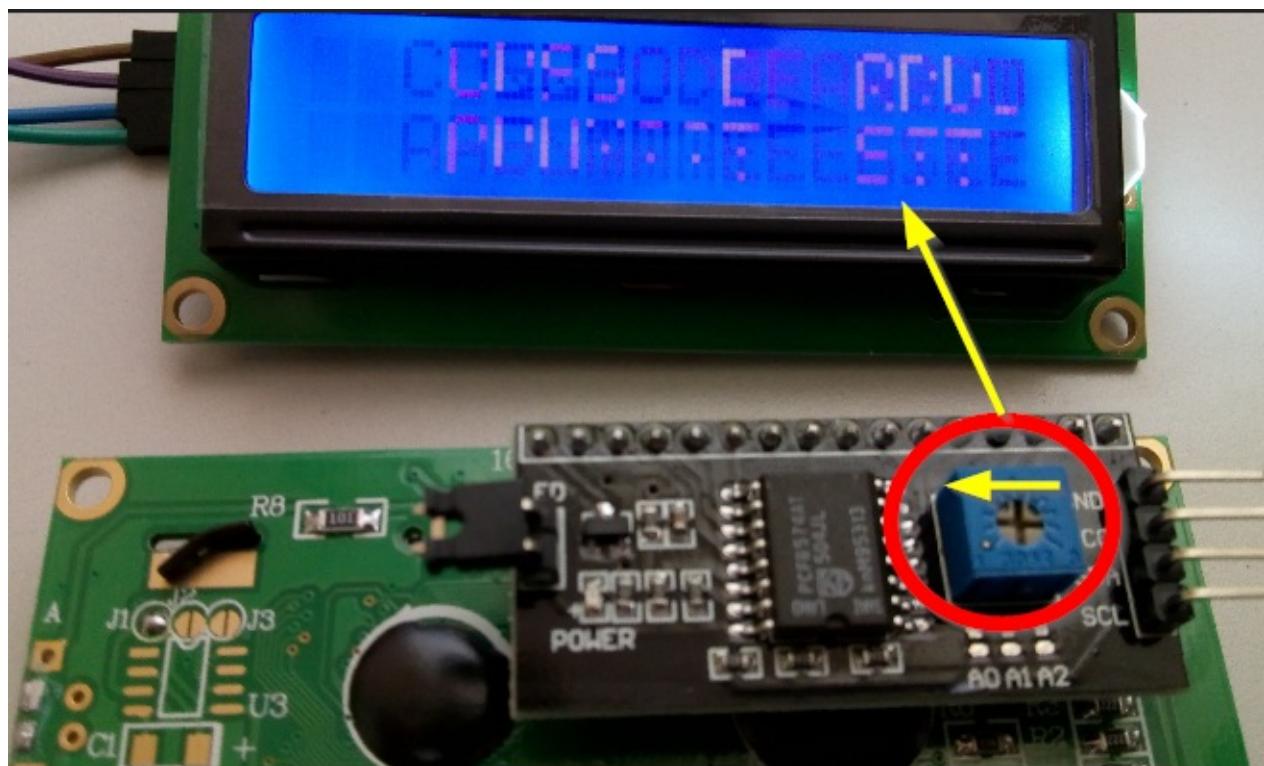
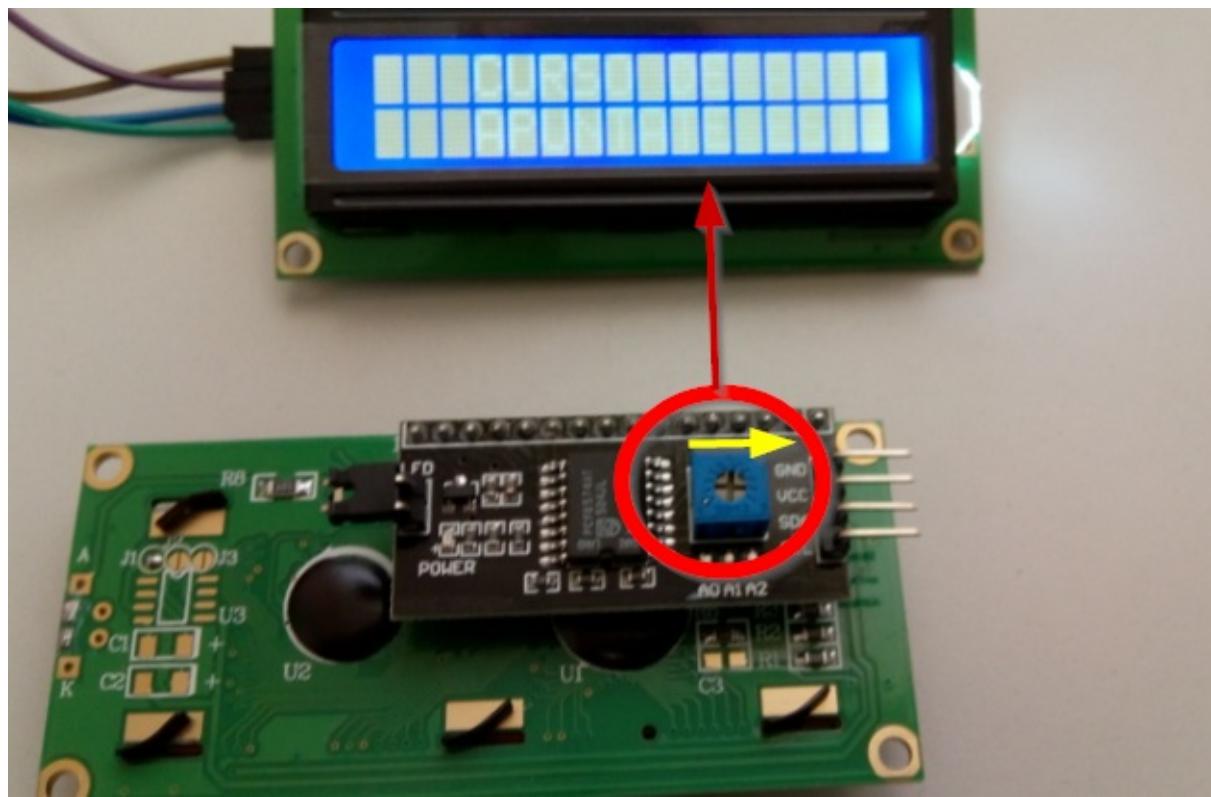
El esquema es muy sencillo:



CC-BY-SA Luis Llamas

Contraste

Tiene un potenciómetro azul para regular el contraste, bastante sensible por cierto (una poca variación hace que nuestro texto no se vea correctamente):



Escaneo

El bus I2C está vinculado en el Arduino Uno a los Pines A4 y A5 como hemos visto en [la anterior página](#), el protocolo serie I2C necesita **la librería Wire.h** que está de forma nativa en el programa [Arduino IDE](#), pero la **dirección de dispositivo no lo sabemos PARA ELLO HAY QUE EJECUTAR ESTE CÓDIGO:**

```

1 #include <Wire.h>
2
3
4 void setup()
5 {
6     Wire.begin();
7
8     Serial.begin(9600);
9     while (!Serial); // Leonardo: wait for serial monitor
10    Serial.println("\nI2C Scanner");
11 }
12
13
14 void loop()
15 {
16     byte error, address;
17     int nDevices;
18
19     Serial.println("Scanning...");
20
21     nDevices = 0;
22     for(address = 1; address < 127; address++)
23     {
24         // The i2c_scanner uses the return value of
25         // the Write.endTransmisstion to see if
26         // a device did acknowledge to the address.
27         Wire.beginTransmission(address);
28         error = Wire.endTransmission();
29
30         if (error == 0)
31         {
32             Serial.print("I2C device found at address 0x");
33             if (address<16)
34                 Serial.print("0");
35             Serial.print(address,HEX);
36             Serial.println(" !");
37
38             nDevices++;
39         }
40         else if (error==4)
41         {
42             Serial.print("Unknown error at address 0x");
43             if (address<16)
44                 Serial.print("0");
45

```

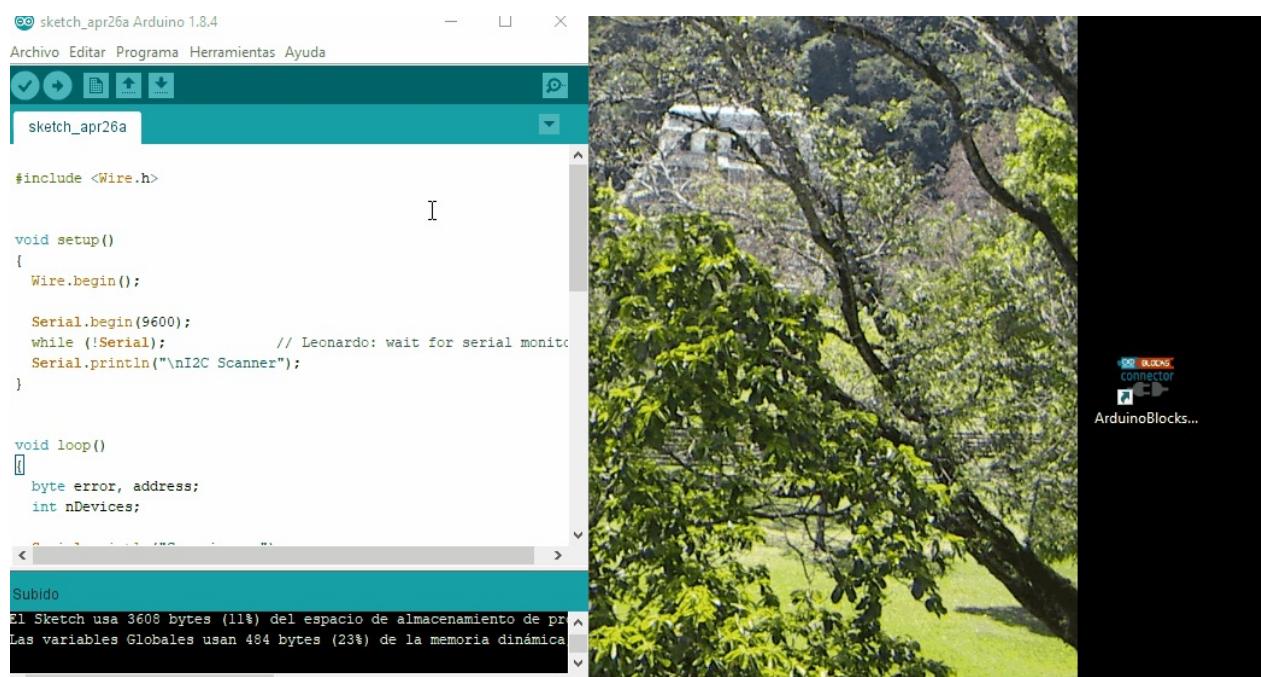
```

46     Serial.println(address, HEX);
47 }
48 }
49 if (nDevices == 0)
50     Serial.println("No I2C devices found\n");
51 else
52     Serial.println("done\n");
53
54 delay(5000);           // wait 5 seconds for next scan
}

```

Extraido de [Arduino.cc](#) o también de [Luis Llamas](#)

Nos tiene que salir lo siguiente:

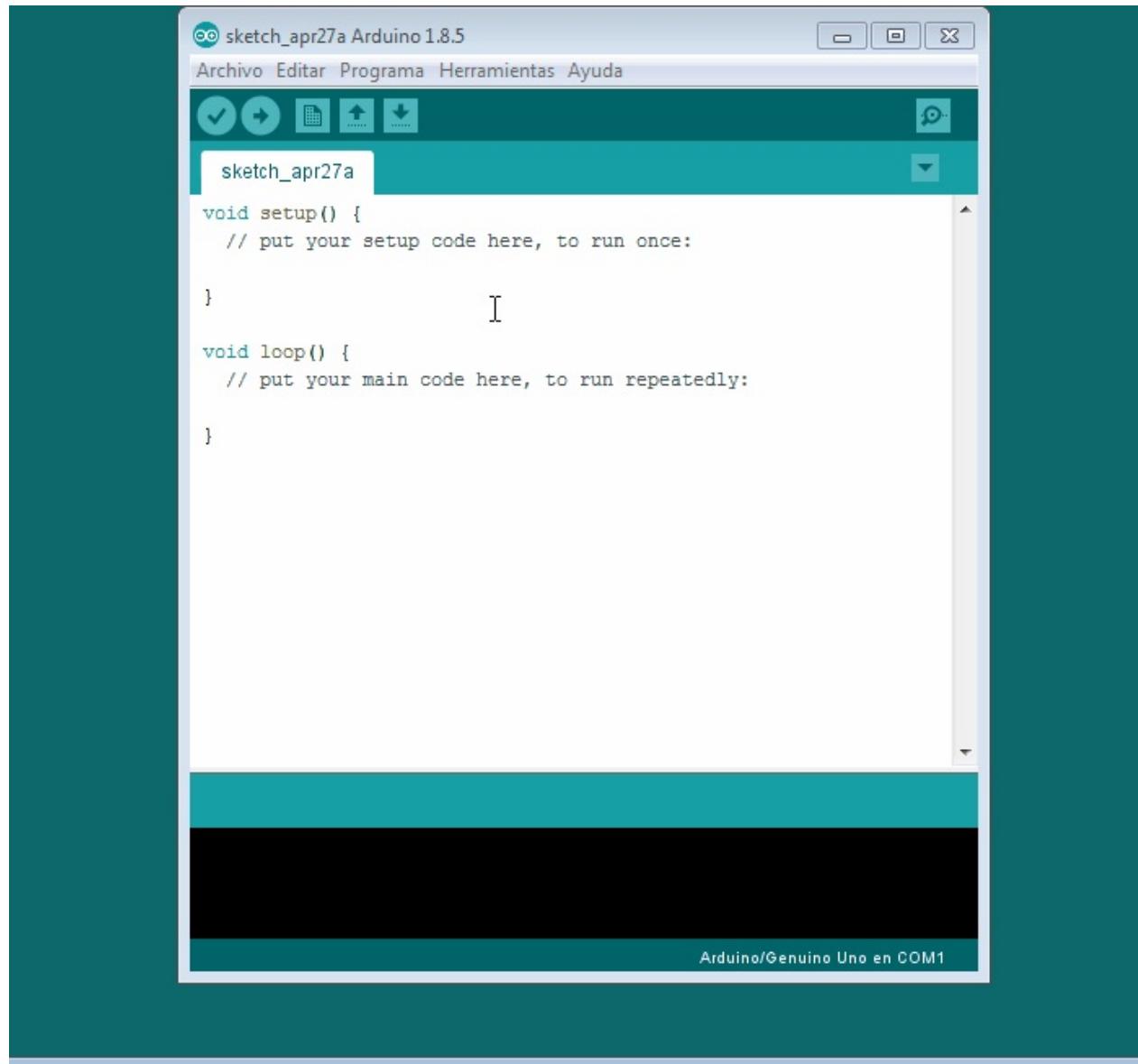


En mi caso como ves la dirección me ha salido **0x3F** pero puede ser cualquier otro, por ejemplo otro valor muy típico es **0x27**

Librería LiquidCrystal

Sólo nos falta incorporar la librería LiquidCrystal_I2C que te [lo puedes descargar aquí](#)

Una vez descargado, es un fichero comprimido .zip o .rar **no lo descomprimas** diréctamente desde el menú del entorno de programación lo incorporas de esta manera :



Principales funciones

LiquidCrystal_I2C(lcd_Addr, lcd_cols, lcd_rows) Crea una variable (informáticamente un objeto de la clase LiquidCrystal_I2C) para poder utilizar sus funciones, hay que indicar entre paréntesis la dirección, columnas y filas indicadas. Por ejemplo LiquidCrystal_I2C lcd(0x3F,16,2);

¿No sabes la dirección?. Eso es que te has saltado [esta lección](#) en mi caso es 0x3F.

Después de crear esa variable hay que inicializarlo con lcd.**init()**

lcd es el nombre de la variable, puedes poner el nombre que quieras

lcd.**clear()** Borra la pantalla y posiciona el cursor en la esquina superior izquierda (0,0).

lcd.setCursor(columna, fila) Posiciona el cursor del LCD en la posición indicada por columna y fila.

lcd.print("texto") Escribe el texto

lcd.scrollDisplayLeft() Se desplaza el contenido de la pantalla (texto y el cursor) un espacio hacia la izquierda.

lcd.scrollDisplayRight() Se desplaza el contenido de la pantalla (texto y el cursor) un espacio a la derecha.

lcd.backlight() Enciende la Luz del Fondo del LCD

lcd.noBacklight(); Apaga la Luz del Fondo del LCD

lcd.createChar (num, datos) Crea un carácter personalizado permite crear hasta 8. Para usar esta función [ver esta página](#).

Montaje Texto en LCD

Vamos a realizar un ejemplo para practicar:

- El encendido y apagado de la pantalla
- Visualización de texto
- Visualización de texto con desplazamiento

Practicaremos qué código tiene que estar en el código dentro de *setup* y qué código dentro de *loop*:



[Video link](#)

El código es sencillo, la primera parte que sólo lo hace una vez, tiene que estar en *setup* y en *loop* sólo la parte de que se desplaza continuamente.

OJO CAMBIATU DIRECCIÓN 0x3F si no es esa

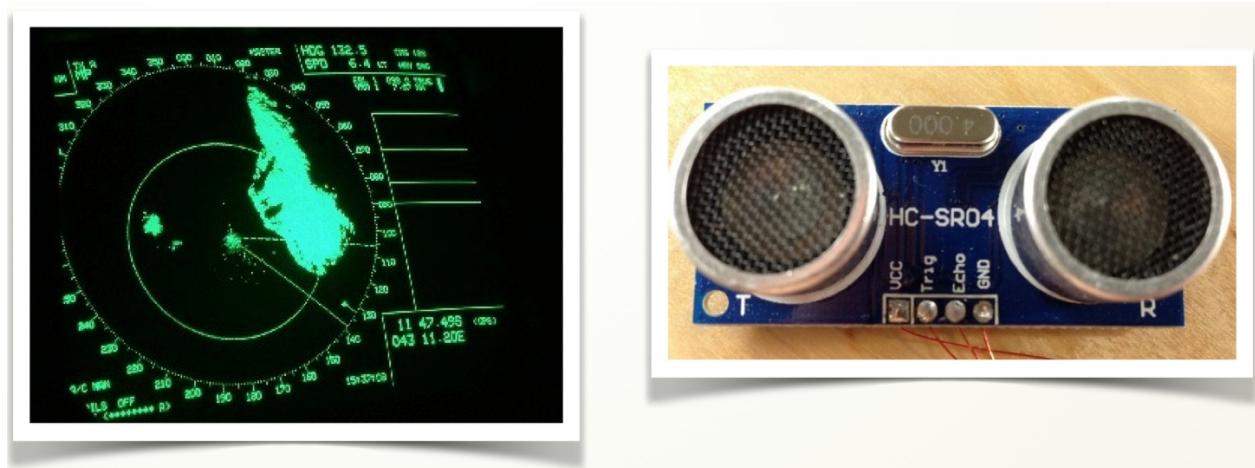
```

1 #include <Wire.h>
2 #include <LiquidCrystal_I2C.h>
3
4 LiquidCrystal_I2C lcd(0x3F, 16, 2); //Crear el objeto lcd dirección 0x3F y
5 16 columnas x 2 filas
6
7
8 void setup() {
9     // Inicializar el LCD
10    lcd.init();
11    for (int i=0; i <= 4; i++){
12        lcd.backlight(); //Encender la luz de fondo.
13        lcd.setCursor(1, 0); // Escribimos el Mensaje en el LCD en una posición
14        1, 0
15        lcd.print("ATENTOS EN:");
16        lcd.setCursor(1, 1); // Escribimos el Mensaje en el LCD en una posición
17

```

```
3.5.5  
18 1,1  
19     lcd.print("www.catedu.es");  
20     delay (400);  
21     lcd.noBacklight(); //apaga la luz de fondo  
22     delay (200);  
23 }  
24 lcd.backlight();  
25 lcd.setCursor(1, 0);  
26 lcd.print("CURSO DE ARDUINO ACTUALIZADO");  
27 lcd.setCursor(1, 1);  
28 lcd.print("APUNTATE ESTE VERANO EN AULARAGON");  
29 }  
30  
31 void loop() {  
    //desplazamos una posición a la izquierda  
    lcd.scrollDisplayLeft();  
    delay(400);  
}
```

Sensor de ultrasonidos



Conocimiento previo

Programación básica de Arduino, escritura en puerto serie.

Objetivos

Manejar este tipo de sensores que son muy comunes en las aplicaciones de robótica para medir distancias. Para ello aprenderás a:

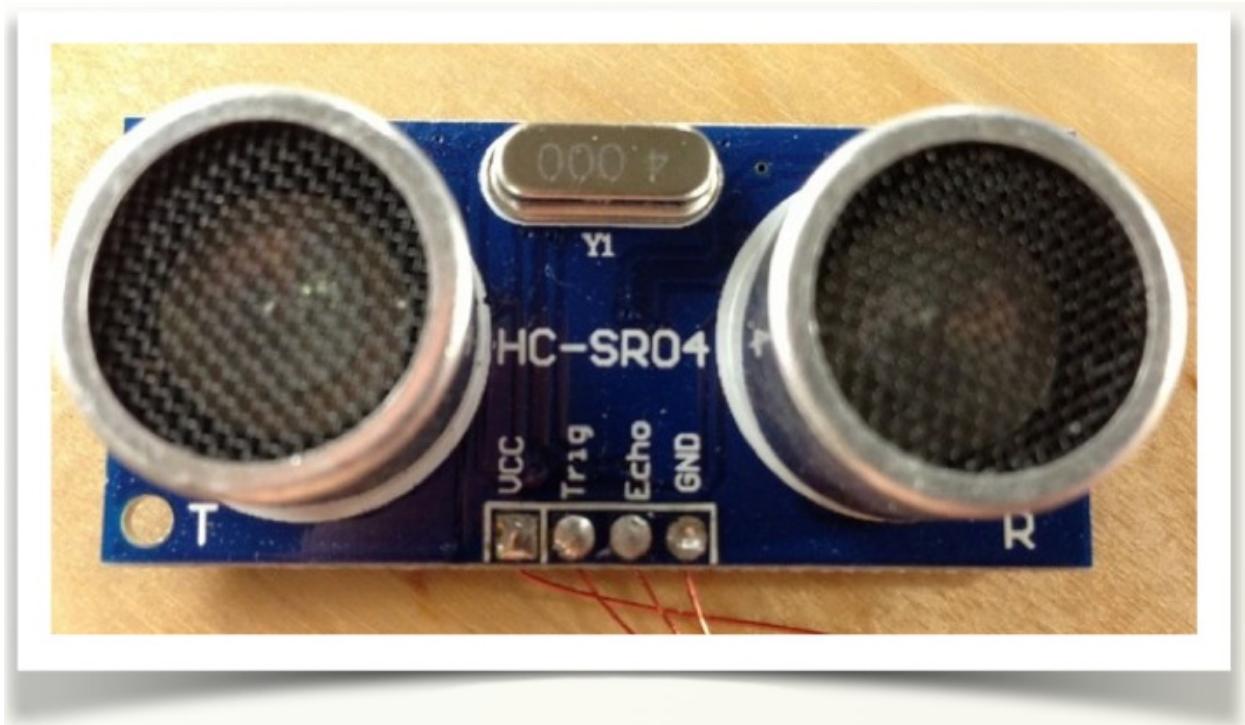
- Realizar las conexiones necesarias sobre el sensor ultrasonidos HC-SR04.
- Conocer el funcionamiento de un radar.
- Cómo convertir el tiempo de rebote de un sonido en distancia.

La lista de materiales necesaria para el montaje es:

- Arduino UNO.
- Sensor de ultrasonidos HC-SR04.
- Protoboard.
- Cableado.

Montaje 8: Medición de la distancia

Este tipo de sensores también nos permite conocer la distancia a un objeto. Es más preciso que el de infrarrojos visto en la sección anterior y su rango de funcionamiento también es mayor. Funciona desde los 2cm hasta los 3 metros.



El sensor tiene 2 partes como puedes ver en la figura. Una se encarga de enviar un sonido (a una frecuencia alta que no podemos escuchar), y la otra parte detecta cuando ese sonido vuelve.

Este sensor es muy útil en robots móviles para diversas acciones como no chocar o mantenerse a cierta distancia de una pared.

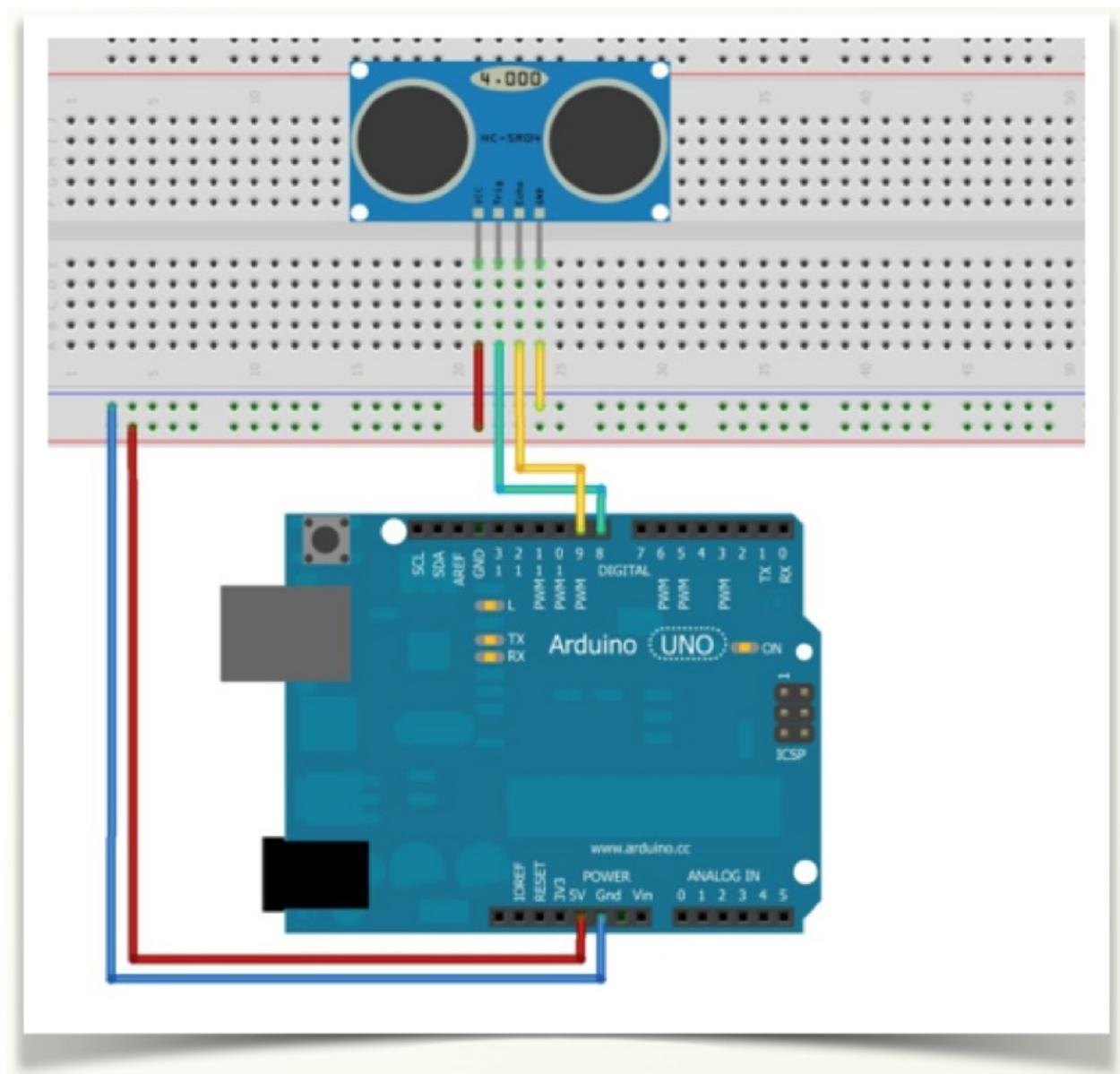
Podemos usar un sensor de ultrasonidos para obtener la distancia a un objeto. Este sensor se basa en el envío de una señal acústica y la recepción del eco de dicha señal. Lo que haremos después, al igual que hace un radar, un aparato de ecografías o un murciélagos es calcular la distancia en función del tiempo que ha tardado el rebotar el sonido y la velocidad del sonido. Podemos encontrar las especificaciones en la página del fabricante. Uno de los modelos más comunes es el HC-SR04:

El sensor que usamos en estos ejemplos tiene 4 pines que corresponden (de izquierda a derecha):

- GND , Vcc (a +5V)
- Trig: es el que emite el ultrasonido
- Echo: Es el que recibe el rebote

(Algunos modelos solo tienen 3 pines -HCSR05- indicándonos por el tercer pin ya directamente un valor proporcional con la distancia.)

No aconsejamos usar la Shield de Edubasica, sino conectar directamente, en este caso no nos supone un ahorro de cableado, no como en los motores:



El programa es:

```

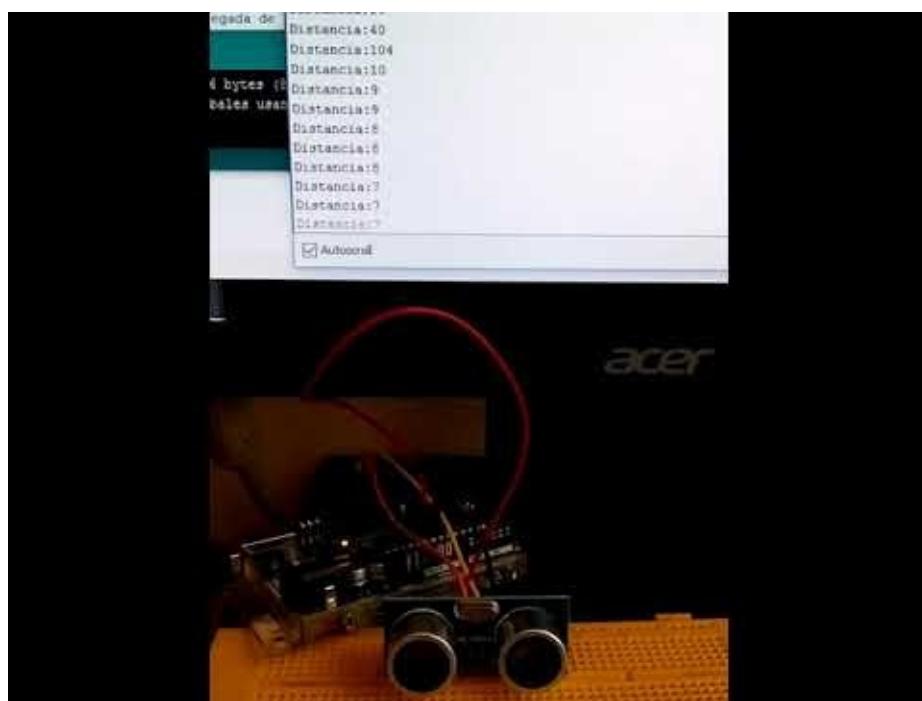
1 int trigPin = 8;
2 int echoPin = 9;
3 long duration; //tiempo de ida/vuelta
4 int cm=0; //Para almacenar el valor obtenido en cm valor=0
5
6 void setup() {
7
8     Serial.begin(9600);
9     pinMode(trigPin, OUTPUT);
10    pinMode(echoPin, INPUT);
11
12 }
13
14 void loop()
15 {
16
17
18 }
```

```

19 //ponemos al trigger a low para activarlo después
20
21   digitalWrite(trigPin, LOW);
22   delayMicroseconds(2);
23
24 //Activar el módulo con un pulso de 10microsec.
25
26   digitalWrite(trigPin, HIGH);
27   delayMicroseconds(10);
28   digitalWrite(trigPin, LOW);
29
30 //Esperamos la llegada de un pulso HIGH
31
32 duration = pulseIn(echoPin, HIGH);
33
34 //tiempo de ida y vuelta, dividimos por 2
35
36 duration=duration/2;
37
38 //La velocidad del sonido es de 340 m/s
39 //es decir, 29 microsegundos por centímetro
40
41 cm = duration/ 29;
42 Serial.print("Distancia:");
43 Serial.println(cm);
44 delay(100);
45
}

```

El resultado :



[Video link](#)

Medición con LCD

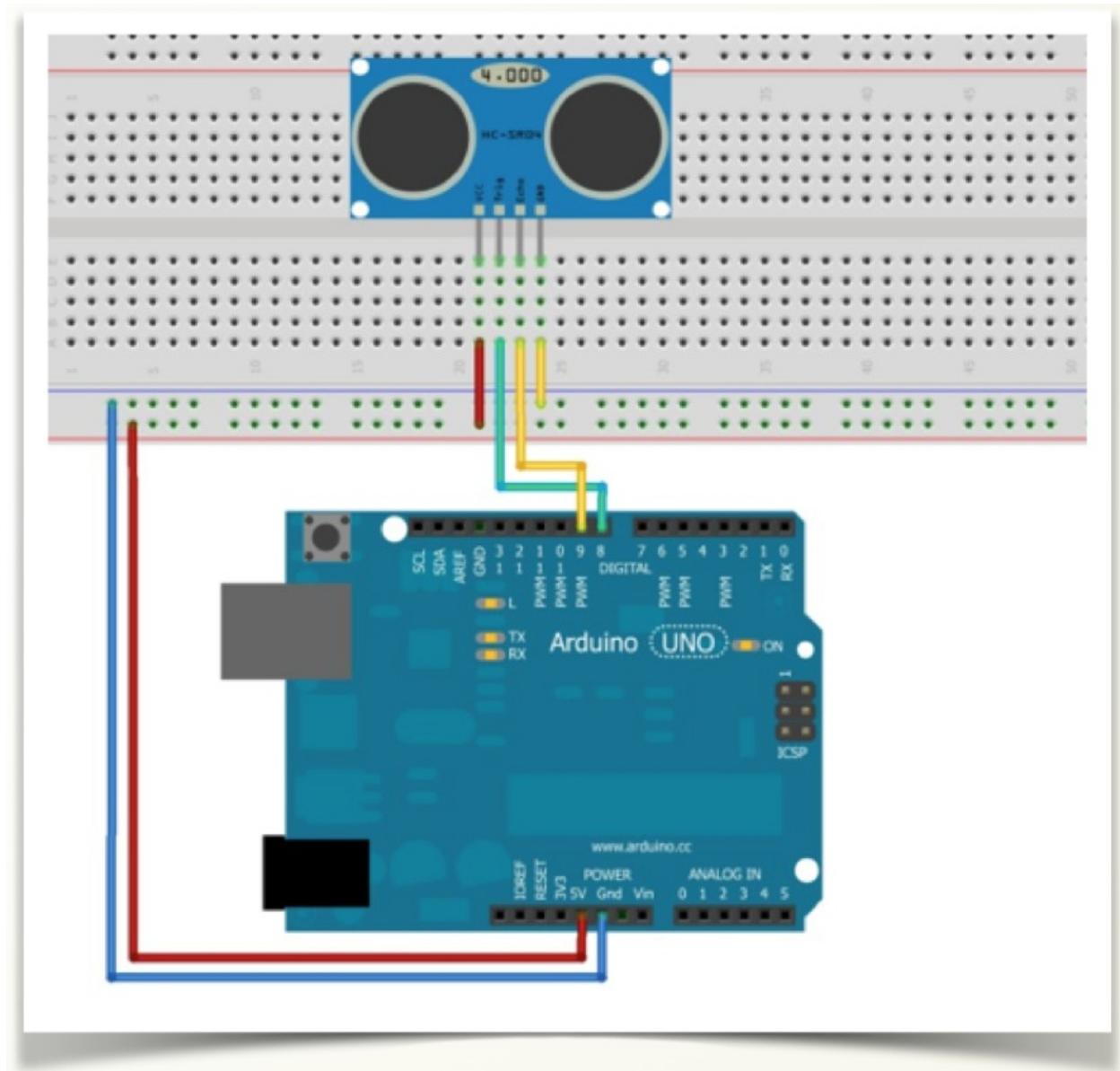
Vamos a repetir el anterior programa pero que lo visualice el LCD



[Video link](#)

Conexión:

Conectar el sensor de Ultrasonidos



Y el LCD



GND	—	GND
Vcc	—	5V
SDA	—	SDA (A4)
SCL	—	SCL (A5)

Código

```

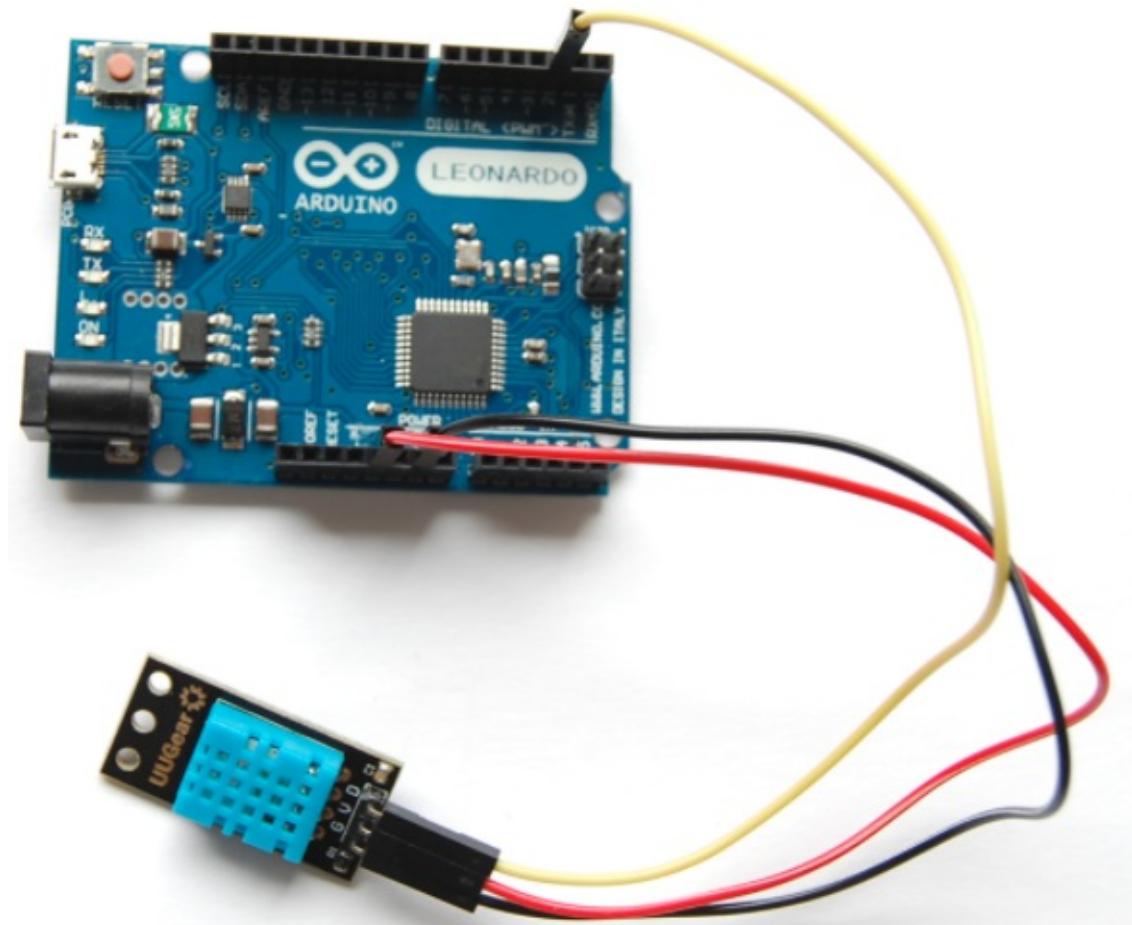
1 #include <Wire.h>
2
3 #include <LiquidCrystal_I2C.h>
4
5 LiquidCrystal_I2C lcd(0x3F, 16, 2); // //Crear el objeto lcd dirección 0x3F y
6 16 columnas x 2 filas
7
8 int trigPin = 8;
9 int echoPin = 9;

```

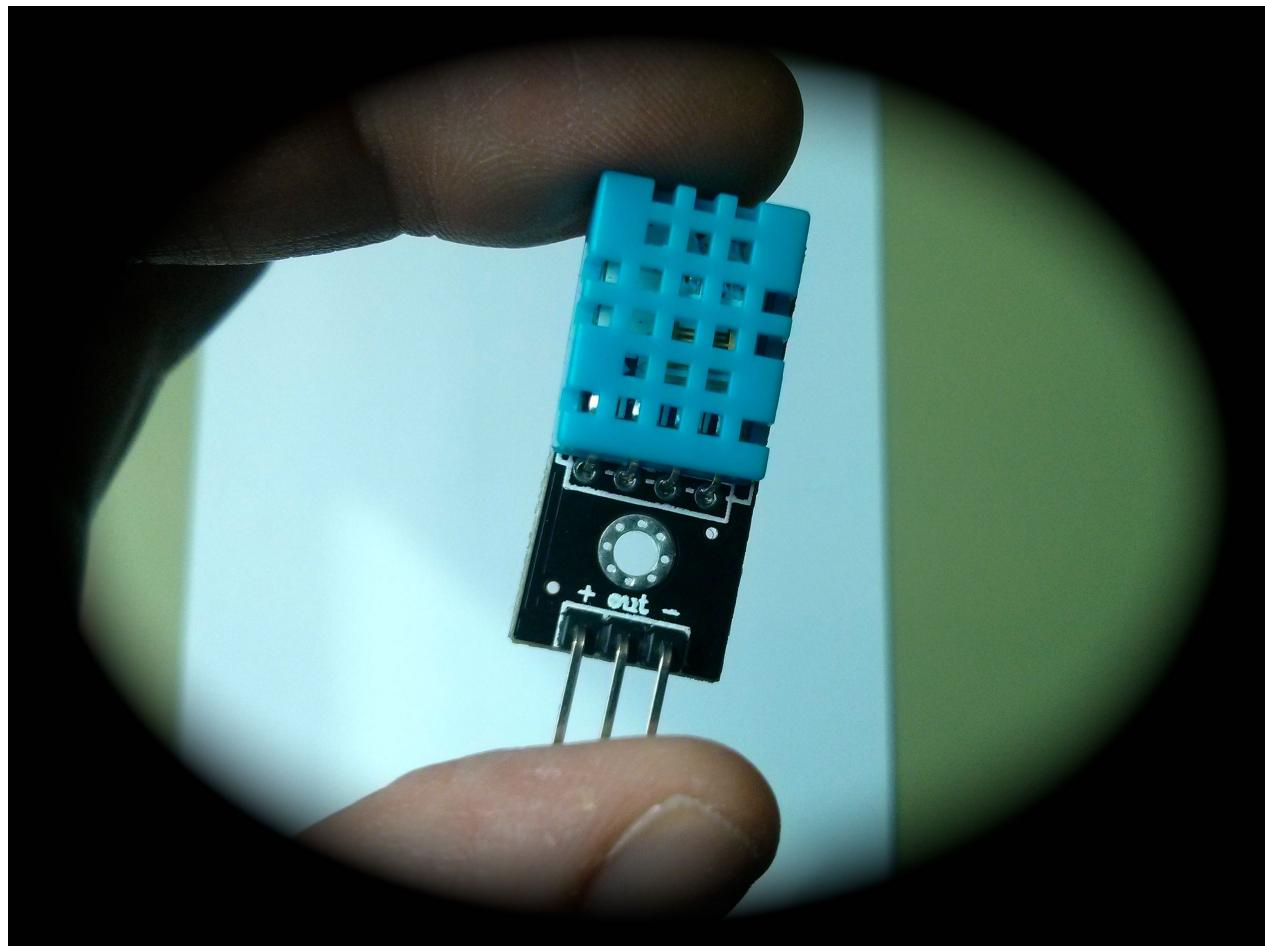
```
10 float duration; //tiempo de ida/vuelta
11 float cm=0; //Para almacenar el valor obtenido en cm valor=0
12
13 void setup() {
14
15     pinMode(trigPin, OUTPUT);
16     pinMode(echoPin, INPUT);
17
18     // Inicializar el LCD
19     lcd.init();
20     lcd.backlight(); //Encender la luz de fondo.
21
22 }
23
24 void loop() {
25     //ponemos al trigger a low para activarlo después
26     digitalWrite(trigPin, LOW);
27     delayMicroseconds(2);
28     //Activar el módulo con un pulso de 10microsec.
29     digitalWrite(trigPin, HIGH);
30     delayMicroseconds(10);
31     digitalWrite(trigPin, LOW);
32     //Esperamos la llegada de un pulso HIGH
33     duration = pulseIn(echoPin, HIGH);
34     //tiempo de ida y vuelta, dividimos por 2
35     duration=duration/2;
36     //La velocidad del sonido es de 340 m/s
37     //es decir, 29 microsegundos por centímetro
38     cm = duration/ 29;
39     lcd.setCursor(0, 0); // Escribimos el Mensaje en el LCD en una posición 1
40 ,0
41     lcd.print("Distancia:");
42     lcd.setCursor(0, 1); // Escribimos el Mensaje en el LCD en una posición 1
43 ,0
44     lcd.print(cm);
45     lcd.print(" cm ");
46     delay(100);
47 }
```

Control de temperatura y humedad

Vamos a utilizar dos sensores para medir estas variables: # **DHT12** y su hermano pequeño **DHT11**. Aunque lo vamos a contar brevemente, te recomendamos [esta página](#) para saber más de estos dos sensores.



Medida de la temperatura y la humedad ambiente.



Realizar un montaje que mida temperatura y humedad mediante el sensor DHT11.

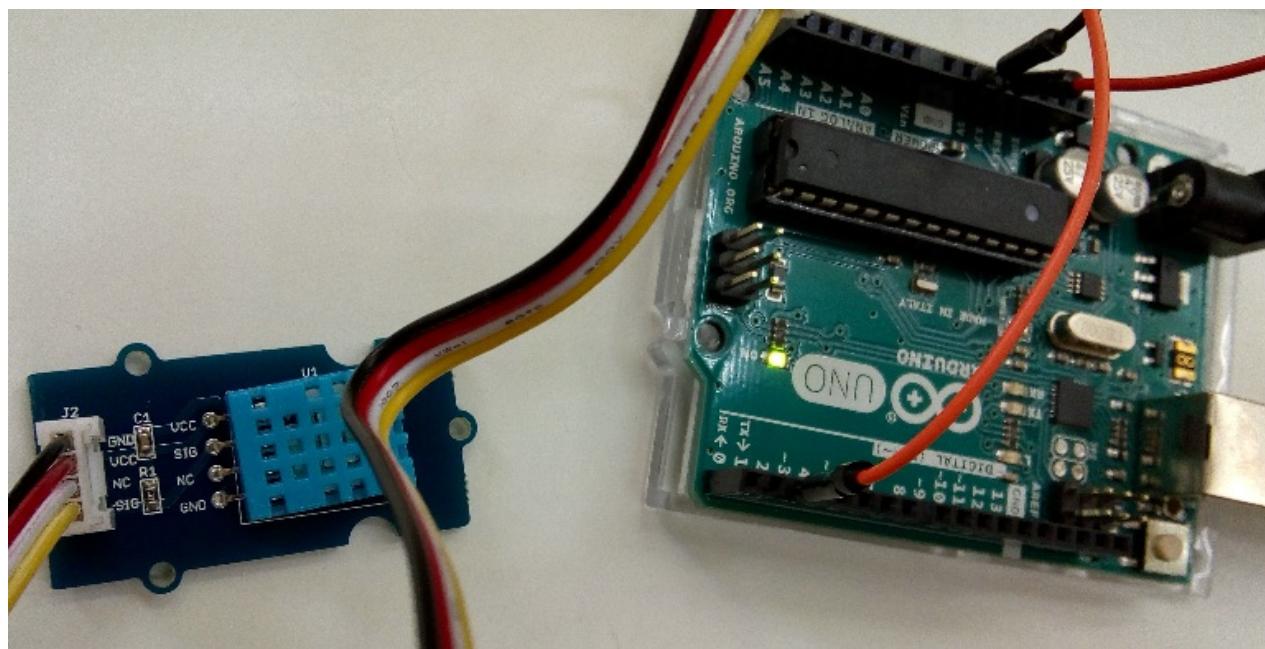
Para ello tendrás que realizar las conexiones necesarias para poder obtener los datos desde Arduino. Se trata de un sensor digital que utiliza 3 pines: Alimentación +5V, tierra/GND (-) y pin de datos (out) por donde se envían los datos de humedad y temperatura. A veces el sensor viene sobre 4 pines, en este caso uno de ellos no se conecta. Como las medidas de humedad y temperatura van por un solo pin, la información se transmite como un tren de pulsos en serie, por lo tanto, necesitamos un programa que "extraiga" esos dos datos de forma diferenciada. Para ello vamos a usar una librería referenciada por [DHT11.h](#) que se explica en [la siguiente página](#). A través del monitor serie del IDE de Arduino podremos ver las medidas obtenidas.

ESQUEMA DHT11 :

Es un modelo "conectar y listo" que ya vienen con los cables preparados, pero si te fijas son en este orden : GND - 5V -NC - D2 donde NC significa NO CONECTADO y D2 son los datos



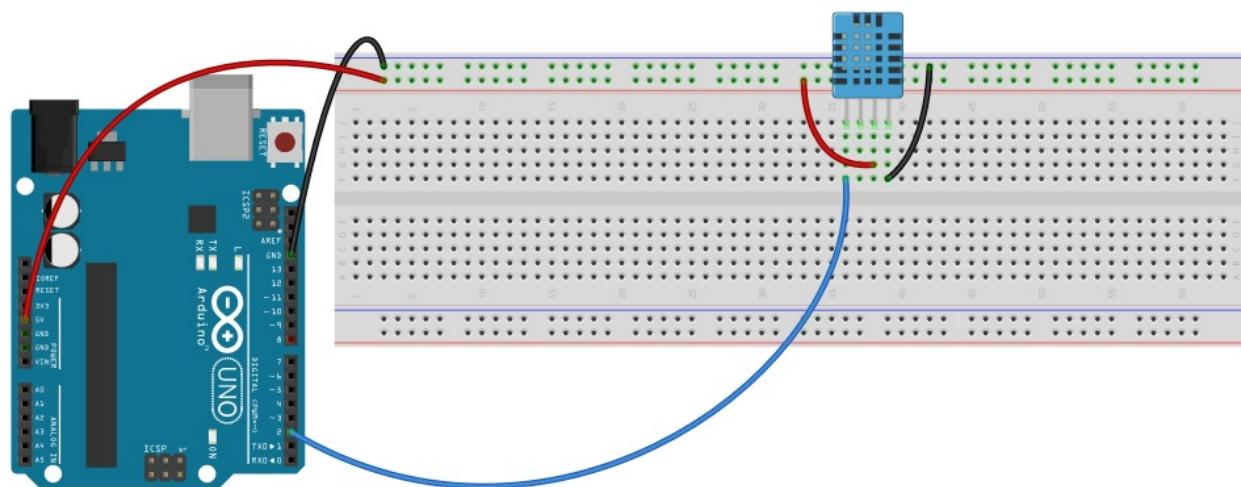
Lo conectamos en el Arduino sin necesidad de placa Protoboard:



utilizando cables macho-macho

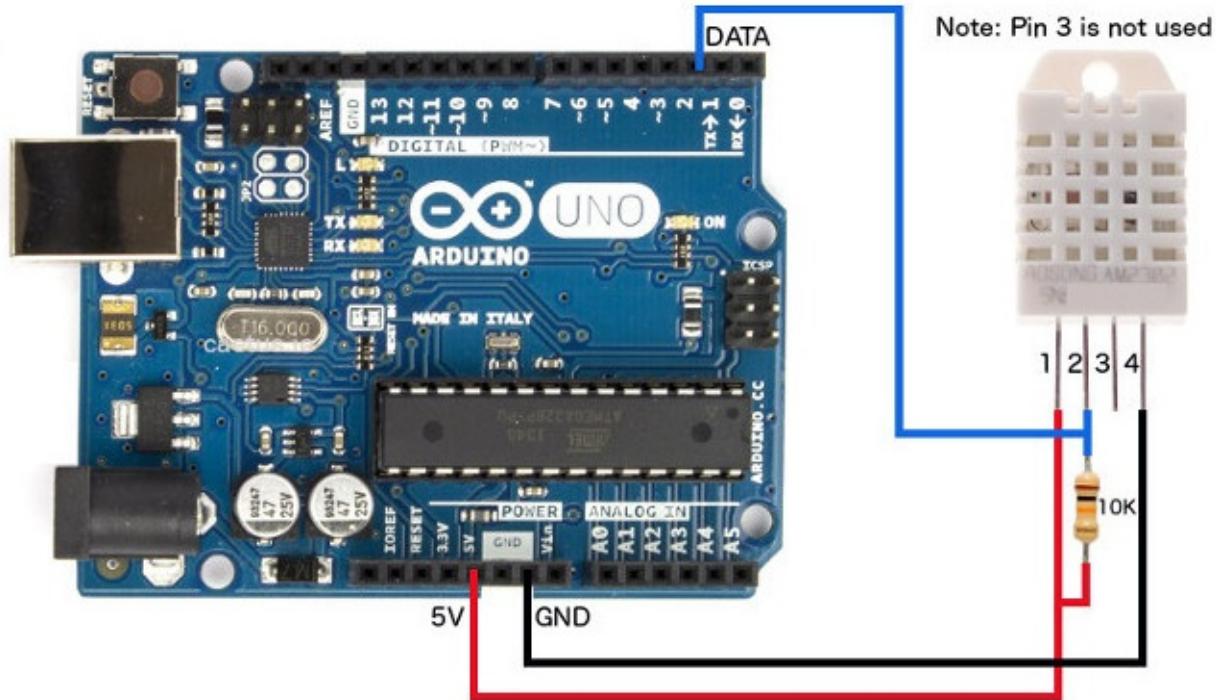


Esquema de conexión:

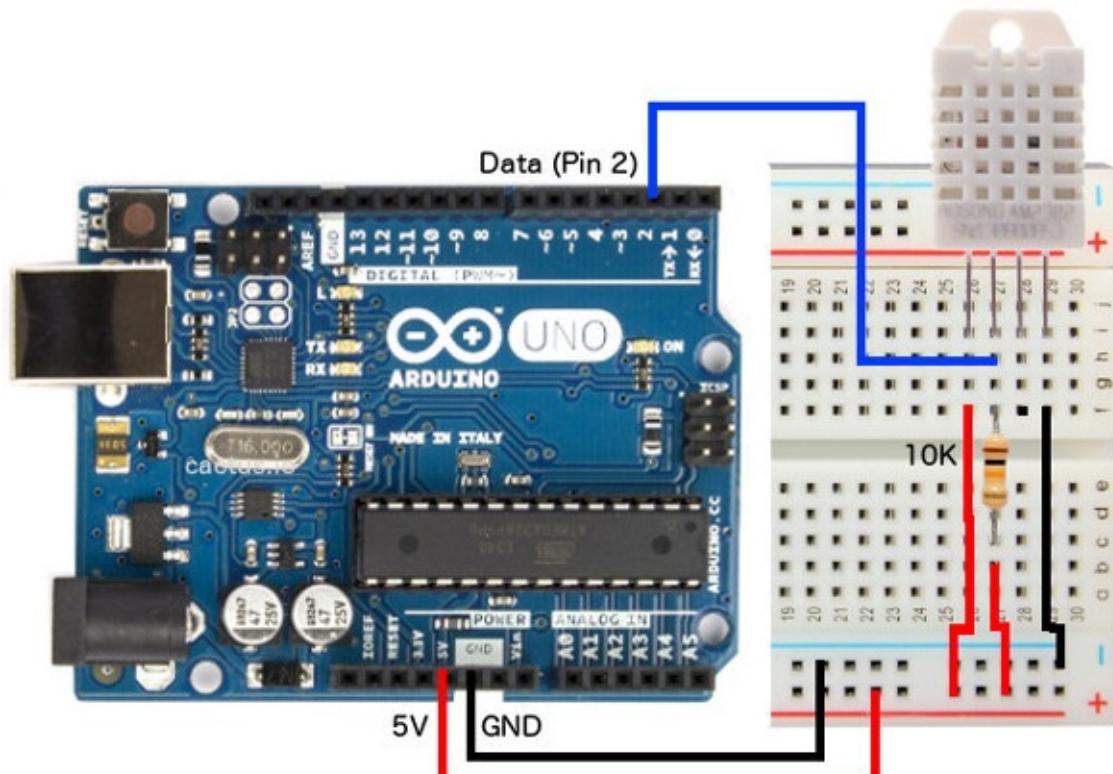


Esquema DHT12

Es un sensor que necesita 3.3V pero si trabajas con 5V necesitas hacer este puente con una resistencia de más o menos 10K



O con la placa Protoboard



Fuente Cactus.io CC BY-NC-SA 3.0

Librería DHT11.h

Para poder utilizar este dispositivo, necesitas esta librería, lo primero que tienes que hacerlo es descargarla, puedes hacerlo desde este enlace de [Drive](#), o desde este de [CATEDU](#) (zip - 4,29 KB), o simplemente buscando DHT11 library en Internet.

Una vez descargado, tienes que incorporarlo en tu librería, aquí tienes cómo hacerlo

¿Cómo incluir una librería en IDE Arduino?

Previamente descargado la librería en formato ZIP



Librería DHT12.h

Para poder utilizar este dispositivo, necesitas esta librería, lo primero que tienes que hacerlo es descargarla, puedes hacerlo desde [este enlace](#), o desde este de [CATEDU](#) (zip - 4,49 KB) (zip - 4,29 KB), o simplemente buscando DHT12 library en Internet.

Una vez descargado, tienes que incorporarlo en tu librería, aquí tienes cómo hacerlo

¿Cómo incluir una librería en IDE Arduino?

Previamente descargado la librería en formato ZIP



Montaje 9 Medición T y H por puerto serie

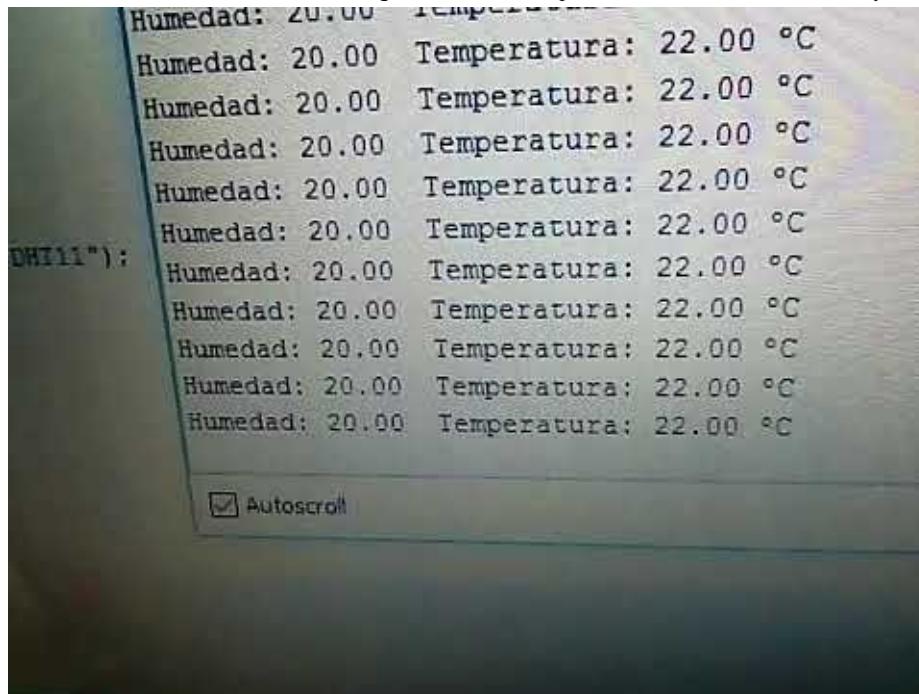
Programa con DHT11:

Te proponemos este programa:

```
1 #include "DHT.h"
2 #define DHTPIN 2
3 #define DHTTYPE DHT11 // DHT 11
4 DHT dht(DHTPIN, DHTTYPE);
5
6 void setup() {
7     Serial.begin(9600);
8     dht.begin();
9 }
10
11 void loop() {
12     delay(2000);
13     float h = dht.readHumidity();
14     float t = dht.readTemperature();
15     if (isnan(h) || isnan(t)) {
16         Serial.println("Fallo al leer el sensor DHT11");
17         return;
18     }
19     Serial.print("Humedad: ");
20     Serial.print(h);
21     Serial.print(" \t");
22     Serial.print("Temperatura: ");
23     Serial.print(t);
24     Serial.println(" °C ");
25 }
```

El resultado se puede ver en este vídeo, simplemente soplando nuestro vaho pasamos de 20% de humedad y 22°C a 93% y 24°C.

Advertencia: Si lo hacéis con niños, enseguida se les ocurre ponerlo en el sobaco, menos mal que solo son 5V;)



[Video link](#)

Programa con DHT12

La librería de este sensor es más potente y nos puede decir la sensación térmica:

```

1 // Example sketch for DHT22 humidity - temperature sensor
2 // Written by cactus.io, with thanks to Adafruit for bits of their library.
3 // public domain
4
5 #include "cactus_io_DHT22.h"
6
7 #define DHT22_PIN 2      // what pin on the arduino is the DHT22 data line connected to
8
9 // For details on how to hookup the DHT22 sensor to the Arduino then check out this page
10 // http://cactus.io/hookups/sensors/temperature-humidity/dht22 hookup-arduino-to-dht22-temp-humidity-sensor
11
12 // Initialize DHT sensor for normal 16mhz Arduino.
13 DHT22 dht(DHT22_PIN);
14
15 // Note: If you are using a board with a faster processor than 16MHz then you need
16 // to declare an instance of the DHT22 using
17 // DHT22 dht(DHT22_DATA_PIN, 30);
18
19 // The additional parameter, in this case here is 30 is used to increase the number of
20 // cycles transitioning between bits on the data and clock lines. For the
21 // Arduino boards that run at 84MHz the value of 30 should be about right.
22
23
24 void setup() {
25
26
27

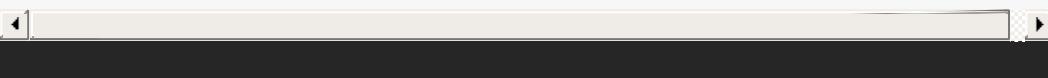
```

```

28 Serial.begin(9600);
29 Serial.println("DHT22 Humidity - Temperature Sensor");
30 Serial.println("RH\t\tTemp (C)\t\tTemp (F)\t\tHeat Index (C)\t\tHeat Index (F)");
31 );
32
33 dht.begin();
34 }
35
36 void loop() {
37 // Reading temperature or humidity takes about 250 milliseconds!
38 // Sensor readings may also be up to 2 seconds 'old' (its a very slow sen
39 sor)
40 dht.readHumidity();
41 dht.readTemperature();
42
43 // Check if any reads failed and exit early (to try again).
44 if (isnan(dht.humidity) || isnan(dht.temperature_C)) {
45   Serial.println("DHT sensor read failure!");
46   return;
47 }
48
49 Serial.print(dht.humidity); Serial.print(" %\t\t");
Serial.print(dht.temperature_C); Serial.print(" *C\t");
Serial.print(dht.temperature_F); Serial.print(" *F\t");
Serial.print(dht.computeHeatIndex_C()); Serial.print(" *C\t");
Serial.print(dht.computeHeatIndex_F()); Serial.println(" *F");

// Wait a few seconds between measurements. The DHT22 should not be read
at a higher frequency of
// about once every 2 seconds. So we add a 3 second delay to cover this.
delay(3000);
}

```

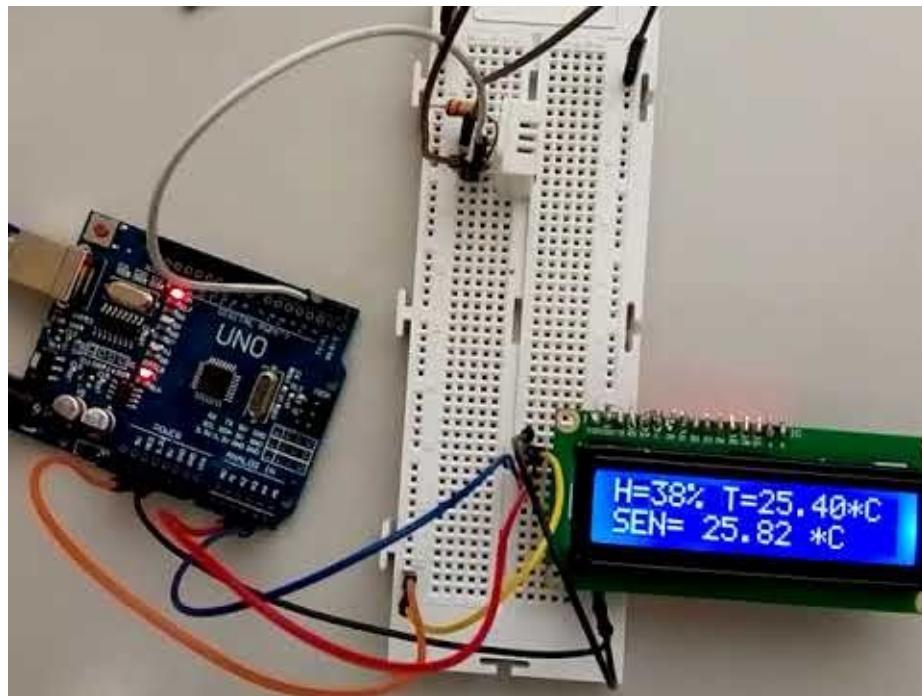


El resultado es:

DHT22 Humidity - Temperature Sensor				
RH	Temp (C)	Temp (F)	Heat Index (C)	Heat Index (F)
43.60 %	21.10 *C	69.98 *F	25.05 *C	77.09 *F
43.60 %	21.10 *C	69.98 *F	25.05 *C	77.09 *F
43.70 %	21.10 *C	69.98 *F	25.05 *C	77.09 *F

Montaje por LCD

Ahora vamos a conectarlo por LCD :



[Video link](#)

Conecciones

Si tienes el DHT11 o si tienes el DH12 [lo has visto ya](#) Ahora añade el [LCD con el I2C](#)

Programa

En este caso lo hacemos con el DHT12 ya sabes que si utilizas DHT11 no mide la humedad y la sensación térmica

```

1 #include <Wire.h>
2 #include <LiquidCrystal_I2C.h>
3
4 #include "cactus_io_DHT22.h"
5
6 #define DHT22_PIN 2      // what pin on the arduino is the DHT22 data line c
7 onnected to
8
9 DHT22 dht(DHT22_PIN);
10 LiquidCrystal_I2C lcd(0x3F, 16, 2); //Create the lcd object address 0x3F and
11 16 columns x 2 rows
12
13 int trigPin = 8;
14 int echoPin = 9;
15 float duration; //time of round trip
16 float cm=0; //To store the obtained value cm value=0
17

```

```
18
19 void setup() {
20
21     // Inicializar el LCD
22     lcd.init();
23     lcd.backlight(); //Encender la luz de fondo.
24
25
26 }
27 void loop(){
28
29     dht.readHumidity();
30     dht.readTemperature();
31
32     lcd.setCursor(0, 0); // Escribimos el Mensaje en el LCD en una posición 1
33 ,0
34     lcd.print("H=");
35     lcd.print(dht.humidity,0); lcd.print("% T=");
36     lcd.print(dht.temperature_C,2); lcd.print("*C");
37     lcd.setCursor(0, 1); // Escribimos el Mensaje en el LCD en una posición 1
38 ,0
39     lcd.print("SEN= ");
40     lcd.print(dht.computeHeatIndex_C()); lcd.print(" *C");
41     delay(3000);
42 }
```

Processing

Es un programa similar al IDE de Arduino que has estado manejando hasta ahora, sólo cambia en un botón de PLAY y en otro de STOP. Es software abierto desarrollado en Java por *Ben Fry* y *Casey Reas* a raíz de una reflexión en un congreso donde se detectó esta necesidad. Te lo puedes descargar de <http://processing.org>

¿Para qué sirve?

Es muy común tener la necesidad de representar los datos que nos da Arduino en un entorno visual mucho más atractivo que el monitor serie que nos ofrece IDE Arduino. Si quieras saber las instrucciones que tiene y más información [consulta esta página](#).

Saber el puerto de conexión

Para empezar a utilizar Processing con nuestro Arduino necesitamos saber en qué puerto se conecta, una forma fácil es cargar y ejecutar este código con el Arduino conectado y que liste los puertos, esta instrucción `printArray(Serial.list());`; nos lo puede decir

Y el resultado puedes ver que sale abajo en la consola [0] "COM4" luego es el 0 en mi caso

```

1 // Example by Tom Igoe
2
3 import processing.serial.*;
4
5 // The serial port
6 Serial myPort;
7
8 // List all the available serial ports
9 printArray(Serial.list());
10
11
12
13
14
15

```

[0] "COM4"

Consola Errores

Montaje 10 Representación gráfica de medidas con Processing.

Una vez obtenidos los datos de temperatura y humedad a través del sensor DHT11 desde Arduino, enviamos, a través del puerto serie, estos datos al PC, donde tenemos ejecutando un programa en Processing que está "escuchando" el puerto serie, obteniendo los datos y representándolos en pantalla. Simultáneamente guardamos los datos en un archivo de texto que posteriormente podremos analizar en una hoja de cálculo.

Programa a cargar en el ARDUINO

Cargamos este programa, fíjate que sólo ponemos un valor de la temperatura, si queremos representar la humedad, quitamos el comentario de la temperatura y ponemos el de la humedad

IMPORTANTE: No hay que tener abierto el monitor serie del IDE de Arduino porque ocupa el puerto y, por lo tanto, no deja leer los datos a Processing

Si fuera un DHT12 en vez de un DHT11 poner comentarios a las 4 primeras líneas delante // y quitárselas a las 3 siguientes

```

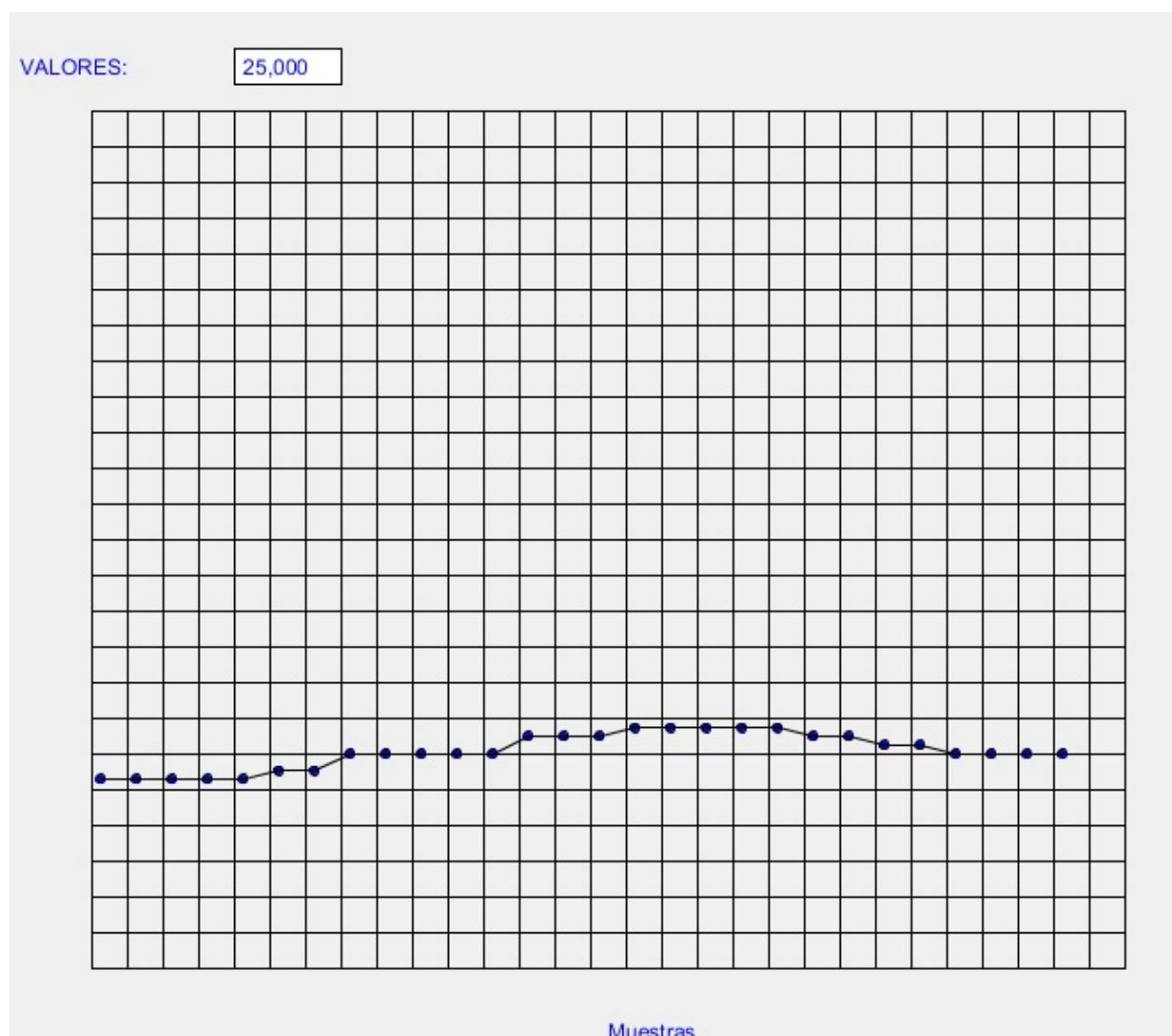
1  //////////////// CON DHT11 ///////////////////////
2  #include "DHT.h"
3  #define DHTPIN 2
4  #define DHTTYPE DHT11 // DHT 11
5  DHT dht(DHTPIN, DHTTYPE);
6  //////////////////////////////// SI FUERA DHT12 /**
7  ///#include "cactus_io_DHT22.h"
8  ///#define DHT22_PIN 2
9  //DHT22 dht(DHT22_PIN);
10 /////////////////
11 void setup() {
12   Serial.begin(9600);
13   dht.begin();
14 }
15 void loop() {
16
17   delay(2000);
18   float h = dht.readHumidity();
19   float t = dht.readTemperature();
20   if (isnan(h) || isnan(t)) {
21     Serial.println("Fallo al leer el sensor DHT11");
22     return;
23   }
24   // Únicamente enviar a Processing una variables t o h
25   //Serial.println(h);
26   Serial.println(t);
27 }
```

Programa a Cargar en PROCESING

[Aquí lo tienes](#) (rar - 1,96 KB), sólo representa un valor, está puesto en el puerto 0 puertoArduino = new Serial(this, Serial.list()[0], 9600);

El resultado puedes verlo aquí abajo para la temperatura, el aumento se debe a aplicar vaho al sensor:

Otra versión más sofisticada lo tienes en [esta página](#)



Otro programa de visualización de datos

En este caso no vamos a representar los datos en forma de gráfica, sino por colores, y además vamos a añadir un botón que encienda un LED conectado por simplicidad en el pin 13

Programa a cargar en el ARDUINO

El programa lee la temperatura y lo escribe en el puerto serie en forma de byte. También lee el puerto serie para cambiar el estado del led.

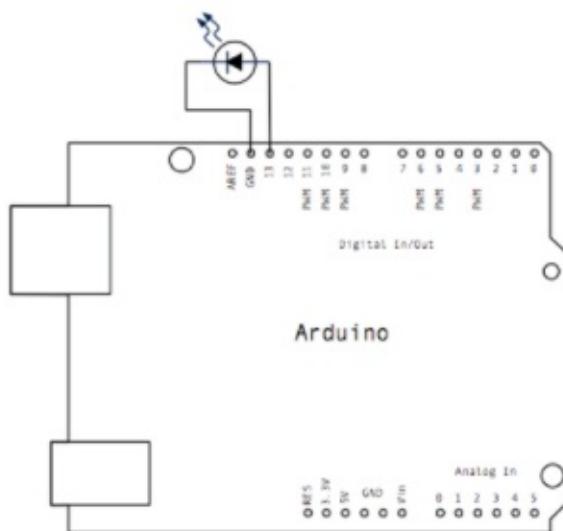
```

1 #include "DHT.h"
2 #define DHTPIN 2
3 #define DHTTYPE DHT11 // DHT 11
4 DHT dht(DHTPIN, DHTTYPE);
5
6 boolean status=LOW; //Estado del led
7 void setup() {
8     Serial.begin(9600);
9 }
```

```

10  pinMode(13, OUTPUT);
11  dht.begin();
12 }
13
14 void loop() {
15   delay(100);
16
17   //float h = dht.readHumidity();
18   int temp = dht.readTemperature();
19   Serial.write(temp); //Enviamos los datos en forma de byte
20   if(Serial.available()>0)//Si el Arduino recibe datos a través del puerto
21   serie
22   {
23     byte dato = Serial.read(); //Los almacena en la variable "dato"
24     if(dato==65) //Si recibe una "A" (en ASCII "65")
25     {
26       status=!status; //Cambia el estatus del led
27     }
28     digitalWrite(13,status);
29   }
30 }
```

En el Arduino tenemos que poner el sensor de temperatura y humedad tal y como se ha explicado en el Montaje 8 y además un led en el 13



Programa en Processing

Extraido de la página <http://diymakers.es/arduino-processing-primeros-pasos/> pero adaptado, te lo puedes descargar aquí (rar - 31,02 KB)(recuerda cambiar port = new Serial(this, Serial.list()[0], 9600); por tu puerto)



Temperatura = 18 °C



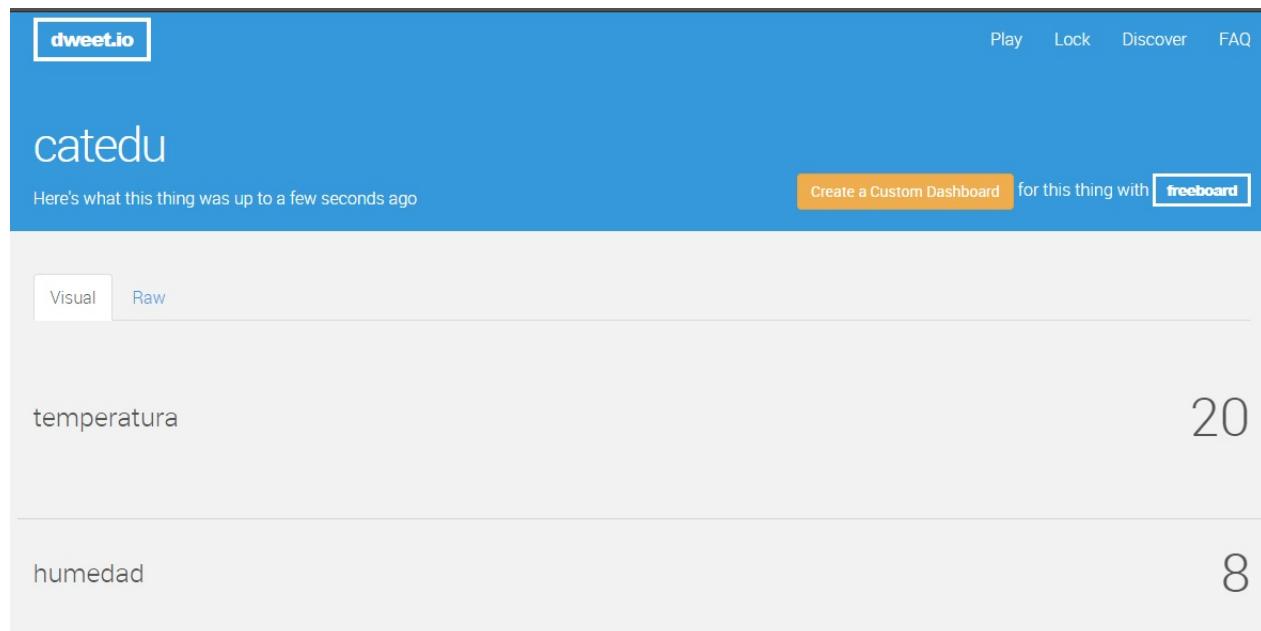
LED OFF

dweet.io

Disponemos del portal web **dweet.io** que nos ofrece un servicio para enviar y representar datos en la nube sin necesidad, ni si quiera, de registrarnos en la plataforma.

Vamos a ver los pasos a seguir:

1. Probamos la plataforma introduciendo un dato, para ello en el navegador tecleamos por ejemplo (cambia **catedu** por tu nombre):
<https://dweet.io/dweet/for/catedu?temperatura=20>
2. Abre otra pestaña del navegador o utiliza un móvil para seguir el dato: <https://dweet.io/follow/catedu>
3. Prueba añadiendo otra variable, en este caso la humedad: <https://dweet.io/dweet/for/catedu?temperatura=20&humedad=8>

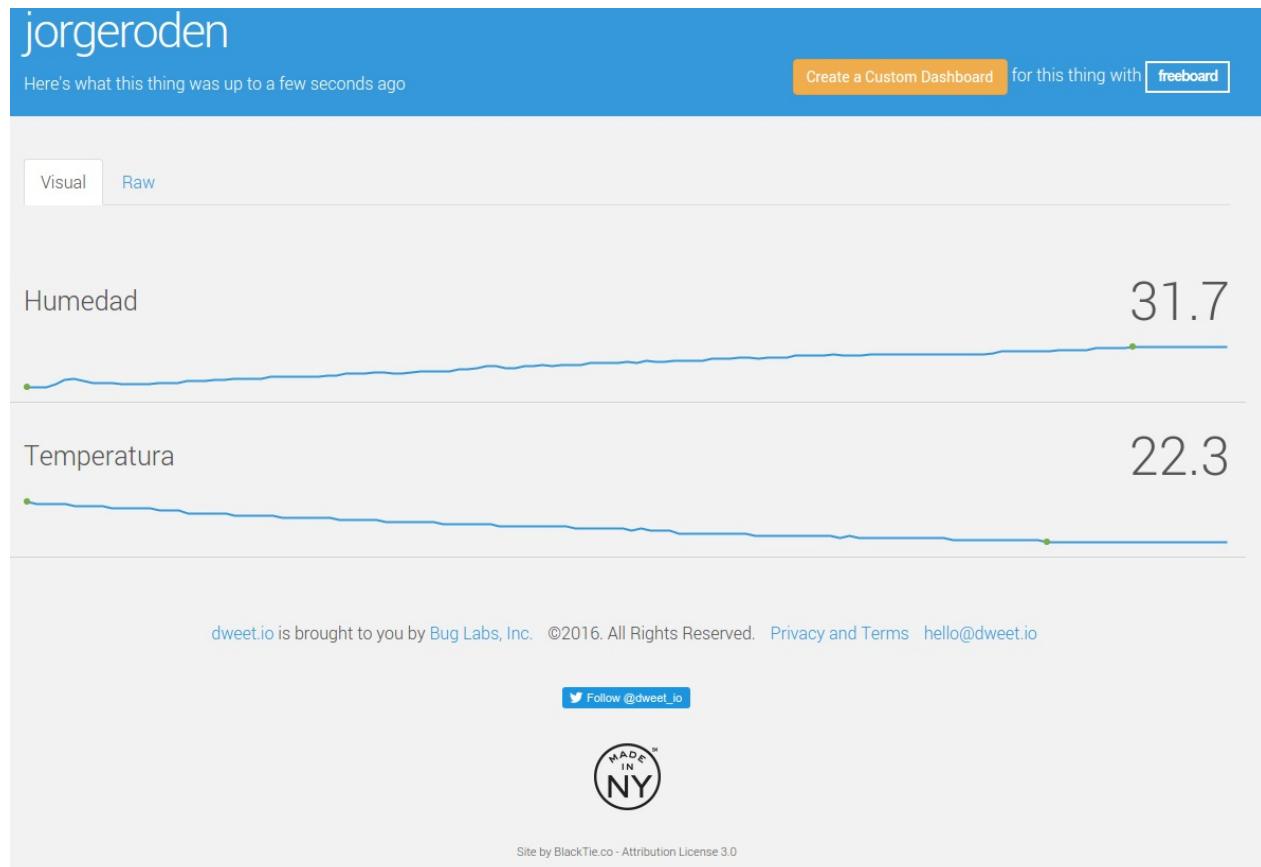


Automatizamos el proceso de recogida de datos desde Arduino con un programa en Processing, que enviará datos a través del navegador a dweet.io.

IMPORTANTE: No hay que tener abierto el monitor serie del IDE de Arduino porque ocupa el puerto y, por lo tanto, no deja leer los datos a Processing.

REPRESENTACIÓN DE DATOS EN EL NAVEGADOR:

Dweet.io nos ofrecerá los datos de la siguiente manera:



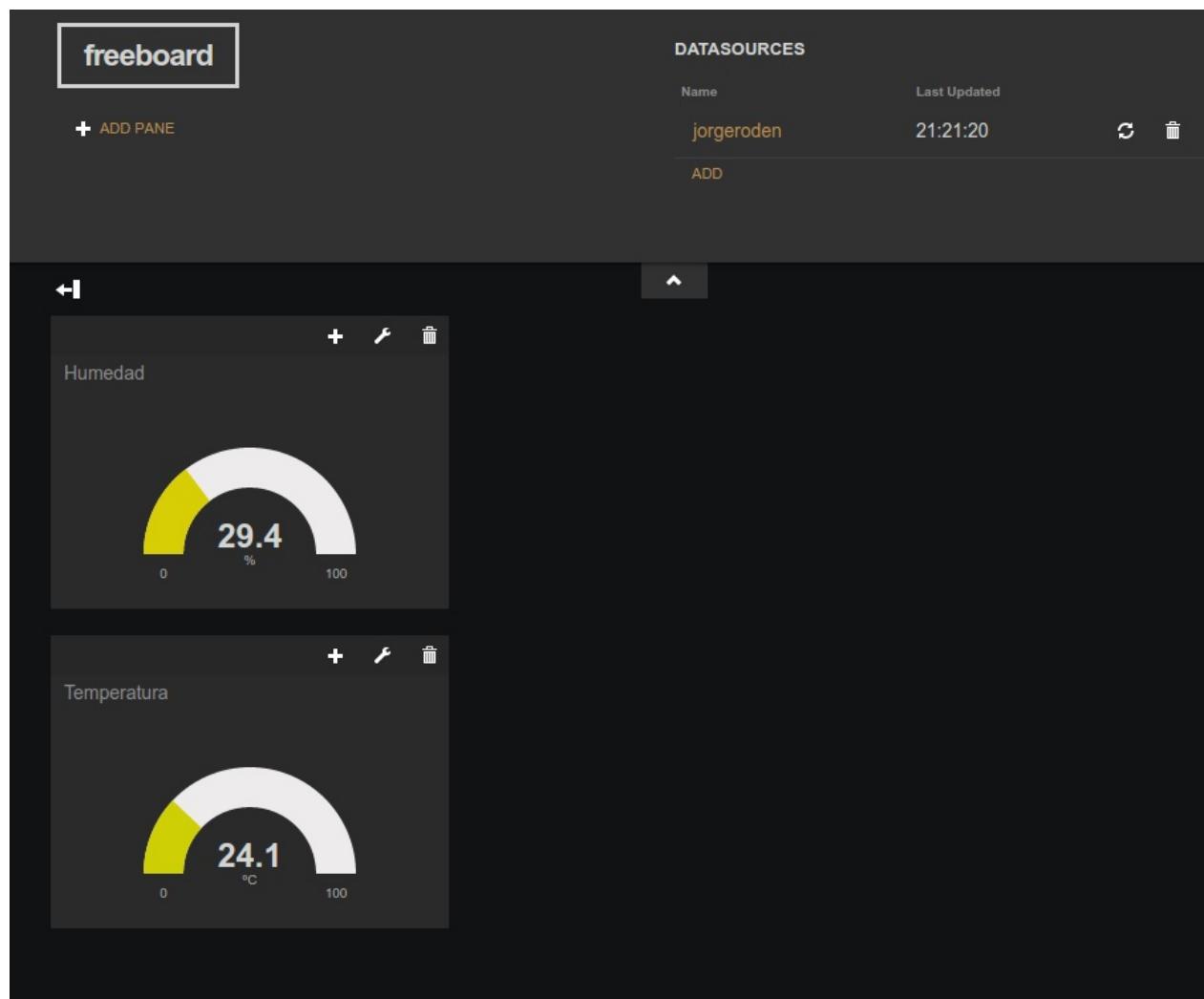
Si queremos algo más vistoso podemos utilizar el servicio [freeboard.io](#) aunque en este caso nos tendremos que registrar en la web.

Una vez registrados podemos crear paneles indicadores configurados a nuestro gusto para visualizar la información. Primero habrá que añadir como fuente de datos Dweet.io y nuestro nombre utilizado allí (jorgeroden en el ejemplo).

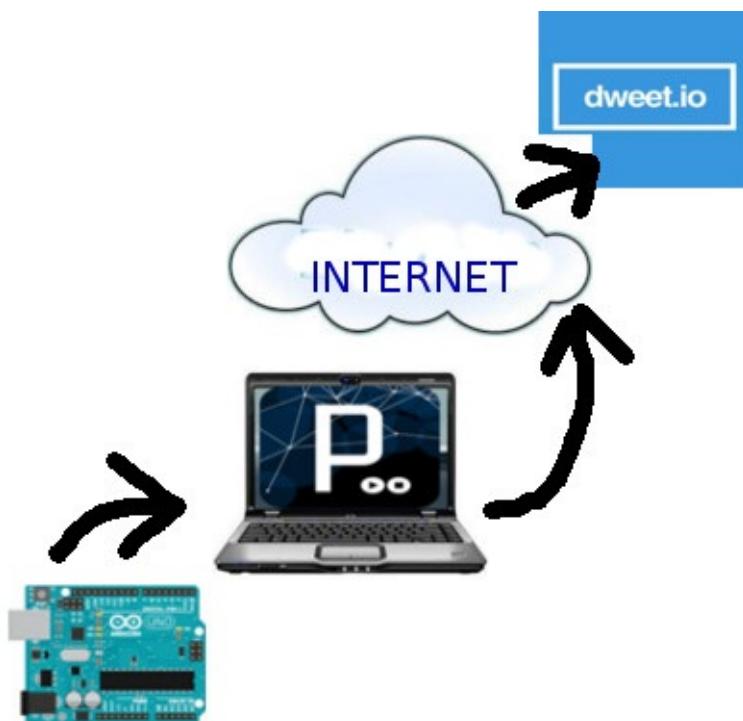
Después creamos un panel indicando que la fuente de datos que queremos utilizar y la variable en cuestión a visualizar.

The screenshot shows the freeboard.io configuration interface. At the top, there is a header with "freeboard" and a "DATA SOURCES" section showing a single entry for "jorgeroden" last updated at 21:20:37. Below this is a "WIDGET" configuration panel. The "TYPE" is set to "Gauge". The "TITLE" is "Temperatura". The "VALUE" field contains the expression "datasources['jorgeroden']['Temperatura']". The "UNITS" field is set to "°C". The "MINIMUM" is 0 and the "MAXIMUM" is 100. There are buttons for "[SAVE]" and "[CANCEL]" at the bottom right. A yellow box highlights the "UNITS" input field.

¡Y resultado puede ser de este tipo!



Montaje 11 Visualización de datos en la nube. Internet de las cosas.



Monitorizar los datos de temperatura y humedad obtenidos del sensor DHT11 en la nube.

Para ello disponemos del portal web **dweet.io** que nos ofrece un servicio para enviar y representar datos en la nube sin necesidad, ni si quiera, de registrarnos en la plataforma.

Vamos a ver los pasos a seguir:

1. Probamos la plataforma introduciendo un dato, para ello en el navegador tecleamos por ejemplo (cambia **CATEDU** por tu nombre):
<https://dweet.io/dweet/for/CATEDU?temperatura=20>
2. Abre otra pestaña del navegador o utiliza un móvil para seguir el dato: <https://dweet.io/follow/CATEDU>
3. Prueba añadiendo otra variable, en este caso la humedad: <https://dweet.io/dweet/for/CATEDU?temperatura=20&humedad=8>

Automatizamos el proceso de recogida de datos desde Arduino con un programa en Processing, que enviará datos a través del navegador a dweet.io.

IMPORTANTE: No hay que tener abierto el monitor serie del IDE de Arduino porque ocupa el puerto y, por lo tanto, no deja leer los datos a Processing.

PROGRAMA A CARGAR EN ARDUINO:

Si fuera un DHT12 en vez de un DHT11 poner comentarios a las 4 primeras líneas delante // y quitárselas a las 3 siguientes

```

1  /// ////////////////// CON DHT11 ///////////////////
2  #include "DHT.h"
3  #define DHTPIN 2
4  #define DHTTYPE DHT11 // DHT 11
5  DHT dht(DHTPIN, DHTTYPE);
6  //////////////////////////////// SI FUERA DHT12 ///

```

```

7 //##include "cactus_io_DHT22.h"
8 //##define DHT22_PIN 2
9 //DHT22 dht(DHT22_PIN);
10 ///////////////////////////////////////////////////////////////////
11
12 void setup() {
13   Serial.begin(9600);
14   dht.begin();
15 }
16 void loop() {
17
18   delay(2000);
19   float h = dht.readHumidity();
20   float t = dht.readTemperature();
21   if (isnan(h) || isnan(t)) {
22     Serial.println("Fallo al leer el sensor DHT11");
23     return;
24   }
25 // Únicamente enviar a Processing las variables t y h
26   Serial.print("Temperatura=");
27   Serial.print(t);
28   Serial.print("&Humedad=");
29   Serial.println(h);
30 }
```

PROGRAMA A EJECUTAR EN PROCESSING :

```

1 // El puerto serie
2 Serial myPort;
3
4 void setup() {
5   // Lista todos los puertos serie
6   printArray(Serial.list());
7   // OJO: Elige el puerto donde tengas conectado Arduino.
8   // Cambia el "0" de Serial.list()[0] por el orden que
9   // tu puerto ocupe en la lista (0, 1, 2,...).
10  // Si no lo tienes claro qué puerto ocupa Arduino mira
11  // en el IDE Arduino en "Herramientas" mira el puerto que esté seleccionado
12  0.
13  //Fíjate que tenemos la velocidad del puerto a la misma que pusimos en Arduino
14  myPort = new Serial(this, Serial.list()[0], 9600);
15 }
16 void draw() {
17   while (myPort.available() > 0) {
18
19     String lectura = myPort.readStringUntil(lf);
20     if (lectura != null) {
21       println(lectura);
22     }
23   }
}
```

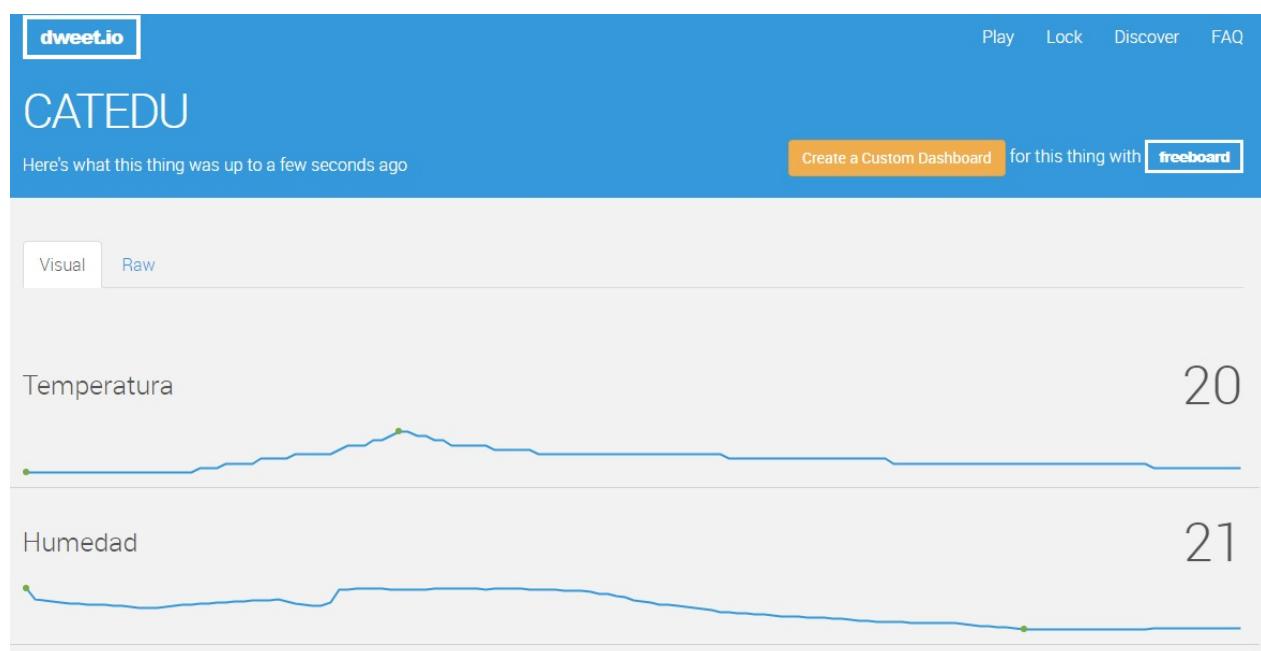
```

3.5.9

24     //IMPORTANTE! cambia CATEDU por tu nombre
25     // visualiza los resultados en esta web https://dweet.io/follow/CATE
26 DU
27
28     loadStrings("https://dweet.io/dweet/for/CATEDU?" + lectura);
29
30 }
31 }
32 }
```

REPRESENTACIÓN DE DATOS EN EL NAVEGADOR:

<https://dweet.io/follow/CATEDU> nos **ofrecería** los datos de la siguiente manera:



No lo hagas, pues NO LO TENGO CONECTADO !! no sale nada !! por eso pone "*nos ofrecería*"

El resultado es espectacular



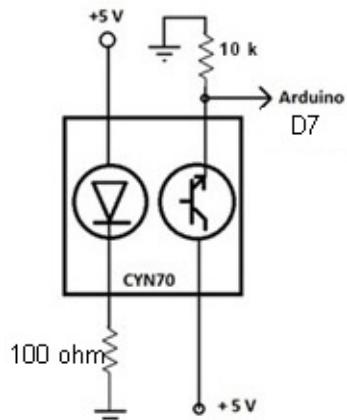
via GIPHY

Sensor de infrarrojos CNY70

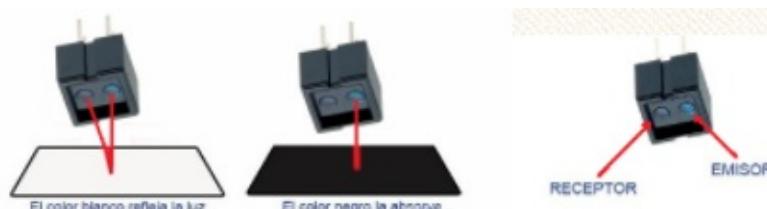
Uno de los sensores más utilizados en robótica, o infinidad de aplicaciones industriales es el CNY-70.

Su nombre técnico es optoacoplador. Se basa en la acción conjunta de un diodo que emite una luz infrarroja (que no vemos) y un fototransistor que detecta el rebote de esta cuando incide sobre algún objeto.

Inicialmente es un sensor analógico y nos da un valor de voltaje proporcional a la luz rebotada, pero podemos utilizarlo también de manera digital. El transistor y el diodo hay que alimentarlo a través de una resistencia, el diodo del orden de Ohmios para dar una señal razonable y el transistor del orden de k para que trabaja en la zona activa.



Su funcionamiento es sencillo, si el receptor recibe la señal del emisor, el transistor conduce, por lo que recibiremos un '1' lógico en el Arduino:



Conocimiento previo

- Programación básica de Arduino.
- Sentencia condicional if-else.
- Comunicación serie.

Objetivos

- Conocer el manejo y aplicaciones del sensor CNY 70
- Realizar las conexiones necesarias sobre el sensor IR.
- Crear un programa para Arduino que obtenga la información proporcionada por el sensor.

Lista de materiales:

- Placa Arduino.
- CNY 70
- Resistencias 10KOhmios (valor orientativo).
- Resistencia de 200 Ohmios (valor orientativo).
- Placa de pruebas.

Vamos a ver una demostración del funcionamiento del sensor:



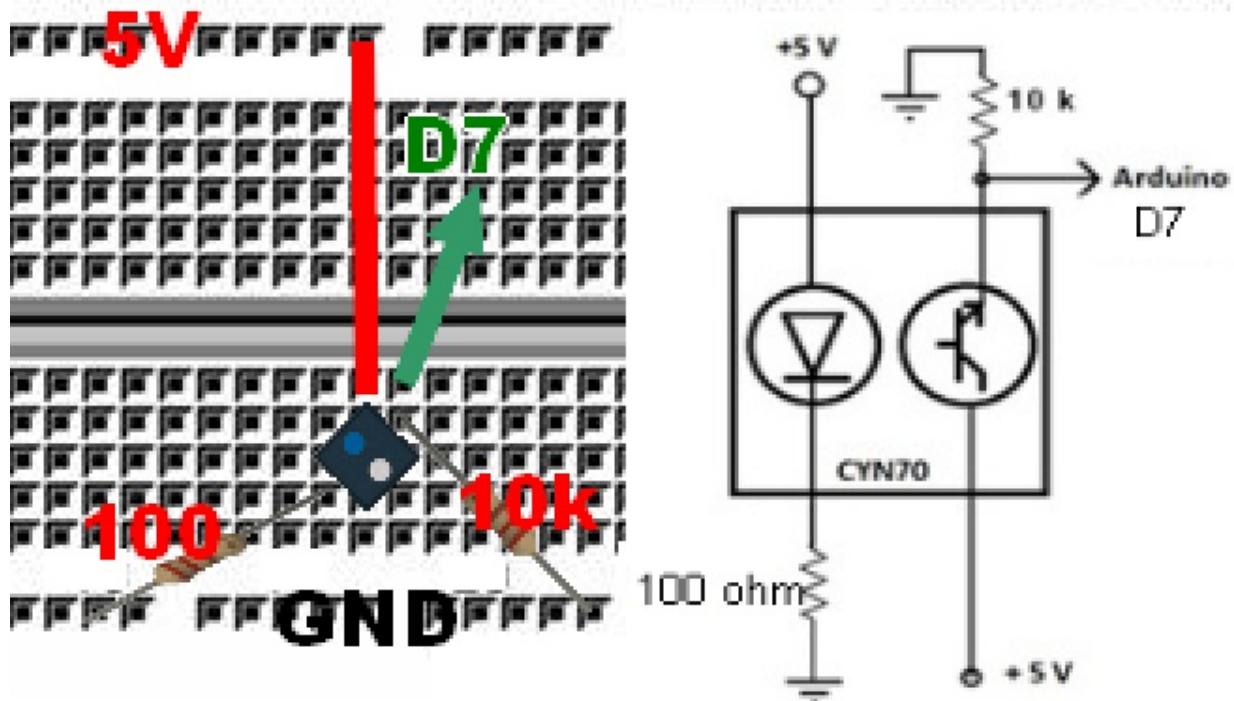
[Video link](#)

Montaje 12 detección linea blanca

El problema del CNY70 es que tiene los pines muy juntos que no se puede poner en medio de la placa protoboard, tenemos pues que utilizar dos opciones:

- Utilizar cables Dupond macho-hembra
- Ponerlo inclinado aprovechando que dos extremos de la diagonal tienen que estar conectados a 5V

Lo mejor es utilizar cables M-H pero si no se tienen, vamos a utilizar la segunda opción, este es el esquema:



Utilizaremos la Edubásica el led rojo, si no tienes, simplemente añade un led al pin 13 [tal y como hicimos en Montaje 1 led parpadeante sin edubásica](#)

El programa a cargar en el Arduino es:

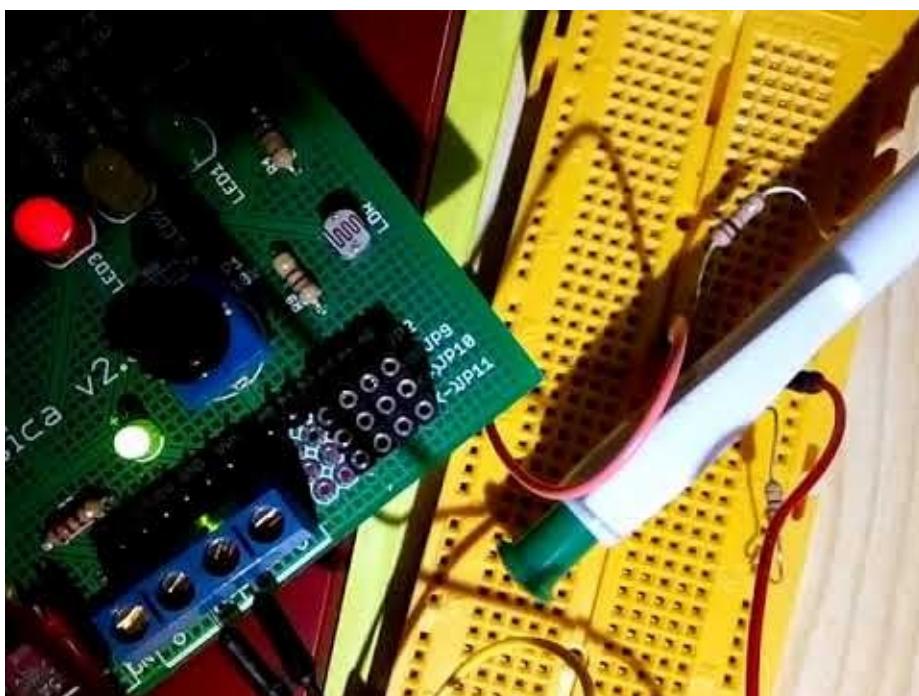
El resultado es:

```

1  /*
2   * Talos Electronics
3   * 8-Noviembre-2015
4   * Rafael lozano Rolón
5   */
6 //Declaracion de variables
7 int sensor = 7;           //ENTRA DIGITAL FÁCILMENTE ACCESIBLE EN EDUBASICA
8 int Valor_cny70 = 0;
9 int Led=5; //LED ROJO DE EDUBÁSICA
10 // cambialo al 13 si no utilizas EDUBÁSICA
11
12 void setup()
13 {
14     Serial.begin(9600);
15     pinMode(sensor, INPUT);
16     pinMode(Led, OUTPUT);
  
```

```

17   digitalWrite(Led, LOW);
18 }
19
20 void loop()
{
21   Valor_cny70=digitalRead(sensor);
22   delay(100); //Esperar 100 ms
23   if(Valor_cny70==0)
24   {
25     Serial.print("Linea negra\n");
26     digitalWrite(Led, HIGH);
27   }
28   else//Si el valor del sensro es 1
29   {
30     Serial.print("Linea blanca\n");
31     digitalWrite(Led, LOW);
32   }
33 }
34 }
```



[Video link](#)

Y en el monitor serie sale:

Linea negra
 Linea negra
 Linea negra
 Linea blanca
 Linea blanca
 Linea blanca
 Linea blanca
 Linea negra

Linea negra

Linea negra

Linea ...

Tecnologías de la comunicación



Las tecnologías de comunicaciones se basan en la transmisión de datos entre puntos distantes. Estos datos, se transmiten en forma de señales eléctricas y pueden ser enviadas através de cables o de manera inalámbrica.

En el Arduino trabajamos con dos tipos de comunicaciones:

- **Alámbrica** Comunicación puerto serie:
 - **PC-Arduino:** La comunicación vía puerto serie:
 - Lo vimos por primera vez en el [semáforo](#).
 - Hay que incializar el puerto serie **Serial.begin(9600)**
 - Con la función **Serial.print** Arduino puede enviar al ordenador los datos que queramos.
 - **Arduino- Arduino**
 - Veremos en esta unidad una forma muy sencilla de comunicarse dos arduinos también por puerto serie.
- **Inalámbrica:** La comunicación vía *Bluetooth*
 - En esta unidad vamos a utilizar un nuevo dispositivo **JY-MCU**
 - Emparejaremos con nuestro Smartphone (Android) y podremos enviar órdenes de nuestro móvil al Arduino.

Si tienes dudas técnicas en este capítulo pon un ticket a <http://soporte.catedu.es/> y te ayudaremos:

Arduino y móvil

Existen módulos adicionales que se pueden conectar a la placa básica Arduino que pueden dotar de una gran funcionalidad a los proyectos que queramos realizar. En esta práctica utilizaremos un **módulo Bluetooth** que nos permite establecer una comunicación inalámbrica con el entorno, el dispositivo elegido más fácil va a ser un **móvil**.

Conocimiento previo

- Programación básica Arduino
- Uso de librerías externas y comunicación serie (para configuración de parámetros).

Objetivos

- Conectar el módulo Bluetooth a Arduino.
- Realizar programas para comunicar Arduino con el exterior vía Bluetooth.
- Configurar los parámetros del módulo de Bluetooth (avanzado).

Lista de materiales:

- Arduino UNO.
- Módulo Bluetooth.
- Móvil con Android



via GIPHY

Teoría Bluetooth

ONDAS:

Una onda es una señal que se propaga por un medio. Por ejemplo el sonido, que es una onda mecánica que viaja usando el aire o cualquier otro material. Pero en el caso de las señales eléctricas pueden ser enviadas por el cable o a través del vacío (no necesitan un medio para transmitirse).

Dependen de 3 parámetros principalmente:

Amplitud: altura máxima de la onda. Hablando de sonido representaría el volumen. Si nos referimos a una onda eléctrica estaríamos representando normalmente el voltaje.

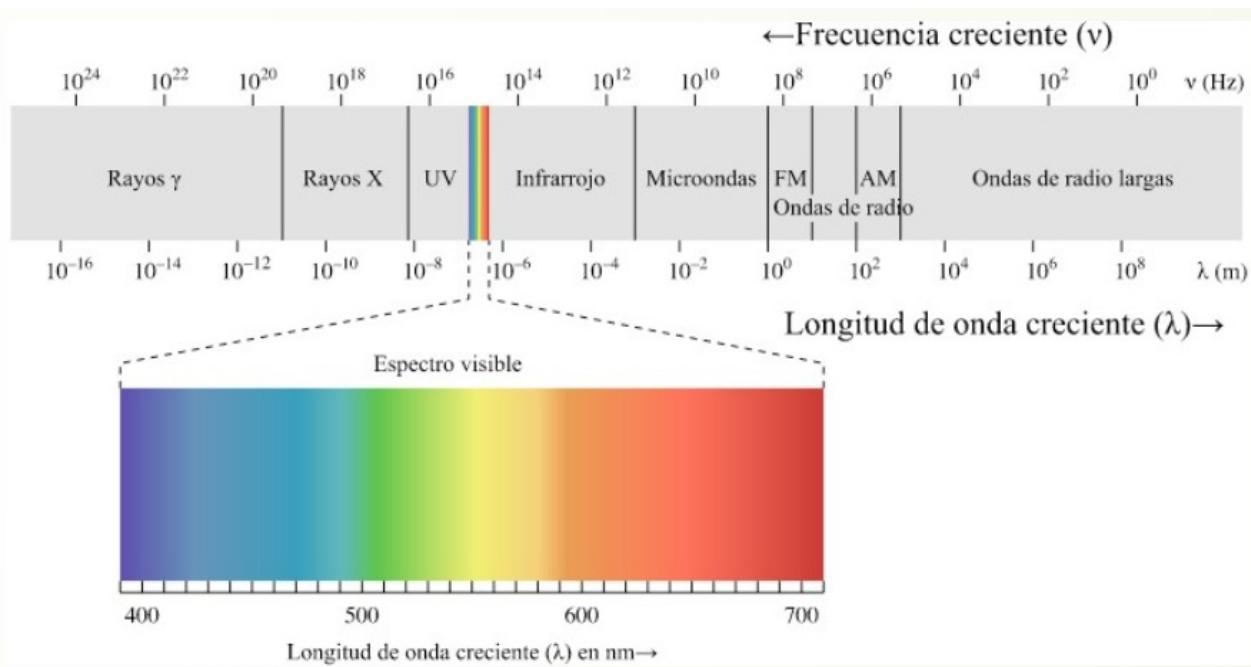
Longitud de onda λ : distancia entre el primer y último punto de un ciclo de la onda (que normalmente se repite en el tiempo).

Frecuencia f : Número de veces que la onda repite su ciclo en 1 segundo (se mide en hertzios).

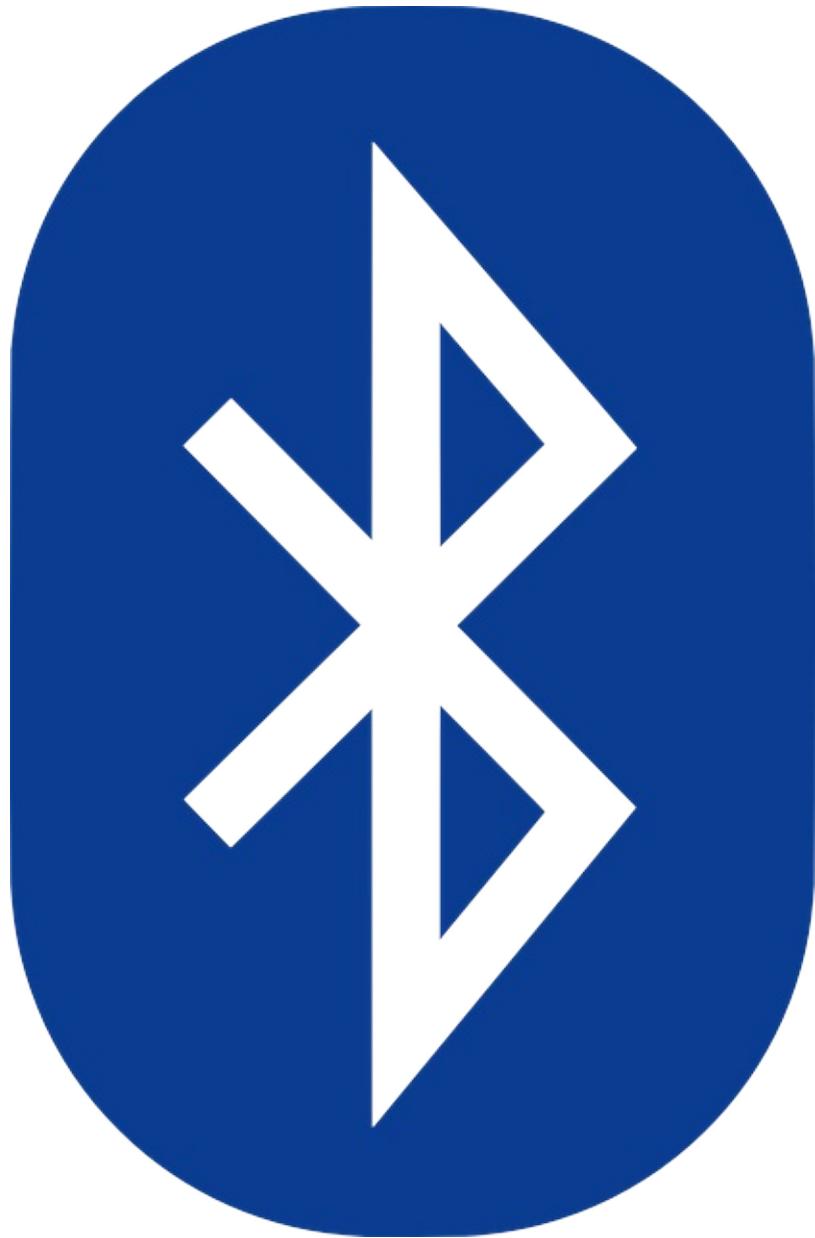
Periodo T es simplemente es la inversa de la frecuencia. $T=1/f$

La relación entre ellas es muy fácil pues las ondas electromagnéticas viajan a la velocidad de la luz c y si velocidad es espacio/tiempo luego $c = \lambda/T$ luego $c = \lambda*f$

Dentro del espectro electromagnético encontramos diferentes tipos de señales dependiendo de las características de su onda.



TRANSMISIÓN INALÁMBRICA: BLUETOOTH.



- Hoy en día, este grupo está formado por miles de empresas y se utiliza no sólo para teléfonos sino para cientos de dispositivos.
- Su curioso nombre viene de un antiguo rey Noruego y Danés, y su símbolo, de las antiguas ruinas que representan ese mismo nombre.
- Bluetooth es una red inalámbrica de corto alcance pensada para conectar pares de dispositivos y crear una pequeña red punto a punto, (sólo 2 dispositivos).
- Utiliza una parte del espectro electromagnético llamado “Banda ISM”, reservado para fines no comerciales de la industria, área científica y medicina. Dentro de esta banda también se encuentran todas las redes WIFI que usamos a diario. En concreto funcionan a 2,4GHz.

Hay 3 clases de bluetooth que nos indican la máxima potencia a la que emiten y por tanto la distancia máxima que podrán alcanzar:

CLASE	POTENCIA	DISTANCIA
Clase 1	100 mW	100 m
Clase 2	2,5 mW	10 m
Clase 3	1 mW	1 m

También es muy importante la velocidad a la que pueden enviarse los datos con este protocolo:

Versión	Velocidad
1.2	1 Mbps
2	3 Mbps
3	24 Mbps
4	24 Mbps

Recuerda que:

- Mbps : Mega Bits por segundo.
- MBps: Mega Bytes por segundo.

¿Te atreves a calcularlo ?

¿Cuantos ciclos por segundo tendrán las ondas que están en la Banda ISM?

¿Cuál es el periodo de esas ondas?

Solución

- a)2.4G
b) $\lambda=c/f= 12.5\text{cm}$

¿Te atreves a calcularlo...?

¿A qué distancia y cuanto tiempo tardarían en enviarse los siguientes archivos por Bluetooth?

1. Un vídeo de 7Mb usando versión 2 clase 2
2. Una imagen de 2.5Mb usando versión 3 clase 1
3. Un archivo de texto de 240KB usando versión 1 clase 1

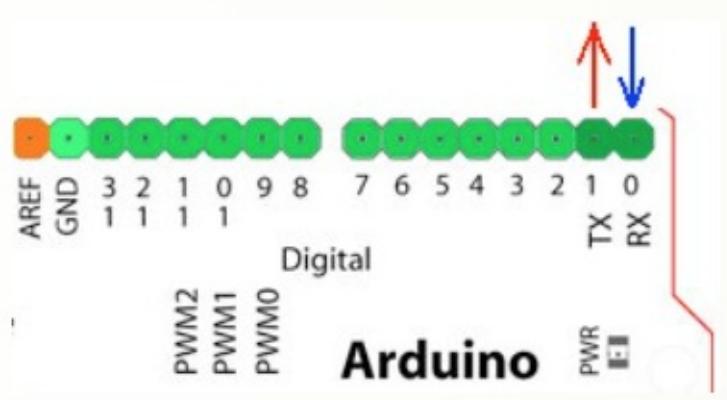
Módulo Bluetooth

Vamos a utilizar en estos ejemplos un módulo **esclavo** de bluetooth **JY-MCU** o también **HC-06** muy común y económico. Es posible usar otros módulos ya que existe un mercado de desarrollo continuo, en cualquier caso el funcionamiento básico es el mismo. Dicho módulo por tratarse de un módulo **esclavo**, está configurado para conectarse a un maestro y recibir órdenes de él.

Inicialmente no necesitas configurarlo, sino que al cargar el código desde el ordenador, conectarás el módulo y este empezará a parpadear indicando que está buscando un master al que conectarse, (por ejemplo tu teléfono o una llave bluetooth usb conectado a un pc).

Como ya sabrás los dispositivos de este tipo tienen que “emparejarse” y tienen que compartir una contraseña para que los datos puedan intercambiarse. Por defecto, estos módulos tienen la contraseña 1234, aunque tanto esto como el nombre, pueden ser actualizados mediante unos comandos especiales, llamados AT y que veremos un poco más adelante.

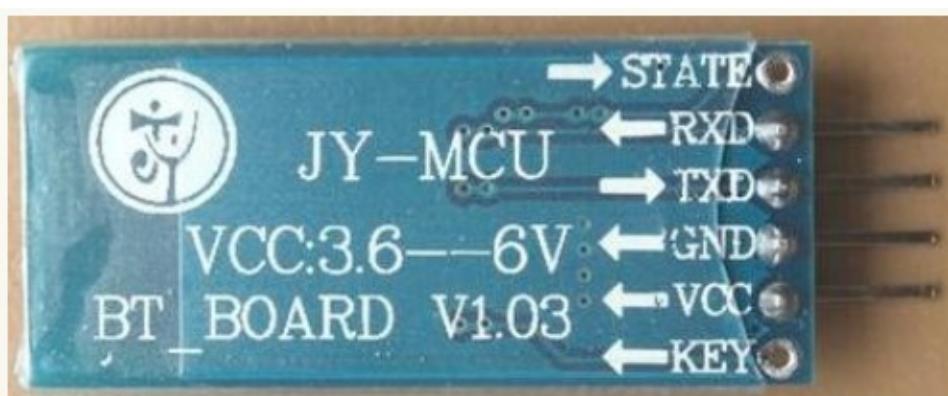
Arduino tiene 2 pines que permiten enviar y transmitir datos serie (uno datos tras otro). Lo usamos continuamente cuando enviamos un programa desde nuestro ordenador a Arduino o cuando hacemos una lectura desde el monitor serie (con un `Serial.print();**`).



Arduino tiene definidos estos pines como:

- pin digital 0: RX <- (Arduino recibe a través de este pin).
- pin digital 1: TX >- (Arduino envía a través de este pin).

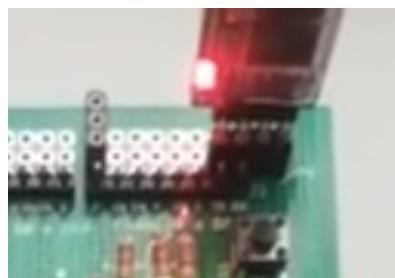
El módulo bluetooth tiene 4 patillas. 2 para la alimentación y 2 para la comunicación.



Es MUY IMPORTANTE conectar de manera correcta estos pines con Arduino para la correcta comunicación. La patilla que emite los datos (TX) en el bluetooth debe estar conectada a la que recibe los datos (RX) en Arduino, y viceversa. Aunque el módulo funciona a 3.3v, normalmente las placas comerciales, (como la que estamos usando), llevan un regulador y las podemos conectar directamente a los 5v de Arduino.

Conección en Edubásica

La conexión es muy fácil, ya tiene JP6 para conectarlo diréctamente, con la luz led mirando hacia dentro de la placa:



Conección sin Edubásica

Es también simple, utilizando una placa Protoboard [[descarga .fzz](#)] pero **intercambiando Rx y Tx** es decir Rx del HC-06 con Tx del Arduino y Tx del HC-06 con Rx del Arduino.



Ordenes

Si la luz está intermitente, el módulo no está vinculado, si está encendido permanente, ya está vinculado.

Una vez vinculado, la orden es sencilla:

```
1 _dato = Serial.read();
```

donde dato es tipo byte : *byte dato;*

Ten en cuenta que estamos usando los 2 mismos pines que Arduino usa para la comunicación USB con el ordenador (0, 1), así que **no puedes usar el monitor serie** para visualizar los datos utilizando el Bluetooth. Igualmente la velocidad tiene que ser igual para entenderse, no pueden ser diferentes.

La APP

Interactuar con el medio es uno de los objetivos primordiales de Arduino. En esta unidad vamos a ver cómo nos podemos comunicar con un dispositivo móvil, posibilitando así el control remoto de la placa.

La comunicación con Arduino es muy sencilla, el uso común de este dispositivo, será como receptor o emisor de datos.

En nuestro caso usaremos caracteres (bytes) que enviaremos desde un master, como un teléfono móvil. Hay muchas aplicaciones gratuitas para enviar datos, por ejemplo, para dispositivos Android podemos utilizar de manera gratuita. Podemos usar **cualquier APP que emita un código por Bluetooth**.

Arduino Bluetooth Control

Esta APP es muy completa y configurable, [aquí para descargarla de Google Play](#).



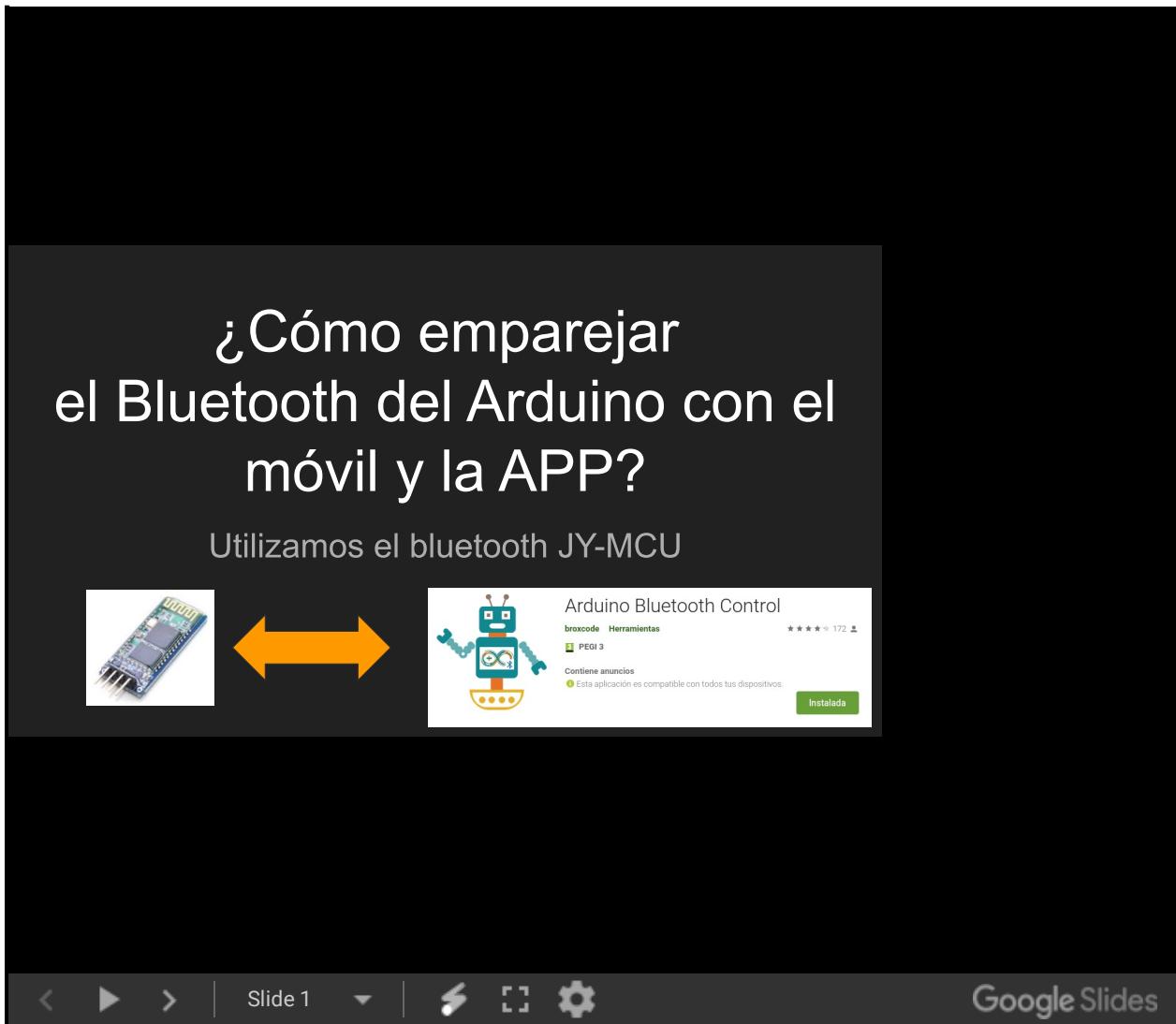
El código de programa que tenemos que cargar en el Arduino se basa en escuchar de forma continua el puerto serie. **Cuando llegue el dato, se ejecutará la acción que le indiquemos.** ¡¡así de sencillo !!



[via GIPHY](#)

Vincular móvil

Hay que vincular nuestro móvil y nuestra APP de Android con el Arduino, para ello sigue [este sencillo tutorial](#):



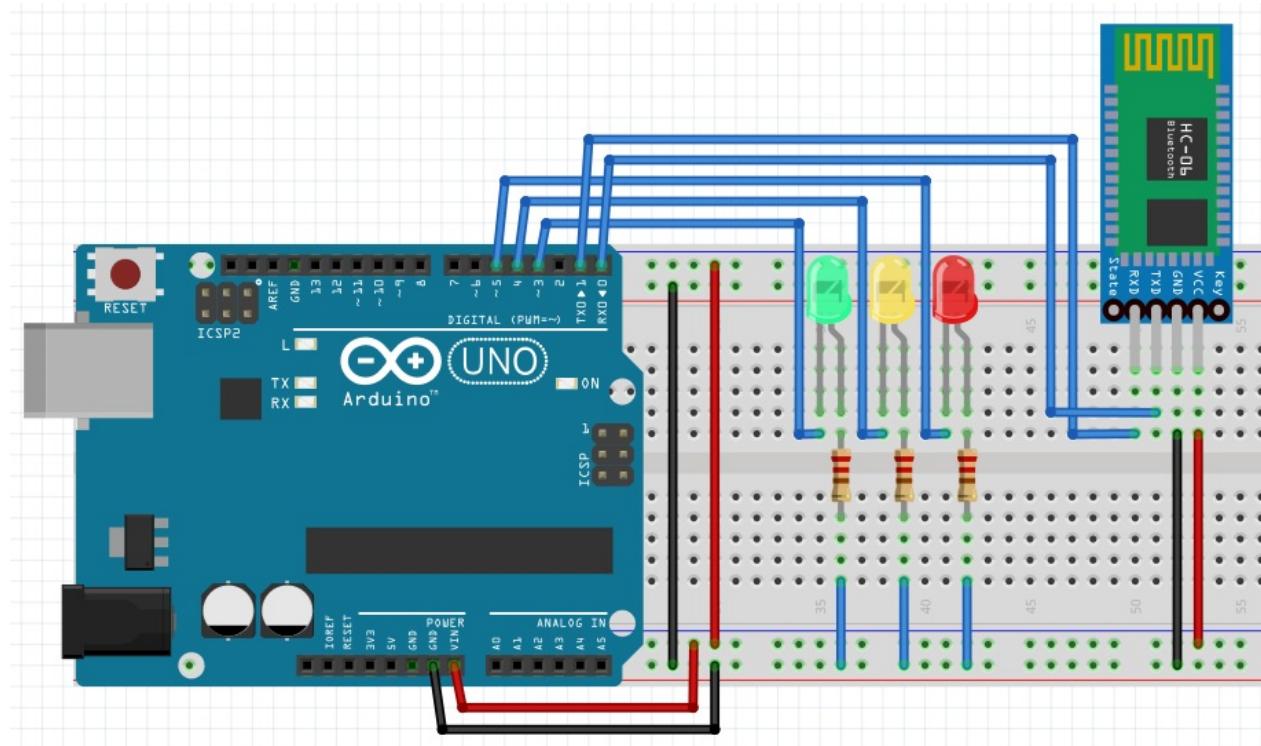
Montaje 1 Encender LEDs

El objetivo de esta práctica es encender los LEDs de EDUBASICA con el móvil:

- Cuando se pulsa la **flecha arriba**, la APP lanza el dato **U** y tiene que encenderse el led **ROJO**.
- Cuando se pulsa el botón **flecha derecha**, la APP lanza el dato **R** y tiene que encenderse el led **AMARILLO**.
- Cuando se pulsa la **flecha abajo**, la APP lanza el dato **D** y tiene que encenderse el led **VERDE**.
- Cuando se pulsa la **flecha izquierda**, la APP lanza el dato **L** y tienen que apagarse todos.

SIN EDUBASICA

No pasa nada, con una placa Protoboard pon 3 leds en D3,D4 y D5 y el módulo Bluetooth.

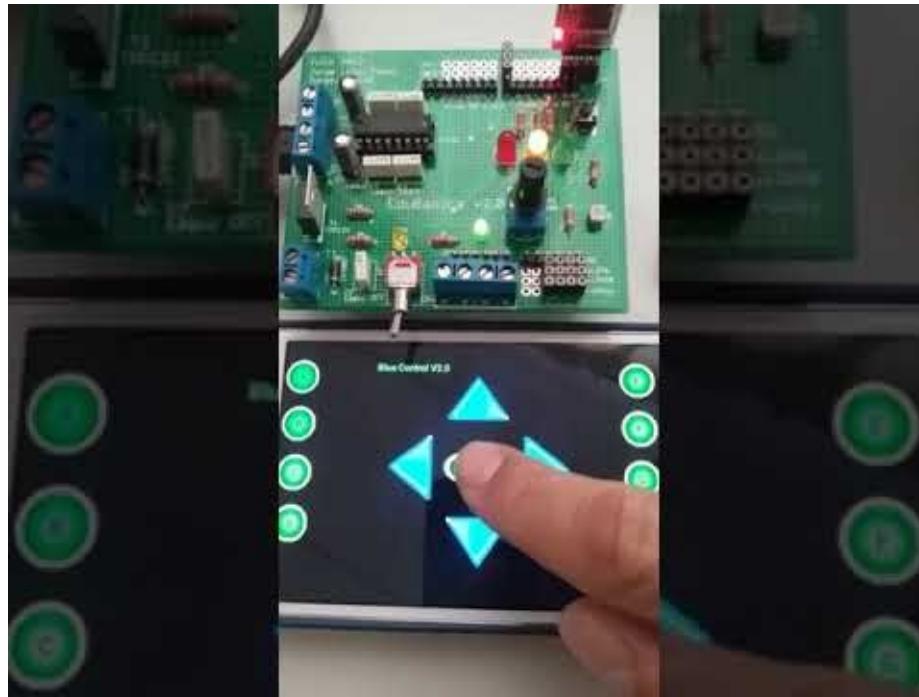


CON EDUBÁSICA

No hay que hacer nada especial, sólo conectar el módulo Bluetooth

RESULTADO

El vídeo está realizado con otra APP ya desfasada, pero sirve igual de ejemplo, es increíble, sólo pasó un mes !



[Video link](#)

PROGRAMA

Este es el programa que tienes que cargar en el Arduino. Síbelo, empareja tu móvil [como hemos visto aquí](#) y conseguirás que la APP encienda los leds como en el vídeo.

```

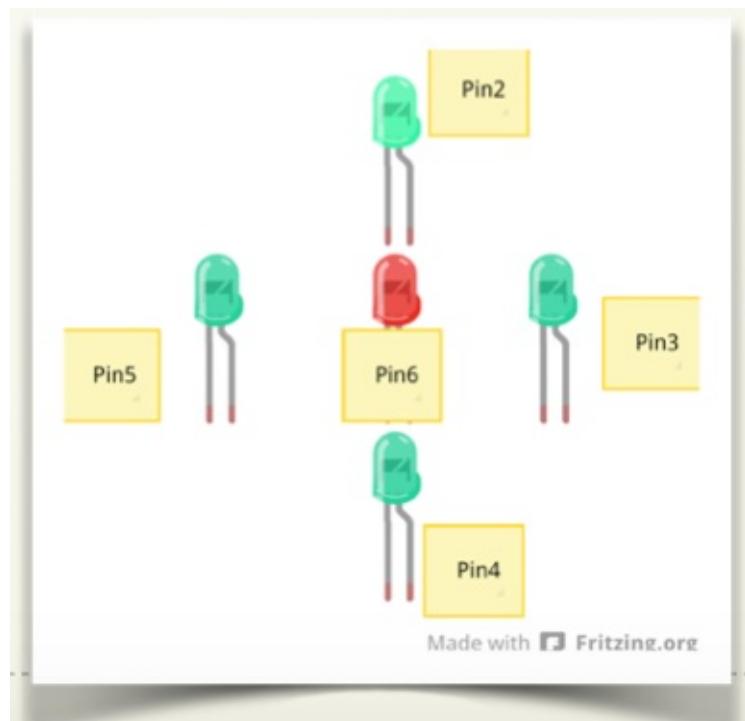
1 //PROGRAMA DONDE SE ENCIENDEN LOS LEDS DE EDUBASICA SEGÚN LA APP ///////////////
2 //////////////////
3 //
4 // FLECHA ARRIBA -> SE ENCIENDE EL LED ROJO
5 // FLECHA DERECHA-> SE ENCIENDE EL LED AMARILLO
6 // FLECHA ABAJO -> SE ENCIENDE EL LED VERDE
7 // FLECHA IZQUIERDA -> SE APAGAN
8 //
9 /////////////////////////////////
10 ///////////////////////////////
11 int ledArriba = 5; //LED ROJO DE EDUBASICA
12 int ledDerecha = 4; //LED AMARILLO DE EDUBASICA
13 int ledAbajo = 3; // LED VERDE DE EDUBASICA
14 /////////////////////////////////
15 ///////////////////////////////
16 void setup() {
17 Serial.begin(9600);
18 pinMode(ledArriba, OUTPUT);
19 pinMode(ledAbajo, OUTPUT);
20 pinMode(ledDerecha, OUTPUT);
21 }
22 /////////////////////////////////
23 ///////////////////////////////
24 void loop() {
25
26

```

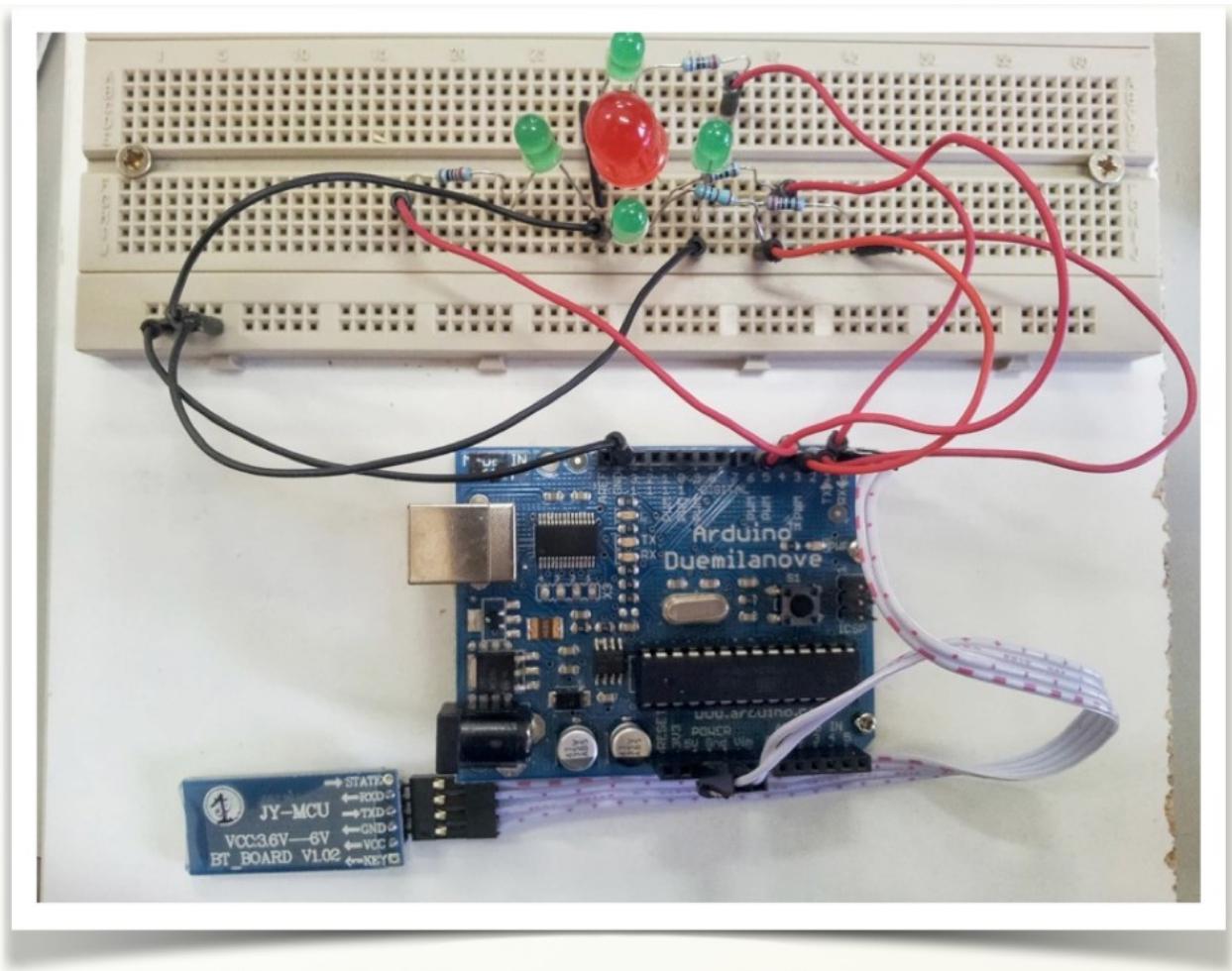
```
27 if(((Serial.read())=='U')){  
28     digitalWrite(ledArriba, HIGH);  
29     digitalWrite(ledAbajo, LOW);  
30     digitalWrite(ledDerecha, LOW);  
31 }  
32 if(((Serial.read())=='D')){  
33     digitalWrite(ledArriba, LOW);  
34     digitalWrite(ledAbajo, HIGH);  
35     digitalWrite(ledDerecha, LOW);  
36 }  
37 if(((Serial.read())=='R')){  
38     digitalWrite(ledArriba, LOW);  
39     digitalWrite(ledAbajo, LOW);  
40     digitalWrite(ledDerecha, HIGH);  
41 }  
42 if(((Serial.read())=='L')){  
43     digitalWrite(ledArriba, LOW);  
44     digitalWrite(ledAbajo, LOW);  
45     digitalWrite(ledDerecha, LOW);  
46 }  
47 }
```

Montaje 1bis Encender LEDs sin EDUBASICA

Este ejemplo es más avanzado, pero no se puede hacer con edubásica: Vamos a ver un ejemplo implementando un **mosaico de LEDs**:



Las conexiones serán las siguientes:



El objetivo es que según la tecla que presionemos en la aplicación “Blue Control”, se encenderá el led correspondiente: (arriba, abajo, izquierda, derecha y centro). Además si pulsamos alguno de los botones laterales, los leds deberán realizar una animación de todos los leds:

- Encendido de los leds en sentido horario.
- Encendido de los leds en sentido antihorario.
- Encendido intermitente de los leds exteriores y el interior.
- Encendido intermitente de todos los leds.

INVENTA MÁS ANIMACIONES PARA INCLUIRLAS EN LOS BOTONES QUE SOBRAN EN LA APLICACIÓN

Para simplificar el código, hemos creado funciones para ejecutar cada una de las animaciones, estas funciones están al final del programa.

La lectura se hace mediante 2 funciones:

- La función **Serial.available()** nos indica si hay un dato disponible en el puerto serie (verdadero/falso)
- Con la función **dato = Serial.read();** guardamos el dato en una variable (de tipo byte)

Con esto tendremos el código ASCII del carácter enviado por el maestro, por ejemplo si hemos enviado una A tendremos el 65, B = 66, a = 97, b = 98, ... (ascii.cl/es/)

Lo único que nos queda es comparar el dato recibido y elegir la acción que tiene que hacer Arduino.

Programa

```

1 //LEDS CONECTADOS EN FORMA DE ESTRELLA Y UNO EN EL CENTRO
2 //MEDIANTE BLUETERM ENVIAMOS CARACTERES (SEGUN LAS TECLAS) Y SE
3 //ENCIENDEN / APAGAN LOS LEDS SELECCIONADOS
4

```

```
5 int ledArriba = 2;
6 int ledCentro = 6;
7 int ledAbajo = 4;
8 int ledDerecha = 3;
9 int ledIzquierda = 5;
10 byte dato;
11 ///////////////////////////////////////////////////////////////////
12 void setup() {
13 Serial.begin(9600);
14 pinMode(ledArriba, OUTPUT);
15 pinMode(ledAbajo, OUTPUT);
16 pinMode(ledIzquierda, OUTPUT);
17 pinMode(ledDerecha, OUTPUT);
18 pinMode(ledCentro, OUTPUT);
19 }
20 void loop() {
21 if (Serial.available()) //Guardamos en la variable dato el valor leido
22     dato= Serial.read();
23 //Comprobamos el dato
24 switch(dato)
25     {//Si recibimos una ...
26     case 85: //ARRIBA
27     {
28         digitalWrite(ledArriba, HIGH);
29         digitalWrite(ledAbajo, LOW);
30         digitalWrite(ledDerecha, LOW);
31         digitalWrite(ledIzquierda, LOW);
32         digitalWrite(ledCentro, LOW);
33
34         break;
35     }
36     case 68: //"/U": ABAJO
37     {
38         digitalWrite(ledArriba, LOW);
39         digitalWrite(ledAbajo, HIGH);
40         digitalWrite(ledDerecha, LOW);
41         digitalWrite(ledIzquierda, LOW);
42         digitalWrite(ledCentro, LOW);
43
44         break;
45     }
46     case 67: //"/D": CENTRO
47     {
48         digitalWrite(ledArriba, LOW);
49         digitalWrite(ledAbajo, LOW);
50         digitalWrite(ledDerecha, LOW);
51         digitalWrite(ledIzquierda, LOW);
52         digitalWrite(ledCentro, HIGH);
53
54         break;
55     }
56     case 76: //"/L": IZQ
57     {
```

```
58         digitalWrite(ledArriba, LOW);
59         digitalWrite(ledAbajo, LOW);
60         digitalWrite(ledDerecha, LOW);
61         digitalWrite(ledIzquierda, HIGH);
62         digitalWrite(ledCentro, LOW);
63         break;
64     }
65     case 82: // "R": DCH
66     {
67         digitalWrite(ledArriba, LOW);
68         digitalWrite(ledAbajo, LOW);
69         digitalWrite(ledDerecha, HIGH);
70         digitalWrite(ledIzquierda, LOW);
71         digitalWrite(ledCentro, LOW);
72         break;
73     }
74     case 97: // Recibimos "a"
75     {
76         sentidoReloj();
77         break;
78     }
79     case 98: // Recibimos "b"
80     {
81         sentidoContrario();
82         break;
83     }
84     case 99: // Recibimos "c"
85     {
86         fueraDentro();
87         break;
88     }
89 }
90 }
91
92 void sentidoReloj(){
93     digitalWrite(ledArriba, HIGH);
94     delay(100);
95     digitalWrite(ledArriba, LOW);
96     delay(10);
97     digitalWrite(ledDerecha, HIGH);
98     delay(100);
99     digitalWrite(ledDerecha, LOW);
100    delay(10);
101    digitalWrite(ledAbajo, HIGH);
102    delay(100);
103    digitalWrite(ledAbajo, LOW);
104    delay(10);
105    digitalWrite(ledIzquierda, HIGH);
106    delay(100);
107    digitalWrite(ledIzquierda, LOW);
108    delay(10);
109 }
110 }
```

```
111 void sentidoContrario(){
112     digitalWrite(ledArriba, HIGH);
113     delay(100);
114     digitalWrite(ledArriba, LOW);
115     delay(10);
116     digitalWrite(ledIzquierda, HIGH);
117     delay(100);
118     digitalWrite(ledIzquierda, LOW);
119     delay(10);
120     digitalWrite(ledAbajo, HIGH);
121     delay(100);
122     digitalWrite(ledAbajo, LOW);
123     delay(10);
124     digitalWrite(ledDerecha, HIGH);
125     delay(100);
126     digitalWrite(ledDerecha, LOW);
127     delay(10);
128 }
129 void fueraDentro(){
130     digitalWrite(ledArriba, HIGH);
131     digitalWrite(ledDerecha, HIGH);
132     digitalWrite(ledAbajo, HIGH);
133     digitalWrite(ledIzquierda, HIGH);
134     delay(1000);
135     digitalWrite(ledArriba, LOW);
136     digitalWrite(ledDerecha, LOW);
137     digitalWrite(ledAbajo, LOW);
138     digitalWrite(ledIzquierda, LOW);
139     delay(10);
140     digitalWrite(ledCentro, HIGH);
141     delay(1000);
142     digitalWrite(ledCentro, LOW);
143     delay(10);
144 }
```

Configuración avanzada

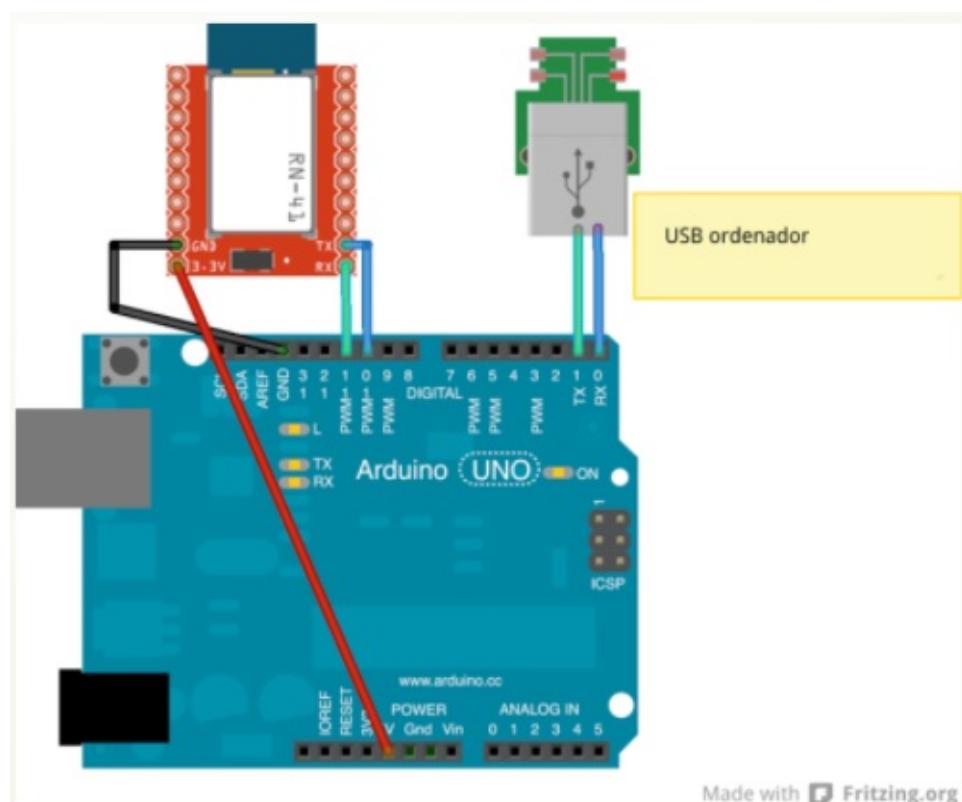
Si quieres modificar cosas como la velocidad de conexión, el nombre o la contraseña de tu módulo, aquí te dejamos un código para que subas a tu arduino y mediante el monitor serie lo configures.

Nota: No se pretende que realices el montaje, pero creemos que es importante que veas que el módulo Bluetooth no es cerrado en su configuración.

Para ello hemos creado un nuevo puerto serie para que no interfiera con el USB y podamos usarlo simultáneamente, lo haremos en las patillas 10 y 11.

Deberás conectar el módulo como ves en la figura, y luego cargar el código. Una vez subido, abre la consola serie y (EN MAYÚSCULAS) ejecuta los comandos que necesites.

Una vez finalizado, puedes desconectar el módulo BT y usarlo con normalidad.



```

1 #include <SoftwareSerial.h>
2 SoftwareSerial Serial1(11, 10);
3 //10:TX DEL MODULO, 11:RX DEL MODULO
4 String command = ""; // guardará la respuesta desde el BT
5 void setup()
6 {
7     Serial.begin(9600); //CONEXION SERIE USB CON ORDENADOR
8     Serial1.begin(9600); //CONEXION SERIE PARA EL MODULO BT
9     Serial.println("Terminal para configurar BT(JY-MCU)");
10    Serial.println("Comandos AT. USA MAYUSCULAS");
11    Serial.println("-----");
12    Serial.println("Comando, Respuesta, Parametros");
13    Serial.println("AT, OK,--Verifica la conexión--");
14    Serial.println("AT+VERSION,--Devuelve la version--");

```

```
15     Serial.println("AT+BAUDx, OKxxxx, Set x to: 1=1200 \
16 2=2400 3=4800 4=9600 5=19200 6=38400 7=57600 8=115200 \
17 --para cambiar la velocidad--");
18     Serial.println("AT+NAMEstring, nombrenuevo (20max)");
19     Serial.println("AT+PINxxxx, Cambia el pin (1234 por defecto)");
20     Serial.println("AT+ROLEx,1=MASTER/0=SLAVE --SOLO MASTER");
21     Serial.println();
22 }
23
24 void loop()
25 {
26     //Chequear si hay datos desde BT
27     if (Serial1.available()) {
28         while(Serial1.available()) {
29             command += (char)Serial1.read();
30         }
31         Serial.println(command);
32         command = "";
33     }
34     //Chequear si hay datos desde USB
35     if (Serial.available()){
36         delay(10); // necesita un pequeño delay
37         Serial1.write(Serial.read());
38     }
39 }
```

Bluetooth maestro

Vamos a hacer una especial mención a este tipo de módulos.

Hemos comentado que las redes bluetooth se crean entre 2 dispositivos. Normalmente uno emite y el otro recibe, pero puede darse que los dos emitan y reciban. Para esto el módulo tiene que ser capaz de poder cambiar de modo master a slave. No todos los BT permiten hacer esto. Si compramos algún módulo económico para arduino, seguramente estaremos comprando un módulo SLAVE. Este sólo podrá recibir datos de otro dispositivo. Si queremos que nuestra arduino envíe datos deberemos usar un módulo MASTER.

El módulo master es físicamente igual que el esclavo , aunque incorpora un firmware distinto **HC-05** (firmware: las instrucciones que hacen que funcione al hardware). Otra diferencia es que lleva soldado al menos un pin más. Este pin llamado key, es necesario para que el módulo entre en modo de “comandos AT”, y así podamos programar su funcionamiento. Esto lo podemos hacer con el mismo código que te hemos mostrado en el punto anterior. Para acceder a este modo especial en el master lo podemos hacer de 2 formas:

1. Conectando Key a 3.3v y encender el módulo. Así funciona a 38400 bauds.
2. Encendiendo el módulo y después conectando el key a 3.3v. Así funciona a 9600 bauds, (es más sencillo pues es el que usa por defecto).

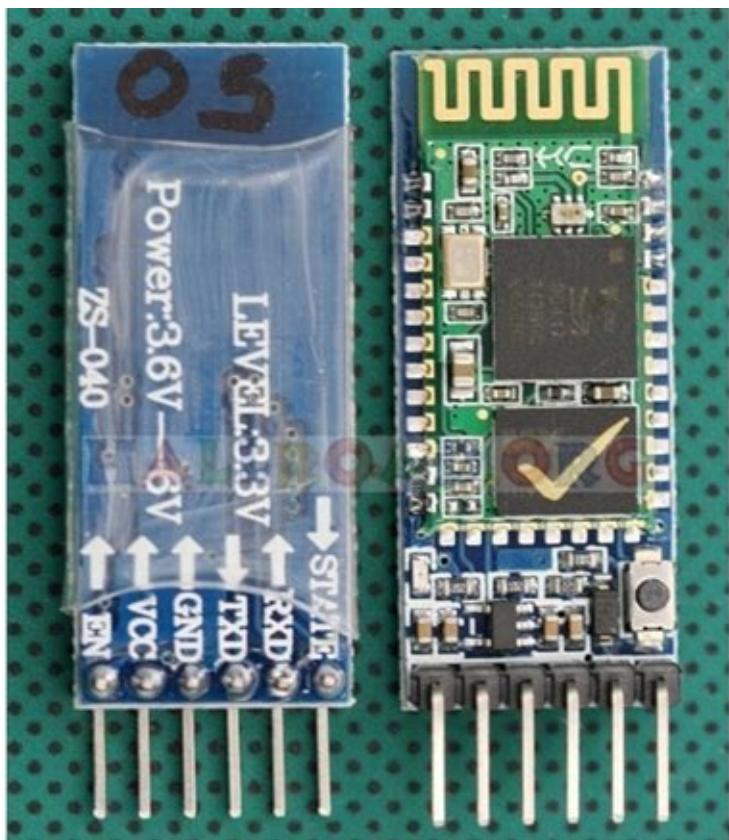
Los comandos AT en HC-05, al contrario que en el HC-06 (esclavo), que es el que tendrá mucha gente , tienen que llevar el símbolo "=", por ejemplo:

En HC-06: AT+NAM Enombre

En HC-05: AT+NAM E=nombre

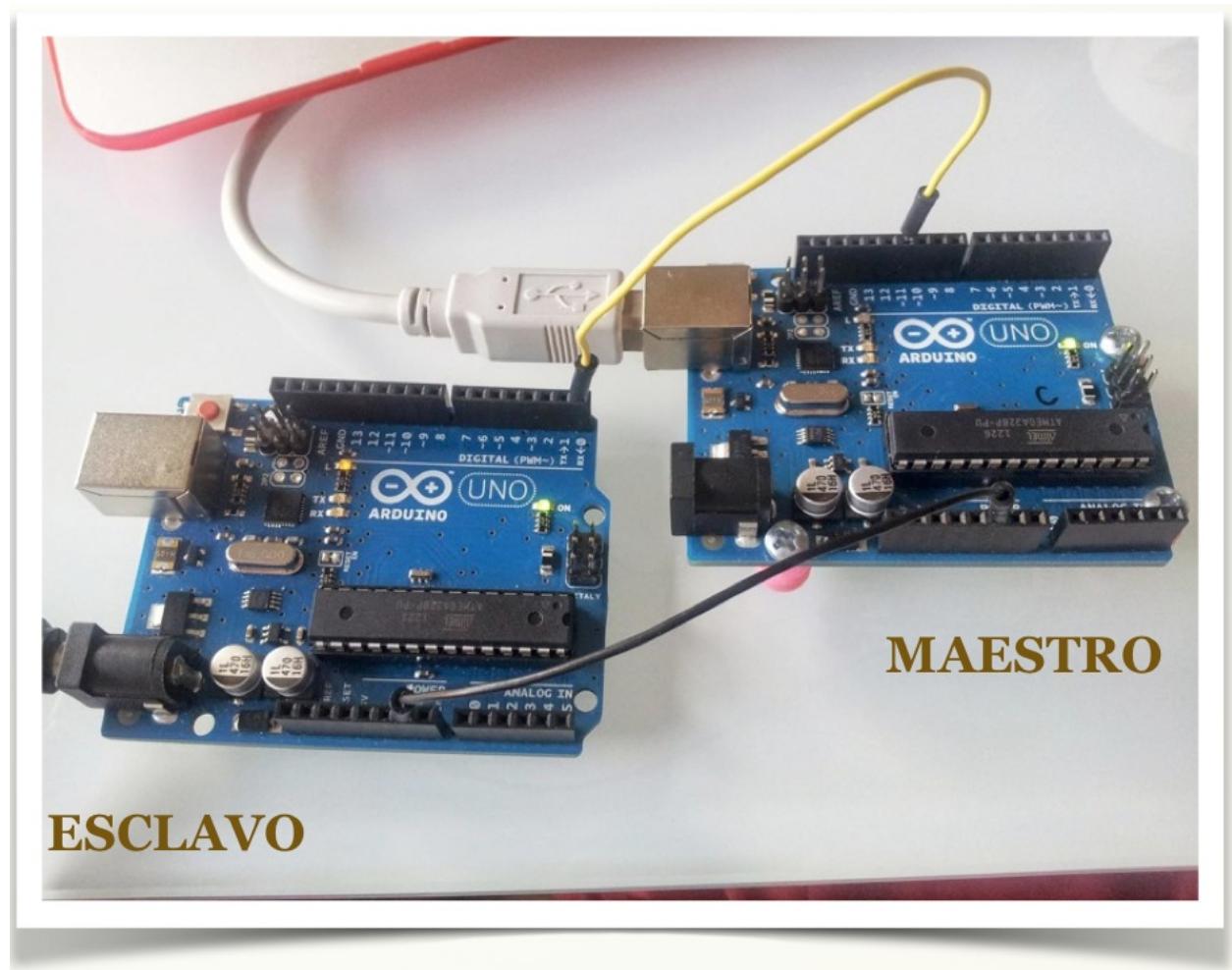
El datasheet indica que por defecto vienen con el modo CMODE=1 (para conectarse a cualquier esclavo disponible), sin embargo hay que comporbarlo (AT+CMODE?) por si tienen el CMODE=0 por lo que se intenta conectar al último emparejado, (en este caso no se emparejaría con ningún esclavo), así que hay que cambiar el CMODE con AT+CMODE=1)

El HC-05 tiene 6 pines:



HC-05

Conexión entre dos Arduinos por cable



Podemos conectar dos placas Arduino de distintas maneras: Bluetooth, Xbee, Ethernet, WIFI...

Pero la forma más sencilla es aprovechar la conexión para la comunicación serie que ya posee Arduino.

Conocimiento previo

- Programación básica de Arduino.
- Bucles **for**, sentencias if-else, switch-case.

Objetivos

- Comunicación serie.
- Configuración maestro / esclavo.
- Crear un nuevo puerto serie.

Lista de materiales:

- 2 placas Arduino.

SI NO DISPONES DE DOS PLACAS DE ARDUINO, TE PROPONEMOS SIMULARLO

Montaje 2: Conectar dos Arduinos

El proceso es parecido al bluetooth. Aquí queremos que una Arduino envíe (MAESTRO) y que otra reciba (ESCLAVO), así que cada una correrá un programa distinto.

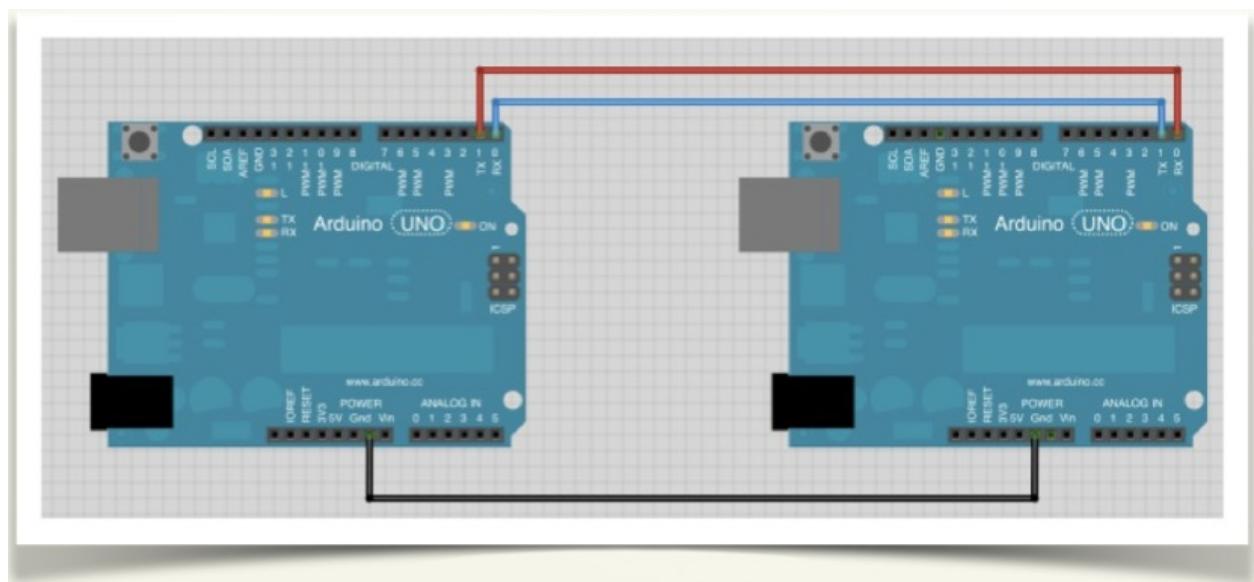
Podemos hacer también que ambas tarjetas envíen y reciban datos, para ello, la modificación sobre lo que expongo aquí serían muy sencillas, (básicamente copiar y pegar los trozos de código intercambiados).

Conecciones:

Usaremos los pines estándar de comunicación serie de Arduino:

- 0 : RX (pin por el que RECIBE los datos serie)
- 1 : TX (pin por el que ENVÍA los datos serie)

Para comunicación en 2 direcciones: los 2 pueden enviar / recibir. Las conexiones TX/RX se intercambian (lo que uno envía -TX- tiene que entrar en el otro -RX-). Cualquiera de las 2 puede ser Maestro o Esclavo. **IMPORTANTE:** Conectar ambas GND de las placas.



En el siguiente ejemplo , el maestro, cada 3 segundos envía un carácter al esclavo.

- Si envía una "r", el esclavo hará parpadear su led (d13) rápido.
- Si envía una "l", el esclavo hará parpadear su led (d13) lento.

El programa para el Arduino MAESTRO es:

```

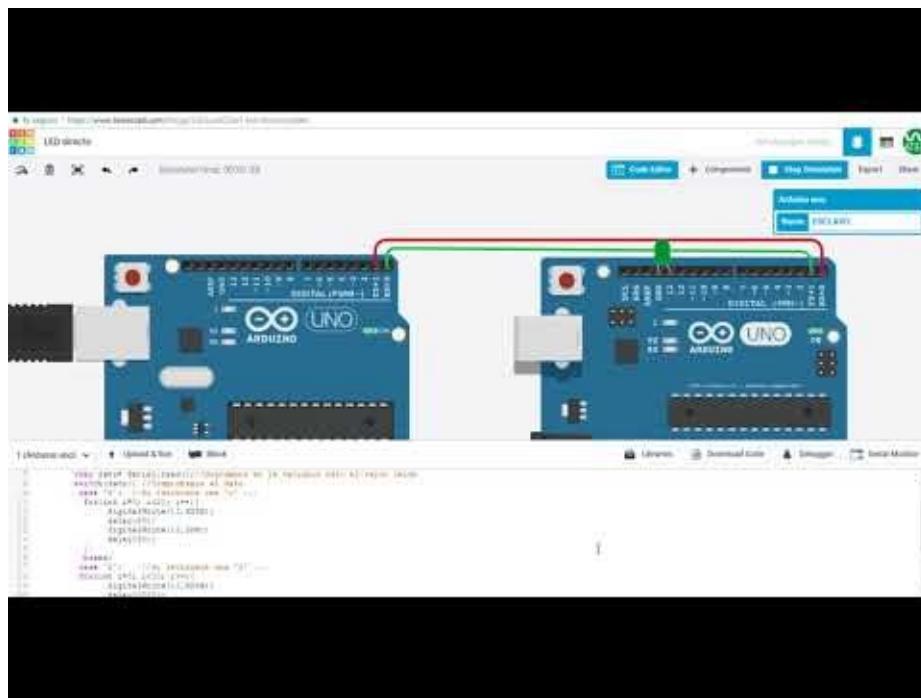
1 //////////////// ARDUINO MAESTRO ///////////
2 void setup(){
3     Serial.begin(9600);
4 }
5 void loop()
6 {
7     Serial.write("r");
8     delay(3000);
9     Serial.write("l");
10    delay(3000);
11 }
```

El programa para el Arduino ESCLAVO es:

```
1 ////////////// ARDUINO ESCLAVO //////////
2 void setup(){
3     pinMode(13,OUTPUT);
4     Serial.begin(9600);
5 }
6 void loop(){
7     char dato= Serial.read(); //Guardamos en la variable dato el valor leido
8     switch(dato){ //Comprobamos el dato
9         case 'r': //Si recibimos una 'r' ...
10            for(int i=0; i<20; i++){
11                digitalWrite(13,HIGH);
12                delay(80);
13                digitalWrite(13,LOW);
14                delay(80);
15            }
16            break;
17         case 'l': //si recibimos una 'l' ...
18            for(int i=0; i<10; i++){
19                digitalWrite(13,HIGH);
20                delay(1000);
21                digitalWrite(13,LOW);
22                delay(1000);
23            }
24            break;
25         default:
26             digitalWrite(13,LOW);
27             break;
28     }
29 }
```

Si no tienes dos ARDUINOS

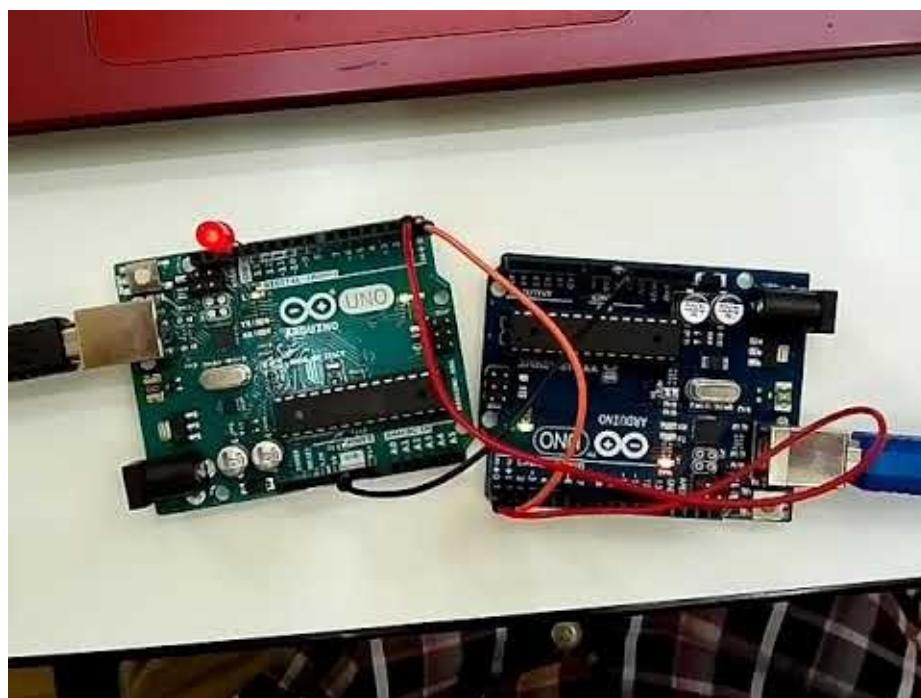
Puedes hacerlo con una simulación en <https://www.tinkercad.com> en nuestro caso este fue el resultado:



[Video link](#)

Si tienes dos ARDUINOS

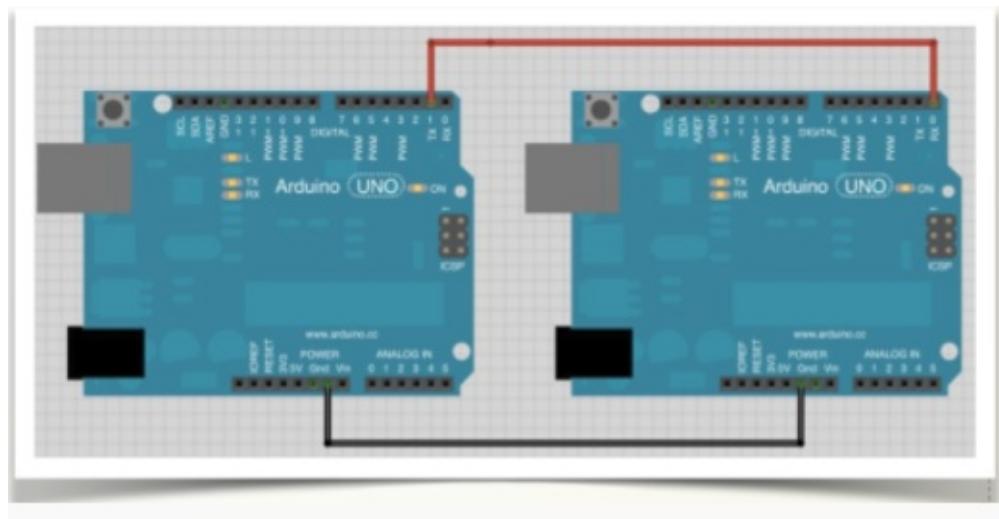
Pues a disfrutar de tu "Red particular" :



[Video link](#)

Otras conexiones

Para la comunicación en una dirección:



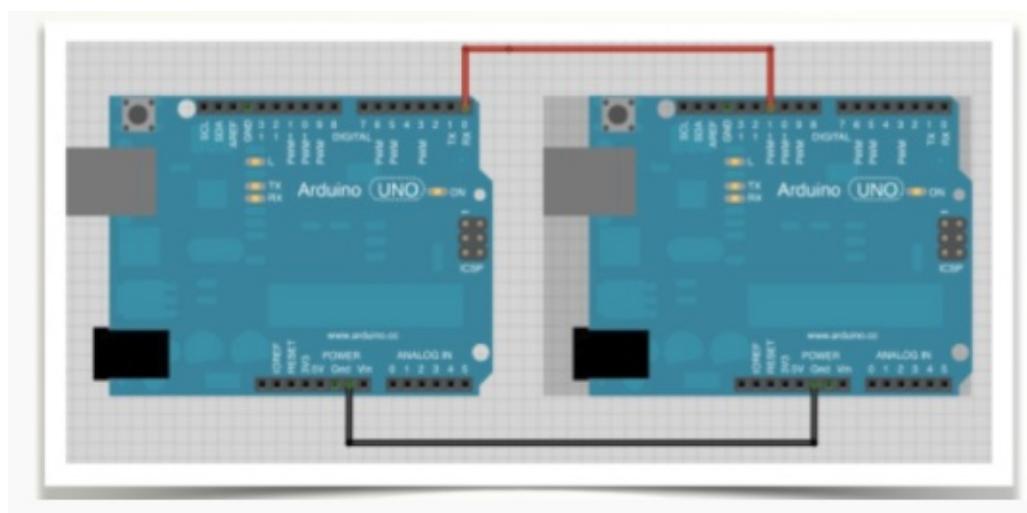
maestro(izquierda) -> esclavo(derecha) sólo necesitamos 1 conexión:

NOTA:

Ocurre que esos pines también los usa para comunicarse por USB cuando está conectado al ordenador, de manera que si queremos tener ambas conexiones (USB/trasmisión serie) deberemos crear una nueva conexión serie (en una conexión software). Sólo podemos conectar 2 Arduinos pues sólo hay un puerto de serie en cada uno de ellos. Aunque la conexión es en un sentido, es necesario conectar los dos cables TX-RX y RX-TX

En este ejemplo, una de las Arduino la vamos a tener conectada al PC, por tanto, en el MAESTRO vamos a crear la conexión software serie sobre los pines 10(RX), 11(TX).

Lo puedes comprobar en la siguiente imagen:



El programa sería el siguiente:

```

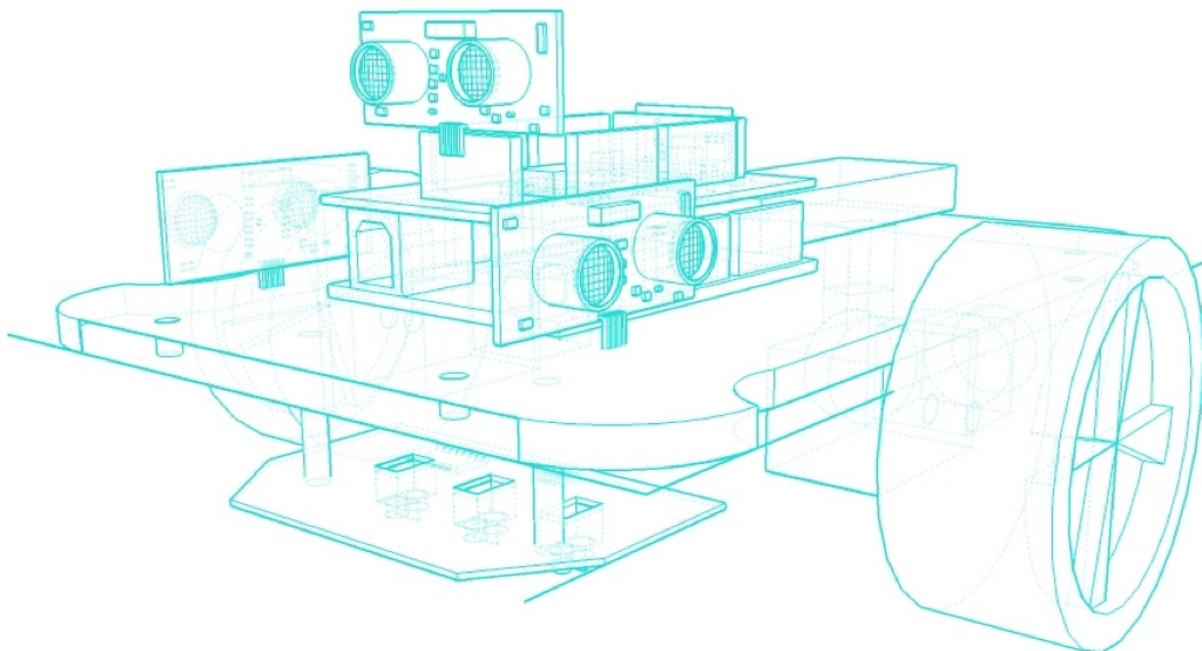
1 ///// MAESTRO
2 int i=0;
3 //CREAMOS UN NUEVO PUERTO SERIE (RX, TX)
4 SoftwareSerial Serie2(10,11);
5

```

```
6 void setup()
7 { pinMode(13,OUTPUT);
8   Serial.begin(9600);           //Inicializa puerto estándar
9   Serie2.begin(9600);          //Inicializa nuevo puerto
10  digitalWrite(13,LOW);
11 }
12 void loop()
13 { Serie2.write("r");
14   delay(3000);
15   Serie2.write("1");
16   delay(3000);
17 }
18 ////////////////////esclavo
19 void setup()
20 {
21 { pinMode(13,OUTPUT);
22   Serial.begin(9600);
23 }
24 void loop()
25
26 { while (Serial.available())
27 {
28   //Guardamos en la variable dato el valor leido
29   char dato= Serial.read();
30   //Comprobamos el dato
31   switch(dato)
32   { //Si recibimos una 'r' ...
33     case 'r':
34     {for(int i=0; i<20 i++)
35      digitalWrite(13,HIGH);
36      delay(80);
37      digitalWrite(13,LOW);
38      delay(80);}
39     break;
40   }
41   case 'l':
42   {for(int i=0; i<10 i++)
43     digitalWrite(13,HIGH);
44     delay(200);
45     digitalWrite(13,LOW);
46     delay(200);}
47   break;
48 }}}}
```

Robótica

En esta capítulo vamos a dar movimiento a nuestro Arduino.



Si tienes dudas técnicas en este capítulo pon un ticket a <http://soporte.catedu.es/> y te ayudaremos:

 [CATEDU](#)

Usuario Invitado | Iniciar sesión

[Inicio Centro de Soporte](#) [Buscar en nuestra base de conocimiento](#) [Abrir un nuevo Ticket](#) [Ver Estado de un Ticket](#)

Buscar en nuestra base de conocimiento [Buscar](#)

[Abrir un nuevo Ticket](#) [Ver Estado de un Ticket](#)

Preguntas destacadas

- # Problema técnico en mi centro
- # Dos o más cursos on-line coinciden
- # Programa de Dietas, Gestión o Inventario FryC
- # ¿Cómo recibo el certificado de mis cursos de **AULARAGON7**, debo de solicitarlos de algún modo?

Bienvenido a la web de **CATEDU**

En este sitio web podrás solicitar soporte técnico para tu centro, crear un ticket y consultar el estado de los mismos. Tendrás un sistema de tickets.

- **Para solicitar soporte**: Si necesitas ayuda técnica, solo debes hacer clic en el botón "Abrir un nuevo ticket". Se asignará un número de ticket y podrás seguir su seguimiento. Se recomienda que utilices el navegador Chrome o Firefox.
- **Para solicitar servicios**: Si necesitas información sobre los servicios que ofrecemos, puedes hacer clic en el botón "Ver Estado de un Ticket". Si tienes una cuenta de usuario registrada, podrás acceder a la sección "Mis Tickets" para consultar el estado de los tickets que has abierto.

Si requiere un entorno de trabajo seguro, por favor, utilice el correo electrónico soporte.catedu@edu.aragon.es.

Control de servomotores



Una de las aplicaciones más utilizadas de los sistemas de control por ordenador y en la robótica están asociados con los motores, que permiten accionar o mover otros componentes, como puertas, barreras, válvulas, ruedas, etc. Uno de los tipos que vamos a ver en este capítulo es el servomotor que posee la capacidad de posicionar su eje en un ángulo determinado entre 0 y 180 grados en función de una determinada señal.

Servos de rotación continua

Son servos por fuera igual que los anteriores, pero pueden girar 360º y se controlan por tiempo, es decir `servoRotCont.write(90);`

Si quieres saber más sobre servomotores te recomendamos estas páginas del Zaragozano Luis LLamas:

- [Servomotores convencionales](#)
- [Servomotores de rotación continua](#)

Recomendamos testear tus servos en el [Montaje 1](#).

Conocimiento previo

- Programación básica de Arduino.
- Uso de bibliotecas internas de Arduino (`Servo.h`).

- Bucles **for**.

Objetivos

- Conocer qué es un servomotor, tipos y funcionamiento.
- Las aplicaciones en el mundo de la automática y el control por ordenador de este tipo de motores.
- Usar la librería que incorpora Arduino para controlar los servos.

Lista de materiales:

- Servomotor.
- Placa Arduino UNO.
- Placa EduBásica (opcional).

Montaje 1: Testea tu servo

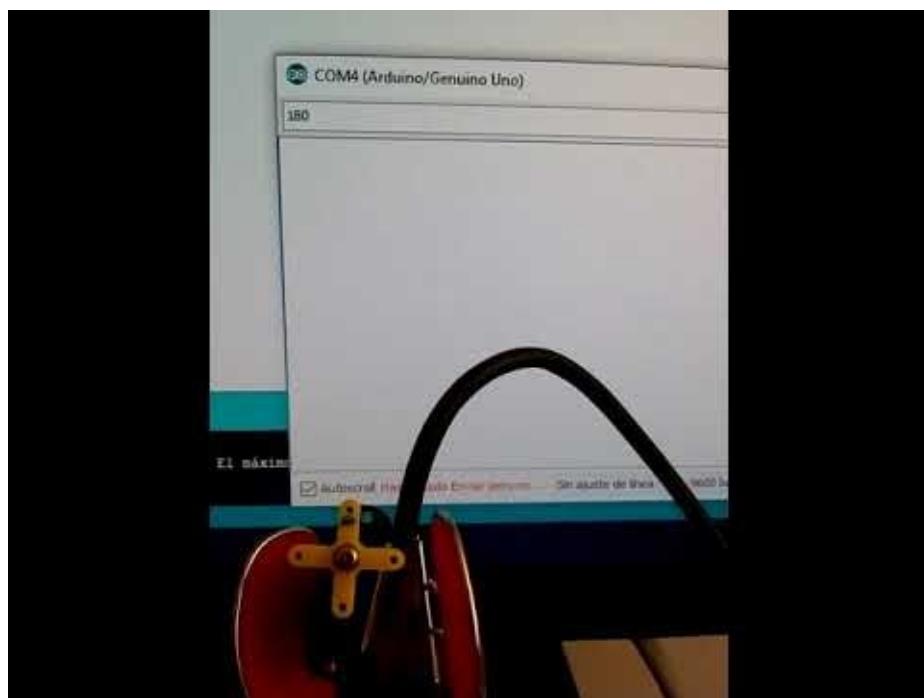
En el siguiente programa de testeo en el [obtenido del forum arduino](#) que lo tienes más abajo, puedes probar tu servo.

- Conecta el servo al pin 7
- Utiliza el puerto serie para teclear el ángulo que quieras con el teclado de tu ordenador.
- No queremos que entiendas todo el código, pues el puerto serie lee es caracteres ASCII y tiene que convertir el carácter a ángulos.
- Si tecleas un valor más grande de 500 se le indica al servo no el ángulo que se tiene que mover, sino cuanto tiempo en ms se tiene que mover.

Por ejemplo en este Servo **HD-1440A** con el anterior programa se ve que es un servo barato:

- No puede hacer ángulos de +180° luego es un servo convencional
- No puede hacer ángulos de menos de 10° no llega a parar, o sea tiene deriva.

El resultado se puede ver aquí:



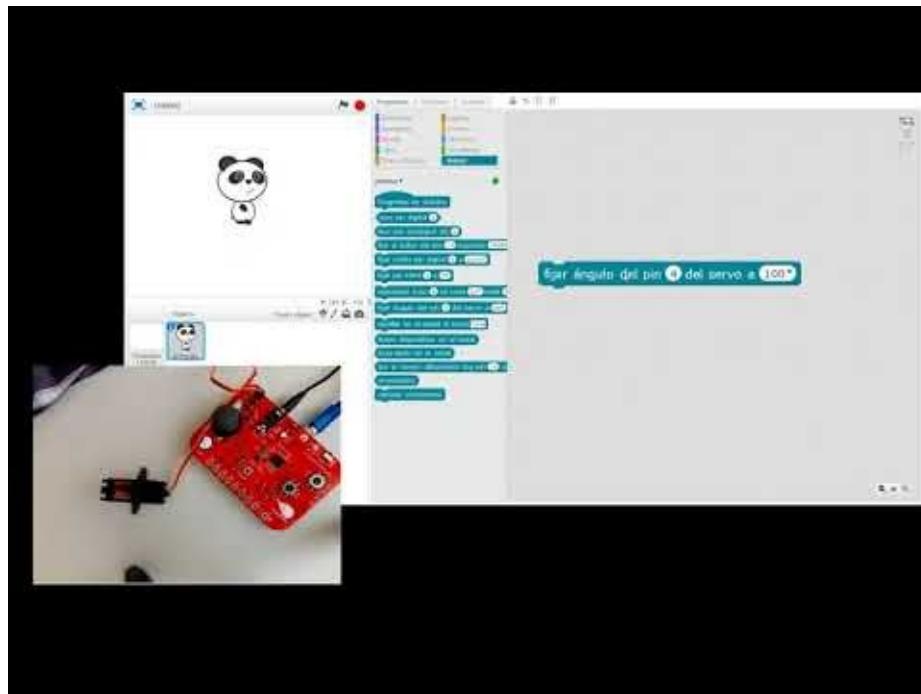
[Video link](#)

Si eliges uno un poco más caro como el **MG90S** no tiene estos problemas en los extremos. [Ver](#)

Mira la diferencia con un **servo de rotación continua**, fíjate como:

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

El vídeo está realizado con otra Shield: [Echidna](#).



[Video link](#)

Programa

```

1 // zoomkat 10-22-11 serial servo test
2 // type servo position 0 to 180 in serial monitor
3 // or for writeMicroseconds, use a value like 1500
4 // for IDE 0022 and later
5 // Powering a servo from the arduino usually *DOES NOT WORK*.
6
7 String readString;
8 #include <Servo.h>
9 Servo myservo; // create servo object to control a servo
10
11 void setup() {
12   Serial.begin(9600);
13   myservo.writeMicroseconds(1500); //set initial servo position if desired
14   myservo.attach(7); //the pin for the servo control
15   Serial.println("servo-test-22-dual-input"); // so I can keep track of what is loaded
16 }
17
18 void loop() {
19   while (Serial.available()) {
20     char c = Serial.read(); //gets one byte from serial buffer
21     readString += c; //makes the string readString
22     delay(2); //slow looping to allow buffer to fill with next character
23   }
24
25
26   if (readString.length() >0) {
27     Serial.println(readString); //so you can see the captured string
28     int n = readString.toInt(); //convert readString into a number

```

```
29 // auto select appropriate value, copied from someone elses code.
30 if(n >= 500)
31 {
32     Serial.print("writing Microseconds: ");
33     Serial.println(n);
34     myservo.writeMicroseconds(n);
35 }
36 else
37 {
38     Serial.print("writing Angle: ");
39     Serial.println(n);
40     myservo.write(n);
41 }
42
43     readString=""; //empty for next input
44 }
45 }
```

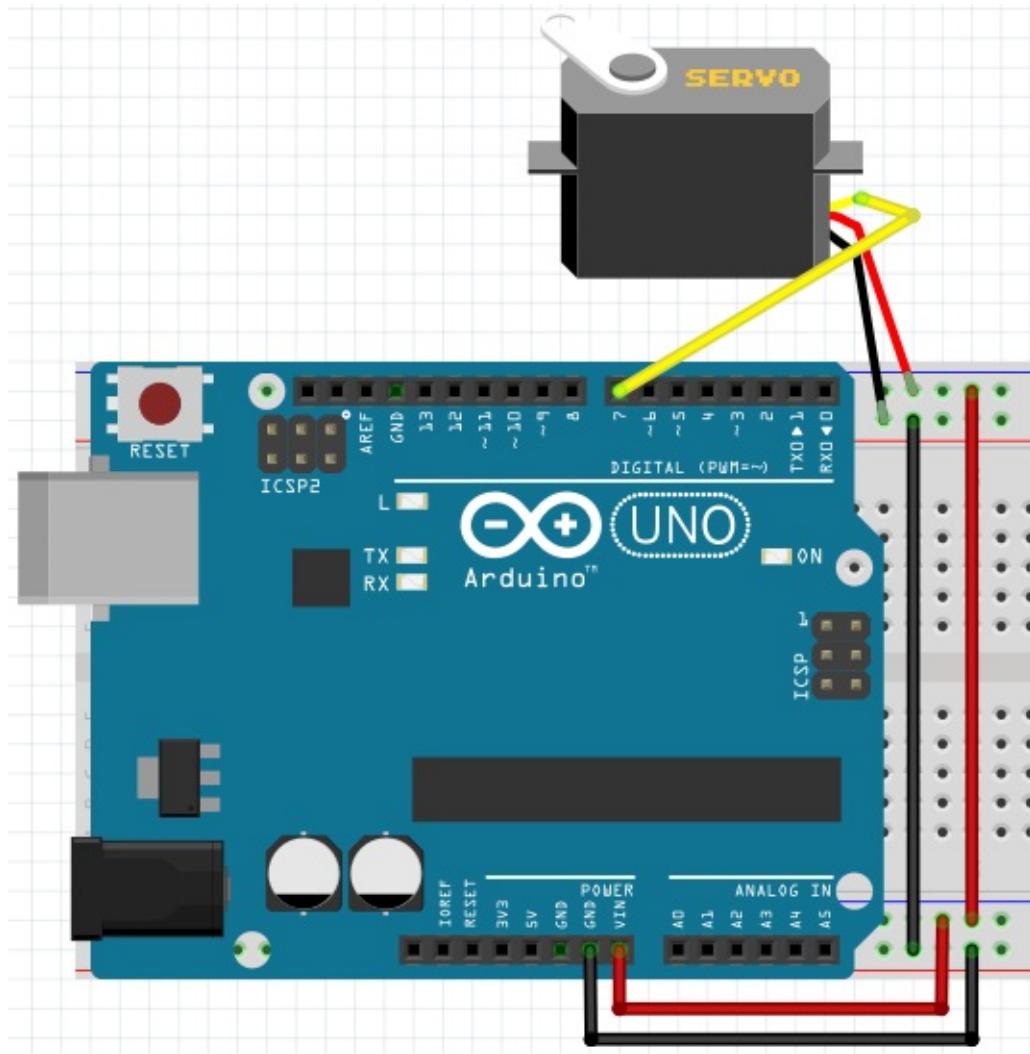
Descripción y esquemas



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

El hardware interno se compone de un potenciómetro y un circuito integrado que controlan en todo momento los grados que gira el motor. De este modo, en nuestro caso, desde Arduino, usando las salidas digitales PWM podremos controlar fácilmente un servo.

CONEXION SIN EDUBÁSICA



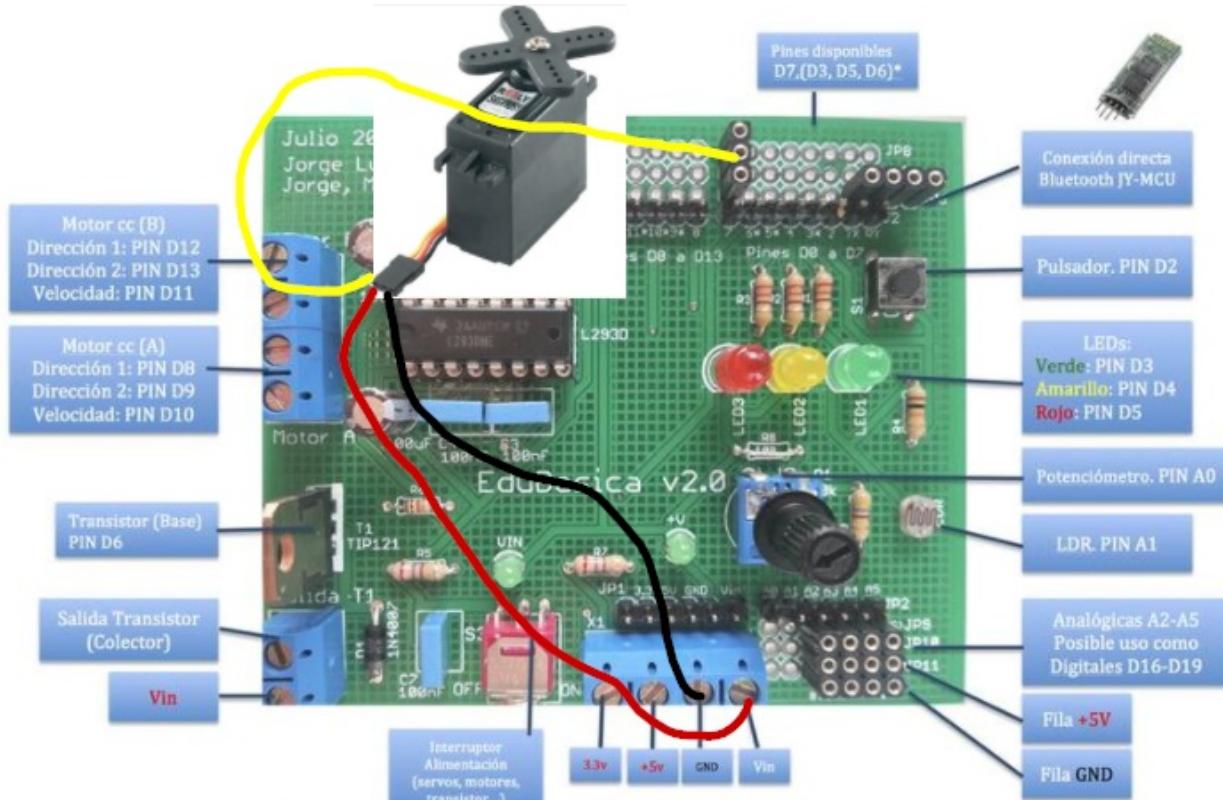
La conexión se realiza mediante 3 cables: 2 de alimentación (+5V/GND) y un tercero, conectado por ejemplo el 7 , donde indicaremos los grados que queremos que gire a través de un programa en Arduino.

CONEXIÓN CON EDUBÁSICA

En Edubásica tenemos una forma muy sencilla de conectar un servo a la tarjeta. Lo puedes hacer mediante las clavijas identificadas con JP3.

De arriba abajo las conexiones son:

- Señal (pin7)
- +Vin
- GND



Recuerda que siempre puedes utilizar los pines analógicos como E/S digitales, del pin 14 al 19.

Por ejemplo, puedes conectar el servo al pin analógico 5, pero declarado como digital en el 19.

Montaje 2 controlando el servo

Arduino incluye una librería con funciones para mover de una manera sencilla un servo, lo primero es introducir un include con la librería al principio, luego creamos el objeto (que será el nombre que usaremos en todo el programa para cada servo que queramos controlar) .

Por último, asociamos el servo al pin al que lo hemos conectado (7).

Una vez declarado, para usarlo, dentro de loop simplemente usamos la función **servo.write(posicion)**

Moverá el servo los grados que le indiquemos mediante la variable entera: **posicion**.

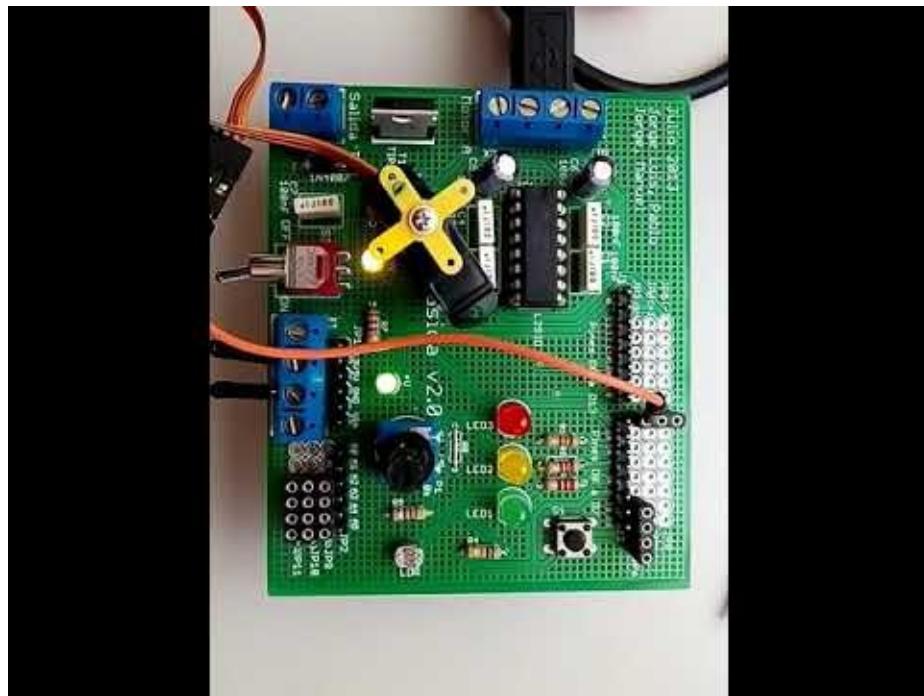
A continuación verás un sencillo ejemplo , dejamos un pequeño delay , para permitir que el servo alcance la posición antes de darle la siguiente orden:

```
1 #include <Servo.h>
2
3 Servo myservo;           // crea un objeto tipo servo para controlar el servo
4 int pos;                 // variable para almacenar la posición del servo
5
6 void setup(){
7     myservo.attach(7);   // En EduBasica el servo se conecta al pin 7
8 }
9
10 void loop()
11 {
12     pos=90;
13     myservo.write(pos);
14     delay(1000);
15     pos=180;
16     myservo.write(pos);
17     delay(1000);
18     pos=45;
19     myservo.write(pos);
20     delay(1000);
21 }
```

Montaje 3 servo paso a paso

En este caso queremos que practiques la función for

```
1 #include <Servo.h>
2
3 Servo myservo;           // crea un objeto tipo servo para controlar el servo
4 int pos;                 // variable para almacenar la posición del servo
5
6 void setup(){
7     myservo.attach(7);   // En EduBasica el servo se conecta al pin 7
8 }
9
10 void loop()
11 {
12     for (pos=10; pos<=180; pos+=10) {
13         myservo.write(pos); //orden con los grados en cada iteración
14         delay(1000);
15     }
16 }
```



[Video link](#)

Montaje 4 servo y potenciómetro

Actividad

Podemos probar una aplicación muy importante que está basada en mover el servo según una determinada entrada analógica. Este nos puede ser muy útil si queremos controlar servos por medio de joysticks por ejemplo o cualquier dispositivo que cuente con potenciómetros para realizar un movimiento.

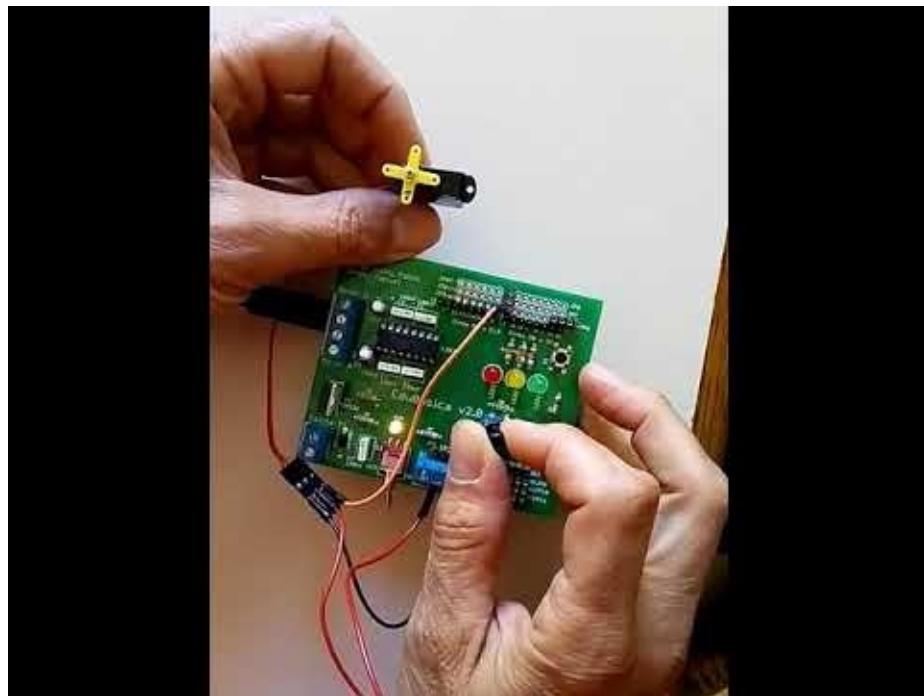
Este está obtenido desde los ejemplos que vienen incluido en la IDE de Arduino (Knob) que encontrarás en: Archivo->Ejemplos->Servo, sólo hemos cambiado esta línea: **myservo.attach(7);**

Lo que hace este programa es variar la posición del servo en función de la posición del potenciómetro que leemos de manera analógica.

Sólo nos queda mapear la lectura para que se mueva de 0 a 180°.

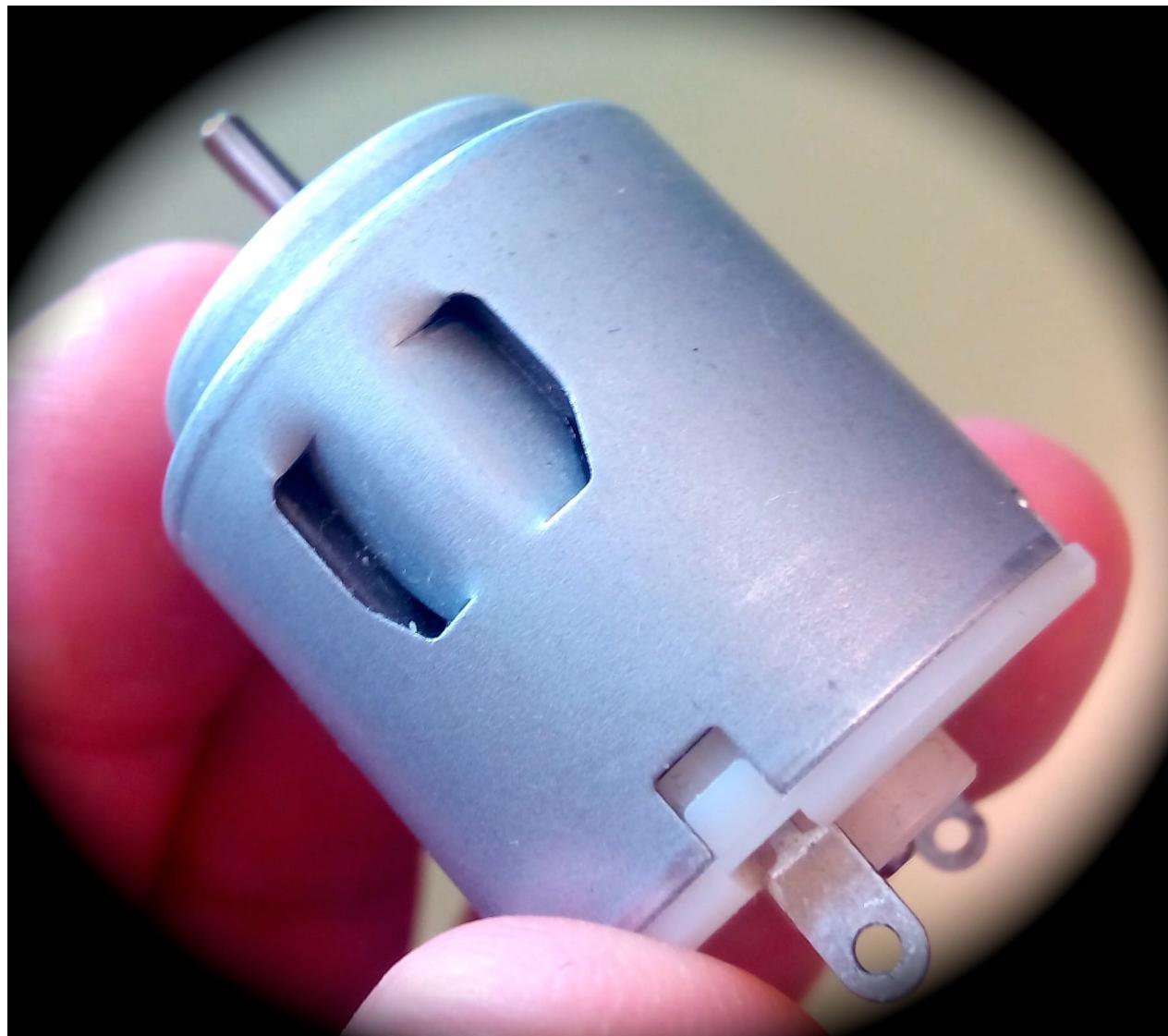
```

1  /*
2   * Controlling a servo position using a potentiometer (variable resistor)
3   * by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>
4
5   * modified on 8 Nov 2013
6   * by Scott Fitzgerald
7   * http://www.arduino.cc/en/Tutorial/Knob
8   */
9
10 #include <Servo.h>
11
12 Servo myservo; // create servo object to control a servo
13
14 int potpin = 0; // analog pin used to connect the potentiometer
15 int val; // variable to read the value from the analog pin
16
17 void setup() {
18     myservo.attach(7); // attaches the servo on pin 9 (bueno ahora 7) to the
19     servo object
20 }
21
22 void loop() {
23     val = analogRead(potpin); // reads the value of the potentiometer
24     // (value between 0 and 1023)
25     val = map(val, 0, 1023, 0, 180); // scale it to use it with the servo
26     // (value between 0 and 180)
27     myservo.write(val); // sets the servo position according
28     // to the scaled value
29     delay(15); // waits for the servo to get there
30 }
```



[Video link](#)

Control de motores DC



Arduino (UNO) no proporciona corriente suficiente para hacer funcionar, en condiciones "dignas", un motor de corriente continua de los que solemos usar en el aula-taller de Tecnologías (motor DC). Conviene alimentar Arduino con una fuente externa (7 -12 V) para poder proporcionar intensidad necesaria para el par motor requerido en los proyectos. Para más información de cómo hacerlo se recomienda consultar el apartado (Alimentación eléctrica de Arduino). Existen "shields" o tarjetas que se encajan sobre Arduino y le añaden funciones específicas como mover motores DC.

Alimentación

Como vimos en la sección de Alimentación eléctrica de Arduino no es recomendable alimentar Arduino, cuando se trabaja con elementos de "alto" consumo como pueden ser los motores DC, con el cable USB. Tenemos la posibilidad de proporcionar más (mili)amperios a través de la conexión jack de Arduino. En el pin Vin tendremos una salida del voltaje que aplicaremos por el jack que servirá para alimentar a los motores a través del integrado mediante el pin 8 (VC ó Vcc2). Para ver los márgenes de voltaje de trabajo del CI consultar la hoja de datos del fabricante.

Si el motor es pequeño y trabaja sin carga, sí que se puede.

Recomendamos las siguientes páginas de Luis Llamas:

- [Tipo de motores](#)
- [Motores paso a paso](#)

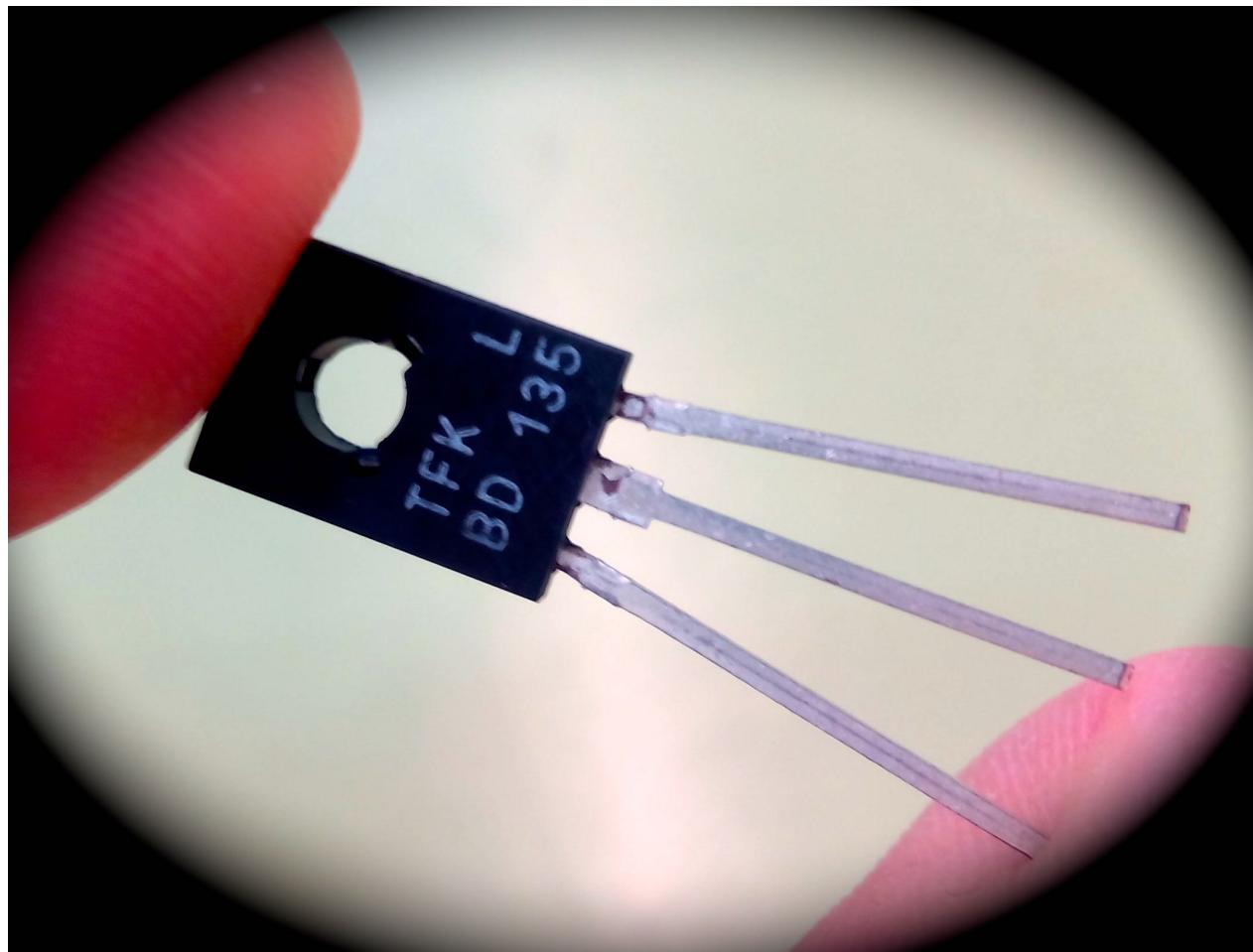
Para obtener más información sobre estas placas se puede consultar la web oficial de Arduino:

<http://www.arduino.cc/en/Main/ArduinoMotorShieldR3>

Además de utilizar una shield específica de motores DC, tenemos otras posibilidades para ponerlo en marcha:

- [Montaje con transistor.](#)
- [Conexión con circuito integrado L293](#)
- [Shield Edubásica.](#)

Montaje con transistor



El transistor es un componente analógico con infinidad de aplicaciones. Como se observa en la imagen, es un dispositivo con tres pines llamados emisor (E), colector (C) y base (B). El funcionamiento a grosso modo es sencillo: Si suministramos "cierta" cantidad de corriente al terminal llamado base, entre los terminales emisor y colector circulará una corriente proporcional a la que entra por la base. Puede ser unas 100 veces mayor (este valor dependerá del transistor y se llama ganancia), por lo tanto, lo que tenemos es un amplificador de corriente, o sea, que con una intensidad pequeña podemos obtener una más grande entre los terminales emisor y colector. Si a estos dos terminales conectamos un motor DC conseguiremos hacerlo girar con garantías. Cuando no hay corriente en la base (o es muy pequeña) el colector y emisor están desconectados y no circulará corriente entre ellos.

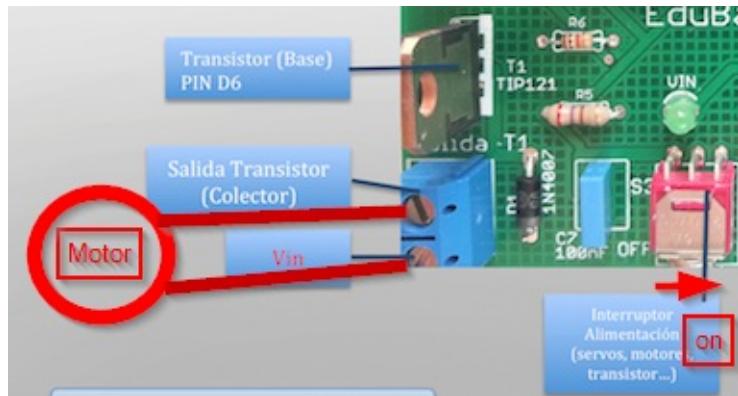
En el tema de Electrónica Analógica - Transistores de este curso se puede encontrar más información sobre este componente electrónico.

Descripción y esquemas

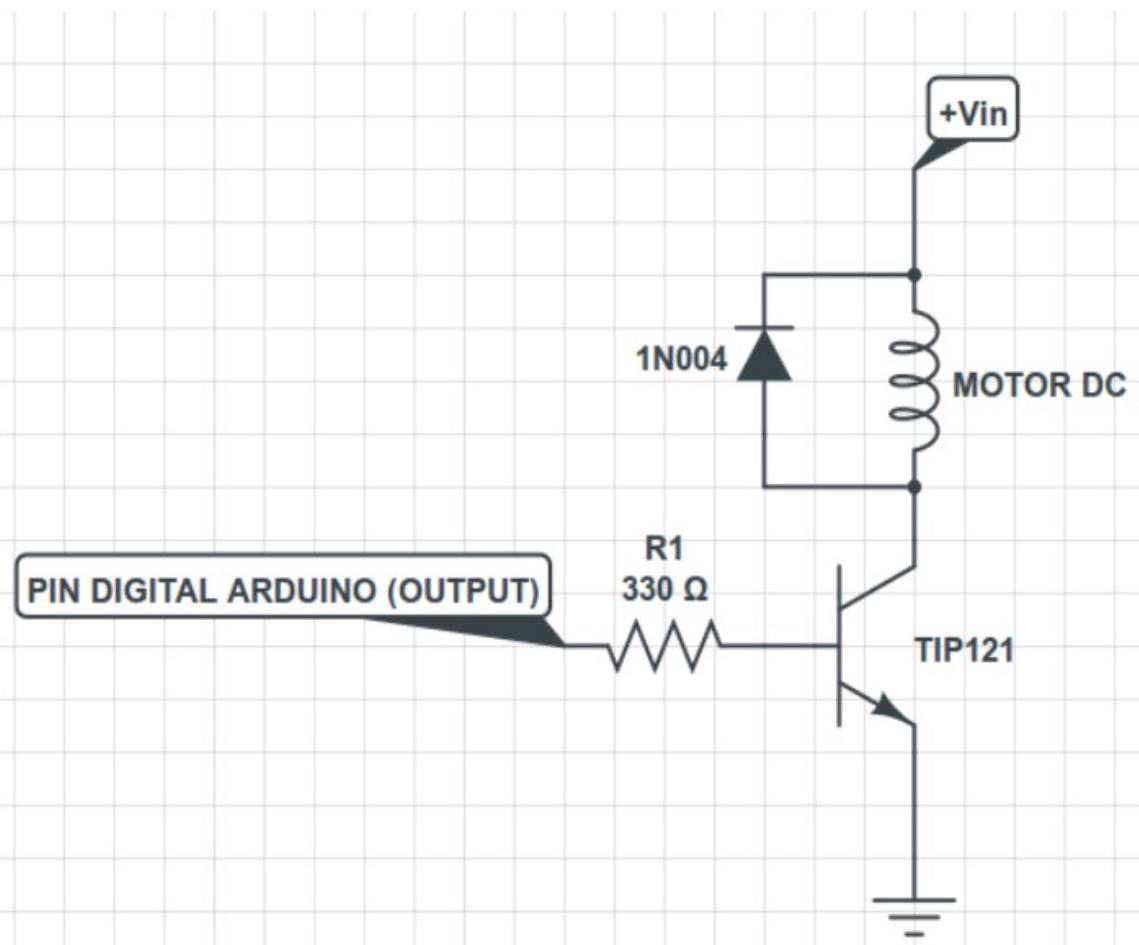
CON EDUBÁSICA

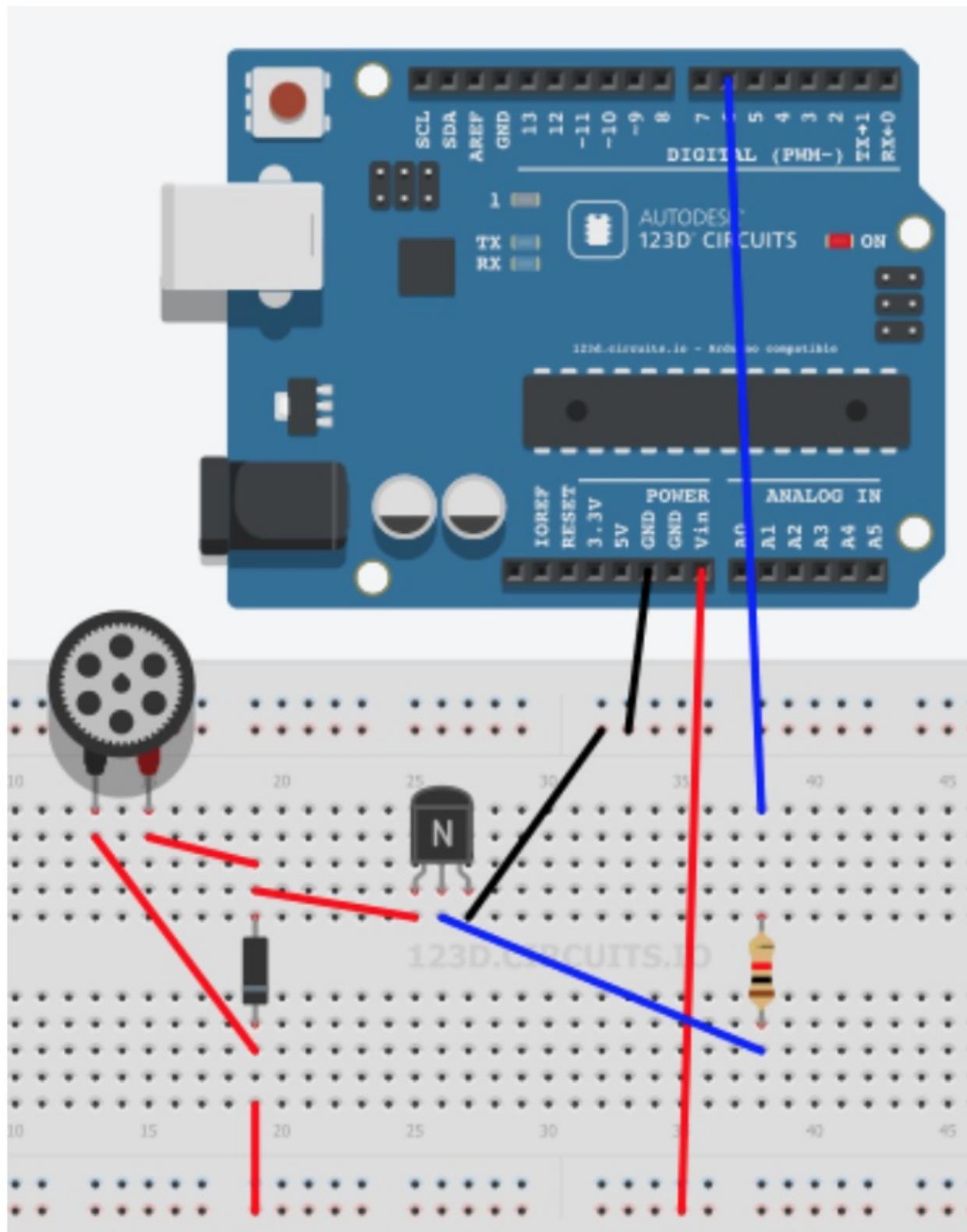
La shield de Edubásica ya tiene incorporado el transistor y los diodos de protección para funcionar el motor, para ello tenemos que hacer:

- Poner el interruptor en ON
- Conectar el motor en la salida del transistor y Vin
- Utilizar D6 como pin de control del motor



SIN EDUBASICA





Desde Arduino (salida digital) actuamos sobre la base si enviamos un HIGH al pin digital donde la conectemos (pin 6 en el esquema). El transistor tiene que tener suficiente ganancia. Se conecta un diodo de protección en antiparalelo (1N004). Cuando el motor se para las bobinas se desmagnetizan y se descargan de energía eléctrica. El diodo proporciona un camino para su descarga (la energía se disipa en forma de calor en el diodo) y así se evita que sufra el transistor.

Otra posibilidad que protege el transistor es, en vez de conectar directamente el motor al colector del transistor, ubicar un relé en esa posición y accionar el motor con alimentación independiente con la comutación del relé. Este ejercicio se propone en la sección de actividades.

Ejemplo de montaje con TIP121 (NPN, par Darlington *):

- Datasheet: <https://www.fairchildsemi.com/datasheets/TI/TIP120.pdf>

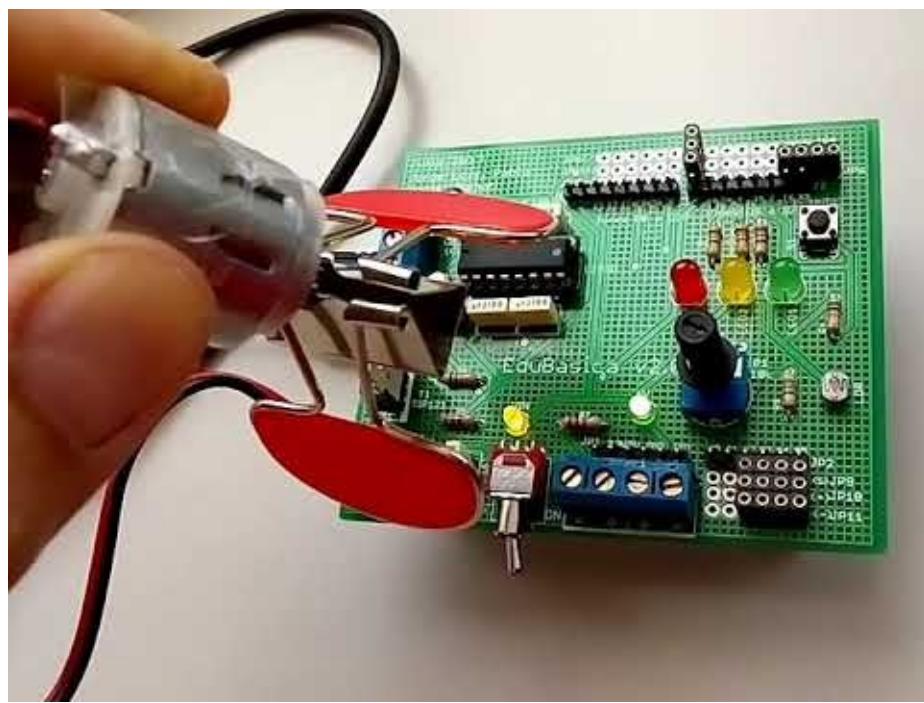
Montaje 5 Motor con transistor

Monta el motor con el Arduino utilizando Edubásica o utilizando un transistor tal y como se ha explicado en las páginas anteriores. NO CONECTES UN MOTOR DIRÉCTAMENTE A LOS PINES DEL ARDUINO puedes dañarlo.

Este es el programa que te proponemos:

```

1 // Activa y desactiva un
2 // motorDC conectado al pin6
3 // durante 2 segundos
4
5 #define motorDC 6
6
7 void setup() {
8
9     pinMode(motorDC, OUTPUT);
10
11 }
12
13 void loop() {
14
15     digitalWrite(motorDC, HIGH);
16     delay(2000);
17     digitalWrite(motorDC, LOW);
18     delay(2000);
19 }
```



[Video link](#)

¿Te atreves?

Diseñar un montaje que active el motor DC mediante un relé conectado al colector del transistor.

Montaje con Circuito L293

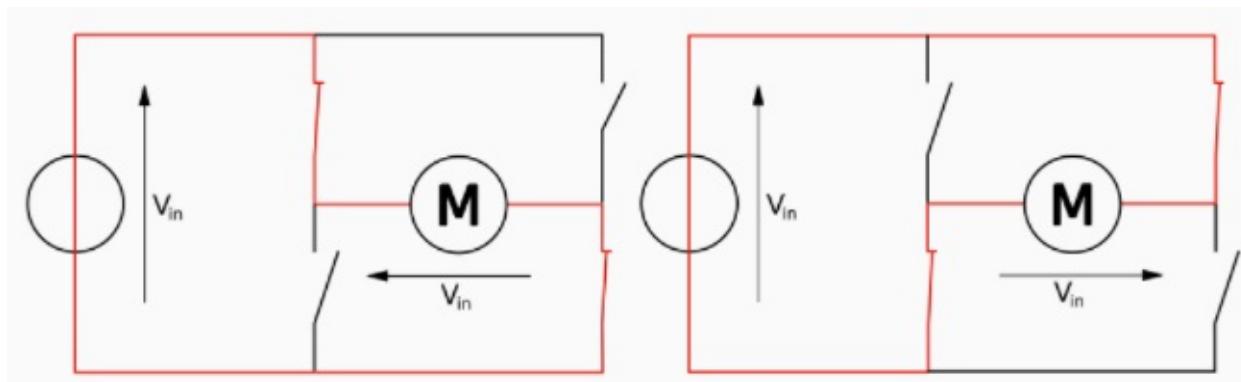
El circuito integrado L293 permite controlar motores DC de pequeña potencia, pero en general bobinas, o sea cualquier elemento que precise pico de potencia, como los relés, etc.. Para utilizarlo hay que hacer un montaje externo a Arduino, en una placa de pruebas, y alimentar a los motores a través de este circuito integrado.

El CI L293 tiene las siguientes características:

- Se pueden controlar hasta 2 motores.
- Proporciona 1A a los motores (en total) y permite cambiar el sentido de giro.
- Utiliza un puente en H que funciona según se observa en las figuras (internamente utiliza transistores para commutar*) :



Modos de operación para invertir el sentido de giro:



"H bridge operating" by Cyril BUTTAY - own work, made using inkscape. Licensed under CC BY-SA 3.0 via Wikimedia Commons.

*Datasheet: <http://www.me.umn.edu/courses/me2011/arduino/technotes/dcmotors/L293/L293.pdf>

Con Edubásica

Edubásica es una tarjeta diseñada para facilitar la tarea en el aula. Elimina gran parte de cableado y conexiones en la placa de pruebas lo que evita muchos errores en las prácticas.

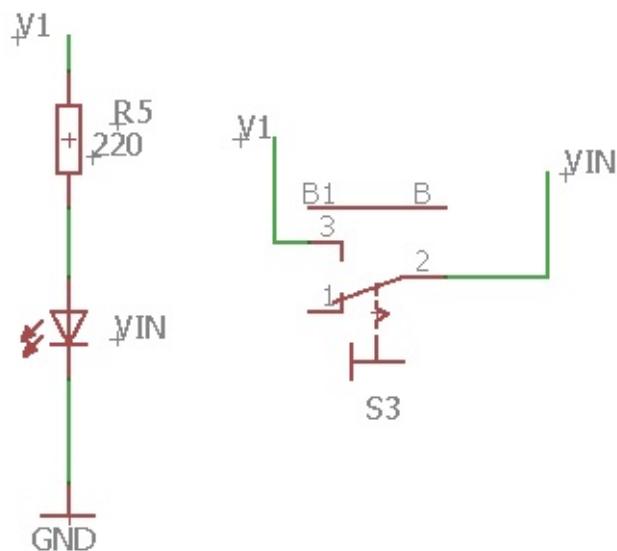
En esta placa disponemos de las dos opciones vistas anteriormente. Edubásica lleva montados, entre otros componentes, un circuito integrado L293 y un transistor. Por lo tanto podemos activar motores **usando ambas opciones**, aunque lo recomendable es utilizar el L293 que permite el cambio de sentido de giro y regular la velocidad actuando con una señal PWM sobre los dos pines de habilitación (dependiendo de la hoja de datos viene como ENABLE o CHIP INHIBIT) del circuito L293. Los pines de Arduino que pueden regular la velocidad por PWM correspondientes a esas patillas de habilitación serán: D10 para el motor A y D11 para el motor B.

Para hacer funcionar dos motores DC con Edubásica sólo tenemos que conectar en las clemas indicadas (serigrafiados en la placa como Motor A y Motor B) los dos cables de cada motor. Según se observa en la imagen, tenemos 4 conexiones para los dos motores. Desde Arduino y con la [tabla de verdad](#) del CI L293 indicada en la sección anterior, podemos regular el sentido de giro y velocidad de cada motor.

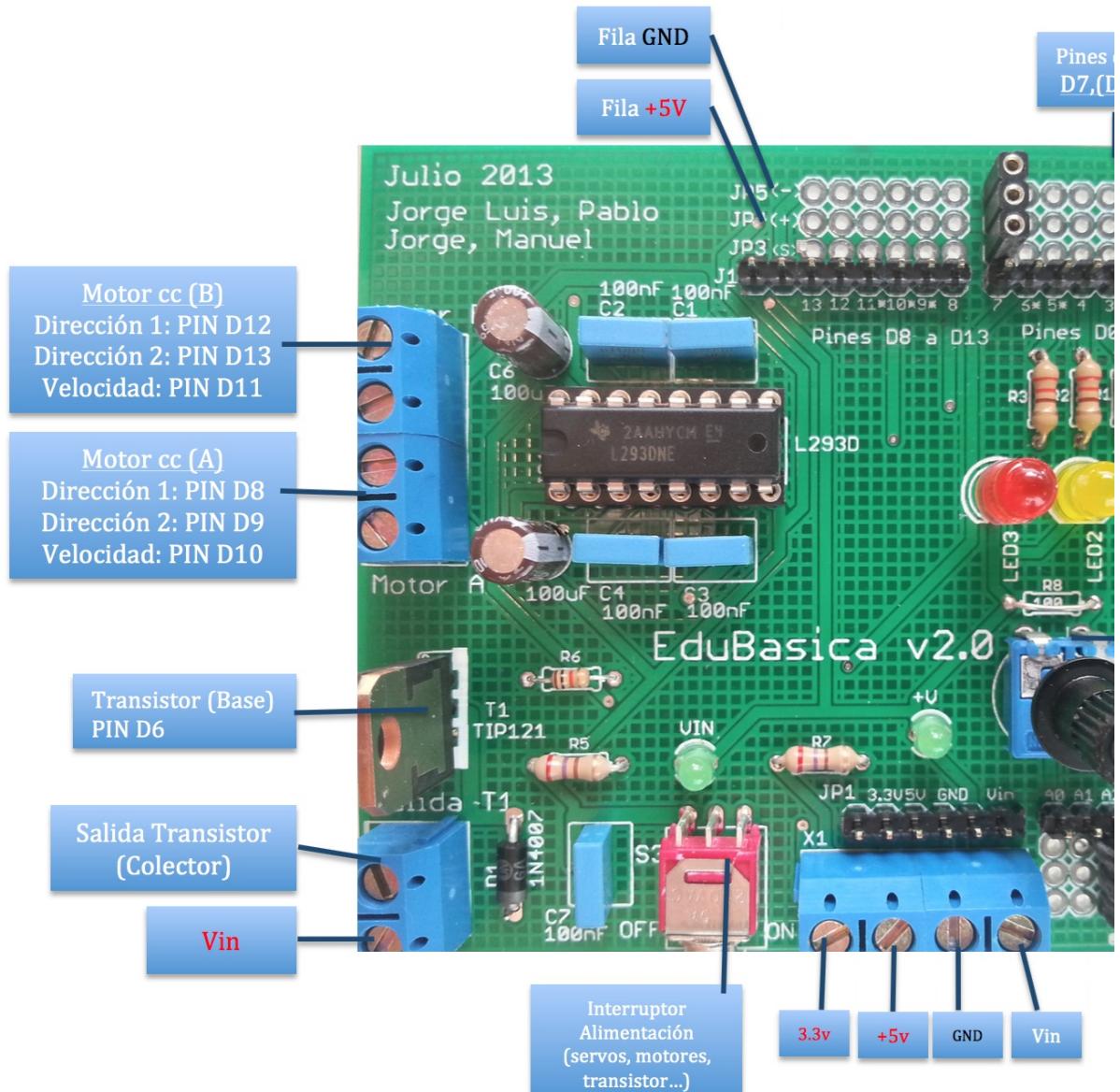
Edubásica lleva un interruptor que permite tomar el voltaje de la salida Vin de Arduino (alimentación externa), necesaria para dar la corriente suficiente para accionar los motores. Cuando el piloto Vin está encendido significa que la alimentación de Edubásica viene de Vin de Arduino, o bien, directamente desde una fuente externa conectada a la clema Vin de Edubásica de la regleta de alimentación (en la imagen la regleta de la parte inferior).

Es muy fácil, podemos conectar hasta dos motores en los pines dispuestos para ello, y utilizaremos:

- Para el motor A el control de velocidad por el pin 10 y las direcciones por 8 y 9
 - Para el motor B el control de velocidad por el pin 11 y las direcciones por 12 y 13.
 - Interruptor en ON



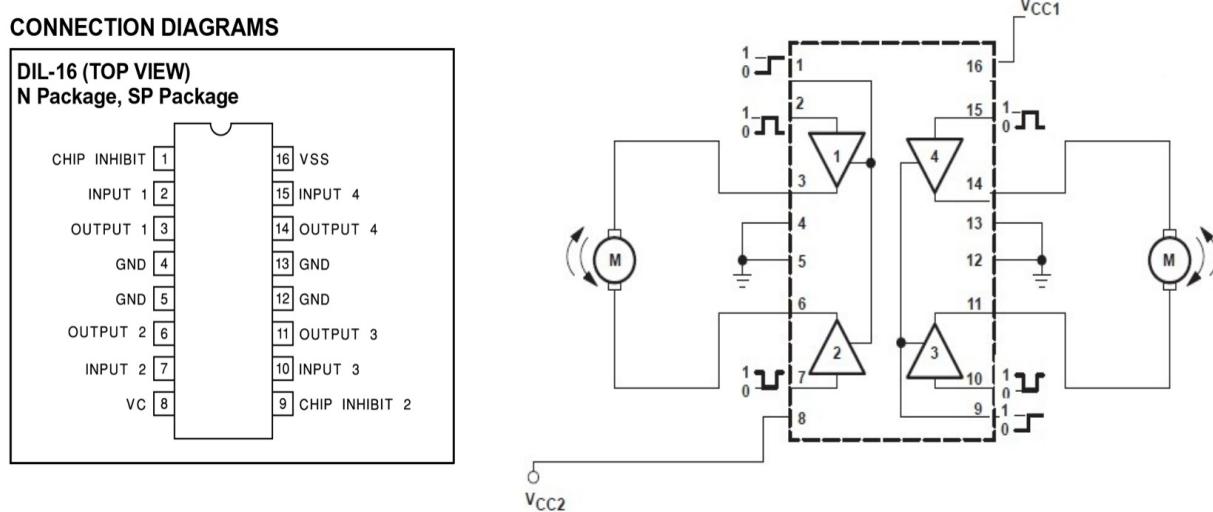
Conectando los dos terminales del motor a la clema del transistor de Edubásica, también podríamos hacerlo funcionar enviando un nivel HIGH al pin digital 6 de Arduino. Este pin (D6) está conectado directamente a la base del transistor de Edubásica. La desventaja respecto al CI L293 es que, en este caso, no podríamos cambiar el sentido de giro.



Sin Edubásica

Vamos a ver de qué manera se pueden activar los motores DC para hacer una secuencia sencilla de giro. Primero de todo vamos a ver cómo se conecta todo.

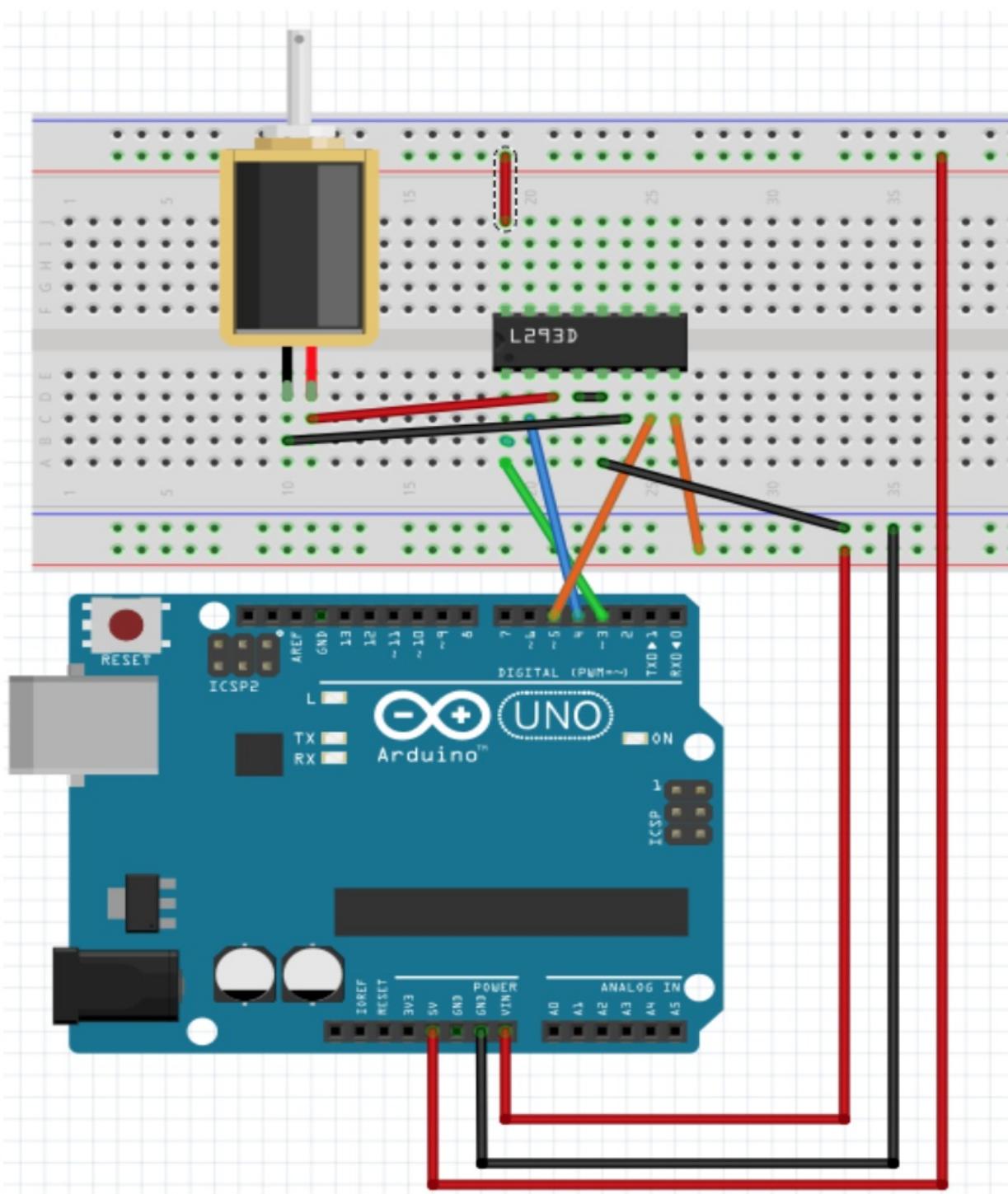
Las conexiones del circuito integrado según podemos ver en la hoja de datos del fabricante son las siguientes:



Vcc2(VC), pin 8 del L293: El voltaje que se introduzca aquí alimentará a los motores (Vin de Arduino).

Vcc1(VSS), pin 18 del L293: El voltaje que se introduzca aquí alimentará al propio circuito integrado (+5V de Arduino).

Las conexiones del circuito con Arduino las podemos ver en el siguiente esquema:



Vamos a detallar la correspondencia entre pines de Arduino y del circuito L293

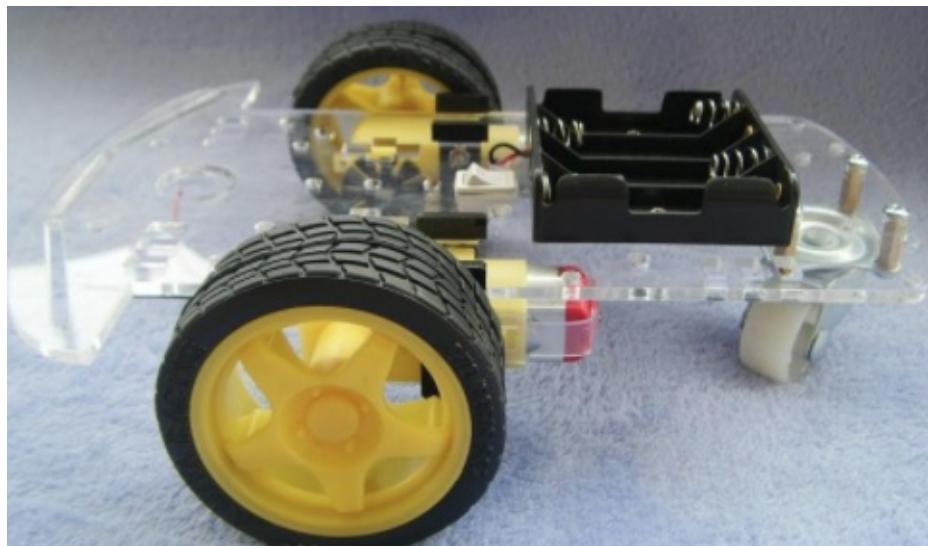
Conexiones	
Arduino	L293
3	1
4	2
5	7
Vin	8
+5V	16
GND	4,5

Motor + → pin 3 (L293)
 Motor - → pin 6 (L293)

Para saber la orientación del sentido de giro disponemos de la siguiente tabla de verdad:

Pin 1	Pin 2	Pin 7	Function
High	High	Low	Turn Anti-clockwise (Reverse)
High	Low	High	Turn clockwise (Forward)
High	High	High	Stop
High	Low	Low	Stop
Low	X	X	Stop

Vamos a darle un poco de emoción



via GIPHY

Chasis coche Con Edubásica

Puedes ver en [esta página de Luis Llamas](#) cómo hacer un coche teledirigido puede salir por menos de 20€, nosotros como ya tenemos el L298N integrado en **Edubásica**, sólo hemos comprado el [chasis](#) por 7€ y no vamos a poner sensor ultrasonidos y sensor de línea:

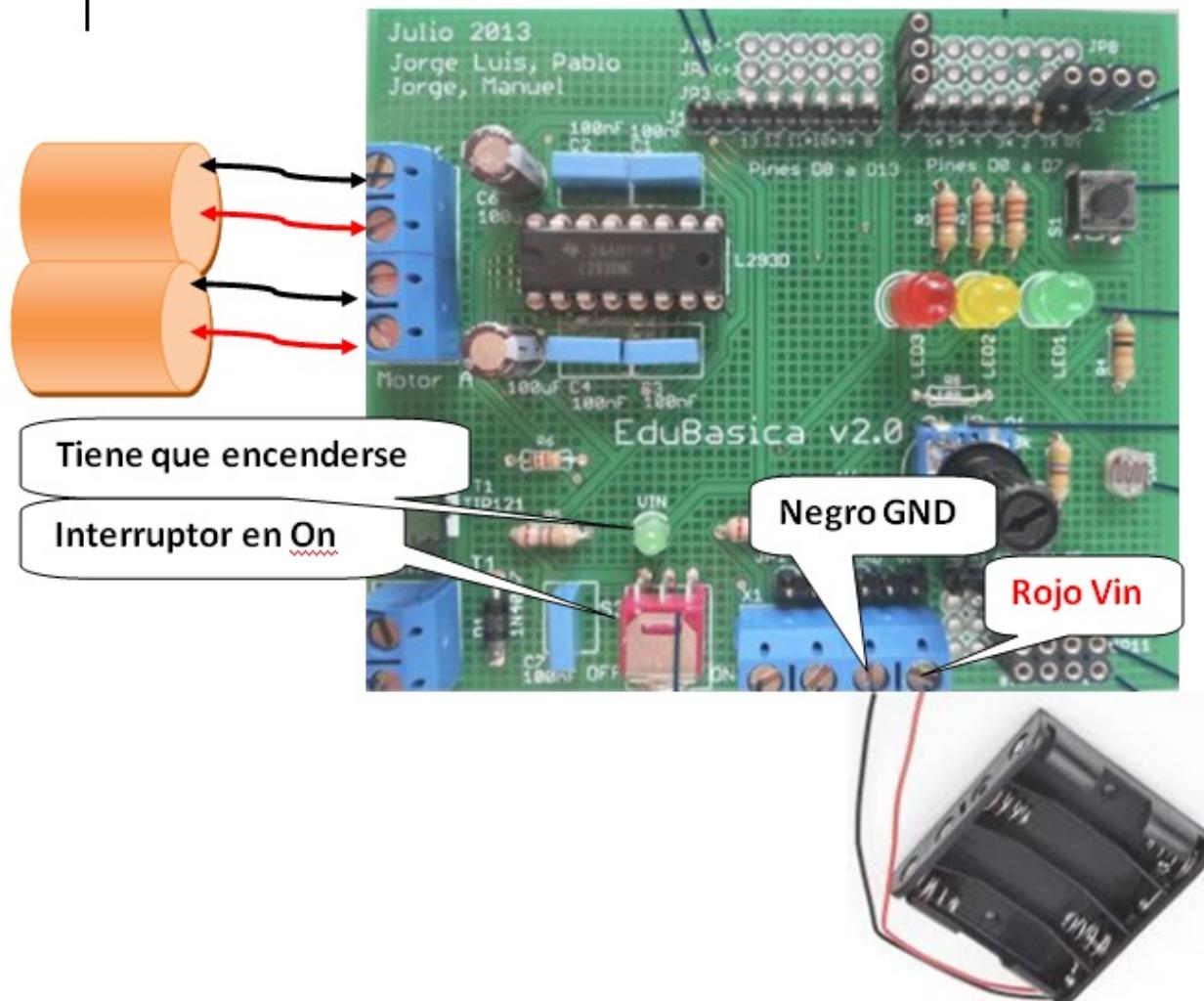


Si quieres hacerlo todo terreno [Luis Llamas](#) te explica cómo hacerlo con cadenas.

Conexiones

Las conexiones son muy sencillas:

- Pines motor A al conector pines motor A de Edubásica
- Pines motor B al conector pines motor B de Edubásica
- Cables de las pilas a Vin y GND



NOTA: Si funciona pero con las órdenes cambiadas (es decir que queremos que gire a la izquierda y va hacia delante, queremos que vaya hacia delante y gira a la izquierda...) es debido a que tenemos los cables rojo y negro de un motor intercambiados.

Ordenes

Estos son los pines que hay que activar:

Orden	MotorA	MotorB
Velocidad	10	11
Dirección 1	8	12
Dirección 2	9	13

La tabla de verdad es muy fácil:

MOTOR A

Pin 10	Pin 8	Pin 9	MotorA
HIGH	HIGH	LOW	giro
HIGH	LOW	HIGH	giro contrario
LOW	X	X	STOP

MOTOR B

Pin 11	Pin 12	Pin 13	MotorB
HIGH	HIGH	LOW	giro
HIGH	LOW	HIGH	giro contrario
LOW	X	X	STOP

AUN TE QUEDA UNA CONEXIÓN

Recuerda que en el transistor aún puedes poner otro motor. Respecto a terminales digitales, te quedan pines para poner servos, sensor de ultrasonidos, sensor de líneas...



via GIPHY

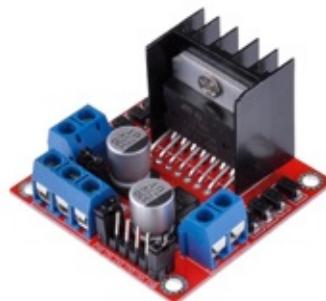
... en fin que las posibilidades son muchas



via GIPHY

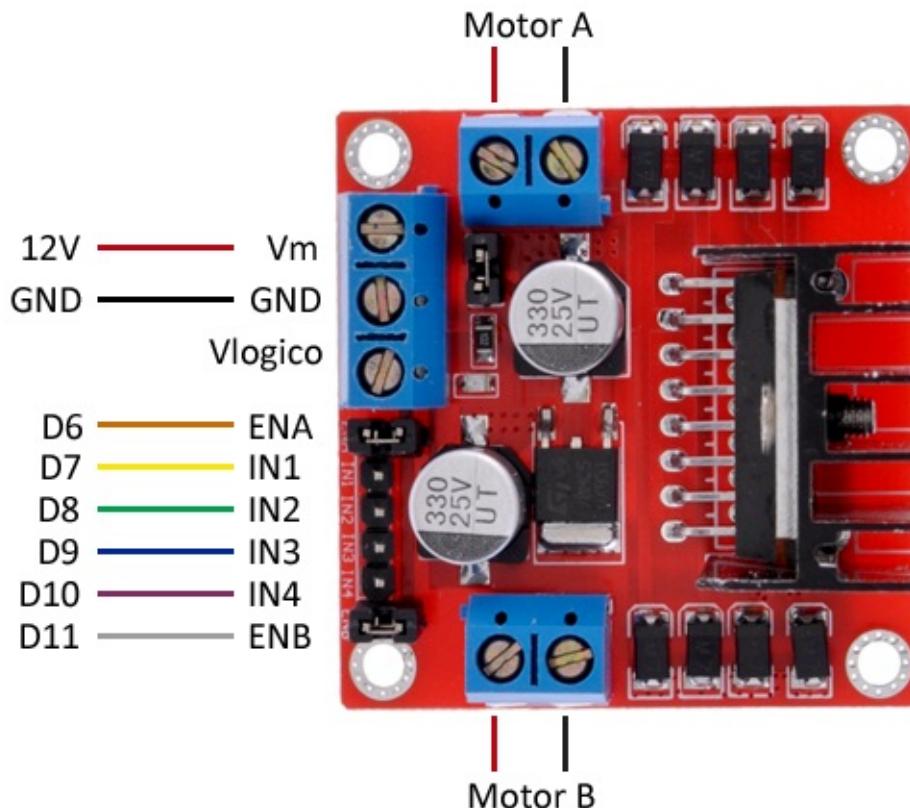
Chasis sin Edubásica

Tenemos que comprar el chasis anterior y además un L298N, recomendamos ver esta página de Luis Llamas



Conexiones

La circuitería se complica, pues necesitamos cablear los pines de control:



Tal y como dice Luis Llamas:

Siempre que uséis más de una fuente de tensión recordar poner en común todos los GND incluido el de Arduino. De lo contrario podéis dañar un componente.



Ordenes

Para respetar los mismos programas que Luis Llamas, estos son los pines que hay que activar:

Orden		MotorA	MotorB
Velocidad	6		11
Dirección 1	7		9
Dirección 2	8		10

La tabla de verdad es muy fácil:

MOTOR A

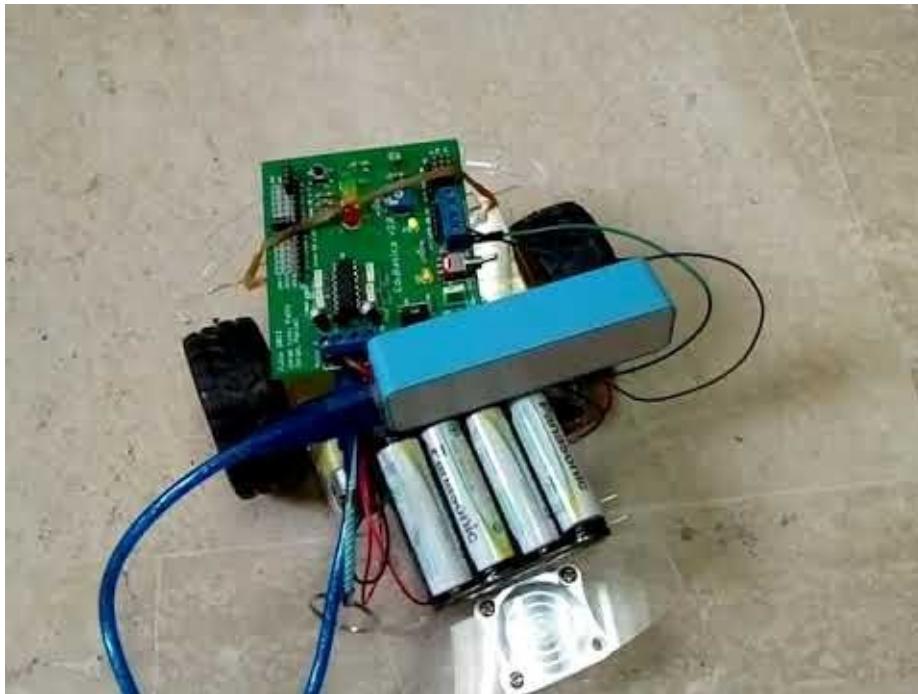
Pin 6	Pin 7	Pin 8	MotorA
HIGH	HIGH	LOW	giro
HIGH	LOW	HIGH	giro contrario
LOW	X	X	STOP

MOTOR B

Pin 11	Pin 9	Pin 10	MotorB
HIGH	HIGH	LOW	giro
HIGH	LOW	HIGH	giro contrario
LOW	X	X	STOP

Montaje 6 Coche loco

Vamos a realizar un programa que cuando se apriete el botón de Edubásica, pues el coche haga un baile:



[Video link](#)

Los programas con motor son sencillos pero engorrosos, este programa es sencillo y verás que tiene muchas líneas, un poco de complicación y se complica mucho más

CODIGO CON EDUBÁSICA

```

1 //////////////// CON EDUBASICA ///////////
2 #define ENABLEA 10
3 #define DIR1A 8
4 #define DIR2A 9
5 #define ENABLEB 11
6 #define DIR1B 12
7 #define DIR2B 13
8 #define BOTON 2
9 #define LEDVERDE 3
10 #define LEDAMARILLO 4
11 #define LEDROJO 5
12 byte dato;
13 /////////////////
14 /////////////////
15 void setup() {
16     ///configuraciones inicio
17     pinMode(ENABLEA, OUTPUT);  pinMode(DIR1A, OUTPUT);  pinMode(DIR2A, OUTPUT);
18     pinMode(ENABLEB, OUTPUT);  pinMode(DIR1B, OUTPUT);  pinMode(DIR2B, OUTPUT);
19     pinMode(BOTON, INPUT);
20     pinMode(LEDVERDE, OUTPUT);  pinMode(LEDAMARILLO, OUTPUT);  pinMode(LEDROJO, O

```

```

21 UTPUT);
22     /// delante
23     digitalWrite(LEDVERDE, HIGH);      digitalWrite(LEDROJO, LOW);    digitalwri
24     te(LEDAMARILLO, LOW);
25     digitalWrite(ENABLEA,HIGH);        digitalWrite(DIR1A,HIGH);    digitalwri
26     te(DIR2A,LOW);
27     digitalWrite(ENABLEB,HIGH);        digitalWrite(DIR1B,HIGH);    digitalwri
28     te(DIR2B,LOW);
29     delay(1000);
30     //atras
31     digitalWrite(LEDVERDE, LOW);      digitalWrite(LEDROJO, HIGH);   digitalWr
32     ite(LEDAMARILLO, LOW);
33     digitalWrite(ENABLEA,HIGH);        digitalWrite(DIR1A,LOW);    digitalwri
34     te(DIR2A,HIGH);
35     digitalWrite(ENABLEB,HIGH);        digitalWrite(DIR1B,LOW);    digitalwri
36     te(DIR2B,HIGH);
37     delay(1000);
38     //derecha
39     digitalWrite(LEDVERDE, HIGH);      digitalWrite(LEDROJO, LOW);   digitalW
40     rite(LEDAMARILLO, HIGH);
41     digitalWrite(ENABLEA,HIGH);        digitalWrite(DIR1A,LOW);    digitalWr
42     ite(DIR2A,HIGH);
43     digitalWrite(ENABLEB,HIGH);        digitalWrite(DIR1B,HIGH);   digitalWr
44     ite(DIR2B,LOW);
45     delay(1000);
46     //izquierda
47     digitalWrite(LEDVERDE, HIGH);      digitalWrite(LEDROJO, HIGH);  digital
48     Write(LEDAMARILLO, HIGH);
49     digitalWrite(ENABLEA,HIGH);        digitalWrite(DIR1A,HIGH);   digitalWr
50     ite(DIR2A,LOW);
51     digitalWrite(ENABLEB,HIGH);        digitalWrite(DIR1B,LOW);   digitalWr
52     ite(DIR2B,HIGH);
53     delay(1000);
54     //stop
55     digitalWrite(LEDVERDE, LOW);      digitalWrite(LEDROJO, LOW);   digitalWr
56     ite(LEDAMARILLO, LOW);
57     digitalWrite(ENABLEA,LOW);
58     digitalWrite(ENABLEB,LOW);
59 }
60 void loop(){
61 }

```

SIN EDUBASICA

Cambia el principio de la cabecera, y omite los LEDs:

```

1 ////////////// SIN EDUBASICA ///////////
2 #define ENABLEA 6
3 #define DIR1A 7
4 #define DIR2A 8

```

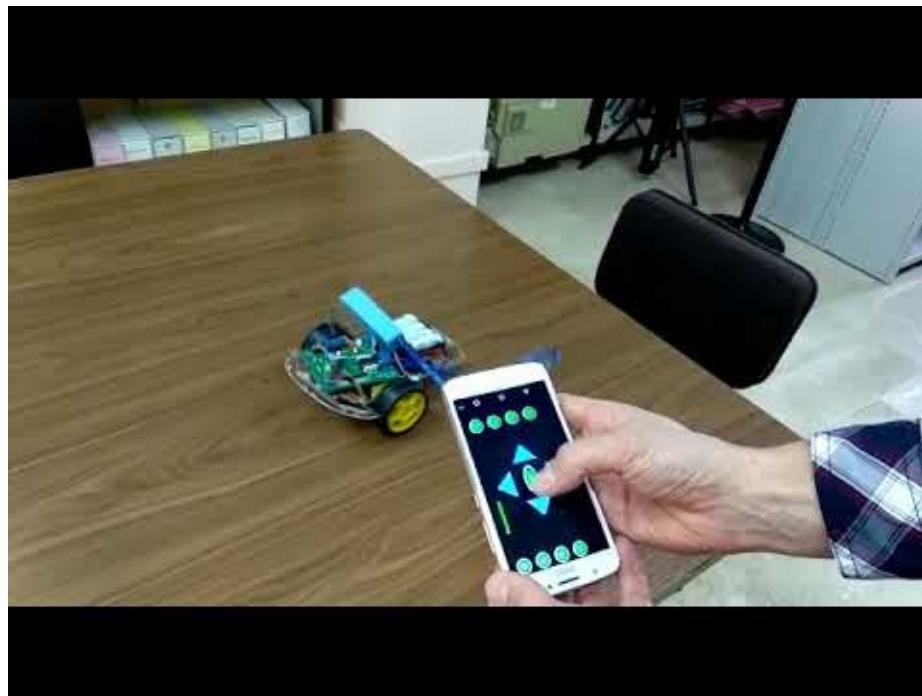
```
5 #define ENABLEB 11  
6 #define DIR1B 9  
7 #define DIR2B 10
```

Reto

¿Qué tal si en vez de activarse con el botón, que simplemente cuando este oscuro (utiliza el valor del pin analógico 2 = LDR), se ponga a bailar?

Montaje 7 Coche teledirigido

Sé que lo estabas pensando... ponerle el [Bluetooth](#), vamos allá:



[Video link](#)

OJO, para realizar este ejercicio tienes que vincular el HC-06 con la APP, [¿te acuerdas cómo lo hicimos?](#)

CODIGO CON EDUBÁSICA

```

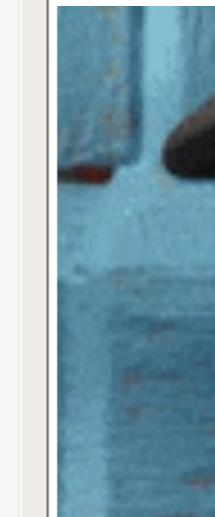
1 //////////////// CON EDUBASICA /////////////
2 #define ENABLEA 10
3 #define DIR1A 8
4 #define DIR2A 9
5 #define ENABLEB 11
6 #define DIR1B 12
7 #define DIR2B 13
8 #define BOTON 2
9 #define LEDVERDE 3
10 #define LEDAMARILLO 4
11 #define LEDROJO 5
12 byte dato;
13 /////////////////////////////////
14 /////////////////////////////////
15 void setup() {
16   Serial.begin(9600);
17   pinMode(ENABLEA, OUTPUT);  pinMode(DIR1A, OUTPUT);  pinMode(DIR2A, OUTPUT);
18   pinMode(ENABLEB, OUTPUT);  pinMode(DIR1B, OUTPUT);  pinMode(DIR2B, OUTPUT);
19   pinMode(BOTON, INPUT);
20   pinMode(LEDVERDE, OUTPUT);  pinMode(LEDAMARILLO, OUTPUT);  pinMode(LEDROJO, O
21   UTPUT);

```

```

22
23
24 }
25 ///////////////////////////////////////////////////////////////////
26 ///////////////////////////////////////////////////////////////////
27 void loop() {
28   if (Serial.available()) dato= Serial.read(); //Guardamos en la variable dato el valor leido
29
30   //Comprobamos el dato
31   switch(dato) { //Si recibimos una ... 85=ARRIBA 68=U=ABAJO 67=D=CENTRO
32     76=L=IZQUIERDA 82=R=DCHA 97=a 98=B 99=C
33     case 'U': //UP HACIA DELANTE
34     {
35       digitalWrite(LEDVERDE, HIGH);      digitalWrite(LEDROJO, LOW);      d
36       digitalWrite(LEDAMARILLO, LOW);    digitalWrite(ENABLEA, HIGH);      d
37       digitalWrite(DIR2A, LOW);         digitalWrite(DIR1A, HIGH);      d
38       digitalWrite(ENABLEB, HIGH);      digitalWrite(DIR1B, HIGH);      d
39       digitalWrite(DIR2B, LOW);
40       break;
41     }
42     case 'D': //"/"D": ABAJO
43     {
44       digitalWrite(LEDVERDE, LOW);      digitalWrite(LEDROJO, HIGH);      d
45       digitalWrite(LEDAMARILLO, LOW);    digitalWrite(ENABLEA, HIGH);      d
46       digitalWrite(DIR2A, HIGH);        digitalWrite(DIR1A, LOW);      d
47       digitalWrite(ENABLEB, HIGH);      digitalWrite(DIR1B, LOW);      d
48       digitalWrite(DIR2B, HIGH);
49       break;
50     }
51     case 'L': //"/"L": IZQUIERDA
52     {
53       digitalWrite(LEDVERDE, HIGH);      digitalWrite(LEDROJO, LOW);      d
54       digitalWrite(LEDAMARILLO, HIGH);    digitalWrite(ENABLEA, HIGH);      d
55       digitalWrite(DIR2A, HIGH);        digitalWrite(DIR1A, LOW);      d
56       digitalWrite(ENABLEB, HIGH);      digitalWrite(DIR1B, HIGH);      d
57       digitalWrite(DIR2B, LOW);
58       break;
59     }
60     case 'R': //"/"R": DERECHA
61     {
62       digitalWrite(LEDVERDE, HIGH);      digitalWrite(LEDROJO, HIGH);      d
63       digitalWrite(LEDAMARILLO, HIGH);    digitalWrite(ENABLEA, HIGH);      d
64       digitalWrite(DIR2A, LOW);         digitalWrite(DIR1A, HIGH);      d
65       digitalWrite(ENABLEB, HIGH);      digitalWrite(DIR1B, LOW);      d
66       digitalWrite(DIR2B, HIGH);
67       break;
68     }
69     case 'S': //si apretamos a start que pare

```



```
        {
            digitalWrite(LEDVERDE, LOW);      digitalWrite(LEDROJO, LOW);
            digitalWrite(LEDAMARILLO, LOW);
            digitalWrite(ENABLEA, LOW);
            digitalWrite(ENABLEB, LOW);
            break;
        }
    }
}
```

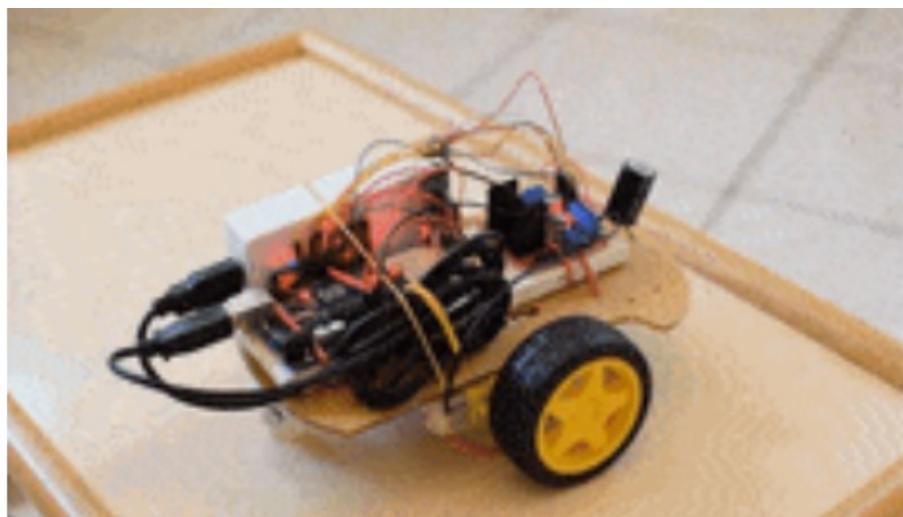
SIN EDUBASICA

Cambia el principio de la cabecera, y omite los LEDs:

```
1 ////////////// SIN EDUBASICA /////////////
2 #define ENABLEA 6
3 #define DIR1A 7
4 #define DIR2A 8
5 #define ENABLEB 11
6 #define DIR1B 9
7 #define DIR2B 10
```



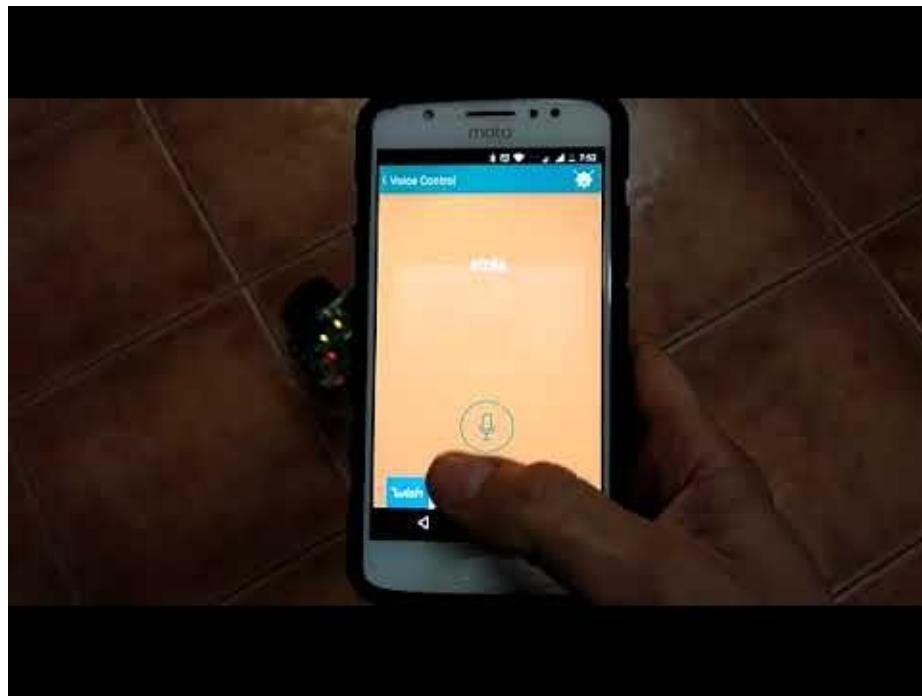
via GIPHY



via GIPHY

Coche teledirigido con voz

¿Y si ahora lo controlamos por voz?



[Video link](#)

Solución

!! Es el mismo código que [6.2.3.4 M7 Coche teledirigido !!!](#)

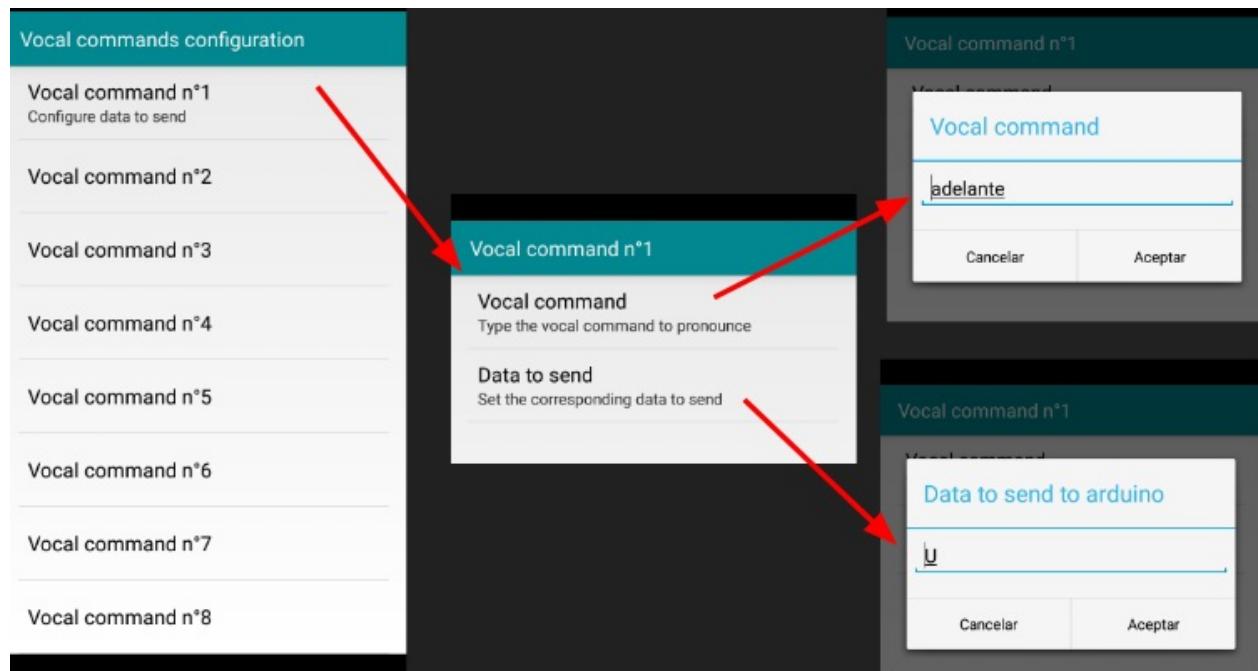
Lo que pasa es en la aplicación [Arduino Blue Control](#) utilizamos en vez del mando con flechas, el control de voz. [Easy-peasy !!](#)



Y previamente hemos configurado los comandos de voz:

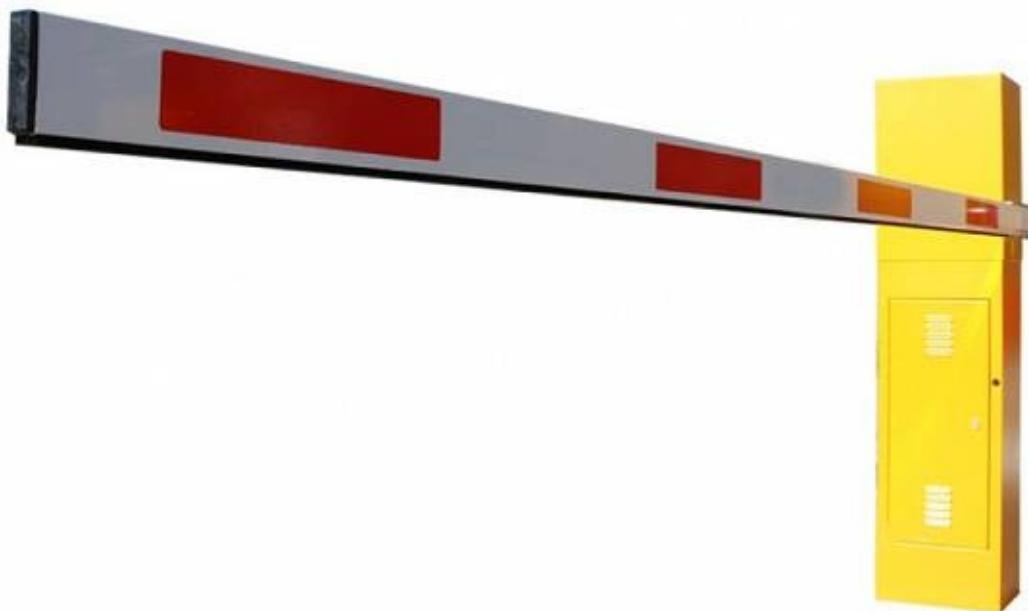
Orden de voz	Comando a enviar
adelante	U
atrás	D
derecha	R
izquierda	L
stop	S

Total 5 comandos de voz a configurar:



Barrera

Vamos ahora a realizar UN PROYECTO donde englobamos varios de los elementos que hemos visto en este curso, algo que visualmente tenga un sentido práctico y motivador en el alumnado



Utilizaremos:

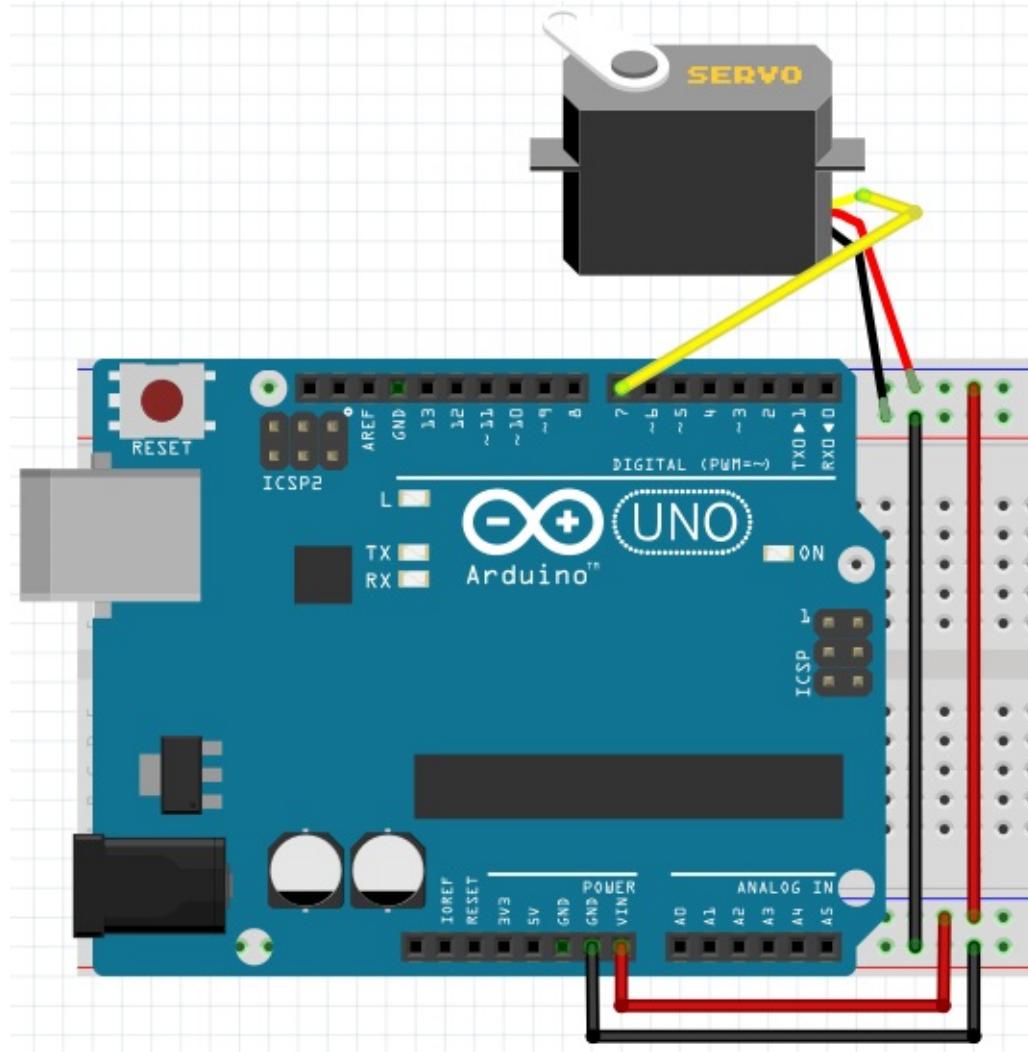
- Placa Shield de Edubásica (opcional) para facilitar las conexiones
- Servo motor
- Dos sensores de ultrasonidos
- Módulo Bluetooth
- Imaginación y maña

Montaje 13 Barrera I por Bluetooth

El propósito es que cuando se pulse la flecha arriba de la [APP DEL MOVIL](#) la barrera suba y se enciende la luz verde, y cuando se pulsa la flecha abajo, baje la barrera y se enciende la luz roja, esta es una manera eficaz de que nadie entre en el recinto si no está autorizado, y que mejor que con una aplicación móvil.

SIN EDUBÁSICA

Hay que utilizar el esquema del servo

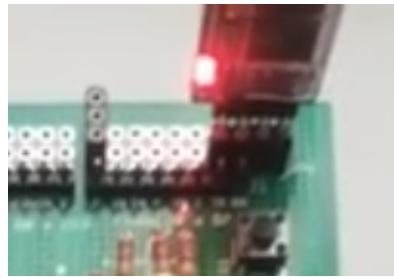


y el esquema del Bluetooth a la vez

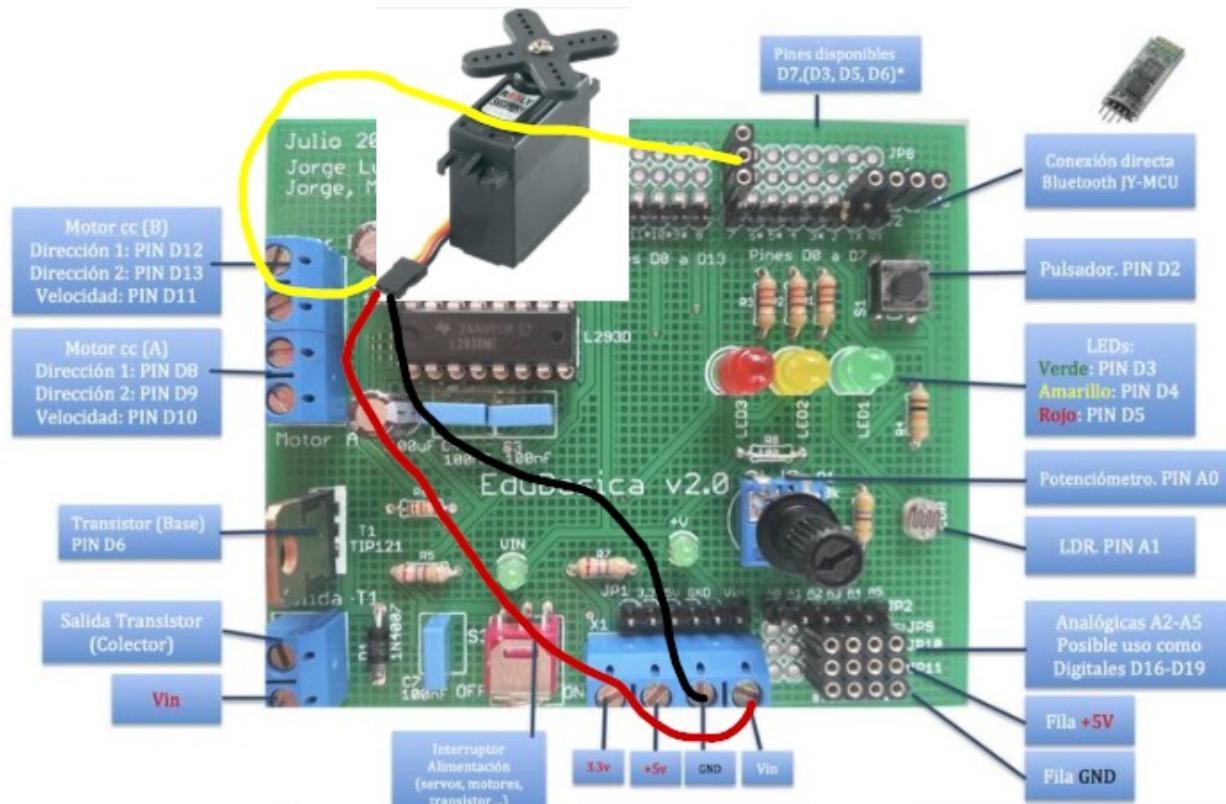


CON EDUBÁSICA

Nos simplifica el cableado, conectando el módulo Bluetooth en el zócalo correspondiente [tal y como vimos](#)



Con piezas de lego fijamos el servo y le añadimos un cartón que simule una barrera. El pin del servo lo conectaremos en el 7 de **Edubásica**, el Vcc y G a Vin y masa.



Continuamos ...

La configuración de los ángulos de abierto y cerrado depende en qué posición atornillamos la barrera, luego lo mejor es [probarlo con el programa que test que vimos](#), y en nuestro caso nos sale que 40° es abierto y 140° es cerrado.

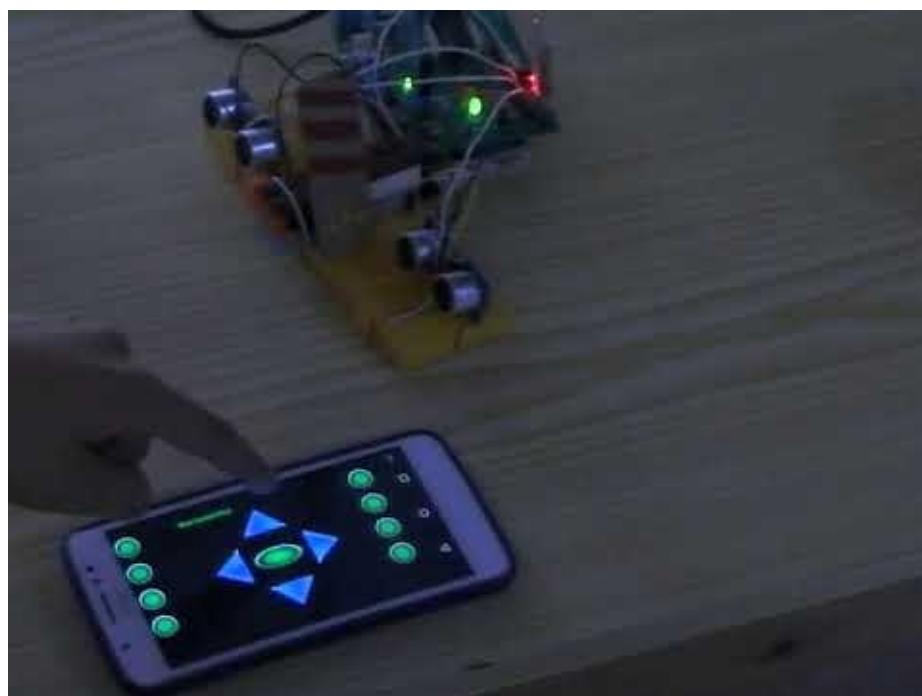


Imagen - Detalle de la unión barrera hecha con cartón y el servo, sujetado con piezas de lego y la barrera utilizando el accesorio cruz del servo y atado con un hilo de cobre

Resultado

(no hagas caso de los sensores de ultrasonidos por ahora, corresponde al siguiente montaje)

No desmontes las conexiones, te servirán para [el siguiente montaje](#).



[Video link](#)

El programa en el Arduino es el siguiente:

```
[REDACTED]
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
  
String readString;  
#include <Servo.h>  
#include <Servo.h>  
  
Servo myservo; // crea un objeto tipo servo para controlar el servo  
int pos ; // variable para almacenar la posición del servo  
byte dato;  
int ledArriba = 3; //LED ROJO DE EDUBASICA
```

```
int ledCentro = 4; //LED AMARILLO DE EDUBASICA
int ledAbajo = 5; // LED VERDE DE EDUBASICA

void setup(){
    myservo.attach(7); // En EduBasica el servo se conecta al pin 7
    Serial.begin(9600);
    pinMode(ledArriba,OUTPUT);
    pinMode(ledAbajo,OUTPUT);
    pinMode(ledCentro,OUTPUT);
}

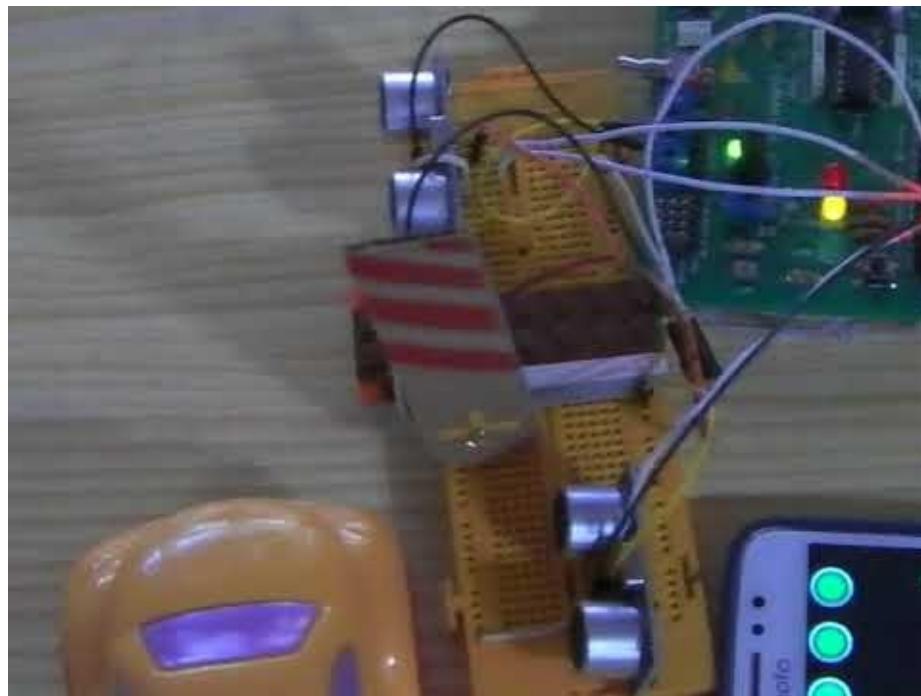
void loop()
{
    if (Serial.available()) //Guardamos en la variable dato el valor leido
        dato= Serial.read();
    //Comprobamos el dato
    switch(dato) { //Si recibimos una ... 85=ARRIBA 68=U=ABAJO
        case 85: //ARRIBA
            digitalWrite(ledArriba, HIGH);
            digitalWrite(ledAbajo, LOW);
            digitalWrite(ledCentro, LOW);
            pos=40;
            myservo.write(pos);
            delay(1000);
            break;
        case 68: // "U": ABAJO
            digitalWrite(ledArriba, LOW);
            digitalWrite(ledAbajo, HIGH);
            digitalWrite(ledCentro, LOW);
            pos=140;
            myservo.write(pos);
            delay(1000);
            break;
    }
}
```

Montaje 14 Barrera II con sensores US

Ahora le añadimos (*esperamos que no hayas desmontado el montaje anterior*) dos sensores de ultrasonidos, si detecta el coche a la entrada de la barrera, se enciende la luz amarilla en espera que el coche pueda abrir con el móvil.

Una vez recibido el código de abrir barrera, se abre y se enciende la luz verde.

Una vez cruzado el coche, lo detecta el ultrasonido de la salida que cerrará la barrera poniendo el semáforo en rojo otra vez.



[Video link](#)

La configuración de pines de los ultrasonidos que hemos elegido esta en los comentarios del programa (o sea, dónde hay que conectar *Trg* y *Echo* de los sensores). Puedes [ver aquí cómo se conectan los ultrasonidos](#).

Si no tienes edubásica tendrás que añadir los leds de semáforo, [pincha aquí](#).

El programa por supuesto es mejorable (tiene fallos a ver si los adivinas).

```

1 String readString;
2 #include <Servo.h>
3 #include <Servo.h>
4
5 Servo myservo;          // crea un objeto tipo servo para controlar el servo
6 int pos;                // variable para almacenar la posición del servo
7 byte dato=0;
8 int ledArriba = 3; //LED VERDE DE EDUBASICA
9 int ledCentro = 4; //LED AMARILLO DE EDUBASICA
10 int ledAbajo = 5; // LED ROJO DE EDUBASICA
11 //////////////////// SENSOR ULTRASONIDOS 1 ENTRADA ///////////////////
12 int trigPinE = 4; //CONECTAR AL D4 EL TRIG ULTRASONIDOS 1
13 int echoPinE = 2; //CONECTAR A D2 EL ECHO ULTRASONIDOS 1
14 long durationE; //tiempo de ida/vuelta
15 int cmE=0; //Para almacenar el valor obtenido en cm valor=0
16 //////////////////// SENSOR ULTRASONIDOS 2 SALIDA ///////////////////

```

```

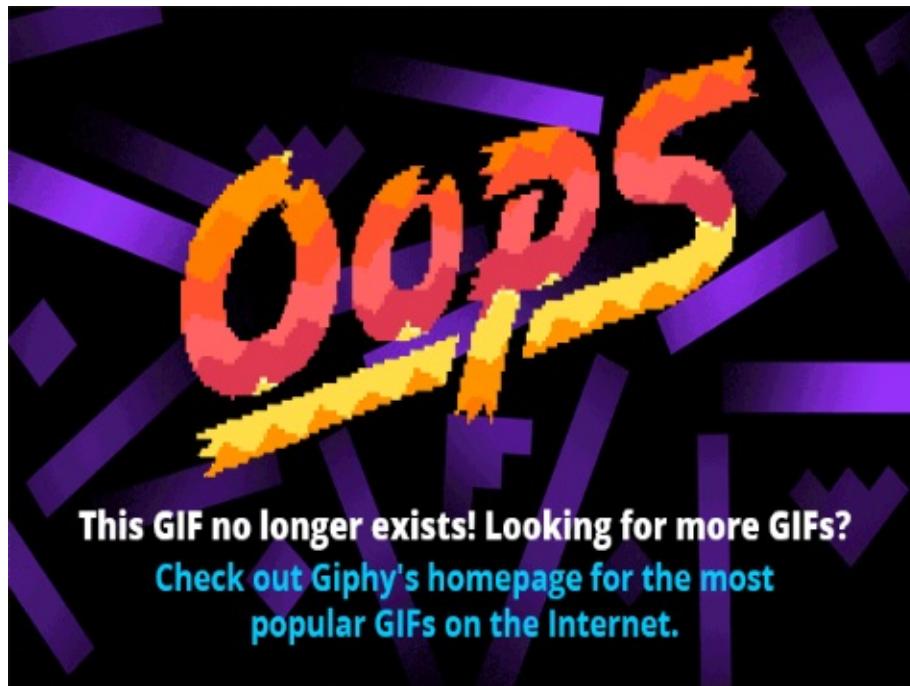
17 int trigPinS = 6; //CONECTAR AL D6 EL TRIG ULTRASONIDOS 2
18
19 int echoPinS = 5; //CONECTAR A D5 EL ECHO ULTRASONIDOS 2
20
21 long durationS; //tiempo de ida/vuelta
22 int cmS=0; //Para almacenar el valor obtenido en cm valor=0
23 ///////////////////////////////////////////////////////////////////
24
25 void setup(){
26     myservo.attach(7); // En EduBasica el servo se conecta al pin 7
27     Serial.begin(9600);
28     //////////////////// configuramos como salida los leds del semáforo
29     pinMode(ledArriba,OUTPUT);
30     pinMode(ledAbajo,OUTPUT);
31     pinMode(ledCentro,OUTPUT);
32     //////////////////// configuramos las entradas y salidas de los ultrasoni
33 dos E y S
34     pinMode(trigPinE, OUTPUT);
35     pinMode(echoPinE, INPUT);
36     pinMode(trigPinS, OUTPUT);
37     pinMode(echoPinS, INPUT);
38     ////////////////// empezamos con el semáforo en rojo
39     digitalWrite(ledArriba, LOW);
40     digitalWrite(ledAbajo, HIGH);
41     digitalWrite(ledCentro, LOW);
42     ////////////////// Cerramos la barrera al principio
43     pos=140;
44     myservo.write(pos);
45 }
46
47 void loop()
48 {
49     digitalWrite(trigPinE, LOW);
50     delayMicroseconds(2);
51     digitalWrite(trigPinE, HIGH);
52     delayMicroseconds(10);
53     digitalWrite(trigPinE, LOW);
54     durationE = pulseIn(echoPinE, HIGH);
55     durationE=durationE/2;
56     cmE = durationE/ 29;
57     Serial.println(cmE);
58     if (cmE<10){ //////////////////HAY UN COCHE EN LA ENTRADA //////
59     //////////////
60         ////////////////// PONEMOS AMARILLO EL SEMAFORO
61         digitalWrite(ledArriba, LOW);
62         digitalWrite(ledAbajo, LOW);
63         digitalWrite(ledCentro, HIGH);
64         ////////////////// ESPERAMOS A QUE DE LA ORDEN DE SUBIR BARRERA
65         if (Serial.available())
66             dato= Serial.read();
67         switch(dato) { //Si recibimos una ... 85=ARRIBA 68=ABAJO
68             case 85: //ARRIBA
69                 digitalWrite(ledArriba, HIGH);

```

```
70     digitalWrite(ledAbajo, LOW);
71     digitalWrite(ledCentro, LOW);
72     pos=40;
73     myservo.write(pos);
74     delay(1000);
75     dato=0;
76     break;
77 }
78
79 }
80 ///////////////////////////////////////////////////
81 //
82 digitalWrite(trigPinS, LOW);
83 delayMicroseconds(2);
84 digitalWrite(trigPinS, HIGH);
85 delayMicroseconds(10);
86 digitalWrite(trigPinS, LOW);
87 durationS = pulseIn(echoPinS, HIGH);
88 durationS=durationS/2;
89 cmS = durationS/ 29;
90 Serial.println(cmS);
91 if (cmS<10){ //////////////////HAY UN COCHE EN LA SALIDA //////
92 //////
93         ////////////////// ponemos con el semáforo en rojo
94     digitalWrite(ledArriba, LOW);
95     digitalWrite(ledAbajo, HIGH);
96     digitalWrite(ledCentro, LOW);
97     ////////////////// Cerramos la barrera
98     delay(3000);
99     pos=140;
100    myservo.write(pos);
101 }
102 }
```

FINNNN

Esperamos que este curso, no sólo te has formado, sino que has disfrutado. Cualquier sugerencia, cambio, propuesta, fallos... puedes hacerlo en www.catedu.es en la sección de Tickets ¡¡gracias!!!



via GIPHY



■ ■

Grupo ROBOTICA EDUCATIVA EN ARAGÓN

En <http://robaragon.chatbro.com> puedes ver un chat sincronizado con Telegram que te puedes unir con [este enlace](#) para:

- Compartir dudas y resolverlas
- Propuestas, eventos e información
- Ver lo que hacen otros compañeros



Créditos

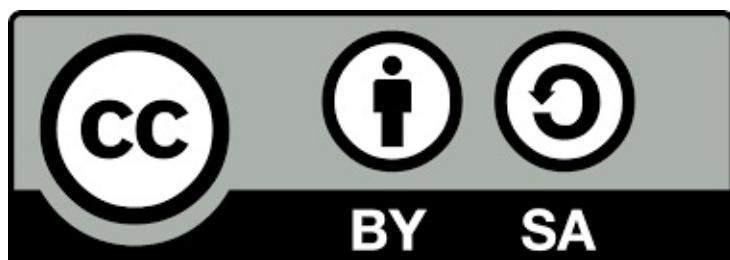
Este material ha sido elaborado por:

- Manuel Hidalgo Díaz.
- Jorge Muñoz Rodenas.
- Pablo Evaristo García Palacios.
- Jorge Luis Loza Luis.

Revisado 2017 por CATEDU (Javier Quintana Peiró).

Los vídeos que aparecen en este manual son de elaboración propia de los autores así como las fotografías de los montajes y esquemas eléctricos. Otras imágenes, principalmente las que ilustran las portadas de los capítulos se han descargado desde repositorios de imágenes libres.

El manual se distribuye bajo licencia Creative Commons tipo by-sa



Este recurso ha sido subvencionado por el Ministerio de Educación, Cultura y Deporte a través de la convocatoria de ayudas para la elaboración de recursos didácticos para su incorporación a las plataformas de acceso público 2014. Todas las actuaciones descritas en este recurso comprometen exclusivamente a los autores del mismo.

Debido al carácter "vivo" de esta publicación, los autores mantienen una réplica en el portal www.practicasconarduino.com donde se publican periódicamente modificaciones al manual, añadiendo nuevos contenidos y revisiones de la placa Edubásica.



Temas optionales

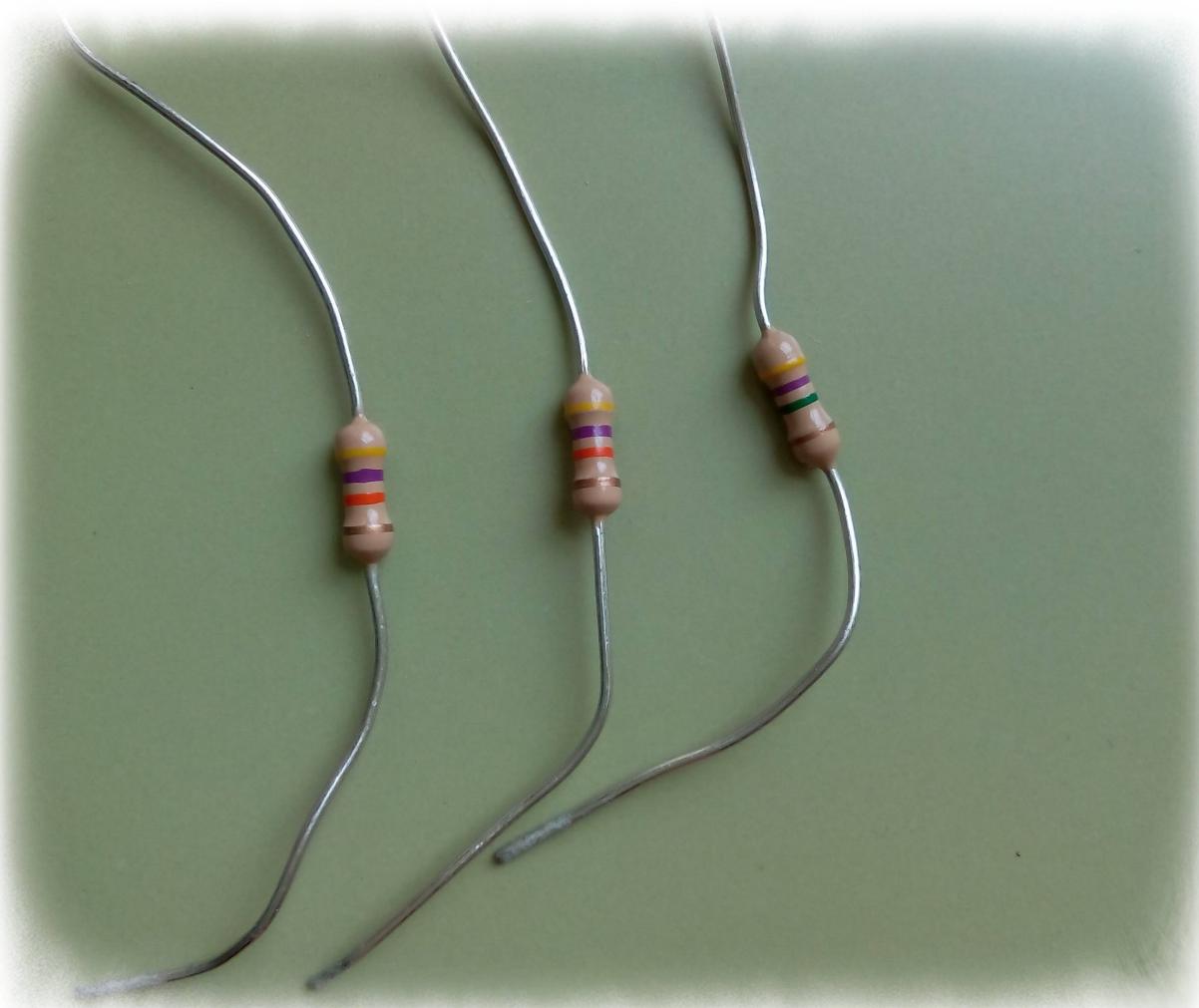
Dejamos este capítulo para temas que ya son **fuera del curso** pero que puede servir al docente para explicar **conceptos teóricos con el Arduino**

Electrónica analógica



La electrónica es la ciencia que estudia y diseña dispositivos relacionados con el comportamiento de los electrones en la materia. En nuestro caso estudiaremos los componentes básicos utilizados en estos circuitos de bajo voltaje y usaremos Arduino, y la placa EduBásica, para practicar y entender mejor su funcionamiento.

Resistencias

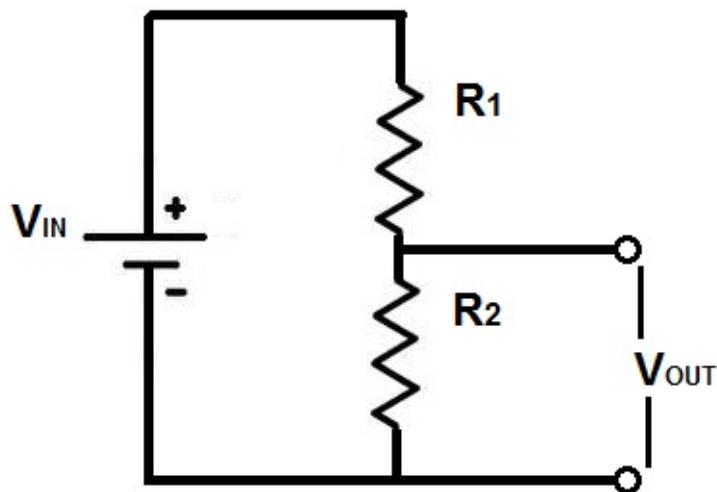


Los componentes electrónicos llamados resistencias se utilizan en los circuitos para variar los valores de intensidad y voltaje. A veces tenemos que alimentar un dispositivo y sólo disponemos de una fuente de voltaje que puede dañarlo si se conecta directamente, como ocurre por ejemplo con los ledes. Al conectarlos directamente a un pin digital de Arduino (+5V), la corriente que circula es demasiado alta para el led y una exposición prolongada puede provocar que se queme. Para evitar esto conectamos en serie con el led una resistencia eléctrica (220 ohmios por ejemplo) que hace que el valor de la intensidad sea menor. El led lucirá algo menos que si lo conectamos directamente pero alargará su vida útil.

El valor de una resistencia se suele identificar mediante unas bandas de colores impresas en su superficie. La interpretación del código de colores la puedes obtener en <http://es.wikipedia.org/wiki/Resistor> aunque suele ser recomendable, a la hora de realizar prácticas con circuitos electrónicos, utilizar un polímero que nos permitirá, entre otras cosas, medir la resistencia eléctrica en ohmios de estos componentes.

Divisor de tensión

Muchas veces necesitamos sacar una tensión que no es la máxima de la alimentación (en este caso 5V) el truco es hacerlo por resistencias, en forma de divisor de tensión:



Si aplicamos la ley de Ohm podemos deducir la siguiente fórmula: [\[aquí si quiere ver la demostración\]](#)

$$V_{OUT} = V_{IN} \frac{R_2}{(R_1 + R_2)}$$

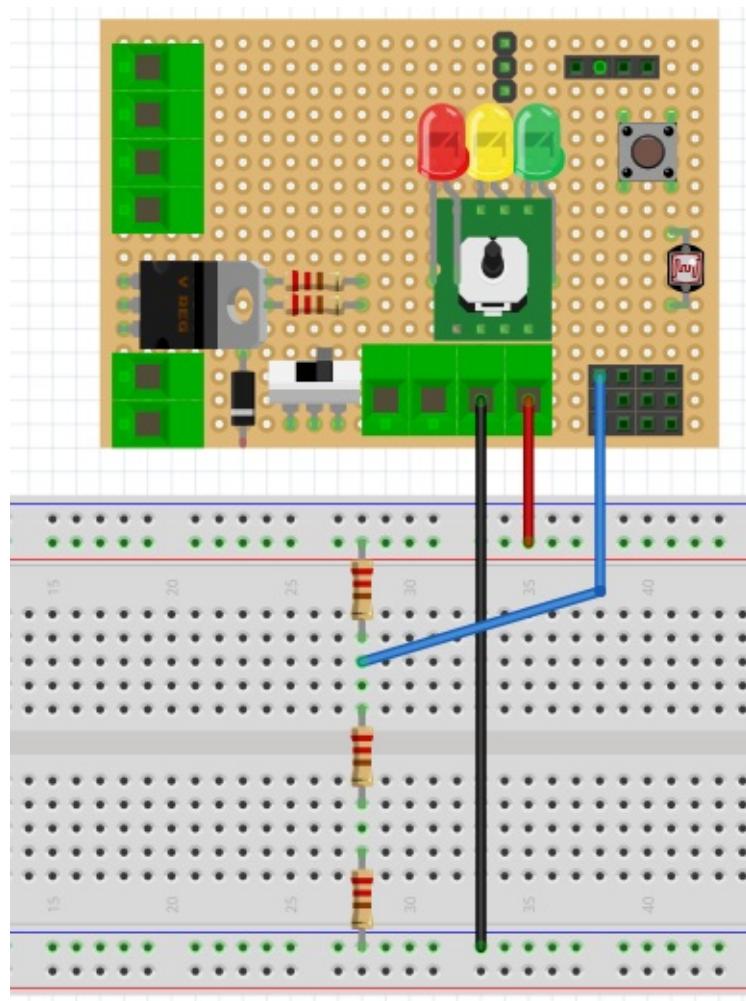
En este caso utilizaremos el divisor de tensión para demostrar que modificando la R2 por resistencias en serie o paralelo, se modifica la tensión Vout que el Arduino lo leerá y lo visualizará en un LED

Montaje 1: Resistencias en serie

Elije cuatro resistencias de cualquier valor para los montajes en serie pero procura que sean de valores muy distintos. Para ello utiliza [la tabla con los códigos de colores](#) o bien mide los valores directamente con un polímetro (medida de ohmios).

Con EDUBÁSICA

Monta el siguiente circuito divisor de tensión:

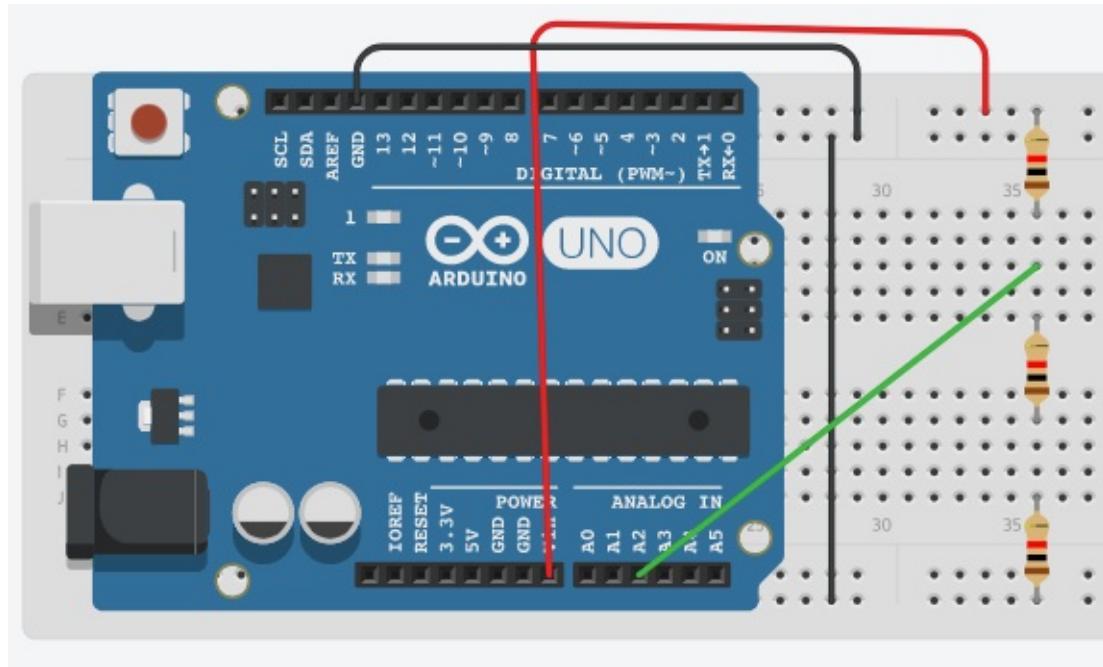


El cable rojo es la Vin (5V) y el negro GND. El cable azul es la salida Vout que la lee A2 por eso está conectado al A2 de EDUBASICA

Usa la protoboard para intercalar, entre los extremos del cable azul y negro, las resistencias que elegiste y prueba distintas combinaciones en serie quitando y poniendo resistencias. Debes observar que la luminosidad del led varía.

SIN EDUBÁSICA

Igual, simplemente que A2, Vin y GND lo tienes en el mismo ARDUINO



Continuamos...

El programa que hay que ejecutar en el arduino es este

```

1 //Conectaremos resistencias en serie entre Vout=A2 y GND
2
3 void setup() {
4     // Pin 3 tiene el LED verde
5     pinMode(3, OUTPUT);
6 }
7 void loop() {
8     analogWrite(3, analogRead(2)/4);
9     //Dividimos el valor entre 4 para adecuar la salida a 255
10 }
```

Reflexión

¿Cómo afecta el valor de las resistencias en serie en la luminosidad del LED?

Solución

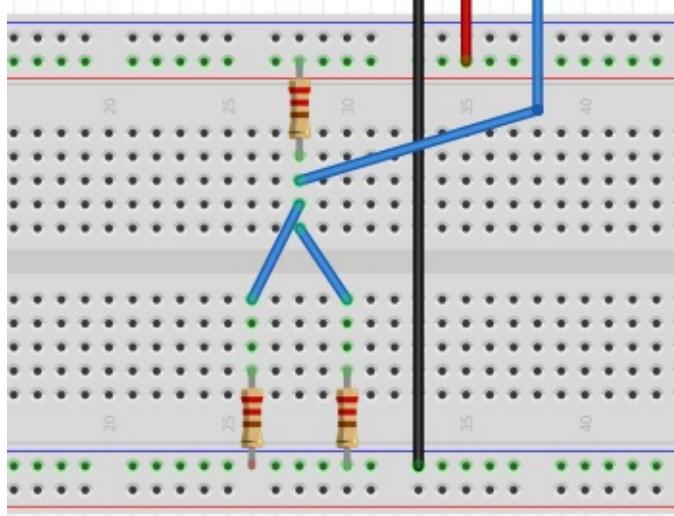
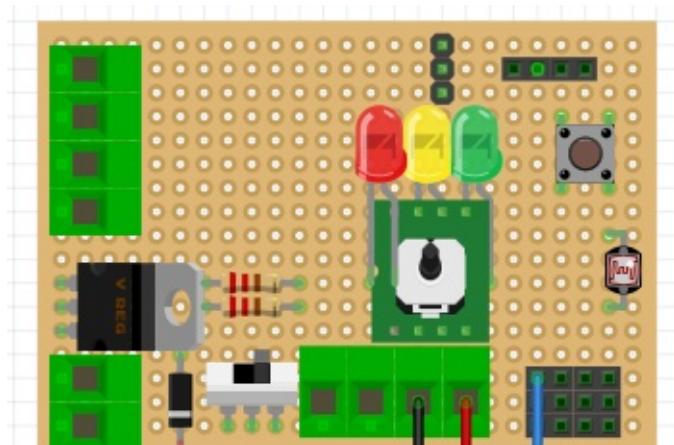
Según la fórmula del divisor de tensión CUANTO MÁS RESISTENCIA HAYA ABAJO (R2 en la fórmula) MÁS TENSIÓN HAY por lo tanto más se ilumina el led que visualiza lo que entra por A2).

Montaje 2: Resistencias en paralelo.

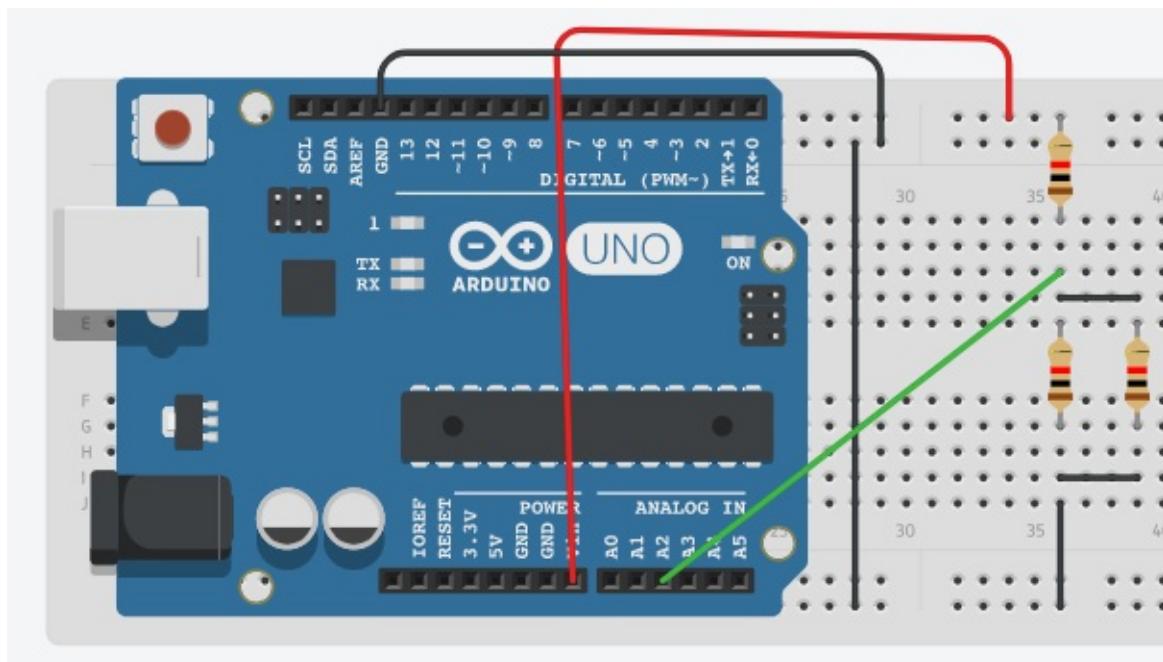
Este ejercicio es similar al anterior, puedes usar las mismas resistencias y el mismo programa.

Une ahora uno de los extremos de las resistencias conectadas en paralelo al pin 2 analógico y el otro extremo a GND. Prueba a quitar alguna de las resistencias y obtén conclusiones de lo que ocurre.

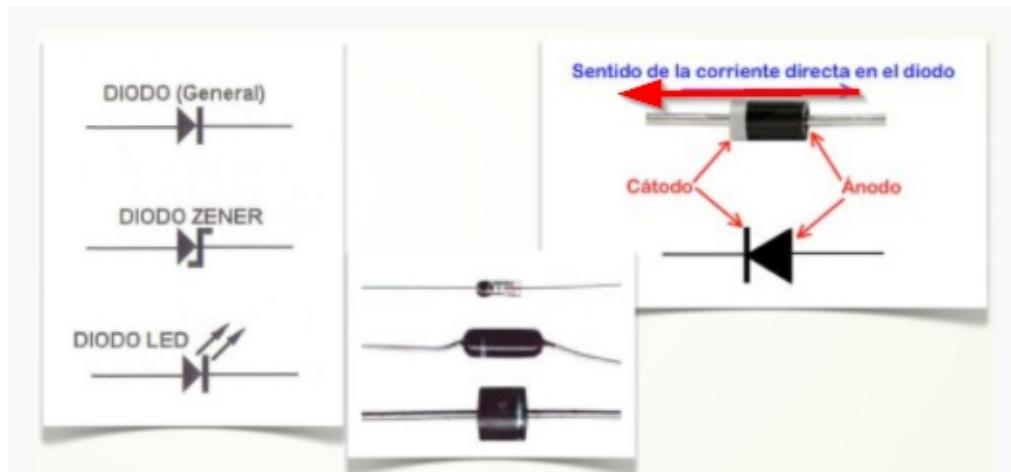
CON EDUBASICA



SIN EDUBASICA



Diodos



Seguro que has oído hablar de los diodos LED (Light-Emitting Diode) pues están muy de moda. Un diodo led no es más que un diodo que emite luz cuando está polarizado correctamente.

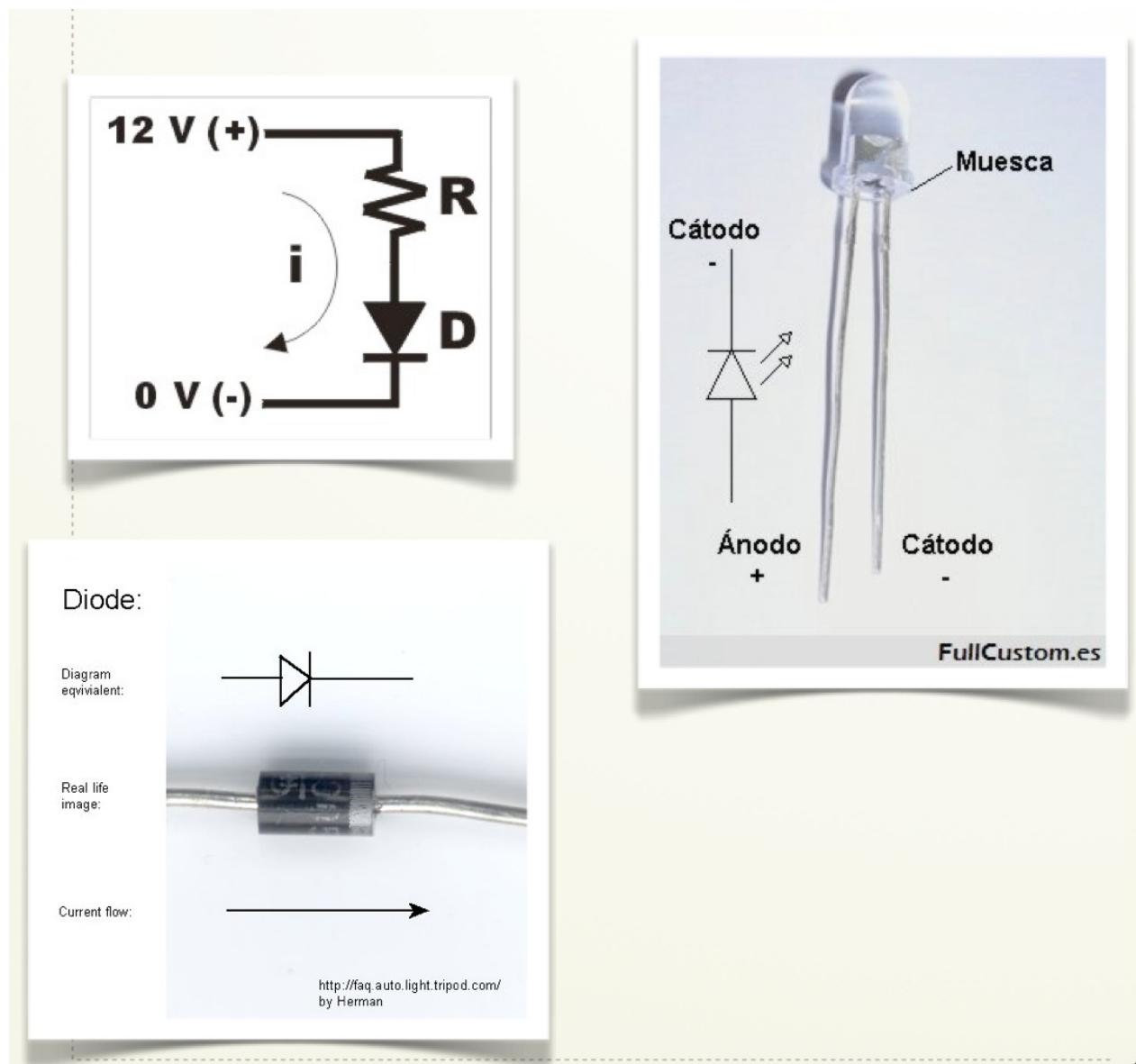
Un diodo (del griego "dos caminos") es un dispositivo semiconductor que permite el paso de la corriente eléctrica en una única dirección con características similares a un interruptor.

De forma simplificada, la curva característica de un diodo ($I-V$) consta de dos regiones: por debajo de cierta diferencia de potencial, se comporta como un circuito abierto (no conduce), y por encima de ella se comporta como un cortocircuito con muy baja resistencia eléctrica.

Veamos si sabes como polarizar un diodo...

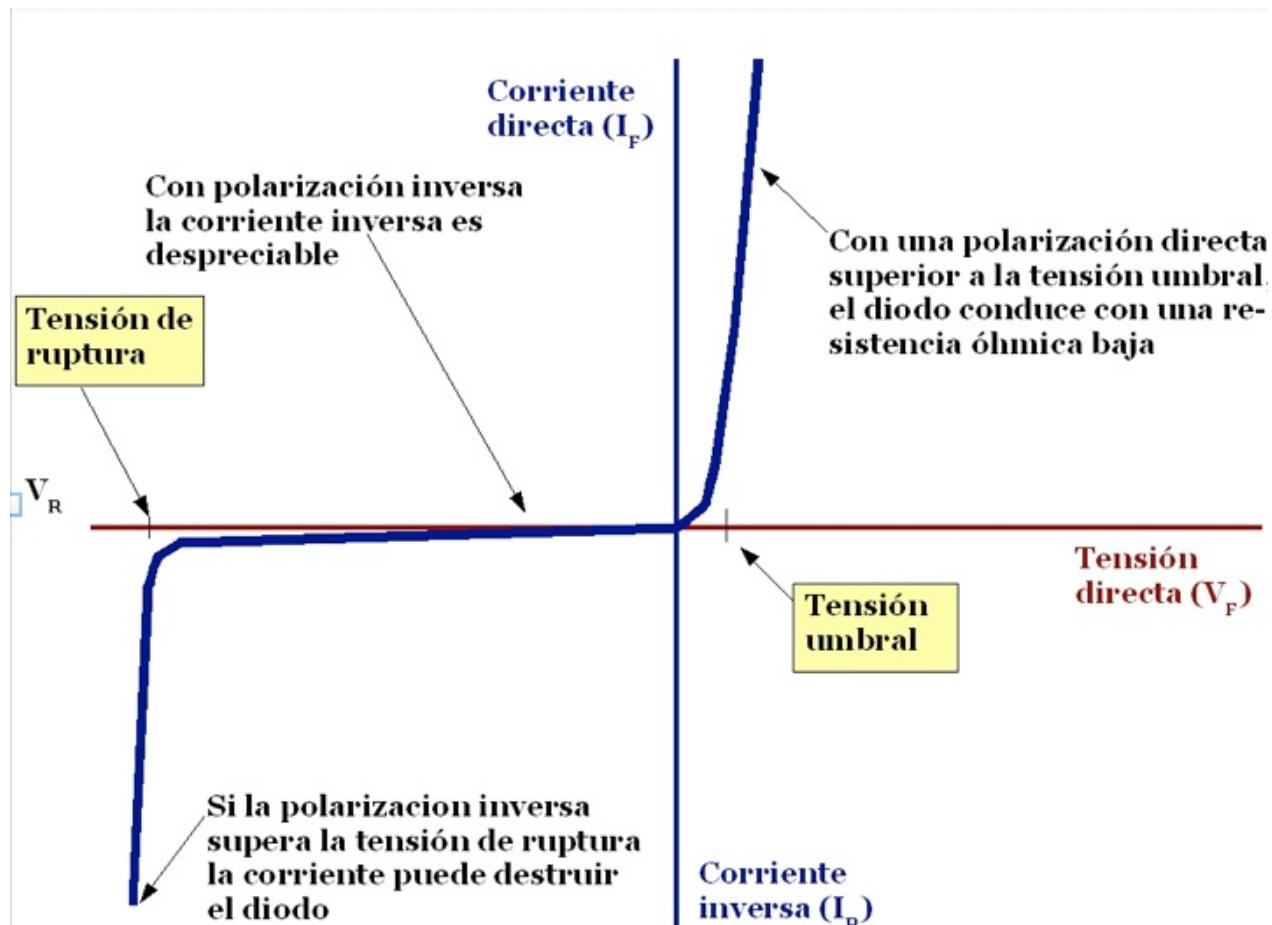
Un diodo se dice polarizado directamente cuando su cátodo está a un potencial inferior al ánodo. Los diodos por tanto deben especificar cual es el ánodo y el cátodo. En la foto puedes ver como un diodo led identifica su cátodo con una patilla recortada.

En otro tipo de diodos se puede identificar el cátodo gracias a una raya dibujada sobre el componente.



CURVA CARACTERÍSTICA DE UN DIODO:

Vamos a estudiar la curva I-V de un diodo de tal manera que comprobaremos que al aplicar un cierto voltaje el diodo conduce y no conduce (estado en corte) si estamos por debajo de ese voltaje o diferencia de potencial. En el caso específico de un diodo la diferencia de potencial a superar es de 0,7 V; si es un diodo LED es más del doble. A partir de ese valor conduce la corriente eléctrica y si es un LED, empieza a iluminarse.

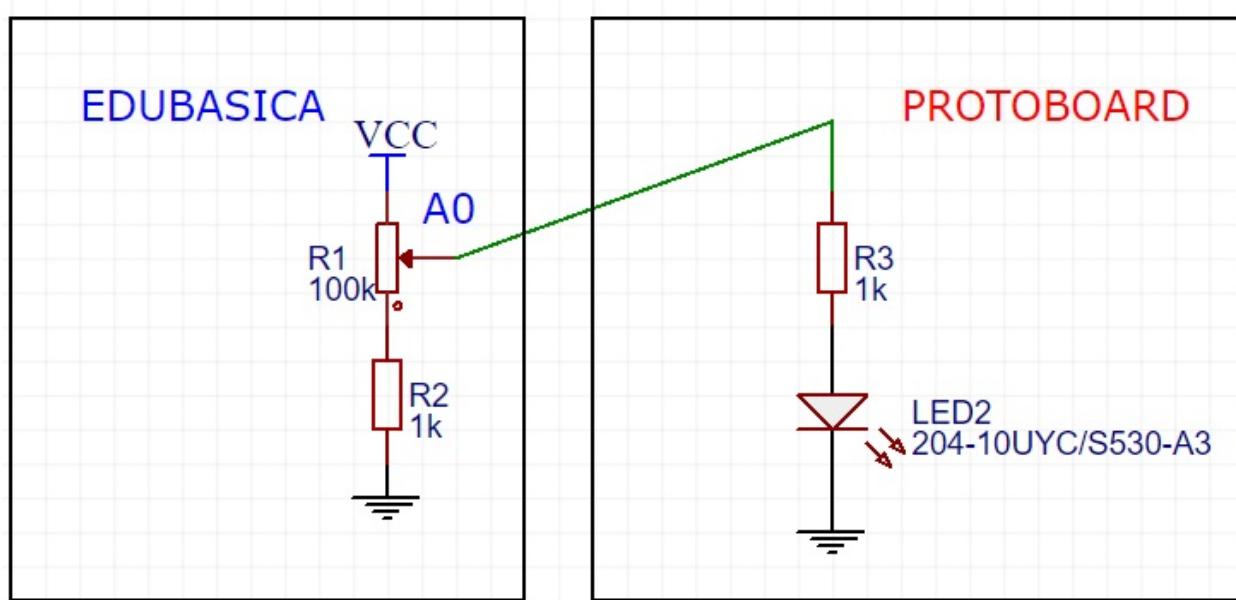


Montaje 7 Estudio de la tensión umbral de un LED

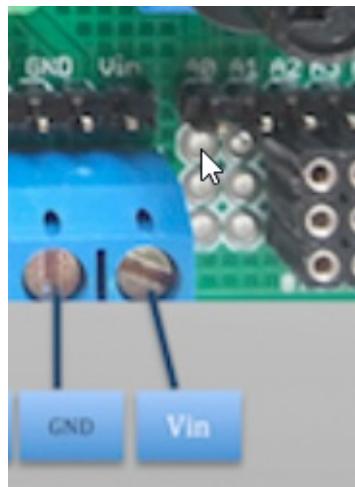
Objetivo: Medir la tensión del diodo con Arduino con una entrada analógica, A0 (por ejemplo) para detectar la tensión umbral. El diodo lo alimentaremos con un potenciómetro para ir subiendo los valores

CON EDUBÁSICA

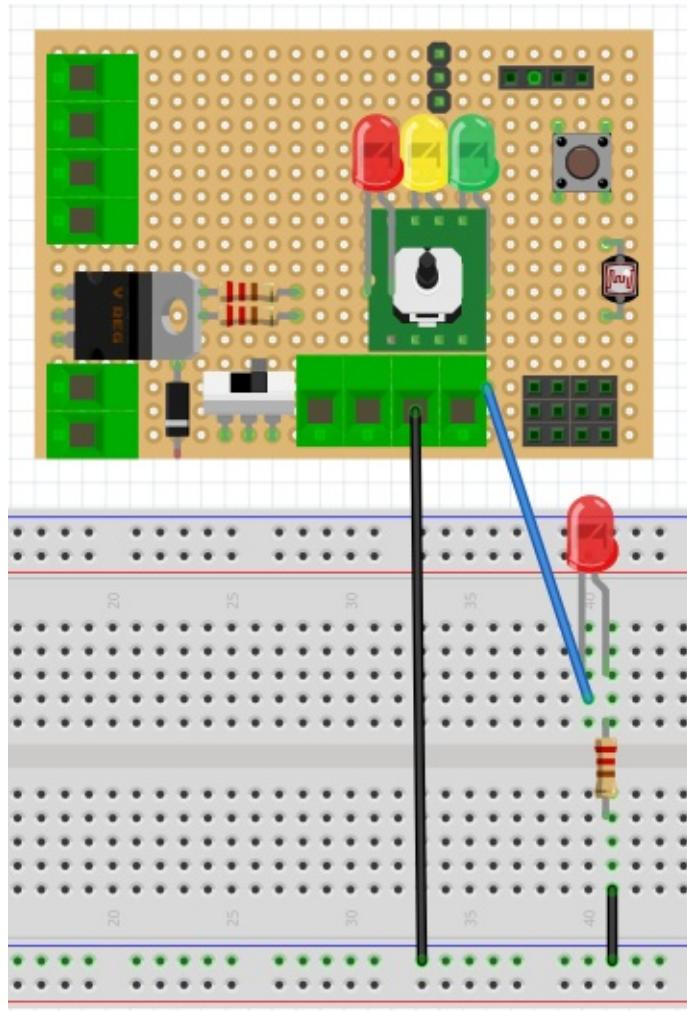
Vamos a realizar el siguiente esquema:



Para conectar un cable con A0 tenemos que localizar el orificio (señalado en esta imagen) y para la masa utilizamos el segundo tornillo



Montamos:



Carga este código en tu placa Arduino:

```

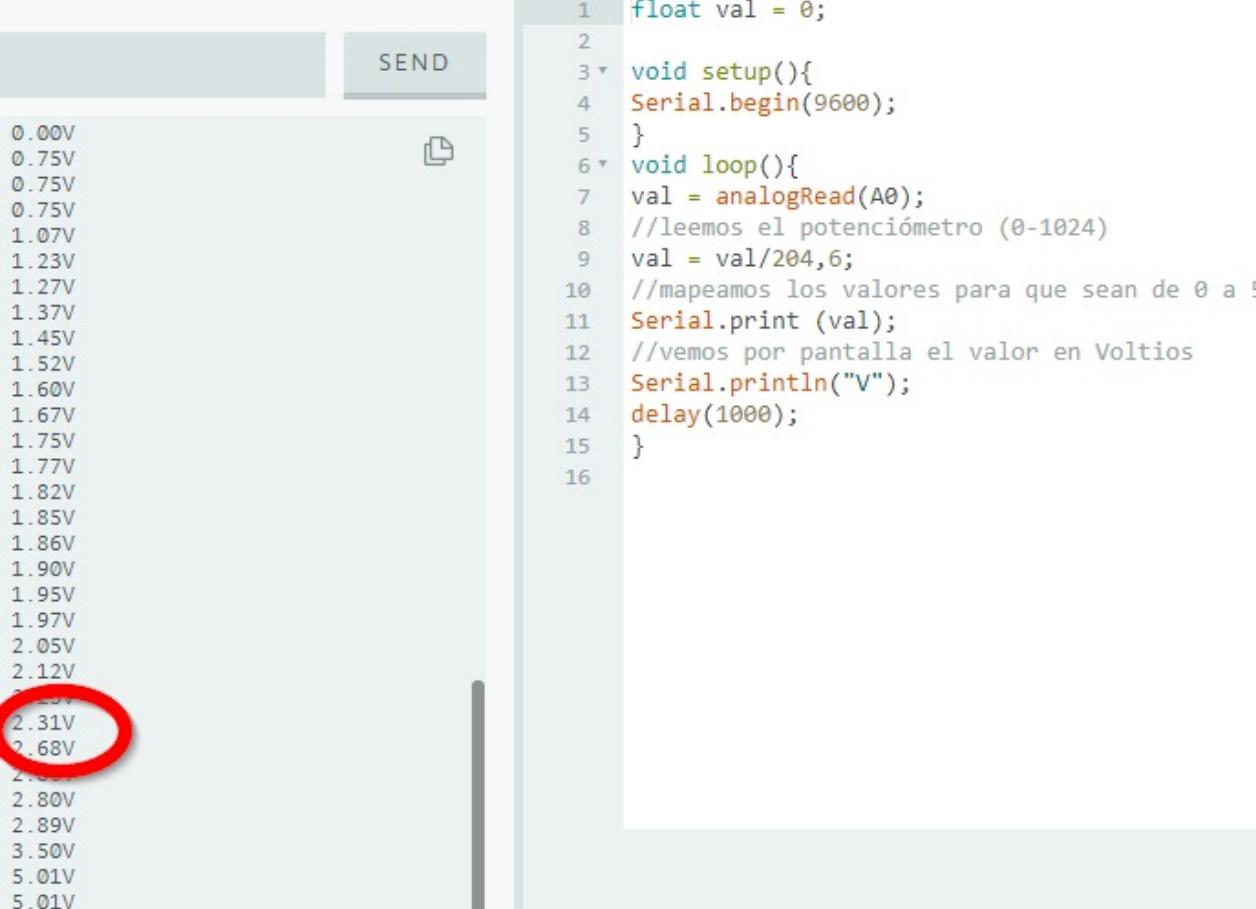
1 float val = 0;
2
3 void setup(){
4     Serial.begin(9600);
5 }
6 void loop(){
7     val = analogRead(A0); //leemos el potenciómetro (0-1024)
8     val = val/204.6; //mapeamos los valores para que sean de 0 a 5V
9     Serial.print (val);
10    Serial.print ("V"); //vemos por pantalla el valor en Voltios
11    Serial.println();
12    delay(1000);
13 }
```

Abre el Monitor serial de Arduino (ctrl+may+M) y verás el voltaje que está ofreciendo Arduino al montaje que has hecho.

Mueve el potenciómetro y verás que el valor va cambiando. Intenta averiguar cual es la tensión umbral a partir de la cual tu led empieza a emitir luz.

Nota: inicialmente la corriente puede ser muy baja por lo que debes fijarte bien cuando empieza a iluminarse.

Verás que alrededor **de 2.5V** el led empieza a iluminarse



```

1 float val = 0;
2
3 void setup(){
4   Serial.begin(9600);
5 }
6 void loop(){
7   val = analogRead(A0);
8   //leemos el potenciómetro (0-1024)
9   val = val/204,6;
10  //mapeamos los valores para que sean de 0 a 5
11  Serial.print (val);
12  //vemos por pantalla el valor en Voltios
13  Serial.println("V");
14  delay(1000);
15 }
16

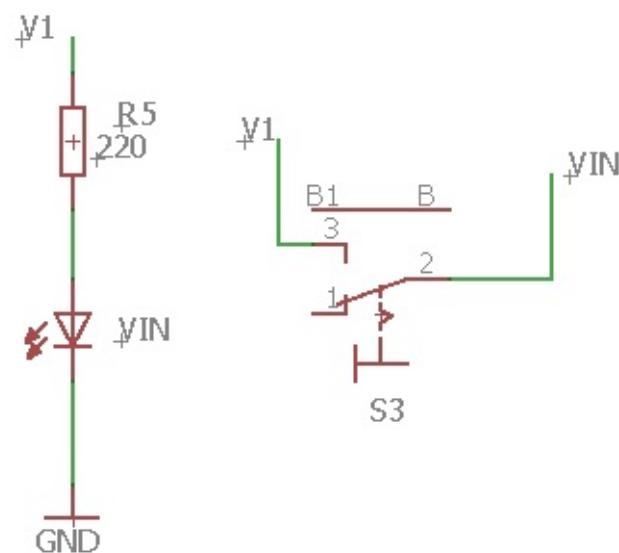
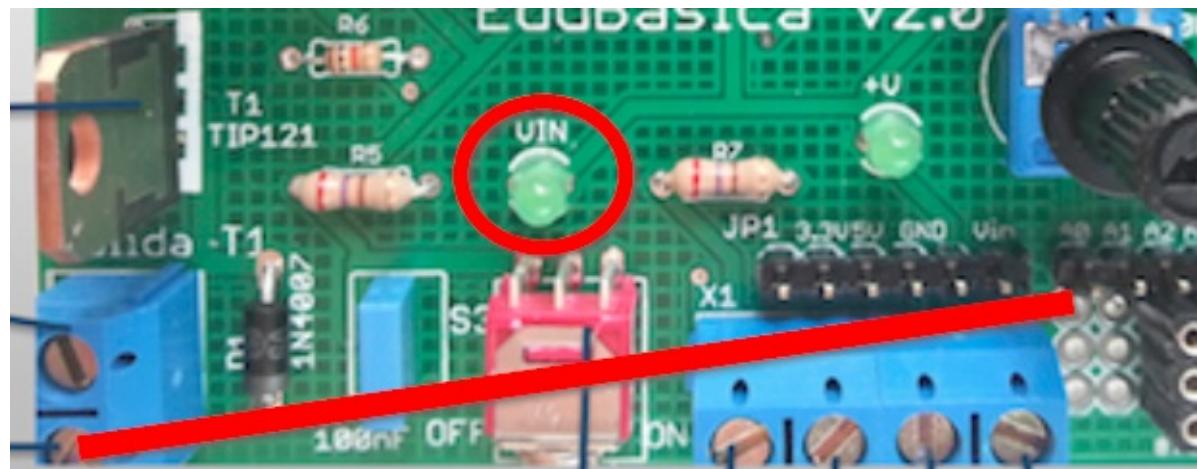
```

0.00V
0.75V
0.75V
0.75V
1.07V
1.23V
1.27V
1.37V
1.45V
1.52V
1.60V
1.67V
1.75V
1.77V
1.82V
1.85V
1.86V
1.90V
1.95V
1.97V
2.05V
2.12V
2.15V
2.31V **2.68V**
2.70V
2.80V
2.89V
3.50V
5.01V
5.01V

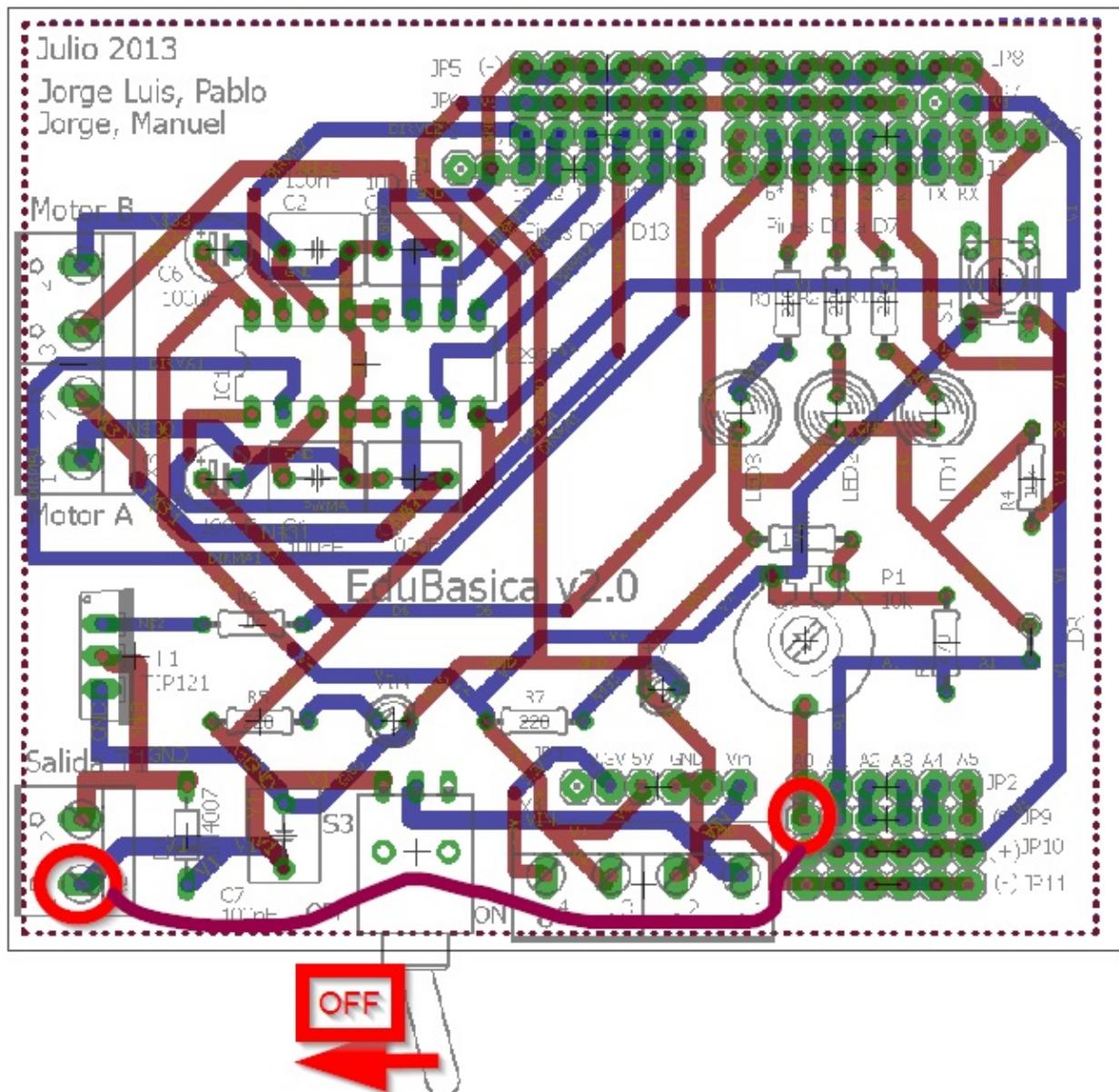
TRUCO

Se puede hacer **sin ningún circuito exterior**, utilizando la resistencia R5 y **el diodo de VIN**, para ello bastaría:

- Desconectar el interruptor que une V1 con Vin
- Unir con un cable V1 con A0

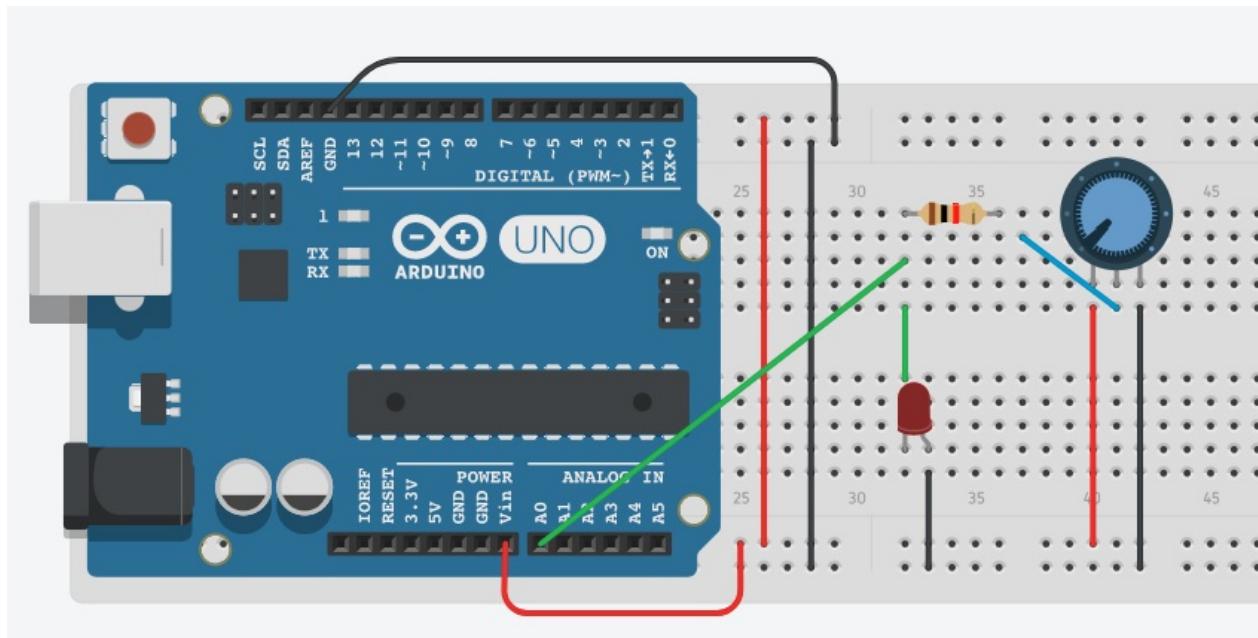


Poner interruptor en OFF y unir el cable uniendo V1 con A0 (dibujado en morado) :



SIN EDUBASICA

Igual, simplemente que tienes que hacer el circuito del potenciómetro exteriormente



Condensadores



Dos conductores cualesquiera separados por un aislante constituye un condensador. Este tipo de dispositivos eléctricos tienen gran variedad de usos como filtrar señales, eliminar ruido eléctrico o almacenar carga eléctrica entre otros.

En casi todas las aplicaciones prácticas cada conductor se encuentra inicialmente descargado y al conectarlos a una batería, mediante transferencia de carga de la batería a los conductores, van adquiriendo una cierta carga (dicho proceso se denomina carga del condensador). En todo momento, ambos conductores tienen igual carga pero de signo opuesto de tal forma que entre ambos conductores existe un campo eléctrico y por tanto una diferencia de potencial que se opone a la externa responsable de su carga. El proceso de carga del condensador se detiene cuando la diferencia de potencial entre los conductores del mismo se iguala a la de la batería.

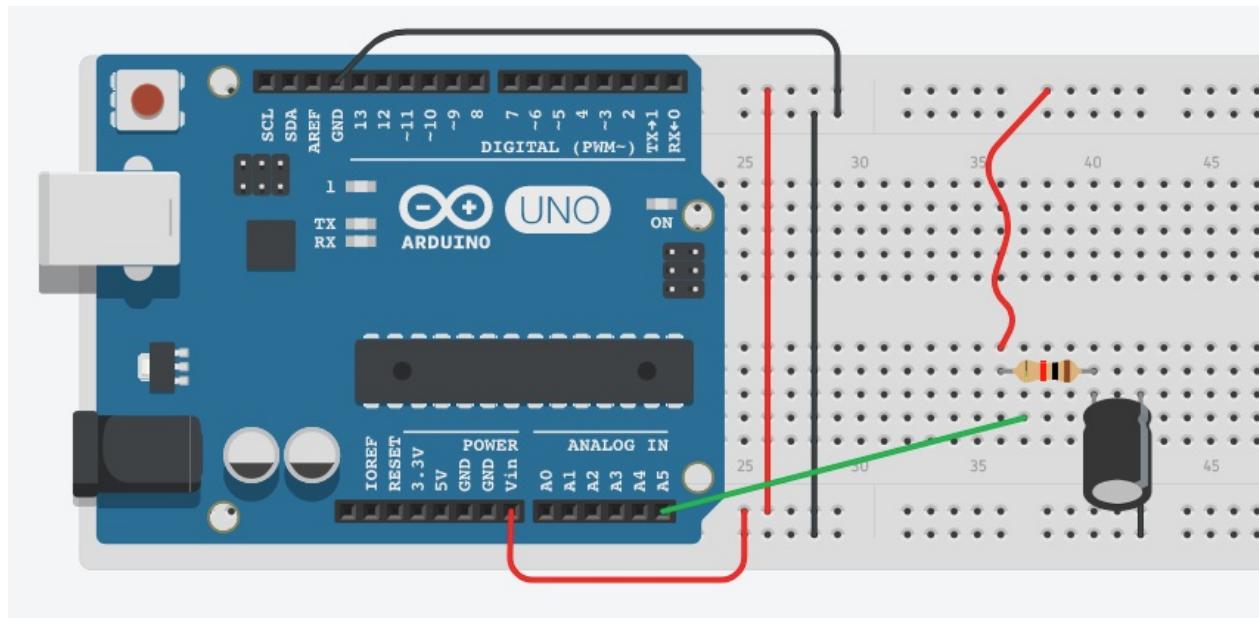
Hay que resaltar que aunque cada placa se encuentra cargada, la carga neta del condensador sigue siendo nula, puesto que la carga de cada conductor es igual a la del otro pero con signo contrario. Es por ello que cuando se dice que un conductor tiene una carga Q realmente lo que se está diciendo es que tiene una carga $+Q$ en el conductor que se encuentra a mayor potencial y una carga $-Q$ en el conductor a menor potencial (supuesto Q positiva).

Montaje 8: Carga de un condensador

El objetivo de esta práctica es **visualizar la carga de un condensador** aprovechando que Arduino puede leer los valores y podemos transmitirlo al ordenador por el puerto serie, usaremos el Arduino como una **capturadora de datos** y así de forma pedagógica enseñamos cómo es la carga.

SIN EDUBÁSICA

Hacemos una carga del condensador moviendo el cable rojo curvo desde masa a la alimentación. La unión entre resistencia y condensador lo conectamos a A5.



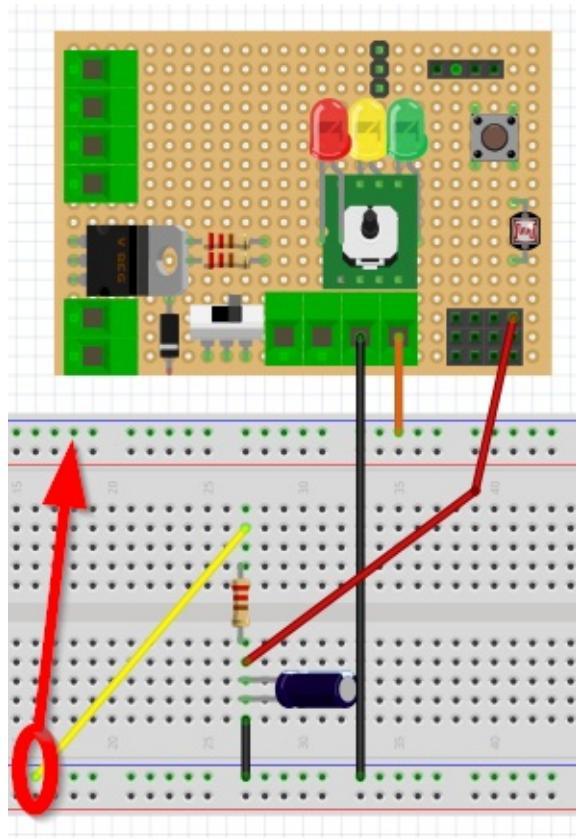
OJO: LA PATA (-) DEL CONDENSADOR TIENE QUE IR A GND (peligro de explosión si el condensador es electrolítico y si es muy grande)

El cable **amarillo** inicialmente lo conectamos a GND y luego procederemos a cargar el condensador soltandolo de GND y conectándolo a 5V

El cable **rojo** mide la tensión del condensador y lo mapearemos por A5 del Arduino.

CON EDUCASICA

(realmente la EDUBASICA en esta práctica no simplifica el cableado)



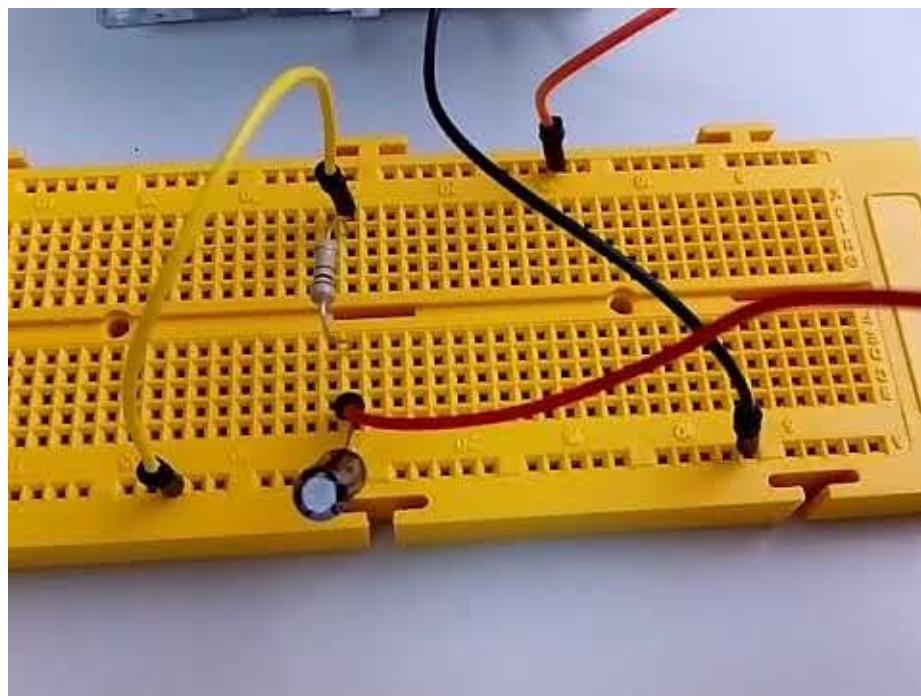
Continuamos

Y ejecutamos el siguiente código:

```

1 float sensorPin = A5;
2 float sensorValue = 0;
3
4 void setup() {
5
6     Serial.begin(9600);
7 }
8
9 void loop() {
10    sensorValue = analogRead(sensorPin);
11    sensorValue = sensorValue/204.6; //lo mapeamos a Voltios
12    Serial.println(sensorValue);
13    delay(100);
14 }
```

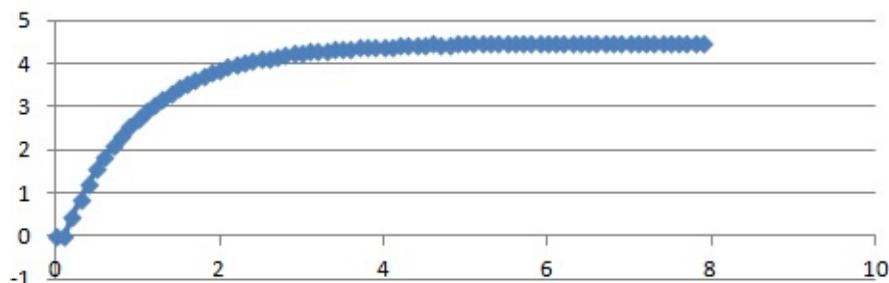
Por el monitor serie se van visualizando los valores, ver el vídeo:



[Video link](#)

La ventaja de utilizar el Arduino, y no el polímetro, es que podemos copiar los valores y pegarlos en una hoja de cálculo, y así visualizar la curva de carga del condensador:

Carga



En este caso, se ha utilizado una resistencia de **100k** y un condensador de **10uF** por lo tanto el tiempo de carga es $T = 5RC = 5$ seg que es lo que aproximadamente refleja la gráfica. En tu kit de robótica para hacer este curso tienes unos valores parecidos.

Nota con los valores X: Como la instrucción delay se ha puesto el valor delay(100) por lo tanto hay 0.1segundo entre número y numero, por lo tanto los valores de la X tienen que ser 0, 0.1, 0.2, 0.3 ... en la hoja de cálculo, en la casilla A3 es = A2+0.1 y A1 tiene el valor inicial 0 segundos.

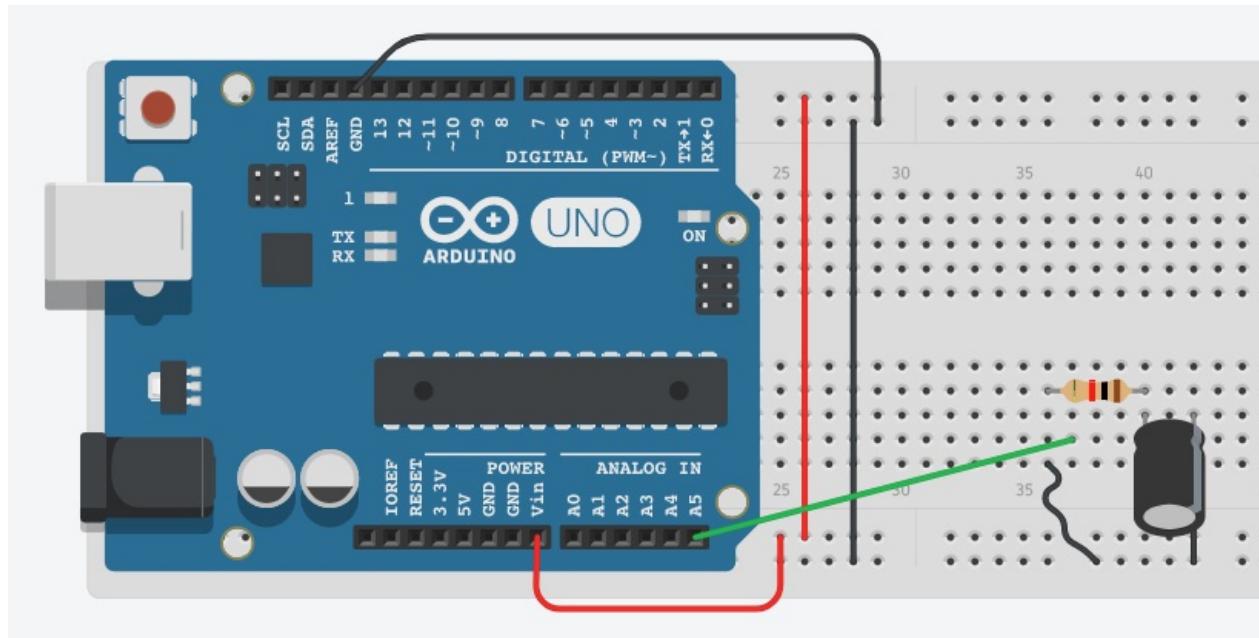
Nota con los valores Y: Si se pega los valores en una hoja de cálculo, no interpreta el carácter "." como una "," por lo que se aconseja utilizar una columna que convierta estos valores con la función en Excell, por ejemplo el valor en B2 puede ser : =VALOR(REEMPLAZAR(C2;2;1;"")) donde C es la columna donde se pegan los valores del monitor serie.

La hoja Excell que se ha utilizado te lo puedes descargar [aquí](#)

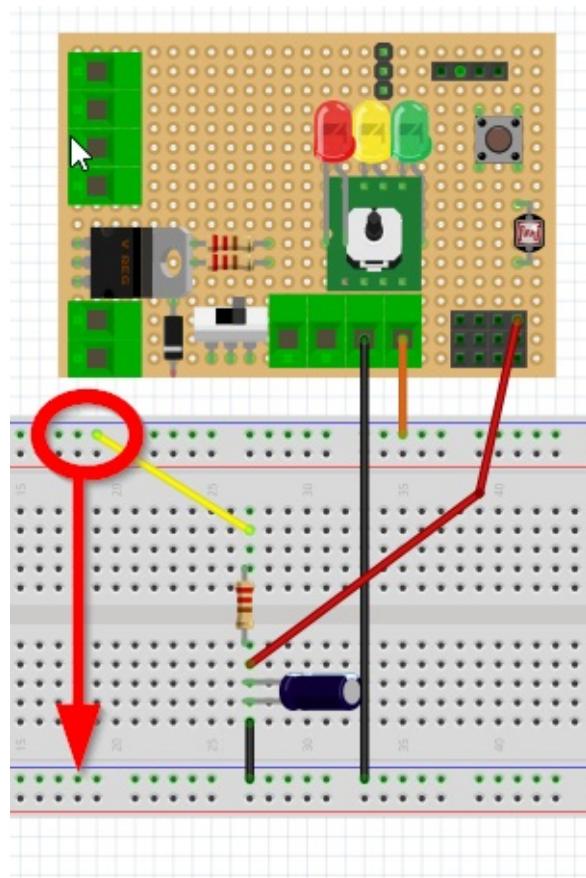
Montaje 9: Descarga del condensador

Si has hecho [el montaje anterior](#), terminarás con el condensador **totalmente cargado**, procede con el mismo programa que va leyendo A5 a desconectar el cable amarillo de 5V y conectarlo a GND entonces el condensador se irá descargando a masa.

SIN EDUBASICA



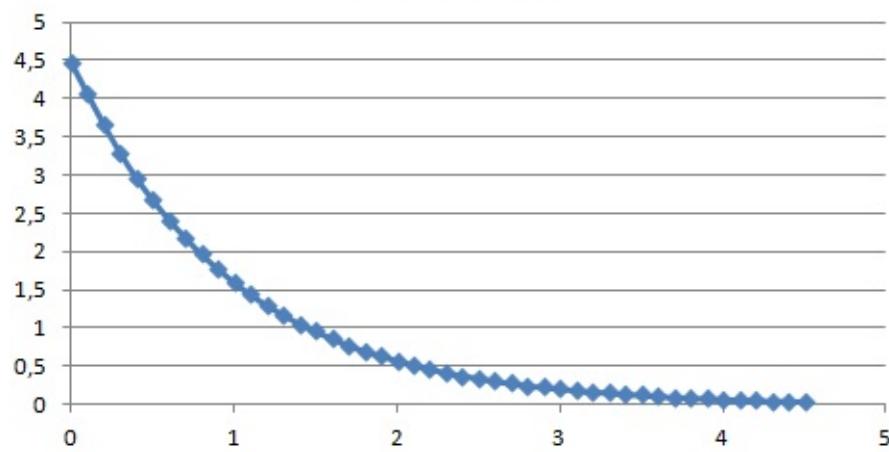
CON EDUBÁSICA



Continuamos ...

Por el monitor serie se van leyendo los valores, puedes copiarlos y pegarlos en la misma hoja excell y verás la gráfica de descarga del condensador, que con el misma fórmula $T=5RC$ nos sale aproximadamente 5 segundos de prácticamente el tiempo de descarga:

Descarga



Actividad

Pregunta Verdadero-Falso

Un condensador es un dispositivo que almacena carga eléctrica.

Verdadero Falso

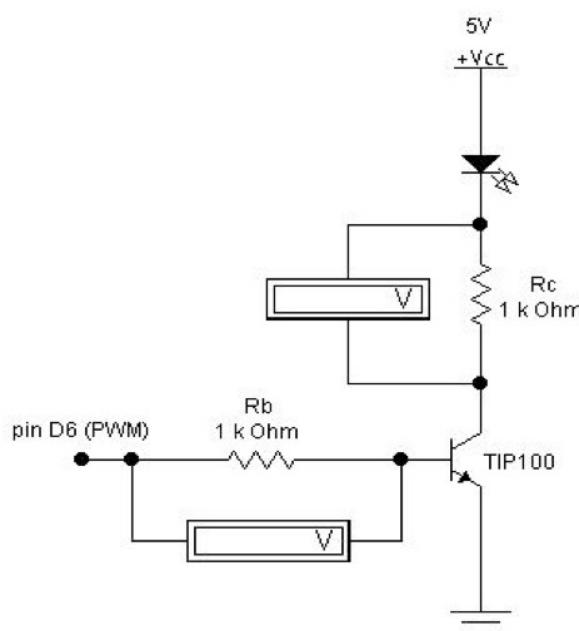
Las placas de un condensador están en contacto unas con otras sin nada que las separe.

Verdadero Falso Un condensador está formado por dos conductores separados por un material aislante.

En el programa anterior se observa que el condensador se descarga sobre un pin digital de Arduino.

Verdadero Falso En la línea `_sensorValue = analogRead(sensorPin);` vemos que la lectura es analógica, por lo tanto, uno de los contactos del condensador está conectado a un pin analógico (A5) y no a un pin digital.

Transistores



Ciertos dispositivos tecnológicos necesitan una corriente eléctrica mayor que la que proporciona la placa EduBásica y para ello se utiliza el transistor.

El transistor es sin duda el componente electrónico más importante. Está formado por 3 capas de material semiconductor, cada una con un terminal metálico para poder conectarlo a los circuitos. Los materiales semiconductores se pueden comportar como aislantes o conductores según la energía que reciban, de ahí su versatilidad en el campo de la electrónica. Los 3 terminales de los transistores son:

Colector: Entrada de corriente.

Base: Entrada de corriente. Regula el funcionamiento.

Emisor: Salida de corriente.

Según la forma en la que conectemos los terminales del transistor a resistencias externas éste puede funcionar de 3 formas:

El funcionamiento típico en circuitos de señales **ANALÓGICAS**

Como **funcionamiento en zona “activa”**: La resistencia conectada a la base del transistor tiene un valor que permite circular corriente a través de ella. De esta manera hay circulación de corriente entre el colector y emisor cuyo valor será proporcional a la corriente que circula por la base. Normalmente mucho mayor con lo que producirá el efecto de amplificación.

En resumen: El transistor actúa como un **amplificador de corriente**

La ganancia de corriente β también llamado **hfe**, se calcula dividiendo la corriente de colector con la corriente de base,

Los funcionamientos típicos de circuitos **DIGITALES**

Como **interruptor abierto o en “corte”**: Si la corriente que circula por la base es nula, normalmente debido a que se conecta a ella una resistencia de un valor muy elevado, el transistor no permite el paso de corriente entre colector y emisor.

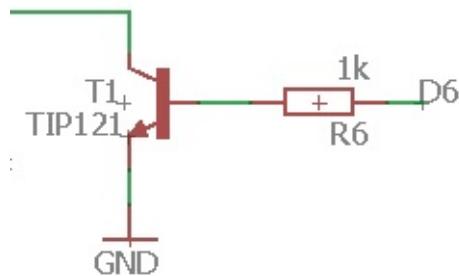
Como **interruptor cerrado en zona de “saturación”**: Si se va aumentando la intensidad que circula por la base llega un momento que la intensidad entre colector y emisor ya no aumenta más; es decir, se satura.

CONEXIONES

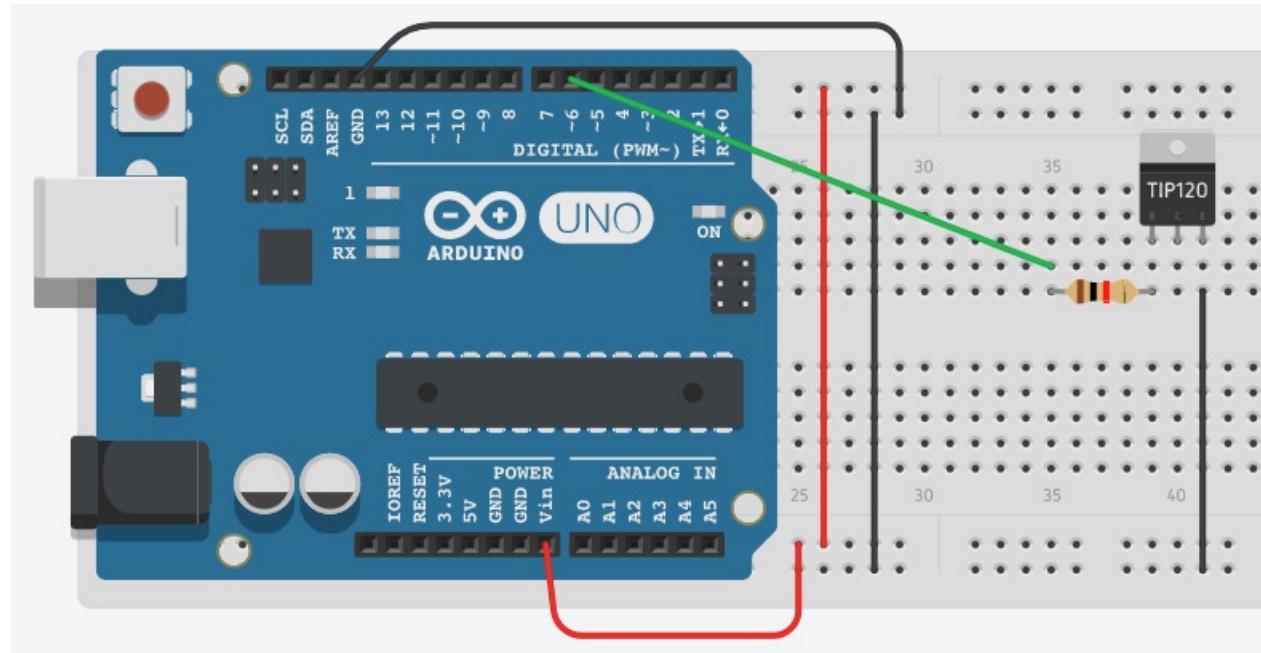
Vamos a definir las conexiones para los siguientes montajes del transistor, utilizaremos tenemos un [TIP121 o TP120](#) es un transistor NPN tipo "Darlington" es decir, está integrado por dos transistores interiormente.

SIN EDUBASICA

Realizaremos el siguiente esquema, conectando la base al D6 y dejando el colector abierto para los experimentos:



por lo tanto:



CON EDUBASICA

En la shield ya tenemos el siguiente esquema:

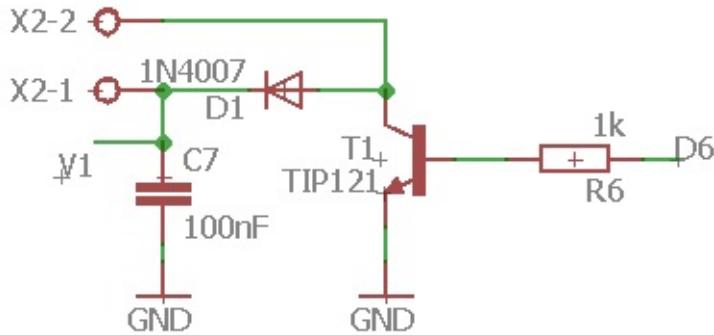
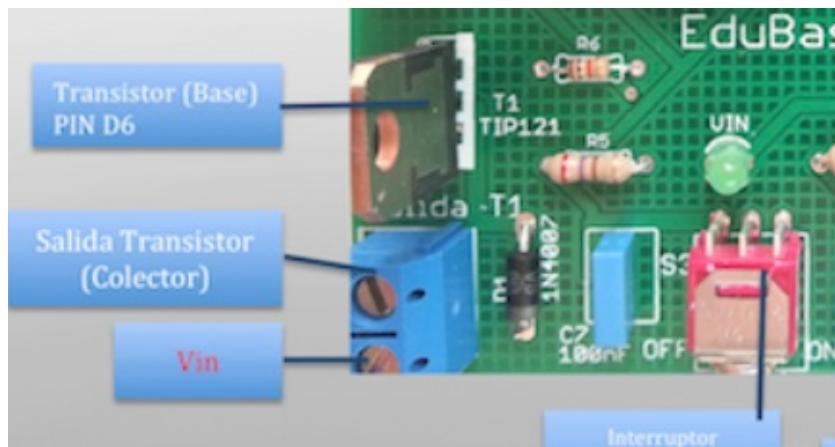


Imagen - Esquema del transistor en EDUBASICA, en nuestro caso real R6 es 10k

Los conectores X2-2 y X2-1 son los de abajo a la izquierda. **X2-2 es el colector** del transistor y **X2-1 es V1** (aunque en la foto pone Vin):



V1 y Vin está conectado por el interruptor según este esquema:

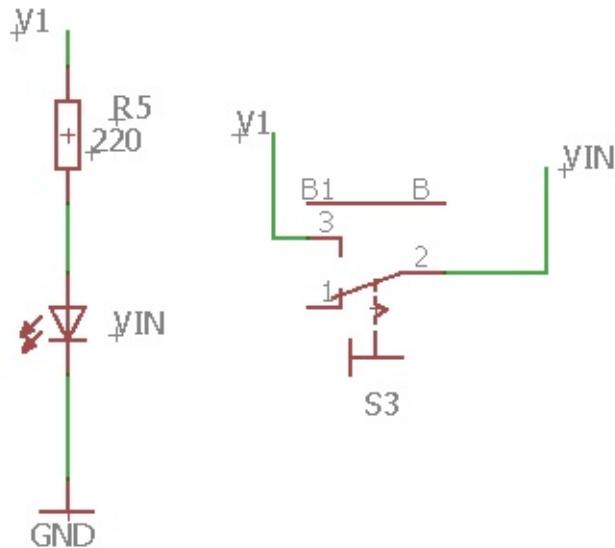


Imagen - Vin y V1 en EDUBASICA

Luego es mejor que el interruptor esté conectado en ON para unir V1 con Vin y cortar el diodo D1, en caso contrario se nos va la corriente de la resistencia de colector por D1 a R5 y al led engañando la medida.

Vin son aproximadamente 5V de potencia de entrada al arduino que se utilizan si no lo alimentamos por el USB.

Montaje 10: saturación y corte

Podemos jugar con el transistor en los estados saturación y corte, para ello vamos a crear una variable n que es un contador (en estado inicial n=1 y en los sucesivos se incrementa una unidad n++), si el contador es par (el resto de n/2 = 0 o sea n%2==0) la intensidad de la corriente base sea 0 y en caso contrario que sea máxima. Pondremos un retardo de 3 segundos para visualizar bien los valores que medimos.

El programa sería:

```

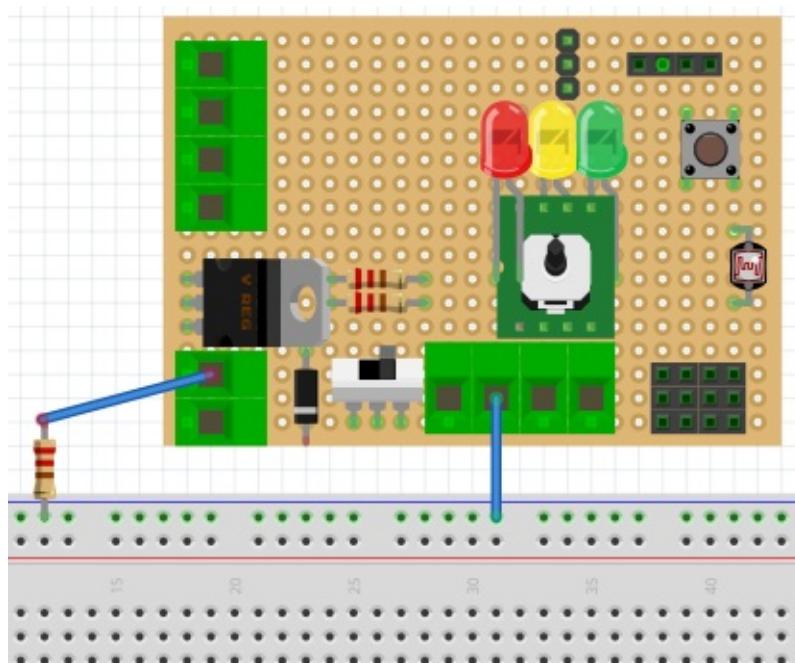
1 int ledPin = 6;
2 int potenPin = A0;
3 int intensity, valor_poten;
4 int n;
5
6 void setup() {
7 pinMode(ledPin, OUTPUT);
8 Serial.begin(9600);
9 n=1;
10 }
11
12 void loop() {
13 if (n%2==0) intensity = 0;
14 else intensity = 255;
15 analogWrite(ledPin,intensity); //Envia una onda PWM especificado en la
16 variable: intensity.
17 Serial.print("n=");
18 Serial.print(String(n));
19 Serial.print("\tintensity = ");
20 Serial.print(String(intensity));
21 Serial.print("\n");
22 n++;
23
24 delay (3000);
}

```

El programa es un poco "tremendo" ¿hay alguna manera de simplificarlo?

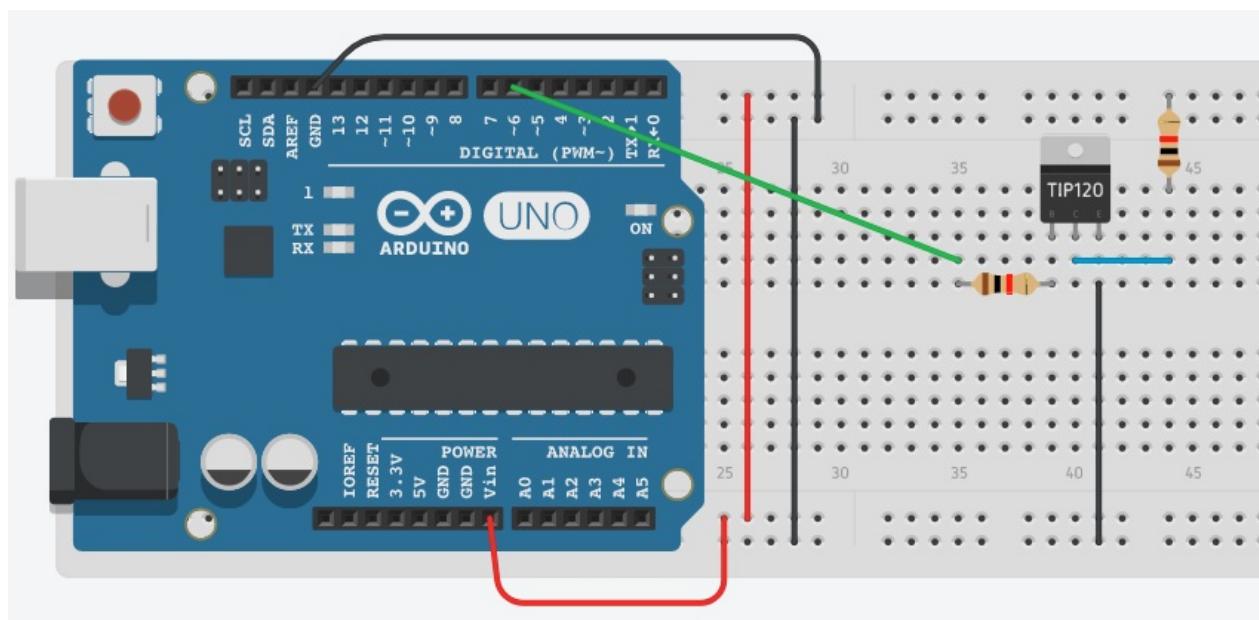
CON EDUBASICA

Pondremos una resistencia de colector de 1k para forzar una corriente de saturación, pero como desde el conector X2-2 hasta V+ hay mucha distancia para conectar los dos extremos de la resistencia, utilizaremos la placa Protoboard:

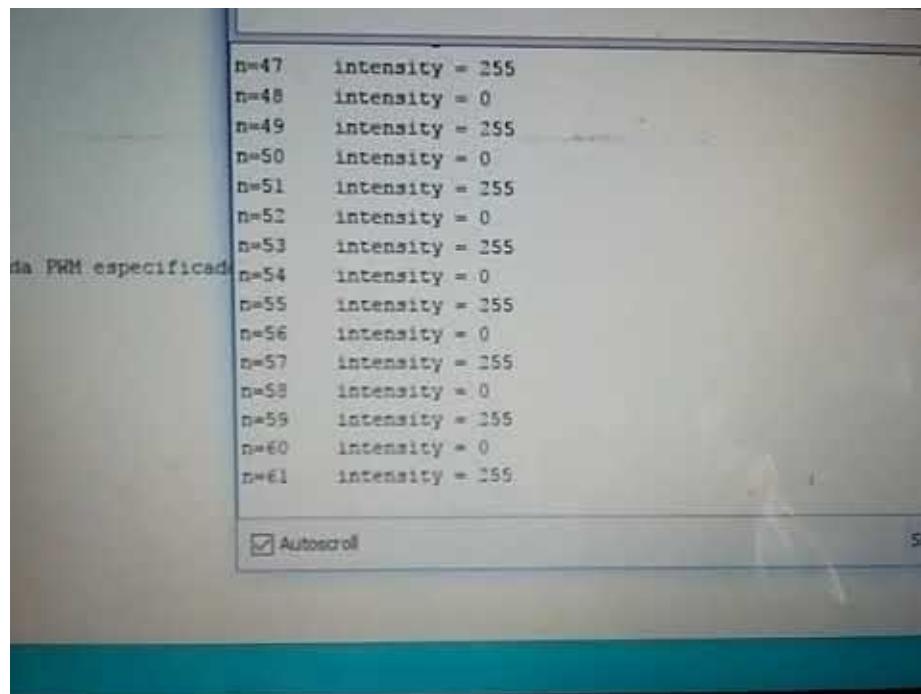


Mediremos entre colector y masa, y vemos que alternativamente pasa de los estado corte (casi 5V) a saturación (casi 0V).

SIN EDUBASICA



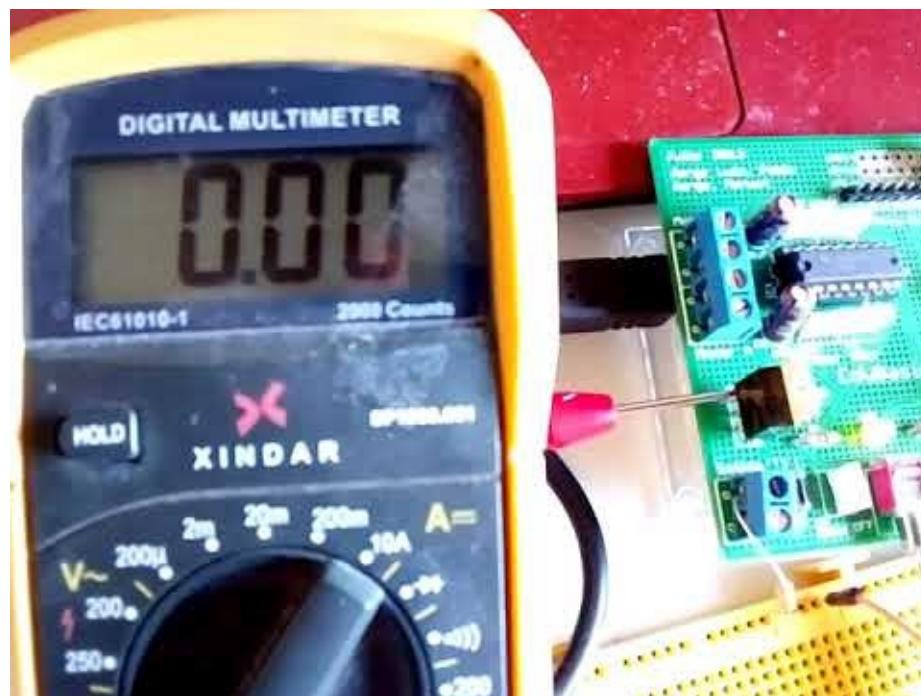
RESULTADO



```
n=47    intensity = 255
n=48    intensity = 0
n=49    intensity = 255
n=50    intensity = 0
n=51    intensity = 255
n=52    intensity = 0
n=53    intensity = 255
n=54    intensity = 0
n=55    intensity = 255
n=56    intensity = 0
n=57    intensity = 255
n=58    intensity = 0
n=59    intensity = 255
n=60    intensity = 0
n=61    intensity = 255
```

[Video link](#)

Si medimos la tensión en la base, vemos que pasa de 0V a 1.28V :



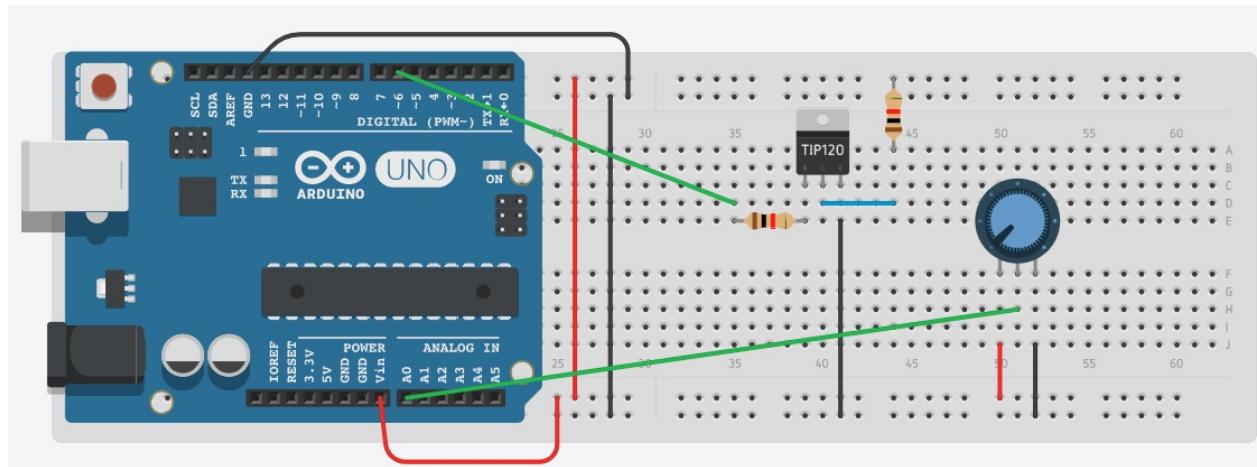
[Video link](#)

Montaje 11: zona activa

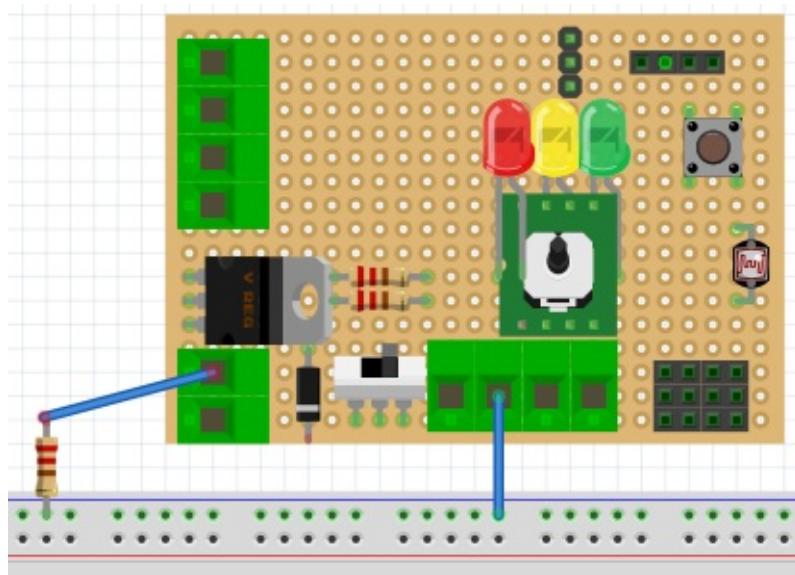
En esta práctica el transistor va a actuar como amplificador de corriente, dentro de la zona activa para ello se va a realizar el siguiente montaje:

SIN EDUBASICA

Ponemos una resistencia en el colector y un potenciómetro en A0:

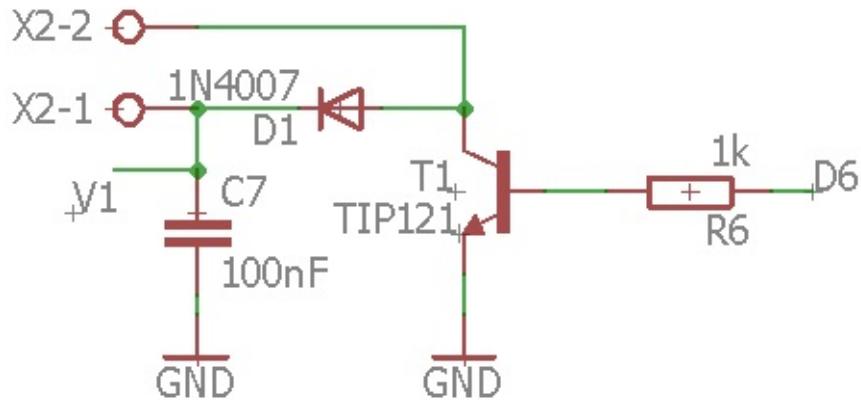


CON EDUBASICA



Continuamos...

El funcionamiento de este circuito es el siguiente: a partir de potenciómetro que tenemos en la placa EduBásica controlamos la salida PWM del pin D6 variando la tensión aplicada a la resistencia de base R_b que en Edubásica es R6:



Y sin edubásica mediremos en la resistencia conectada en la base.

Utilizaremos el siguiente código:

```

1 int ledPin = 6;
2 int potenPin = A0;
3 int intensity, valor_poten;
4 int n;
5
6 void setup() {
7 pinMode(ledPin, OUTPUT);
8 Serial.begin(9600);
9 n=1;
10 }
11
12 void loop() {
13     valor_poten=analogRead(potenPin);
14     intensity = map(valor_poten, 0, 1024, 0, 255);
15     analogWrite(ledPin,intensity);
16     Serial.print("valor analógico leído=");
17     Serial.println(analogRead(valor_poten));
18     delay (1000);
19 }
```

El resultado es que podemos tener el transistor dentro de la zona activa manejando el potenciómetro, en el siguiente vídeo se muestra el polímetro que mide la tensión del colector y como varía de **corte** (casi 5V) a **saturación** (casi 0V) pasando por la zona **activa**:

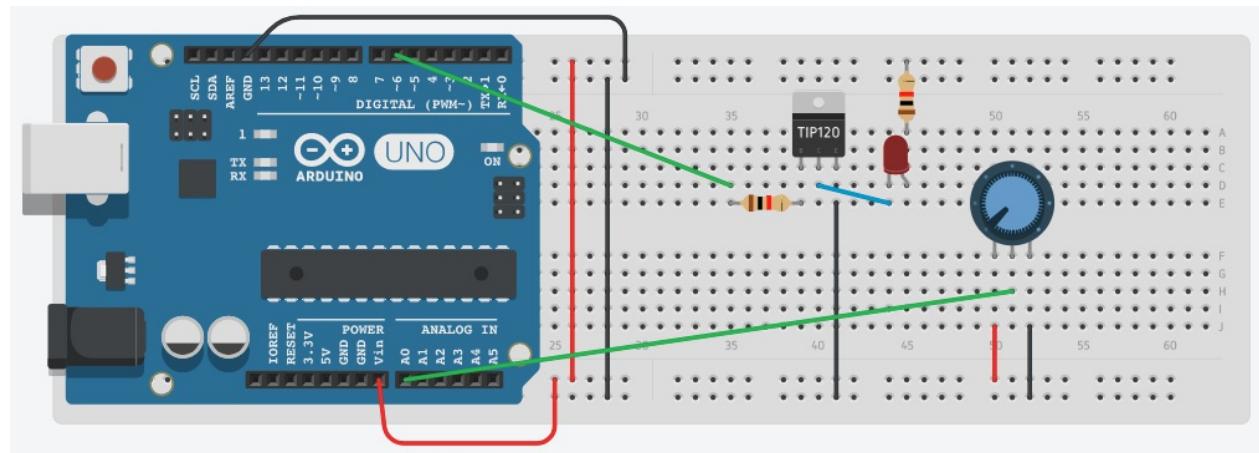


[Video link](#)

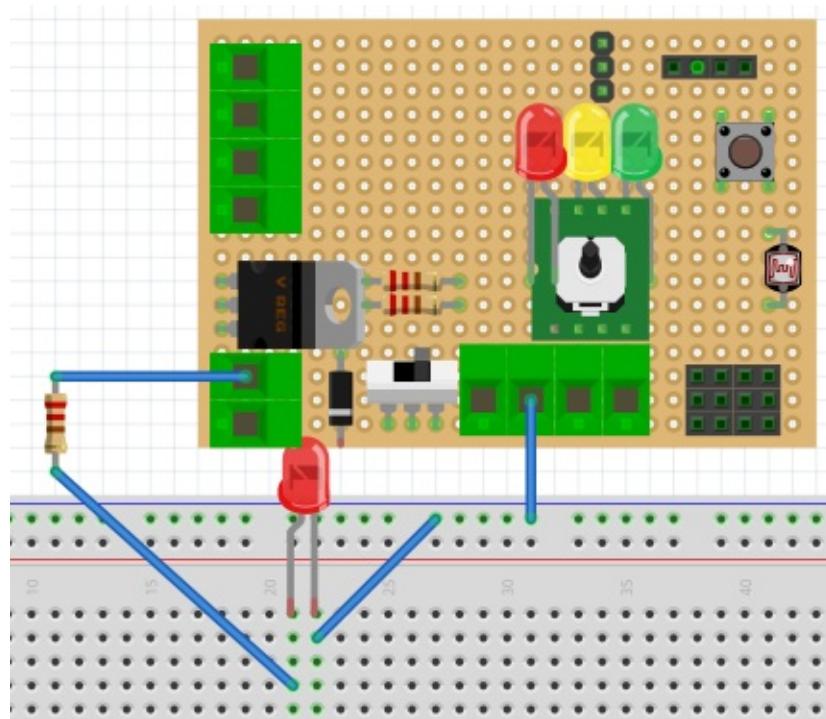
Montaje 12: Con un diodo se ve bien

Podemos insertar un diodo en el colector y se visualiza bien las zonas:

SIN EDUBASICA



CON EDUBASICA

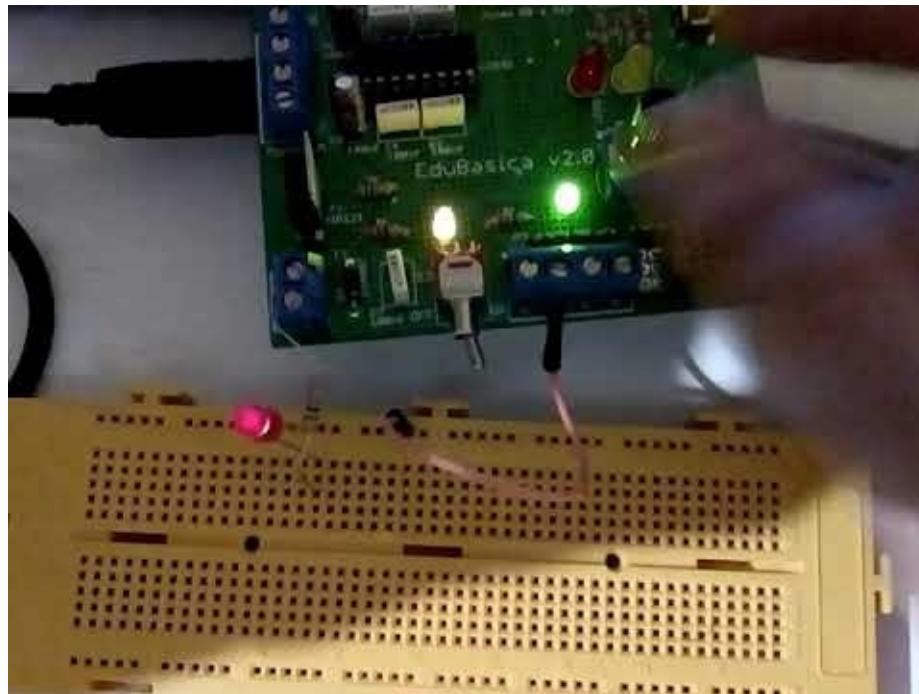


Continuamos

El programa es el mismo que el anterior y se puede ver en el vídeo las tres formas de funcionamiento del transistor. Se trata de ir variando el valor del potenciómetro R1, con el fin de conseguir que el diodo led pase de estar apagado (corte), a encendido en diferentes grados de luminosidad, llegando finalmente hasta su máxima luminosidad (saturación).

La mayor o menor intensidad del diodo led nos indicará la mayor o menor corriente que fluye por el colector (IC) al variar la corriente de base (IB).

Resultados



[Video link](#)

Mediante un polímetro se pueden medir los valores de tensión. Para medir estas corrientes recurriremos, como en otras ocasiones, al polímetro en forma de voltímetro y aplicar la ley de Ohm.

Encendido del LED	V _{BE}	V _{CE}	V _{RB}	V _{RC}	I _C	I _B	β
Apagado							
Luz mínima							
Luz media							
Luz máxima							

A nosotros nos ha salido [resultados](#) (xlsx - 15,87 KB)

	V _{be}	V _{ce}	V _{rb}	I _c (mA)	I _b (mA)	h _{fe}
LED APAGADO	0	3,3	0	1,7	0	#DIV/0!
LUZ MINIMA	0,03	3,14	0,09	1,86	0,009	207
LUZ MEDIA	0,18	2,28	0,57	2,72	0,057	48
LUZ MAXIMA	1,27	0,58	3,52	4,42	0,352	13

Montaje 13: cálculo de hfe

Con el mismo circuito anterior, se mide la tensión en los terminales de R_b (que está en la placa) y R_c (la que hay que montar en un protoboard) completando la siguiente tabla:

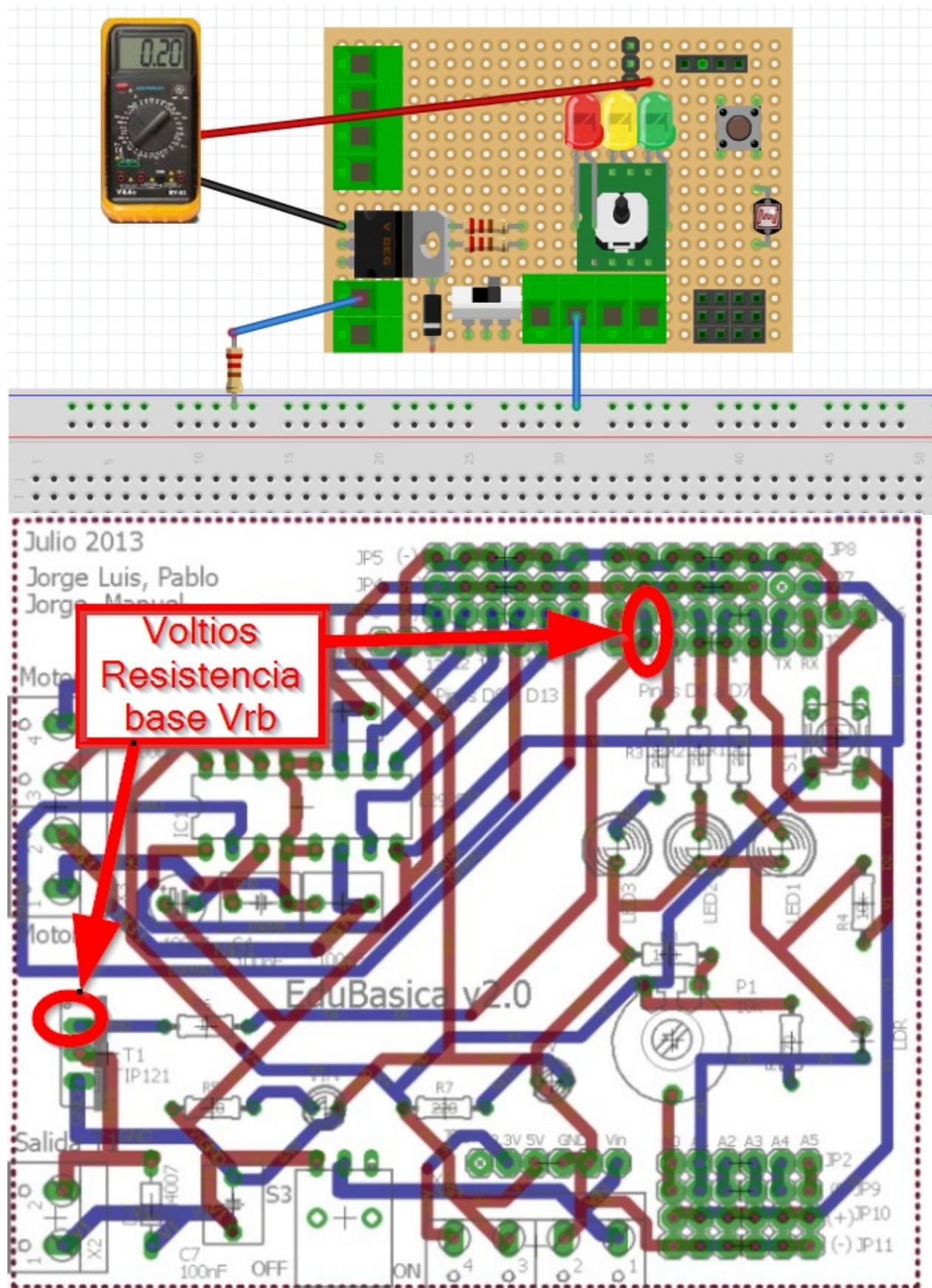
V _{rb}	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2	2,2	2,4	2,6	2,8	3	3,2	3,4	3,6	R _b =10k
V _{rc}																				R _c =1k
I _b mA)																				I _b =V _{rb} /R _b
I _c (mA)																				I _c =V _{rc} /R _c

Se varía el potenciómetro, progresivamente para conseguir las tensiones en la resistencia de base que figuran en la tabla Una vez obtenidos todos los valores, calcular la Intensidad de colector y la intensidad de la base con estas fórmulas

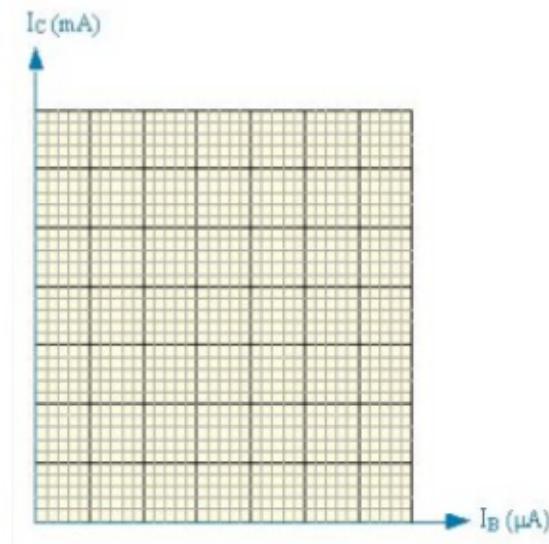
$$I_B = \frac{V_{RB}}{R_B} \quad I_C = \frac{V_{RC}}{R_C}$$

que en el caso de Edubásica R_b es R6 y son 10k

Creemos que la mejor forma de leer V_{rb} es desde los extremos de la base del transistor y D6:



Podemos llevarlos a una gráfica y trazar la curva característica $I_c = f(I_b)$



La ganancia de corriente β se calcula:

$$\beta = \frac{I_C}{I_B}$$

Según la [DataSheet](#) de este transistor te tiene que salir alrededor de 1.000, nosotros hemos realizado la práctica y nos sale 13 ¡no lo entendemos! :(

[[resultados](#) (xlsx - 15,87 KB)]

Recta de carga de un transistor

Se trata de comprobar de forma práctica el efecto de amplificación de corriente de un transistor, así como de determinar su punto de trabajo para un determinada corriente de base y la recta de carga. El circuito es el mismo que la práctica anterior.

La siguiente expresión se corresponde con la ecuación de la recta de carga:

$$I_C = \frac{V_{CC} - V_{CE}}{R_C}$$

Para dibujar esta recta sobre la curva característica determinaremos primero sus extremos ($I_C = 0$ y $V_{CE} = 0$):

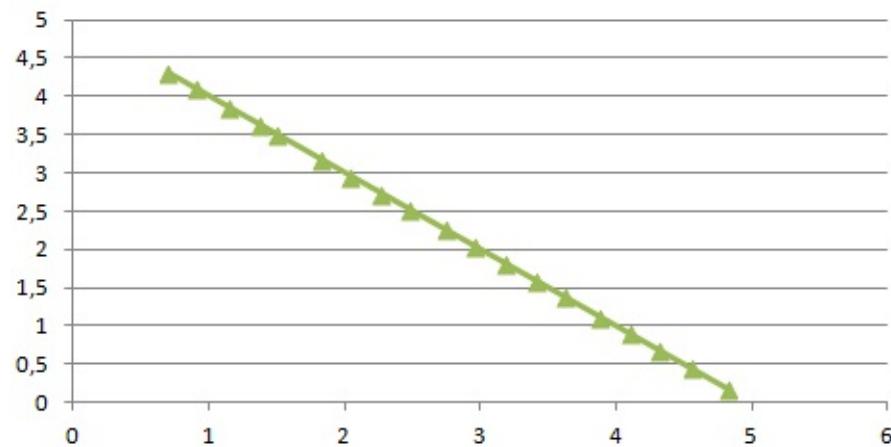
Para $V_{CE} = 0$; $I_C = \frac{V_{CC}}{R_C}$

Para $I_C = 0$; $V_{CE} = V_{CC}$

Te proponemos realizar los cálculos teóricos y dibujar la recta de carga en una gráfica I_C (eje y) y V_{CE} (eje x) y sobre esta gráfica representar los puntos I_C V_{CE} que has medido experimentalmente, utilizando las fórmulas

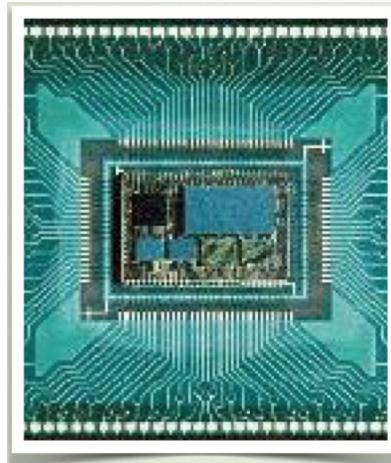
Te puede salir una cosa así:

Ic frente a Vc



Vrb	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2	2,2	2,4	2,6	2,8	3	3,2	3,4	3,6	Rb=10k
Vrc	0,18	0,45	0,69	0,9	1,12	1,38	1,59	1,81	2,05	2,26	2,53	2,73	2,96	3,18	3,5	3,63	3,85	4,1	4,3	Rc=1k
Ib mA)	0	0,02	0,04	0,06	0,08	0,1	0,12	0,14	0,16	0,18	0,2	0,22	0,24	0,26	0,28	0,3	0,32	0,34	0,36	Ib=Vrb/Rb
Ic (mA)	0,18	0,45	0,69	0,9	1,12	1,38	1,59	1,81	2,05	2,26	2,53	2,73	2,96	3,18	3,5	3,63	3,85	4,1	4,3	Ic=Vrc/Rc
hfe		22,5	17,3	15	14	13,8	13,3	12,9	12,8	12,6	12,7	12,4	12,3	12,2	12,5	12,1	12	12,1	11,9	hfe=Ic/Ib
Vce	4,82	4,55	4,31	4,1	3,88	3,62	3,41	3,19	2,95	2,74	2,47	2,27	2,04	1,82	1,5	1,37	1,15	0,9	0,7	Vce=Vcc-Vrc

Electrónica digital



Entendemos por electrónica digital, la que se encarga de sistemas electrónicos en los que sólo existen 2 valores o estados: verdadero/falso, "0" ó "1".

En este capítulo trabajaremos los conceptos principales del Álgebra de Boole y de las puertas lógicas que utilizaremos para diseñar circuitos. Utilizaremos Arduino y la placa EduBásica.

Conocimiento previo

- Programación básica de Arduino.

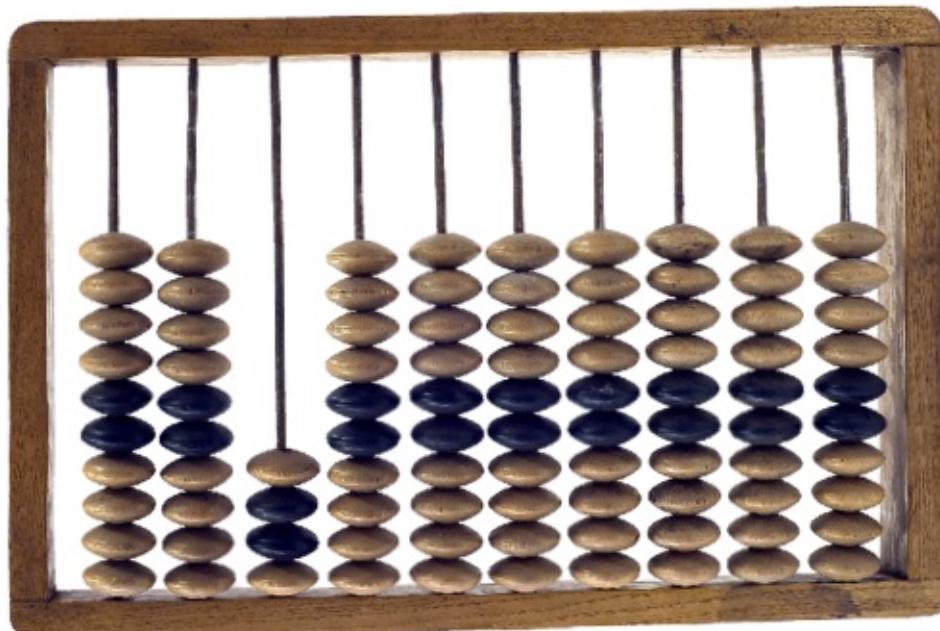
Objetivos

- Conocer las propiedades del Álgebra de Boole.
- Operar con variables binarias.

Lista de materiales:

- Placa Arduino.

Álgebra de Boole



- **Magnitudes analógicas:** Tienen un número infinito de valores, por ejemplo, todas las magnitudes físicas como temperatura, velocidad, electricidad, tiempo, etc ...
- **Magnitudes digitales:** Consideraremos señales digitales binarias que son las que sólo toman dos valores discretos: el uno o el cero. Representarán estados “activados” o “desactivados” Por ejemplo una bombilla puede estar encendida o apagada.

Para poder trabajar con datos binarios, el matemático inglés George Boole (1815-1864) creó una estructura algebraica que consta únicamente de 2 elementos (bits). Una álgebra que seguro conoces y utilizas a diario es el álgebra elemental, basado en la aritmética decimal que consta de 10 elementos (números), operaciones (suma, resta,) y propiedades (comutativa...). Toda álgebra consta de esos 3 apartados: elementos, operaciones y propiedades. El álgebra de boole comprende:

- Elementos: 0 y 1.
- Operaciones: multiplicación, suma, negación.
- Propiedades: conmutativa, asociativa, distributiva, elemento negado.

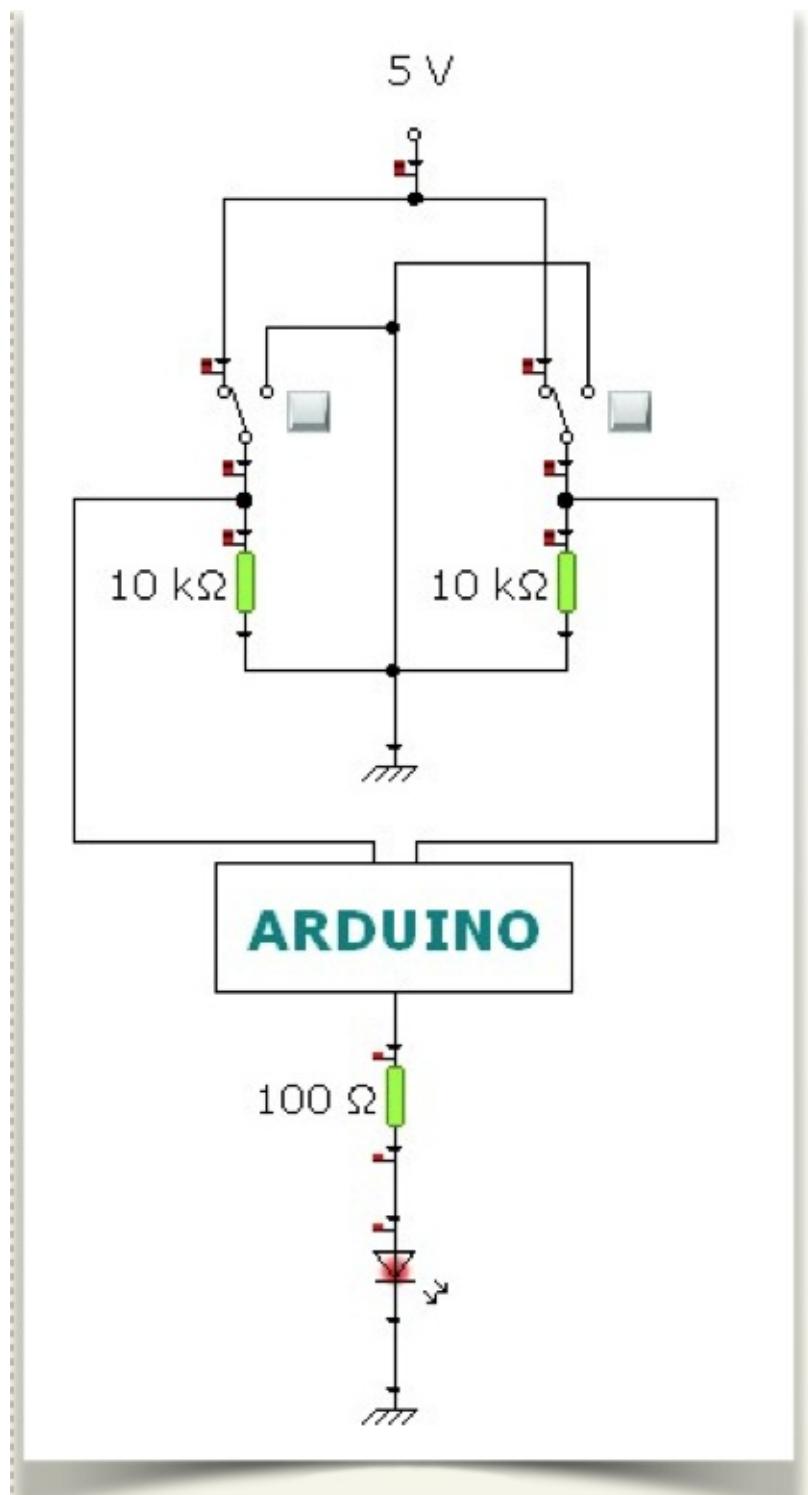
A los elementos de un circuito eléctrico se pueden asociar estados de “1” ó “0” si están encendidos o apagados y cerrados (conectados) o abiertos (desconectados) respectivamente.

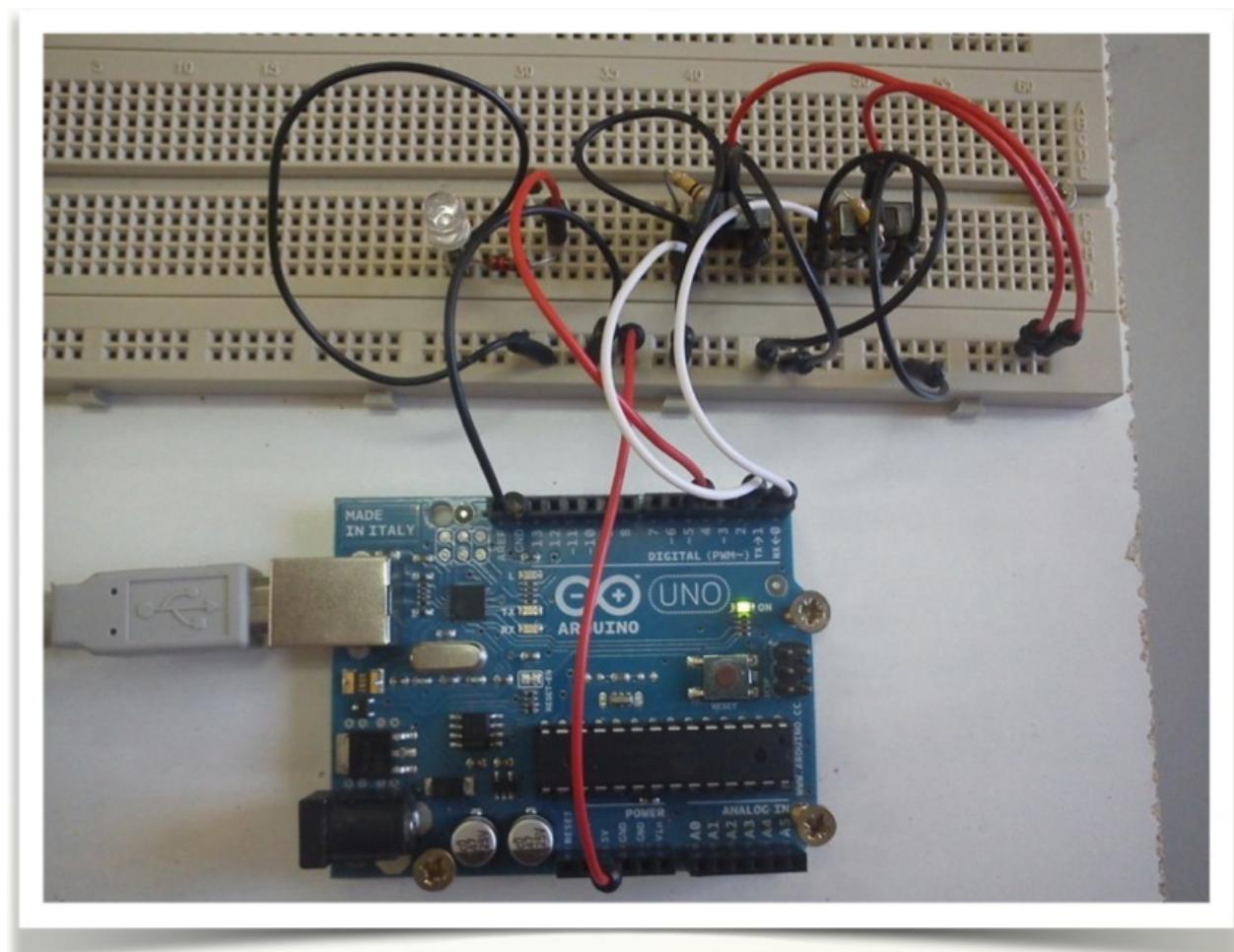
Montaje 1 AND sin EDUBASICA

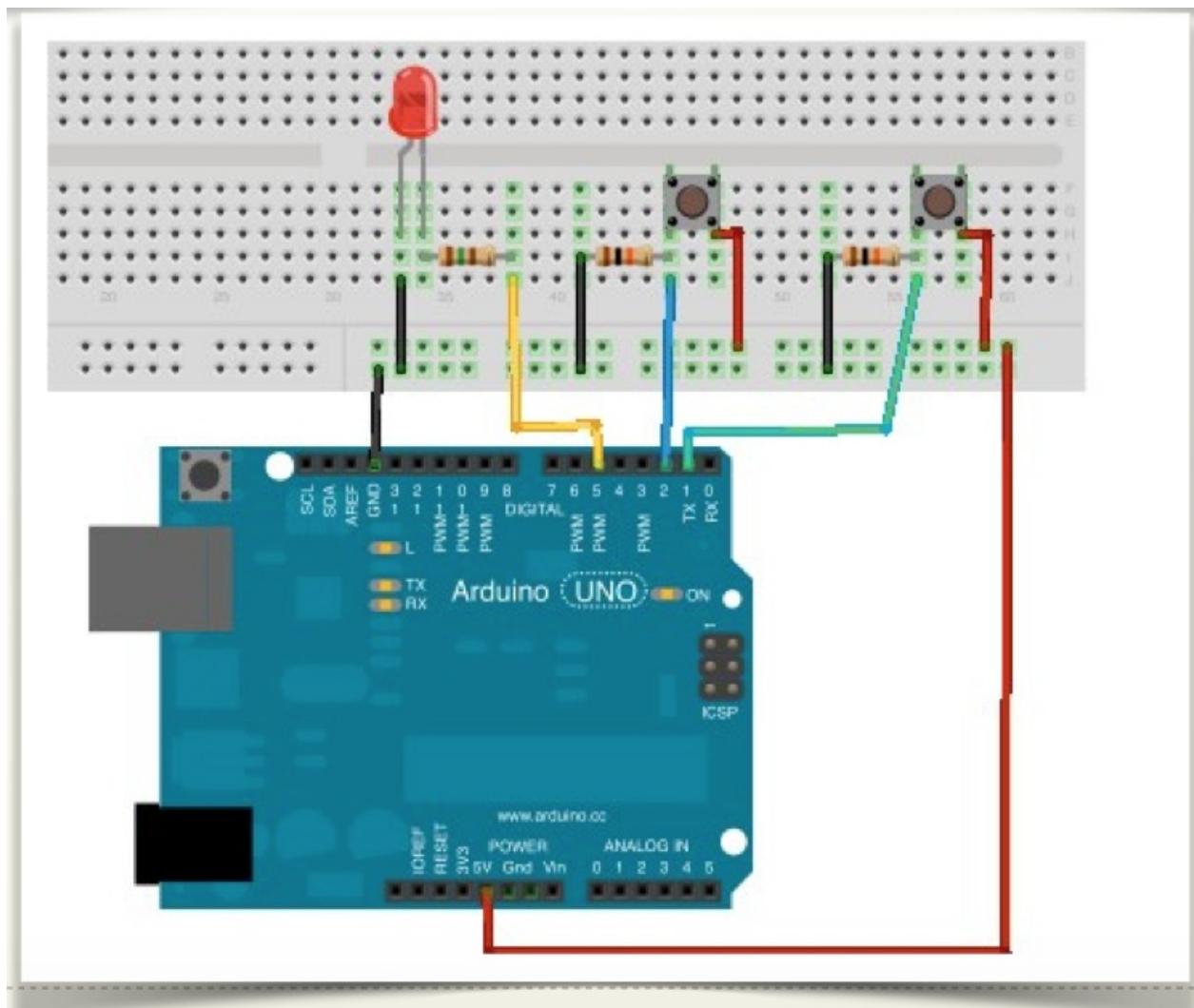
PULSADORES:

Vamos a simular dos entradas lógicas (“1” ó “0”) con dos pulsadores, (pueden ser conmutadores). En este ejemplo usaremos la función AND de manera que, según el estado de las 2 entradas, obtendremos una señal de salida (“1” ó “0”) conforme a la tabla de verdad de la operación. Si te animas puedes montar el circuito tú mismo en una protoboard siguiendo este esquema:

Las entradas están en los pines digitales 1 y 2. Y la salida del sistema es un led (en pin 5) que estará encendido/apagado según el resultado de aplicar la función AND a las 2 variables de entrada.







PROGRAMA:

```

1 /*
2  *Boole
3  *Función AND con 2 variables
4 */
5
6 int var1 = 7; //Pin de entrada del pulsador 1
7 int var2 = 2; //Pin de entrada del pulsador 1
8 int led = 5; //Pin de salida para el led(rojo)
9 int estado1 = 0; //Para almacenar el estado de la variable1
10 int estado2 = 0; //Para almacenar el estado de la variable2
11 int resultado = 0; //Para almacenar el resultado
12
13 void setup() {
14     pinMode(var1, INPUT); //Iniciliza el pin de entrada 1 como salida
15     pinMode(var2, INPUT); //Iniciliza el pin de entrada 2 como salida
16     pinMode(led, OUTPUT); //Iniciliza el pin del led como salida
17 }
18
19 void loop(){
20

```

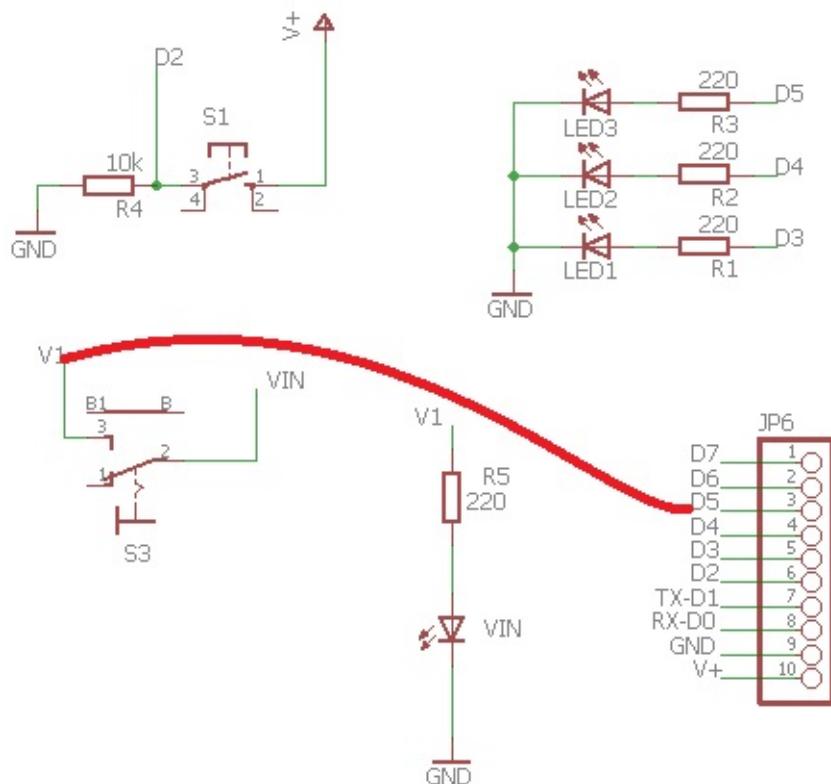
```
21  estado1 = digitalRead(var1); //Lee el estado del botón y lo almacena
22  estado2 = digitalRead(var2); //Lee el estado del botón y lo almacena
23  resultado = (estado1 && estado2); //Función AND con los dos estados
24  digitalWrite(led, resultado); //Escribimos el resultado en el led
}
```

Montaje 1bis AND con EDUBASICA

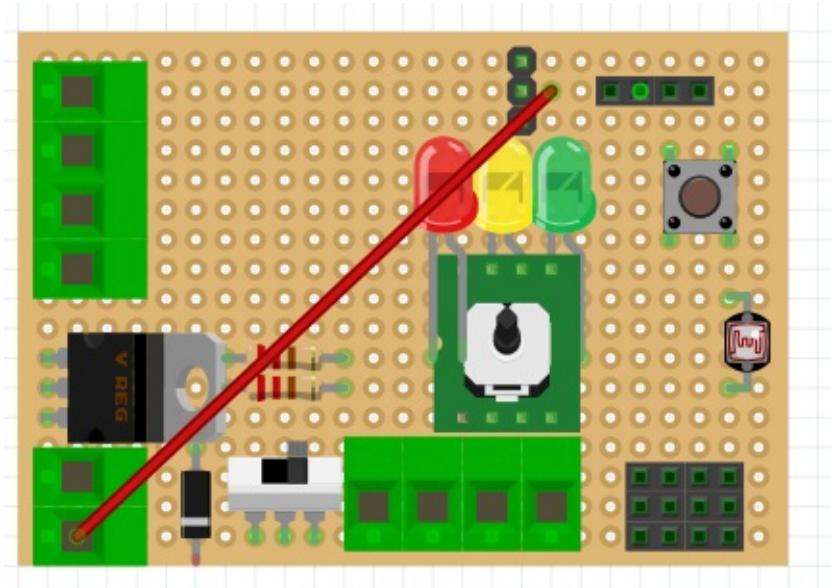
En este caso, para ahorrar cableado, vamos a utilizar:

- Como variable 1 de entrada el pin D2 que ya tiene el **pulsador**, y lo visualizaremos en el pin D4 que tiene el **LED VERDE**.
- Como variable 2 de entrada el pin D5 que lo conectaremos con un cable a V1 que tiene el **interruptor** y ya se visualiza en el **LED AMARILLO**.
- Como variable de salida el pin D3 que es el **LED ROJO**.

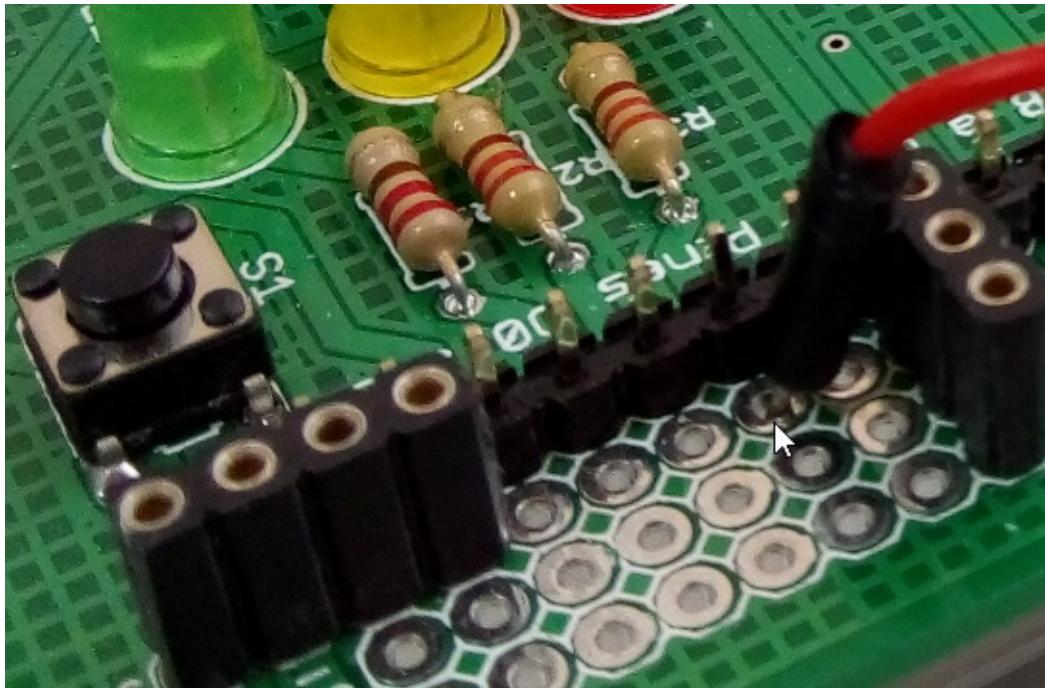
El esquema sería el siguiente:



Y el esquema de conexiones es muy fácil, conectar V1 con el agujero D5 de JP6:



Aquí tenéis el detalle de dónde está el agujero D5



El código sería:

```

1  /*
2   * Boole
3   * Función AND con 2 variables
4   */
5   ///////////////// entradas /////////////////////////////////
6   /////////////////
7   int var1 = 2;           //Pin D2 de entrada del pulsador 1
8   int var2 = 5;           //Pin D5 de entrada del pulsador 1 y led(rojo), conecta
9   mos V1 con D5 con un cable
10  int ledvari = 4;        //Pin D4 de salida para el var1 led(amarillo)
11  ///////////////////// salidas /////////////////////////////////
12  /////////////////////
13  int ledsalida = 3;      //Pin de salida para el led(verde)
14  /////////////////////
15  /////////////////////
16  int estado1 = 0;         //Para almacenar el estado de la variable1
17  int estado2 = 0;         //Para almacenar el estado de la variable2
18  int resultado = 0;       //Para almacenar el resultado
19
20 void setup() {
21     pinMode(var1, INPUT);      //Iniciliza el pin de entrada 1 como salida
22     pinMode(var2, INPUT);      //Iniciliza el pin de entrada 2 como salida
23
24     pinMode(ledvari, OUTPUT);   //Inicialliza led de var1 como salida
25     ////////////////////////////// no hace falta inicializar D5 como salid
26 a, pues estará con un cable visualizando V1
27     pinMode(ledsalida, OUTPUT); //Iniciliza el pin del led como salida
28 }
29
30 void loop(){

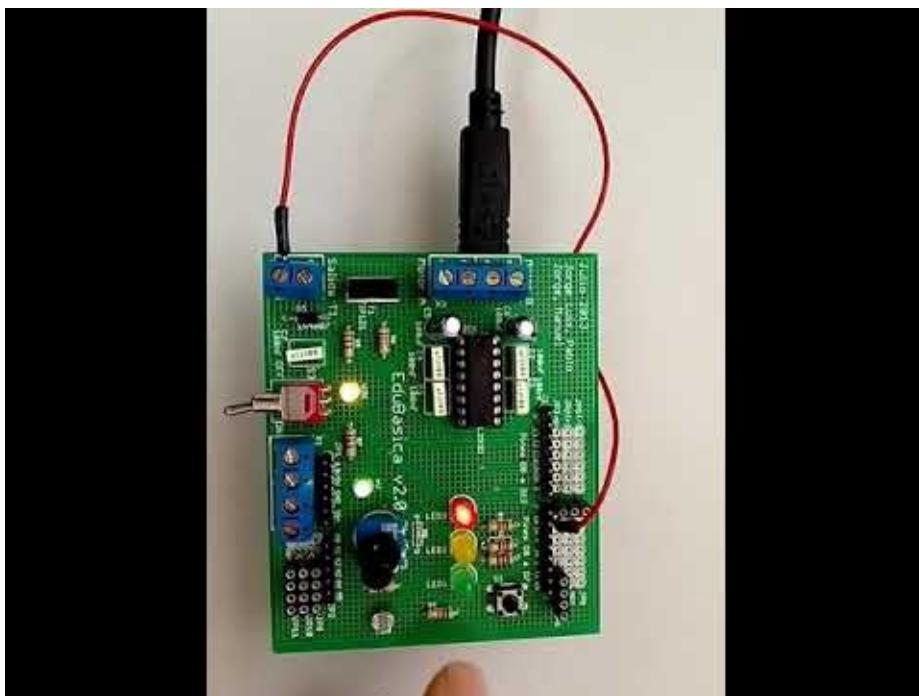
```

```
31  estado1 = digitalRead(var1);      //Lee el estado del botón y lo almacena
     digitalWrite(ledvar1, estado1);    //Se visualiza en el led amarillo la en
     trada var1
     estado2 = digitalRead(var2);      //Lee el estado del botón y lo almacena

     resultado = (estado1 && estado2); //Función AND con los dos estados
     digitalWrite(ledsalida, resultado); //Escribimos el resultado en el led

 }
```

Y el resultado es :



[Video link](#)

Montaje 2 ELEVADOR sin EDUBASICA

FUNCIONES LÓGICAS:

Tenemos un elevador neumático que se puede controlar desde 2 habitaciones contiguas del taller. Para que el elevador funcione debe estar accionado cualquiera de los 2 pulsadores existentes, pero por seguridad no funcionará si dos operarios la accionan simultáneamente.

Identificar las variables de entrada y función de salida:

Entradas:

- **A:** un pulsador
- **B:** un pulsador

Salida o valor de la función:

- **Motor que acciona el compresor del elevador**

TABLA DE VERDAD:

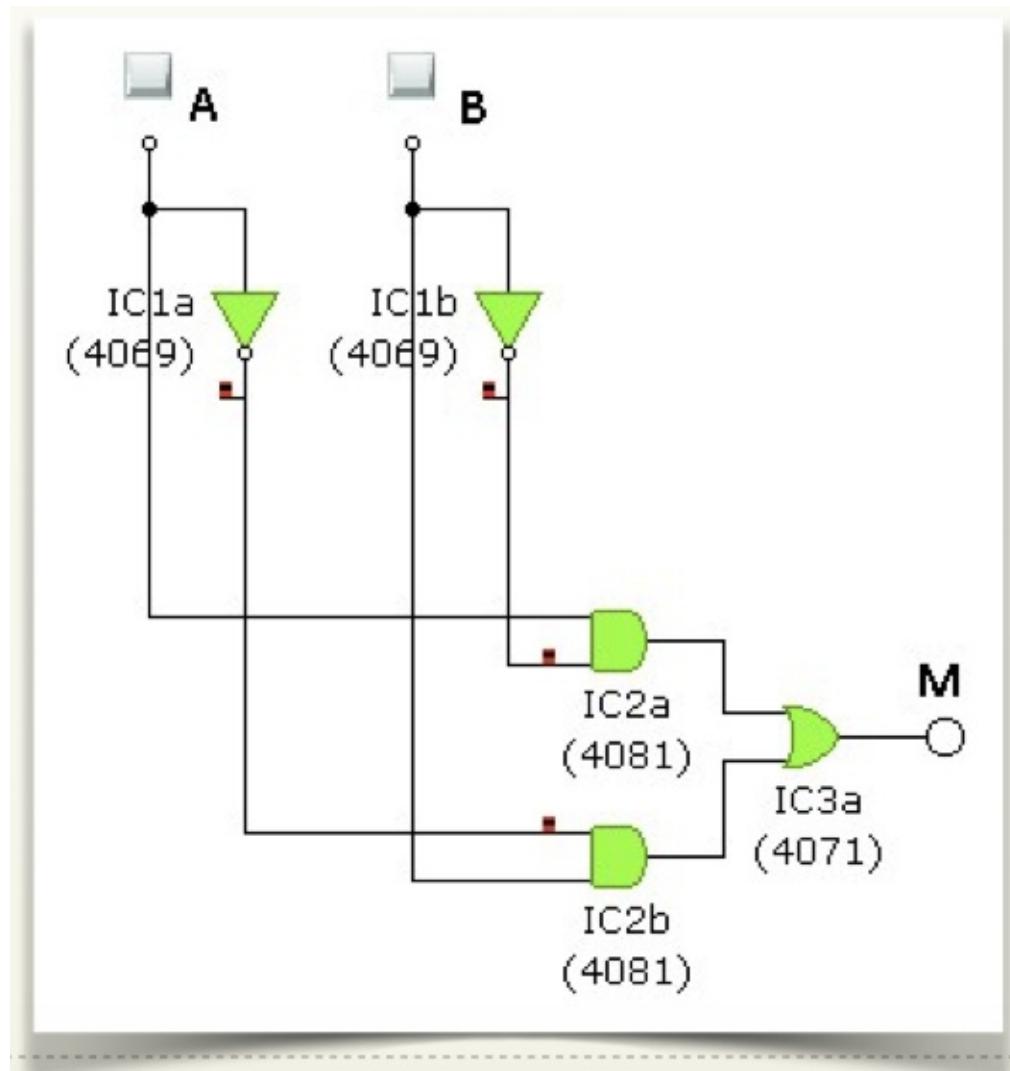
VARIABLES ENTRADA		FUNCTION SALIDA
A	B	M
0	0	0
0	1	1
1	0	1
1	1	0

Función Canónica: $M(FC1) = \text{NOT}(A)B + A\text{ NOT}(B)$

Puertas Lógicas:

- 2 puertas NOT
- 2 puertas AND
- 1 puerta OR

(Es decir, una [función XOR](#))



PROGRAMA:

```

1  /*
2   * Implementación de Función lógica
3   */
4
5  int var1 = 1;    //Pin de entrada del pulsador 1
6  int var2 = 2;    //Pin de entrada del pulsador 2
7  int led = 5;     //Pin de salida para el led(rojo)
8
9  int A = 0;       //Para almacenar el estado de la variable1
10 int B = 0;      //Para almacenar el estado de la variable2
11 int resultado = 0; //Para almacenar el resultado
12
13 void setup() {
14     pinMode(var1, INPUT);      //Init pin de entrada 1 como salida
15     pinMode(var2, INPUT);      //Init pin de entrada 2 como salida
16     pinMode(led, OUTPUT);      //Iniciliza el pin del led como salida
17 }
18
19 void loop(){

```

```
20 A = digitalRead(var1); //Lee el estado 1 y lo almacena
21 B = digitalRead(var2); //Lee el estado 2 y lo almacena
22 //Función Lógica ----- R=(A*B)+(A*B)
23 resultado = (!A && B) || (A && !B);
24 digitalWrite(led, resultado); //Escribimos el resultado en el led
25 }
```

Montaje 2bis ELEVADOR con EDUBASICA

Con EDUBASICA es utilizar el mismo esquema que [AND con EDUBASICA](#) pero cambiando la función AND por la XOR

```

1  /*
2   * Boole
3   * Función XOR con 2 variables
4   */
5  ///////////////// entradas /////////////////////////////////
6  /////////////////
7  int var1 = 2;          //Pin D2 de entrada del pulsador 1
8  int var2 = 5;          //Pin D5 de entrada del pulsador 1 y led(rojo), conecta
9  mos V1 con D5 con un cable
10 int ledvar1 = 4;        //Pin D4 de salida para el var1 led(amarillo)
11 ///////////////// salidas /////////////////////////////////
12 /////////////////
13 int ledsalida = 3;      //Pin de salida para el led(verde)
14 /////////////////
15 /////////////////
16 int A = 0;              //Para almacenar el estado de la variable1
17 int B = 0;              //Para almacenar el estado de la variable2
18 int resultado = 0;       //Para almacenar el resultado
19
20 void setup() {
21     pinMode(var1, INPUT);           //Iniciliza el pin de entrada 1 como salida
22     pinMode(var2, INPUT);           //Iniciliza el pin de entrada 2 como salida
23
24     pinMode(ledvar1, OUTPUT);        //Inicialliza led de var1 como salida
25     ////////////////////////////// no hace falta inicializar D5 como salida
26 a, pues estará con un cable visualizando V1
27     pinMode(ledsalida, OUTPUT);      //Iniciliza el pin del led como salida
28 }
29
30 void loop(){
31     A = digitalRead(var1);          //Lee el estado del botón y lo almacena
32     digitalWrite(ledvar1, A);        //Se visualiza en el led amarillo la entrada
33 var1
34     B = digitalRead(var2);          //Lee el estado del botón y lo almacena
35     resultado = (!A && B) || (A && !B);
36     digitalWrite(ledsalida, resultado); //Escribimos el resultado en el led
37
38 }
```

El resultado es:



[Video link](#)

Montaje 3 ALARMA EN VIVIENDA con EDUBASICA

Gracias a la lógica programable podemos programar alarmas muy complejas y mucho más eficientes que las alarmas convencionales. Las alarmas convencionales usan finales de carrera y, en definitiva, interruptores que activan una alarma. En nuestro caso vamos a dotar a la alarma de cierta lógica que nos proporcione mejores y más cómodos resultados. Las posibilidades son ilimitadas y depende de tu imaginación . En esta práctica y sólo como ejemplo vamos a suponer algunas cosas que, si bien no tienen por qué ajustarse a la realidad, sí que sirven como ejemplo para mostrar y dar a entender las posibilidades de la alarma. puerta, encender la luz y cerrar la puerta.

Partimos de las siguientes premisas :

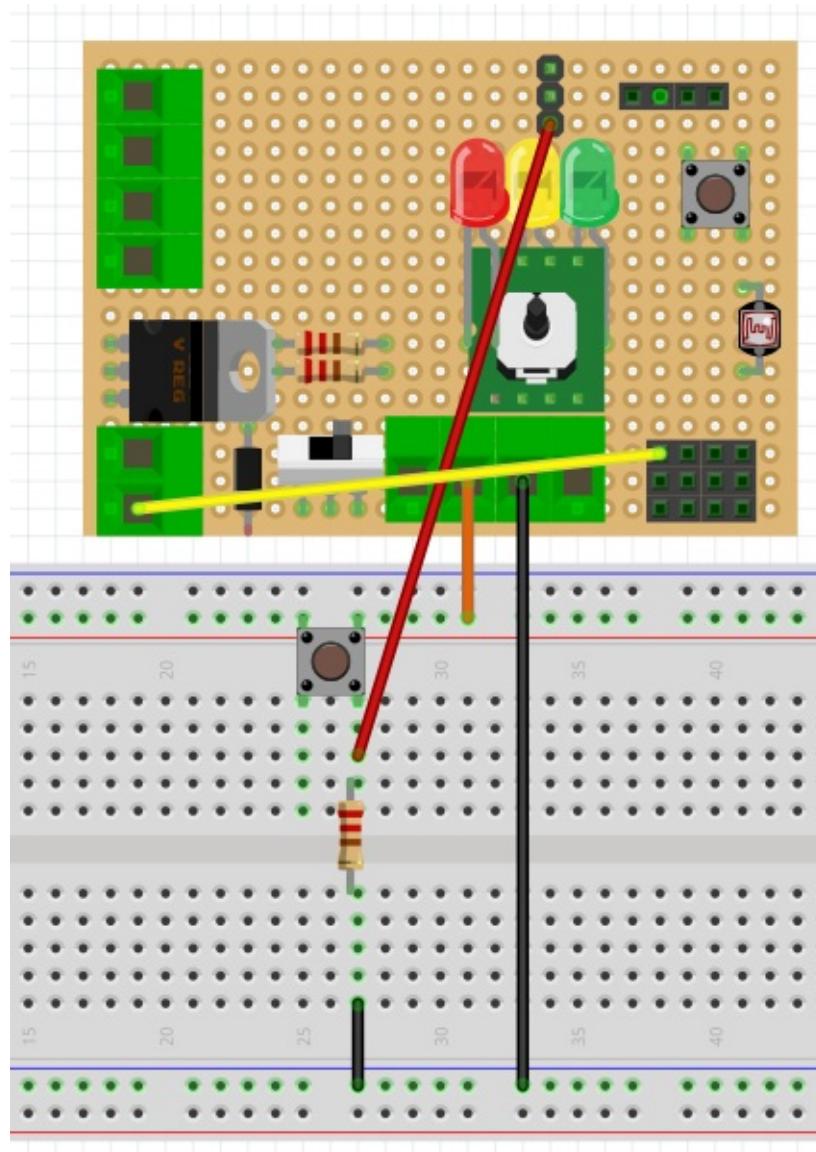
1. El ladrón puede entrar sólo por la ventana o por la puerta. Fíjate en las puertas y ventanas de los comercios de tu localidad. Seguro que has visto más de uno.
2. Como la ventana de la casa da a una calle principal muy transitada el ladrón no intentará entrar nunca por la ventana cuando sea de día.
3. La entrada de nuestra casa es oscura y no tiene ventanas al exterior, por lo tanto nuestro comportamiento habitual es abrir la puerta, encender la luz y cerrar la puerta.
4. Sólo abrimos las ventanas de día, nunca por la noche.

Ten en cuenta que los interruptores podrían sustituirse en un caso real con relé un Reed conocido también como interruptor magnético. Son elementos económicos y de muy fácil instalación.

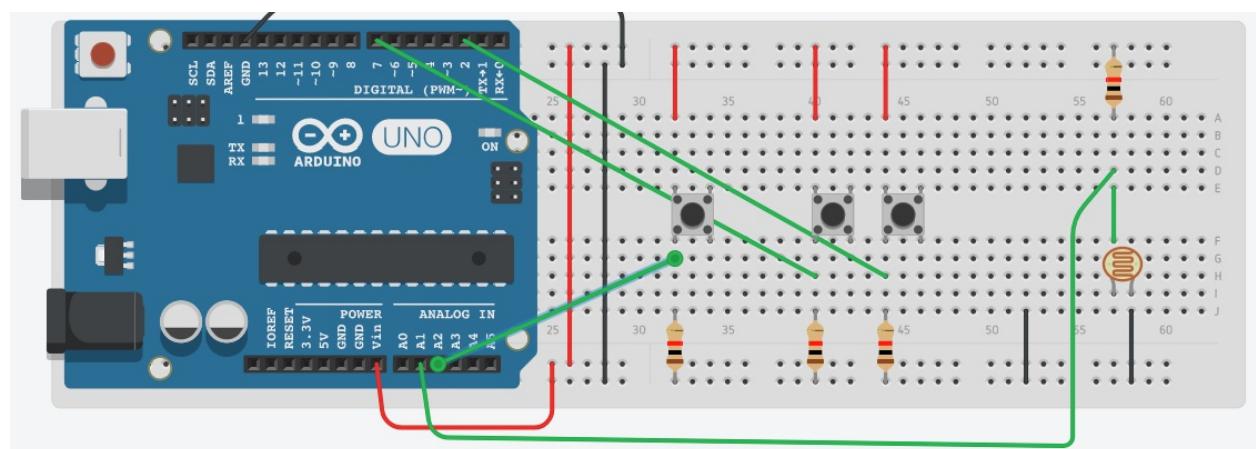
Con EDUBASICA:

- Como detector de **apertura de puerta** vamos a usar el pulsador de la PLACA EDUBASICA (**D2**)
- Como detector de la **ventana** vamos a usar un pulsador que montaremos sobre la protoboard que lo conectaremos a la clema **D7**.
- Sabremos si es de **día** o de **noche** gracias al **LDR** de la EduBásica.
- Utilizaremos el interruptor V1 como **RESET** por lo tanto lo conectaremos a A2 para leerlo

El esquema sería el siguiente



SIN EDUBASICA



Este sería el programa:

```
1 int ventana, puerta, luz, reset; //definimos variables
```

```

2   bool alarma;
3   void setup() {
4     pinMode(7, INPUT);
5     pinMode(5, OUTPUT); //led rojo
6     pinMode(2, INPUT);
7     Serial.begin(9600);
8     reset=0;
9     alarma =false;
10 }
11
12 void loop() {
13   ventana=digitalRead(7); //guardamos el estado de la ventana
14   Serial.print(" VENTANA ");
15   Serial.print(ventana);
16   puerta=digitalRead(2); //guardamos estado de puerta
17   Serial.print(" PUERTA ");
18   Serial.print(puerta);
19   luz=analogRead(1); //guardamos estado de LUZ
20   Serial.print(" LUZ ");
21   Serial.print(luz);
22   reset= analogRead(2);
23   Serial.print(" RESET=");
24   Serial.println(reset);
25
26 }
```

Abre el “monitor serial” y prueba a activar los pulsadores. Verás que cuando están activos el valor obtenido es 1 y cuando están desactivados su valor es 0. Comprueba qué interruptor se corresponde con la ventana y cual con la puerta. Tapa ahora el LDR y observa el valor que obtienes cuando no recibe luz (será el mismo valor que si es de noche). A nosotros nos ha salido:

```

1 VENTANA 0 PUERTA 0 LUZ 920 RESET=356      <---- estado normal
2 VENTANA 0 PUERTA 1 LUZ 917 RESET=356      <---- apretando el pulsador de la
3   placa Edubásica
4 VENTANA 0 PUERTA 0 LUZ 1011 RESET=356      <---- tapando el LDR
5 VENTANA 0 PUERTA 0 LUZ 1016 RESET=875      <---- activando el interruptor de
   Edubasica
6 VENTANA 0 PUERTA 0 LUZ 884 RESET=356      <---- quitando el interruptor (e
   stato normal)
```

Atendiendo a los supuestos anteriores carga este programa y observa su funcionamiento. Si el led rojo se enciende es que la alarma se ha disparado.

```

1 int ventana, puerta, luz, reset;//definimos variables
2 bool alarma;
3 void setup() {
4   pinMode(7, INPUT);
5   pinMode(5, OUTPUT); //led rojo
6   pinMode(2, INPUT);
```

```
7 Serial.begin(9600);
8 reset=0;
9 alarma =false;
10
11 }
12 void loop() {
13 ventana=digitalRead(7); //guardamos el estado de la ventana
14 Serial.print(" VENTANA ");
15 Serial.print(ventana);
16 puerta=digitalRead(2); //guardamos estado de puerta
17 Serial.print(" PUERTA ");
18 Serial.print(puerta);
19 luz=analogRead(1); //guardamos estado de LUZ
20 Serial.print(" LUZ ");
21 Serial.print(luz);
22 //implementamos la logica de la puerta
23 if (puerta==1) {//la puerta esta abierta
24     delay(3000); //esperamos hasta que encienda la luz
25     if (luz > 1000) alarma=true; //no han encendido la luz
26 }
27 //implementamos logica de ventana
28 if (ventana==1 && luz < 1000) alarma=true;
29 if (alarma){
30     digitalWrite(5,HIGH);
31     Serial.print(" ##### ALARMA ##### !!!!! ");
32 }else{
33     digitalWrite(5,LOW);
34 }
35 reset= analogRead(2);
36 if (reset>800) {
37     alarma=false;
38     Serial.print(" ##### RESET ##### !!!!! ");
39 }
40 Serial.print(" RESET=");
41 Serial.println(reset);
42 }
```

El resultado es :

```

VENTANA 0 PUERTA 0 LUZ 921 RESET=355
VENTANA 0 PUERTA 0 LUZ 920 RESET=355
VENTANA 0 PUERTA 0 LUZ 919 RESET=355
VENTANA 0 PUERTA 0 LUZ 919 RESET=356
VENTANA 0 PUERTA 0 LUZ 919 RESET=355
VENTANA 0 PUERTA 0 LUZ 921 RESET=355
VENTANA 0 PUERTA 0 LUZ 923 RESET=355
VENTANA 0 PUERTA 0 LUZ 926 RESET=356
VENTANA 0 PUERTA 0 LUZ 931 RESET=355
VENTANA 0 PUERTA 0 LUZ 932 RESET=356
VENTANA 0 PUERTA 0 LUZ 932 RESET=355
VENTANA 0 PUERTA 0 LUZ 931 RESET=356
VENTANA 0 PUERTA 0 LUZ 930 RESET=355
VENTANA 0 PUERTA 0 LUZ 928 RESET=355
VENTANA 0 PUERTA 0 LUZ 926 RESET=356
VENTANA 0 PUERTA 0 LUZ 923 RESET=355
VENTANA 0 PUERTA 0 LUZ 922 RESET=355
VENTANA 0 PUERTA 0 LUZ 920 RESET=355
VENTANA 0 PUERTA 0 LUZ 915 RESET=356

```

[Video link](#)

Actividad

Ampliación: Pon el transistor y el altavoz ¡¡que suene!!

Actividad

Como comprobarás una vez que la alarma se activa permanece en ese estado. Para desactivarla debes de activar el interruptor.

Piensa en otra solución para poder desactivar la alarma, por ejemplo abriendo la ventana y la puerta a la vez.

Actividad

Ampliación: Usamos EduBásica porque dispone de elementos hardware ya instalados, como led y pulsador, pero piensa que sin EduBásica tu Arduino dispone de 13 entradas digitales y 6 analógicas. Piensa en un sistema más completo de alarma en el que puedas conectar sensores de humo o de movimiento (PIR).

Actividad

Proyecto propuesto: Realiza todo el proceso para implementar, mediante funciones lógicas, el siguiente sistema:

Se trata de una máquina de control de una cinta transportadora. Tenemos un sensor de temperatura basado en un termistor que nos dirá si se ha producido un sobrecalentamiento en la máquina. También hay un sensor de presión que detecta la presencia de un objeto sobre la cinta transportadora. Por último, la cinta transportadora sólo estará en funcionamiento si el operario mantiene apretado un pulsador. Tendremos un led que avisará si hay sobrecalentamiento y detendrá la cinta si está en movimiento. Un zumbador avisará cuando la cinta esté en movimiento.