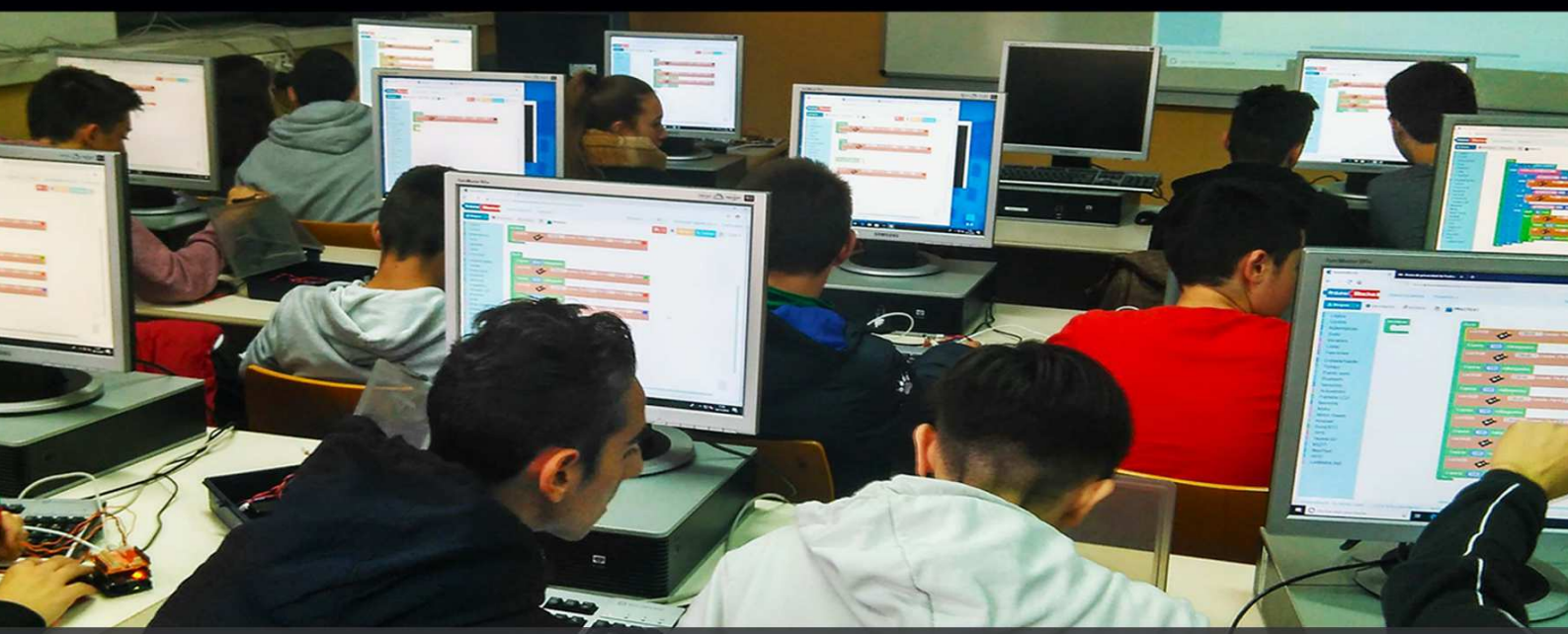
Arduino Blocks

robótica · educativa

Free Book



JUAN JOSÉ LÓPEZ ALMENDROS

arduinoblocks.com

Free Book

ArduinoBlocks.com

junio 2020

Juan José López Almendros

Colabora:

1 Introducción

La plataforma ArduinoBlocks permite iniciarnos en el mundo de la electrónica, robótica y el mundo "*maker*" de una forma sencilla e intuitiva.

La motivación de desarrollar la plataforma ArduinoBlocks y escribir este libro nace de mi trabajo como docente con alumnos de entre 12 y 18 años, sin previos conocimientos de programación, que quieren adentrarse en el mundo Arduino partiendo de una nociones básicas de electricidad y electrónica.

ArduinoBlocks es la herramienta perfecta para niños, jóvenes y no tan jóvenes que quieren iniciarse en el mundo Arduino de una forma divertida e intuitiva.

Aprender a programar con bloques, aunque parece un juego, es una forma totalmente válida de obtener los conceptos básicos de programación que posteriormente se podrán aplicar a cualquier otro lenguaje de programación, ya sea visual o escrito. Programando con bloques nos olvidamos principalmente de los fallos de sintaxis pero tenemos que aprender de igual forma conceptos básicos como secuenciación, bucles, variables, condiciones, funciones, etc. Desarrollamos la capacidad de abstracción de un problema para convertirlo en una secuencia de comandos, o bloques, que resolverán el problema, es decir, aprenderemos a pensar como un programador.

Por experiencia, creo que el aprendizaje con un sistema visual suele ser menos estresante y permite obtener resultados mucho más rápido, obteniendo una experiencia mucho menos fustrante para el programador novel. Y en muchos casos, se utiliza esta iniciación como trampolín para dar el salto a una programación avanzada mediante código escrito obteniendo muchas más éxito.

1.1 Plataforma Arduino

Arduino es un proyecto de hardware libre que ideó una plataforma completa de hardware y software compuesta por placas de desarrollo que integran un microcontrolador y un entorno de desarrollo IDE. La idea surgió para facilitar el uso de la electrónica en proyectos multidisciplinarios.

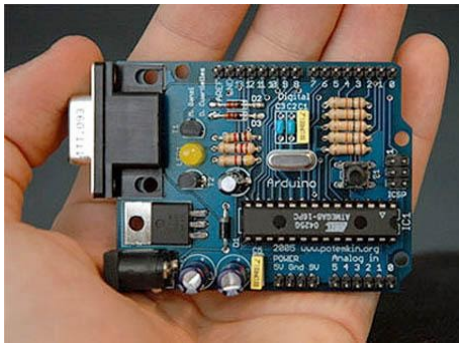
El hardware consiste en una placa de circuito impreso con un microcontrolador, normalmente Atmel AVR, y puertos digitales/analógicos de entrada/salida donde conectar sensores y/o actuadores.

La primera placa Arduino fue introducida en 2005, ofreciendo un bajo costo y facilidad para uso de novatos y profesionales.

Puedes consultar más información sobre la historia y el entorno Arduino en wikipedia:

https://es.wikipedia.org/wiki/Arduino

Arduino 2006



Arduino 2016

Existen múltiples placas Arduino con diferentes características y distintos microcontroladores.

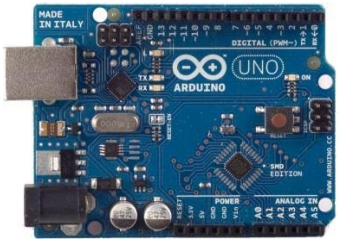
El más utilizado y estándar es el Arduino UNO, sin embargo en algunos casos podemos necesitar otra placa Arduino para adaptarnos al tipo de proyecto a realizar.

https://www.arduino.cc/en/Main/Boards

Algunas de las placas Arduino más utilizadas:

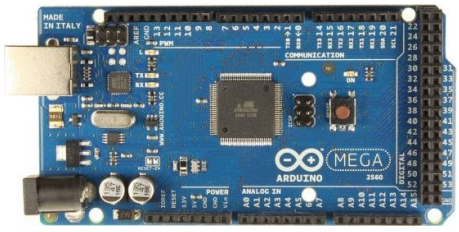
Arduino UNO

Es el modelo más estándar y el más utilizado.



Arduino MEGA

Mayor potencia, más recursos hardware y más memoria

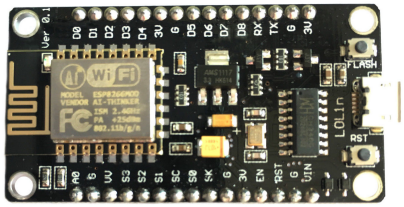


Arduino Nano

Similar potencia que el Arduino UNO pero de menor tamaño

NodeMCU

Placa basada en el microcontrolador ESP8266 programable en entorno Arduino



Dentro de nuestros proyectos, la placa Arduino será el "*cerebro*" que gestiona toda la información recogida desde los sensores, toma decisiones y actúa sobre los elementos de control conectados.

Según las necesidades del proyecto deberemos elegir la placa Arduino más apropiada. ArduinoBlocks soporta multitud de placas y robots basados en Arduino: Arduino UNO, Nano y MEGA, Leonardo, ProMicro, EasyPlug, Otto DIY, KeyBot, 3dBot,...

Por otro lado el "*framework Arduino*" se ha adaptado a otros microcontroladores de bajo costo y muy potentes como esl el caso del *"ESP8266"*, en concreto podremos programas las placas de desarrollo basadas en este chip: NodeMCU y WeMos D1 / mini..

Una de las características de Arduino, es su *"bootloader"* que permite grabar el programa dentro del microcontrolador directamente desde la conexión USB del ordenador.

El tamaño de memoria Flash para el programa de un Arduino UNO es de 32KB de los cuales debemos restar el tamaño del "*bootloader*" pregrabado (0.5KB en Arduino UNO y 2KB en Arduino NANO)

1.2 Plataforma ArduinoBlocks

ArduinoBlocks es una plataforma donde podemos programar nuestra placa Arduino de forma visual sin necesidad de conocer el lenguaje C++/Processing que utiliza Arduino IDE.

ArduinoBlocks se basa en el framework Blockly de Google y la programación se realiza con bloques al estilo AppInventor o Scratch. No tenemos que escribir líneas de código y no nos permitirá unir bloques incompatibles evitando así posibles errores de sintaxis.

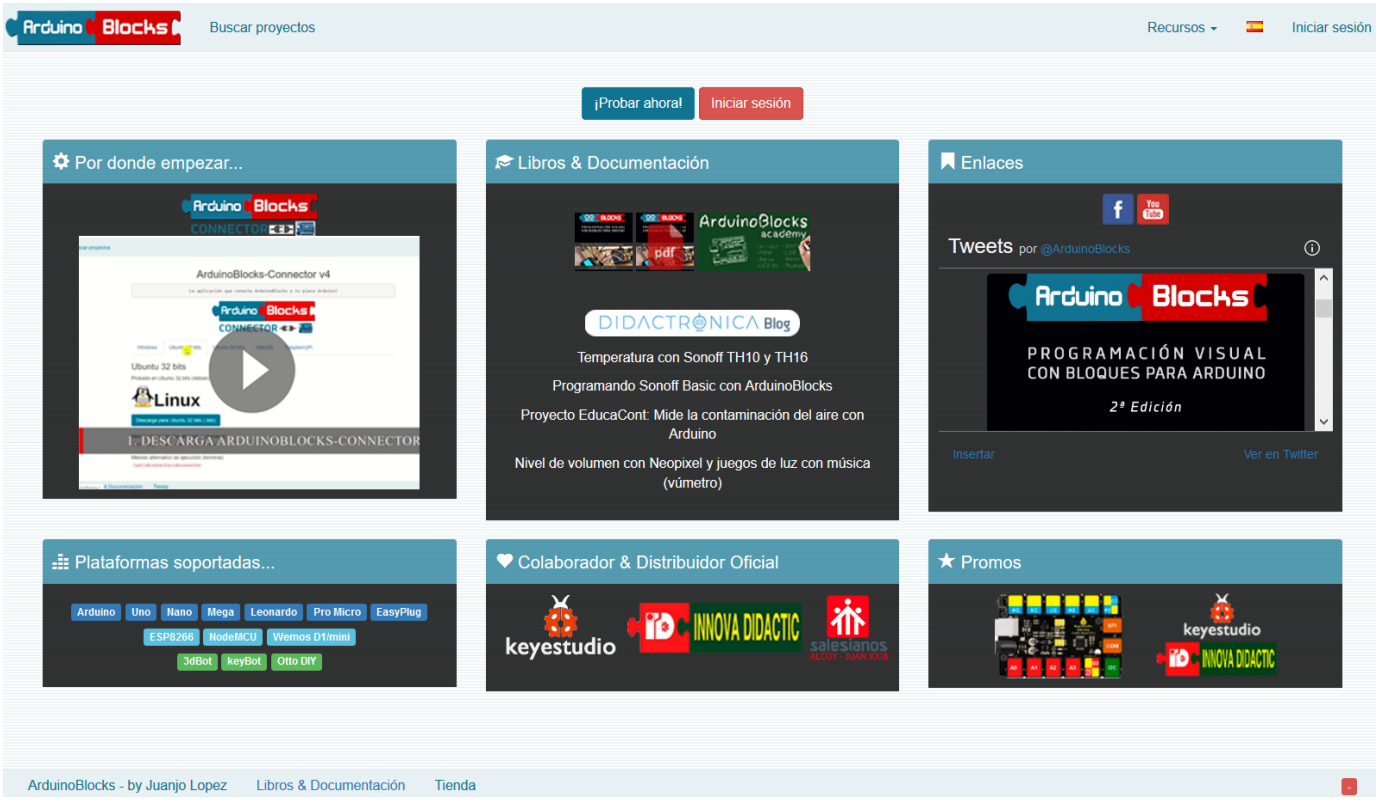
La plataforma ArduinoBlocks genera, compila y sube el programa a la placa Arduino por medio de la conexión USB. Una vez subido el programa, la placa Arduino no necesitará de la conexión al PC para funcionar pudiendo alimentarla con baterías o una fuente de alimentación para que funcione de forma autónoma.

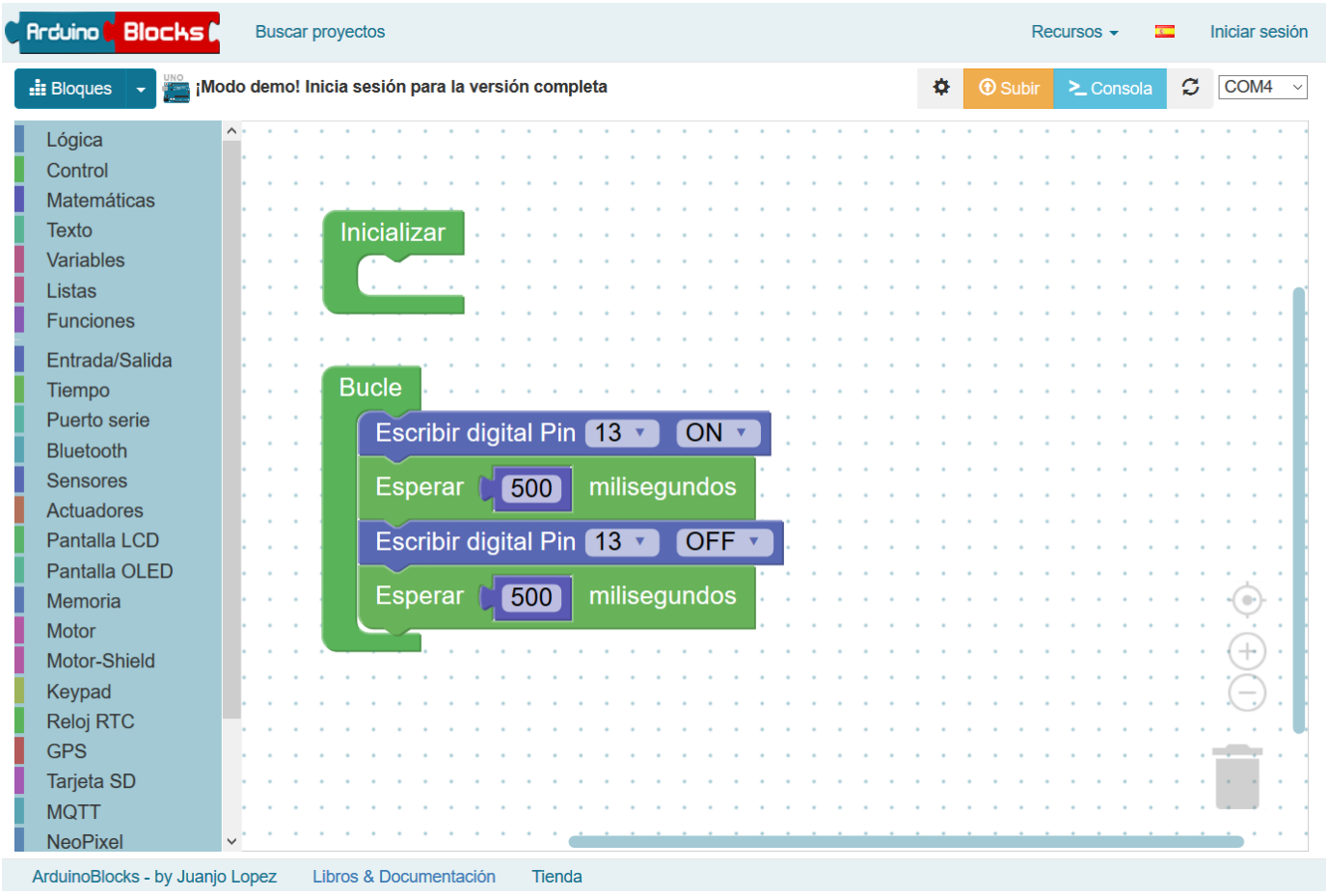
ArduinoBlocks actualmente funciona con todos los navegadores de última generación: Mozilla Firefox, Google Chrome, Opera, Safari, etc. (recomandado Firefox y Chrome)

Registrándonos como usuarios de la plataforma ArduinoBlocks podemos aprovechar todas estas posibilidades:

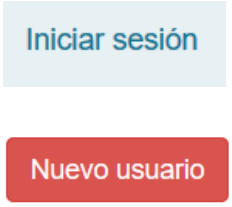
- Guardar tus proyectos en la nube de ArduinoBlocks.
- Añadir información al proyecto: descripción, componentes utilizados, imágenes, etc.
- Añadir archivos adjuntos relacionados con el proyecto: esquemas, fotos, archivos para impresión 3D, aplicaciones, etc.
- Compartir proyectos con el resto del mundo.
- Importar proyectos compartidos por otros usuarios.
- Importar y exportar proyectos en archivo para publicar en web o compartir.
- Exportar el código .ino para utilizarlo en Arduino IDE
- Exportar el proyecto completo como .zip incluyendo librerías para Arduino IDE
- Programar directamente Arduino desde el propio navegador (Con la aplicación: ArduinoBlocks-Connector, ver apdo. 1.3)
- Utilizar la consola serie desde el propio navegador

www.arduinoblocks.com

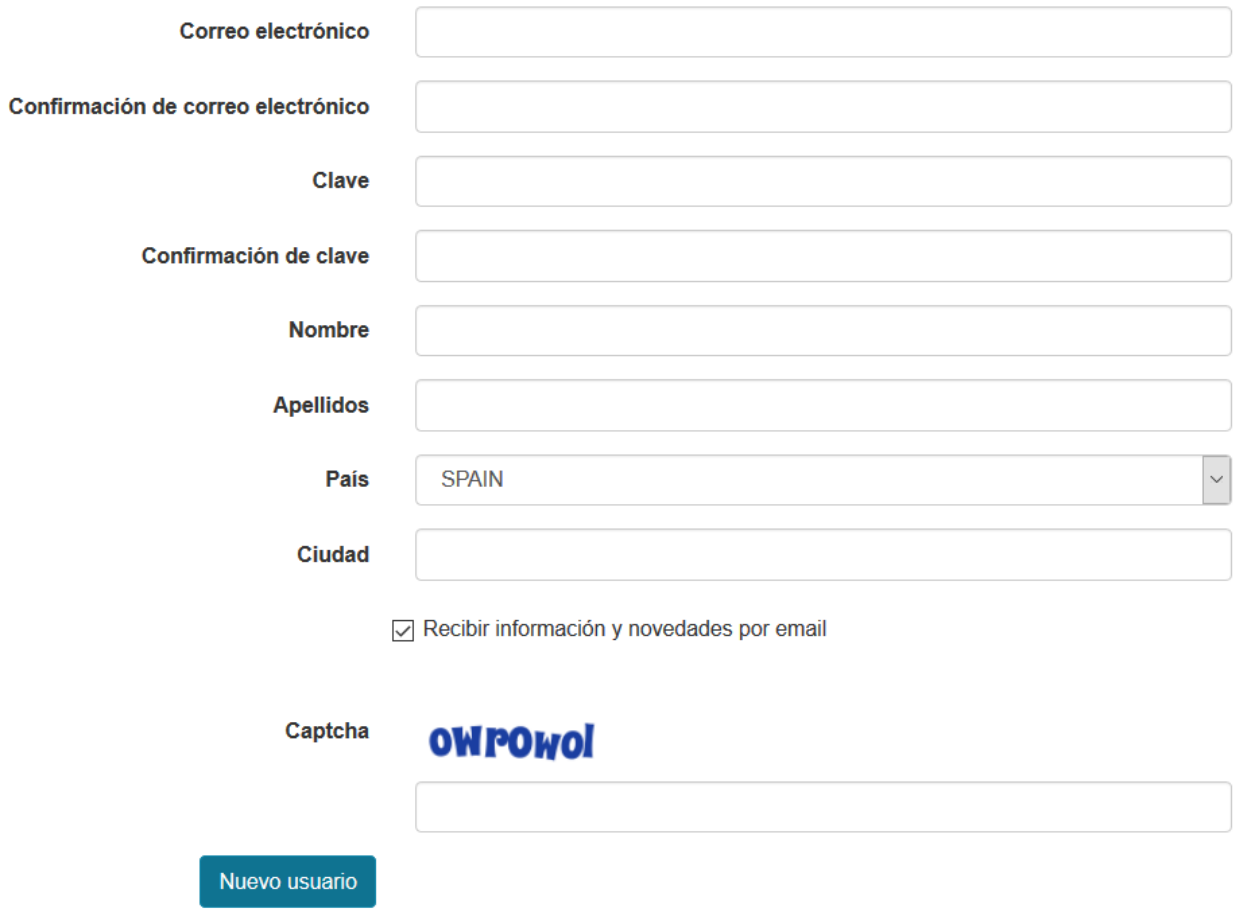




- **Registro de nuevo usuario:** Para empezar a usar ArduinoBlocks y tener acceso a todas sus funcionalidades debemos de registrarnos para tener nuestra cuenta de usuario. Para ellos debemos pinchar en "*iniciar sesión*" y a continuación en la opción "*nuevo usuario*":



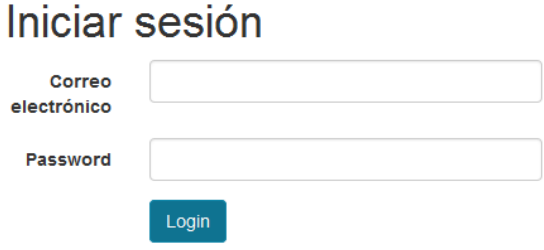
En el formulario de registro debemos completar nuestros datos, y usar un correo electrónico válido donde recibirremos un correo de confirmación para activar la



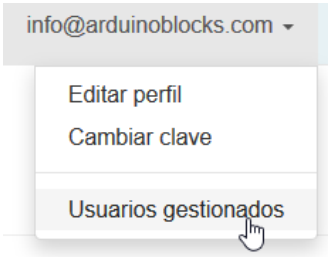
A continuación revisa tu bandeja de entrada del correo electrónico y recibirás un correo con un enlace para confirmar y activar tu cuenta de usuario. A partir de ese momento puede iniciar sesión y crear tus proyectos dentro de tu cuenta de ArduinoBlocks.

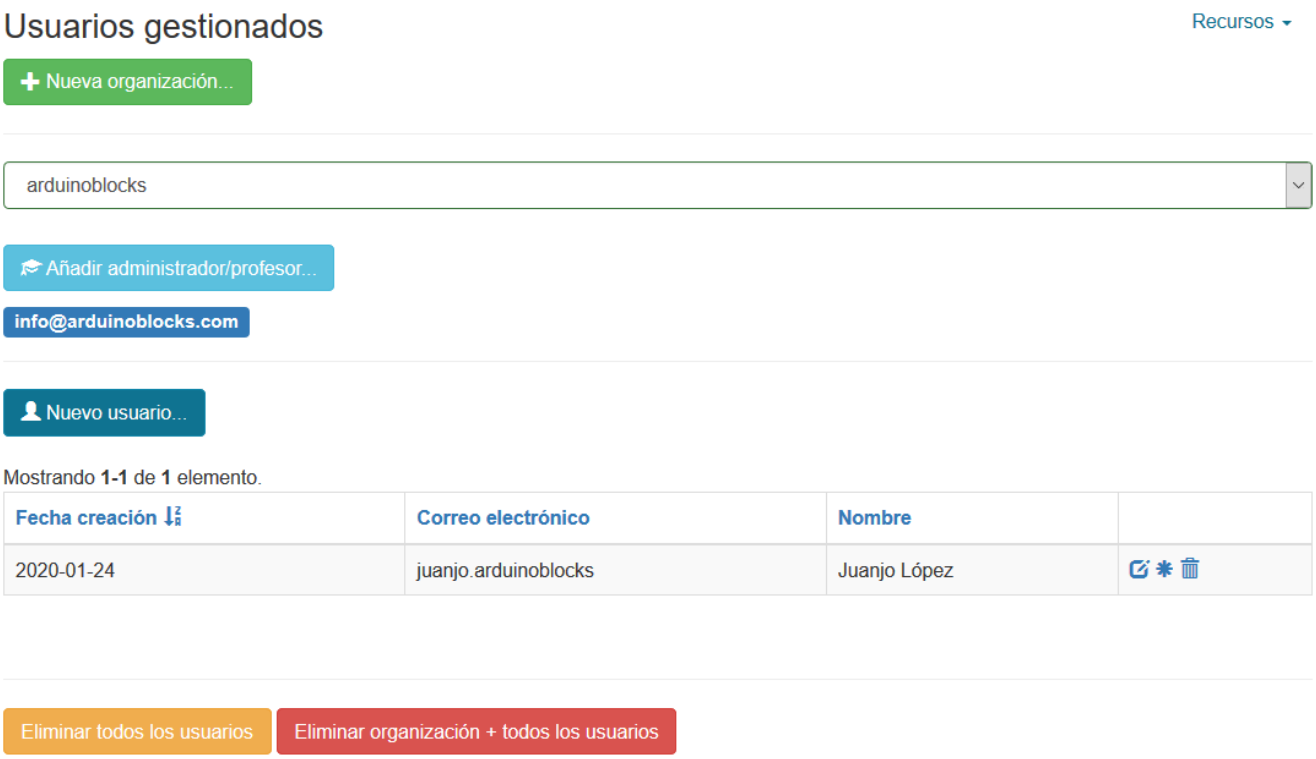
Si eres profesor y quieres crear cuentas para tus alumnos o centro educativo sin utilizar el correo electrónico de los alumnos, puedes crear “cuentas gestionadas”.

- **Inicio de sesión:** Debemos iniciar sesión o crear previamente una nueva cuenta en caso de acceder por primera vez. Esto nos permitirá acceder a nuestros proyectos en la nube y a todas las ventajas de la comunidad ArduinoBlocks.



- **Usuarios gestionados:** Si eres profesor y quieres crear cuentas para tus alumnos o centro educativo puedes utilizar la opción de usuarios gestionados. De esta forma no se necesita correo electrónico de los alumnos y los usuarios están bajo el control de la cuenta de usuario del profesor (o profesores). Los usuarios gestionados tienen algunas limitaciones como no permitir archivos adjuntos en los proyectos o no poder crear proyectos como profesor.





Lo primero que debemos hacer es crar una organización. Puede ser el nombre de nuestro centro, academia, etc.

Podemos añadir a otros profesores a partir de su cuenta (con el correo electrónico) para que sean colaboradores dentro de la organización y también puedan gestionar alumnos.

Cada usuario gestionado tendrá un nombre y para iniciar sesión el nombre de usuario será:

nombre.organización

Los usuarios administradores de la organización podrán:

-Crear nuevos usuarios, editar y eliminar usuarios gestionados

-Cambiar la clave de los usuarios (no pueden ver la actual, pero sí cambiarla)

-Crear proyectos como profesor para estos alumnos

-Eliminar la organización y todos los usuarios

Los usuarios gestionados dentro de la organización podrán:

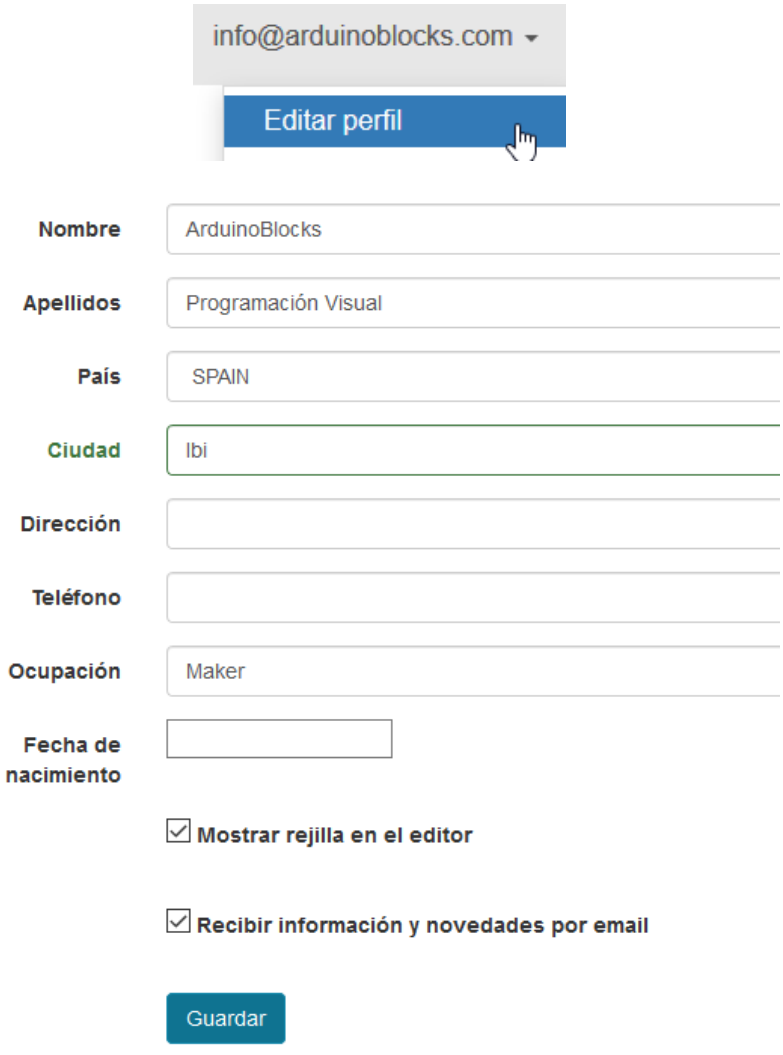
-Crear proyectos personales (sin archivos adjuntos)

-Unirse a proyectos de profesor, siempre que el profesor sea administrador de la organización.

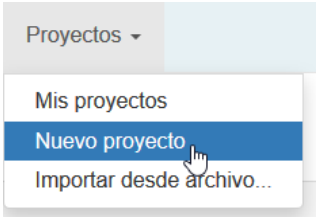
-Cambiar su propia clave

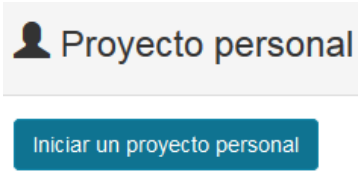
-Exportar un proyecto a archivo (si tienen una cuenta de usuario normal, de esta forma pueden exportar e importar proyectos entre cuentas)

- **Perfil de usuario:** Dentro del perfil del usuario podemos ajustar datos personales y configuración específica del usuario:

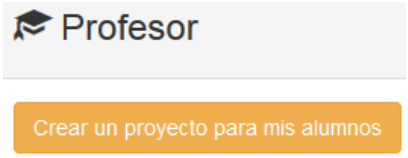


- **Creación de un nuevo proyecto :** Para iniciar un nuevo proyecto debemos hacer clic en el menú:

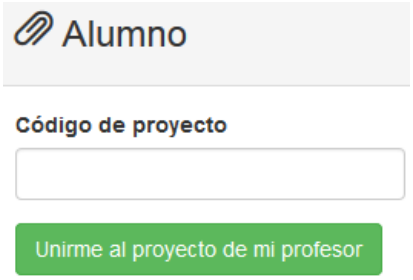




Iniciar un nuevo proyecto que sólo será accesible para el usuario. Posteriormente se puede compartir al resto de la comunidad si se desea.



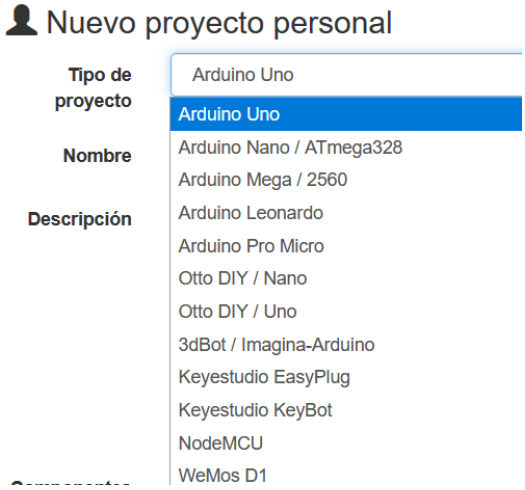
Iniciar un proyecto como profesor. De esta forma no se inicia un proyecto como tal, sino que se especifican los datos del proyecto y se genera un código para que los alumnos se puedan suscribir al proyecto. El profesor podrá supervisar y valorar los proyectos de sus alumnos.



De esta forma nos unimos a un proyecto planteado por el profesor. Nosotros realizaremos el proyecto como si de un proyecto personal se tratara, pero el profesor podrá supervisar y valorar nuestro trabajo.

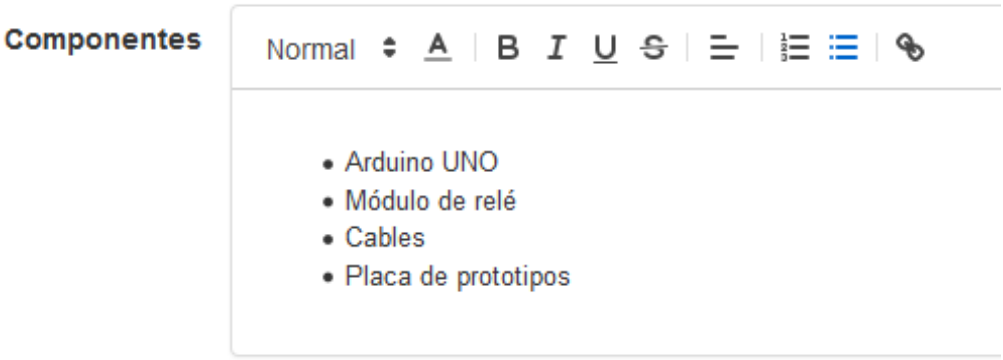
Podemos obtener un código o un enlace para que los alumnos se puedan suscribir al proyecto fácilmente.

En caso de un proyecto personal o como profesor debemos seleccionar en el siguiente paso la placa Arduino a utilizar:

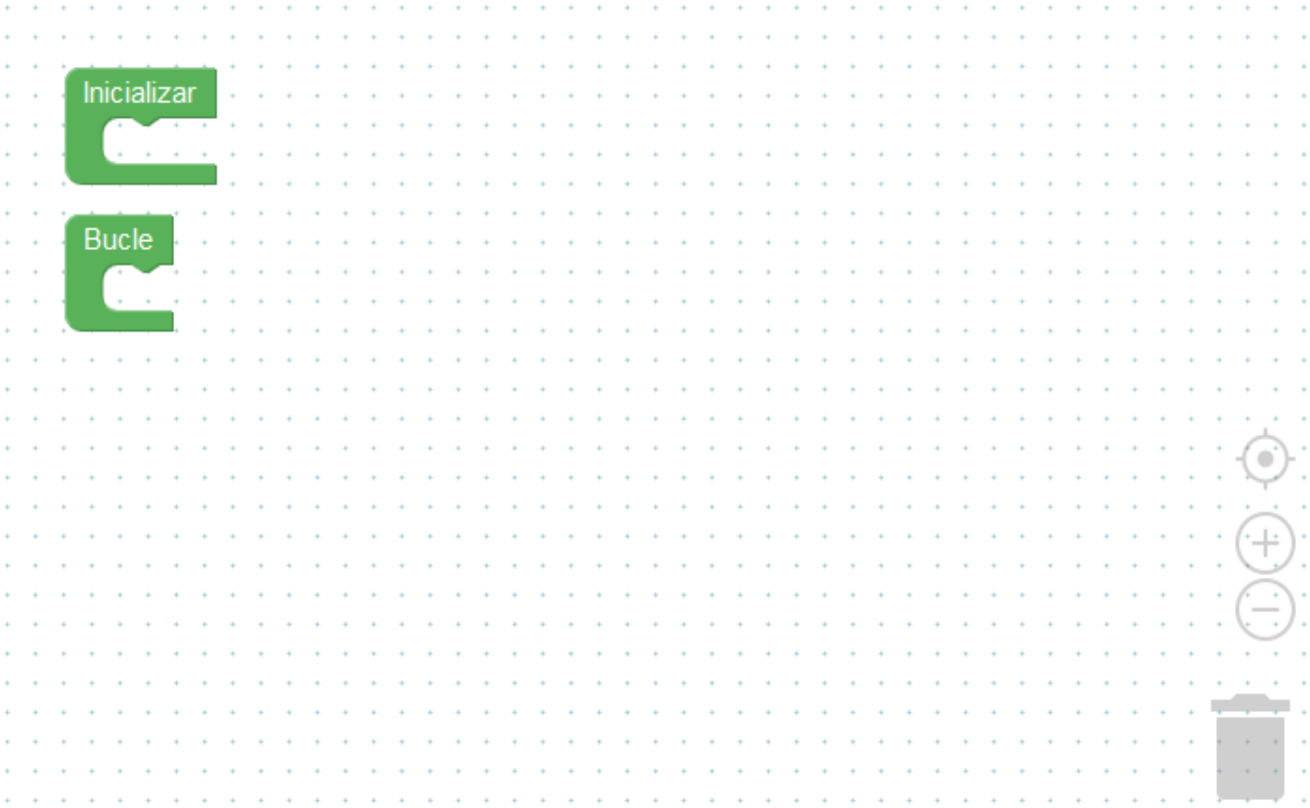


Se debe indicar un nombre descriptivo corto y una descripción más detallada.

En la sección componentes podemos indicar los componentes utilizados en el proyecto:



- **Área de programación del proyecto:**

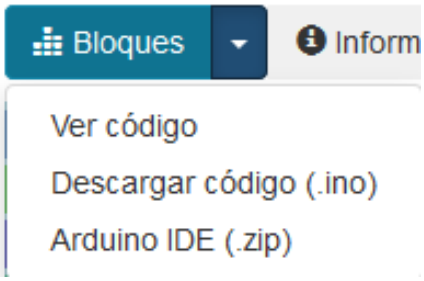


Las principales secciones del área de programación son:

Para añadir bloques al programa arrastramos desde la barra de herramientas al área de programa, insertando dentro del bloque de inicialización o de bucle.

Los bloques que estén fuera del bloque de inicialización o del bloque del bucle del programa principal serán ignorados a la hora de generar el programa (excepto los bloques de funciones o bloques especiales como interrupciones y otros eventos).

ArduinoBlocks genera el código de Arduino a partir de los bloques. El programa se puede compilar y subir directamente a la placa Arduino gracias a la aplicación ArduinoBlocks-Connector (disponible para descargar desde la web), sin embargo si deseamos ver o descargar el código podemos realizarlo desde el área de bloques.



Código generado por ArduinoBlocks

Código Arduino

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

  pinMode(13, OUTPUT);
    Serial.println(String("ArduinoBlocks!"));

}


void loop()
{

    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);

}
```

Código en Arduino IDE

Si descargamos o copiamos el código generado por ArduinoBlocks en Arduino IDE podemos necesitar algunas librerías no incluidas con Arduino IDE. Para ello debemos descargarla y añadirlas a la plataforma Arduino IDE para una correcta compilación del programa, o utilizar la opción descargar .zip para Arduino IDE lo que nos facilitará un archivo comprimido con el código de nuestro programa y todas las librerías necesarias incluidas en la misma carpeta.

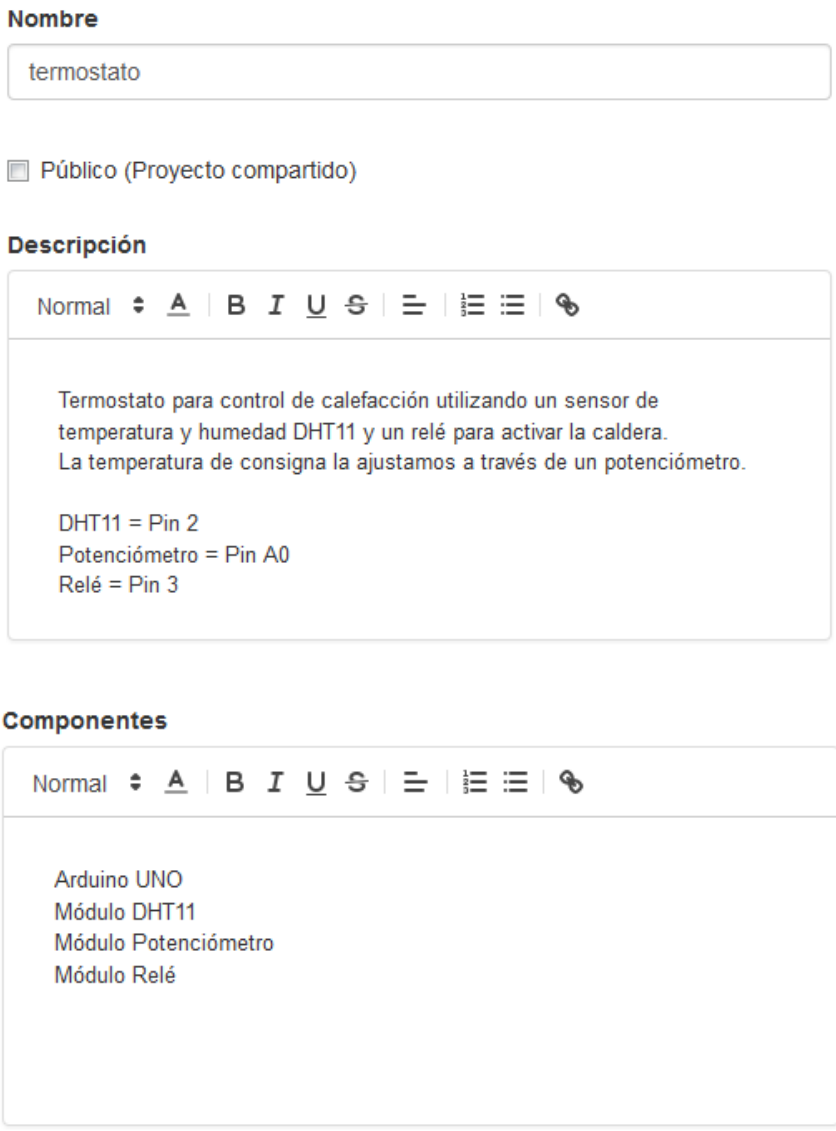
http://www.arduinoblocks.com/web/help/libraries

Nombre	Descripción	Download
ablibs.zip	ArduinoBlocks Library Pack	

La opción más rápida y sencilla es la compilación y programación directa desde el propio navegador junto a la aplicación ArduinoBlocks-Connector:

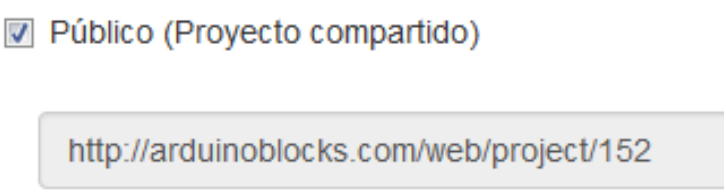
http://www.arduinobocks.com/web/site/abconnector

- **Área de información del proyecto:** Un proyecto electrónico debe estar siempre correctamente documentado. En la sección información podemos añadir información o modificar la indicada durante la creación del proyecto.



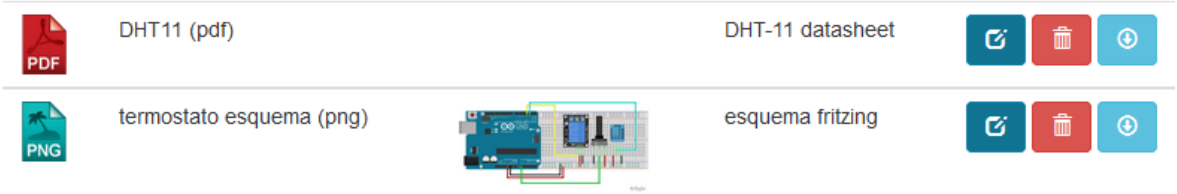
Señalando la opción “público” podemos hacer que nuestro proyecto esté disponible públicamente para que otros usuarios busquen nuestro proyecto (sin poder editarlo) y pueden importar una copia a su propia cuenta.

Ejemplo: enlace público para compartir nuestro proyecto:



Al indicar nuestro proyecto como “público” aparecerá en la lista búsqueda de proyectos compartidos para todos los usuarios de ArduinoBlocks.

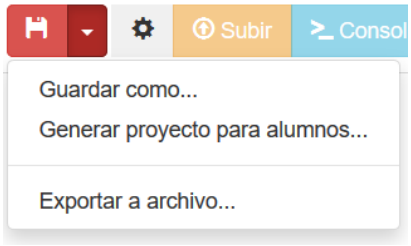
- **Área de archivos adjuntos del proyecto:** De igual forma podemos adjuntar imágenes, hojas de datos o cualquier otro archivo relacionado con el proyecto.



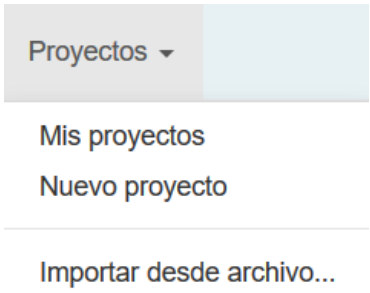
- **Guardar:** ArduinoBlocks guarda automáticamente el proyecto cada cierto tiempo. En caso de querer asegurarnos el guardado podemos pulsar el botón “*Guardar*”. También podemos crear un nuevo proyecto a partir del actual pulsando la opción “*Guardar como...*”. Automáticamente se abrirá el nuevo proyecto creado a partir del primero

- **Guardar proyecto para alumnos:** Con esta opción generamos un proyecto de profesor a partir del actual, es decir un tipo de proyecto que el profesor plantea a los alumnos y puede supervisar y evaluar y al cual los alumnos se suscriben con un código. De esta forma podemos plantear proyectos a a los alumnos con bloques para que los alumnos los corrijan o los complenten, pues si iniciamos un nuevo proyecto como profesor siempre estará vacío.

- **Exportar a archivo:** La opción “Exportar a archivo” permite generar un archivo con toda la información del proyecto: bloques, información, archivos adjuntos, etc. De esta forma podemos enviar un proyecto por email, subirlo a un sitio web para su descarga, o como copia de seguridad del proyecto.

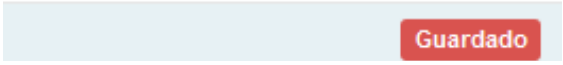


- **Importar desde archivo:** Permite importar un nuevo proyecto desde un archivo exportado anteriormente desde la plataforma.

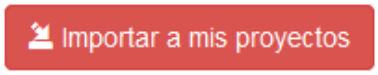


- **Captura de pantalla:** Mediante el botón con el icono de la cámara de fotos, podemos obtener una captura del área de bloques del proyecto en formato de imagen PNG. Esta función permite obtener la imagen de los bloques para documentar proyectos, prácticas, tutoriales, presentaciones, etc.

- **Barra de información:** En la parte inferior derecha podemos obtener la información de guardado y algunos avisos que nos muestra la aplicación.



- **Importar desde un proyecto compartido:** Si accedemos a visualizar un proyecto compartido por otro usuario aparecerá un botón “importar a mis proyectos”, de esta forma podemos crear una copia del proyecto en mis proyectos personales para poder modificarlo a nuestro gusto.



- **“Me gusta”:** De igual forma si accedemos a ver proyectos compartidos por otros usuarios aparecerá un botón “me gusta” para valorar positivamente el trabajo realizado por el usuario.

- **Estructura de un nuevo proyecto:** Un proyecto Arduino tiene siempre dos estructuras importantes en su interior, esto se ve reflejado claramente al crear un nuevo proyecto en ArduinoBlocks.

1. **Bloque “inicializar” o “setup”:**

El contenido de este bloque sólo se una vez durante el inicio del microcontrolador de Arduino (o si pulsamos el reset y la placa Arduino se reinicia).
Este bloque se utiliza para inicializar variables, configurar sensores, actuadores o periféricos, etc.

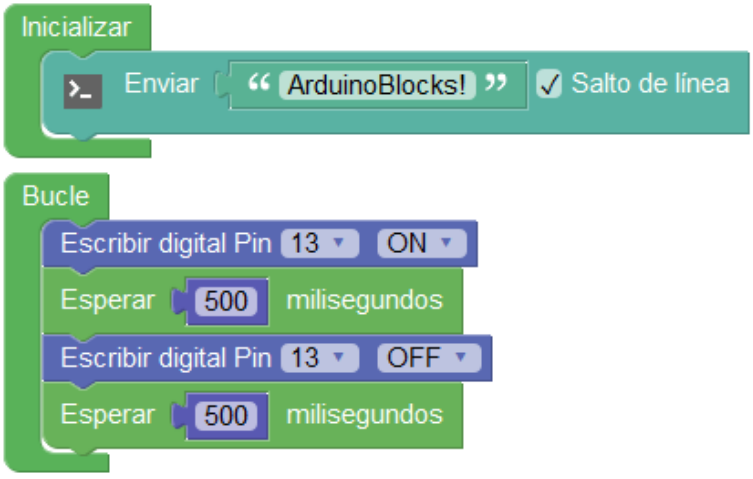
2. Bloque “bucle” o “loop”:

El contenido de este bloque se repite indefinidamente. Dentro de este bloque añadiremos los bloques de nuestro programa con la funcionalidad deseada.

Cualquier bloque que no esté dentro del bloque de inicialización o de bucle y no forme parte de una función (ver apartado 3.2.6) será ignorado a la hora de generar el código.

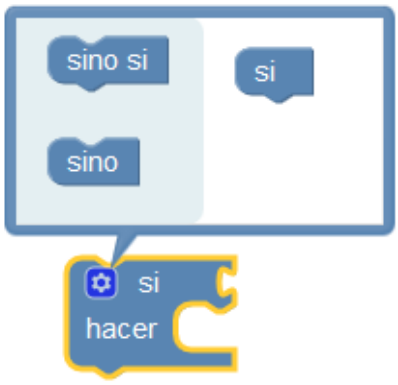
Ejemplo: Al iniciar se establece la variable a 0. Se envía y se incrementa cada segundo indefinidamente:

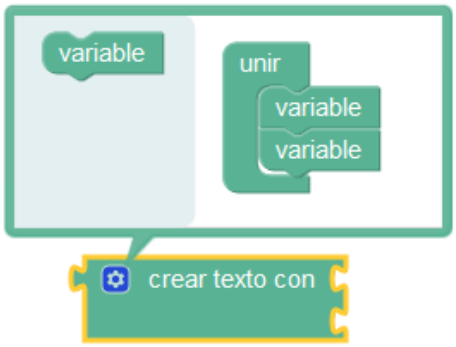
Ejemplo: Al iniciar (o reset) se envía un mensaje por el puerto serie. El led conectado al pin 13 se ilumina, espera 500ms, se apaga y espera otros 500ms (este ciclo ser repetirá indefinidamente).



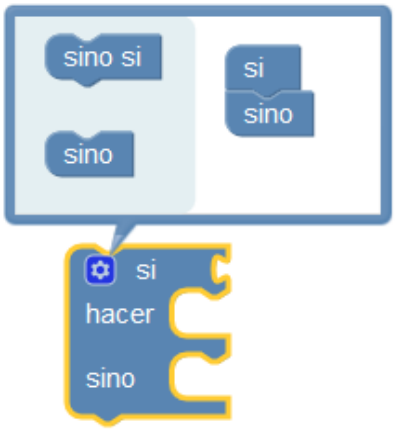
Importante: El “*bootloader*” de Arduino normalmente tiene configurada la opción de resetear el microcontrolador cuando se inicia una conexión serie, por tanto si conectamos con la consola serie del PC hay que tener en cuenta que se reiniciará el programa y se ejecutará el bloque “inicializar

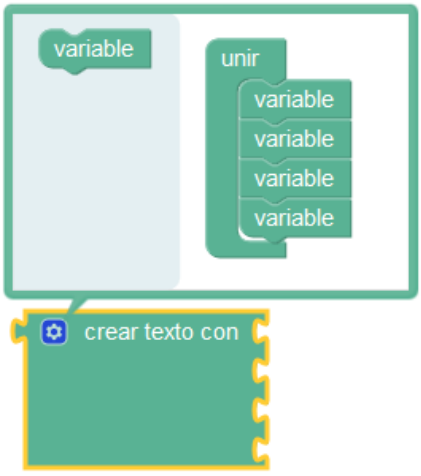
- **Configuración de bloques:** Algunos bloques permiten configurar o alterar su funcionamiento. Para desplegar las opciones posibles pinchamos sobre el icono superior izquierda del bloque con apariencia de rueda dentada:



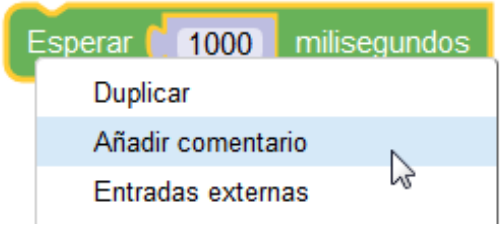


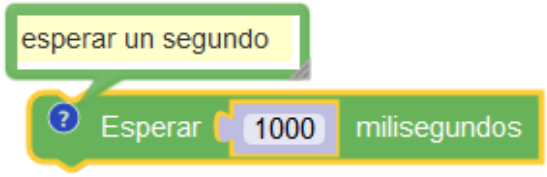
La configuración se realiza arrastrando los modificadores del bloque de la parte izquierda a la parte derecha:

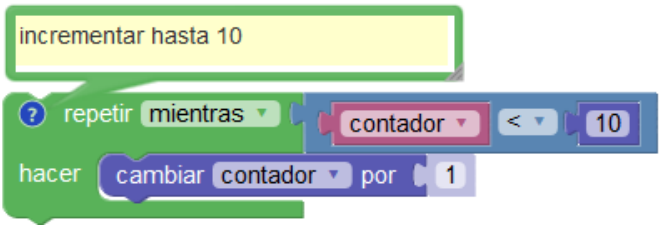




- **Comentarios:** Si necesitamos añadir un comentario a un bloque desplegamos las opciones del bloque pinchando con el botón derecho y añadimos un comentario pinchando en el icono del interrogante:

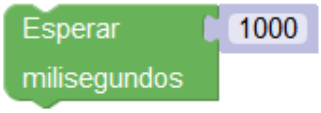






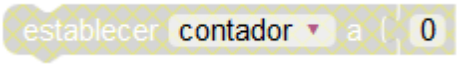
- **Otras opciones de bloque** (botón derecho sobre el bloque):

  - **Duplicar**: Crea una copia del bloque actual.
  - **Entradas en línea**: Modifica el aspecto del bloque de forma compacta o en línea.

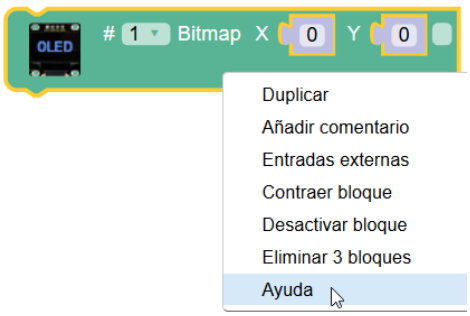


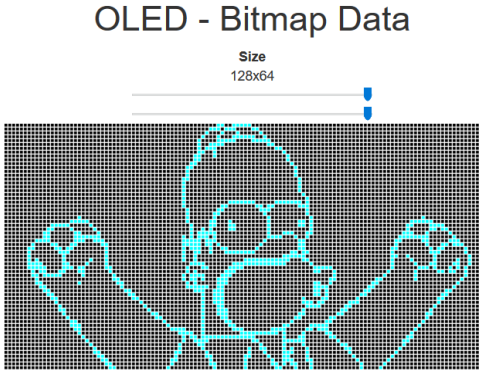
  - **Contraer / expandir bloque:** Reduce el tamaño del bloque para ahorrar espacio mientras no necesitamos editarlo.

  - **Desactivar bloque:** El generador de código no tendrá en cuenta este bloque.

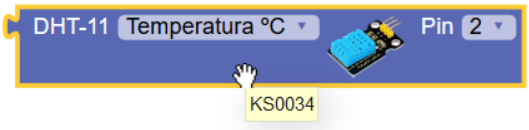


  - **Eliminar**: Elimina el bloque.
  - **Ayuda**: Abre un enlace con ayuda on información sobre el bloque. En algunos bloques la opción ayuda abrirá una aplicación específica, por ejemplo en las pantallas en el bloque Bitmap de las pantallas OLED abrirá un editor de mapas de bits





- **Ayuda contextual:** Permite información rápida del bloque, por ejemplo en los sensores podemos obtener la referencia del módulo recomendado.



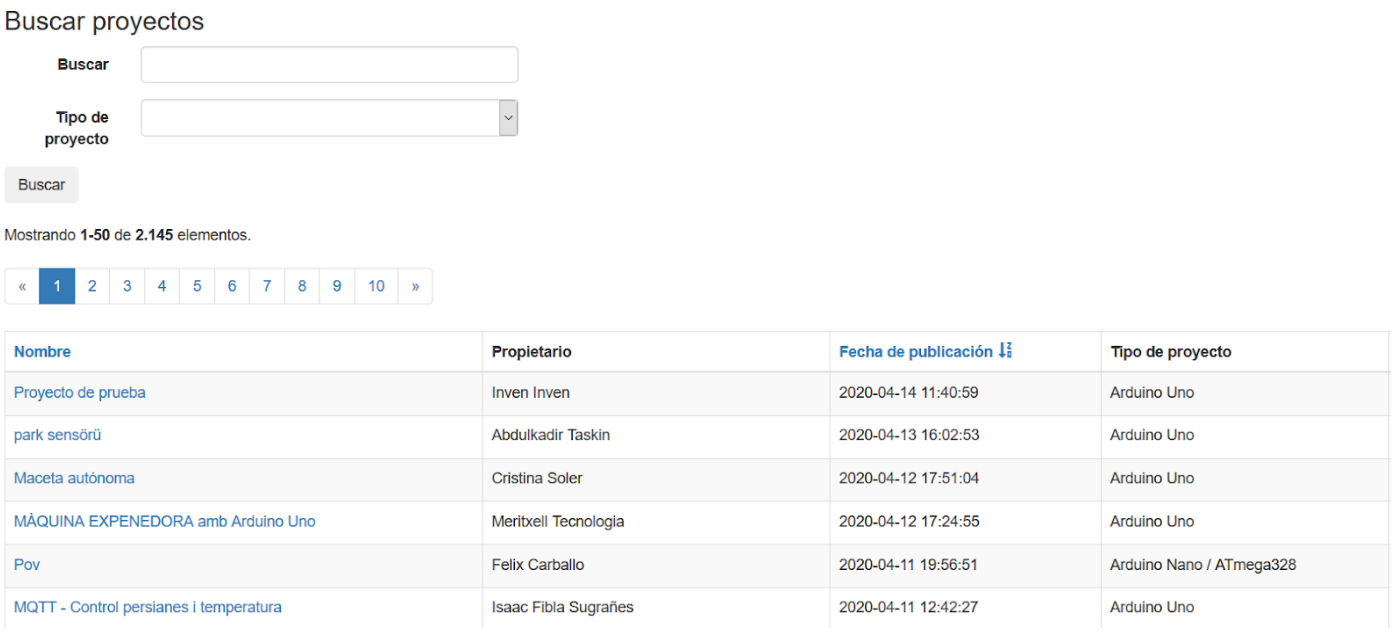
- **Iconos del editor:**

Restaurar escala y centrar.

Ampliar o reducir escala (zoom).

Arrastrando bloques sobre la papelera podemos eliminarlos fácilmente.

- **Búsqueda de proyectos compartidos por otros usuarios:** Indicando un parámetro de búsqueda podemos buscar proyectos compartidos por otro usuario y accede a visualizarlos



Buscar proyectos

Buscar

Tipo de proyecto

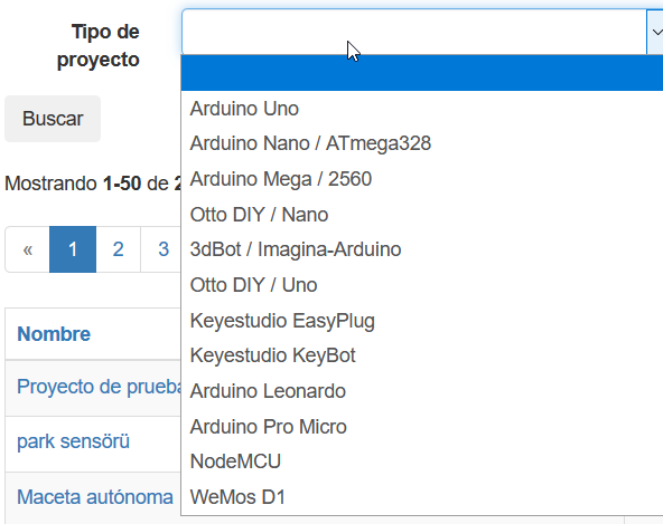
Buscar

Mostrando **1-50** de **2.145** elementos.

« 1 2 3 4 5 6 7 8 9 10 »

Nombre	Propietario	Fecha de publicación	Tipo de proyecto
Proyecto de prueba	Inven Inven	2020-04-14 11:40:59	Arduino Uno
park sensörü	Abdulkadir Taskin	2020-04-13 16:02:53	Arduino Uno
Maceta autónoma	Cristina Soler	2020-04-12 17:51:04	Arduino Uno
MÀQUINA EXPENEDORA amb Arduino Uno	Meritxell Tecnologia	2020-04-12 17:24:55	Arduino Uno
Pov	Felix Carballo	2020-04-11 19:56:51	Arduino Nano / ATmega328
MQTT - Control persianes i temperatura	Isaac Fibla Sugrañes	2020-04-11 12:42:27	Arduino Uno

Filtro para buscar por tipos de proyectos:



1.3 ArduinoBlocks - Connector

ArduinoBlocks-Connector es una aplicación nativa que hace de puente entre la plataforma on-line ArduinoBlocks y el hardware Arduino.

La aplicación ArduinoBlocks-Connector se encarga de recibir el código generado por ArduinoBlocks, compilarlo y subirlo a la placa Arduino, sin esta aplicación ArduinoBlocks funciona pero no puede subir el programa a la placa Arduino pues el navegador web no dispone de posibilidad de realizar estas funciones por sí sólo.

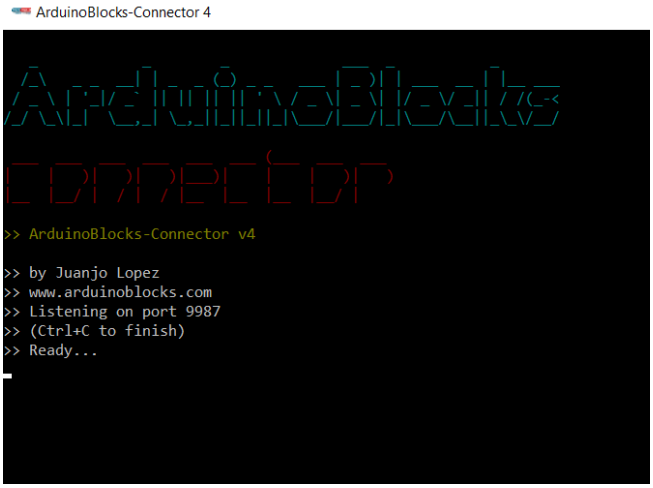
ArduinoBlocks-Connector está disponible para los principales sistemas operativos. Accede al área de descargas de arduinoblocks.com para obtener la última versión y más información sobre el proceso de instalación y configuración.

http://www.arduinoblocks.com/web/site/abconnector

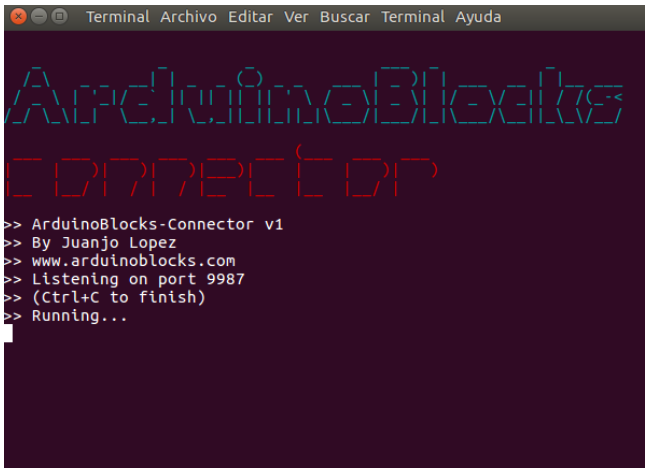
Windows, Linux (debian/ubuntu), macOS, RaspberryPi

Ejemplo: ventana de la aplicación ArduinoBlocks-Connector en Windows y en Linux

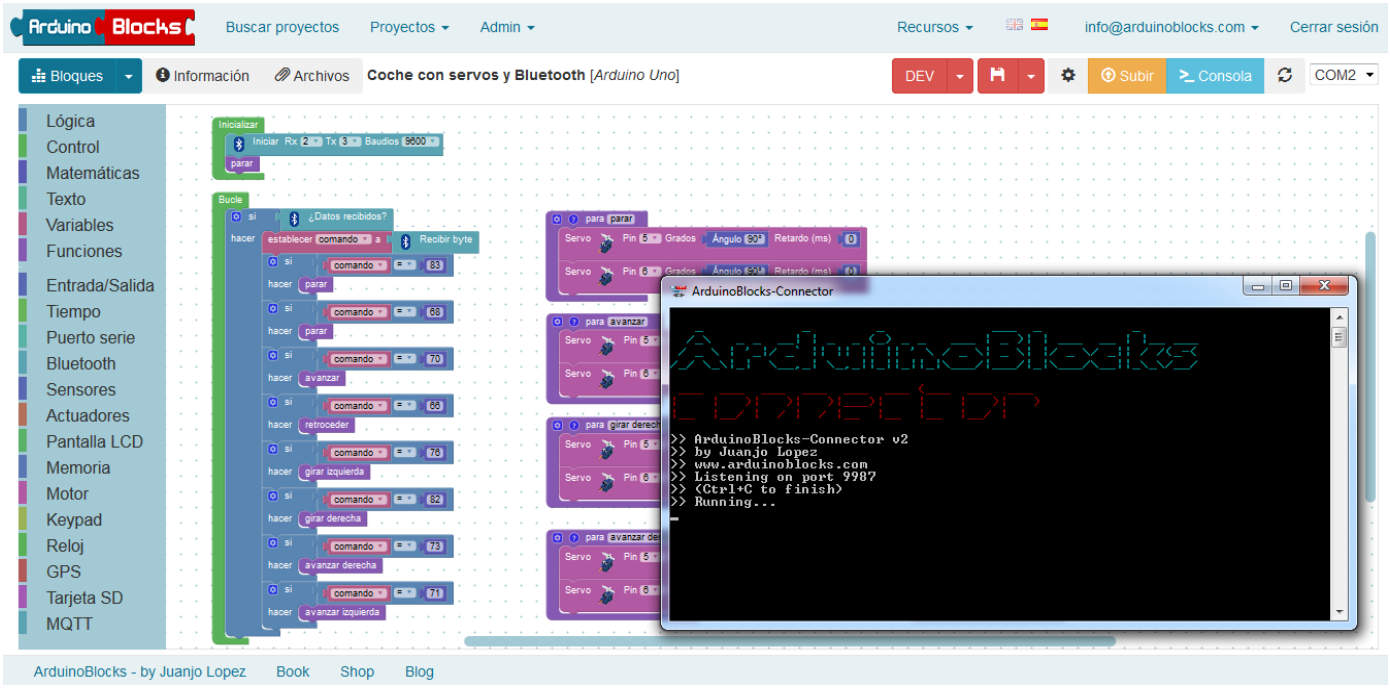
ArduinoBlocks-Connector ejecutándose bajo Windows



ArduinoBlocks-Connector ejecutándose bajo Ubuntu

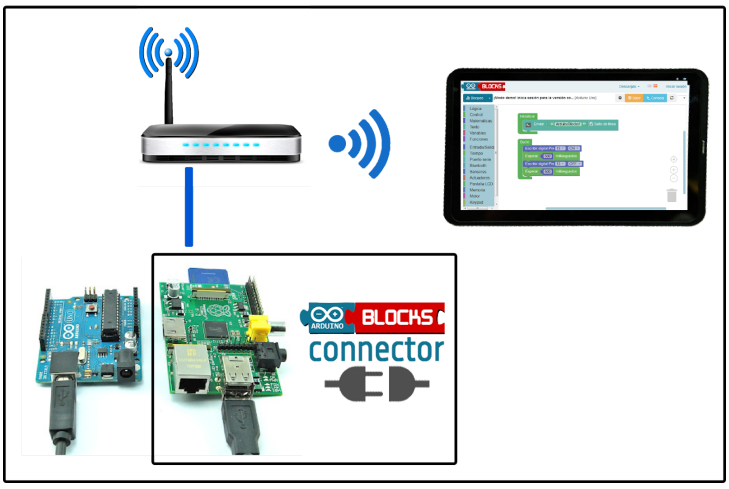


ArduinoBlocks-Connector debe ejecutarse en el equipo donde está conectado Arduino físicamente (por conexión USB). La configuración normal es instalar ArduinoBlocks-Connector en el mismo equipo donde se trabaja con ArduinoBlocks.



Podemos configurar la plataforma ArduinoBlocks para conectarse con la aplicación ArduinoBlocks-Connector en otro equipo.

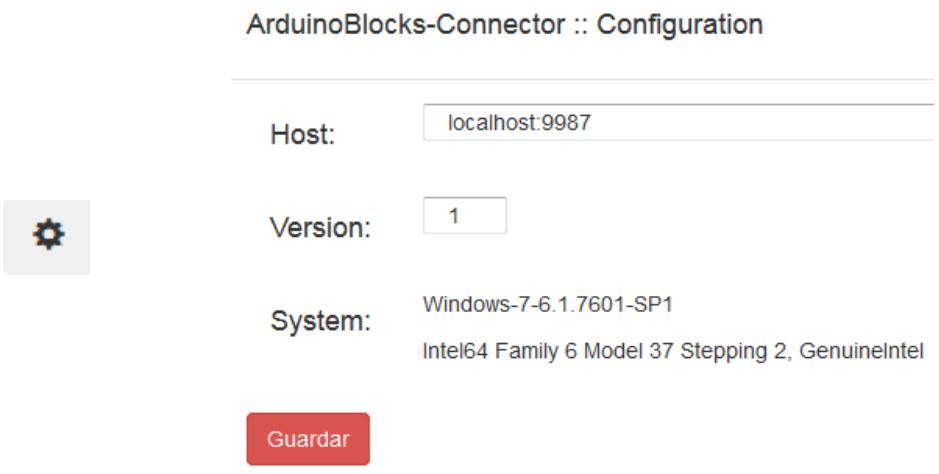
Ejemplo: Arduino conectado por USB a una RaspberryPi con ArduinoBlocks-Connector instalado. La programación sería realizada desde una Tablet Android. El programa se compila y sube a la placa Arduino de forma remota a través de la red local.



Para programar remotamente un Arduino conectado a un ordenador en red con la aplicación ArduinoBlocks-Connector, debemos modificar el Host con la dirección IP del equipo en la red al que está conectado la placa Arduino (en lugar de localhost).

Ventana de configuración al ArduinoBlocks-Connector,

por defecto en el mismo equipo (localhost):



2 Hardware

La parte hardware del proyecto Arduino está formada por el conjunto de placas Arduino disponibles en el mercado o que tú mismo te puedes fabricar (Arduino es un proyecto totalmente abierto)

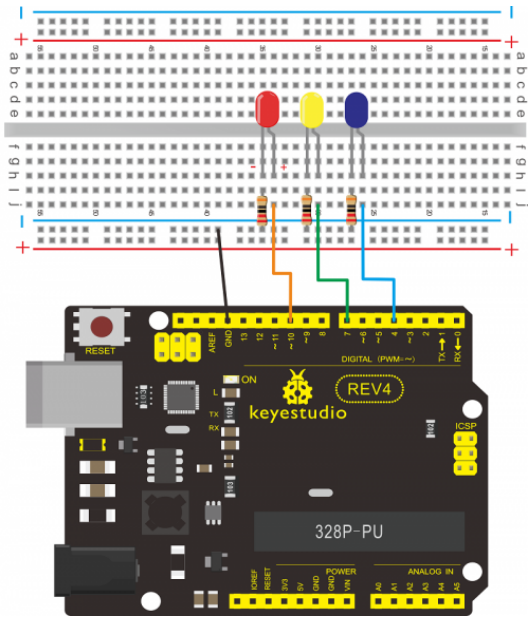
Además de la placa Arduino para cualquier proyecto robótico o de automatización debemos añadir un conjunto de sensores y actuadores para realizar las funciones necesarias.

Las conexiones entre sensores, actuadores y Arduino se pueden realizar mediante la ayuda de una placa de prototipos, de forma modular, o incluso con kits de conexión fácil (EasyPlug).

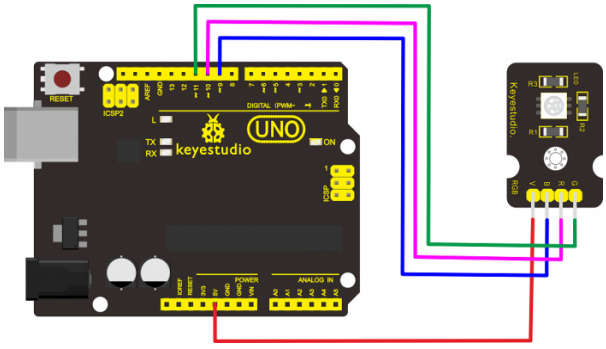
ArduinoBlocks no restringe al uso de ningún hardware en concreto, aunque con fines educativos y para facilitar el uso recomendamos el uso de componentes modulares o de fácil conexión para los más pequeños.

ArduinoBlocks toma como referencia en sistemas modulares y fácil conexión los kits del fabricante Keyestudio por su calidad y facilidad, aunque se pueden utilizar otros fabricantes e incluso fabricarte tus propios sensores y actuadores a partir de los elementos electrónicos.

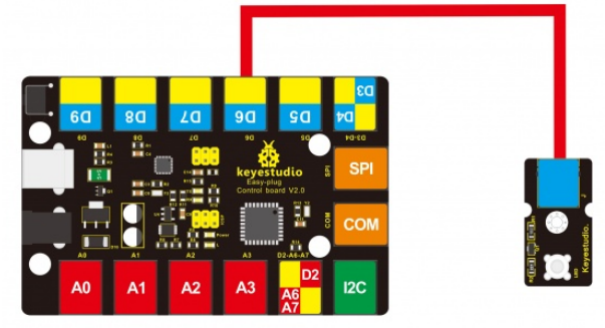
Placa prototipos / Breadboard



Modular



EasyPlug



Ejemplo: Algunos Kits de Arduino modular y EasyPlug

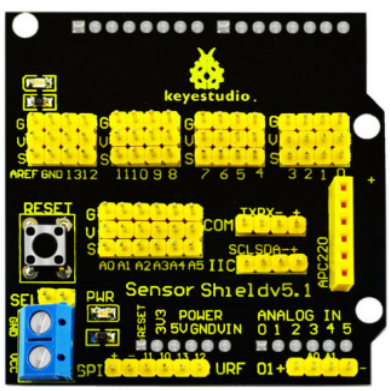
Modular: http://shop.innovadidactic.com/index.php?id_category=87&controller=category

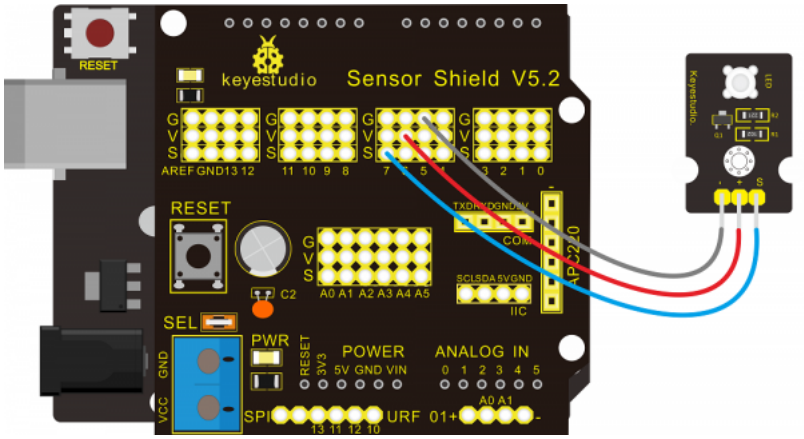
EasyPlug: http://shop.innovadidactic.com/index.php?id_category=86&controller=category

En caso de utilizar un sistema modular (por lo general es lo más recomendable para la mayoría de casos y entornos educativos), tendremos que conectar distintos sensores y actuadores a la placa Arduino por lo que tendremos que seguir usando en algunos casos la placa de prototipos para “ramificar” algunos pines como por ejemplo la alimentación de 5v o 3.3v o los pines de GND.

Como alternativa a la placa de prototipos para el conexionado modular tenemos varias opciones:

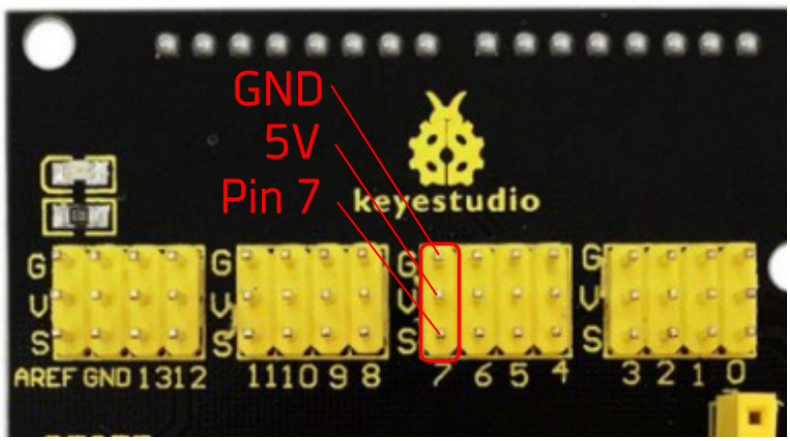
- Sensor shield: Es una shield sin electrónica extra, sustituye a la protoboard facilitándonos pines extras de alimentación (5v y GND) junto a cada pin digital o analógico. Además nos facilita la conexión en los pines I2C o de comunicación serie. Esta shield se puede añadir a cualquier placa Arduino (existe para los distintos modelos más populares).





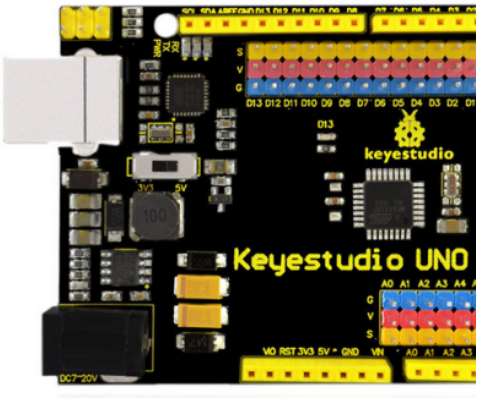
La placa sensor shield agrupa los pines G (GND) , V (5v) , S (Señal del pin correspondiente)

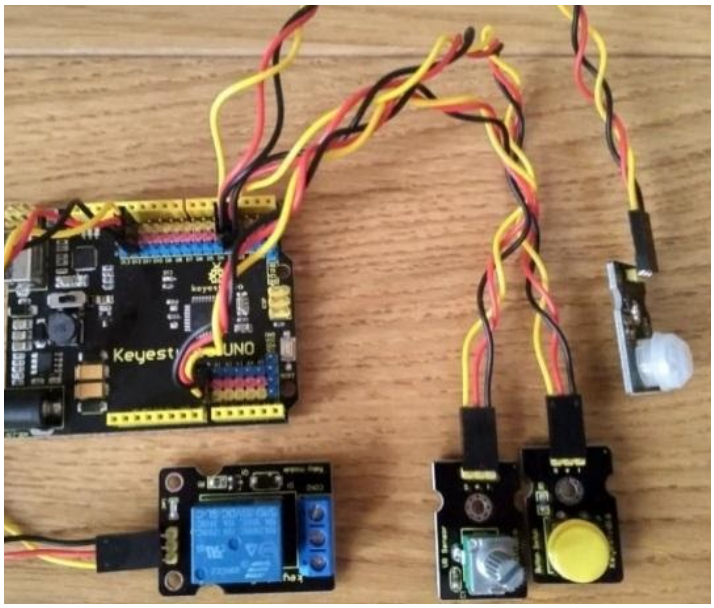
Por tanto si ampliamos, por ejemplo para el pin 7, tendriamos los pines:



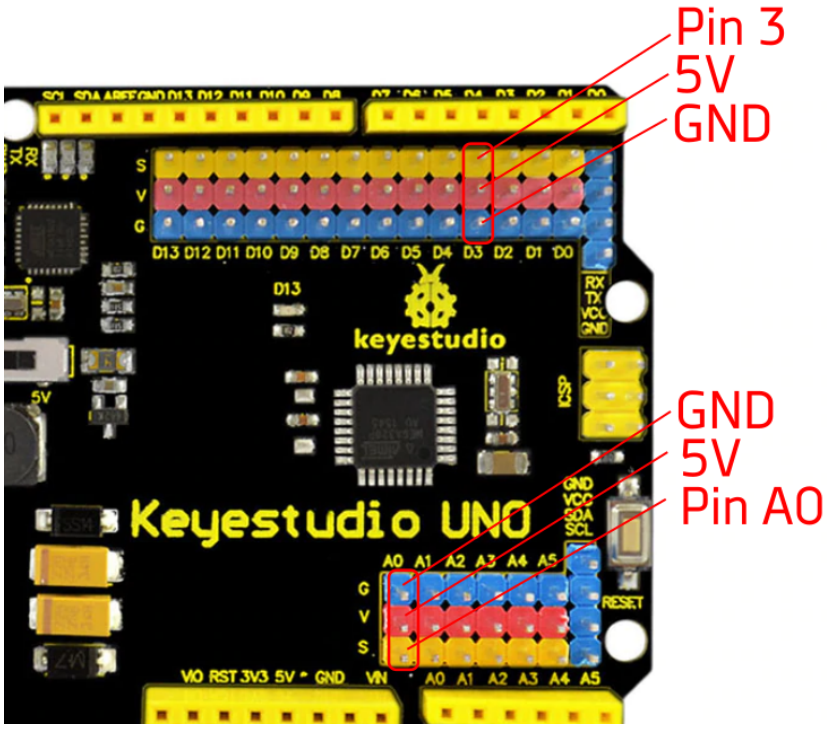
Internamente todos los pines G (GND) y V (5v) están interconectados entre ellos.

- Arduino con pines extras: Fabricantes como Keyestudio fabrican modelos de Arduino mejorados como éste, donde tenemos los pines normales (hembra) ya en la misma placa los pines extra de alimentación con los pines de señal (machos) además de la conexión I2C y serie. Es quizás la opción ideal que sirve como Arduino estándar, y en caso de necesitarlo equivale a un Arduino + Sensor shield todo en uno.





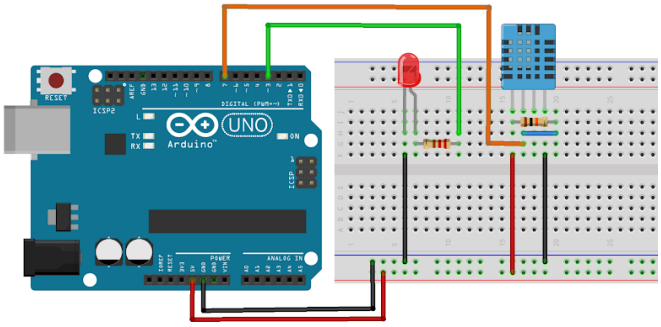
Ejemplo del detalle de las conexiones “extras” para los pines D3 y A0.



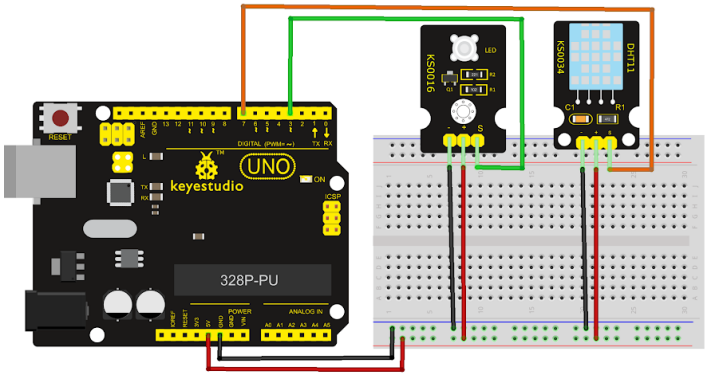
Internamente todos los pines G (GND) y V (5v) están interconectados entre ellos.

Comparativa de conexionado con Arduino UNO (Led y sensor DHT11)

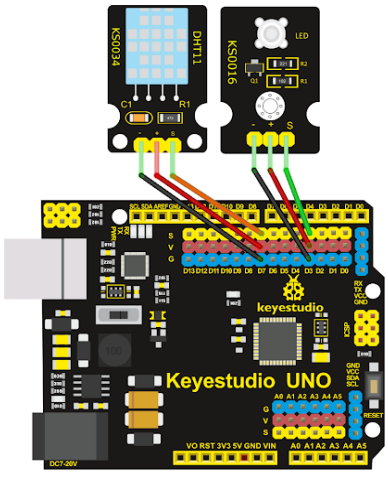
- **Componentes electrónicos con protoboard:** Permite conectar cualquier componente electrónico a Arduino sin soldar. Implica conocer el conexionado y funcionamiento detallado del componente y en la mayoría de casos componentes extras como resistencias, condensadores, etc. Este sistem requiere unos conocimientos más avanzados de electrónica.



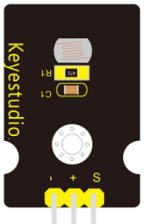
- **Módulos con protoboard:** Permite fácilmente conectar módulos usando la protoboard para ramificar los pines de alimentación (5V y GND). También se puede combinar con el sistema anterior, mezclando módulos y componentes por separado en la misma protoboard.



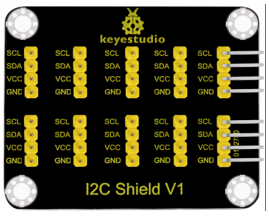
- **Módulos con sensor shield o Arduino con pines extra:** Es el sistema más sencillo, cada pin viene acompañado de los pines de alimentación por lo que no necesitamos la protoboard para ramificar las conexiones. Los módulos unifican sus pines de forma sencilla: 5V (+), GND (-) y las señales de entrada o salida correspondientes. Existen placas para ramificar los pines i2c también si fuera necesario (hub i2c).



Ejemplo: Módulo sensor de luz LDR



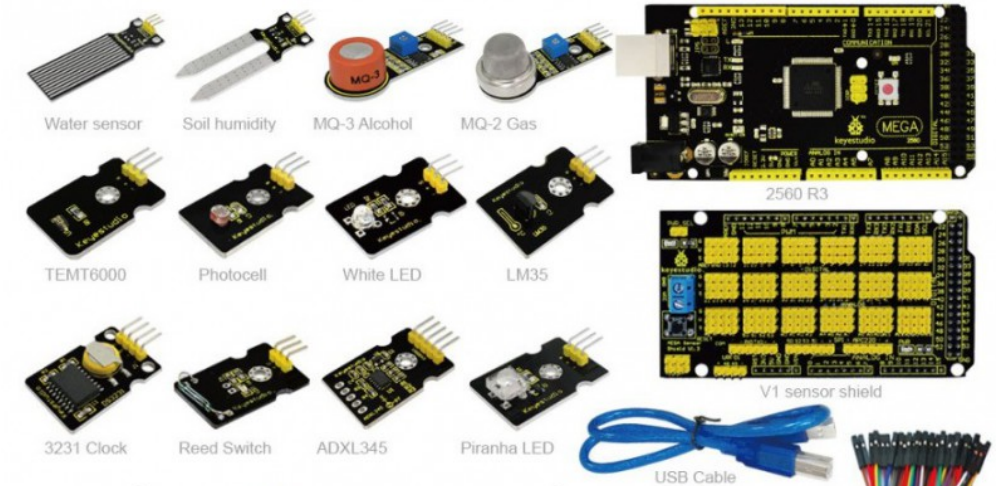
Ejemplo: hub i2c para ramificación i2c



Kits Arduino modulares:

http://shop.innovadidactic.com/index.php?id_product=665&controller=product

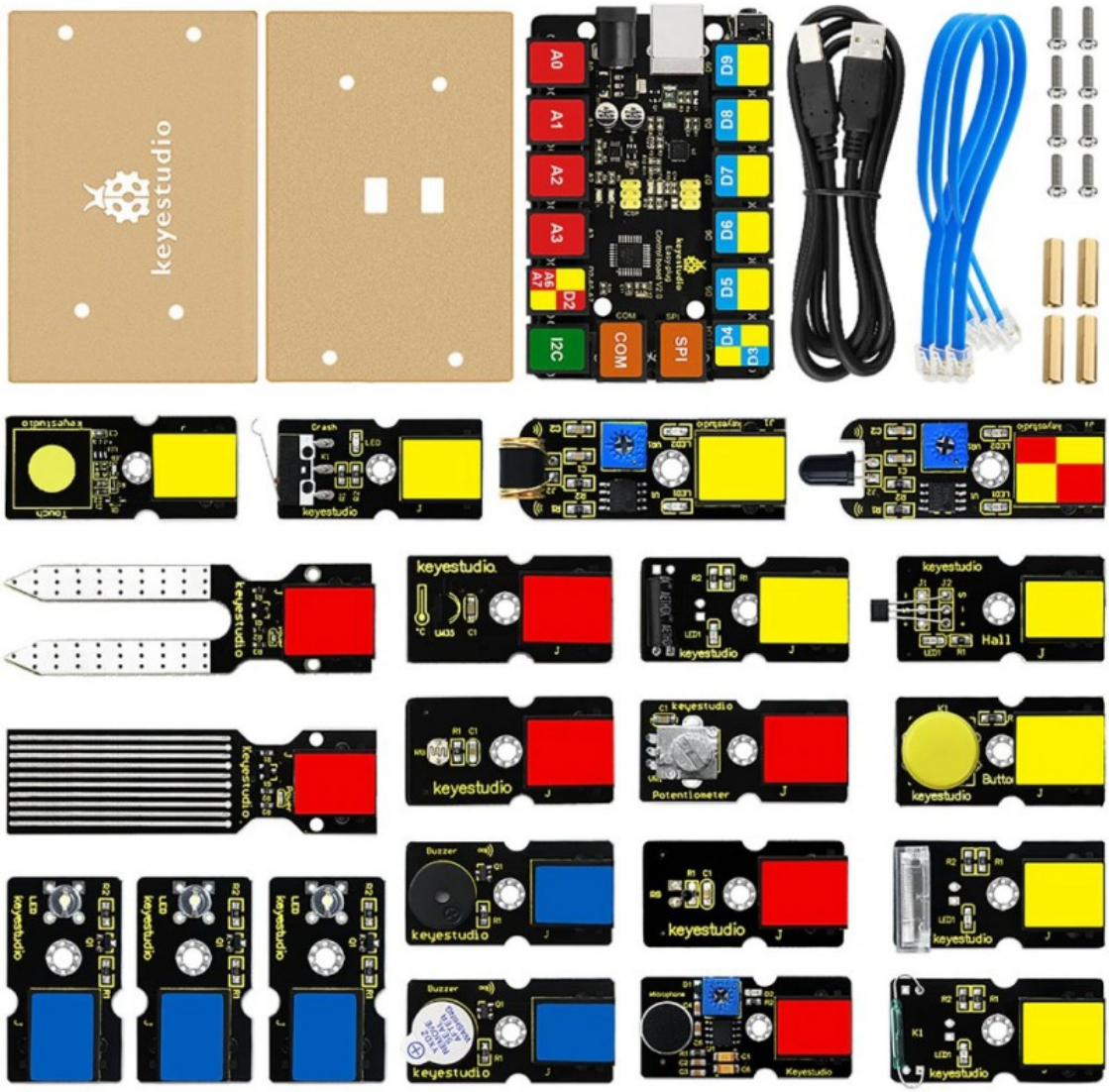
http://shop.innovadidactic.com/index.php?id_product=666&controller=product



Kit EasyPlug + módulos con conexión RJ

http://shop.innovadidactic.com/index.php?id_product=578&controller=product

Se trata de una versión de Arduino personalizada con conectores RJ en vez de pines. Facilita la conexión y con la ayuda de colores evita equivocacioens. Un opción ideal para los más pequeños.



Al crear un nuevo proyecto en ArduinoBlocks podemos indicar el tipo de proyecto "*EasyPlug*":

2.1 Conceptos básicos de electrónica

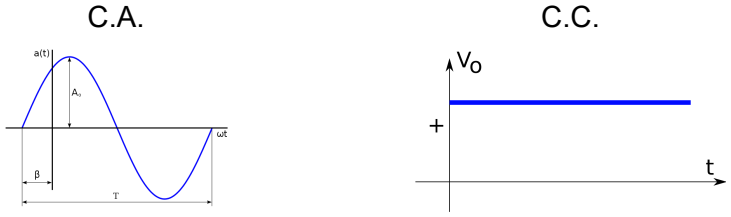
A la hora de iniciar un proyecto de robótica debemos tener claras algunas nociones de electricidad y electrónica básicas. El propósito de este libro no es aprender estos conceptos sobre electrónica, por lo tanto se asumen unos conocimientos previos básicos de electricidad y electrónica.

Vamos a hacer un breve repaso de los conceptos más importantes que debemos conocer:

La corriente eléctrica se define como el movimiento de los electrones a través de un conductor, según el tipo de desplazamiento se define como corriente continua o alterna.

En la corriente alterna los electrones cambian de dirección de movimiento 50 veces por segundo (en redes eléctricas como la de España de 50Hz, en otros países puede ser 60Hz). El movimiento descrito por los electrones es sinusoidal.

En la corriente continua los electrones se desplazan siempre en la misma dirección. Arduino funciona con corriente continua.



Las principales magnitudes físicas que debemos conocer son:

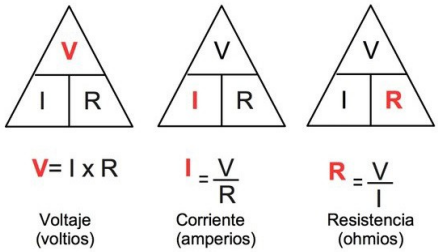
Voltaje o tensión eléctrica: Energía acumulada por unidad de carga que hace que las cargas circulen por el circuito (genera una corriente). Se mide en voltios (V)

Intensidad: número de electrones que atraviesan la sección de un conductor por unidad de tiempo. Se mide en amperios (A)

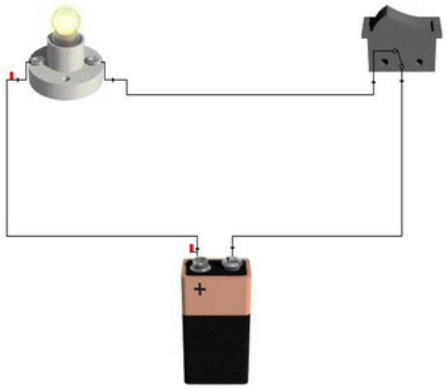
Resistencia: mide la oposición que ofrece un material al paso de corriente eléctrica. Se mide en Ohmios (Ω)

Potencia: es la energía consumida o desprendida por un elemento en un momento determinado. Se mide en Watios (W) $P = V \cdot I$

Ley de Ohm: Es una sencilla fórmula matemática que relaciona estas tres magnitudes básicas entre sí.



Circuito eléctrico: Conjunto de elementos unidos de tal forma que permiten el paso de corriente eléctrica para conseguir algún efecto (luz, calor, movimiento, etc.)



2.2 La fuente de alimentación

La placa Arduino necesita energía para funcionar, existen varias formas de alimentar la placa Arduino:

-**A través del conector USB:** cuando conectamos al ordenador para programarlo o utilizando un “power bank” con conexión USB por ejemplo.

-**A través del conector de alimentación externa:** La fuente de alimentación conectada debe ofrecer un voltaje DC de 7 a 12v. Internamente la placa Arduino UNO regula la tensión a 5v.

-**A través de los pines 5v o Vin:** A través del pin 5v podemos alimentar con una fuente de alimentación de 5v, o a través del pin Vin podemos alimentar de igual forma que con el conector externo con un voltaje de entre 7 a 12v, pues se ajutará a 5v con el regulador interno.

En el caso de alimentar desde USB o a través del conector externo, por los pines 3.3v , 5v, GND y Vin obtenemos la alimentación para circuitos auxiliares, sensores, shields, etc.:

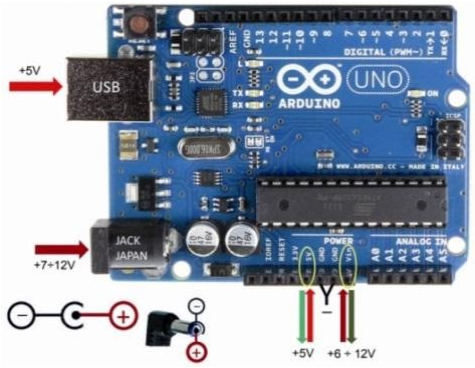
• 3.3v proporciona una tensión de 3.3v y una corriente máxima de 50mA

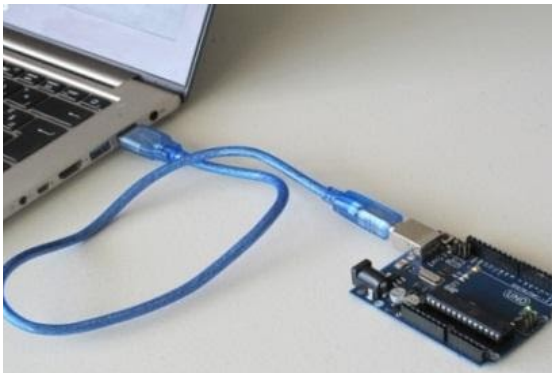
• 5v proporciona una tensión de 5v y una corriente máxima de 300mA

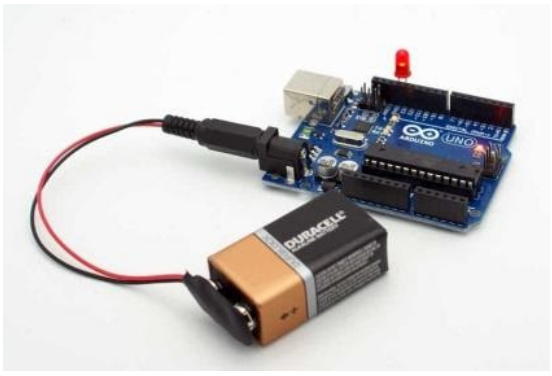
• GND es el nivel 0v de referencia

• Vin proporciona la tensión de alimentación conectada al conector de alimentación (sin regular, igual a la tensión de la fuente de alimentación conectada)

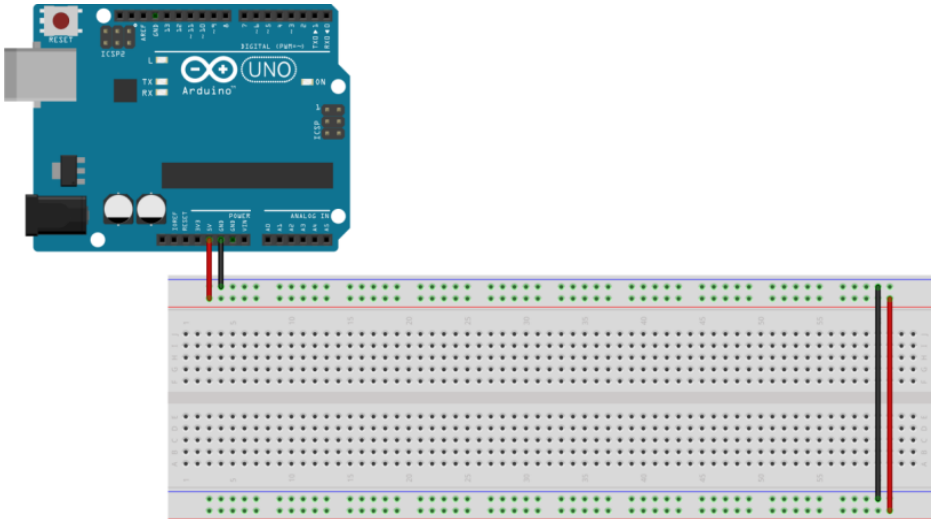
Normalmente alimentaremos la placa Arduino a través del USB durante su programación desde el PC. Si la placa Arduino va a funcionar de forma autónoma sin interactuar con el PC podemos alimentarla desde una fuente de alimentación o con una batería a través del conector Jack (aplicar de 7 a 12v).







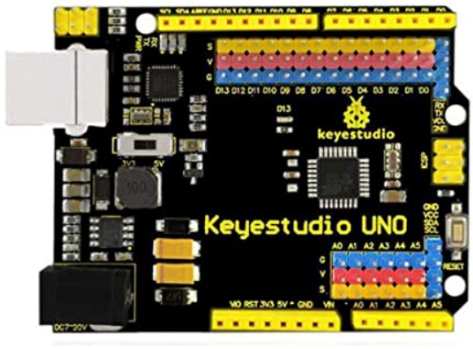
Ejemplo: Conexión recomendada de la tensión de alimentación a la placa de prototipos

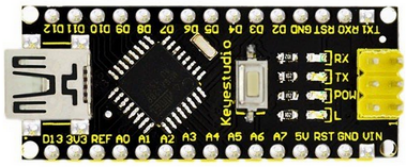


Placa de prototipos o "*breadboad*": https://es.wikipedia.org/wiki/Placa_de_pruebas

2.3 Arduino: UNO y Nano

Arduino UNO es la placa Arduino más utilizada de todas las versiones existentes, Arduino Nano es muy similar a Arduino UNO en características pero en un formato más pequeño. Aqui se pueden ver dos modelos de la placa Arduino UNO y Arduino Nano:





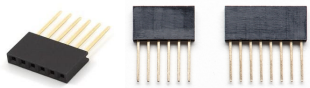
http://shop.innovadidactic.com/index.php?id_product=690&controller=product

Especificaciones técnicas:

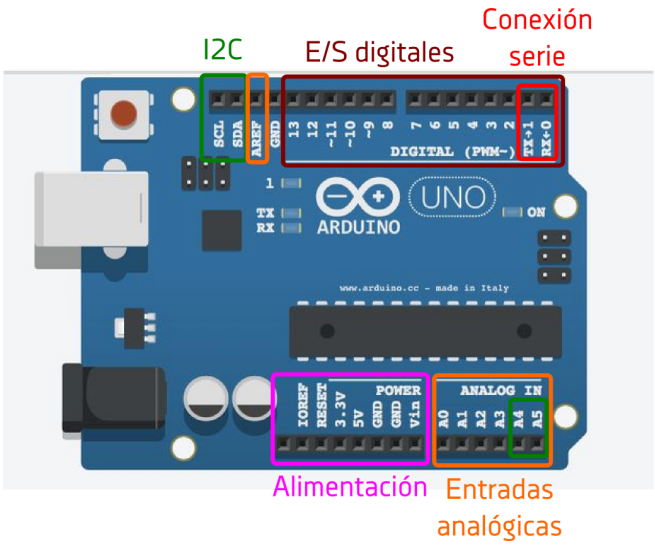
Microcontrolador	ATmega328P
Alimentación	5V
Alimentación (recomendada)	7-12V
Alimentación (límite)	6-20V
Número de pines E/S	14 (6 con salida PWM)
Número de pines PWM	6
Número de pines analógicos	6
Corriente pines E/S	20 mA
Corriente pin de 3.3V	50 mA
Memoria Flash	32 KB (ATmega328P) (0.5 KB para el bootloader)
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Velocidad de reloj	16 MHz
Largo	68.6 mm
Ancho	53.4 mm
Peso	25 g

El tamaño de la memoria para programa de la placa Arduino UNO es de 32 KBytes.

La placa Arduino UNO dispone de múltiples pines de conexión en formato de conector hembra:



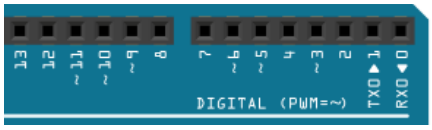
Los pines están agrupados por función o tipo:



- **Pines de alimentación:** Permiten obtener la tensión necesaria para alimentar sensores, actuadores u otros periféricos conectados a la placa Arduino

IOREF	Indica la tensión de trabajo de las E/S de este modelo de placa. (Arduino UNO IOREF = 5V)
RESET	Permite reiniciar la placa a través de este pin
3.3V	Suministra 3.3v
5V	Suministra 5v
GND	Tierra o 0V (negativo)
Vin	Obtiene el voltaje aplicado por la fuente de alimentación con la que se está alimentando. También permite alimentar la placa por este pin, aplicando tensión de entrada (7-12v)

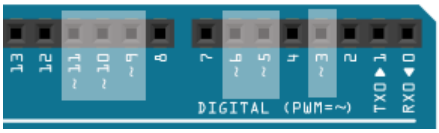
- **Pines de entradas/salidas digitales:** Los pines digitales permiten trabajar con dos estados (ON/OFF, Activado/Desactivado, 1/0). Los pines se pueden configurar como entrada o como salida según se necesite conectar un sensor o un actuador.



Pin digital configurado como entrada:	Tensión aplicada externamente al pin: 0v…1,5v = OFF / 0 / desactivado 3v…5v = ON / 1 / activado
Pin digital configurado como salida:	Tensión suministrada por el pin: OFF / 0 / desactivado = 0v ON / 1 / activado = 5v

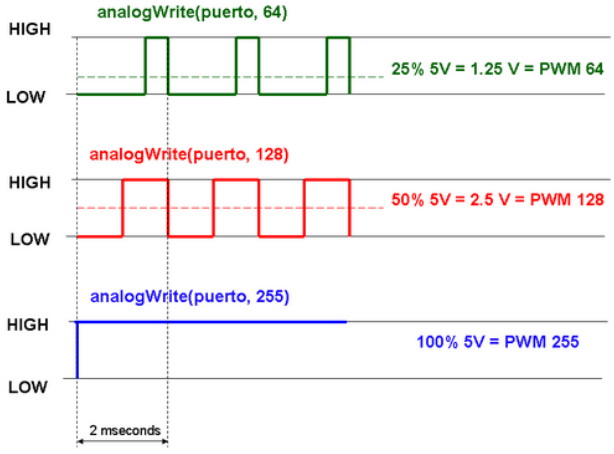
Pines: 0, 1	Estos dos pines se pueden utilizar como entradas / salidas digitales. ArduinoBlocks no utiliza los pines 0,1 como pines de E/S. Los reserva para la conexión serie y la programación desde el PC.
Pines: 2…13	Pines digitales de uso general. Podemos utilizarlos como entrada o salida. Según utilicemos un actuador o un sensor ArduinoBlocks configurará automáticamente el pin como entrada o salida según sea necesario.
Pines A0…5	Los pines de entrada analógicos también se pueden usar como pines de E/S digitales convencionales.

- **Pines de salidas analógicas (PWM):** Arduino no tiene salidas puramente analógicas, pero podemos imitar a una salida analógica mediante la técnica PWM (Pulse Width Modulation = Modulación en Anchura de Pulso).



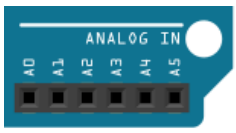
Pines: 3,5,6,9,10,11	Dentro de los pines digitales estos pines permiten utilizar como salida digital PWM (modulación en ancho de pulso) para simular una salida pseudo-analógica.

Ejemplo: Gráficas del funcionamiento del PWM:



- **Pines de entradas analógicas:** Estos pines sólo se pueden utilizar como entradas. Las entradas analógicas leen un voltaje entre 0 y 5V y a través de un ADC (Analog to Digital Converter) obtienen un valor de 10 bits proporcional a la señal de entrada.

10 bits = 1024 valores (0 … 1023)



Pines: A0…A5	6 Entradas analógicas (resolución 10 bits: 0…1023)

- **Pines de comunicación serie:** Estos pines conectan con la unidad serie (UART) interna del microprocesador de Arduino. Una conexión serie utiliza un pin para la señal de envío de datos (TX) otro para la recepción de datos (RX) y la señal GND.

Pin 0	RX: a través de este pin se reciben datos hacia Arduino
Pin 1	TX: a través de este pin se envían datos desde Arduino

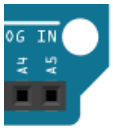
Los pines 0,1 conectan con el puerto serie implementado en el hardware Arduino. En caso de necesidad se podrán implementar otras conexiones serie a través de otros pines digitales emulando el puerto serie con librerías software (por ejemplo para la conexión con el módulo Bluetooth HC-06 explicado más adelante)

https://www.arduino.cc/en/Reference/SoftwareSerial

(más información sobre la conexión serie: Apartado 2.6.1)

- **Pines de comunicación I2C:** El bus de comunicación I2C permite conectar redes de periféricos con una comunicación bidireccional entre Arduino y el periférico.

Pin A4 / SDA	Línea de datos del bus I2C
Pin A5 / SCL	Línea de reloj del bus I2C



(más información sobre el bus I2C: Apartado 2.6.2)

- **Pines de comunicación SPI:** El bus de comunicación SPI permite conectar redes de periféricos con una comunicación bidireccional entre Arduino y el periférico.

Pin 12 / MISO	Master In Slave Out
Pin 11 / MOSI	Master Out Slave In
Pin 13 / SCK	Serial Clock
Pin 10 / SS	Slave Select

(más información sobre bus SPI: Apartado 2.6.3)

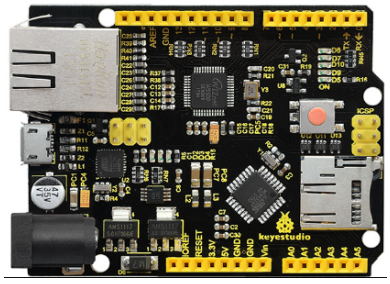
Una de las ventajas de la placa Arduino es que incorpora un programa pregrabado en el microcontrolador. Este programa conocido como “bootloader” o cargador de arranque permite desde el principio reprogramar el microcontrolador de Arduino a través de su puerto USB sin necesidad de un programador externo ni el uso del sistema ICSP (In Circuit Serial Programming) utilizado en otros sistemas.

Otra ventaja evidente del sistema Arduino es el entorno de programación “Arduino IDE” sencillo ofrecido de forma totalmente libre y que facilita enormemente la programación de este tipo de microcontroladores para inexpertos.

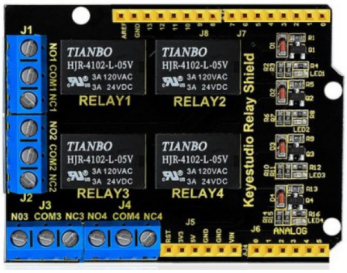
La clave del éxito de la plataforma Arduino es que es una plataforma totalmente abierta y existe una gran comunidad de colaboradores y desarrolladores. Un ejemplo de las aportaciones de la comunidad Arduino son las conocidas como “shields”, que son módulos de extensión apilables para Arduino con las que podemos añadir rápidamente funcionalidades a la placa Arduino.

Ejemplos de “Shields” para Arduino UNO:

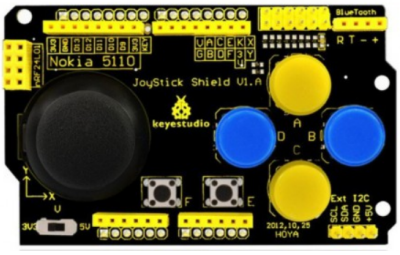
Shield Ethernet



Shield Relés



Joystick y botones



Shield Educativa TDR STEAM innovadidactic

Shield GPS

Motor shield 1

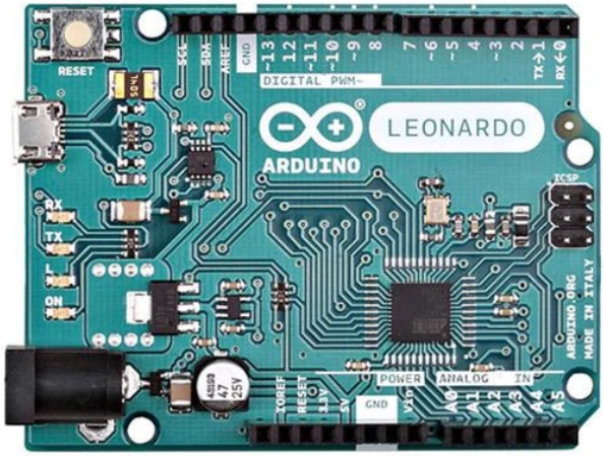
http://shop.innovadidactic.com/index.php?id_category=93&controller=category

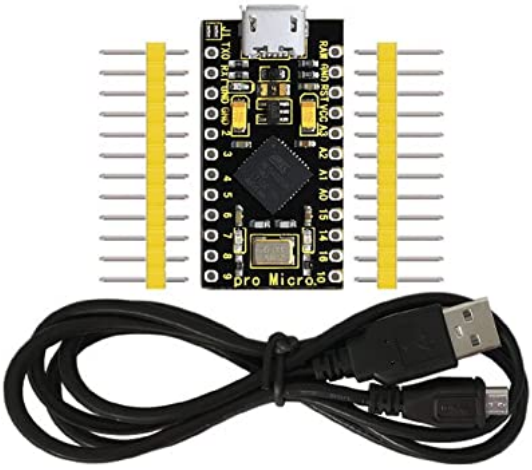
2.4 Arduino: Leonardo y ProMicro

Los modelos Arduino Nano y ProMicro son similares en características y tamaño a los modelos UNO y Nano respectivamente, la principal diferencia es que la comunicación USB para la programación y la comunicación con el PC está integrada en el mismo controlador. Gracias a esta característica estos modelos de Arduino además pueden emular el comportamiento de un dispostivio USB tipo HID (Human Interface Device), es decir, ser detectados por el PC y comportarse como un teclado o un ratón.

De esta forma es sencillo implementar periféricos HID para el control del PC o como forma de intercambiar información. Por ejemplo podemos implementar dispostivos para controlar el puntero del ratón y las pulsaciones de los botones o implementar un controlador de juegos enviando los códigos de teclas pulsadas como si de un teclado se tratara.

Más información sobre los bloques de programación de teclado y ratón





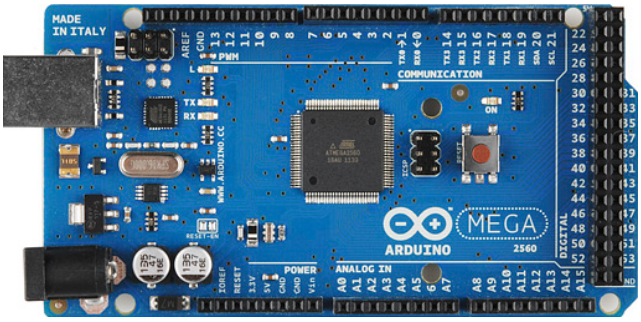
http://shop.innovadidactic.com/index.php?id_product=728&controller=product

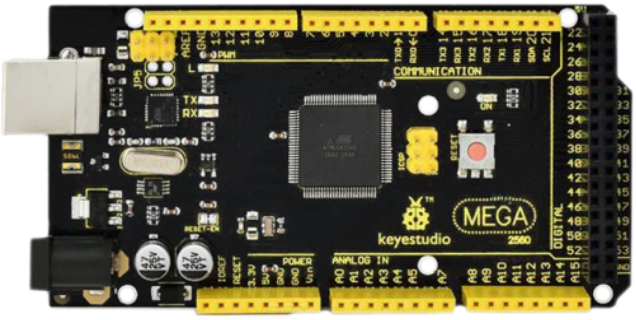
2.5 Arduino: Mega

Arduino Mega implementa un número considerablemente mayor de pines de E/S tanto digitales como analógicos, además de varios puertos serie implementados por hardware (a parte del que se utiliza para comunicación USB y programación). Por otro lado tiene una mayor cantidad de memoria RAM y memoria FLASH para almacenar el programa.

Debemos elegir este tipo de Arduino cuando nuestro proyecto vaya a necesitar un número de entradas/salidas superior al Arduino UNO o en casos de realizar programas grandes donde la memoria de Arduino UNO pueda ser un problema. Otro aspecto importante a tener en cuenta es la

posibilidad de utilizar varias conexiones serie (puertos serie) implementados por hardware en el propio microcontrolador. Mienstras que los Arduinos de gama inferior tienen un único puerto serie (compartido con la comunicación USB), Arduino MEGA tiene 4 puertos serie hardware.



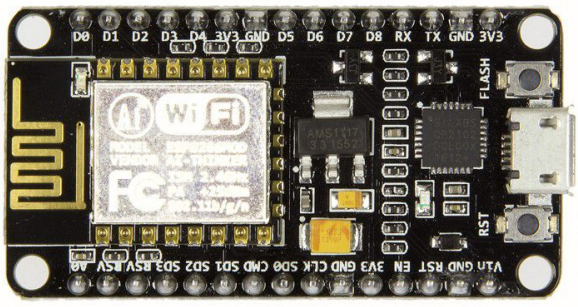


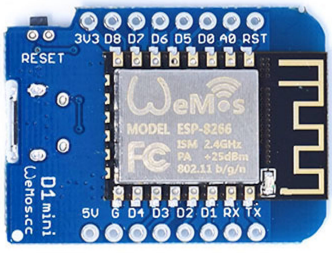
http://shop.innovadidactic.com/index.php?id_product=848&controller=product

2.6 ESP8266: NodeMCU y WeMos

El microcontrolador ESP8266 es un microcontrolador de bajo coste con características tan interensantes como el WiFi integrado, una velocidad de reloj y una tamaño de memoria mucho mayores que los modelos de Arduino. El chip ESP8266 se utiliza como base para algunas placas de desarrollo como NodeMCU o WeMos (versión normal o mini). La comunidad Arduino desarrolló un “framework” compatible con Arduino para permitir progamar las placas basadas en el micrcontrolador ESP8266 de una forma similar a Arduino, de esta forma la mayoría de librerías se han podido reutilizar o adaptar fácilmente.

Una de las características en las que se diferencian de Arduino principalmente es que trabajan con un voltaje de 3.3v en vez de 5v, por lo que hay que llevar cuidado con ésto y en algún caso será necesario utilizar adaptadores de tensión 3.3v - 5v





Tienda: http://shop.innovadidactic.com/index.php?id_product=859&controller=product

2.7 Sensores

Un sensor es un objeto capaz de detectar magnitudes físicas o químicas y transformarlas en variables eléctricas.

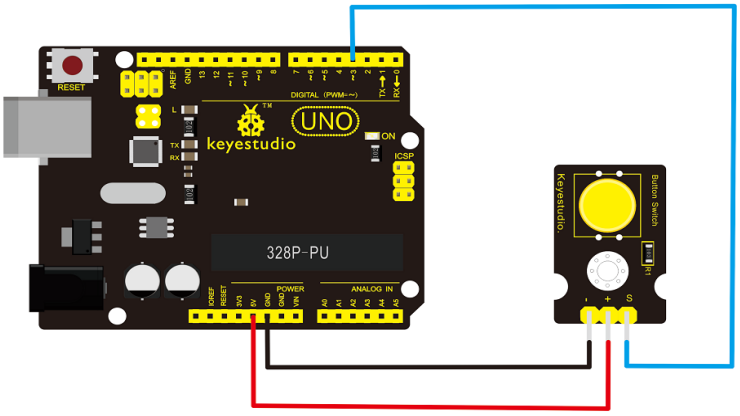
Los sensores o periféricos de entrada nos permiten obtener información del mundo real para utilizarla desde el programa de Arduino.

La interfaz de conexión de un sensor con Arduino lo podemos clasificar en tres tipos: digital, analógico o datos.

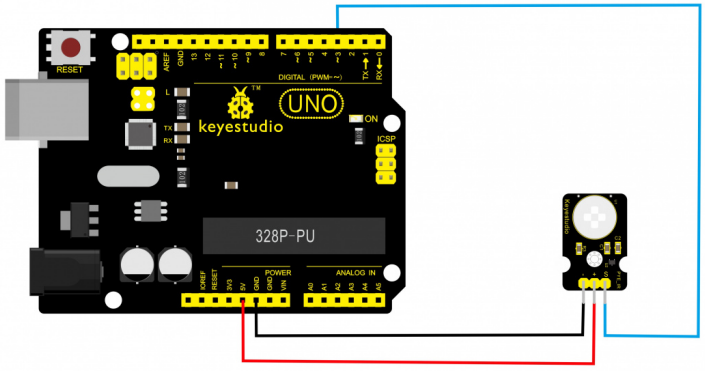
- **Digital**: un sensor digital sólo tiene dos estados: activado/desactivado, ON/OFF, 1/0, Alto/Bajo, … En este caso conectaremos el sensor a una de las entradas digitales de Arduino para leer el estado.

Ejemplo: un pulsador es un tipo de sensor sencillo que sólo nos da dos estados, “pulsado o no pulsado”. Conectado a la placa Arduino debe generar 0v en reposo y 5v al pulsarlo. De esta forma desde el programa de Arduino podremos leer el estado del botón.

Ejemplo: conexión de un sensor digital (pulsador):

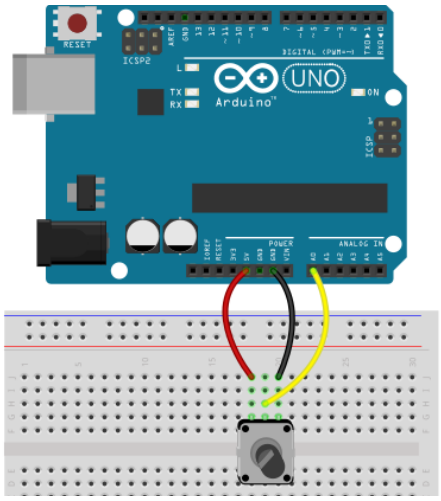


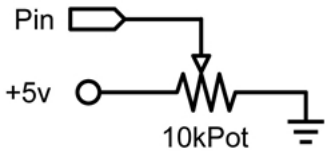
Ejemplo: Conexión de un sensor digital de movimiento (PIR):



- **Analógico**: el sensor nos puede dar un rango de valores, normalmente se traduce en un valor de tensión o de corriente variable en función de la señal captada al sensor. En este caso conectaremos el sensor a una de las entradas analógicas de Arduino. El rango de entrada será una tensión entre 0v (GND) y 5v.

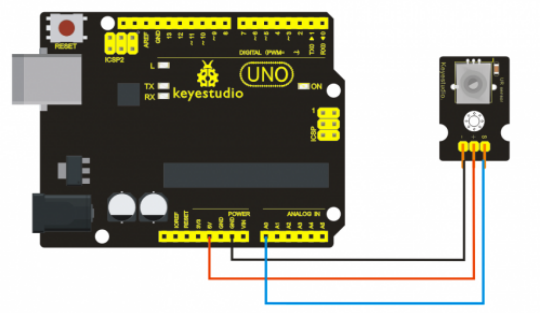
Ejemplo: Conexión de un sensor potenciómetro al pin de entrada analógico A0:





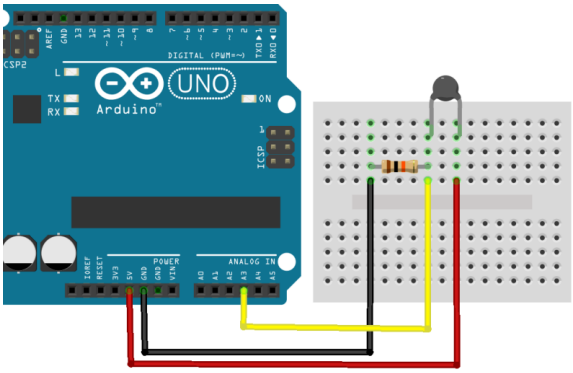
Cuando el potenciómetro está en un extremo el voltaje aplicado al pin de Arduino es 5v, en el otro extremo es de 0v. Durante el recorrido, gracias a la variación de resistencia del potenciómetro, se aplica el valor de voltaje proporcional a la posición del potenciómetro entre 0 y 5 voltios.

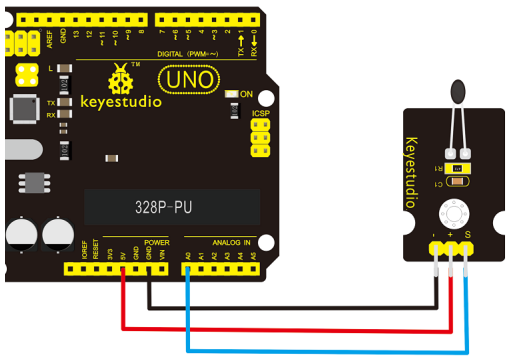
(versión modular)



Ejemplo: Conexión de una resistencia NTC (variable según la temperatura):

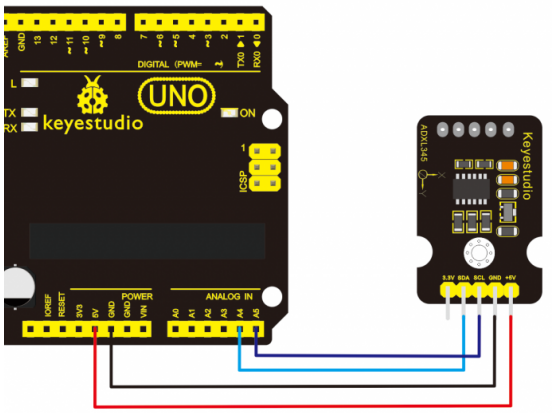
(versión modular)



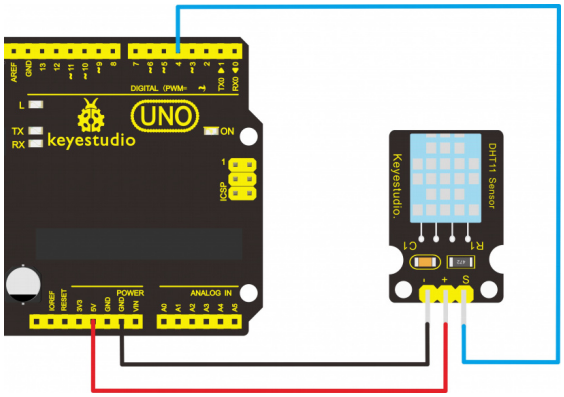


- **Datos**: el sensor ofrece su información a través de una interfaz de comunicación. La forma de comunicación puede ser por sistemas estándar como I2C o SPI (ver apartado 2.6 sobre buses de comunicación) o algunos sensores usan su propio protocolo para codificar la información y debemos realizar desde el software la decodificación correcta para interpretar los datos del sensor (normalmente los desarrolladores de este tipo de sensores ofrecen una librería software para Arduino que hace todo el trabajo)

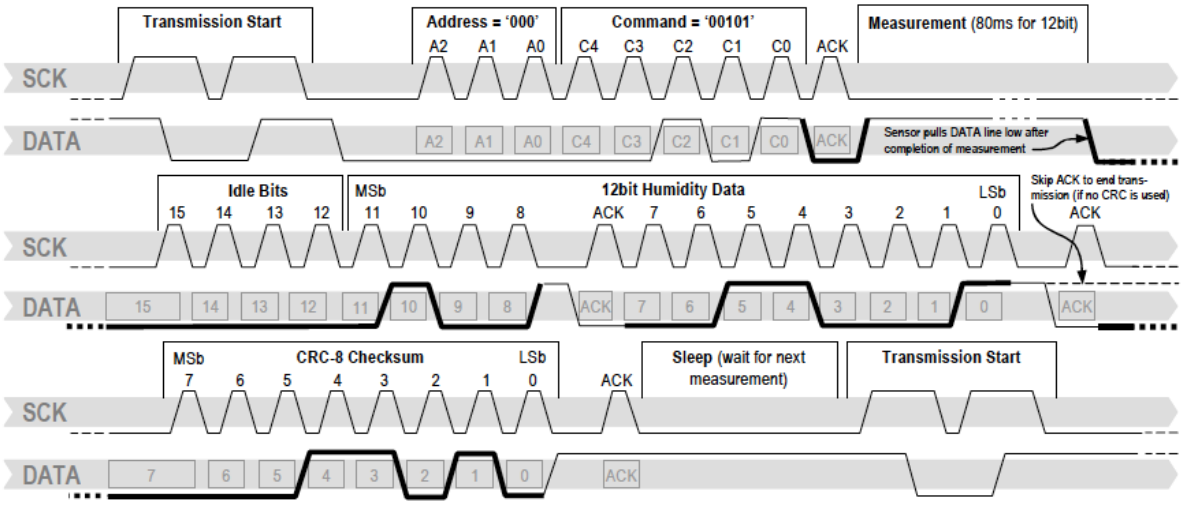
Sensor de acelerómetros con conexión I2C



Sensor DHT11: temperatura y humedad con protocolo de comunicación propio

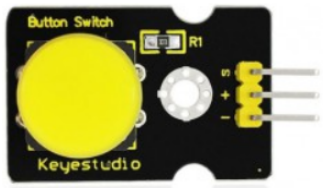


Ejemplo: Trama de datos recibida desde el sensor DHT11



Algunos módulos de sensores utilizados con Arduino:

Pulsador



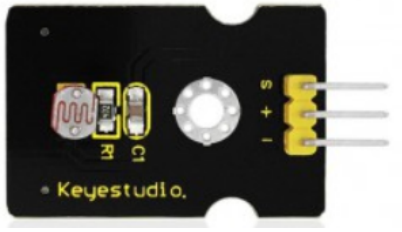
Potenciómetro deslizante

Sensor de distancia

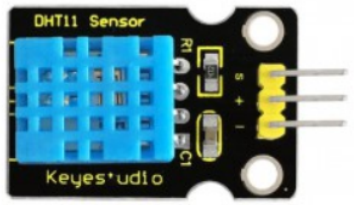
Sensor de temperatura NTC

Sensor magnético

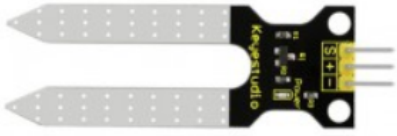
Sensor de luz LDR



Sensor DHT-11 de temperatura y humedad

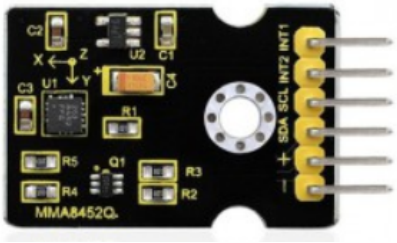


Sensor humedad suelo

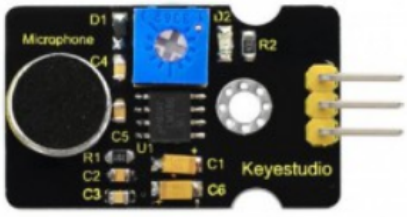


Encoder

Acelerómetro



Sensor de sonido



Receptor IR

http://shop.innovadidactic.com/index.php?id_category=87&controller=category

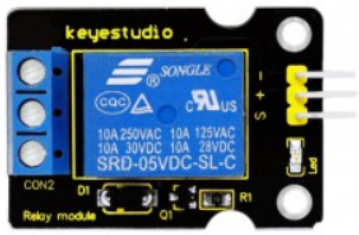
2.8 Actuadores

Un actuador es un dispositivo capaz de transformar la energía eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre elementos externos.

Un actuador o periférico de salida permite actuar sobre el mundo real desde el programa de Arduino.

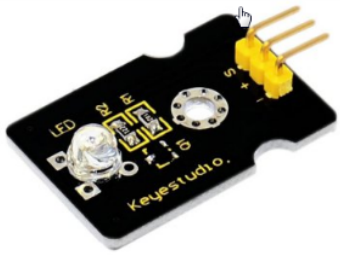
Algunos módulos de actuadores utilizados con Arduino:

Módulo relé:



Servomotor:

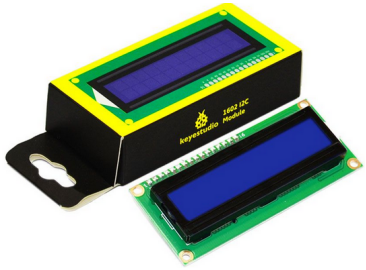
Módulo led:



Módulo led RGB:

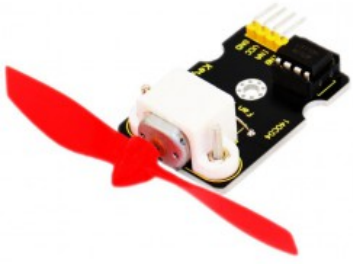
Módulo zumbador:

Pantalla LCD:

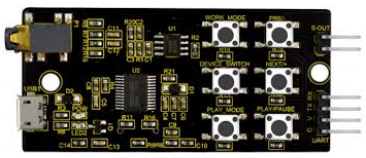


Motor paso a paso:

Motor DC:



Reproductor MP3:



http://shop.innovadidactic.com/index.php?id_category=87&controller=category

2.9 Comunicaciones

La placa Arduino permite múltiples vías de comunicación con el exterior, por un lado disponemos del bus I2C o del SPI pensado para periféricos externos o sensores mientras que como vía de comunicación principal para la programación o monitorización tenemos la conocida como conexión serie (puerto serie) a través del conector USB.

2.9.1 Comunicación serie

El microcontrolador Atmel de Arduino dispone de un controlador de comunicación serie (UART) integrado. La comunicación se realiza de forma bidireccional, utilizando un pin para transmitir los datos y otro para recibir.

Es muy importante tener en cuenta que este puerto serie es el que se utiliza para “subir” el firmware y reprogramar la placa Arduino desde un ordenador (bootloader).

Las primeras placas Arduino disponían de un conector de puerto serie tipo DB9 de 9 pines utilizado antiguamente para este tipo de conexiones. Hoy en día se utiliza un chip de conversión serie a USB que permite emular en el equipo un puerto serie estándar.

Durante el uso normal podemos utilizarlo como vía de comunicación sencilla entre el microcontrolador y el un PC.

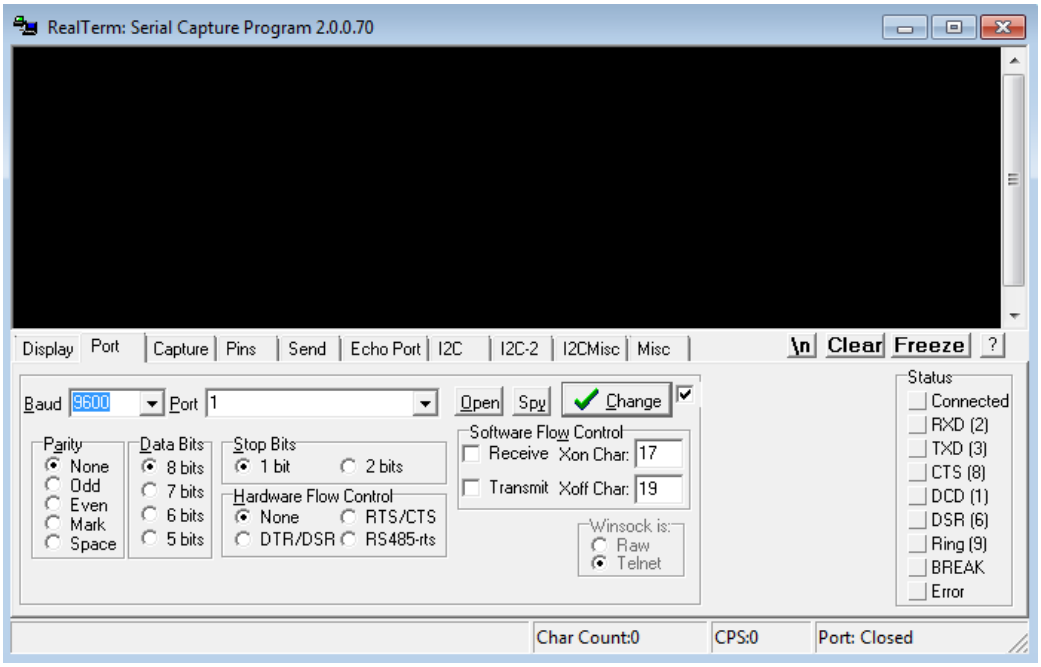
Arduino UNO sólo dispone de un puerto serie hardware aunque podemos emular más puertos serie vía software. La conexión serie es utilizada por algunos periféricos o sensores para interactuar con Arduino:

Shield GPS
(comunicación serie)

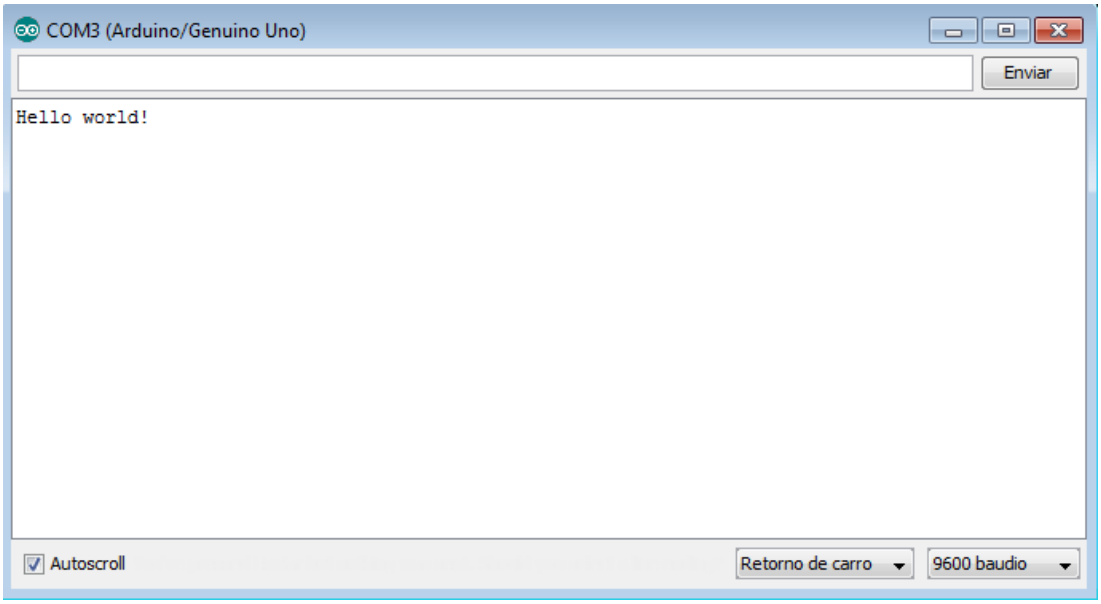
Módulo Bluetooth HC-06
(comunicación serie)

Para poder desde un ordenador visualizar los datos recibidos vía puerto serie debemos utilizar una aplicación de tipo “terminal” o “consola” serie:

Realterm - consola serie para Windows:



Arduino IDE - serial monitor:



ArduinoBlocks – consola serie
(Se necesita instalar la aplicación ArduinoBlocks-Connector)

A la hora de establecer una conexión serie los dos extremos que intervienen en la conexión (en este caso Arduino y el PC) deben establecer el mismo valor en la velocidad de la conexión.

Velocidad en baudios por defecto: 9600 bits por segundo

Otras velocidades utilizadas: 4800, 19200, 38400, 57600, 115200, …

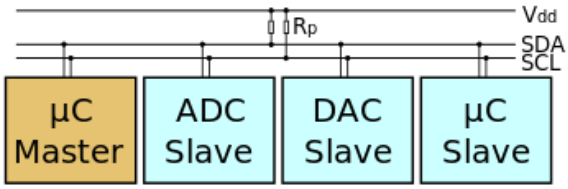
2.9.2 Comunicación I2C / TWI

El bus I2C (I²C o TWI) es un bus de datos seire desarrollado por Philips.

Se utiliza principalmente internamente para la comunicación entre diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos integrados.

Atmel introdujo por motivos de licencia la designación TWI (Two-Wired-Interface) actualmente utilizada por algunos otros fabricantes. Desde el punto de vista técnico, TWI e I2C son idénticos.

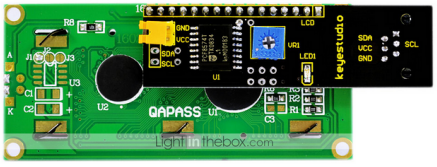
El I2C está diseñado como un bus maestro-esclavo. La transferencia de datos es siempre inicializada por un maestro; el esclavo reacciona. Es posible tener varios maestros (Multimaster-Mode). En el modo multimaestro pueden comunicar dos maestros entre ellos mismos, de modo que uno de ellos trabaja como esclavo. El arbitraje (control de acceso en el bus) se rige por las especificaciones, de este modo los maestros pueden ir turnándose.

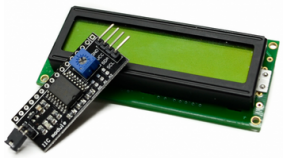


La dirección de I2C estándar es el primer byte enviado por el maestro, aunque los primeros 7 bits representan la dirección y el octavo bit (R/W-Bit) es el que comunica al esclavo si debe recibir datos del maestro (low/bajo) o enviar datos al maestro (high/alto). Por lo tanto, I2C utiliza un espacio de direccionamiento de 7 bits, lo cual permite hasta 112 nodos en un bus (16 de las 128 direcciones posibles están reservadas para fines especiales).

Cada uno de los circuitos integrados con capacidad de soportar un I2C tiene una dirección predeterminada por el fabricante, de la cual los últimos tres bits (subdirección) pueden ser fijados por tres pines de control. En este caso, pueden funcionar en un I2C hasta 8 circuitos integrados. Si no es así, los circuitos integrales (que precisan ser idénticos) deben ser controlados por varios buses I2C separados.

Pantalla LCD con módulo de conexión I2C

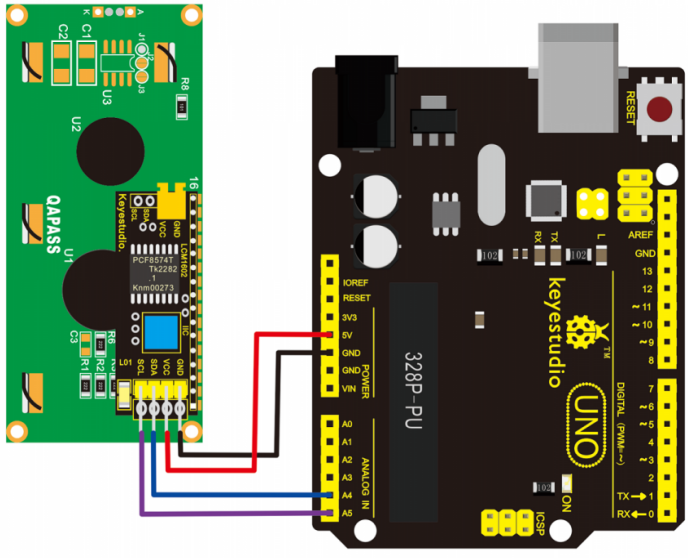




La conexión I2C en Arduino UNO se realiza en los pines:

SDA: Pin A4 / **SCL:** Pin A5

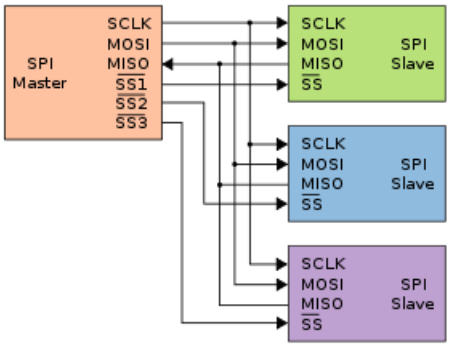
Ejemplo: conexión de módulo I2C para control de pantalla LCD:



2.9.3 Comunicación SPI

El Bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica).

Incluye una línea de reloj, dato entrante, dato saliente y un pin de Chip Select, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse. De esta forma, este estándar permite multiplexar las líneas de reloj.

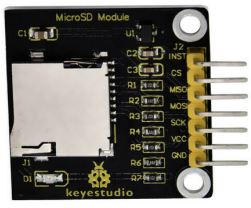


La sincronización y la transmisión de datos se realiza por medio de 4 señales:

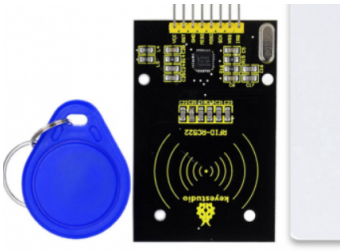
- **SCLK (Clock)**: Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit. También llamado TAKT (en Alemán).
- **MOSI (Master Output Slave Input):** Salida de datos del Master y entrada de datos al Slave. También llamada SIMO.
- **MISO (Master Input Slave Output):** Salida de datos del Slave y entrada al Master. También conocida por SOMI.
- **SS/CS/Select:** Para seleccionar un Slave, o para que el Master le diga al Slave que se active. También llamada SSTE.

Algunos periféricos SPI:

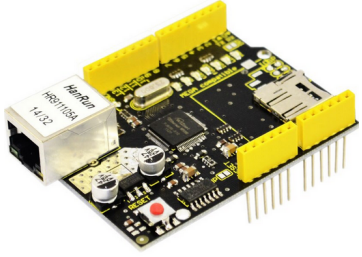
Tarjeta micro SD



Lector RFID



Ethernet shield



La conexión SPI en Arduino UNO se realiza en los pines:

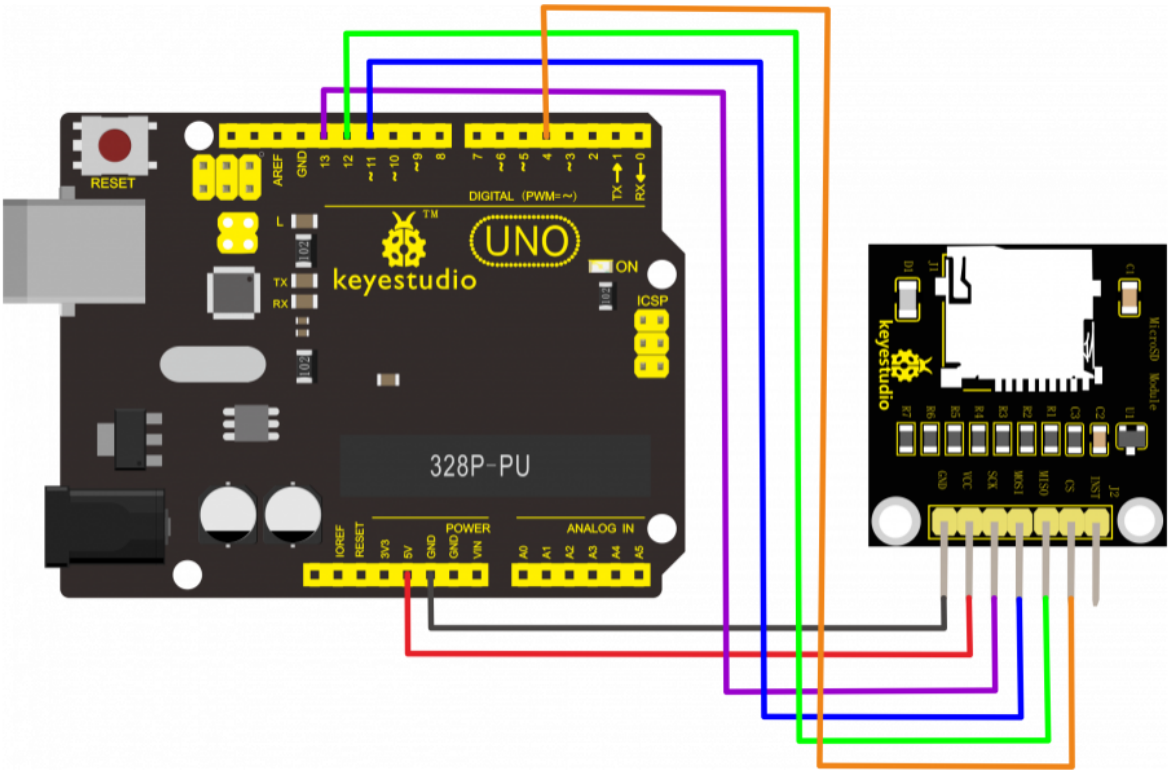
MOSI: Pin 11

MISO: Pin 12

SCLK: Pin 13

SS/CS: Depende de la programación, puede usarse cualquier pin.

Ejemplo: conexión de módulo para tarjetas SD (el pin SS está conectado al pin 4):



2.9.4 Comunicación Ethernet

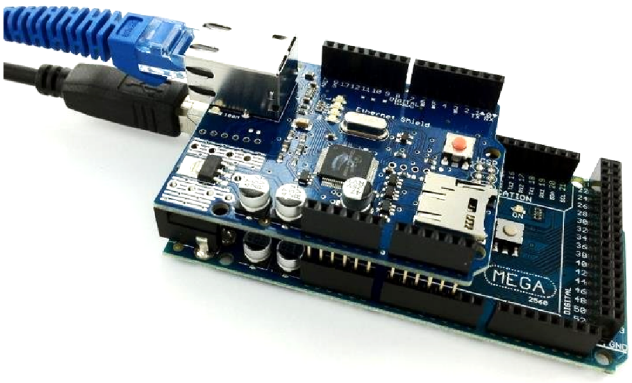
La conexión Ethernet nos permite la conexión a la red (normalmente para acceder a internet) a través de un cable Ethernet, conectado a nuestro routero o switch. Esta conexión se caracteriza por usar el conector RJ45 y es una forma sencilla de conectar nuestro Arduino a la red.

Para añadir la conectividad lo más sencillo es usar una Shield Ethernet w5100.

http://shop.innovadidactic.com/index.php?id_product=679&controller=product

Arduino UNO + Shield Ethernet

Arduino Mega + Shield Ethernet



Para más información consultar el uso de los bloques MQTT para intercambio de información a través de conexiones IP con conexión Ethernet o WiFi.

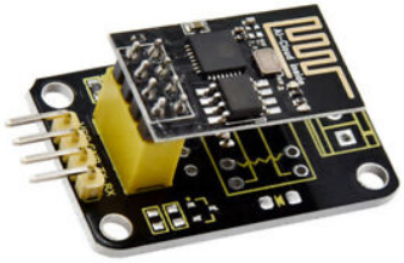
2.9.5 Comunicación WiFi

Otra forma de conectar a la red por la que podemos optar con Arduino es mediante una conexión inalámbrica WiFi. Sin embargo las shields WiFi a parte de tener un precio bastante alto (comparado con el precio habitual de los componentes Arduino) la mayoría han sido descatalogadas en favor de placas con WiFi integrado.

Una opción sencilla y económica para la conexión de cualquier modelo Arduino a internet a través de WiFi es usar un ESP-01 (placa basada en el chip ESP8266) como periférico WiFi a través de una conexión serie, es decir, usando dos pines (RX,TX).

El periférico ESP-01 necesita un sencillo adaptador para ajustar los niveles de voltaje en la comunicación:

Módulo ESP-01 con adaptador:



http://shop.innovadidactic.com/index.php?id_product=866&controller=product

El módulo ESP-01 comunica con protocolo serie y por defecto trabaja a una velocidad de 115200 bps, por lo que sólo funcionaría correctamente conectándolo a un puerto serie hardware de

Arduino (en otros pines se usaría una comunicación serie emulada por software a un máximo de velocidad de 57600 bps).

Por tanto a velocidad de 115200 bps , en Arduino UNO solo podemos usar el periférico ESP-01 en los pines:

Arduino - Pin 0 (RX) ->	*ESP-01 - Pin TX*
Arduino - Pin 1 (TX) ->	*ESP-01 - Pin RX*

IMPORTANTE: a la hora de programar la placa Arduino a través del USB debemos desconectar (al menos la alimentación) del periférico ESP-01 porque interferirá en el proceso de programación que utiliza también los pines 0,1

En Arduino Mega, disponemos de varios puertos serie hardware por lo que podemos dejar libre los pines 0,1 para no interferir en la progrmación USB, y utilizar cualquiera de los otros puertos implementados en la placa.

Puertos serie en Arduino MEGA:

	Pin RX	*Pin TX*
Serial 0 *(compartido con USB):*	0	1
Serial 1	19	20
Serial 2	17	16
Serial 3	15	14

En las placas basadas en ESP8266, como el NodeMCU o WeMos, no necesitamos añadir ningún periférico extra para la conexión pues el propio microcontrolador implementa la conexión WiFi.

De igual forma que con la conexión Ethernet, la conexión a la red por cable o WiFi nos permite utilizar los bloques MQTT para intercambiar información a través de la red o internet.

3 Software

Una vez tenemos definido el hardware necesario para un proyecto el siguiente paso es programar el microcontrolador de la placa Arduino para que realice las tares necesarias para el funcionamiento deseado.

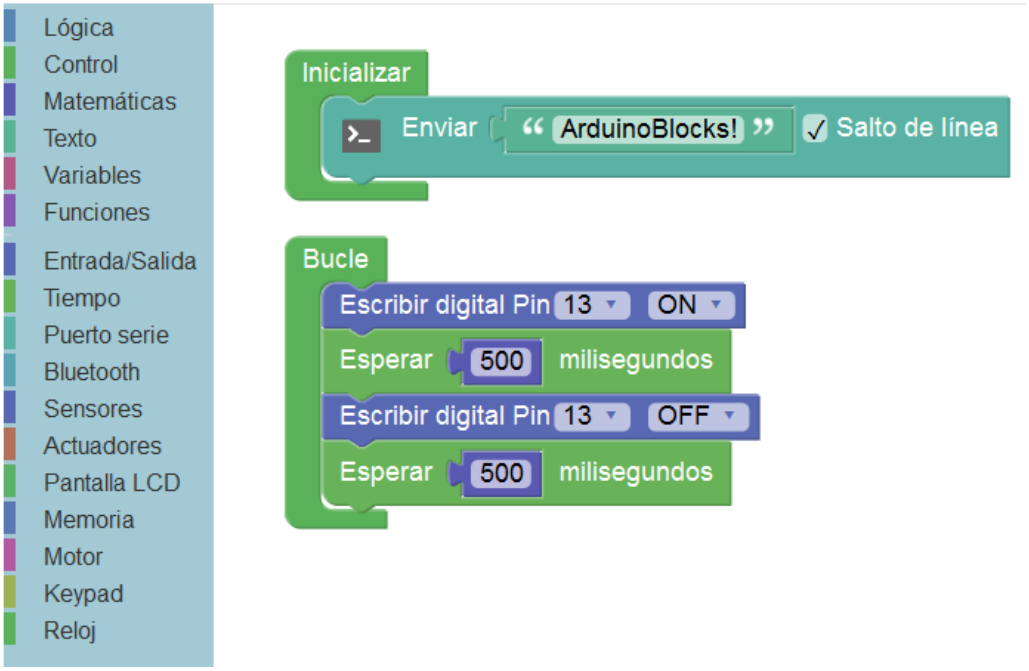
La programación de la placa Arduino se realiza normalmente en lenguaje C++ desde el entorno Arduino IDE. Para programar debemos conocer primero este lenguaje, lo cual supone mucho tiempo del que muchas veces no disponemos.

En los últimos años han aparecido entornos mucho más sencillos e intuitivos para desarrollar aplicaciones que nos permiten introducirnos de forma práctica y sencilla en el mundo de la programación. Es el caso de Scratch, un entorno de desarrollo de videojuegos multiplataforma, y AppInventor, un entorno de desarrollo de aplicaciones para dispositivos móviles Android.

ArduinoBlocks, al igual, es un entorno online que nos permite programar Arduino (sin necesidad de conocer el lenguaje de programación C++) de forma visual al estilo de programación de bloques.

ArduinoBlocks implementa bloques generales comunes a cualquier entorno de programación y por otro lado bloques específicos para Arduino donde podemos acceder a leer/escribir datos de los pines de entrada/salida, acceder a información de sensores conectados, manejar actuadores, periféricos como la pantalla LCD y muchas funcionalidades más.

Programa de ejemplo generado automáticamente en modo “prueba”



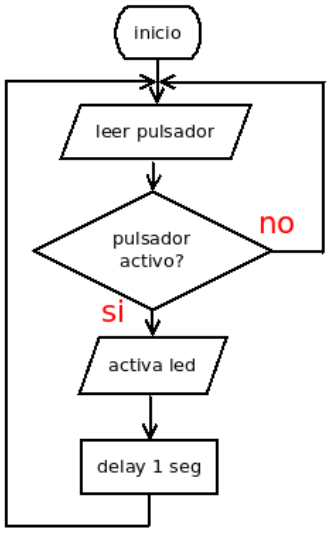
3.1 Algoritmos

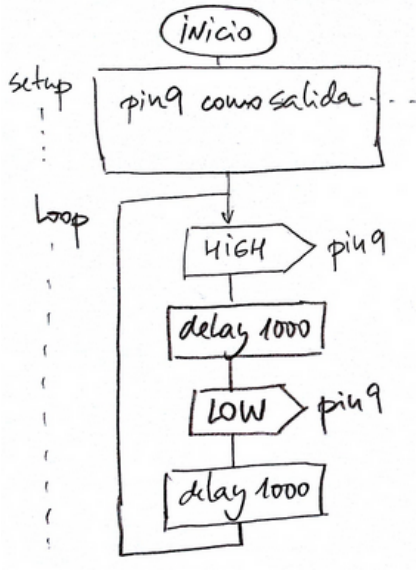
Un algoritmo es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.

A la hora de programar en cualquier lenguaje de programación lo primero que tenemos que hacer es plantear el algoritmo que queremos desarrollar y posteriormente implementarlo en el lenguaje de programación elegido.

A pesar de que la programación por bloques es muy intuitiva y visual, siempre es recomendable plantear el algoritmo antes de empezar un proyecto.

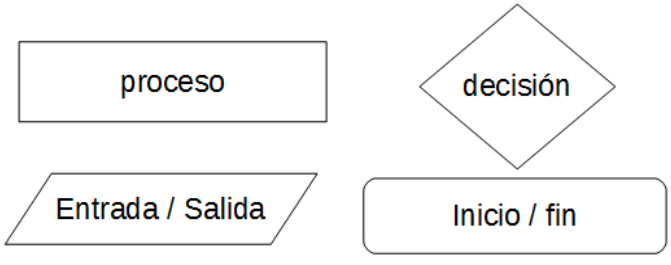
Ejemplos de diagramas de flujo para definir un algoritmo:





La definición previa del algoritmo nos permitirá agilizar el proceso de creación del programa.

Simbología básica para la definición gráfica de un algoritmo:



3.2 Bloques de uso general

Los bloques de uso general nos permiten implementar funciones comunes en cualquier entorno o sistema programable. Esto incluye funciones lógicas, matemáticas, condiciones, bucles, funciones de texto, etc.

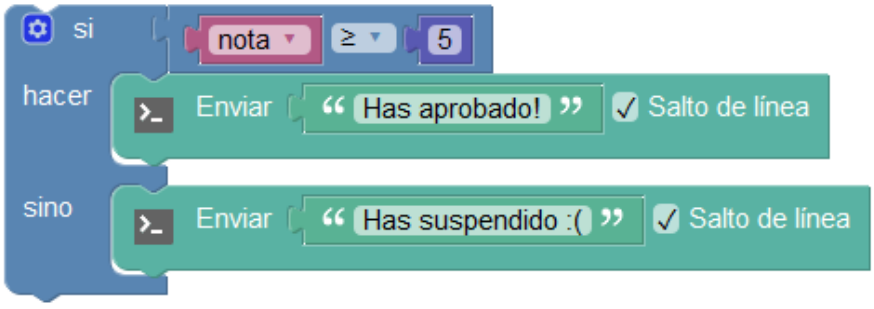
3.2.1 Lógica

Con estos bloques tenemos acceso a las funciones lógicas necesarias para implementar en nuestro programa de Arduino.

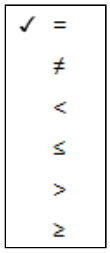
Las funciones lógicas trabajan con valores o expresiones de “verdadero” o “falso”

- **Condición / decisión**: Evalúa una condición lógica, si se cumple realiza el bloque “hacer” si no se cumple realiza el bloque “sino” (opcional)

Ejemplo de comparaciones:

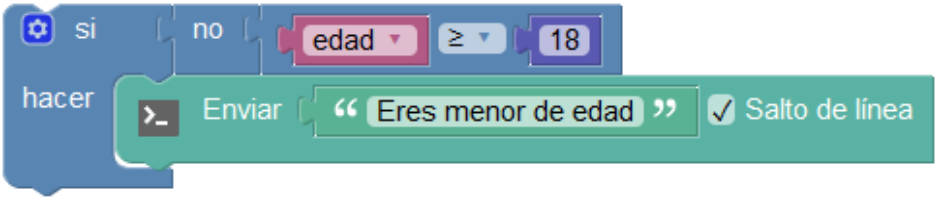


- **Evaluar condición (númerica):** Devuelve verdadero o falso según si la condición indicada se cumple entre los dos operandos (numéricos).



=	Igual
≠	Distintos
<	Menor que
≤	Menor o igual que
>	Mayor que
≥	Mayor o igual que

Ejemplo: Comparción numérica “mayor o igual que”



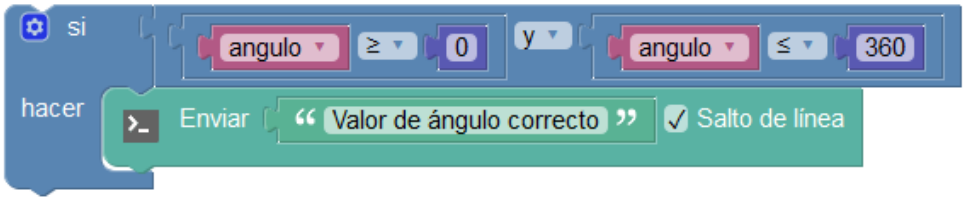
- **Evaluar condición (booleana):** Devuelve verdadero o falso según si la condición indicada se cumple entre los dos operandos booleanos (de verdadero o falso).

=	Igual
≠	Distintos

- **Conjunción/Disyunción (AND):** Evalúa dos expresiones lógicas y devuelve verdadero o falso según la función lógica seleccionada.

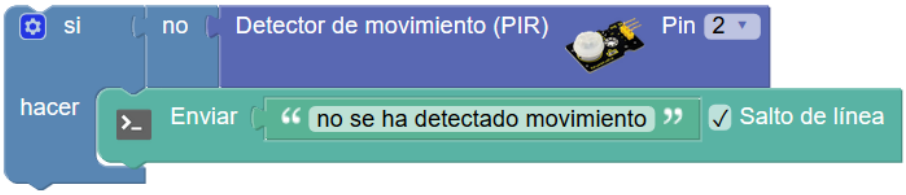
y (AND)	Se cumple si las dos operandos son verdaderos
o (OR)	Se cumple si alguno de los dos operandos es verdadero.

Ejemplo: condición lógica “y” (AND)



- **Negación:** Permite negar (invertir) un valor lógico de verdadero o falso.

Ejemplo: negación de un detector para saber cuando “no” hay movimiento



- **Constantes lógicas:** son valores booleanos indicando uno de los dos estados posibles

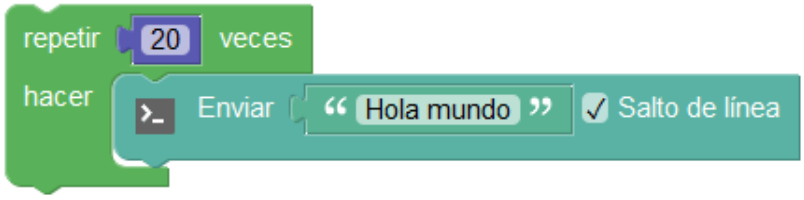
On	= Verdadero
Off	= Falso

3.2.2 Control

Las estructuras de control nos permiten realizar bucles e iteraciones.

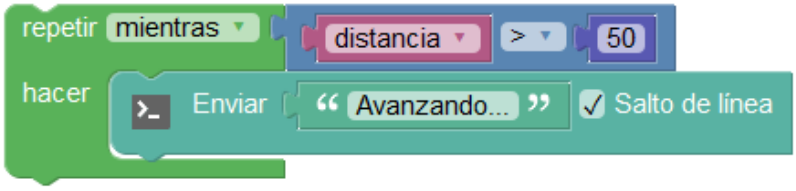
- **Repetir:** Repite (n) veces los bloques de su interior.

Ejemplo:



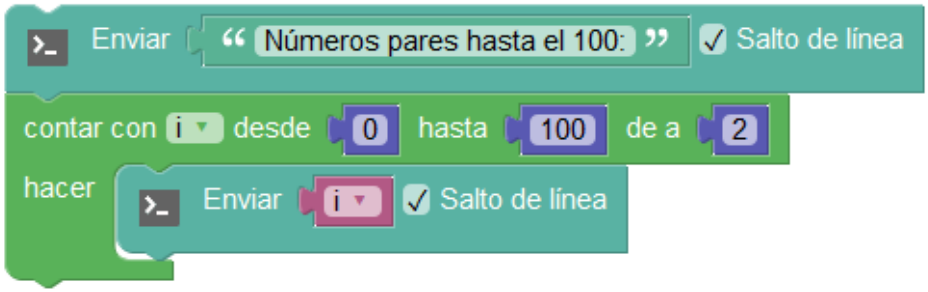
- **Repetir según condición:** Repite mientras o hasta que se cumpla una condición.

Ejemplo:



- **Contar:** Realiza un bucle contando con un variable índice. Se define un valor de inicio, una valor de fin y los incrementos que se realizarán en cada iteración del bucle. Dentro del bucle podremos usar esta variable.

Ejemplo:



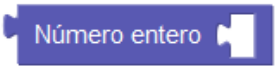
3.2.3 Matemáticas

- **Constante numérica**: Permite especificar un valor numérico entero o decimal

Ejemplo:

- **Número entero / sin signo:** Trata el valor como un entero. Si especificamos sin signo, trata el valor como una variable sin signo internamente.

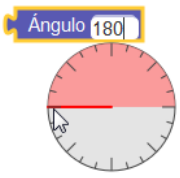
Para las variables ArduinoBlocks utiliza el tipo de dato “double” cuando traduce el programa a lenguaje C++. En caso de hacer la conversión se trata como un “cast” a un tipo de datos “long” o “unsigned long”

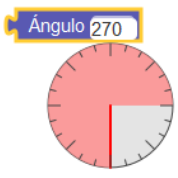


Trata el valor como tipo entero

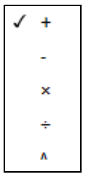
Trata el valor como tipo entero sin signo

- **Ángulo:** Permite definir un valor de ángulo en grados. Es un valor numérico tal cual, pero con la ventaja que permite definir el valor de una forma visual viendo el ángulo gráficamente.





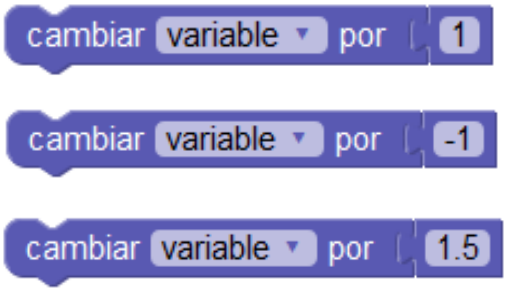
- **Operaciones básicas:**



+	Suma
-	Resta
x	Multiplicación
÷	División
^	Potencia

Ejemplo:

- **Cambiar variable por:** Aumenta o disminuye el valor de una variable por el valor indicado (si es un valor positivo aumenta si es negativo disminuye)

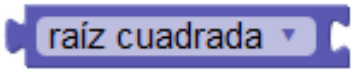


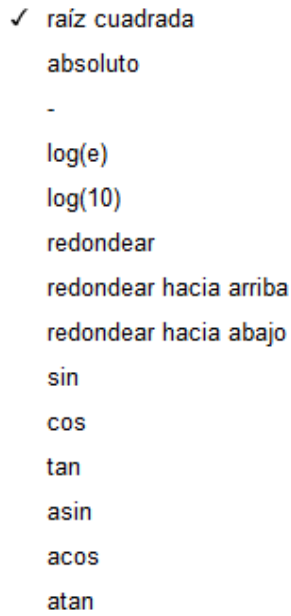
Aumenta la variable en +1
variable = variable + 1

Disminuye la variable en -1
variable = variable – 1

Aumenta la variable en +1.5
variable = variable + 1.5

- **Funciones matemáticas:**





- **Atan2:** Calcula la arco-tangente de y/x, siendo y el primer parámetro y x el segundo.

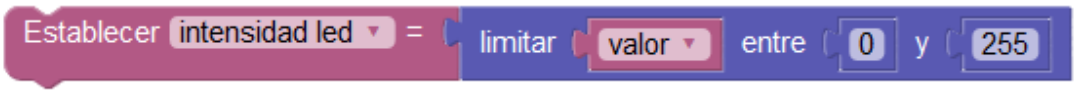
- **Mapear:** Permite modificar el rango de un valor o variable desde un rango origen a un rango destino. Esta función es especialmente útil para adaptar los valores leídos de sensores o para adaptar valores a aplicar en un actuador.

Ejemplo:
-Sensor de temperatura: 10ºC … 50ºC
-Arduino lectura analógica: 0 … 1023

Necesitamos convertir del rango 0-1023 leído al rango 10ºC-50ºC:

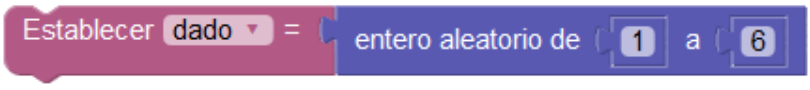
- **Limitar:** Permite acotar el valor mínimo y máximo.

Ejemplo:



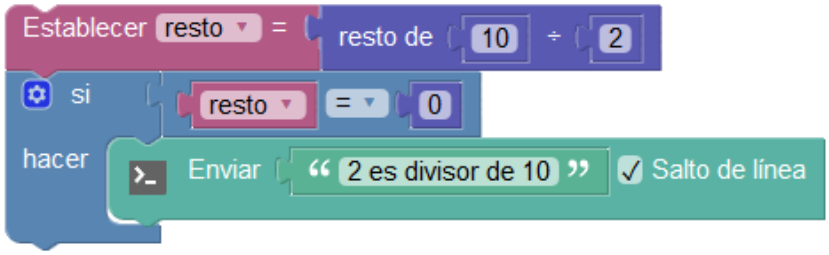
- **Número aleatorio:** Genera un valor aleatorio entre los valores especificados.

Ejemplo:



- **Resto:** Obtiene el resto de la división de los dos operandos.

Ejemplo:



Ejemplo: uso de varios bloques de operaciones matemáticas:

3.2.4 Texto

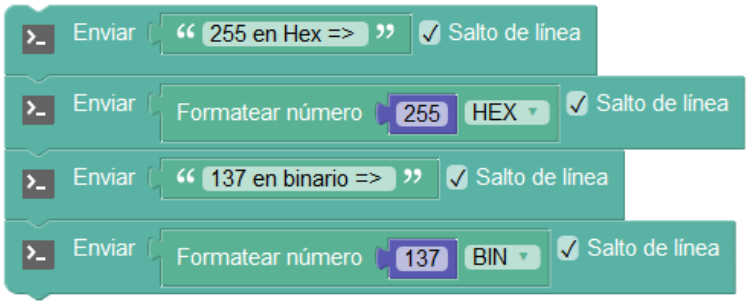
Las funciones de texto son especialmente útiles con la utilización en el puerto serie (consola), y otros periféricos como pantallas LCD. Permiten trabajar con variables de tipo texto o con textos prefijados.

- **Constante de texto:** Define un texto de forma estática.

- **Formatear número:** Obtiene en forma de texto el valor de una variable o constante numérica en el formato especificado.

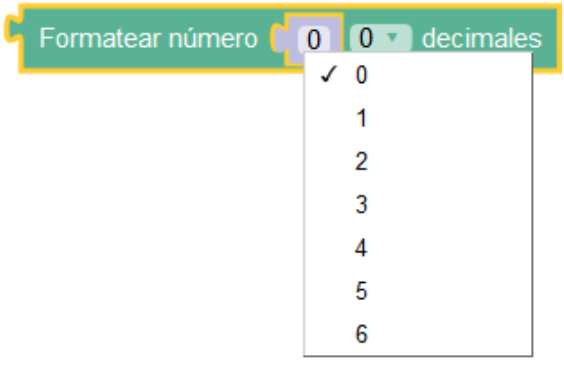
HEX	Genera el texto con la representación hexadecimal del valor.
DEC	Genera el texto con la representación decimal del valor.
BIN	Genera el texto con la representación binaria del valor.

Ejemplo:



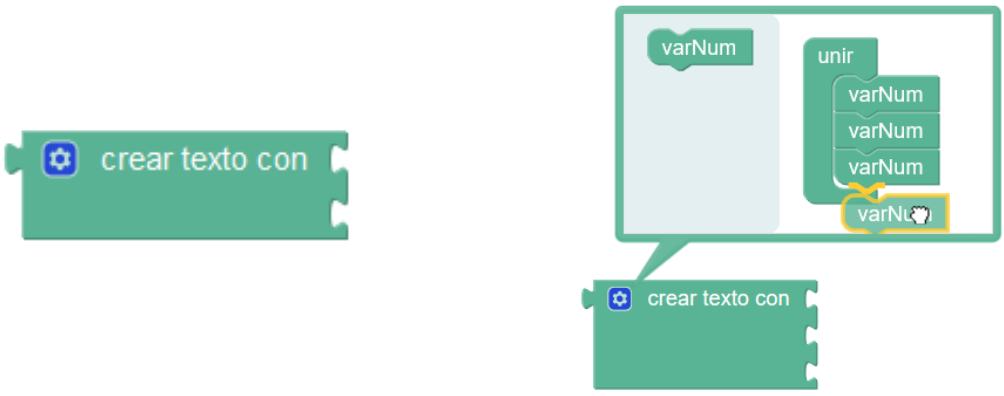

```
connected at 9600
255 en hexadecimal => ff
137 en binario=> 10001001
```

- **Formatear número con decimales:** Realiza la conversión de una variable o constante numérica a texto igual que el bloque anterior pero pudiendo indicar el número de decimales a mostrar.

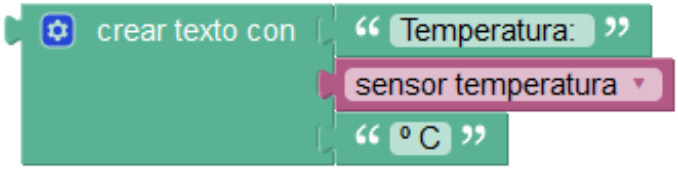


Ejemplo: enviar por la consola el valor de una variable como texto formateado con 6 y 1 decimales respectivamente:

- **Crear texto con:** Crea un texto a partir de la unión de otros textos o variables. Las variables especificadas se convertirán a texto con formato decimal.

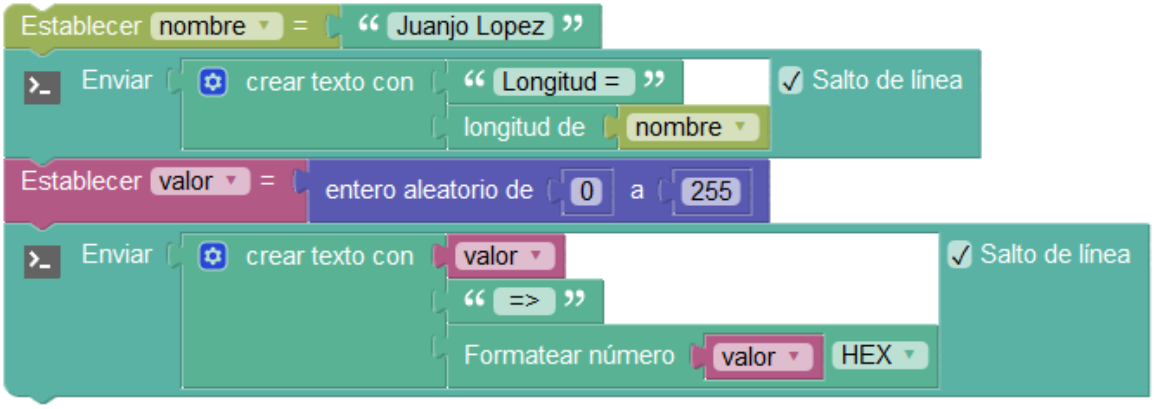


Ejemplo:

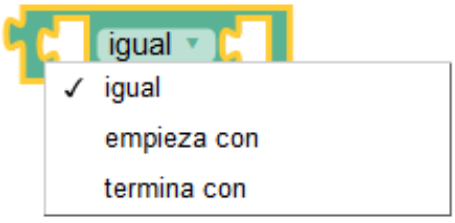


- **Longitud** : Obtiene el número de caracteres del texto.

Ejemplo de uso de bloques de texto:

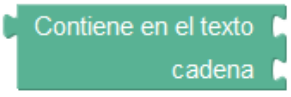


- **Comparación de textos:** Permite comparar dos cadenas de texto. El resultado es un valor lógico de verdadero o falso.

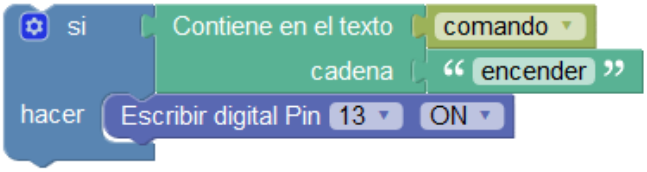


Ejemplo: comparación de texto y variables de tipo texto

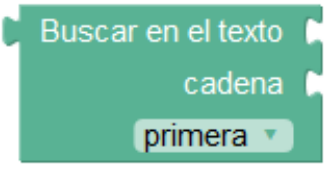
- **Contiene el texto:** Comprueba si existe un texto dentro del texto indicado. Devuelve verdadero si existe y falso en caso contrario.



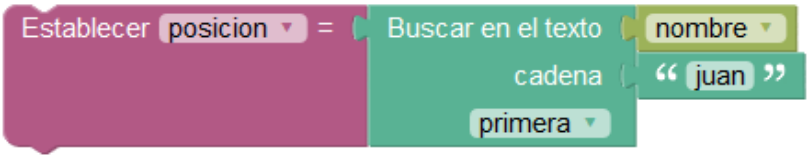
Ejemplo:



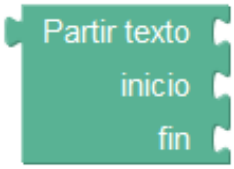
- **Buscar en el texto:** Busca la posición de un texto dentro de otro texto. Si el texto buscado no se encuentra devuelve el valor 0, en otro caso devuelve la posición donde empieza el texto.



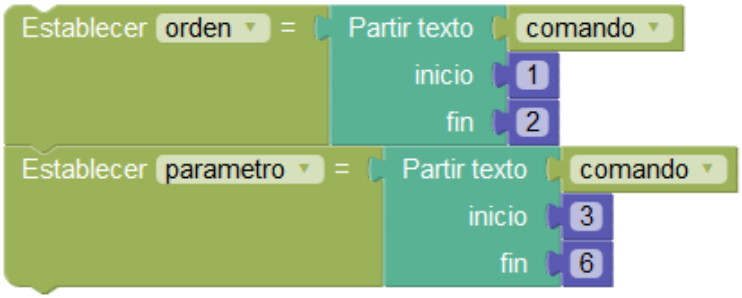
Ejemplo:



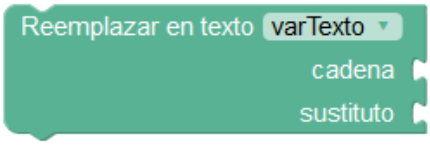
- **Partir texto:** Obtiene una parte del texto, indicando la posición de inicio y fin dentro del texto para crear la subcadena.



Ejemplo:



- **Reemplazar en texto:** Reemplaza todas las ocurrencias del texto indicado por el nuevo dentro de la variable de texto seleccionada.



Ejemplo:

- **Valor ASCII:** Obtiene un valor numérico correspondiente al valor de la tabla ASCII del caracter correspondiente. Este bloque es útil para detectar pulsaciones de teclas que son enviadas por comunicación serie, bluetooth, etc. como bytes independientes.

Tabla ASCII de los caracteres imprimibles:

Caracteres ASCII imprimibles					
32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

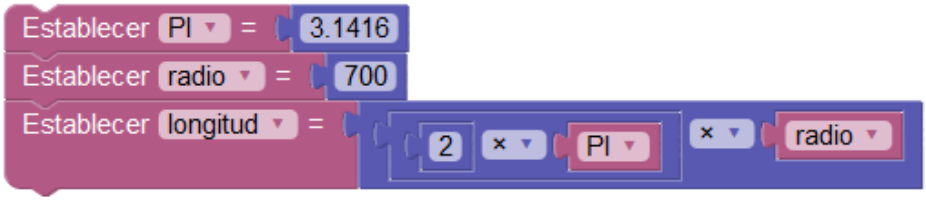
3.2.5 Variables

Una variable es un hueco en la memoria donde el programa puede almacenar valores numéricos. El sistema nos permiten asignarles un nombre simbólico como por ejemplo “temperatura exterior”, “velocidad”, “posición servo 1”,”estado”,… para facilitar su uso.

Hay tres tipos de variables en ArduinoBlocks: numéricas, booleanas y de texto.

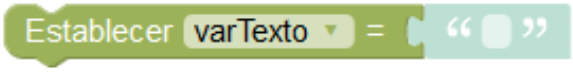
- **Variables numéricas:** permite valores numéricos enteros o con decimales, internamente se representan con el tipo de datos “doublé” a la hora de generar el código para Arduino. Este tipo utiliza 4 bytes y permite almacenar valores en el rango: -3.4028235E+38 a 3.4028235E+38

Ejemplo:

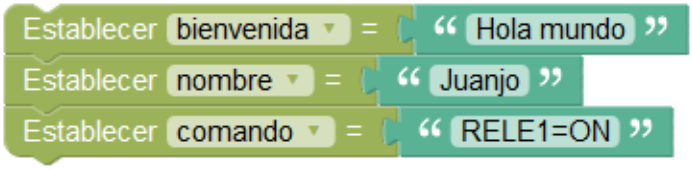


Ejemplo:

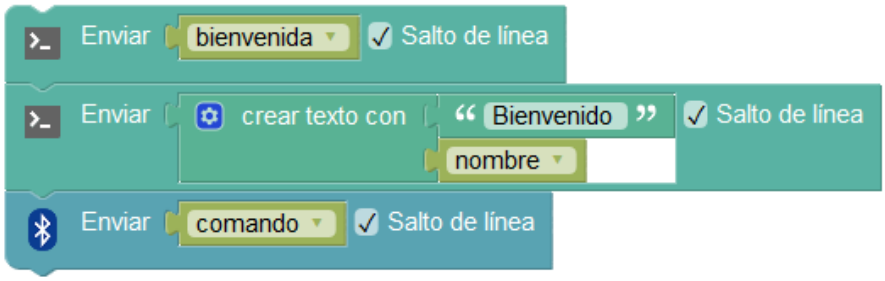
- **Variables de texto:** permite almacenar valores de texto. Internamente utiliza el tipo de dato “String” a la hora de generar el código para Arduino.



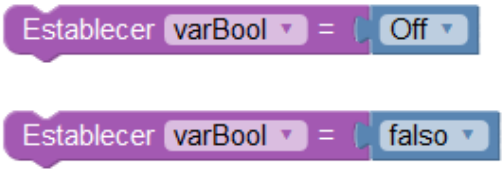
Ejemplo:



Ejemplo:



- **Variables booleanas:** permite almacenar valores lógicos booleanos de dos estados (verdadero/falso, ON/OFF, HIGH/LOW, …)



Ejemplo:

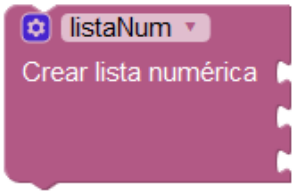
Ejemplo:

3.2.6 Listas

Las listas de datos nos permiten almacenar un listado de valores y acceder a ellos por su posición en la lista. Las listas pueden ser de tipo numéricas o de texto.

- **Listas numéricas:**

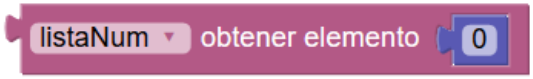
  Podemos crear una lista asignándole un nombre a la lista y asignándole valores iniciales.



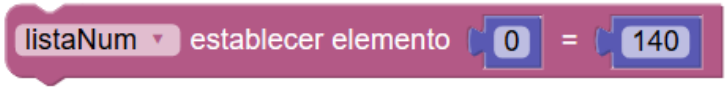
Ejemplo:

Para saber el número de elementos que tenemos en una lista podemos usar el bloque:

En una lista podemos obtener el valor de una posición (desde la 0 hasta el número de elementos -1 en la lista) con el bloque, por ejemplo si tenemos 10 elementos podremos acceder a ellos desde el 0 al 9:

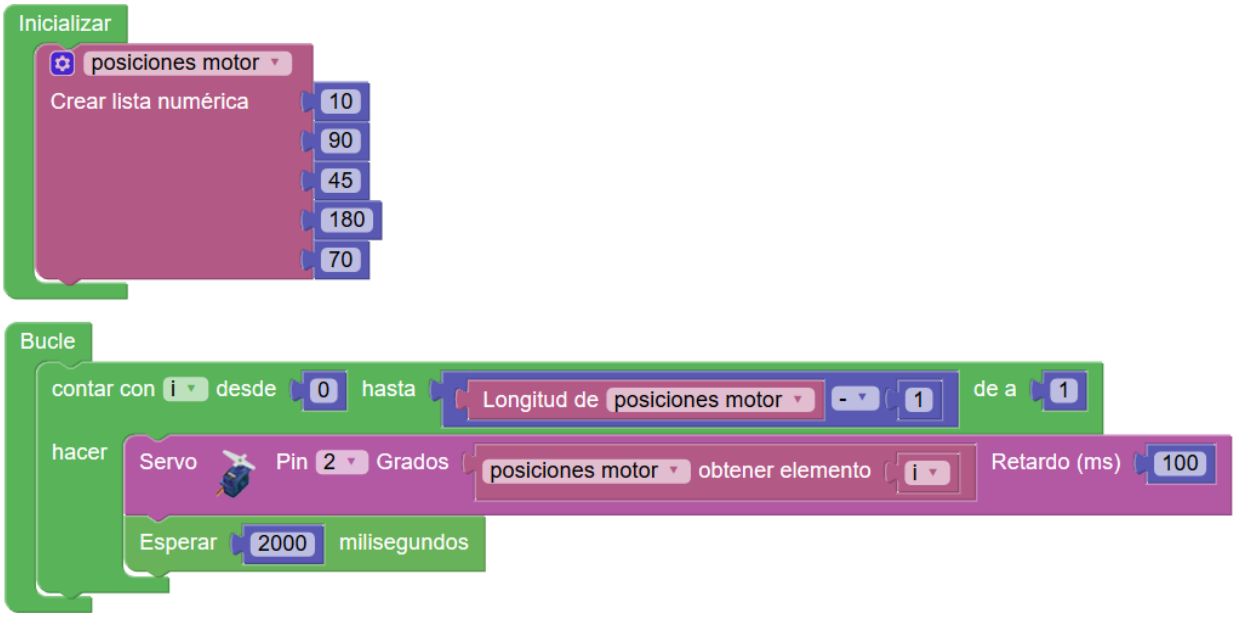


O cambiar el valor de un elemento indicando su posición y el nuevo valor:



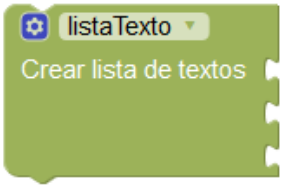
Ejemplo:

Posicionamiento de un motor (servo) en distintas posiciones (grados) prefijados en una lista.

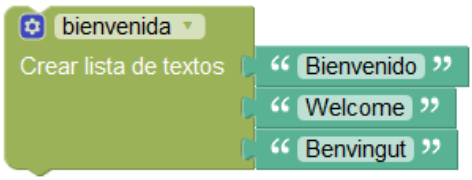


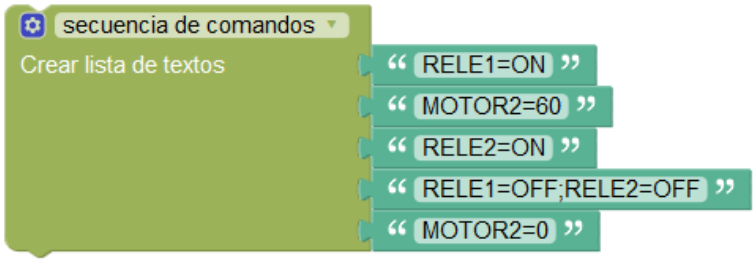
- **Listas de textos:**

  Podemos crear una lista asignándole un nombre a la lista y asignándole valores iniciales.



  Ejemplo:



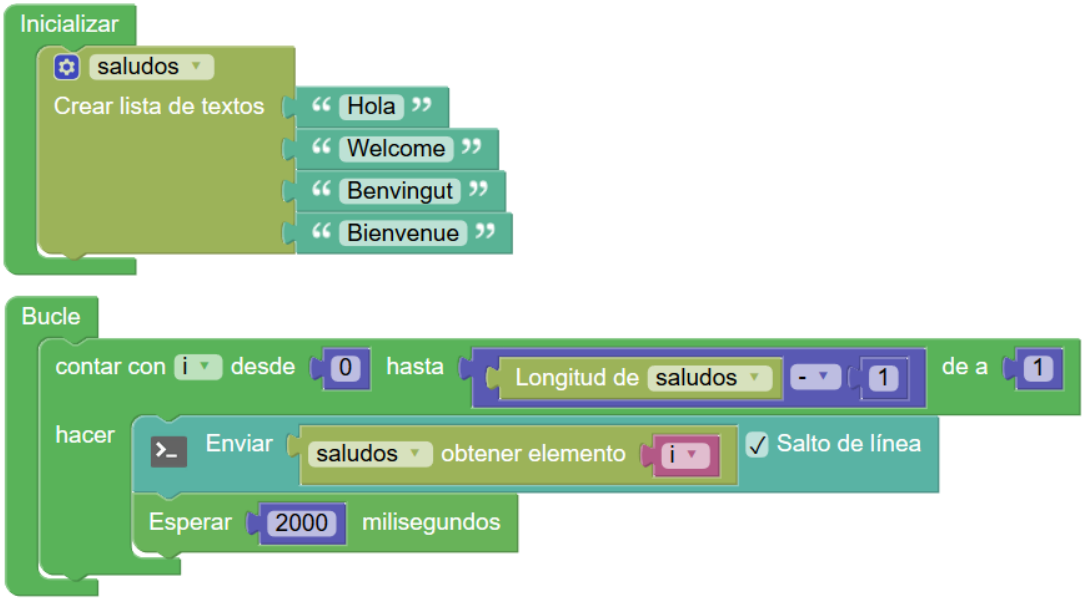


  Para saber el número de elementos que tenemos en una lista podemos usar el bloque:

En una lista podemos obtener el valor de una posición (desde la 1 hasta el número de elementos en la lista) con el bloque:

O cambiar el valor de un elemento indicando su posición y el nuevo valor:

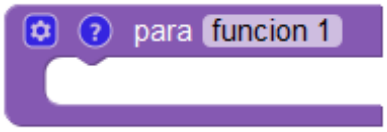
Ejemplo: Mostrar saludos en distintos idiomas a partir de una lista



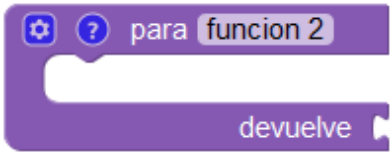
3.2.7 Funciones

Las funciones permiten agrupar bloques de código. Esto es útil cuando un bloque de código se repite en varias partes del programa y así evitamos escribirlo varias veces o cuando queremos dividir el código de nuestro programa en bloques funcionales para realizar un programa más entendible.

- **Definición de una función:** La definición consiste en crear el grupo donde podremos insertar el código de bloques que forma la función. Debemos darle un nombre representativo que utilizaremos para llamar a esa función y ejecutarla.

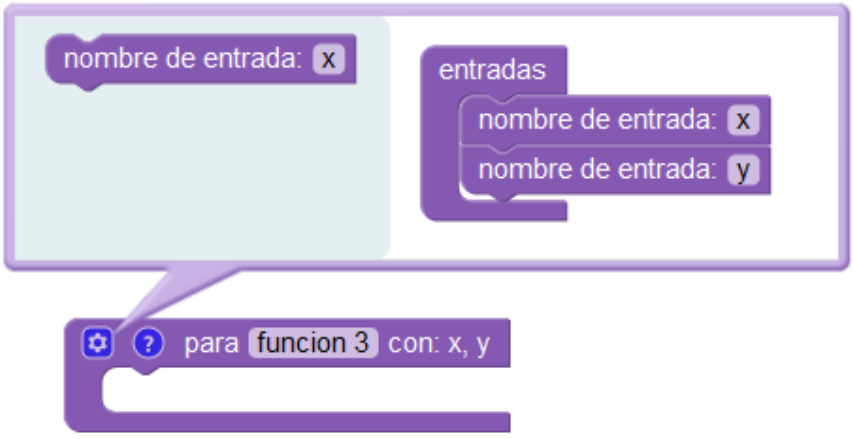


Función sin valor de retorno.
La función ejecuta los bloques de su interior y vuelve al punto de llamada.

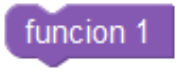


Función con valor de retorno.
La función ejecuta los bloques de su interior y devuelve un resultado.

- **Parámetros:** A las funciones se les pueden añadir parámetros para especificar en la llamada.

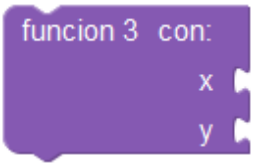


- **Llamada a una función:** Permite llamar a la ejecución de la función, se ejecutarán los bloques internos de la función y al terminar se seguirá la ejecución por donde se había realizado la llamada a la función.



Llamada a una función sin valor de retorno.

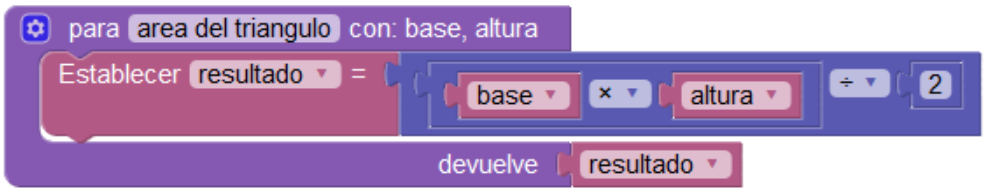
Llamada a una función con valor de retorno.



Llamada a una función sin valor de retorno
y con 2 parámetros

Ejemplo: Función para calcular el área de un triángulo

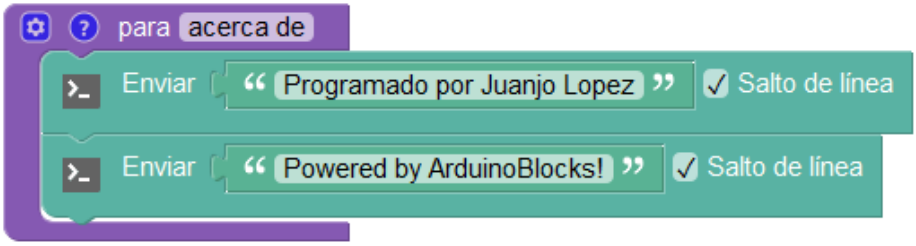
Definición:



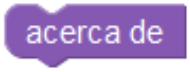
Llamada:

Ejemplo: Función para enviar información por la consola

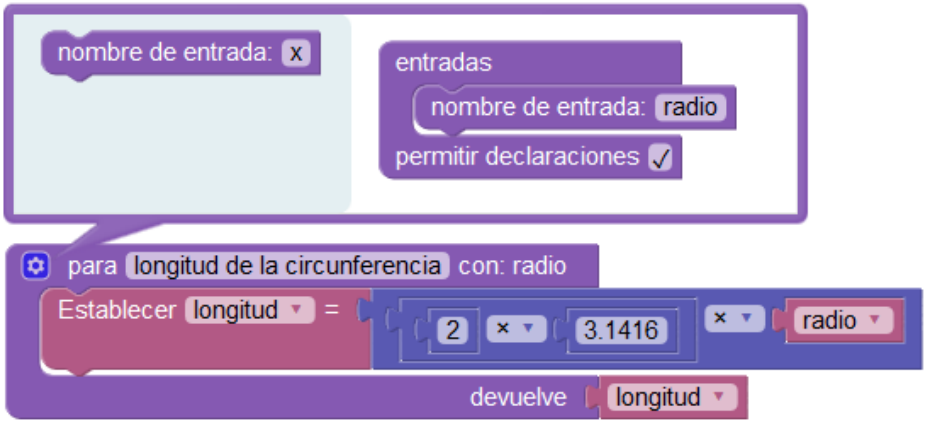
Definición:



Llamada:

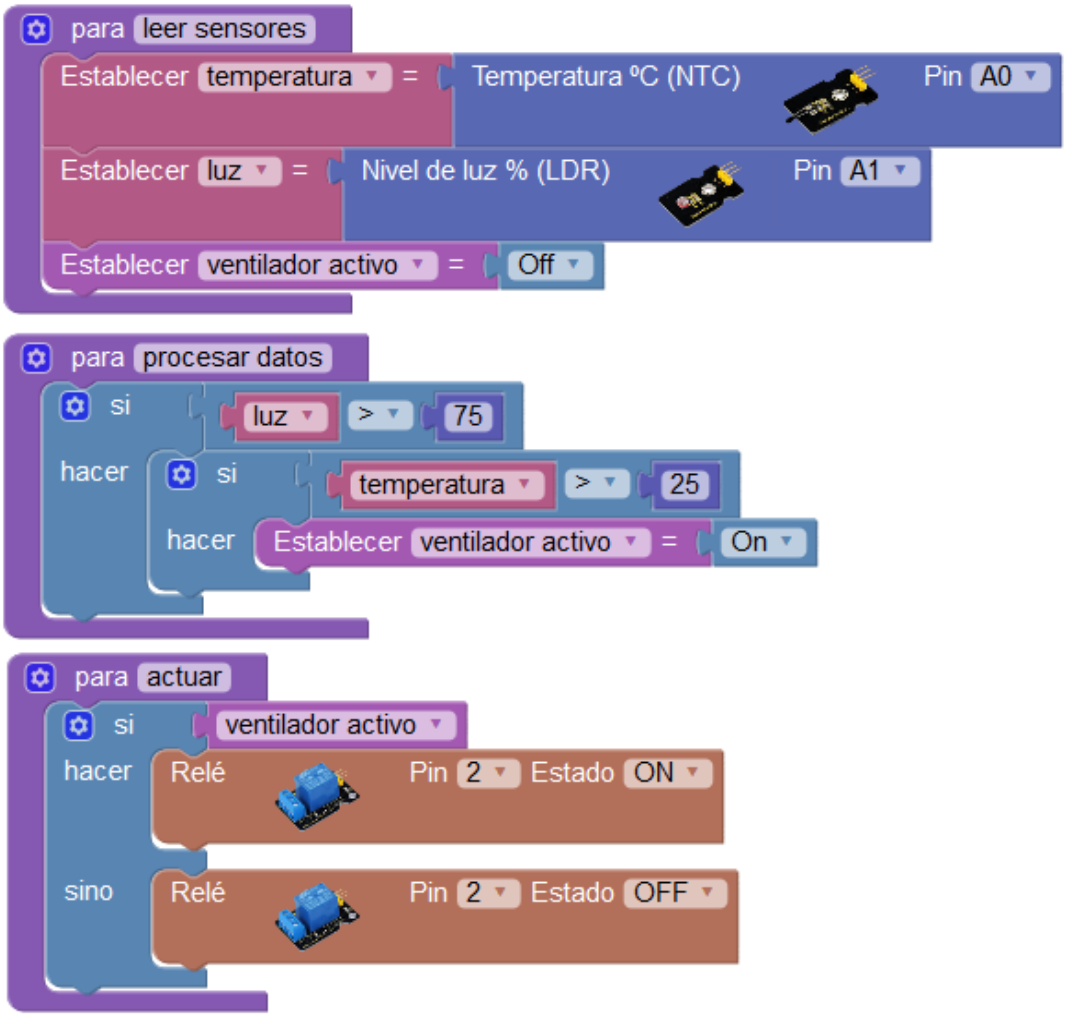


Ejemplo: Función para calcular la longitud de una circunferencia

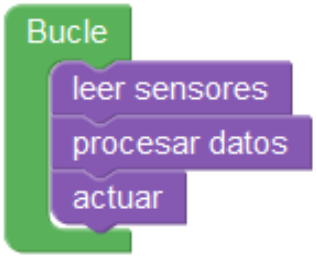


Ejemplo: División en partes funcionales de un programa real.

Definición:



Llamada desde el bucle principal del programa:



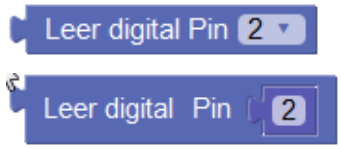
3.3 Bloques Arduino

En el siguiente apartado veremos los bloques relacionados con funciones propias de la placa Arduino. Estos bloques nos permitirán acceder a funcionalidades del propio microcontrolador y otros estarán orientados a sensores, actuadores o periféricos que podemos conectar a la placa Arduino para desarrollar nuestros proyectos.

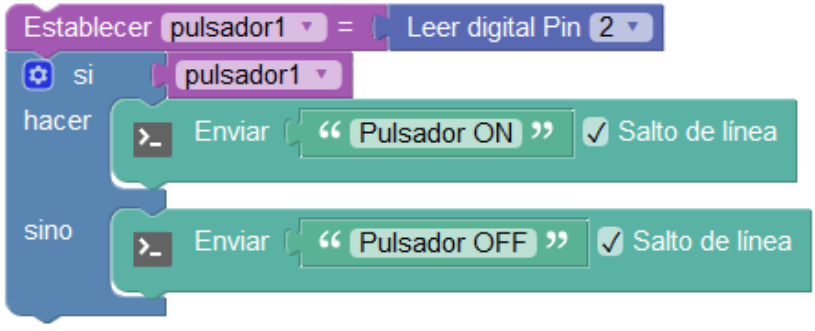
3.3.1 Entrada/Salida

Las funciones de entrada/salida genéricas nos permiten leer o escribir en los pines digitales y analógicos de la placa Arduino descritos en el apartado 2.3.

- **Leer pin digital:** Obtiene el valor digital del pin (0/1, ON/OFF, verdadero/falso). (Recuerda para leer un ON/1 debemos aplicar 5v en la entrada digital y 0v para leer un OFF/0)



Ejemplo:



- **Escribir pin digital:** Escribe el valor en un pin digital pin (0/1, ON/OFF, verdadero/falso).

  (Si se activa, la salida suministrará 5v en caso contrario 0v)

Ejemplo:

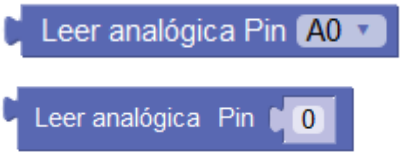
Versión equivalente:

- **Leer pin analógico:** Lee el valor de una entrada analógica.

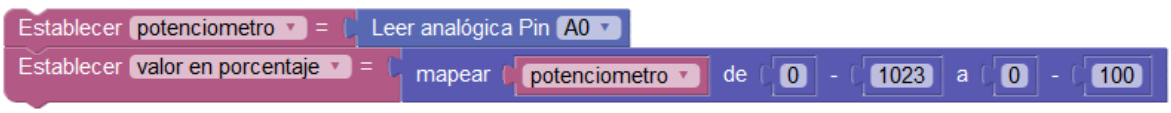
  El conversor interno DAC (Digital Analog Converter) es de 10 bits por lo que los valores leídos de una entrada analógica van de 0 a 1023

  10 bits = 2 ^ 10 = 1024 posibles valores

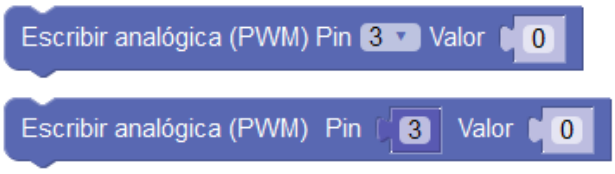
Voltaje en la entrada analógica	Valor leído
0 voltios	0
2.5 voltios	512
5 voltios	1023



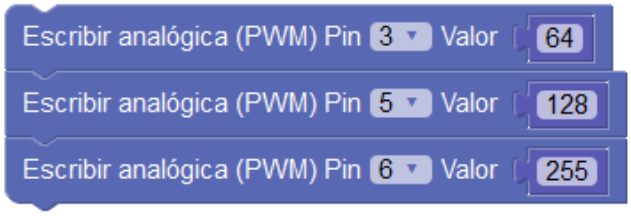
Ejemplo:



- **Escribir pin analógico:** Establece el valor del ciclo de pulsos activo/inactivo de una salida digital PWM. El valor debe estar en el rango entre 0 y 255.



Ejemplo: pin 3 al 25%, pin 5 al 50% , pin 6 al 100%

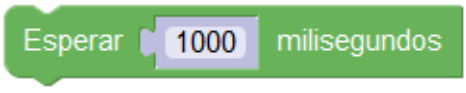


- **Leer pulso:** Lee un pulso en un pin hasta que el valor de la entrada cambie a estado alto (ON) o bajo (OFF). Mide la duración del pulso en microsegundos. Si se supera el tiempo de espera indicado sin cambiar de estado devolverá el valor 0.

3.3.2 Tiempo

Las funciones de tiempo o retardo nos permiten realizar pausas y obtener información sobre el tiempo transcurrido dentro del microcontrolador.

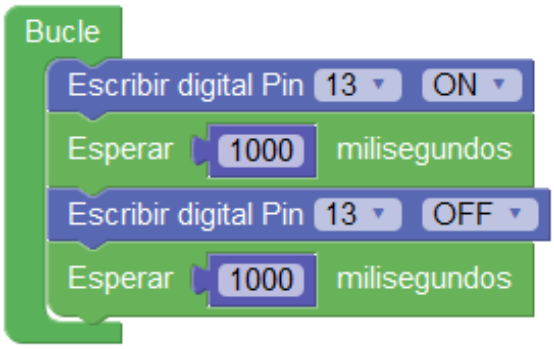
- **Esperar:** Realiza una pausa (**bloquea la ejecución del programa**) hasta seguir con la ejecución del siguiente bloque.



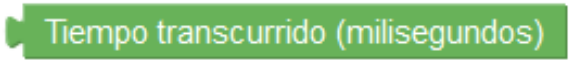
Milisegundos

Microsegundos

Ejemplo: Led 1 segundo encendido, 1 segundo apagado…



- **Tiempo transcurrido:** Obtiene un valor con el tiempo transcurrido desde el inicio o reset del microcontrolador de la placa Arduino. El valor puede ser en milisegundos o microsegundos.



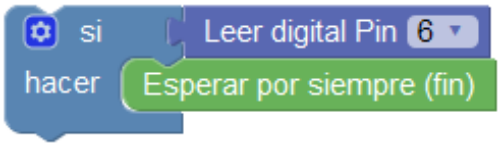
Milisegundos

Microsegundos

Ejemplo : Ejecutar la Tarea1 cada 3 segundos y la Tarea2 cada 7 segundos sin bloquear la ejecución del programa:

- **Esperar por siempre:** Bloquea indefinidamente la ejecución finalizando por tanto el programa.

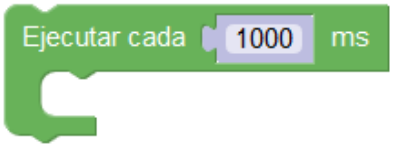
Ejemplo: al activar la entrada del pin 6 se finaliza la ejecución.



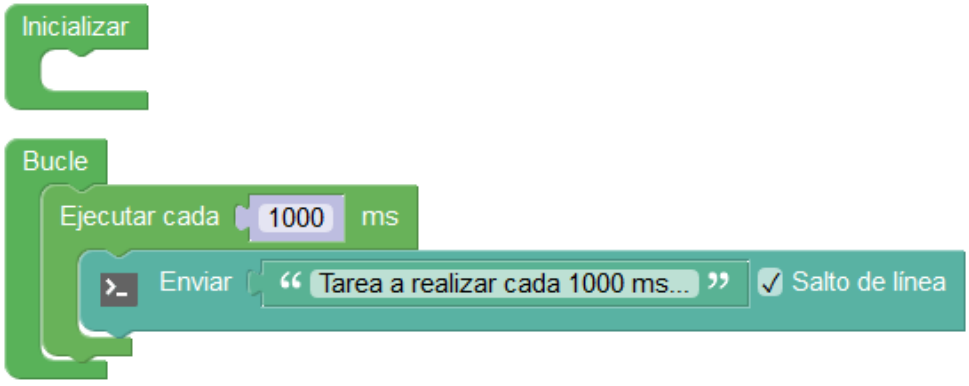
Ejemplo: Funcionamiento equivalente (esperar por siempre):

- **Ejecutar cada:** Bloque que implementa automáticamente la función de tareas explicada anteriormente.

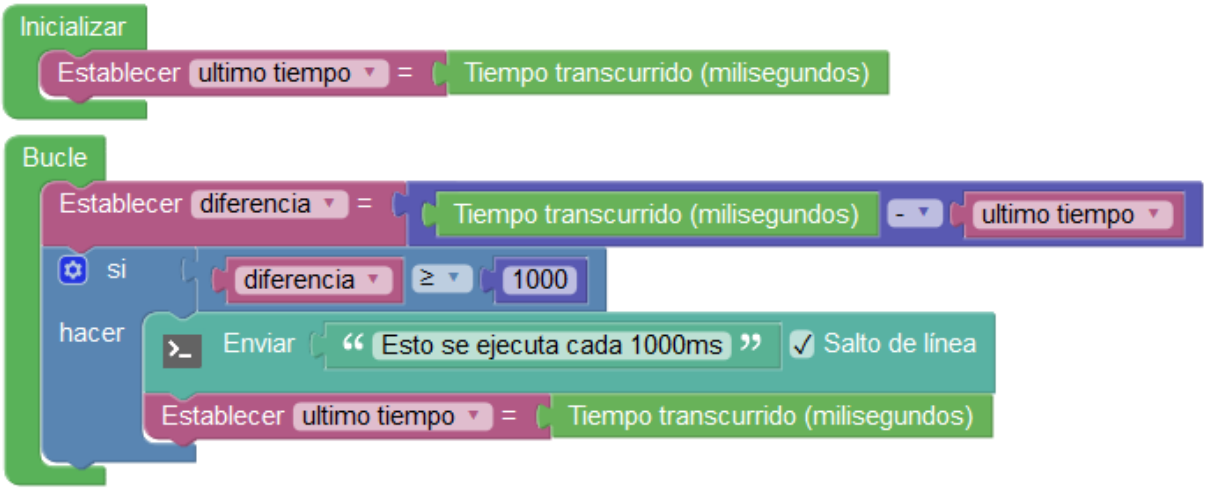
  IMPORTANTE: Este bloque NO bloquea la ejecución del programa



Ejemplo: Ejecuta los bloques en su interior si el tiempo transcurrido desde la última ejecución es mayor o igual a 1000 ms



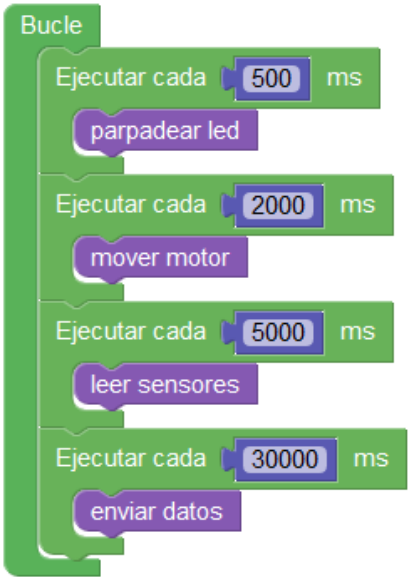
Programa equivalente:



Cuando necesitemos realizar distintas tareas periódicas y que parezca que se ejecuten paralelamente sin bloquearse unas a otras utilizaremos este tipo de bloque “ejecutar cada”.

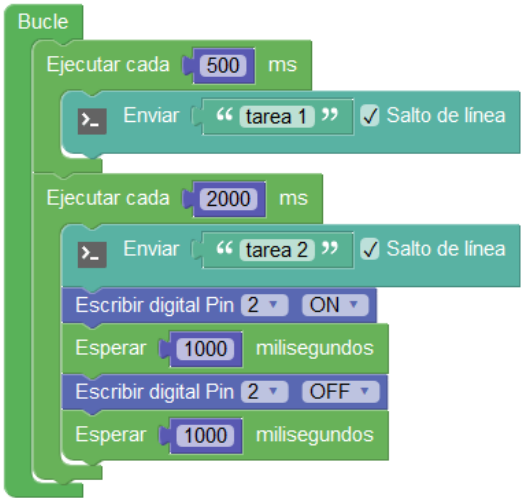
Si en el programa utilizamos bloques como por ejemplo el GPS (Apdo. 3.3.12) obligatoriamente debemos evitar los bloques de “esperar” si queremos que el programa funcione correctamente. Consultar Anexo I para ver los bloques incompatibles con bloques tipo “esperar”

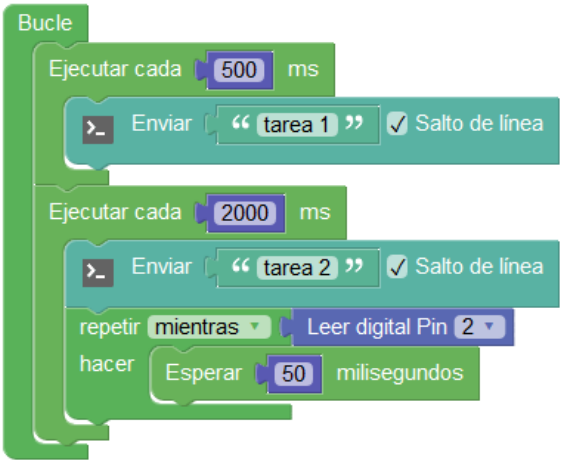
Ejemplo: tareas simultáneas con distintos periodos de ejecución



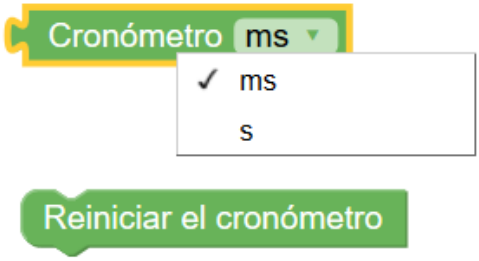
La precisión de la ejecución de tareas de esta forma depende del tiempo que emplea cada tarea, si una tarea “tarda” mucho bloqueará y “retrasará” al resto. Para un funcionamiento correcto cada tarea debe ejecutarse en el menor tiempo posible y no usar nunca bloques de tipo esperar o realizar bucles de indeterminada duración que puedan quedarse en ejecución por tiempo indefinido.

*Ejemplos de lo que **NO** se debería hacer dentro de las tareas:*

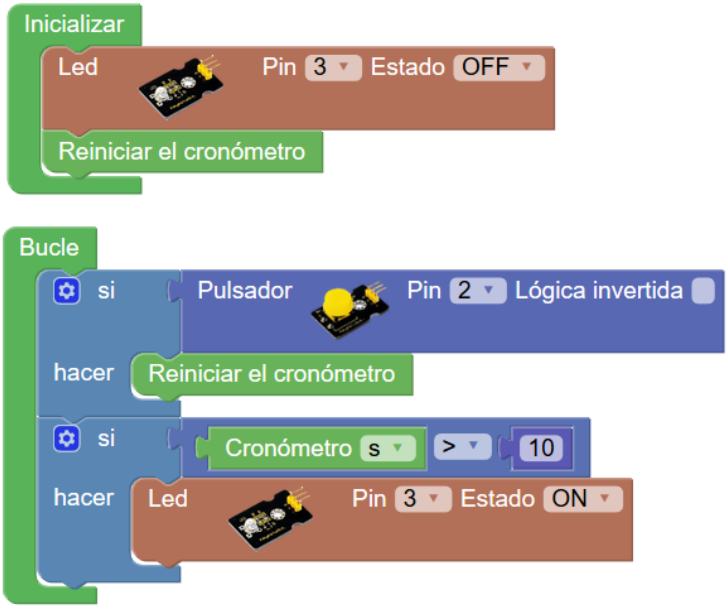




- **Cronómetro:** Permite obtener el tiempo transcurrido en ms o s, y es fácilmente puesto a 0 con el bloque reiniciar.



Ejemplo: Juego en el que tenemo que pulsar un botón periódicamente sin dejar que pase nunca más de 10s entre pulsaciones, en caso contrario se ilumina un led indicando que hemos perdido



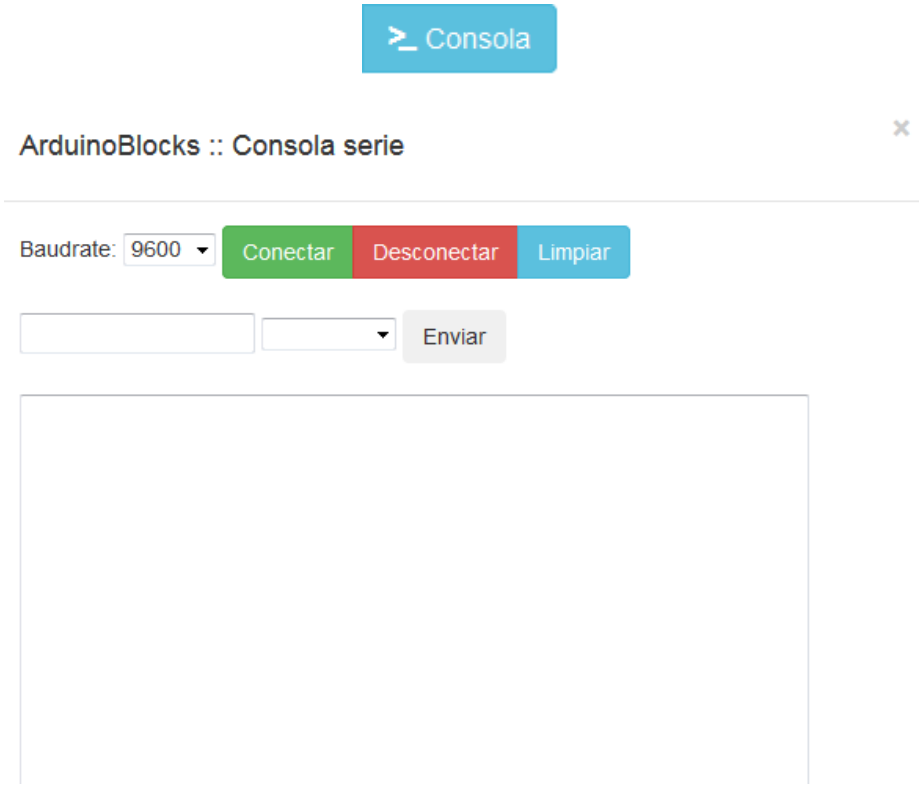
- **Dormir:** Este bloque pone la placa Arduino en modo bajo consumo durante unos milisegundos.

3.3.3 Puerto serie

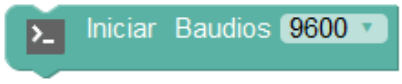
La comunicación vía puerto serie es muy utilizada. Es una vía de comunicación bidireccional sencilla que nos permite enviar información desde Arduino que visualizaremos en la consola o al contrario, enviar información desde la consola que recibiremos en el Arduino.

En muchas ocasiones simplemente se utiliza como una forma de depurar o mostrar información para saber si nuestro programa dentro del microcontrolador de Arduino está funcionando bien, en otros casos se puede utilizar de una forma más compleja sirviendo de vía de comunicación con aplicaciones en un PC, con periféricos como un GPS o comunicando con otros sistemas o por qué no, con otra placa Arduino.

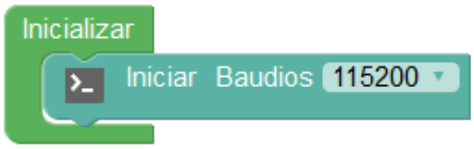
En ArduinoBlocks tenemos acceso a la consola via web (con ArduinoBlocks-Connector instalado) aunque podemos utilizar si lo preferimos cualquier aplicación de consola o terminal serie compatible con nuestro sistema.



- **Iniciar**: Configura la velocidad de la comunicación serie. Este valor debe ser igual en la consola y en el programa Arduino para establecer una comunicación correcta. Por defecto, y si no se pone nada, la velocidad es 9600bps.

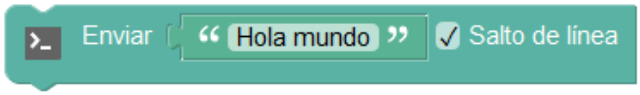


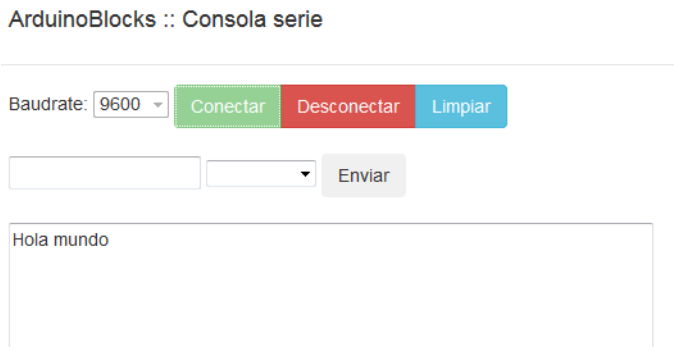
Ejemplo:



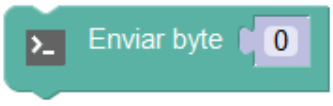
- **Enviar**: Escribe un valor de texto o el valor de una variable en el puerto serie. La opción “Salto de línea” permite añadir o no un retorno de carro al final del envío para bajar de línea.

Ejemplo:

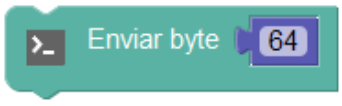




- **Enviar byte:** Envía un valor numérico como un byte (8 bits). Por tanto el valor debe estar comprendido entre 0 y 255.



Ejemplo: Enviar byte con valor 64

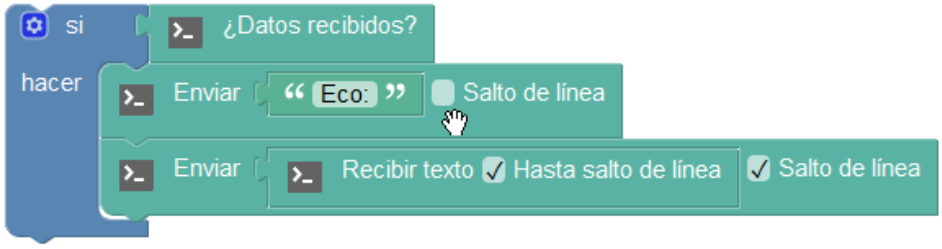


- **¿Datos recibidos?:** Obtiene un valor de verdadero si hay datos recibidos pendientes de procesar o falso si no se ha recibido nada por la conexión serie.

Ejemplo: Si hay datos pendientes de leer activar el pin 13

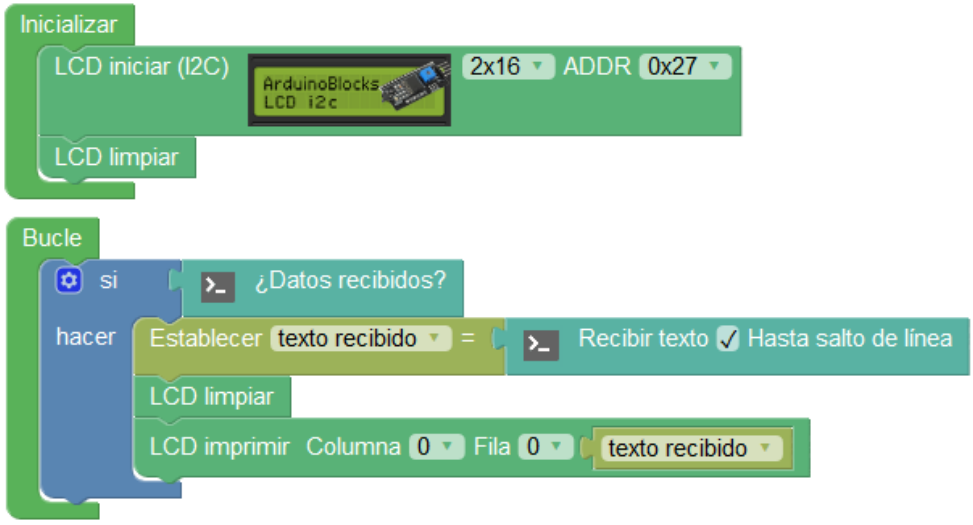
- **Recibir texto:** Lee una cadena de texto recibida por el puerto serie. Si se indica la opción “hasta salto de línea” en cuanto se encuentra un salto de línea devuelve el texto recibido. Si no, hasta que se dejen de recibir datos.

Ejemplo: Devolver como eco lo mismo que se ha recibido



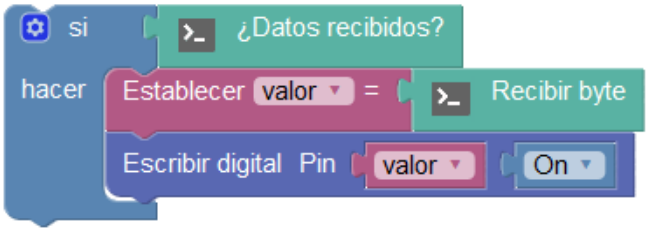
Ejemplo:

Ejemplo: Mostrar texto recibido por serie en una pantalla LCD



- **Recibir byte:** Leer un byte (8 bits) del puerto serie

Ejemplo: Activar el pin correspondiente al byte recibido



- **Recibir como número:** Leer una cadena de texto recibida por el puerto serie e intenta interpretarla como un número (analiza la cadena de texto buscando un formato numérico válido)

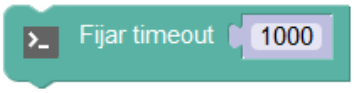
Al pulsar “Enviar” en la consola serie, se envía a Arduino:

“1” + código de salto de línea (\n)

En este caso si no está activada la casilla “hasta salto de línea” leeremos el valor enviado “1” y luego un “0” (por error de intentar interpretar el salto de línea como un número)

Ejemplo: Recibe un número enviado como texto desde la consola. Interpreta el número. Si es "1" activa el pin 13 , si es "2" apaga el pin 13:

- **Fijar timeout:** Establece el tiempo máximo de espera en la recepción de datos (valor en milisegundos).



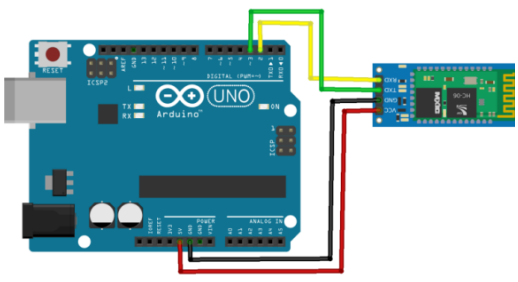
3.3.4 Bluetooth

La comunicación con el módulo Bluetooth HC-06 es exactamente igual que la del puerto serie, de hecho lo que hace el módulo Bluetooth es encapsular toda la información serie a través de una conexión serie virtual a través de un perfil Bluetooth de emulación de puerto serie.

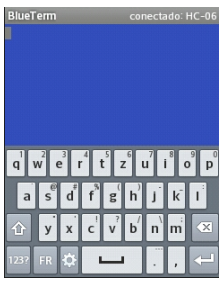
Podemos simular una conexión serie con un dispositivo móvil (con Bluetooth compatible con el perfil de puerto serie), un PC u otro módulo Bluetooth similar en otro dispositivo.

Arduino UNO sólo posee un puerto serie implementado en su hardware, para no utilizar el módulo Bluetooth en los pines 0 y 1 (correspondientes al puerto serie hardware) e interferir con la comunicación serie o la programación del dispositivo (como hacen otros entornos) los bloques de Bluetooth implementan un puerto serie software que funciona exactamente igual pudiendo configurarse en cualquier otro pin digital tanto para RX (recibir) como para TX (transmitir).

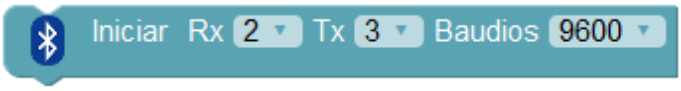
Ejemplo de conexión del módulo Bluetooth HC-06



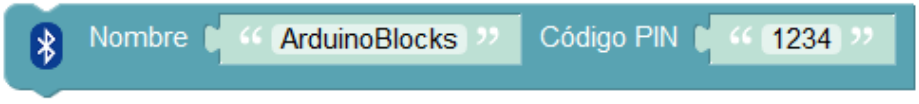
BlueTerm Android



- **Iniciar**: Permite configurar los pines donde está conectado el módulo Bluetooth y la velocidad a la que vamos a trabajar.

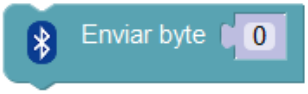


- **Nombre:** El módulo Bluetooth HC-06 permite configurar el nombre y el código PIN a través de comandos. Con este bloque podemos hacerlo fácilmente, el único requisito para que funcione es que ningún dispositivo Bluetooth esté conectado en ese momento al módulo HC-06. Por otro lado normalmente es necesario reiniciar el módulo para que aparezca la nueva configuración (y desemparejar el dispositivo móvil si ya lo estaba).

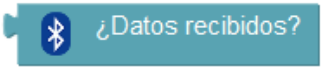


- **Enviar:** Escribe un valor de texto o el valor de una variable en el puerto serie. La opción “Salto de línea” permite añadir o no un retorno de carro al final del envío para bajar de línea.

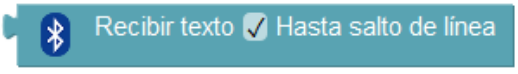
- **Enviar byte:** Envía un valor numérico como un byte (8 bits). Por tanto el valor debe estar comprendido entre 0 y 255.



- **¿Datos recibidos?:** Obtiene un valor de verdadero si hay datos recibidos pendientes de procesar o falso si no se ha recibido nada por la conexión serie.

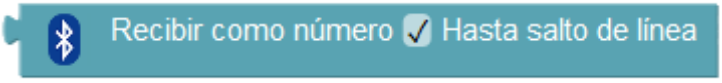


- **Recibir texto:** Lee una cadena de texto recibida por el puerto serie. Si se indica la opción “hasta salto de línea” en cuanto se encuentra un salto de línea devuelve el texto recibido. Si no, hasta que se dejen de recibir datos.

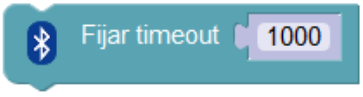


- **Recibir byte:** Leer un byte (8 bits) del puerto serie.

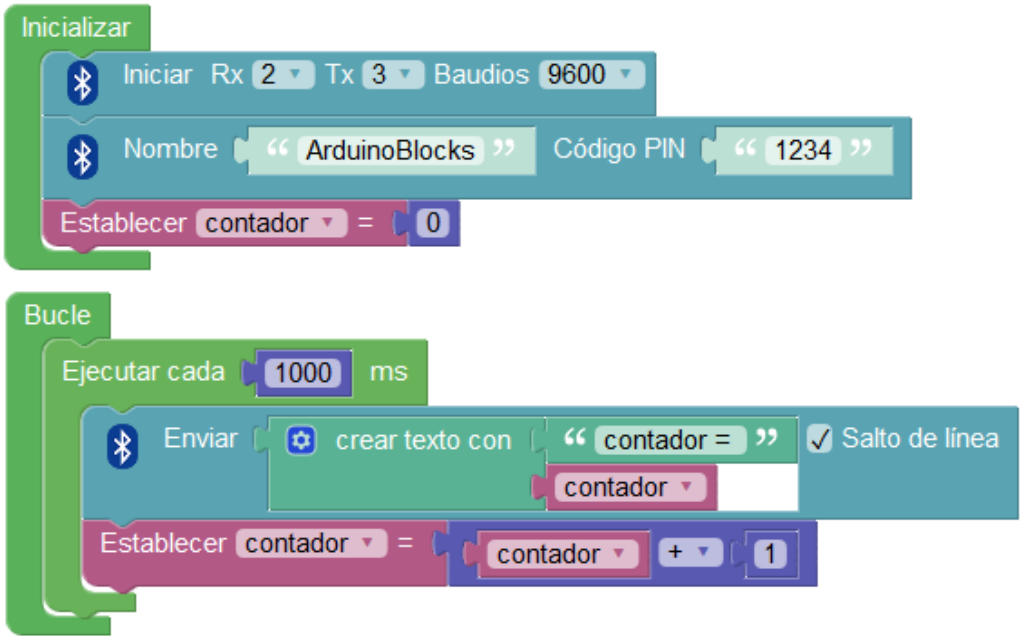
- **Recibir como número:** Leer una cadena de texto recibida por el puerto serie e intenta interpretarla como un número. Funciona igual que el boque del puerto serie (ver detalles de funcionamiento en el puerto serie)



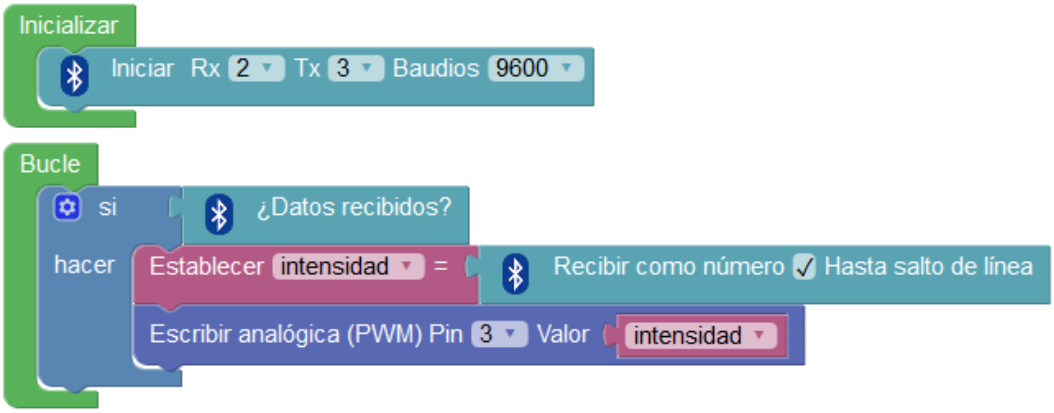
- **Fijar timeout:** Establece el tiempo máximo de espera en la recepción de datos por la conexión serie Bluetooth (valor en milisegundos)



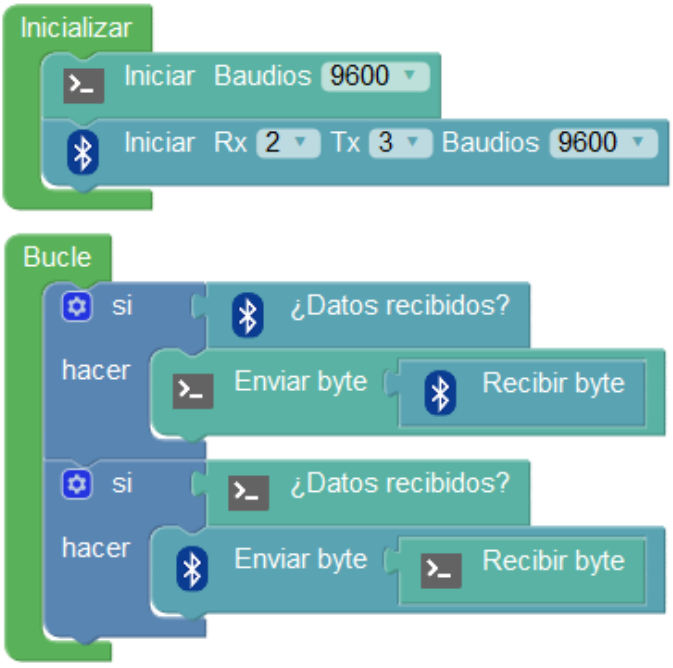
Ejemplo: Envío de una variable contador a través de Bluetooth



Ejemplo: Recepción de un valor por para establecer la intensidad de un led



Ejemplo: pasarela serie <-> Bluetooth



3.3.5 Sensores

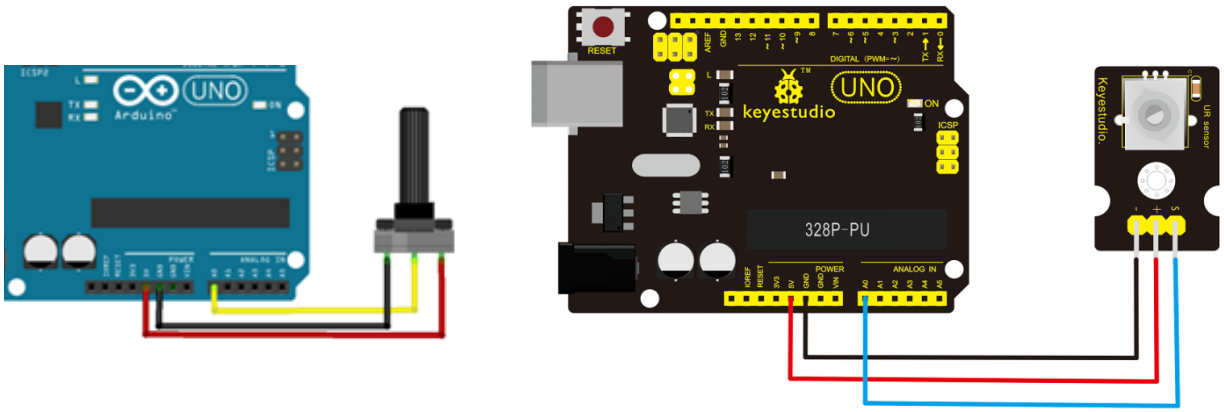
En el mercado existen infinidad de sensores y módulos para Arduino, aunque con los bloques genéricos descritos en el apartado 3.3.1 (entrada/salida) podemos leer la información de la mayoría de sensores digitales y analógicos ArduinoBlocks implementa bloques específicos para los sensores más comunes del mercado. Estos bloques a veces se limitan a leer la información digital o analógica, según el tipo de sensor, y en otros casos realizan una adaptación de los datos leídos para ajustarlos a la realidad (por ejemplo al leer un sensor de temperatura adapta la lectura a grados centígrados con un cálculo interno).

ArduinoBlocks incorpora bloques para la mayoría de sensores modulares que podemos encontrar en el mercado, algunos muy populares como los sensores de Keyestudio y similares.

ArduinoBlocks es una plataforma online en continua evolución por lo que seguramente desde la edición de este libro ya incorporará nuevos sensores con nuevas funcionalidades.

- **Sensor potenciómetro:** Nos permite obtener la posición del mando rotativo. Ángulo de operación de unos 270º. Varía el valor de voltaje aplicado a la entrada en función de la posición de su resistencia variable interna.

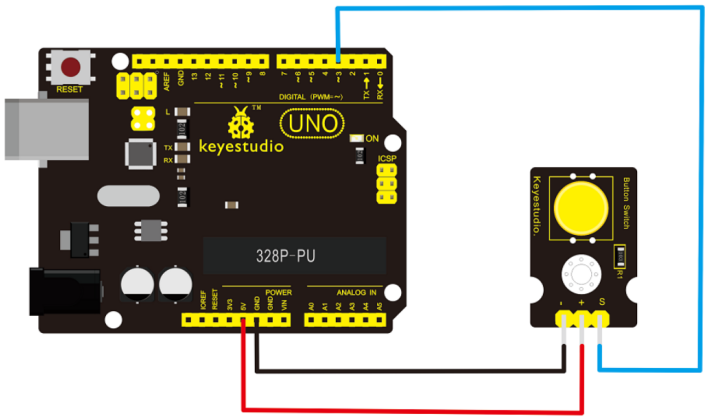
  Tipo: Analógico Pin: A0-A5 Valor: 0-100 (%) / 0..1023



Ejemplo: Sensor potenciómetro conectado al pin analógico A0 para ajustar una variable de temperatura a un valor entre 5 y 30 grados.

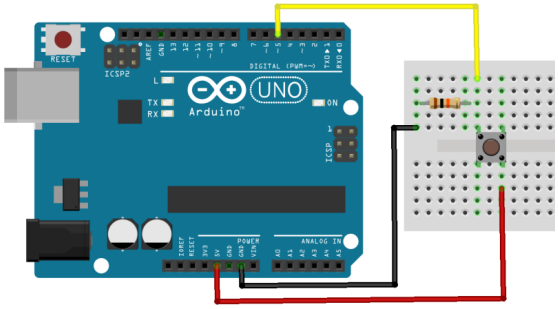
- **Sensor pulsador:** Botón para interactuar de forma táctil.

  Tipo: Digital Pin: 2-13/A0-A5 Valor: 0/1 (F/V, Off/On)

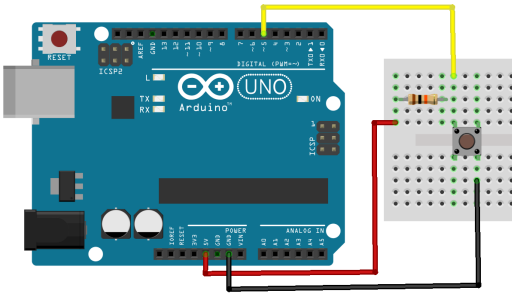


Dependiendo de la conexión que hagamos del pulsador, o en caso de utilizar módulos de pulsador de diferentes fabricantes, la lógica de funcionamiento del pulsador puede ser diferente:

Conexión:
sin presionar "off" / presionado: "on"



Conexión:
sin presionar "on" / presionado: "off"

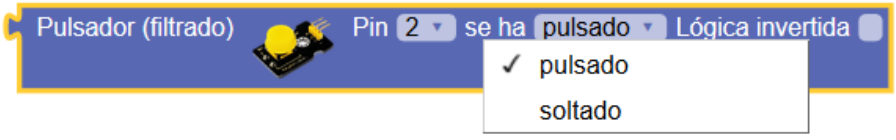


Alguno módulos de pulsador internamente trabajan de forma inversa por su conexión interna. En ese caso el pulsador siempre está dado una señal "On" y cuando lo pulsamos genera la señal "Off". En ese caso podemos invertir la condición para detectar cuando está pulsado.

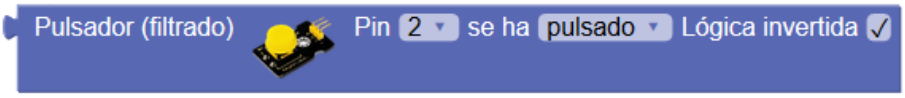
Ejemplo: encender un led cuando el pulsador está presionado y apagar si no lo está

Ejemplo con funcionamiento similar al anterior:

- **Pulsador filtrado:** Es un bloque similar al anterior pero que internamente hace un procesamiento de la señal del pulsador evitando “rebotes” y además detectando el “pulsar” y el “soltar”

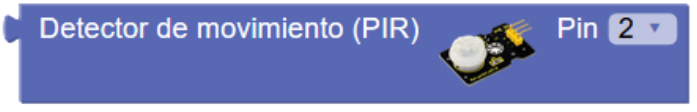


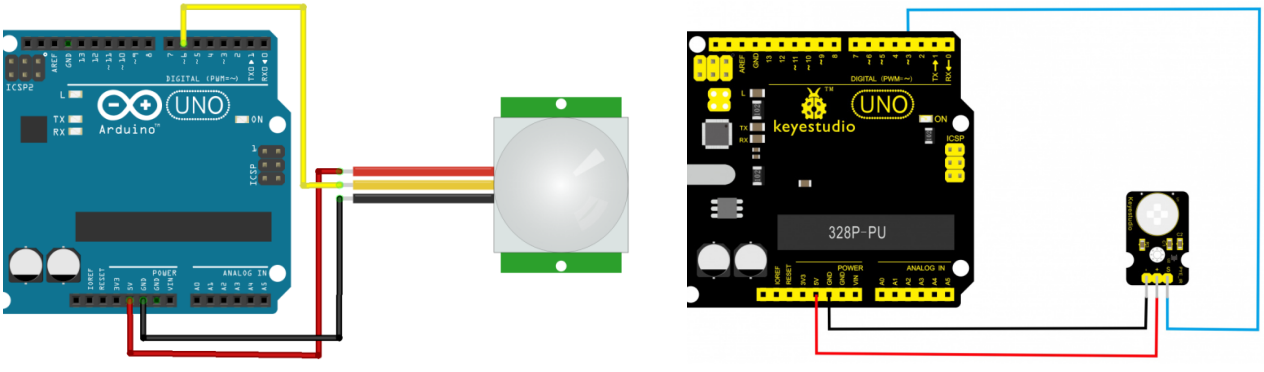
Al igual que en el anterior, si por la conexión interna del pulsador la señal lógica funciona al revés (“on” en reposo y “off” al pulsar) podemos indicar que trabajo con lógica invertida:



- **Sensor de movimiento (PIR):** Se activa cuando detecta movimiento a su alrededor, a partir de un tiempo sin detección el sensor vuelve a desactivarse.

Tipo: Digital Pin: 2-13/A0-A5 Valor: 0/1 (F/V, Off/On)



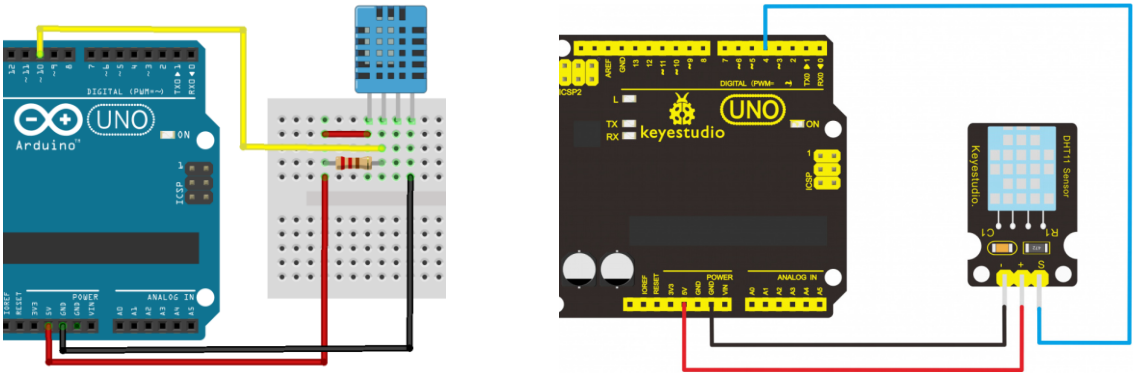


Ejemplo: Encendido del led del pin 13 al detectar movimiento.
Sensor PIR conectado al pin 6:

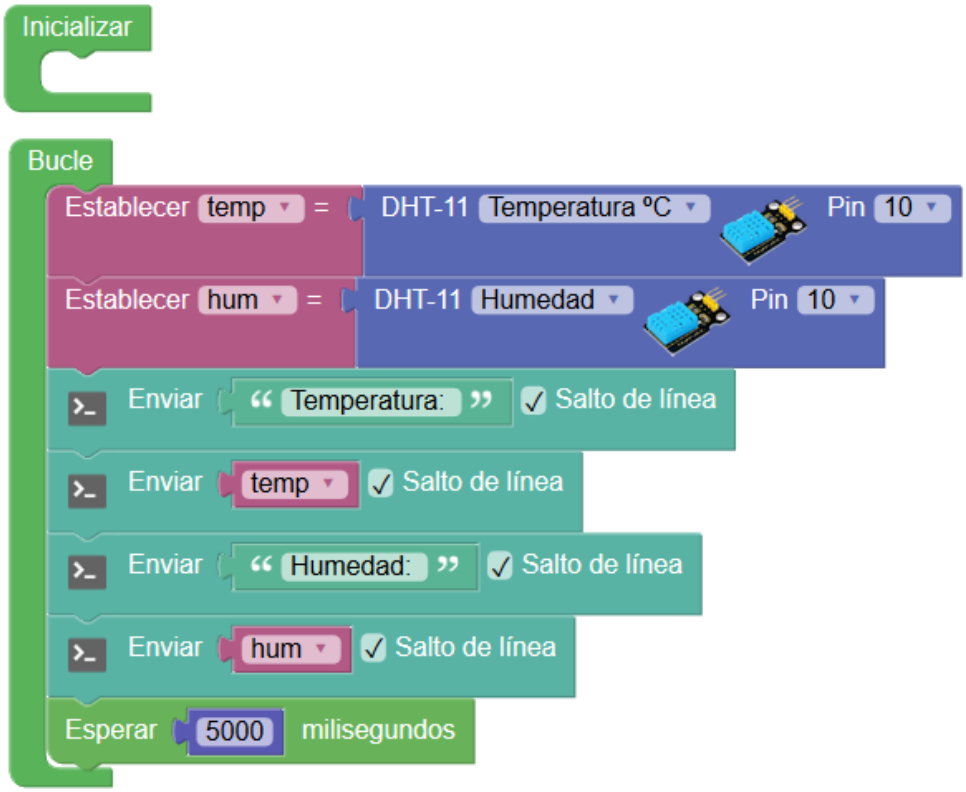
- **Sensor de temperatura y humedad (DHT-11):** El sensor DHT-11 es un sensor que utiliza un protocolo de comunicación propio para facilitarnos el valor de temperatura y humedad ambiente. ArduinoBlocks internamente utiliza una librería para obtener la información decodificada del sensor. Es un sensor de baja precisión pero muy económico y versátil. En un único pin nos permite obtener dos valores con una precisión suficiente para muchas aplicaciones sencillas.

  Tipo: Datos Pin: 2-13/A0-A5

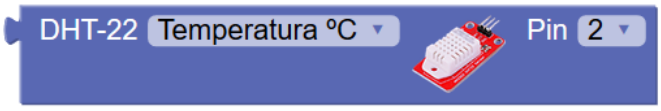
  Valor: Temperatura: 0-50ºC ±2ºC / Humedad: 20-90% ±5%



Ejemplo: Mostrar por la consola cada 5 segundos el valor de temperatura y humedad. Sensor conectado al pin 10.



- **Sensor de temperatura y humedad (DHT-22):** El sensor DHT-22 es una versión mejorada del sensor DHT-11 con mayor rango de medida y precisión.

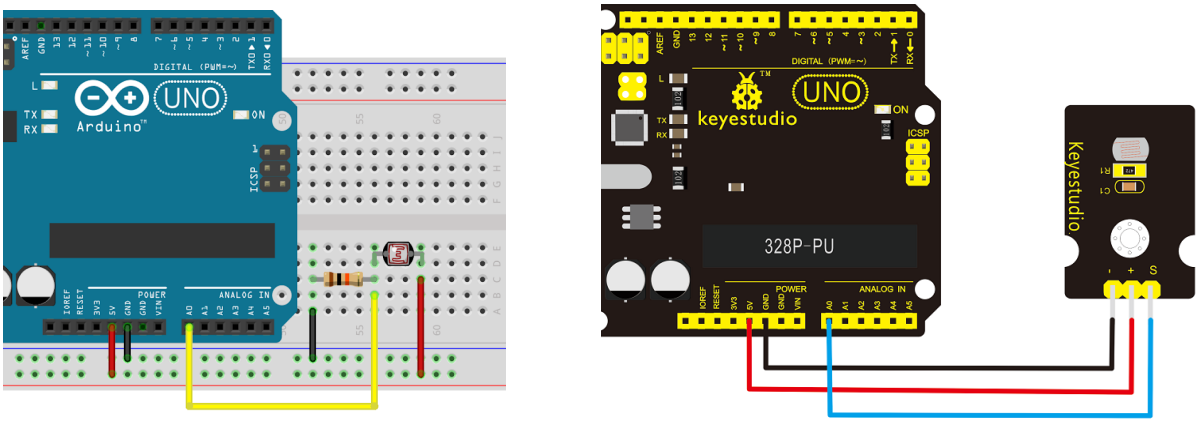


Tipo: Datos Pin: 2-13/A0-A5

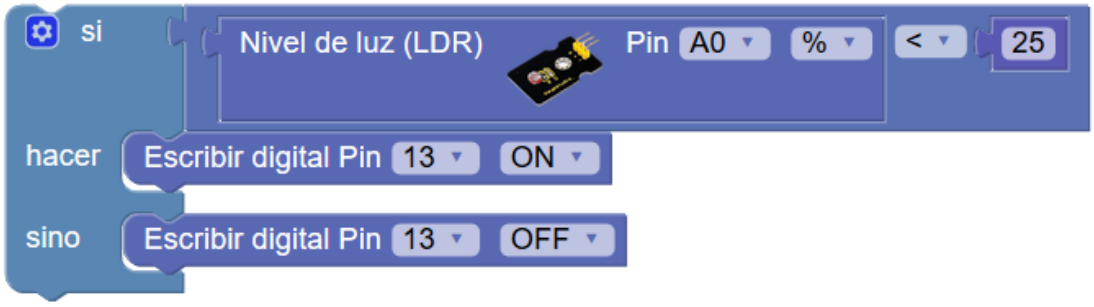
Valor: Temperatura: -40º - 125ºC ±0.5ºC / Humedad: 0-100% ±2%

- **Sensor de luz (LDR):** Obtiene el nivel de luz ambiente mediante la resistencia LDR que varía en función de la luz ambiente aplicada.

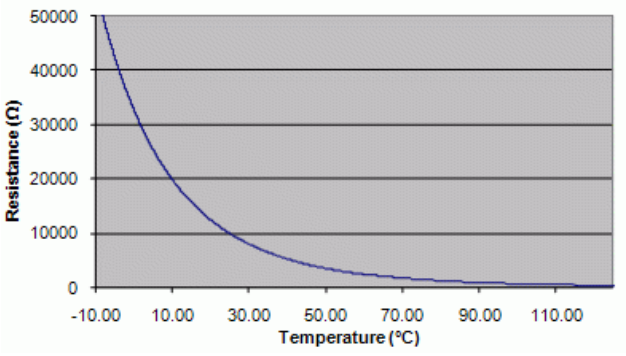
Tipo: Analógico Pin: A0-A5 Valor: 0-100 (%) / 0..1023



Ejemplo: Encendido de un led cuando el nivel de luz es inferior al 25%

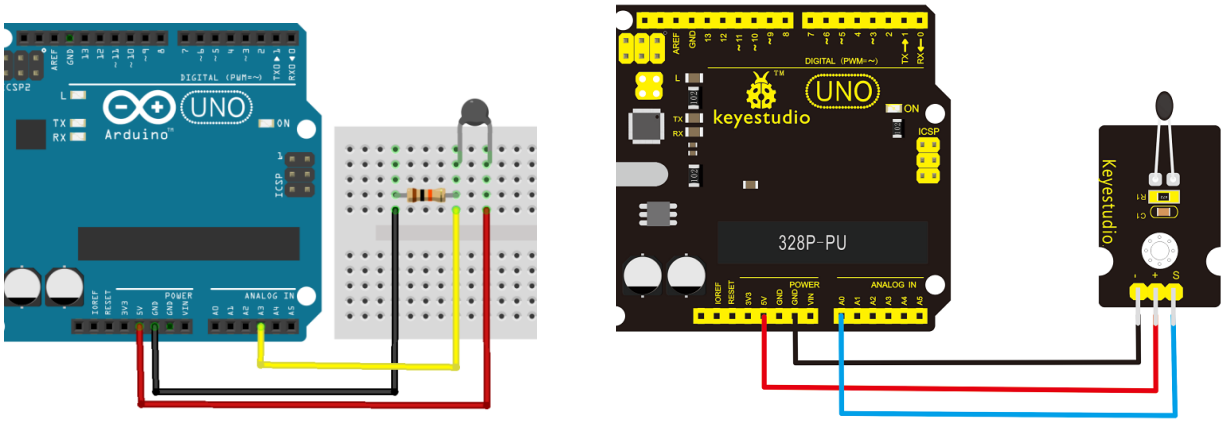


- **Sensor de temperatura (NTC):** Obtiene el valor de la temperatura ambiente. Utiliza una resistencia variable NTC que varía su valor en función de la temperatura ambiente. La relación resistencia/temperatura no es lineal, pero internamente se calcula el valor en grados aplicando la siguiente fórmula para obtener el valor corregido en ºC:



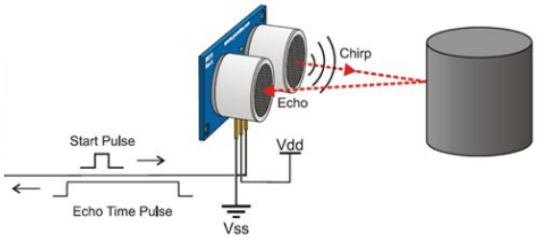
$$T_C = \frac{1}{\frac{1}{B}\ln(\frac{R}{R_N}) + \frac{1}{T_N + 273}} - 273$$

Tipo: Analógico Pin: A0-A5 Valor: -40…125ºC



Ejemplo: Mostrar temperatura por la consola serie cada 5 segundos.
Sensor conectado al pin A3:

- **Sensor de distancia (HC-SR04):** El sensor genera una serie de tonos de ultrasonidos (no audibles), estos tonos si rebotan en una superficie vuelven y son captados por un receptor de ultrasonidos que incorpora el propio sensor. Midiendo el tiempo que tardan en volver los ultrasonidos podemos calcular la distancia a la que se encuentra el objeto sobre el que han rebotado.



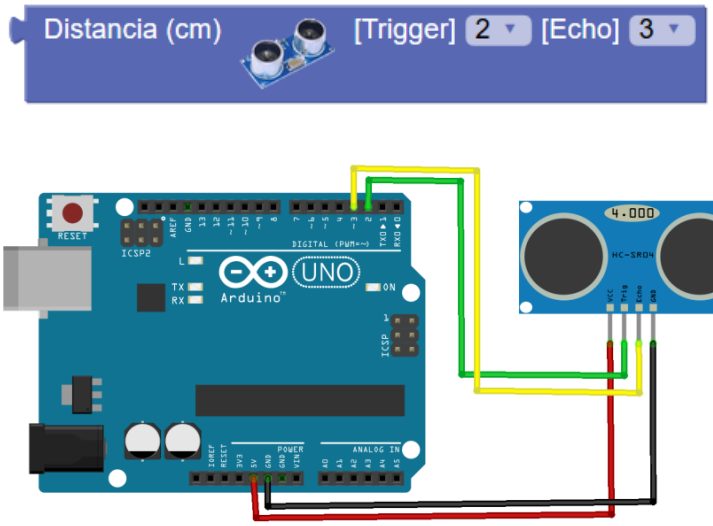
$$d_{sensor} = \frac{\Delta t \cdot 340\frac{m}{s}}{2}$$

Tipo: Datos

Pin Trigger (emisión): 2-13/A0-A5

Pin Echo (recepción): 2-13/A0-A5

Valor: 2 – 400 cm



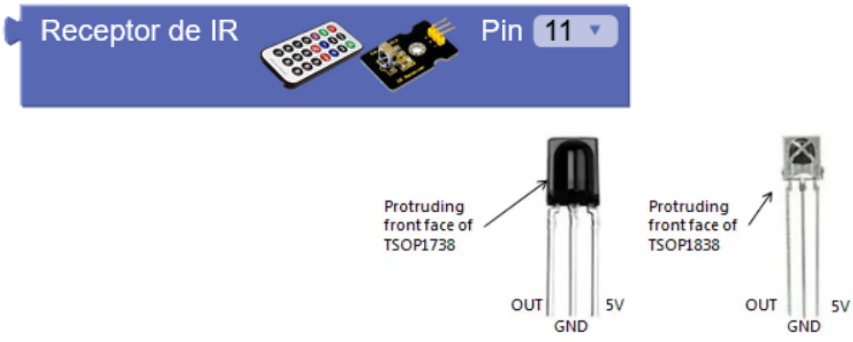
Ejemplo: Activación del led en el pin 13 cuando se detecta un objeto entre 10 y 80 cm de distancia

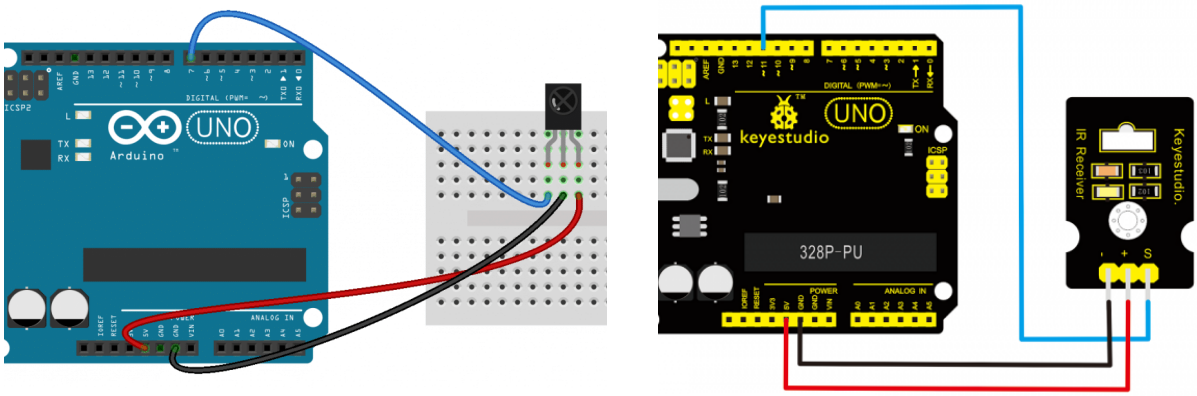
- **Sensor receptor de infrarrojos**: Permite decodificar los protocolos de señales de pulsos infrarrojos utilizados por los mandos a distancia.

  Protocolos detectados: RC5, RC6, NEC, SONY, PANASONIC,JVC, SAMSUNG, WHYNTER, AIWA, LG, SANYO, MITSUBISHI, DENON.

  Tipo: Datos Pin: 2-13/A0-A5

  Valor: código recibido / 0 = ningún código detectado.





Dependiendo el tipo de mando recibiremos unos códigos con valores de un tamaño u otro. Algunos mandos utilizan códigos de 32 bits, al almacenar el valor del código recibido en una variable de ArduinoBlocks se convierte a un valor decimal de 32 bits con signo y eso puede producir una alteración en el valor mostrado (número decimales extraños).

Para evitar este problema **debemos tratar el valor como un valor entero sin signo de 32 bits** añadiendo el bloque "*Número entero sin signo*" visto en el apartado de bloques matemáticos (3.2.3).

Ejemplo: Mostrar por consola el código recibido:

Si utilizamos mandos genéricos RC5 como el modelo de Keyestudio, podemos acceder a los códigos directamente con el bloque correspondiente:

Ejemplo: Detectar teclas "izquierda" y "derecha" de un mando a distancia

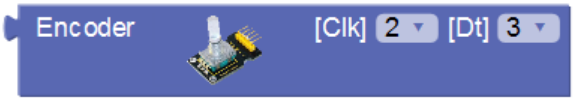
- **Sensor encoder (codificador) rotativo:** Un encoder rotativo es un elemento que indica su posición mediante posiciones codificadas. Cuando pasamos por cada paso se nota un pequeño salto que indica que se ha llegado a la nueva posición. Estos codificadores constan de dos pines de señal para el codificador y un pin para un pulsador que lleva integrado. Los dos pines del codificador nos dan la información en forma digital con un total de 4 combinaciones: 00, 01, 10, 11.

El encoder no tiene ninguna posición predefinida y no tiene límite de giro en ningún sentido. Automáticamente mantiene un valor interno con la posición virtual según los pasos en un sentido u otro, empezando siempre en 0.

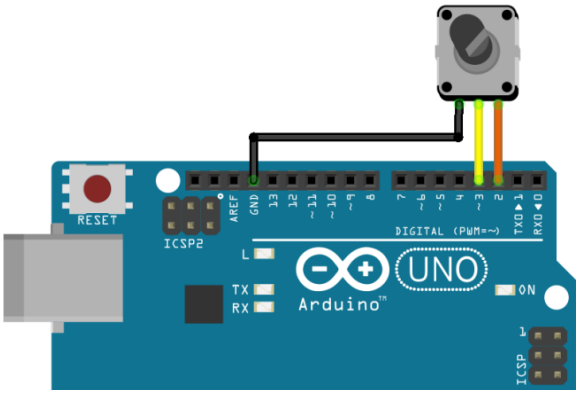
Tipo: Datos Pin: Clk (A): 2 / Dt (A): 3

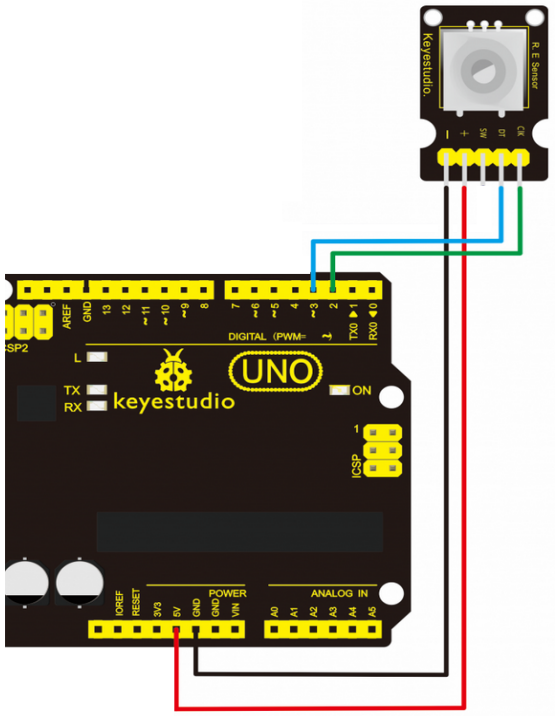
Valor: posición virtual del encoder (variable interna)

Obtener la posición "*virtual*" actual del encoder:

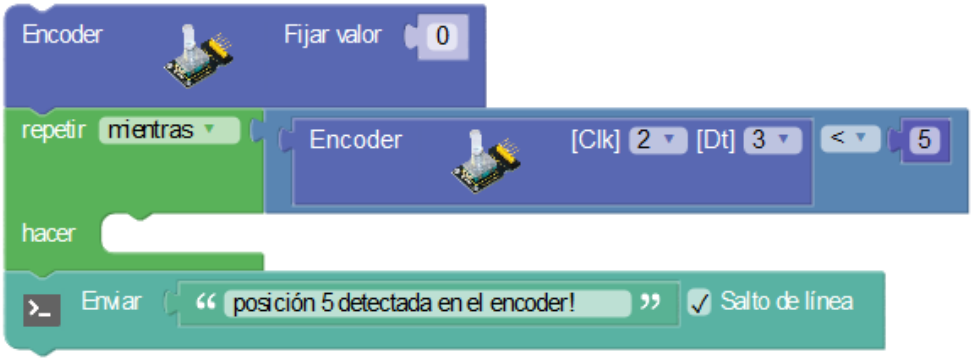


Fijar el valor de posición "*virtual*" a un valor.





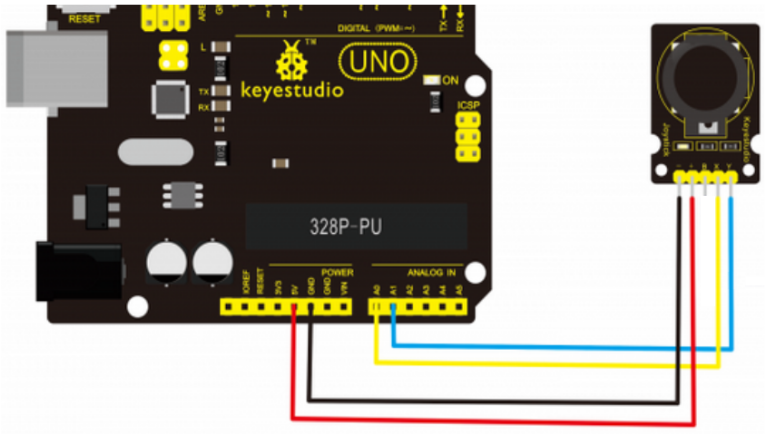
Ejemplo: Esperar hasta que el encoder recorra 5 saltos (clicks) hacia la derecha.



- **Sensor de joystick:** Este tipo de sensor de palanca se basa en dos potenciómetros que detectan la posición en cada uno de los ejes X e Y.

  Tipo: Analógico Pin: A0-A5

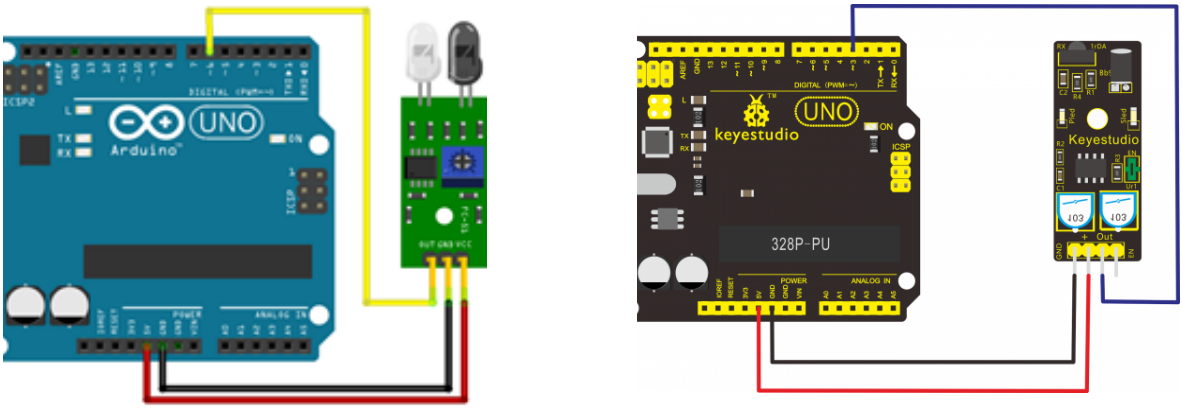
  Valor: 0-100 (%) / 0..1023



Ejemplo: Mostrar en la consola la posición X e Y:

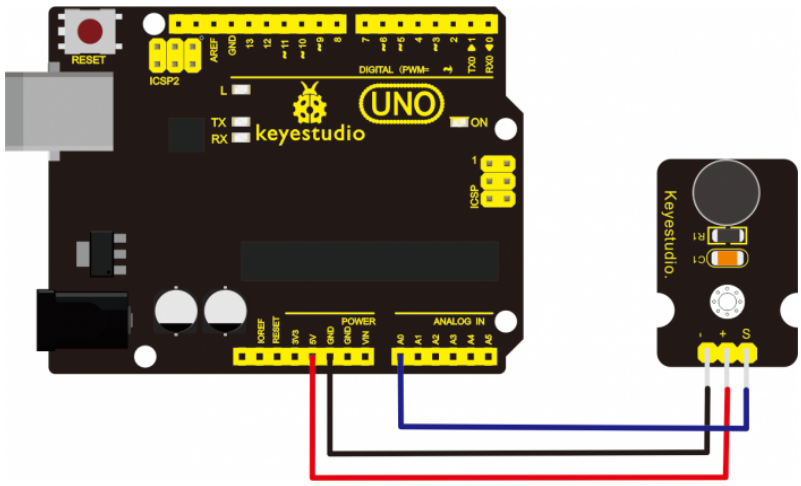
- **Sensor detector de obstáculos (IR):** Mediante el uso de un diodo emisor de IR y un fototransistor receptor de IR permite detectar cuando hay un obstáculo cerca por el reflejo de la luz IR. Este tipo de sensor permite normalmente un ajuste para definir la distancia a la que se activa el sensor.

  Tipo: Digital Pin: 2-13/A0-A5 Valor: 0/1 (F/V, Off/On)

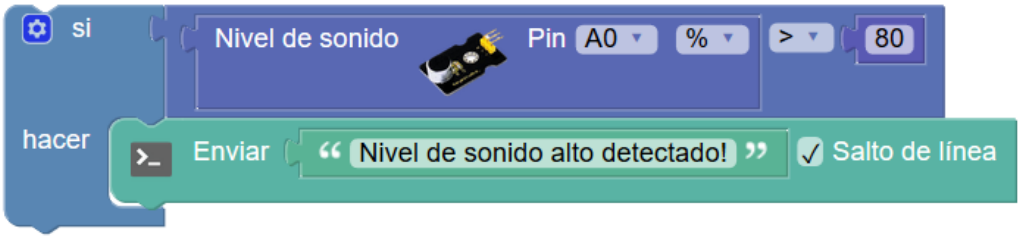


- **Sensor de nivel de sonido:** Detecta el nivel de sonido ambiente.

  Tipo: Analógico Pin: A0-A5/A0-A5 Valor: 0-100 (%) / 0..1023



Ejemplo - detector de nivel de sonido alto. Sensor en pin A0



- **Sensor sigue líneas:** Su funcionamiento es idéntico al detector de obstáculos por IR. Se utiliza para detectar superficies blancas/negras.

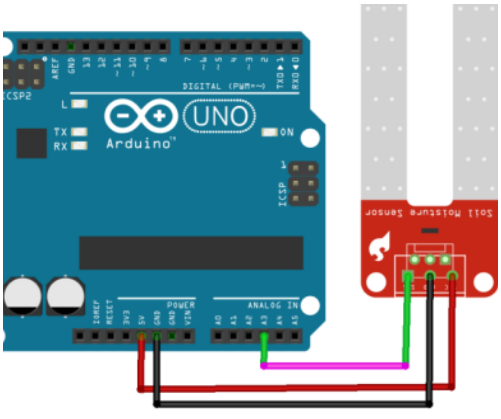
  Tipo: Digital Pin: 2-13/A0-A5 Valor: 0/1 (F/V, Off/On)

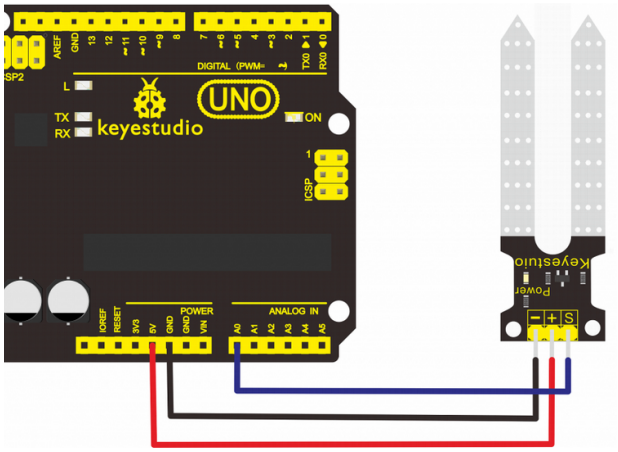
- **Sensor foto-interruptor:** Detecta cuando un objeto interrumpe un haz de luz entre un emisor y receptor.

  Tipo: Digital Pin: 2-13/A0-A5 Valor: 0/1 (F/V, Off/On)

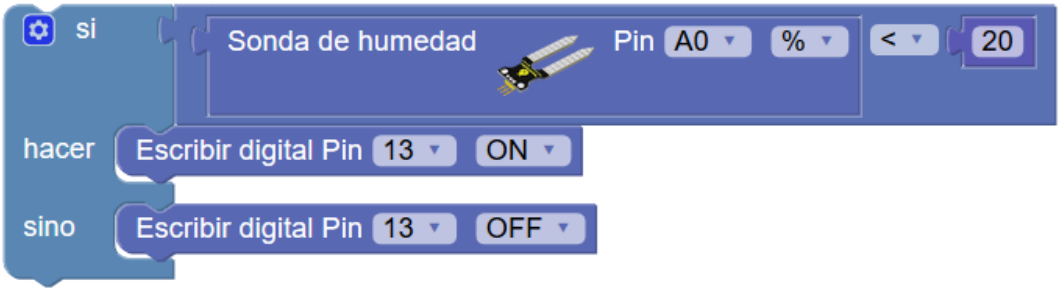
- **Sensor sonda de humedad:** Mide la humedad con la ayuda de una sonda que se introduce en la tierra.

  Tipo: Analógico Pin: A0-A5/A0-A5 Valor: 0-100 (%) / 0..1023





Ejemplo: Activación del led conectado al pin 13 en caso de detectar un nivel de humedad inferior al 20%. Sensor conectado en el pin A3



- **Sensor de lluvia/agua:** Mide el nivel de agua o lluvia.

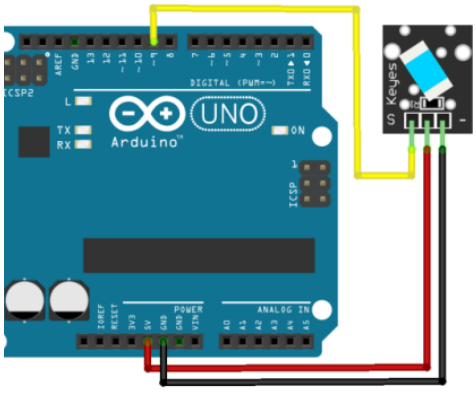
  Tipo: Analógico Pin: A0-A5 Valor: 0-100 (%) / 0..1023

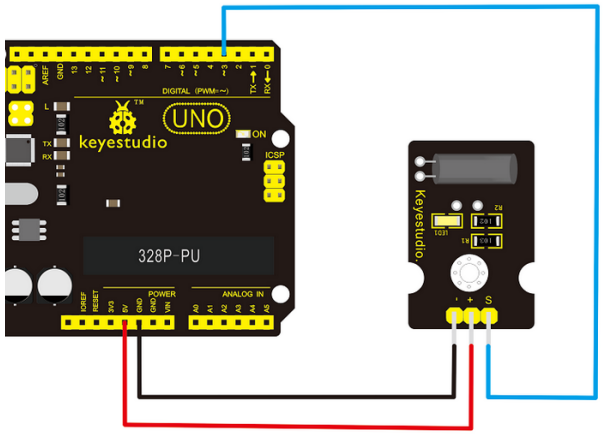
- **Sensor de golpe:** Detecta un impacto o golpe.

  Tipo: Digital Pin: 2-13/A0-A5 Valor: 0/1 (F/V, Off/On)

- **Sensor de orientación:** Detecta si la orientación es vertical / horizontal.

  Tipo: Digital Pin: 2-13/A0-A5 Valor: 0/1 (F/V, Off/On)





- **Sensor de campo magnético:** El sensor se activa con la presencia de un campo magnético cerca. Por ejemplo al acercar un imán al sensor.

  Tipo: Digital Pin: 2-13/A0-A5 Valor: 0/1 (F/V, Off/On)

- **Sensor de vibración:** Se activa cuando detecta una vibración.

  Tipo: Digital Pin: 2-13/A0-A5 Valor: 0/1 (F/V, Off/On)

- **Sensor detector de llama:** Detecta el nivel de fuego o una llama detectando la frecuencia de luz del fuego.

  Tipo: Analógico Pin: A0-A5/A0-A5 Valor: 0-100 (%) / 0..1023

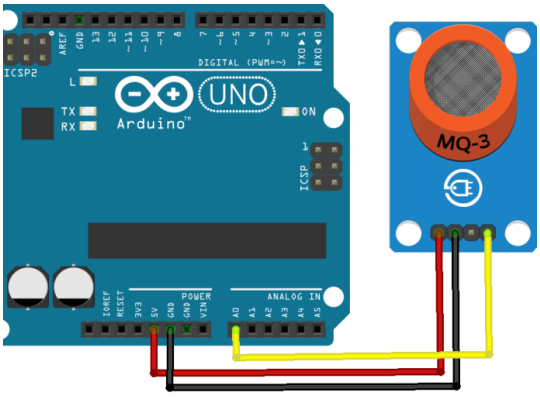
- **Sensores de gas MQ-X:** Están diseñados para detectar ditintos gases. Por lo general no son muy precisos. Los datos de estos sensores se obtienen sin procesar, con un valor entre 0...1023 o ajustado a %. En todo caso se deberá consultar las características detalladas del fabricante del sensor en concreto para interpretar el dato.

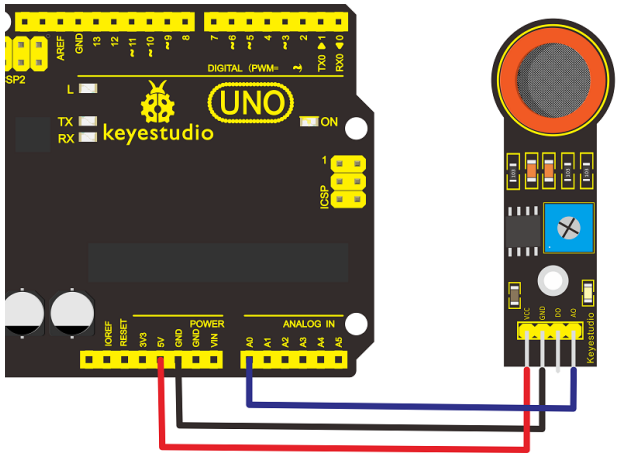
  Tipo: Analógico Pin: 0-A5/A0-A5 Valor: 0-100 (%) / 0..1023

MQ-2	*Metano, butano, GLP, humo*
MQ-3	*Alcohol, Etanol, humo*
MQ-4	*Metano, gas natural comprimido (GNP)*
MQ-5	*Gas natural, GLP*

MQ-6	*Butano, GLP*
MQ-7	*Monóxido de carbono*
MQ-8	*Hidrógeno*
MQ-9	*Monóxido de carbono, gases inflamables*
MQ-135	*Benceno, alcohol, humo, calidad del aire*
MQ-131	*Ozono*
MQ-136	*Ácido sulfhídrico*
MQ-137	*Amoniaco*
MQ-138	*Benceno, tolueno, alcohol, acetona, propano, formaldeido, hidrógeno*

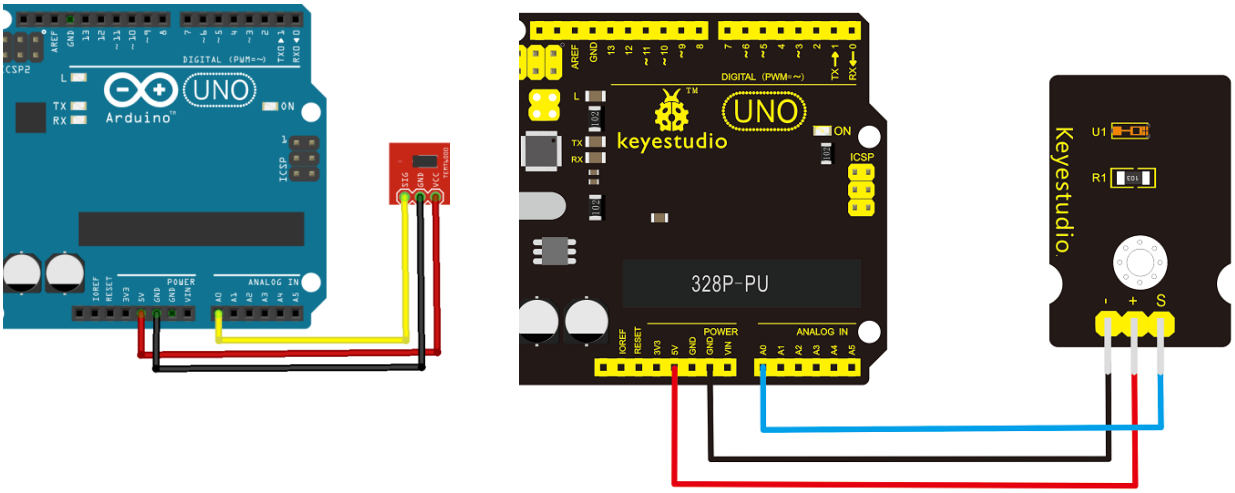
Tipo: Analógico Pin: A0-A5 Valor: 0-100 (%) / 0..1023





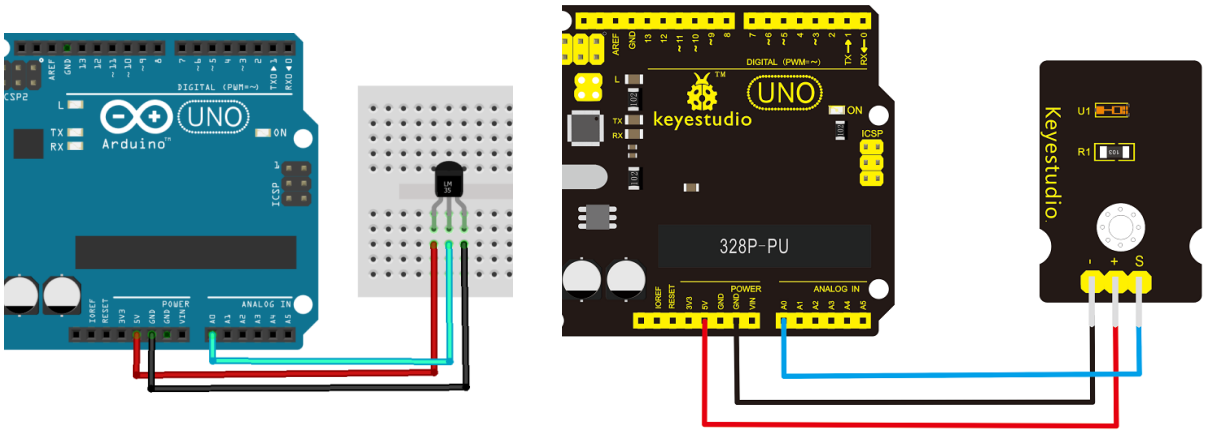
- **Sensor de nivel de luz (TEMT6000):** Permite medir la luz ambiente con alta precisión además de ser un sensor que mide sólo la luz ambiente que percibe el ojo humano, filtrando el espectro de luz no visible y realizando así una medición más real para ciertas aplicaciones.

Tipo: Analógico Pin: A0-A5 Valor: 0-100 (%) / 0..1023



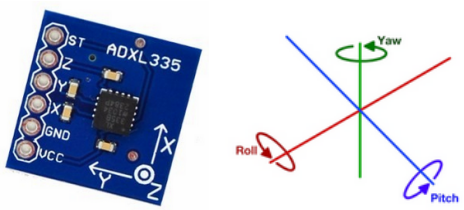
- **Sensor de temperatura (LM35 / TMP36):** Sensor de temperatura calibrado con precisión de 1ºC. La salida es lineal y cada ºC equivale a 10mV. El rango de medida es de -55ºC hasta 150ºC.

  Tipo: Analógico Pin: A0-A5 Valor: -55…150ºC

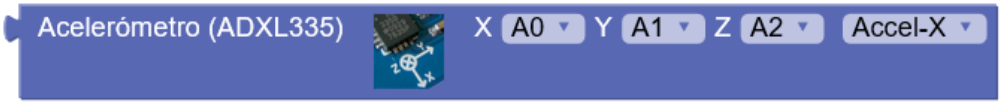


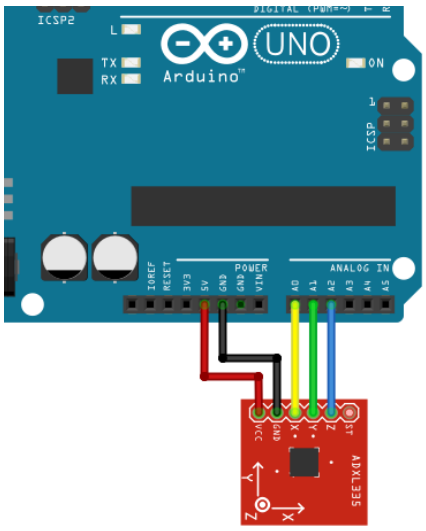
- **Sensor acelerómetro (ADXL335):** Este tipo de sensores permite medir la aceleración en los tres ejes espaciales X,Y y Z. En reposo los ejes X e Y deben tener una valor aproximo de 0G y el eje Z debe tener un valor aproximadamente de 1G (la aceleración de la gravedad). Ante cualquier sacudida o movimiento el sensor nos indicará la aceleración en cada eje en unidades G. Este sensor permite medir desde -3G hasta 3G.

  Internamente con cálculos trigonométricos se pueden obtener los ángulos de rotación en X (Roll) y en Y (Pitch). Sin embargo para calcular la rotación en Z (Yaw) debemos utilizar otro tipo de acelerómetros que se complementan con un giroscopio.



Tipo: Analógico Pin: A0-A5 Valor: -3G … 3G

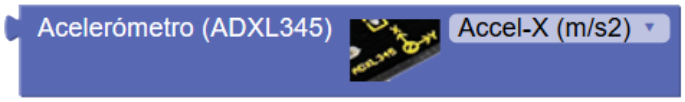


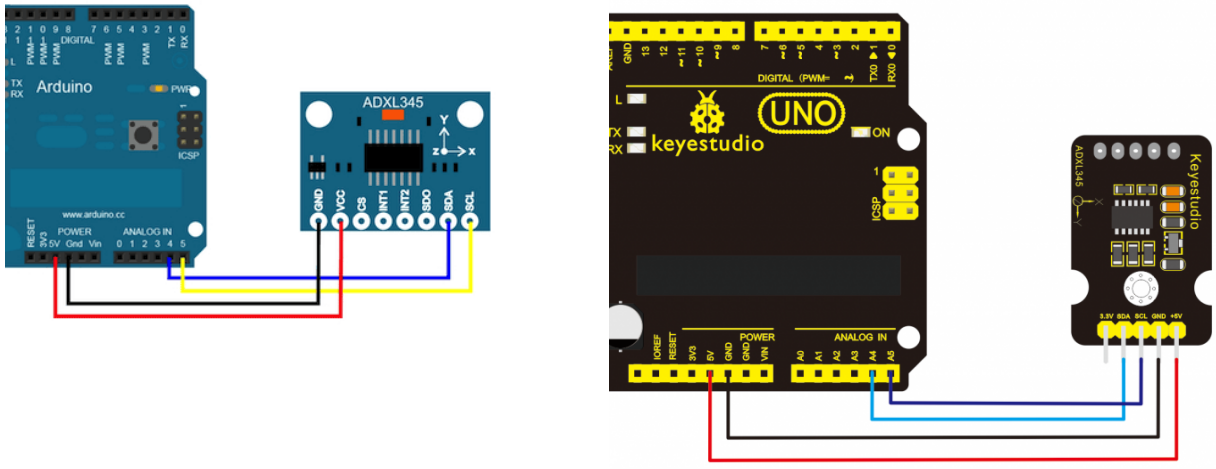


Ejemplo: Mostrar los valores de aceleración por consola serie

- **Sensor acelerómetro (ADXL345):** Acelerómetro de 3 ejes con conexión I2C que permite obtener las aceleraciones en X,Y,Z y los grados de giro X-Roll,Y-Pitch. La conexión I2C lo hace muy fácil de utilizar y obtiene uno datos más precisos que el modelo ADXL335 analógico.

  Tipo: Datos I2C Pin: A4(SDA) / A5 (SCL) Valor: m/s2, º , 0...1023

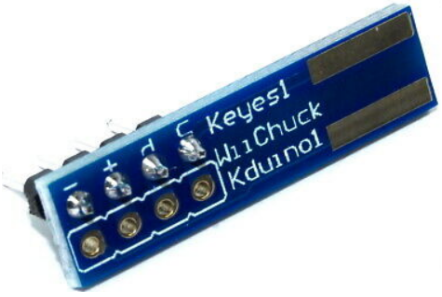




- **Sensor nunchuck:** El controlador Nunchuck se conectar a través de i2c y permite utilizar obtener los datos de sus sensores internos: acelerómetros, joystick y botones.

  Tipo: Datos I2C Pin: A4(SDA) / A5 (SCL)

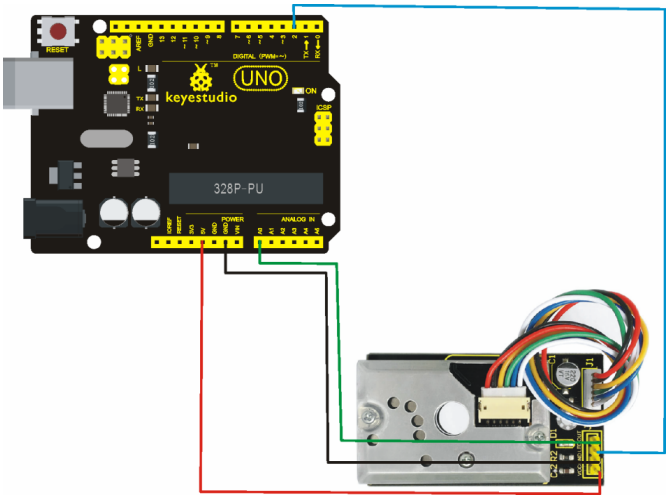
Adaptador/Conector para facilitar la conexión del Wii Nunchuck a los pines Arduino



- **Sensor de polvo PM2.5:** Este sensor mide con alta precisión las partículas en el aire.

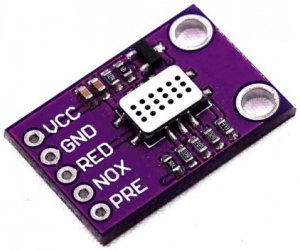
  Tipo: Datos Pin: Digital + Analógico Valor: densidad

Ejemplo de conexión:



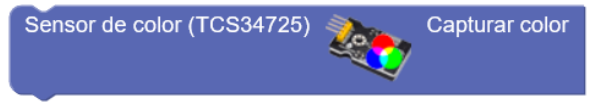
- **Sensor NO2/CO (MICS4514):** Sensor de Monóxido de carbono y nitrógeno.

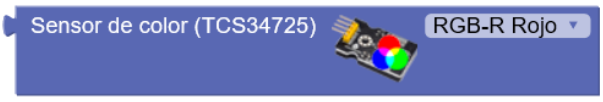
  Tipo: Datos Pin: Digital + Analógico Valor: NO2 / CO

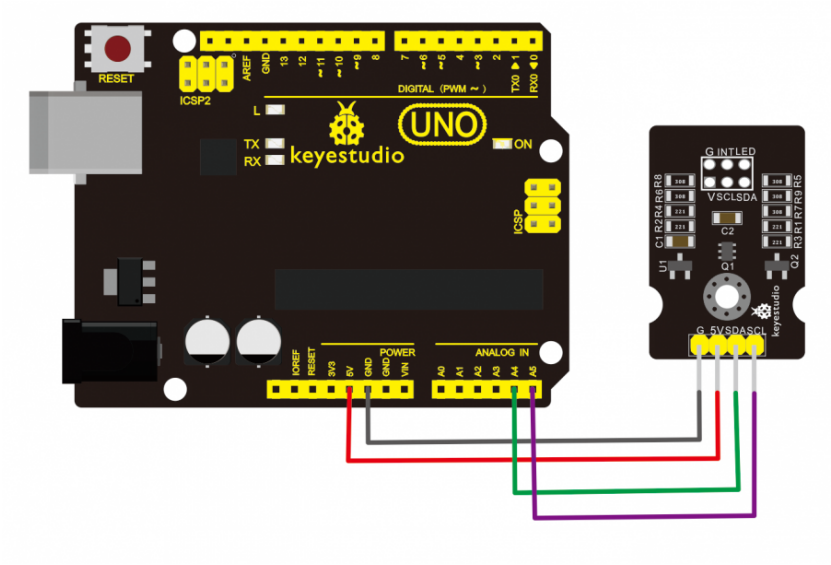


- **Sensor de color (TCS34725):** Sensor con conexión i2c para reconocimiento de colores de las superificies cercanas.

  Tipo: Datos I2C Pin: A4(SDA) / A5 (SCL) Valor: RGB / HSV





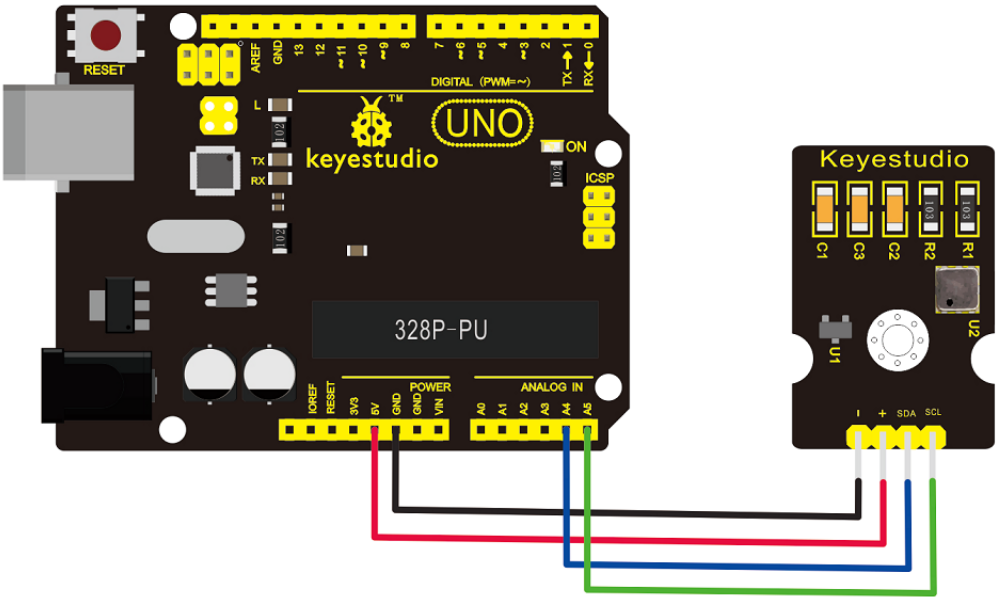


Ejemplo: obtener los valores R.G,B del color detectado

Ejemplo: detectar algunos colores

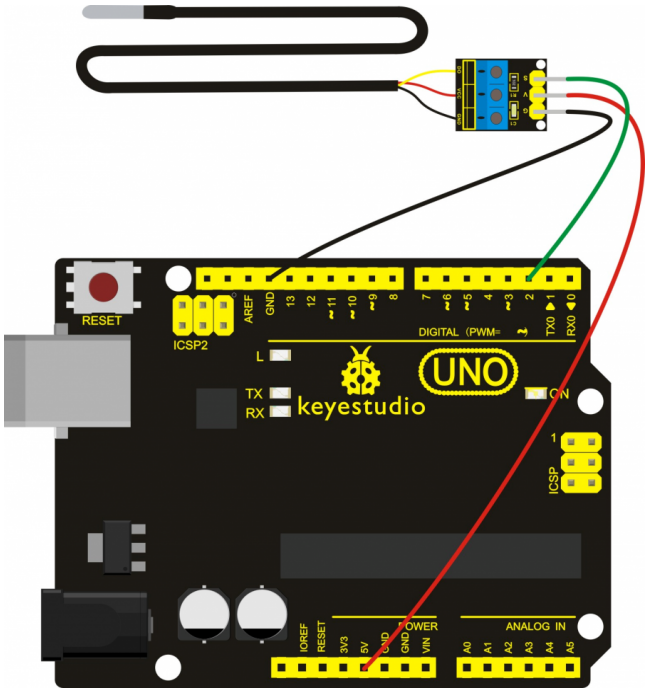
- **Sensor barométrico (BMP180):** Sensor de presión con conexión i2c.

  Tipo: Datos I2C Pin: A4(SDA) / A5 (SCL) Valor: presión (mb)



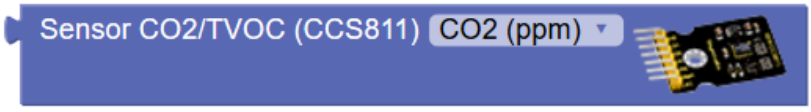
- **Sensor de temperatura (DS18B20):** Sensor con sonda de temperatura, permite varias sondas (#0, #1, …) y utiliza la conexión 1-wire

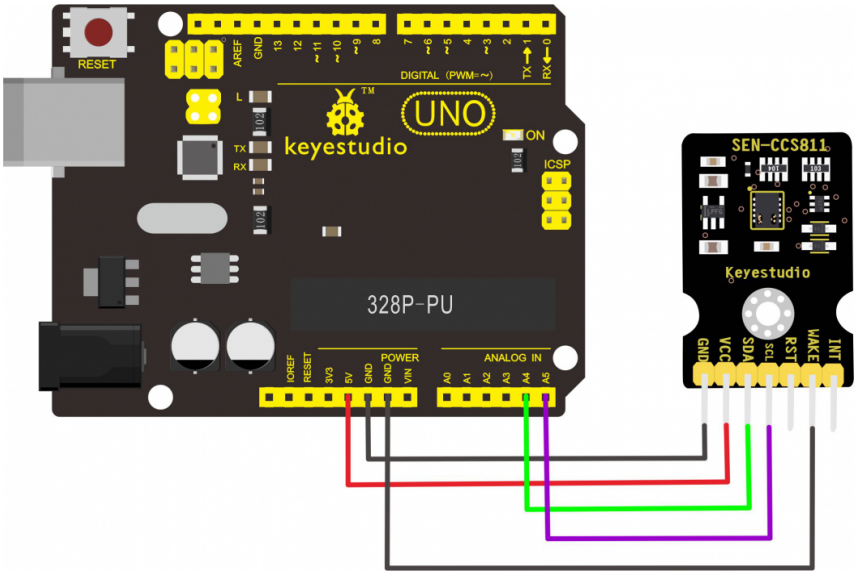
  Tipo: Datos 1-wire Pin: Digital Valor: Temperatura ºC



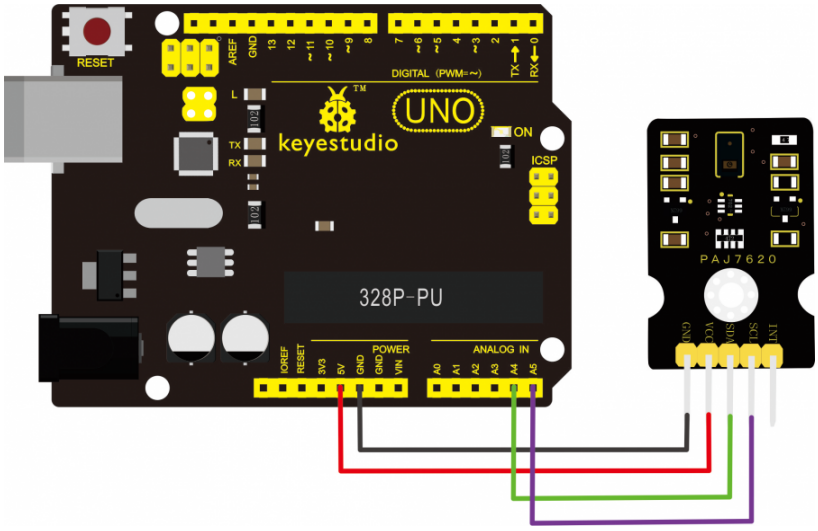
- **Sensor de CO2/TVOC (CCS811):** Sensor de calidad del aire. Obtiene el valor de Compuestos Volatiles Organicos (TVOC) y de Dioxido de Carbono Equivalente (eCO2)

  Tipo: Datos I2C Pin: A4(SDA) / A5 (SCL) Valor: CO2 (ppm)





- **Sensor de gestos (PAJ7620):** Este sensor identifica hasta 9 gestos que se realizan con la mano cerca del sensor: izquierda, derecha, arriba, abajo, giro horario, antihorario y ola.



Ejemplo: detección de varios gestos básicos

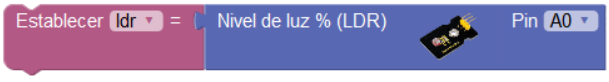
Uso de otros sensores:

ArduinoBlocks implementa los sensores vistos anteriormente para simplificar el uso. En algunos casos procesa los datos leídos para obtener un valor (por ejemplo el sensor de temperatura NTC o LM35) y en otros casos "normaliza" el valor a un rango de % (0 a 100) para simplificar su uso.

En algunos casos puede que necesitemos leer el valor del sensor directamente para obtener una mayor precisión o simplemente porque el sensor no está implementado en ArduinoBlocks y necesitamos usar los bloques de entrada/salida genéricos para obtener datos del sensor digital o analógico.

Ejemplo 1: Lectura del valor de luz con LDR para obtener más precisión.

Sensor de luz:
valor del sensor en % (de 0 a 100)

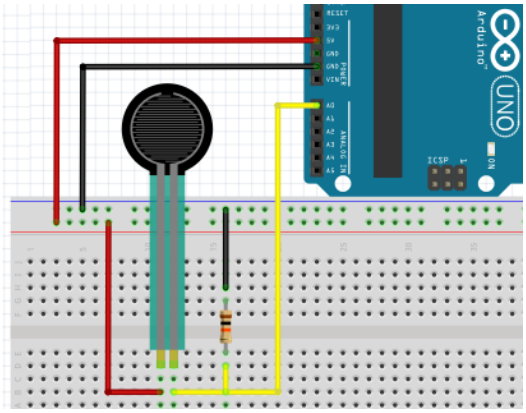


Valor del sensor directamente de la entrada analógica (valor de 0 a 1023)

Ejemplo 2: sensor de presión analógico.

Este sensor modificar su resistencia en función de la presión que se ejerce sobre él y por tanto conectado a una entrada analógica variará el voltaje leído en ella. La fuerza aplicada se traduce en un valor analógico leído de 0 a 1023. En estos casos debemos consultar las especificaciones del fabricante para interpretar el dato obtenido.

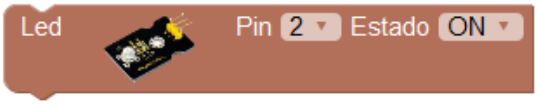
Force (lb)	Force (N)	FSR Resistance	(FSR + R) ohm	Current thru FSR+R	Voltage across R
None	None	Infinite	Infinite!	0 mA	0V
0.04 lb	0.2 N	30 Kohm	40 Kohm	0.13 mA	1.3 V
0.22 lb	1 N	6 Kohm	16 Kohm	0.31 mA	3.1 V
2.2 lb	10 N	1 Kohm	11 Kohm	0.45 mA	4.5 V
22 lb	100 N	250 ohm	10.25 Kohm	0.49 mA	4.9 V



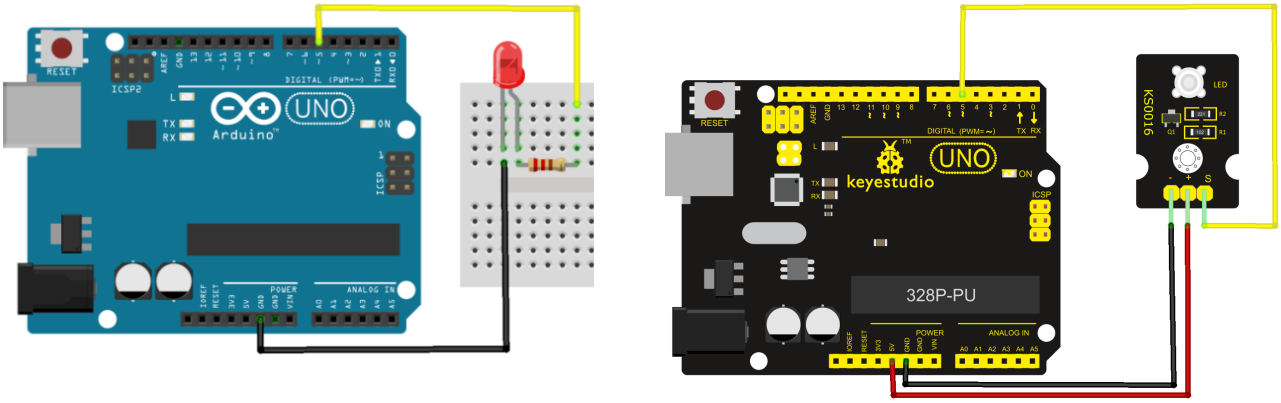
3.3.6 Actuadores

- **Led:** Permite controlar el encendido/apagado de un led (diodo emisor de luz).

  Tipo: Digital Pin: 2-13 Valor: 0/1 (F/V, Off/On)



Ejemplo: Parpadeo de un led cada segundo. Led conectado al pin 5:



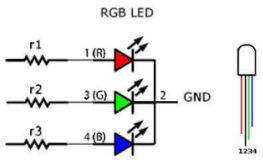
- **Led intensidad (PWM):** Permite controlar la intensidad de iluminación de un led conectado a una salida PWM.

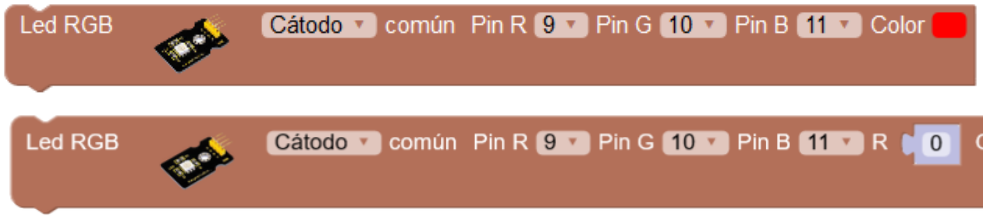
  Tipo: PWM Pin: 3,5,6,9,10,11 Valor: 0-255

Ejemplo: Aumento progresivo de la intensidad del led:

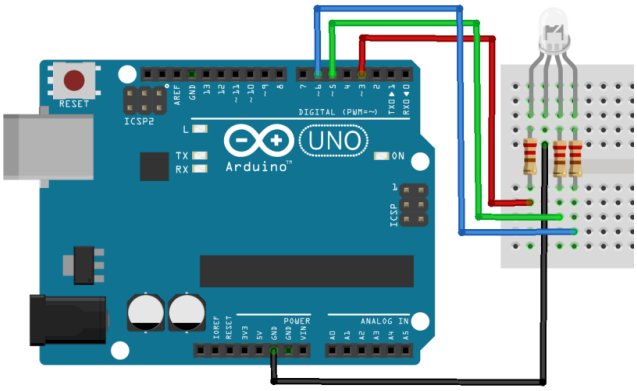
- **Led RGB:** Controla un led RGB. Define un color calculando automáticamente los valores de cada componente R,G y B para definir el color.

Tipo: PWM
Pin R/G/B: 3,5,6,9,10,11
Valor: Color

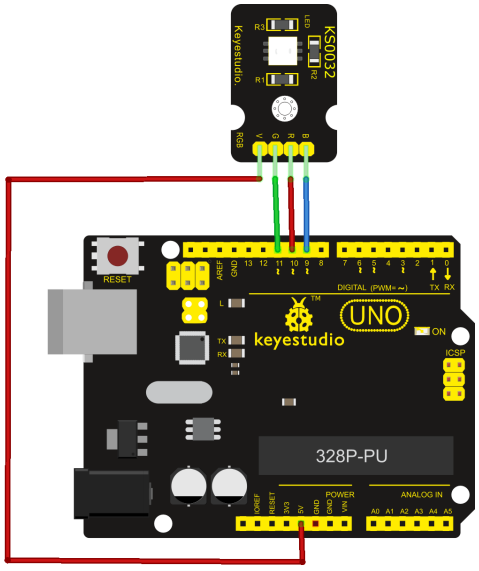




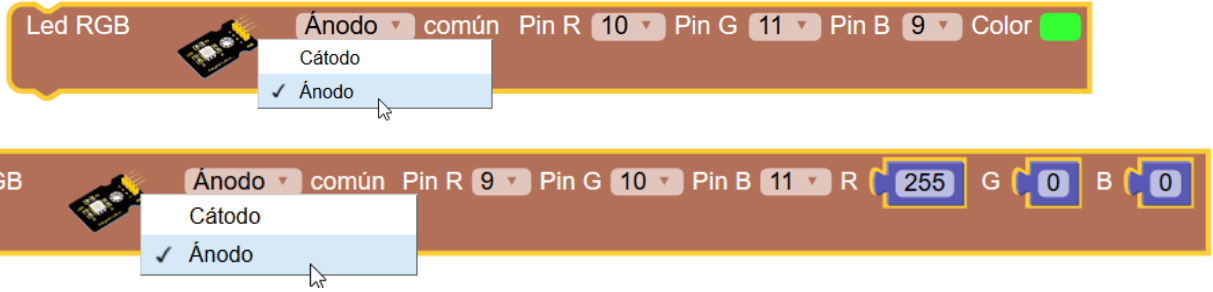
Ejemplo de conexión de un led RGB cátodo (-) común



Ejemplo de conexión de un módulo led RGB ánodo (+) común,

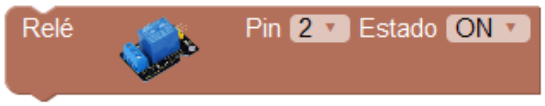


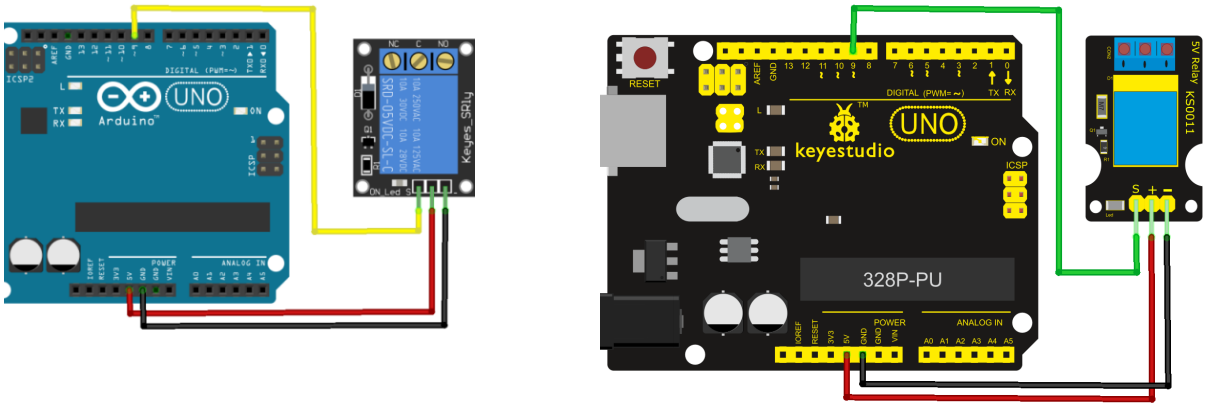
Cada color se activa mediante un valor de 0 (mínimo) a 255 (máximo), en caso de especificar el tipo "ánodo común" el valor se invertirá internamente automáticamente (255 - valor especificado).



- **Relé:** Controla la activación de un relé.

  Tipo: Digital Pin: 2-13 Valor: 0/1 (F/V, Off/On)



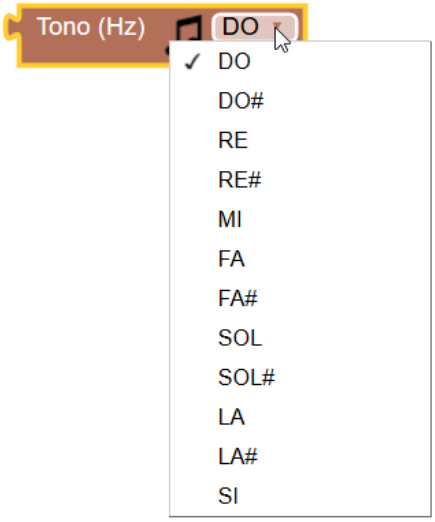


Ejemplo: Activación/desactivación de un relé cada 5 segundos:

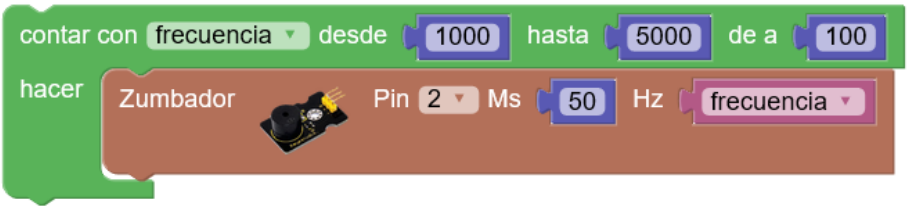
- **Zumbador (pasivo):** Un zumbador pasivo es un dispositivo piezoeléctrico que permite generar sonidos. Podemos generar tonos de la frecuencia deseada. (Si utilizamos un zumbador activo el propio zumbador genera su frecuencia y lo podremos activar o desactivar con una simple salida digital).

  Tipo: Digital Pin: 2-13 Hz: frecuencia del tono / Ms: duración en mseg.

Mediante el bloque de tonos predefinidos podemos indicar las notas musicales en lugar de tener que indicar el valor de la frecuancia de forma numérica.

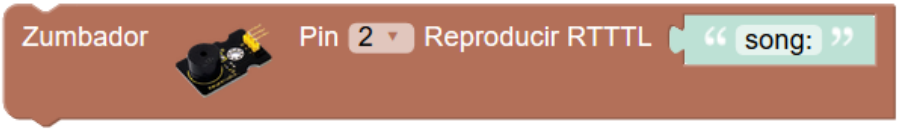


Ejemplo: reproducir tono desde 1000 hz hasta 5000 hz en pasos de 100hz con una duración de 50ms para cada tono



Ejemplo: Tonos de medio segundo. Zumbador conectado al pin 3:

- Zumbador melodía RTTTL: Este bloque permite reproducir una melodía a partir de un texto con formato RTTTL.



El formato RTTTL es un formato creado por Nokia para las primeras melodías de los teléfonos móviles. Existen sitios webs donde descargar estas melodías.

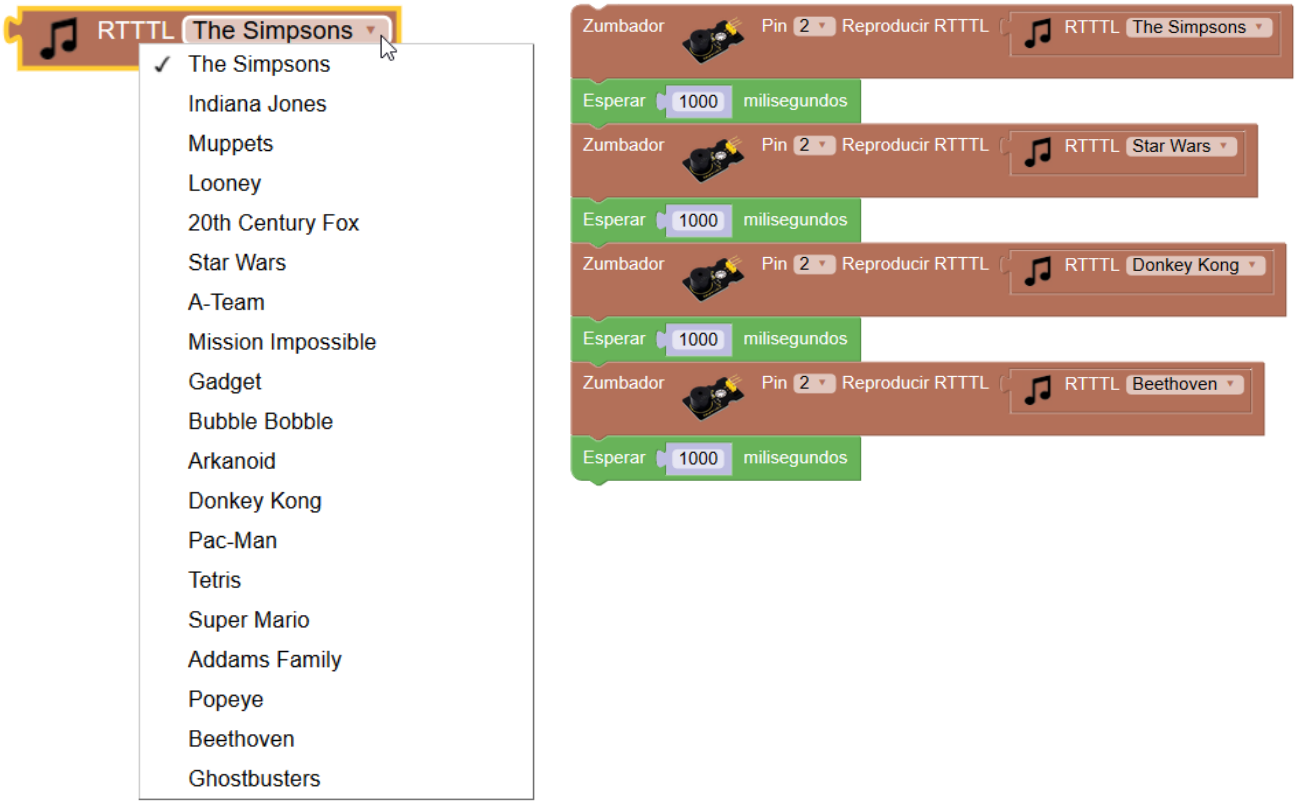
Ejemplo: melodía Pac-Man en formato RTTTL

pacman:d=4,o=5,b=112:32b,32p,32b6,32p,32f#6,32p,32d#6,32p,32b6,32f#6,16p,16d#6,16p,32c6,32p,32c7,32p,32g6,32p,32e6,32p,32c7,32g6,16p,16e6,16p,32b,32p,32b6,32p,32f#6,32p,32d#6,32p,32b6,32f#6,16p,16d#6,16p,32d#6,32e6,32f6,32p,32f6,32f#6,32g6,32p,32g6,32g#6,32a6,32p,32b.

Información sobre el formato RTTTL:
https://en.wikipedia.org/wiki/Ring_Tone_Transfer_Language

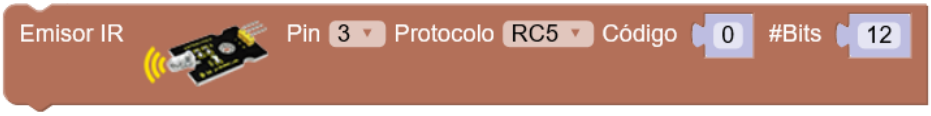
Descarga de melodías RTTTL:
http://arcadetones.emuunlim.com/

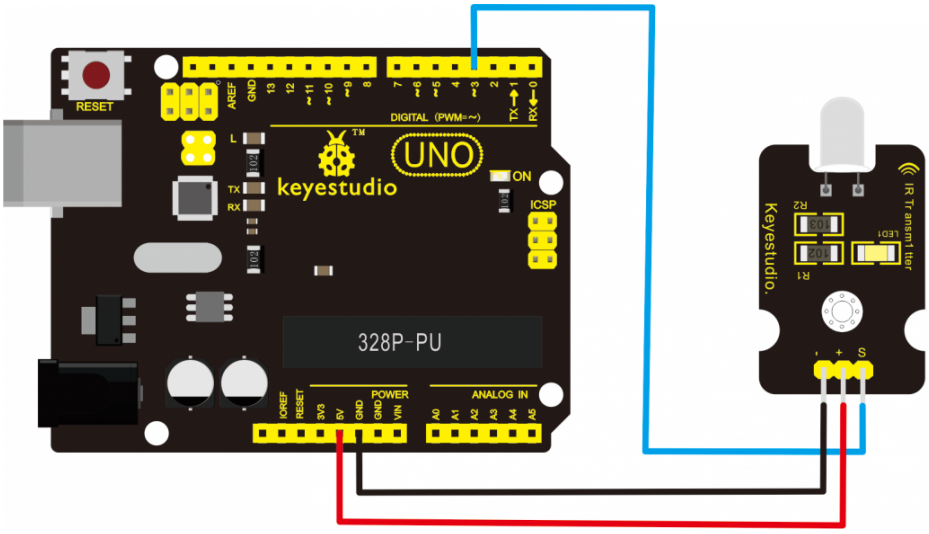
El bloque de melodías RTTTL predefinidas permite reproducir algunas melodías de forma sencilla.



- **Emisor IR:** Mediante el módulo emisor IR podemos emitir códigos de infrarojos como un mando a distancia.

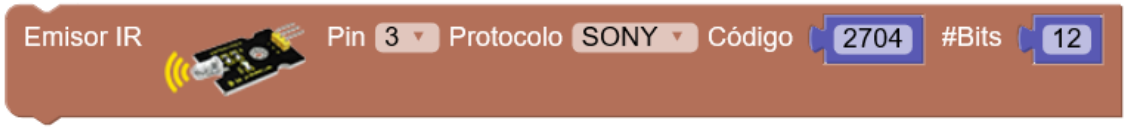
  Tipo: Digital Pin: 3 Protocolo y código IR a enviar





Ejemplo: enviar el código de Sony para encender/apagar la TV

Código: 0xA90 (decimal: 2704)



Para obtener la información de un mando IR cualquiera y decodificar el protocolo, número de bits y código de cada tecla podemos seguir este tutorial (tutorial con código avanzado en Arduino IDE)

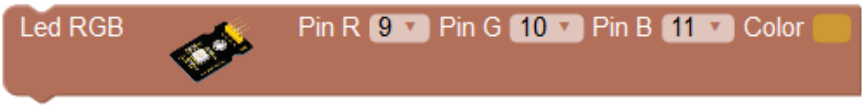
https://learn.adafruit.com/using-an-infrared-library/hardware-needed

Uso de otros actuadores:

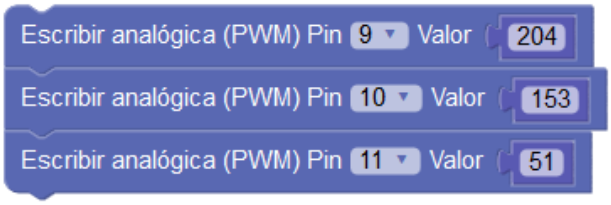
Al igual que con los sensores, puede que tengamos actuadores digitales o analógicos no implementados en ArduinoBlocks o que queramos controlar de una forma diferente a como lo hacen los bloques. Para ello podemos utilizar directamente los bloques de entrada/salida genéricos.

Ejemplo 1: Control de un led RGB con salidas analógicas PWM:

Con el bloque específico:

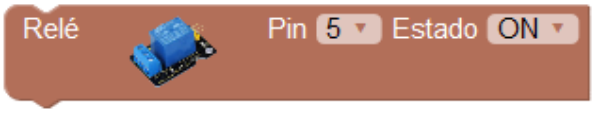


Con los bloques genéricos (Naranja => Rojo = 204, Verde=153, Azul=51)



Ejemplo 2: Control de un relé con salida digital

Con bloque específico:



Con bloque genérico

Ejemplo 3: Control de un led con salida digital

Con bloque específico:

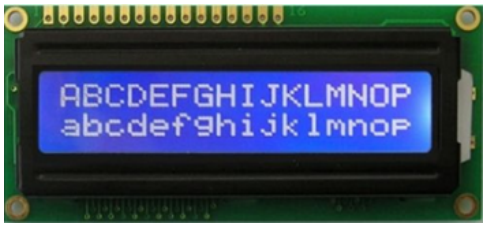
Con bloque genérico

3.3.7 Pantalla LCD

Uno de los periféricos más utilizados y versátil que podemos conectar a Arduino es una pantalla LCD (display) para mostrar información e interactuar con el usuario.

La pantalla LCD más sencilla y utilizada es el de tipo alfanumérico de 2 líneas y 16 caracteres por línea, o de 4 líneas y 20 caracteres por línea

LCD 2x16

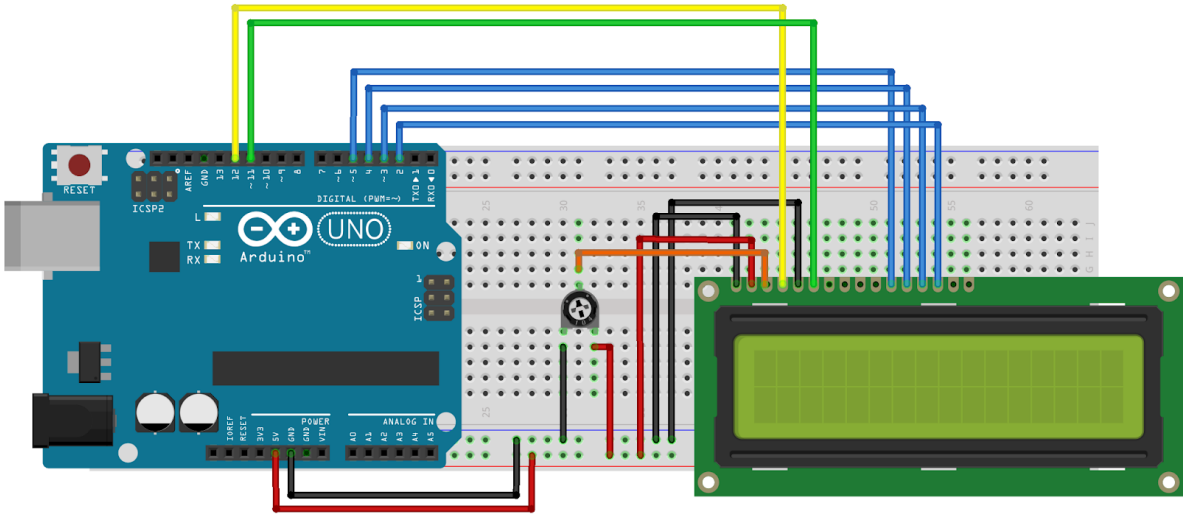


LCD 4x20

ArduinoBlocks nos permite conectar una pantalla de dos forma diferentes:

- **Conexión con bus de 4 bits + control EN / RS:**

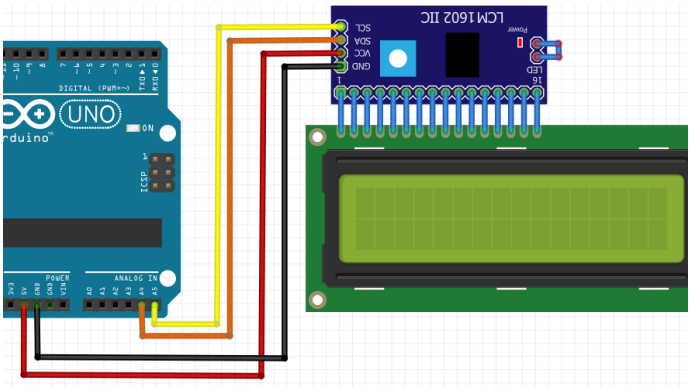
Necesitamos 4 bits para datos y dos señales de control En (Enable) y Rs (Register select). La conexión RW la conectamos fija a GND. Además se debe añadir una resistencia ajustable o un potenciómetro para regular el contraste de la pantalla.

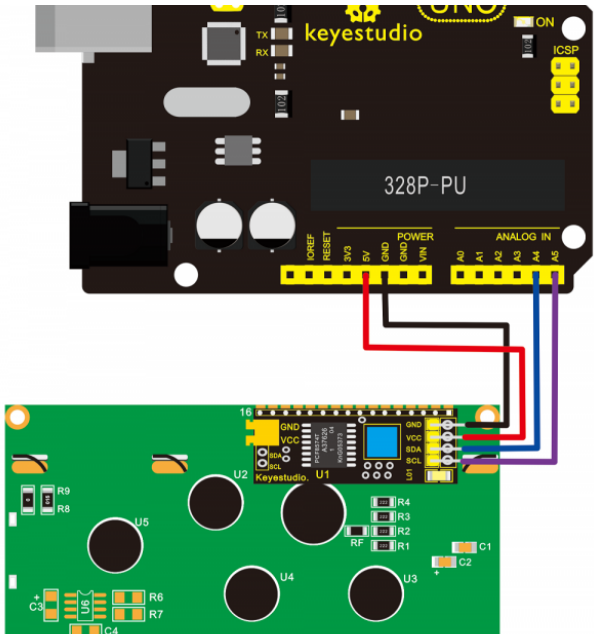


Esta conexión es bastante compleja y utiliza muchos pines, es muy recomendable utilizar la pantalla con conexión i2c. A cualquier pantalla LCD le podemos añadir un módulo para convertirla en conexión i2c.

- **Conexión por bus de comunicaciones I2C:**

Es la forma más sencilla, necesitamos una pantalla con interfaz I2C o un módulo adaptador que realiza todo el trabajo.

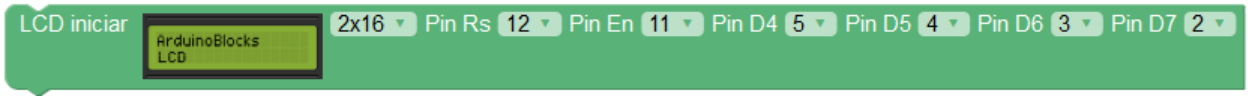




SDA = Arduino Pin A4
SCL = Arduino Pin A5

- **LCD iniciar**: Permite configurar la forma de conexión de la pantalla LCD a la placa Arduino. Recomendable en el bloque “inicializar”.

-Iniciar con conexión de 4 bits:

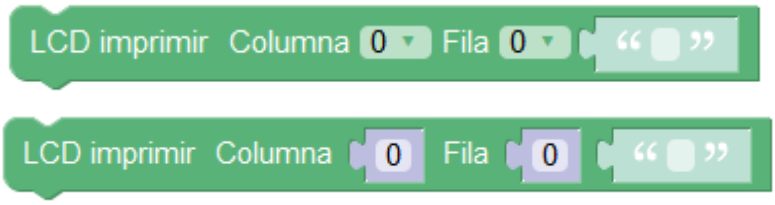


-Iniciar con conexión I2C:

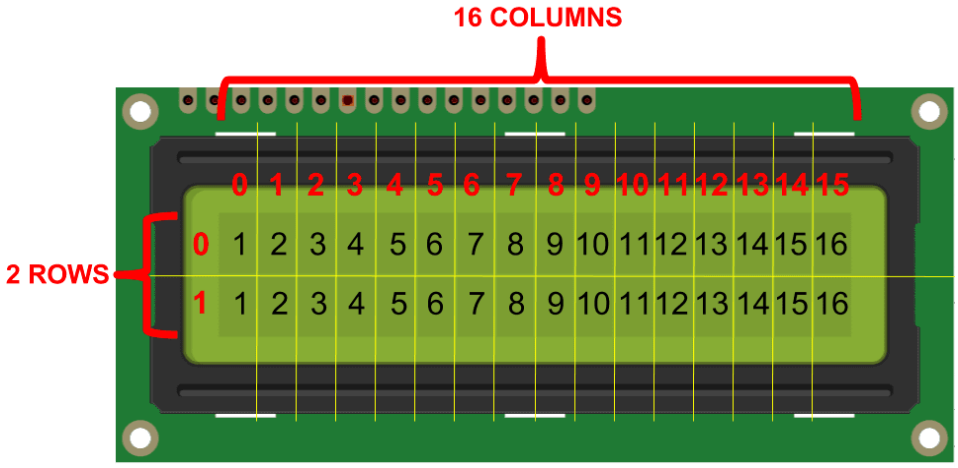
(La dirección I2C depende del módulo o pantalla LCD, 0x27 y 0x3F son las más comunes)

- **LCD limpiar:** Borra el contenido de toda la pantalla LCD

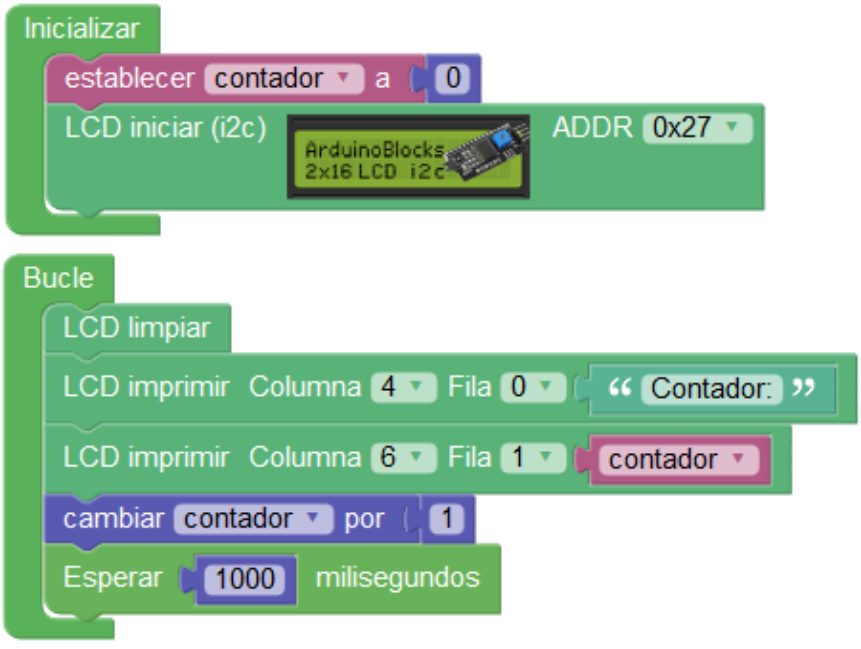
- **LCD imprimir:** Imprime un texto o variable en la fila y columna seleccionada dentro de la pantalla.



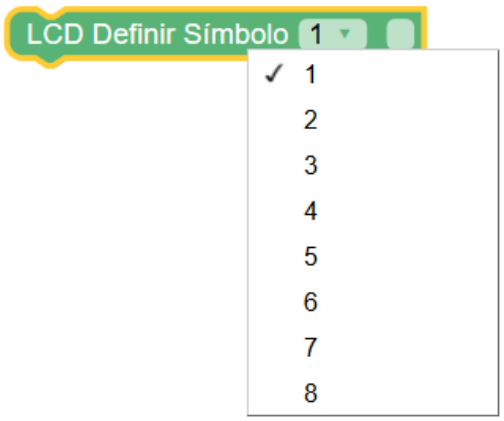
Sistema de filas y columnas en una pantalla LCD de 2x16:



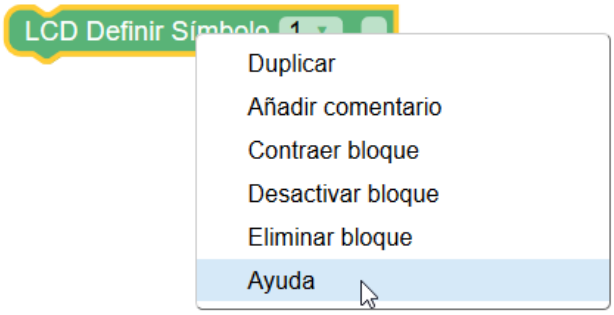
Ejemplo: Contador en pantalla LCD con conexión I2C

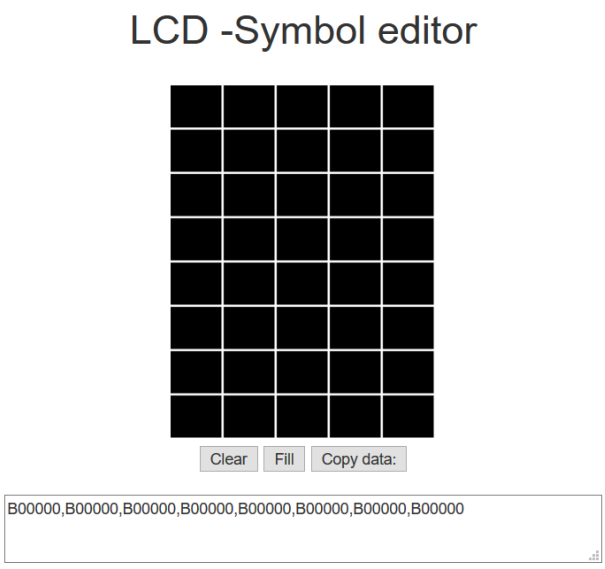


- **LCD definir símbolo:** Permite definir uno de los 8 símbolos personalizables que puede almacenar la pantalla LCD. El símbolo se define por un mapa de bits (1's y 0's indicando cada píxel del símbolo). Los símbolos tienen una resolución de 5x8 píxeles (blanco o negro).



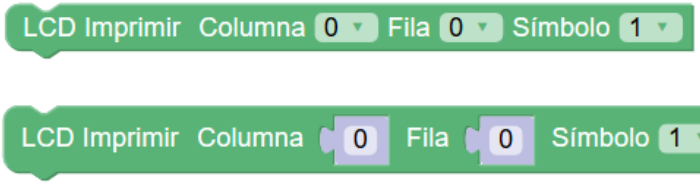
Mediante click derecho y la opción ayuda se abre el editor de símbolos





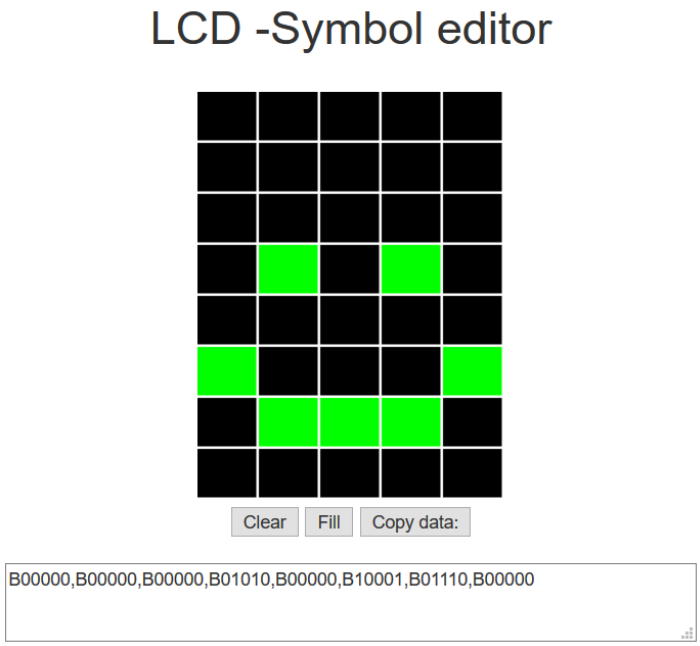
El bloque definir símbolo normalmente irá dentro del bloque "inicializar" de Arduino.

- **LCD imprimir símbolo :** Imprime un símbolo (definido anteriormente) en la fila y columna seleccionada dentro de la pantalla.



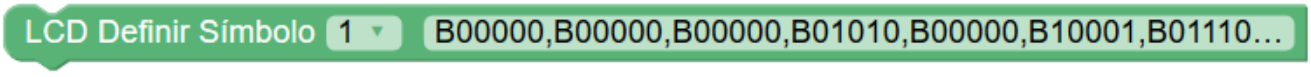
Ejemplo: definir un símbolo con una cara sonriente

e imprimirlo en el centro de la primar fila de la pantalla LCD



Copiamos el código del mapa de bits:

B00000,B00000,B00000,B01010,B00000,B10001,B01110,B00000

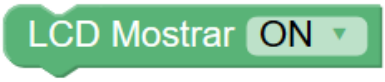


Y usamos el bloque imprimir símbolo indicando el símbolo (1 al 8) y la posición:

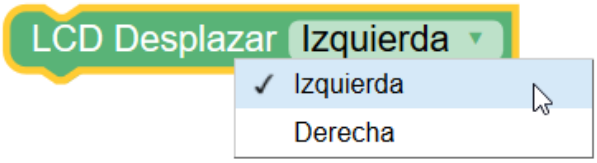
- **LCD luz de fondo:** Activa o desactiva la luz de fondo de la pantalla LCD

- **LCD cursor:** Configura la visualización o no del cursor y si parpadea o no.

- **LCD mostrar:** Activa o no la visualización del LCD sin alterar su contenido.



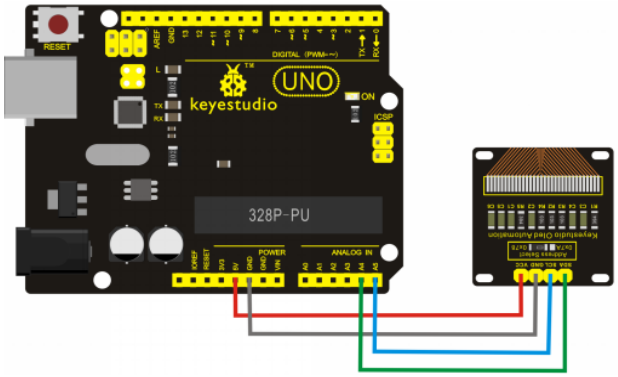
- **LCD desplazar:** Realiza el desplazamiento (scroll) a la izquierda o a la derecha del contenido de la pantalla. Al limpiar la pantalla se reinicia el desplazamiento.



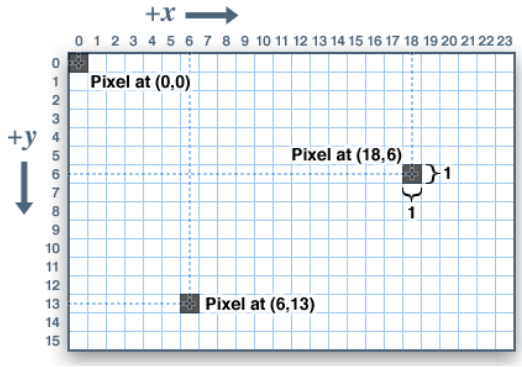
3.3.8 Pantalla OLED

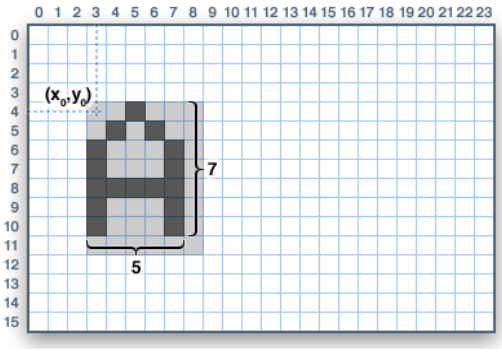
Las pantallas OLED (leds orgánicos) son muy comunes en la actualidad, y las podemos incluir en nuestros proyectos con Arduino de una forma bastante sencilla.

La pantalla más habitual para Arduino y la soportada por ArduinoBlocks es la pantalla monocromo de 0.96″ (pequeña pero matona!) con conexión i2c.



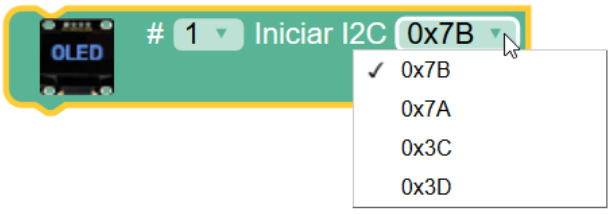
Este tipo de pantalla tiene un tamaño de 128×64 píxeles (Ancho x Alto).



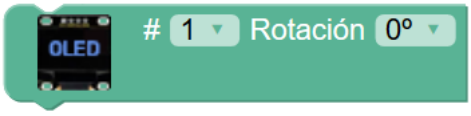


http://shop.innovadidactic.com/index.php?id_product=751&controller=product

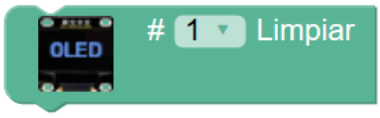
- **OLED iniciar:** Indicamos el número de pantalla (#), pues podemos usar varias a la vez, indicando la dirección a la que está conectada. Si queremos más de una pantalla en el bus i2c y controlarlas de forma independiente deben tener direcciones distintas, por lo que debemos modificar los pines de configuración.



- **OLED rotación:** Permite indicar en qué posición está situada la pantalla.

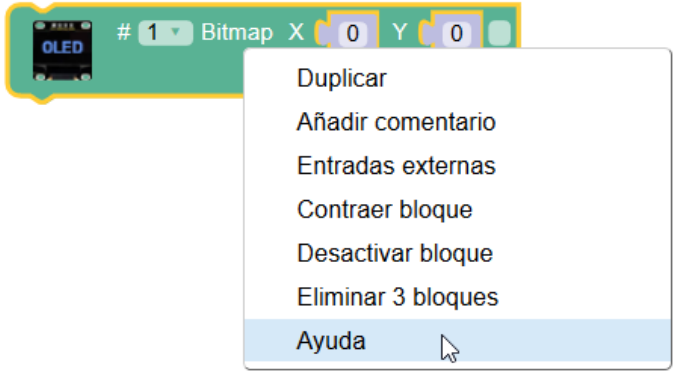


- **OLED limpiar:** Elimina todo el contenido de la pantalla.

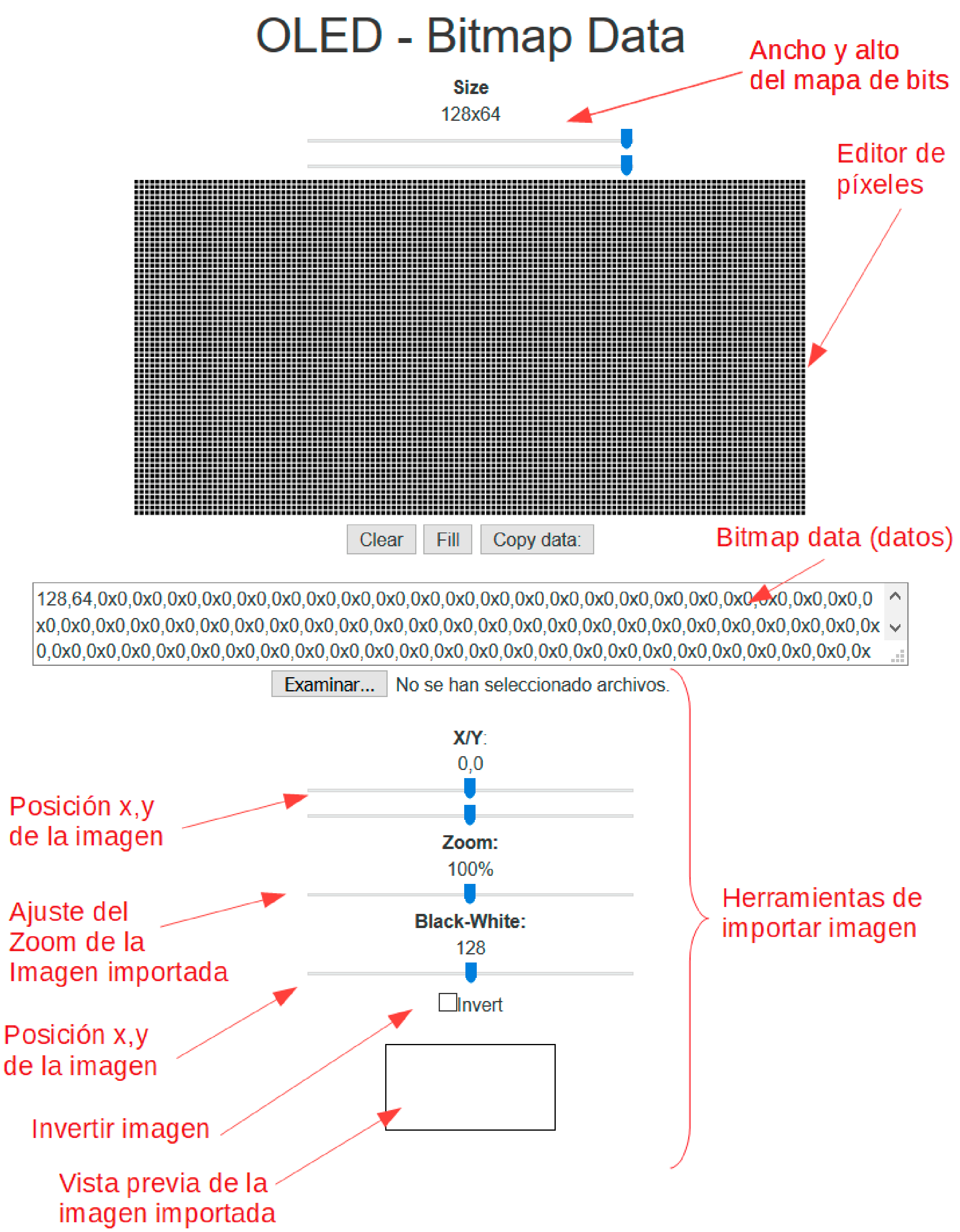


- **OLED texto:** Muestra un texto en la posición indicada. Ya que la pantalla es monocromo, podemos indicar si el texto se muestra en positivo (encendiendo píxeles en la pantalla) o en negativo (apagando píxeles). Además podemos especificar 3 tamaños de pantalla.

- **OLED bitmap:** Permite dibujar una imagen a partir de una mapa de bits. Con la ayuda del editor de mapas de bits podemos procesar una imagen y mostrarla en la pantalla OLED de forma bastante sencilla.



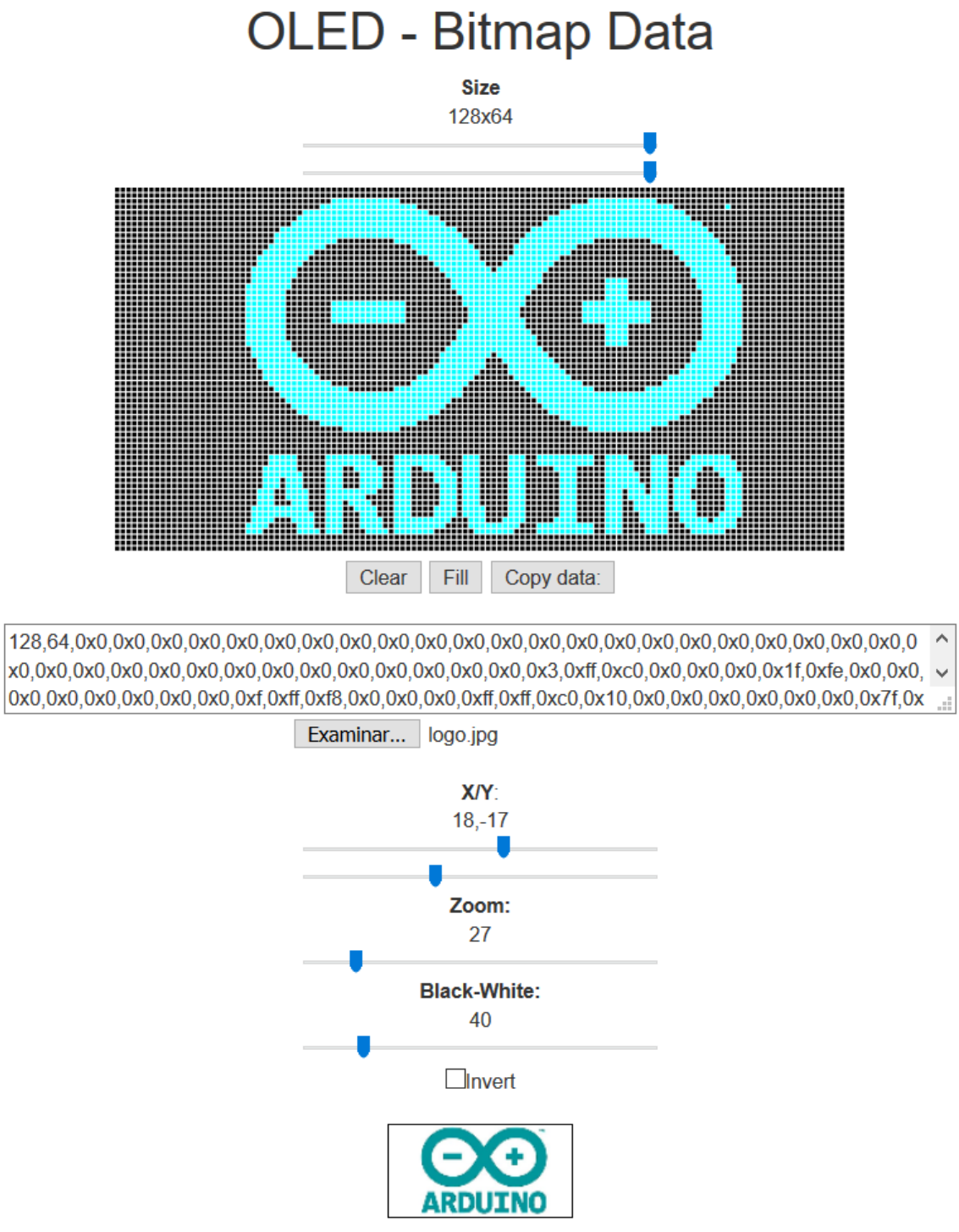
Editor de mapa de bits:



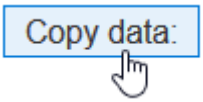
Ejemplo: procesar una imagen con el editor de mapa de bits

y mostrarla en la pantalla OLED

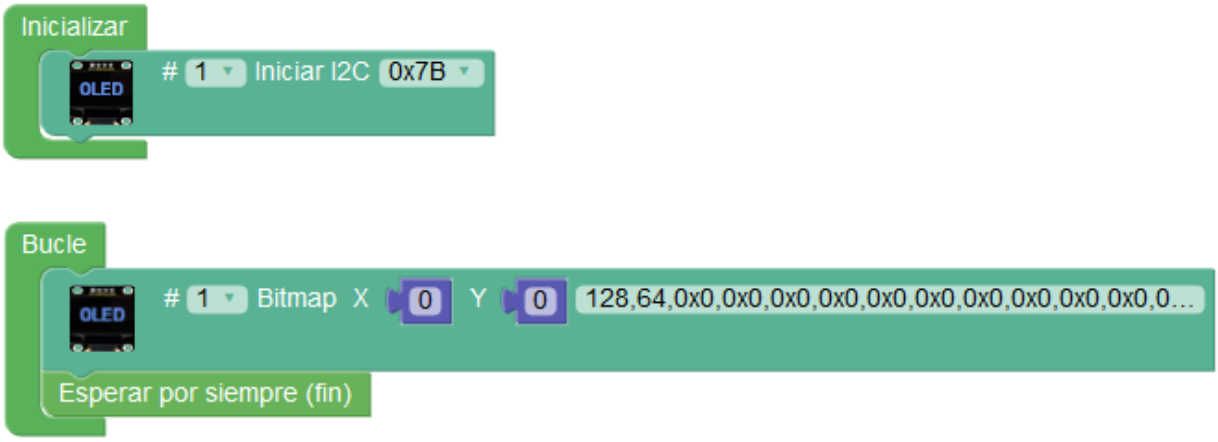
Ajuste de parámetros en el editor de bitmaps:



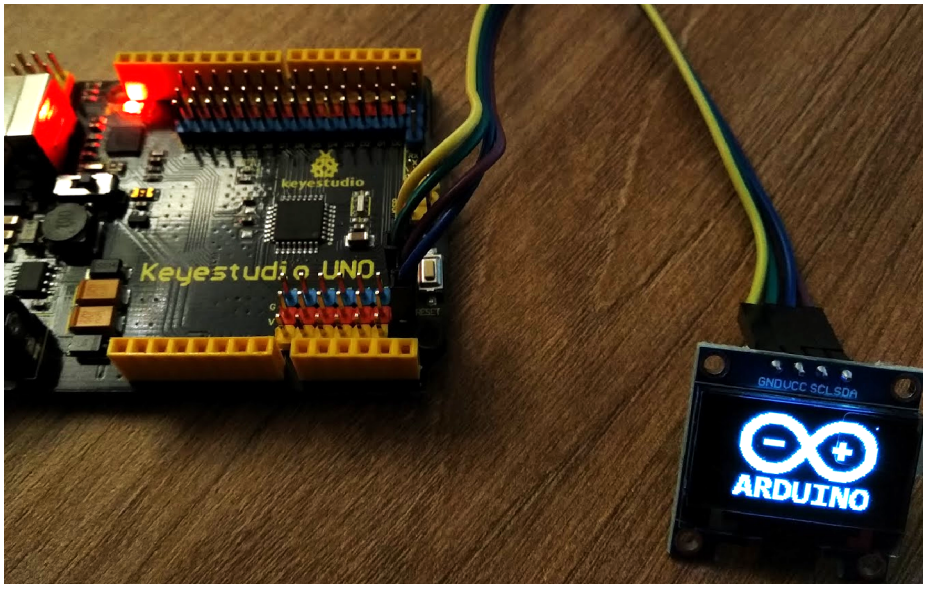
Copiar los datos obtenidos:



Programa de ejemplo con los datos anteriores para mostrar la imagen:



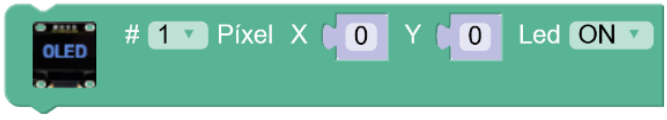
Resultado:



+Información sobre gráficos y animaciones en pantallas OLED:

http://arduinoblocks.didactronica.com/2019/05/graficos-y-animaciones-en-pantallas/

- **OLED píxel:** Dibuja un píxel en las coordenadas especificadas.



- **OLED línea:** Dibuja un línea recta entre dos coordenadas especificadas.

  Origen: X1, Y1

  Destino: X2,Y2

- **OLED rectángulo:** Dibuja un rectángulo indicando en las coordenadas especificadas

  Origen: X1, Y1

  Ancho: W

  Alto: H

  Podemos seleccionar si queremos que el rectángulo esté relleno o sólo se dibuje el borde.

- **OLED círculo:** Dibuja un píxel en las coordenadas especificadas

  Centro: X,Y

  Radio: R

  Podemos seleccionar si queremos que el círculo esté relleno o sólo se dibuje el contorno.

3.3.9 Memoria EEPROM

La memoria EEPROM es un memoria interna del microcontrolador de Arduino que nos permite guardar información. Tiene la propiedad de no ser volátil, por lo que la información permanece guardada en ella aunque quitemos la alimentación eléctrica.

Esta memoria es perfecta para almacenar información de configuración de la aplicación o valores de estado que se necesiten recuperar después de un corte de la alimentación eléctrica.

El microcontrolador de la placa Arduino UNO tiene 1024 bytes de memoria EEPROM, sin embargo en ArduinoBlocks cada variable usada internamente utiliza 4 bytes por lo que a la hora de almacenar o recuperar una variable de la memoria EEPROM sólo podemos almacenar en 256 posiciones (256 x 4 = 1024 bytes).

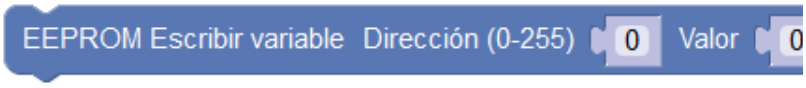
Direccionamiento Arduino: 0-1023

0
1
2
3
4
5
6
7
8
9
10
11
…

Direccionamiento ArduinoBlocks: 0-255

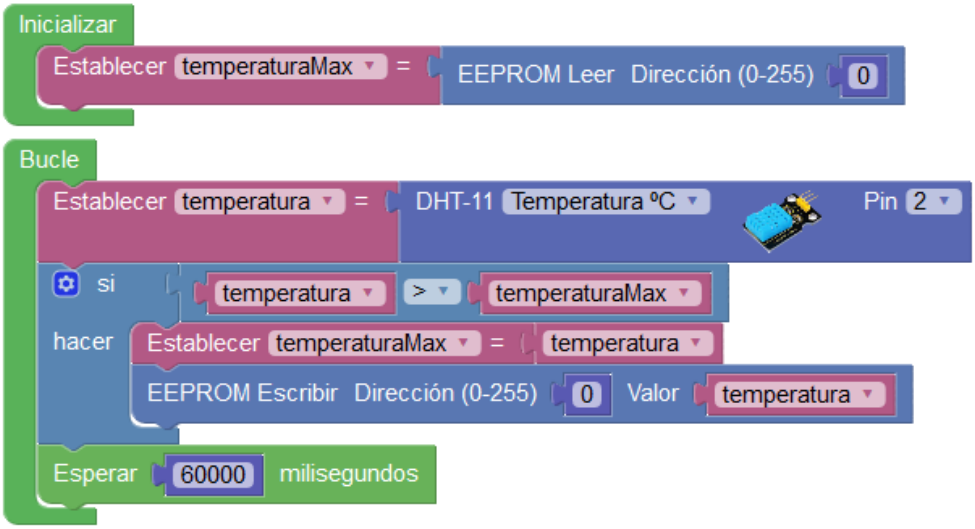
0
1
2
…

- **EEPROM escribir variable:** Guarda un valor o variable en una posición de memoria de la memoria.



- **EEPROM leer variable:** Leer un valor de una posición de la memoria.

Ejemplo: Lee la temperatura cada minuto y guarda la temperatura máxima en la memoria EEPROM para reservarla aunque cortemos la alimentación:



IMPORTANTE:

La memoria EEPROM suele venir inicializada a 0xFF

por lo que para un uso correcto deberíamos ponerla a 0 en algunos casos:

3.3.10 Motores

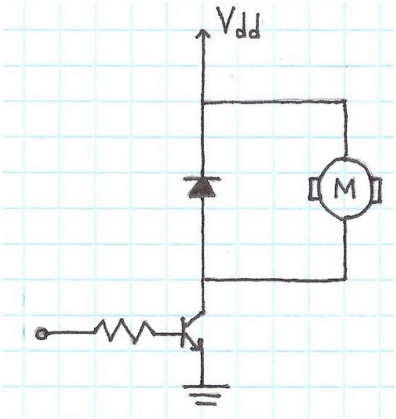
Desde la placa Arduino es fácil controlar varios tipos de motores.

- **Motores de corriente continua (DC / CC)**

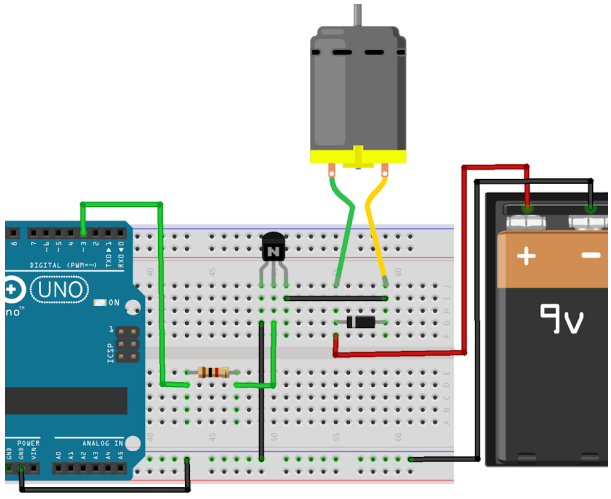
Las salidas de la placa Arduino no proporcionan suficiente corriente para controlar un motor de corriente continua (Arduino proporciona unos 50mA y un motor puede consumir unos 1000mA) por lo que necesitaremos realizar un pequeño circuito con un transistor para controlar una corriente mucho mayor.

Utilizaremos un transistor NPN en modo corte/saturación que permitirá, como un interruptor, el paso de una intensidad de corriente mucho más alta desde un fuente de alimentación auxiliar.

Esquema de conexión:



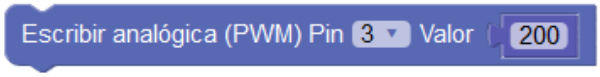
Montaje en placa de prototipos



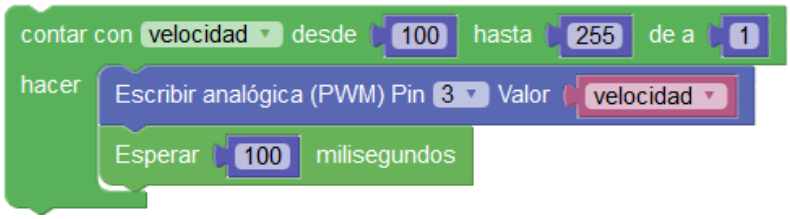
La pila de 9v genera la corriente necesaria para mover el motor. A través del pin 3 generamos la señal que activa el transistor y permite el paso de corriente de la pila. Si utilizamos la salida como PWM podremos controlar la velocidad del motor (si se escribe un valor bajo, menos de 100 aproximadamente, el motor no girará por no aplicarle la suficiente energía)

Activar giro del motor:

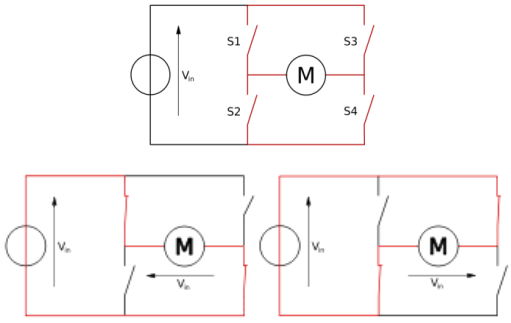
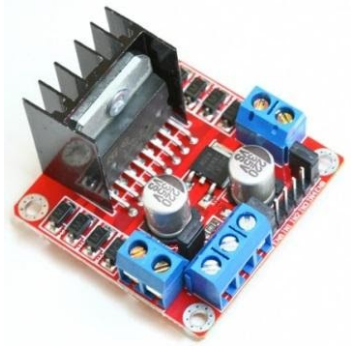
Activar giro controlando velocidad



Ejemplo: Aumento progresivo de la velocidad



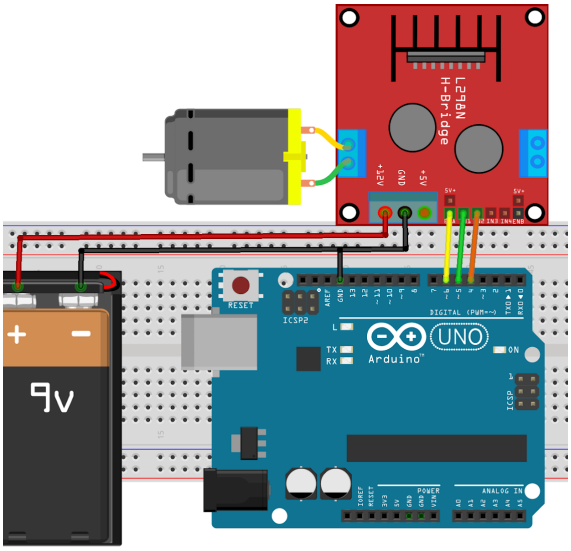
Si necesitamos controlar además el sentido de giro de motor debemos utilizar un “puente en H” que nos permite invertir la polaridad en el motor. Lo más fácil es utilizar un driver integrado como el chip L293D o un módulo para Arduino que integre todos los componentes.

Esquema de un puente en H para controlar la dirección de giro de un motor	*Módulo típico con configuración en puente H para control de motores de C.C.*
	

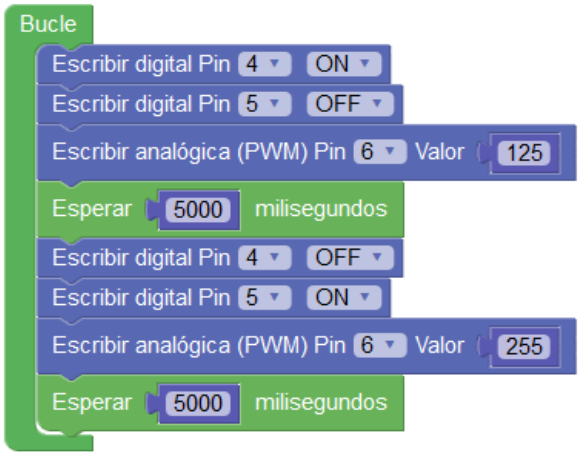
Estos módulos suelen integrar el control para dos motores. Los pines de un módulo de control de motores en puente en H suele tener estas conexiones:

IN1, IN2	Controla el sentido de giro del motor 1	
	IN1 = ON / IN2 = OFF	Giro en un sentido
	IN1 = OFF / IN2 = ON	Giro en sentido contrario
	IN1 = OFF / IN2 = OFF	Parado
IN3, IN4	Controla el sentido de giro del motor 2	
	IN3 = ON / IN4 = OFF	Giro en un sentido
	IN3 = OFF / IN4 = ON	Giro en sentido contrario
	IN3 = OFF / IN4 = OFF	Parado
EN1	Habilita el motor 1 (control de velocidad del motor 1 con PWM)	
EN2	Habilita el motor 2 (control de velocidad del motor 2 con PWM)	

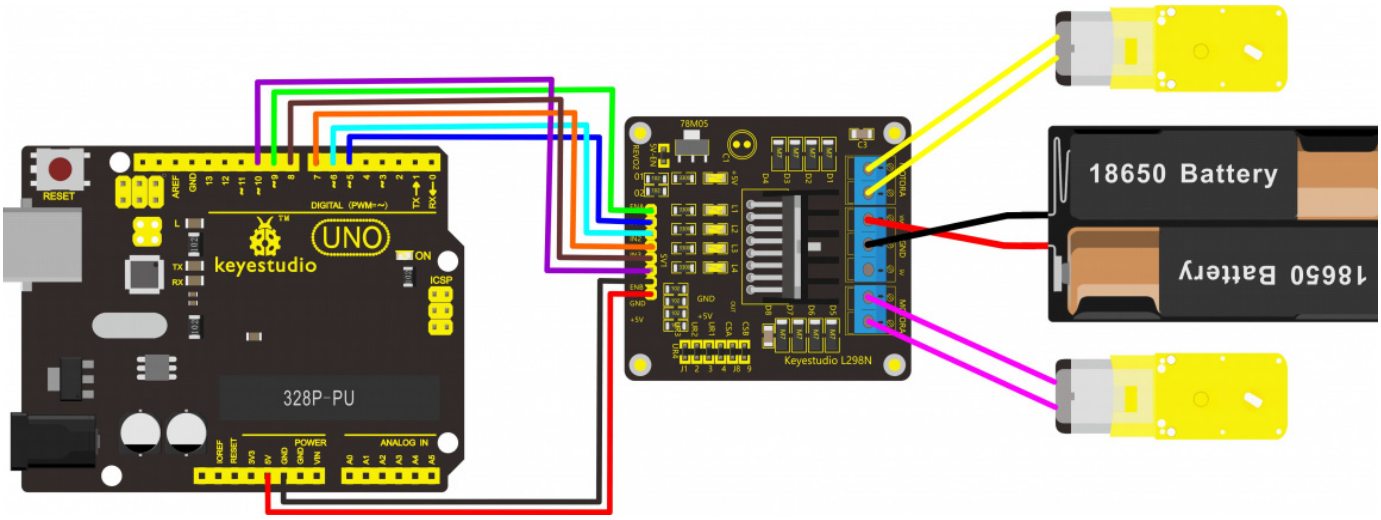
Pines 4,5 (IN1, IN2): control de giro
Pin 6 (EN1): control velocidad PWM



Giro cada 5s en un sentido
con distinta velocidad



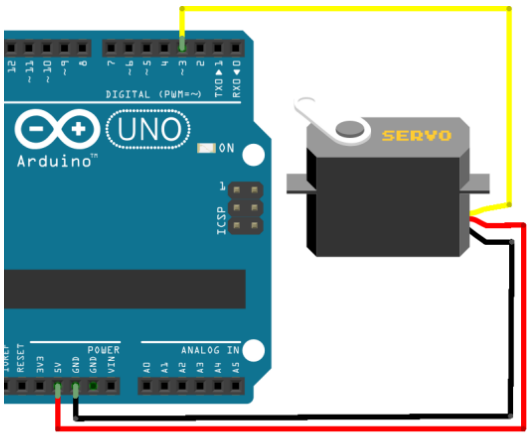
Ejemplo: control de 2 motores C.C. con módulo L298N y alimentación externa



- **Servomotor**

Los servomotores son motores DC a los que se les ha añadido una reductora y una electrónica de control PID que permite controlar el motor situándolo en una posición muy precisa. El servomotor está intentando siempre situarse en la posición indicada, de forma que si se le fuerza o impide ir hasta la posición indicada intentará moverse a la posición indicada continuamente.

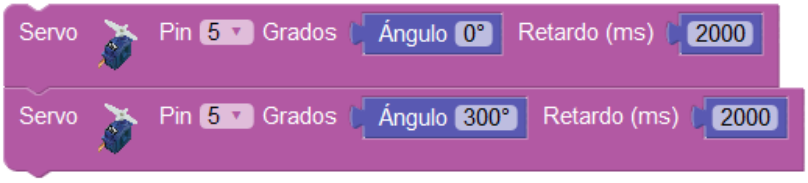
Los servomotores pueden situarse en una posición entre 0º y 300º aproximadamente según el modelo. Un servomotor no permite el giro libre a no ser que se modifique con ese propósito.



El control de la posición de un servomotor se realiza mediante PWM por lo que necesitamos conectarlo a una salida digital de tipo PWM.

En ArduinoBlocks tenemos un bloque que nos permite controlar fácilmente un servomotor indicándole la posición en grados donde queremos que se sitúe y el retardo en milisegundos para darle tiempo a que se mueva hasta la posición indicada.

Ejemplo: movimiento de un servomotor conectado al pin 5:

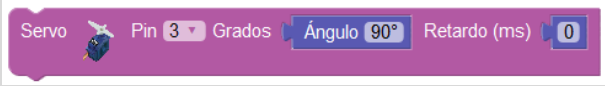
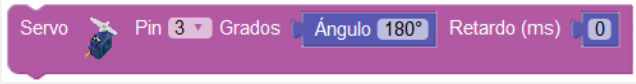


Ejemplo: Mover el servo de 0 a 300 grados de 10 en 10 grados

Existe un tipo especial de servomotor que permite la rotación continua. En algunos casos se trata de servomotores “trucados” de forma que se modifican para permitir la rotación continua quitando los topes mecánicos y se sustituye el potenciómetro por un divisor de tensión con dos resistencia iguales (en algunos casos no se ponen resistencias y se bloquea el potenciómetro para que no gire dejándolo justo en su punto central).

En cualquier caso también podemos comprar un servomotor de rotación continua listo para funcionar sin tener que hacer bricolaje.

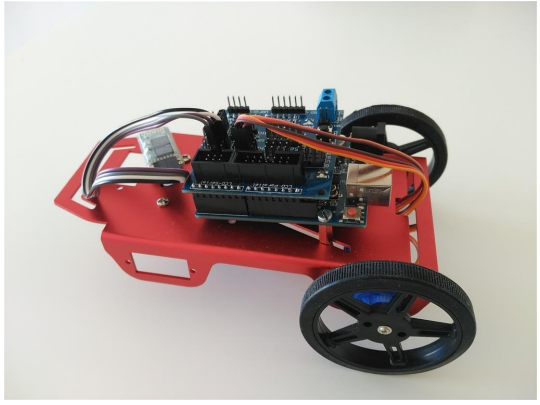
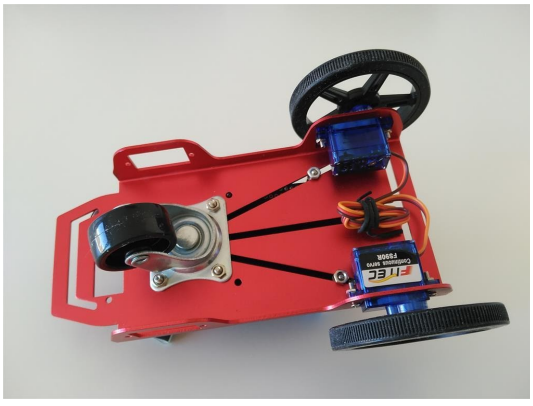
El control de un servomotor de rotación continua se realiza de igual manera, pero su reacción es diferente.

0º		Giro en un sentido (máxima velocidad)
90º		Parado
180º		Giro en sentido contrario (máxima velocidad)

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

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

Ejemplo: robot propulsado por dos servos de rotación continua

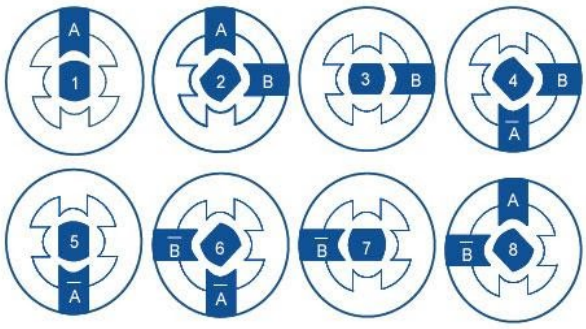
- **Motor paso a paso**

Este tipo de motor es capaz de avanzar una serie de grados (paso) dependiendo de su entrada de control. Son ideales para los mecanismos donde se requiera mucha precisión, por ejemplo son utilizados para los mecanismos de movimiento de las impresoras 3D.

Estos motores están formados por un rotor sobre el que van aplicados varios imanes permanentes y por un cierto número de bobinas excitadoras en su estator.

La excitación de las bobinas se controla externamente y determina el giro del rotor.

Secuencia de activación de las bobinas para giro del motor en una dirección

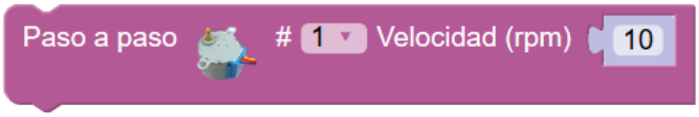


Motor paso a paso y módulo de control

Para controlar un motor paso a paso utilizamos un módulo capaz de controlar cada una de las 4 señales de control que activan cada bobina. Realizando la secuencia correcta movemos el motor, según el número de pasos y velocidad a la que avanzamos en la secuencia. Todo esto se realiza internamente automáticamente.

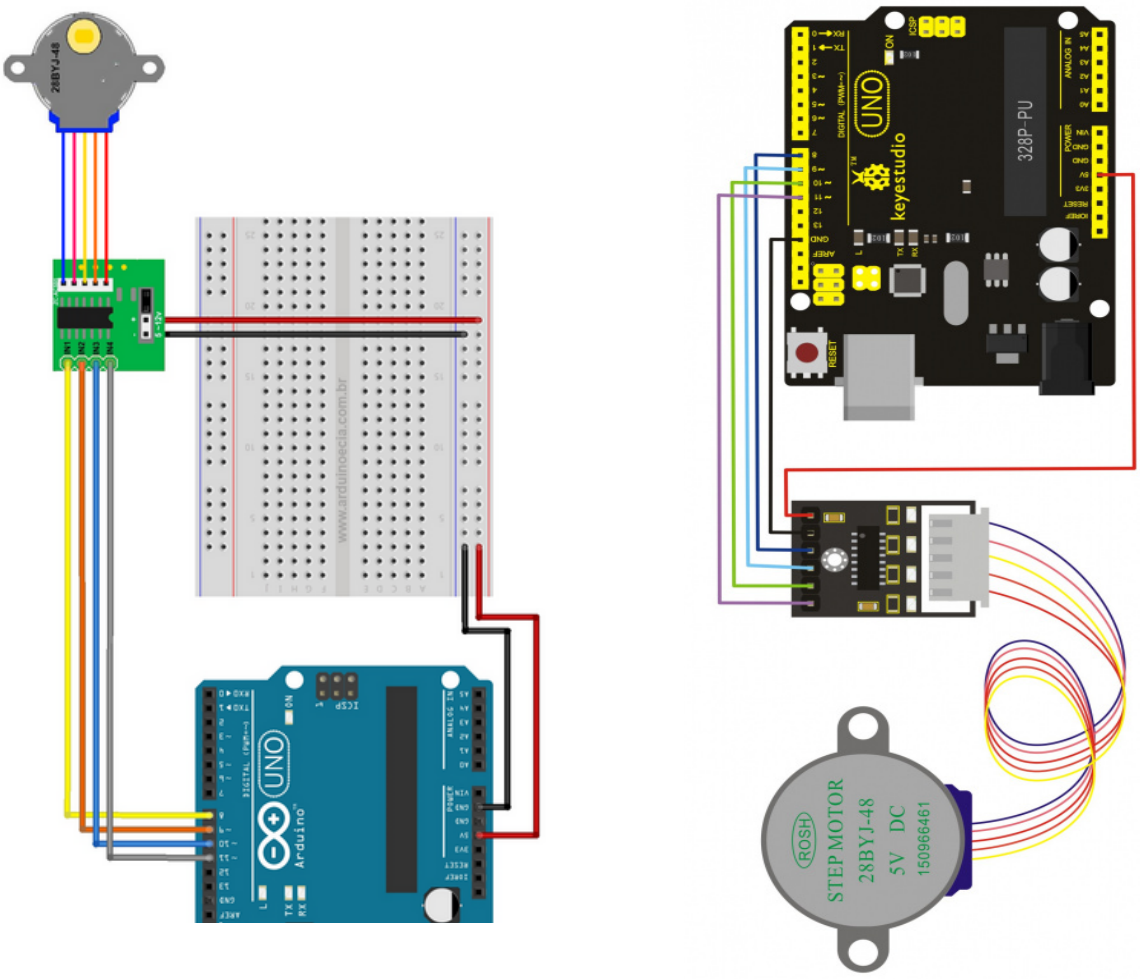
- **Pasos/vuelta:** Configura la conexión de las bobinas del motor paso a paso así como el parámetro de pasos por vuelta para controlar el motor correctamente.

- **Velocidad:** Establece la velocidad de giro del motor en rpm (revoluciones por minuto).

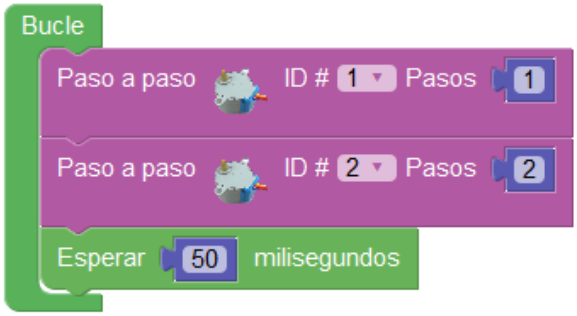


- **Pasos:** Mueve el motor un número de pasos. Si el número de pasos es negativo girará en dirección contraria.

Ejemplo: Conexión de motor paso a paso con módulo controlador



Ejemplo - Control de dos motores paso a paso

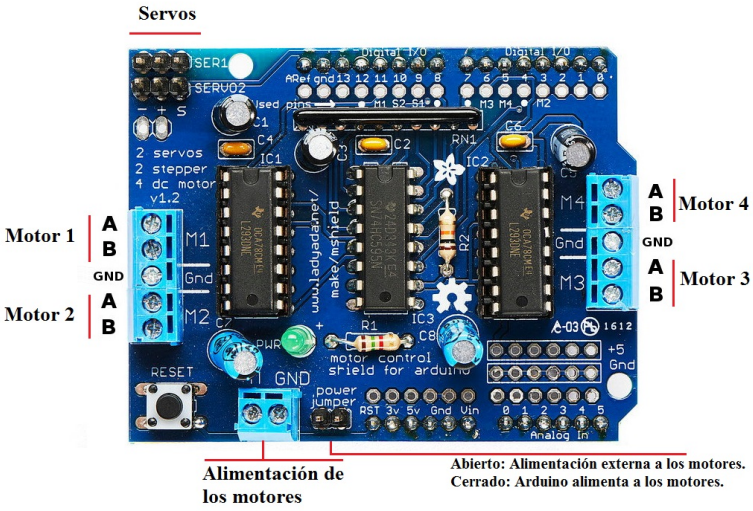


3.3.11 Motor-Shield

Una opción sencilla y muy utilizada para controlar motores de una forma sencilla es utilizar una shield de control de motores. La más utilizada posiblemente sea la motorshield v1 diseñada por Adafruit. Con esta shield podemos controlar 2 servos, 4 motores C.C. y 2 motores paso a paso.

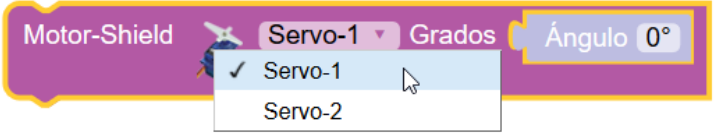
Es importante saber que la shield utiliza todos los pines digitales excepto 0,1,2,13. Y que podemos tener acceso a los pines analógicos A0...A5 que podremos usar como pines digitales o analógicos de E/S y el bus i2c a través de A4,A5 sin problemas.

La shield permite alimentar los motores desde una fuente externa a través de un conector atornillable en la propia placa.

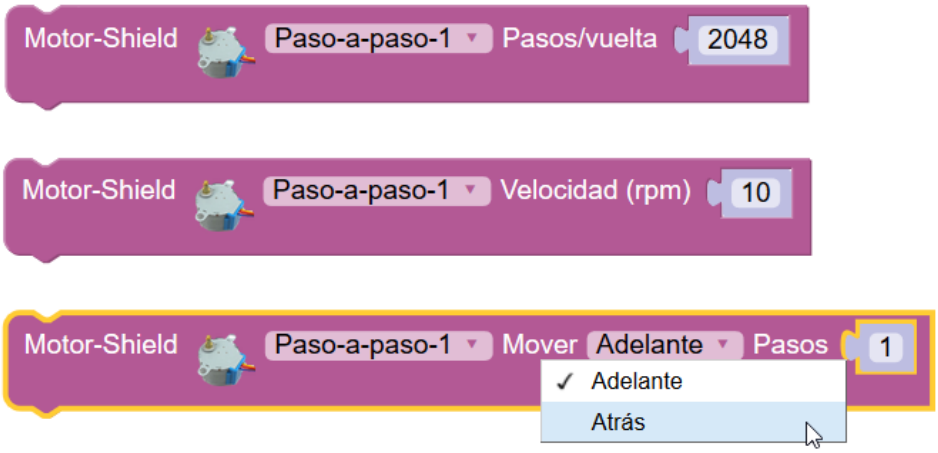


Los bloques son similares a los indicados en el apartado Motores, pero adaptados a la shield.

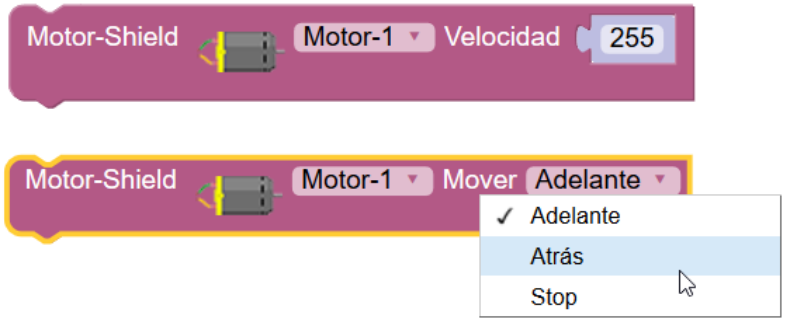
- **Motor-Shield Servo:** Permite controlar uno de los dos servos.



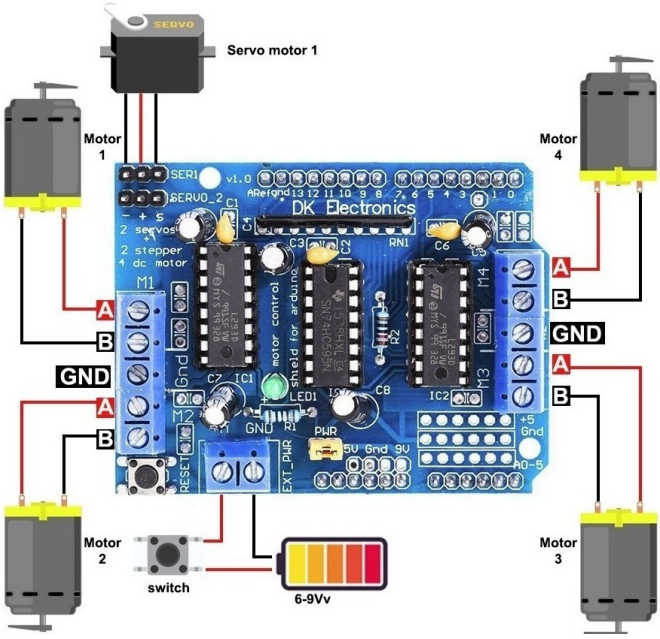
- **Motor-Shield paso a paso:** Permite seleccionar uno de los dos posibles motores paso a paso que se pueden controlar con la shield. El funcionamiento es igual a los bloques indicados en el apartados Motores.



- **Motor-Shield motor (DC / CC):** Permite controlad uno de los 4 motores CC que podemos conectar a la shield. Tensmo un bloque para fijar la velocidad (0 … 255) y otro para controlar el movimiento.

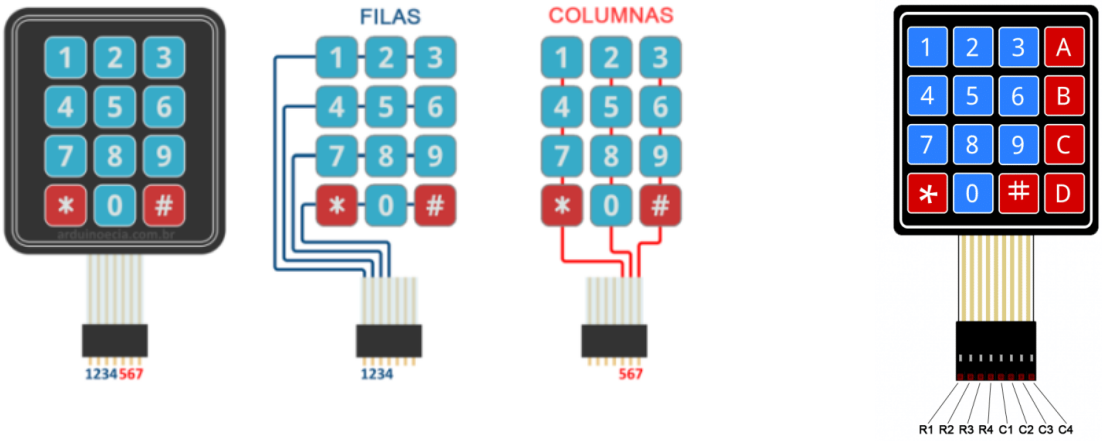


Ejemplo: conexión 4 motores DC y un servo



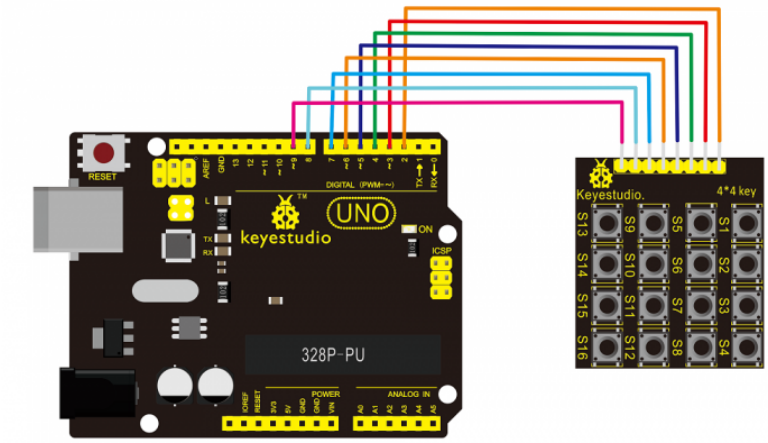
3.3.12 Keypad

El teclado o “keypad” nos permite de una forma sencilla añadir un pequeño teclado numérico a nuestro proyecto. Se basa en una botonera conectada de forma matricial por filas y columnas. ArduinoBlocks gestiona automáticamente la detección de filas y columnas activadas para detectar la tecla pulsada.



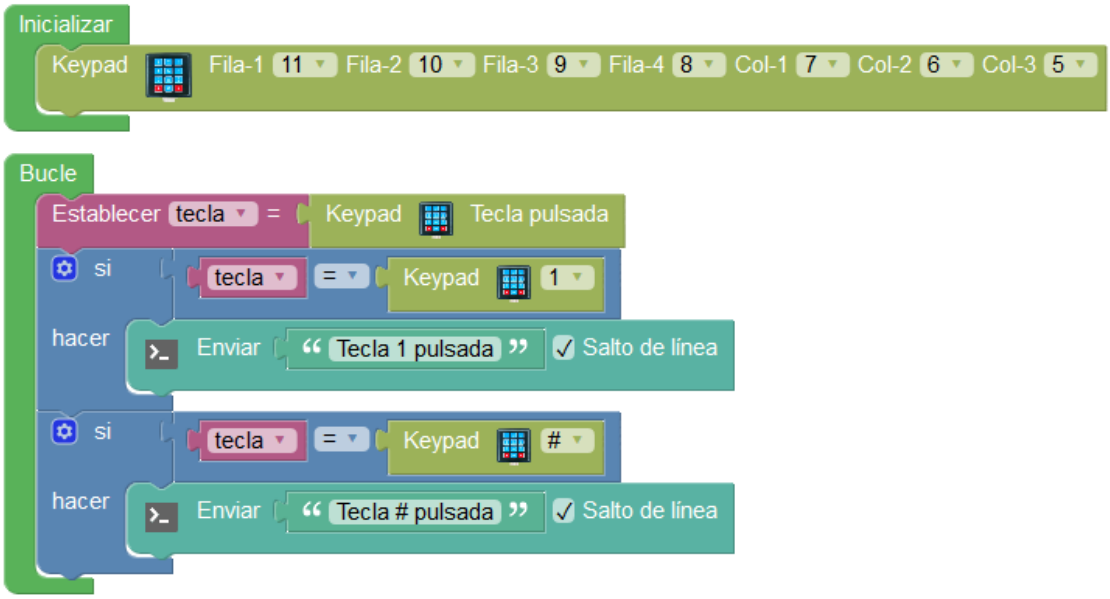
- **Configuración del keypad:** define los pines de conexión para las filas y columnas del keypad. Podemos configurar un keypad de 3x4 o de 4x4

Ejemplo de conexión y configuración:



- **Tecla pulsada:** obtiene la tecla pulsada actualmente en el keypad.

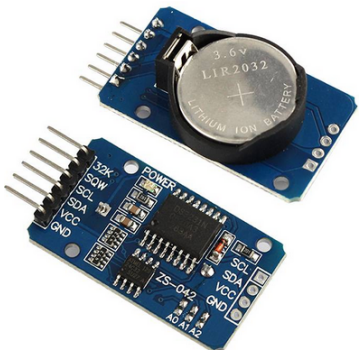
Ejemplo: Detección de las teclas '1' y '#'

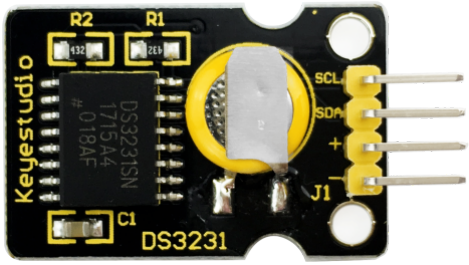


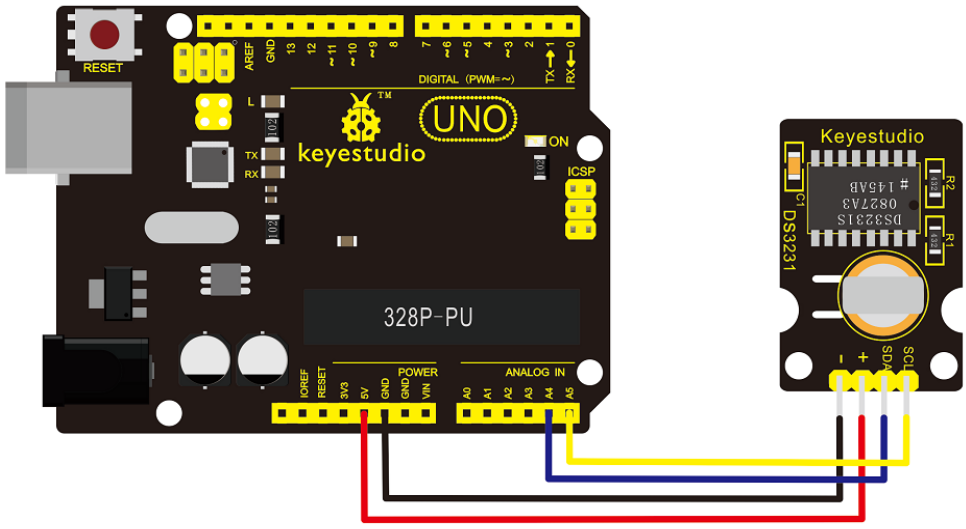
3.3.13 Reloj de tiempo real (RTC DS3231)

El reloj de tiempo real DS3231 es un reloj de alta precisión. El reloj incorpora una batería para guardar la fecha y la hora cuando la placa Arduino pierde la alimentación.

Se comunica con el microcontrolador de Arduino por comunicación I2C.

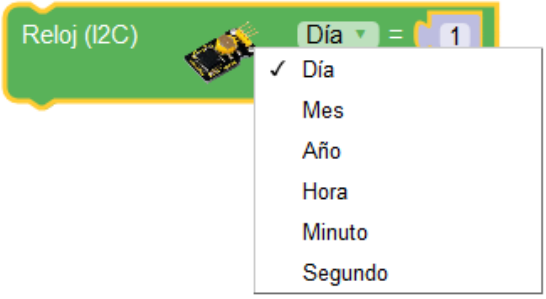






- **Reloj fijar fecha/hora:** Permite establecer los valores de fecha y hora.

- **Fijar campo de fecha/hora**: Permite establecer uno de los campos de la fecha / hora por separado.

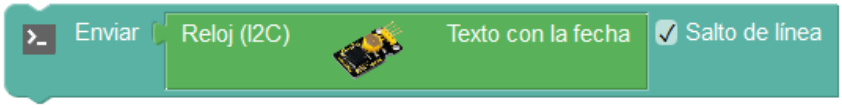


- **Obtener campo de fecha/hora:** Permite obtener los campos de fecha y hora de forma independiente.

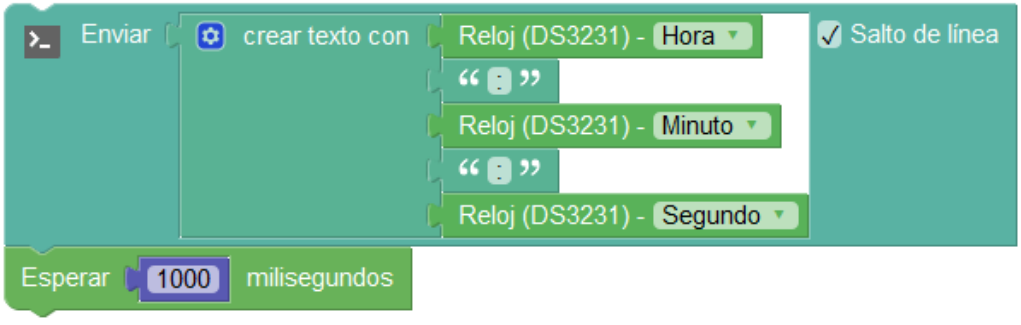
- **Obtener texto con la fecha:** Permite obtener un valor de tipo texto con la fecha formateada como *DD/MM/YYYY*

- **Obtener texto con la hora:** permite obtener un valor de tipo texto con la fecha formateada como *hh:mm:ss*

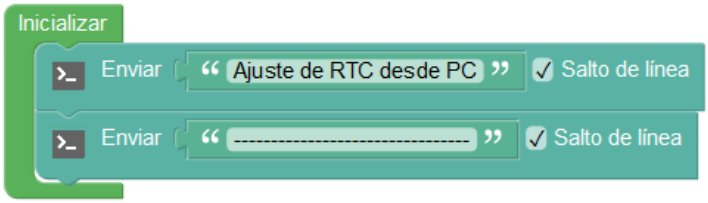
Ejemplo: enviar la fecha por la consola serie

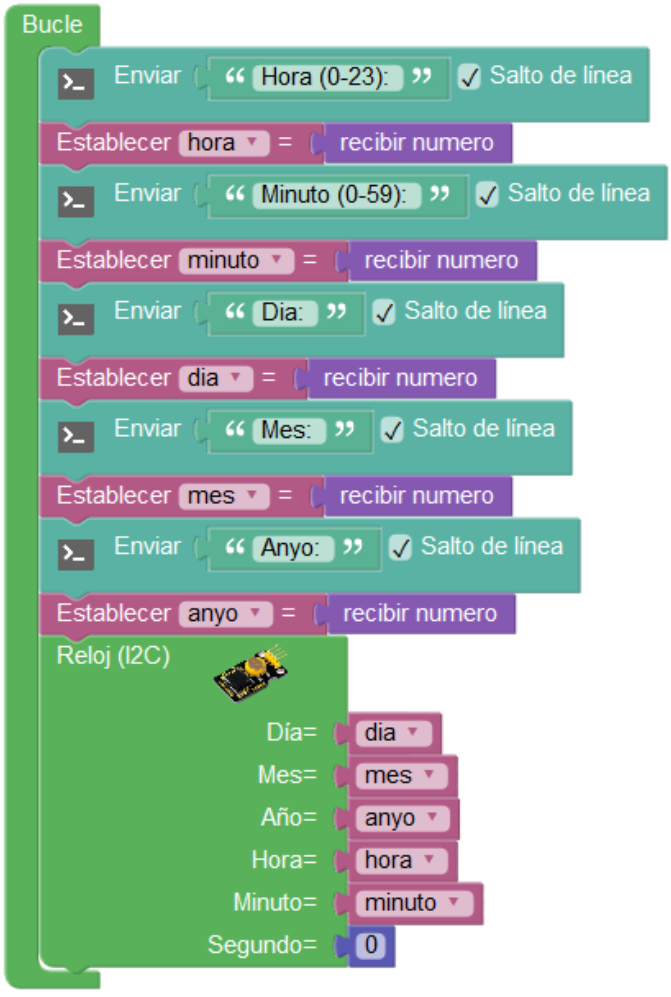


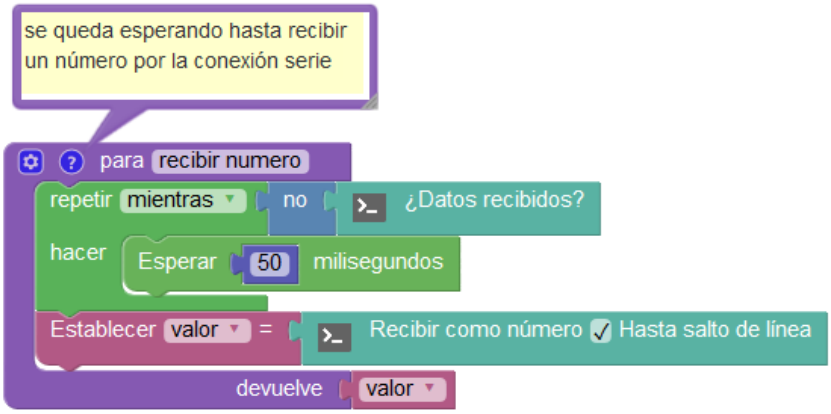
Ejemplo: enviar la hora completa cada segundo por la conexión serie



Ejemplo: Ajuste de fecha y hora del reloj RTC desde PC (vía consola serie)



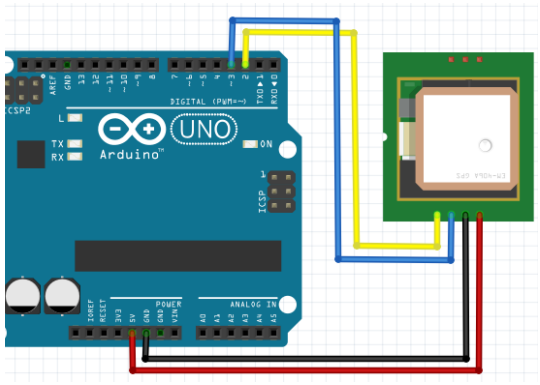




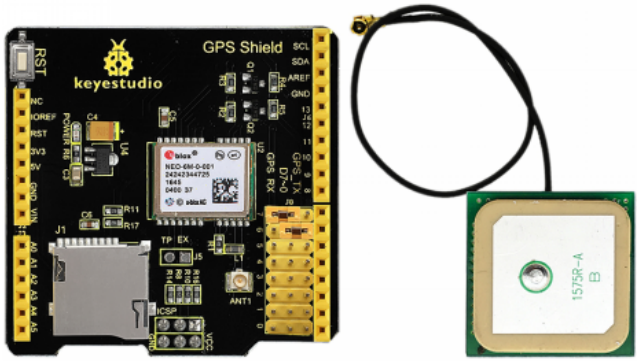
3.3.14 GPS

Los módulos GPS nos permiten de forma sencilla obtener los datos de posición global (latitud/longitud), velocidad, orientación, altitud, … facilitados por el sistema de posicionamiento global. El módulo GPS conectado debe ser un módulo de conexión serie que proporcione los datos según el protocolo NMEA. Uno de los módulos más utilizados de este tipo son los GPS NEO-6.

Conexióm módulo GPS serie



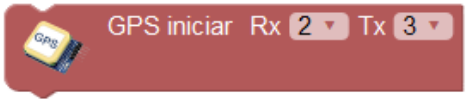
Shield GPS



http://shop.innovadidactic.com/index.php?id_product=844&controller=product

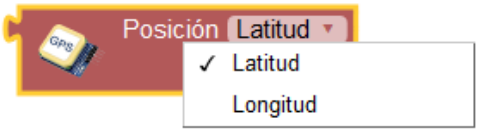
http://shop.innovadidactic.com/index.php?id_product=733&controller=product

- **GPS Iniciar**: Inicia el módulo GPS indicando los pines utilizados para la comunicación serie con el módulo.

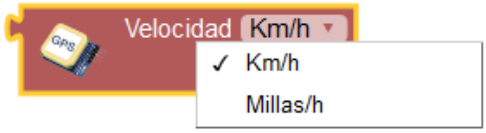


- **¿Datos válidos?:** Indica verdadero en caso de que el módulo GPS reciba señal desde los satélites GPS de forma correcta y los datos obtenidos sean válidos, si no obtendremos valor falso.

- **Posición:** Obtiene la latitud y longitud para así obtener la información de la posición actual. Los valores de latitud y longitud son valores decimales que determinan nuestra posición sobre la Tierra.

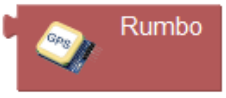


- **Velocidad:** Obtiene el valor de la velocidad a la que nos movemos, puede ser en Km/h o Millas/h.

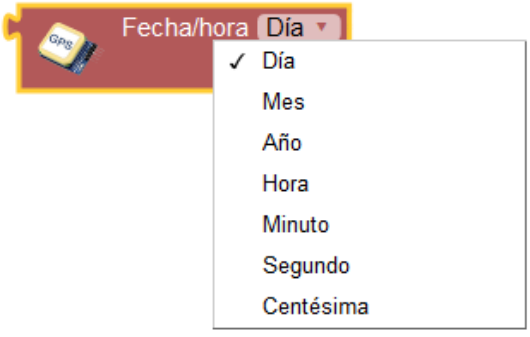


- **Altitud:** Obtiene la altitud en metros sobre el nivel del mar.

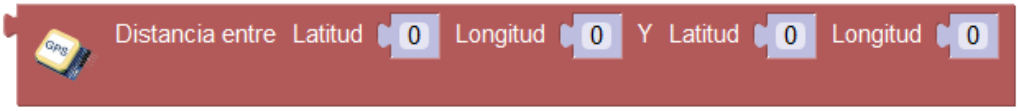
- **Rumbo:** Indica el valor en grados de la dirección a las que nos dirigimos.



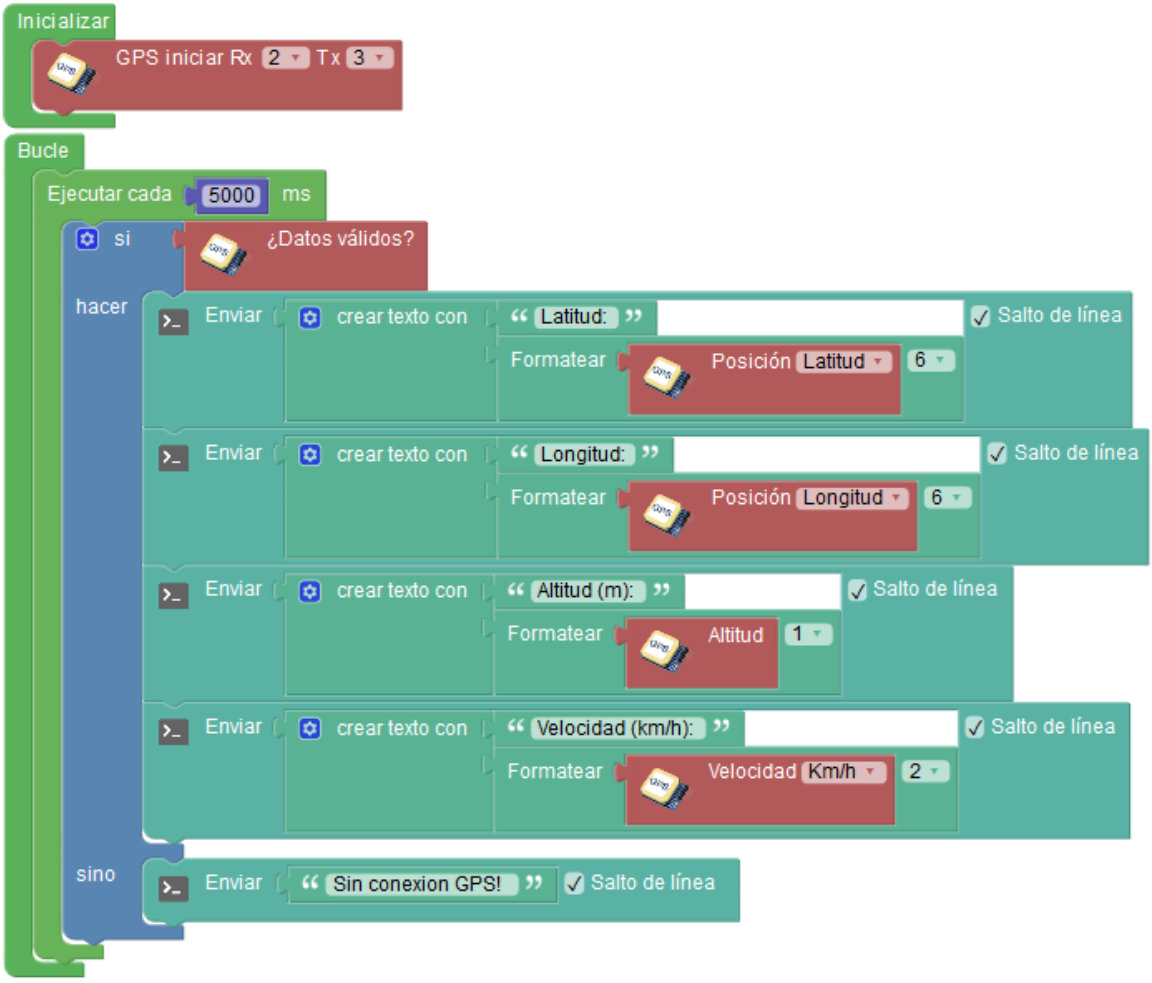
- **Fecha/hora:** Obtiene los valores de fecha y hora recibidos desde el satélite GPS.



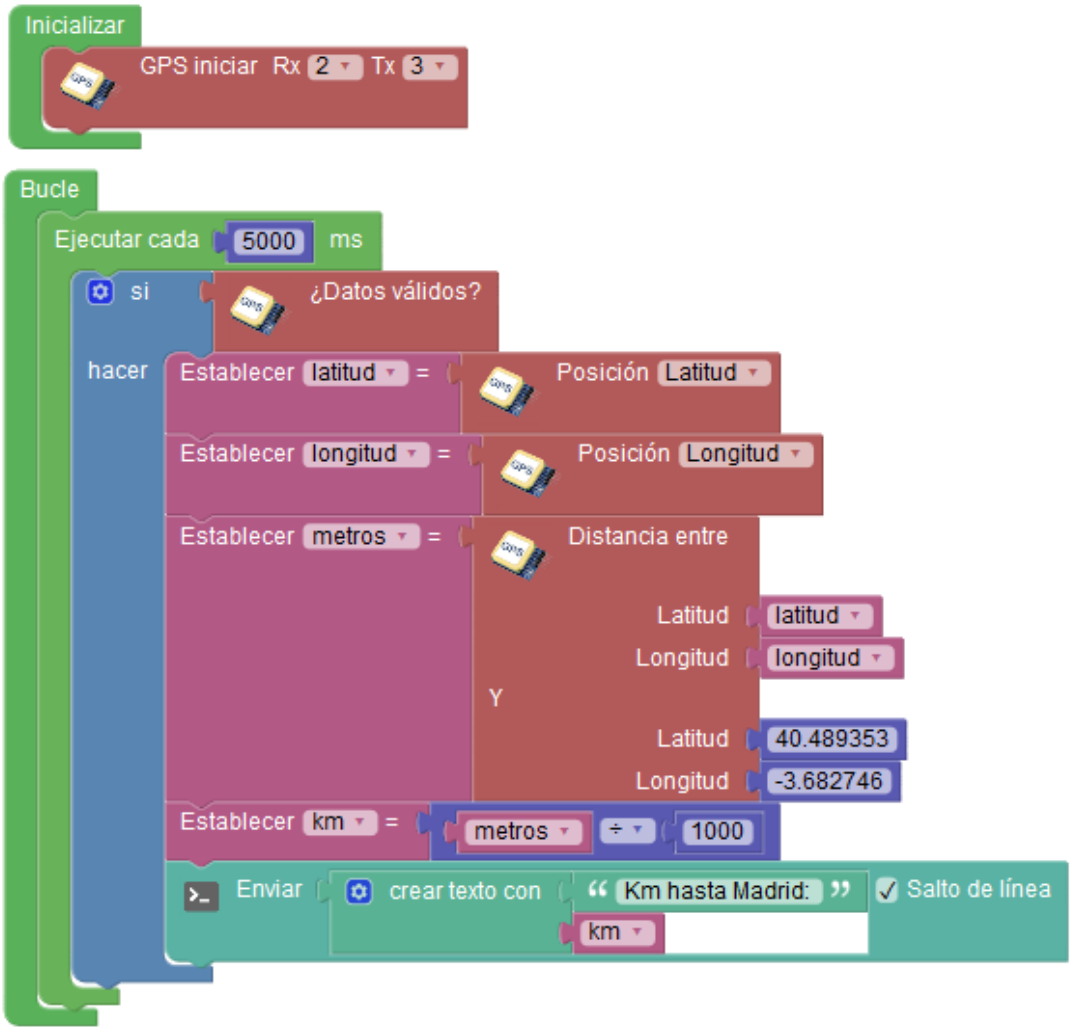
- **Distancia entre:** Calcula los metros de distancia en línea recta entre dos puntos indicando la latitud y longitud del punto inicial y final.



Ejemplo: Mostrar la información de la posición GPS por la conexión serie cada 5s



Ejemplo: Mostrar la distancia en Km hasta Madrid desde nuestra posición actual cada 5s



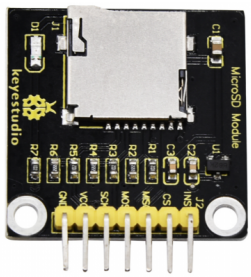
3.3.15 Tarjeta SD

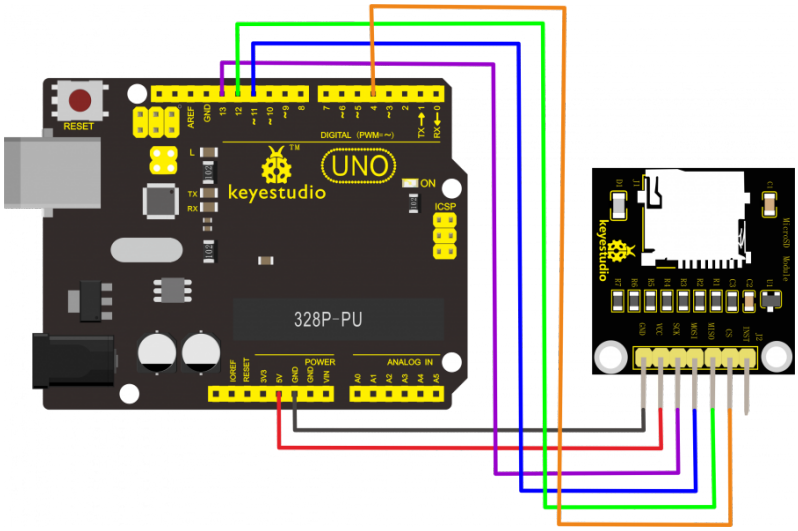
Los bloques de tarjeta SD nos permiten trabajar con archivos almacenados en una tarjeta SD o microSD conectada a Arduino. Los módulos para tarjetas SD utilizan la conexión SPI para comunicarse con la placa Arduino.

Este tipo de almacenamiento nos permite realizar aplicaciones de registro de datos (datalogger), guardar configuración, etc.

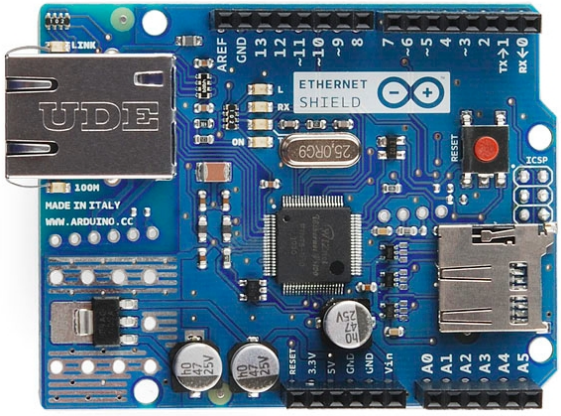
Los módulos o shields SD se conectan con la interfaz SPI utilizando los pines 11,12 y 13 y otro pin para CS (normalmente las shields utilizan el pin 4).

Módulo SD

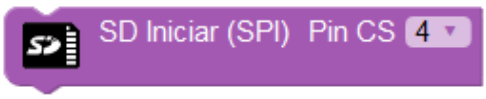




Algunas shields como la “Ethernet” incorporan también un módulo para tarjetas SD. Debemos comprobar su documentación para asegurarnos los pines que utilizan (en el caso de la shield “Ethernet” utiliza el pin 4 para CS)



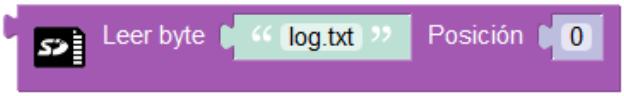
- **Iniciar SD:** Inicia el uso del módulo de tarjetas SD indicando los pines donde está conectado. (los pines SPI son fijos, sólo indicamos el pin CS)



- **Imprimir:** Escribe un texto dentro de un archivo de texto en la tarjeta SD. El texto se añade al final del contenido actual del archivo.

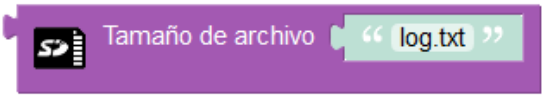
- **Escribir byte:** Escribe un byte al final del archivo indicado.

- **Leer byte:** Lee un byte del archivo indicado de la posición seleccionada.

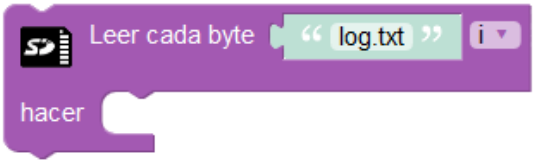


- **Eliminar archivo:** Elimina un archivo de la tarjeta SD.

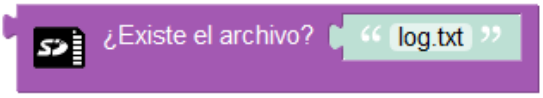
- **Tamaño de archivo:** Obtiene el tamaño en bytes del archivo indicado.



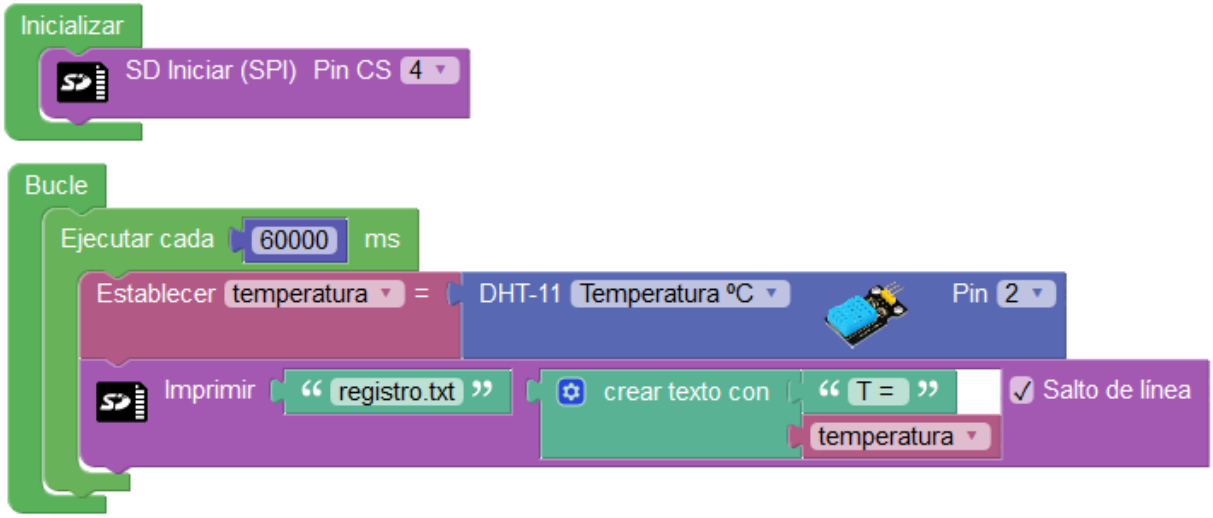
- **Leer cada byte:** Permite leer byte a byte todos los datos de un archivo.



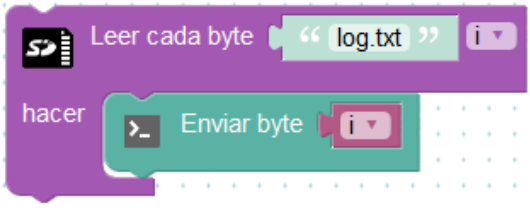
- **Existe el archivo:** Obtiene el valor verdadero si el archivo existe o falso en caso contrario.



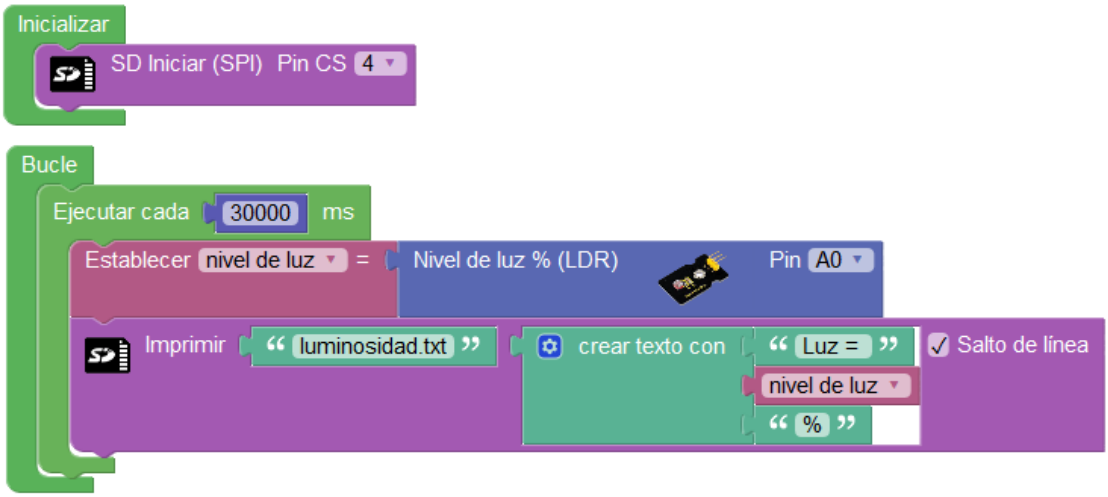
Ejemplo: Registrar la temperatura en un archivo de texto cada minuto



Ejemplo: Volcar todo el contenido de un archivo por la consola serie



Ejemplo: Registrar el nivel de luminosidad medido con una LDR cada 30s



3.3.16 MQTT

Mediante los bloques MQTT podemos conectar nuestro Arduino o placa ESP8266 al IoT (internet de las cosas) para realizar aplicaciones de control remoto, monitorización, recogida de datos, etc.

Para conectar la placa de desarrollo a internet tenemos varias posibilidades:

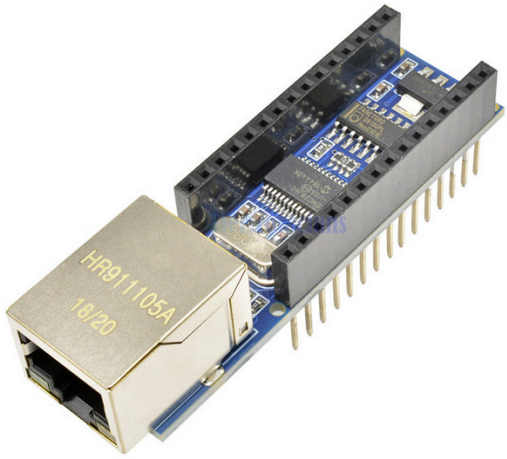
- **Arduino + shield ethernet:**

  La shield Ethernet utiliza los pines 10,11,12 y 13 (SPI) . Además utiliza el pin 4 si utilizamos el módulo de tarjetas SD que incorpora.

La shield Ethernet incorpora un conector RJ45 para cable Ethernet que debemos conectar a nuestro router o switch con conexión a internet.

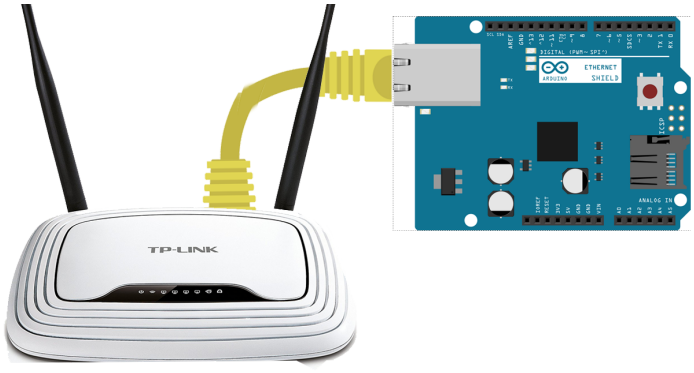
Arduino UNO + Ethernet shield

Arduino Nano Ethernet shield



http://shop.innovadidactic.com/index.php?id_product=679&controller=product

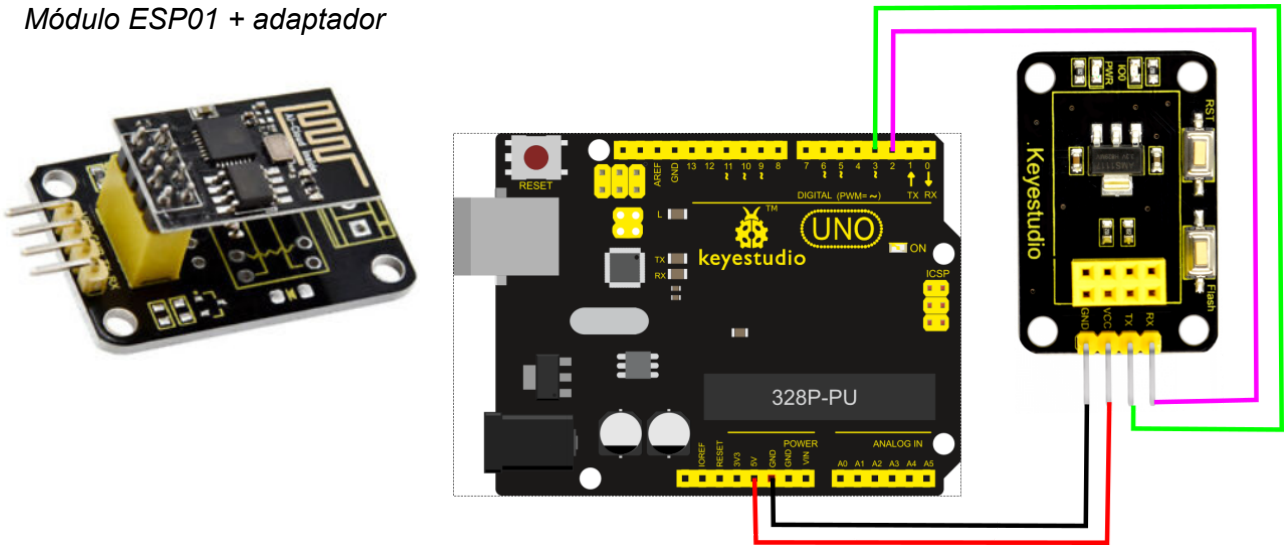
Conexión de Arduino al router doméstico mediante cable RJ45:



- **Arduino + ESP01 WiFi:**

La opción más sencilla y económica para añadir conectividad WiFi a nuestra placa Arduino es mediante el módulo ESP01. Este módulo se basa en el chip ESP8266 y permite conectarlo a nuestro Arduino como un periférico serie que dotará de conectividad WiFi de forma muy económica y sencilla. Para conectar el módulo ESP01 necesitamo un modulo adaptador para ajsutar los niveles de voltaje entre el módulo y la placa.

Módulo ESP01 + adaptador



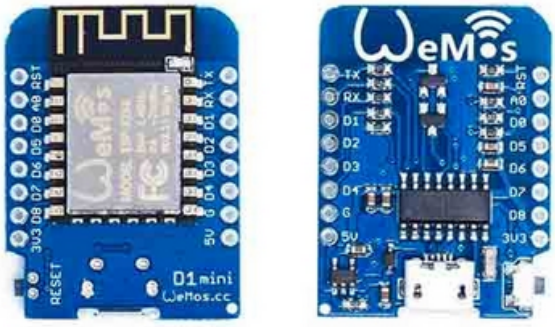
+Información sobre conectividad WiFi en el apartado de comunicaciones

- **NodeMCU o WeMos (*chip ESP8266 con WiFi incorporado*):**

Las placas de desarrollo basadas en el microcontrolador ESP8266 incorporan la conectividad WiFi en el propio micro. Estas placas no necesitan hardware extra para conectarse a la red.

NodeMCU

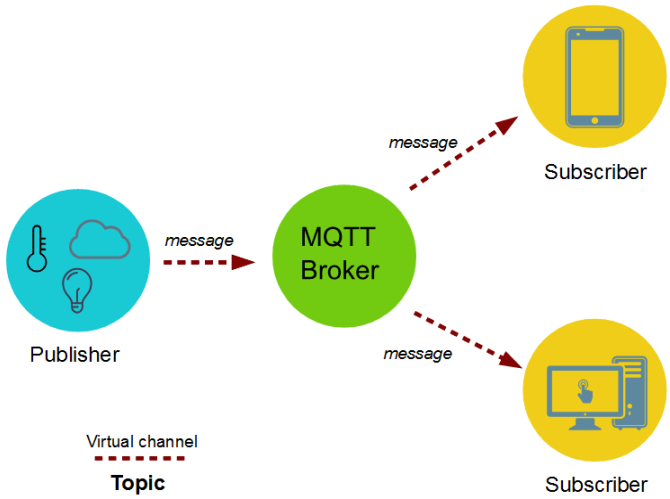
WeMos Mini



Protocolo MQTT

MQTT es un protocolo de comunicación para redes TCP/IP muy sencillo y ligero en el que todos los dispositivos se conectan a un servidor (llamado "*broker*"). Los dispositivos pueden enviar (*publicar*) o recibir (*suscribirse*) mensajes asociándoles un "*topic*" (tema).

El "*broker*" se encarga de gestionar los mensajes y distribuirlos entre todos los dispositivos conectados.



Podemos implementar nuestro propio servidor/broker. Existen brokers MQTT de código libre como "Mosquitto" que podemos instalar en diferentes sistemas operativos de forma sencilla. Un ejemplo típico es configurar una Raspberry Pi como servidor MQTT en casa. Si queremos que el sistema esté abierto a internet deberemos configurar nuestra conexión adecuadamente al igual que obtener nuestra IP pública actual o contratar una IP pública fija.

Por otro lado podemos utilizar brokers MQTT públicos disponibles en internet con fines experimentales o docentes y en cualquier otro caso podemos contratar servicios de brokers de pago con diferentes limitaciones de ancho de banda o número de conexiones según nuestras necesidades.

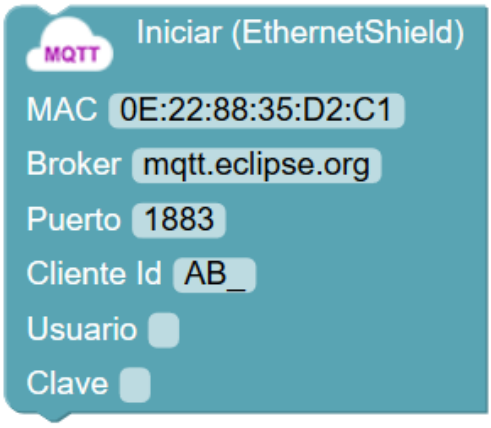
Algunos brokers MQTT para utilizar:
iot.eclipse.org
broker.hivemq.com
www.cloudmqtt.com

La comunicación entre los nodos de un sistema MQTT se realizan enviando mensajes. Los nodos envían los mensajes al broker y éste se encarga de distribuirlos entre el resto de nodos. Cada mensaje consta de un "topic" o tema y el cuerpo del mensaje en sí. Un nodo se puede suscribir a un "topic" de forma que recibirá todos los mensajes que tengan ese "topic". Cada nodo puede publicar mensajes con el "topic" deseado.

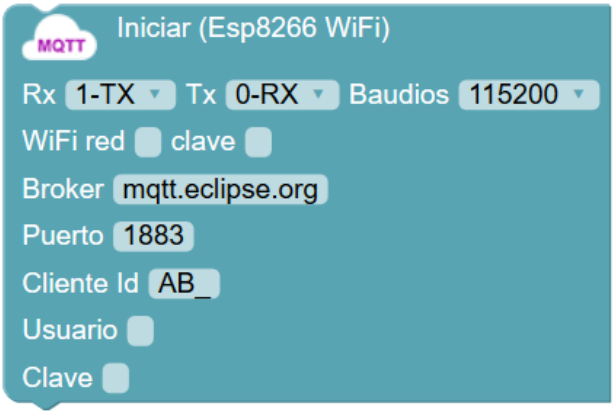
Por ejemplo podemos utilizar el topic: "temp/comedor" para que un nodo envíe la temperatura del comedor, por otro lado todos los nodos que deseen conocer la temperatura del comedor se suscribirán al topic : "temp/comedor" y recibirán automáticamente los mensajes de este tipo.

Bloques para la programación MQTT:

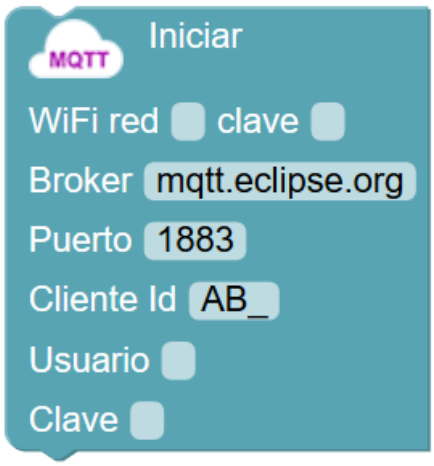
- **Iniciar MQTT (EthernetShield):** Inicia la conexión MQTT a través de la shield Ethernet. La dirección MAC generada es aleatoria, si nuestra shield incluye una etiqueta con la dirección MAC debemos ponerla. Por otro lado indicaremos el broker y puerto a utilizar, el usuario/clave si es necesario y el identificador del cliente MQTT. La tarjeta de red Ethernet intentará obtener la configuración IP de forma automáticamente por lo que nuestra red deberá tener un servidor DHCP activo que proporcione esta información (cualquier router doméstico lleva esta opción activa por defecto).



- **Iniciar MQTT (Esp8266 WiFi):** Utiliza un módulo ESP-01 serie para establecer la conexión serie. Debemos especificar los pines donde se conectan los pines RX y TX del módulo. Si el módulo está configurado (por defecto) para comunicar a una velocidad de 115200 bps tendremos obligatoriamente que usar los pines 0,1. Si el módulo está configurado para trabajar a una velocidad inferior podemos especificar otros pines. En caso de usar los pines 0,1 debemos desconectar el módulo durante la programación de la placa Arduino. Por otro lado especificaremos el nombre de la red WiFi y la clave así como el broker y puerto a utilizar, el usuario/clave si es necesario y el identificador del cliente MQTT. El router WiFi debe tener el servicio DHCP activado para asignarnos una IP de forma automática.



- **Iniciar MQTT (NodeMCU o WeMos):** En caso de las placas con WiFi integrado sólo debemos especificar el nombre de la red WiFi y la clave, así como los datos de conexión del broker, puerto, nombre identificativo del cliente y usuario/clave del broker si fuera necesario.



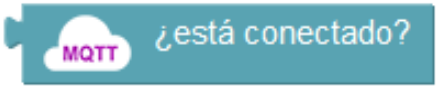
- **Publicar MQTT:** Permite enviar un mensaje al broker para que los nodos suscritos a este topic reciban el valor. El tema es el “topic” a publicar y el valor puede ser un valor fijo (texto o numérico) o el valor de una variable.

- **Suscribir MQTT:** Este bloque realiza la suscripción a un “topic” o tema. ArduinoBlocks mapea el valor recibido en el mensaje a una variable de forma que cuando se recibe un mensaje del “topic” automáticamente el valor de la variable se actualizará.

Para variables numéricas (el valor recibido debe ser un número válido y se guardara en la variable indicada)

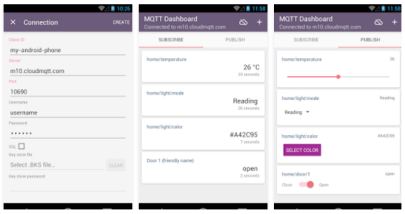
En este caso almacenaremos el mensaje recibido en una variable de tipo texto:

- **¿Está conectado?**: Obtiene el estado de la conexión, indicando verdadero si se ha podido establecer la conexión con el broker o falso en caso contrario.

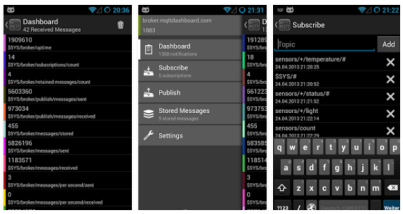


Existen multitud de aplicaciones, especialmente para dispositivos móviles, para conectarse a un broker MQTT y publicar o suscribirse a topics. Algunas de ellas además permiten crear paneles de control y monitorización muy llamativos.

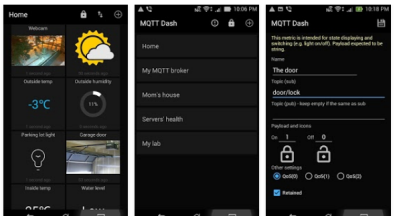
MQTT Dashboard (Android)



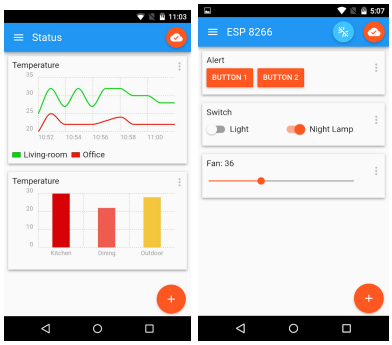
MyMQTT (Android)



MQTT Dash (Android)

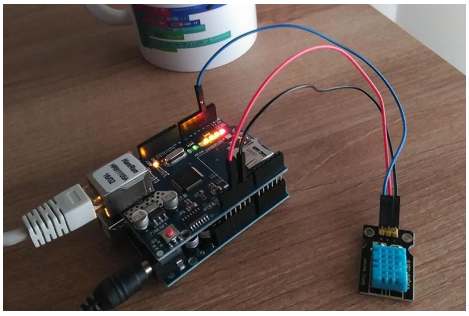


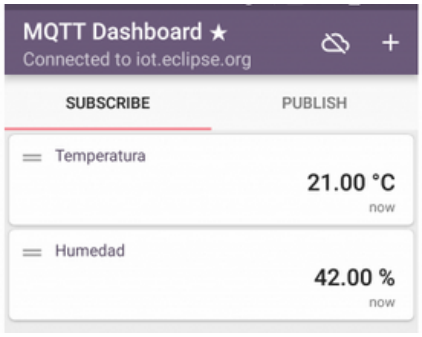
IoT MQTT Panel (Android)

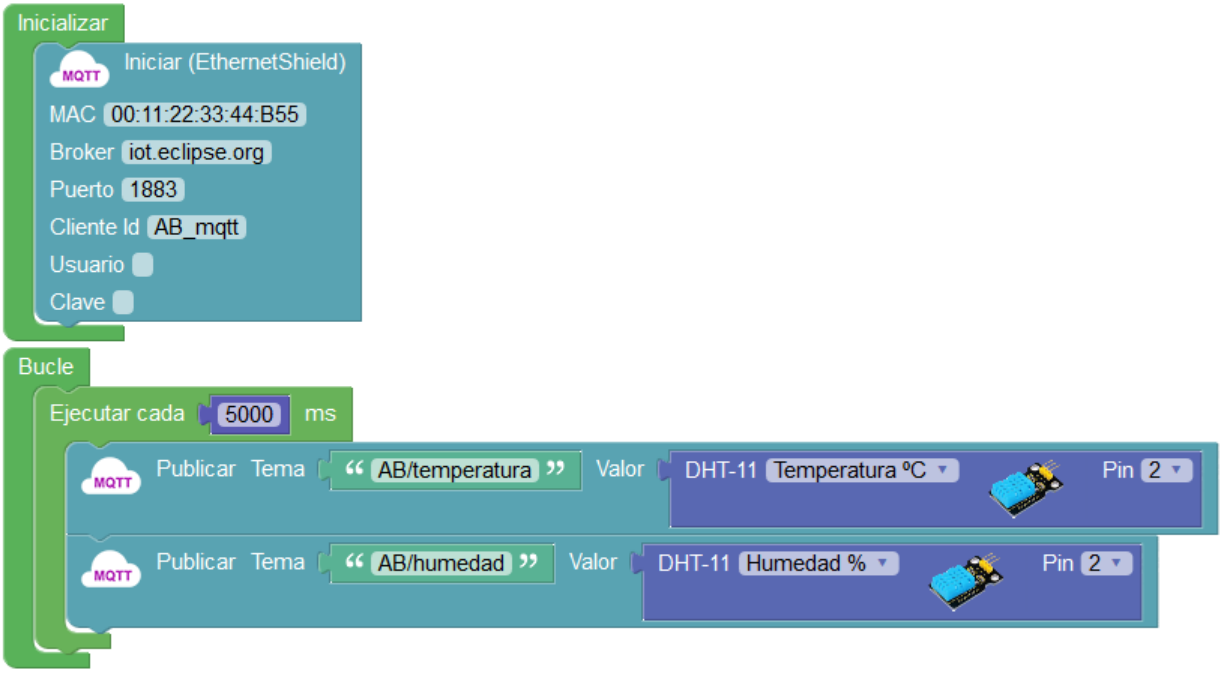


Ejemplo: Enviar la temperatura y humedad medido cada 5s

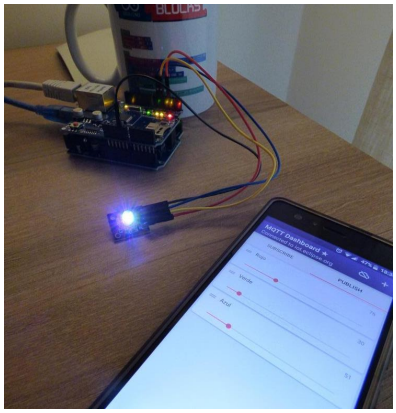
(recuerda que no debes utilizar bloqueos de tiempo para que el sistema MQTT funcione bien)

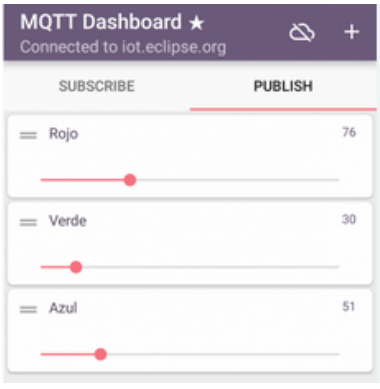


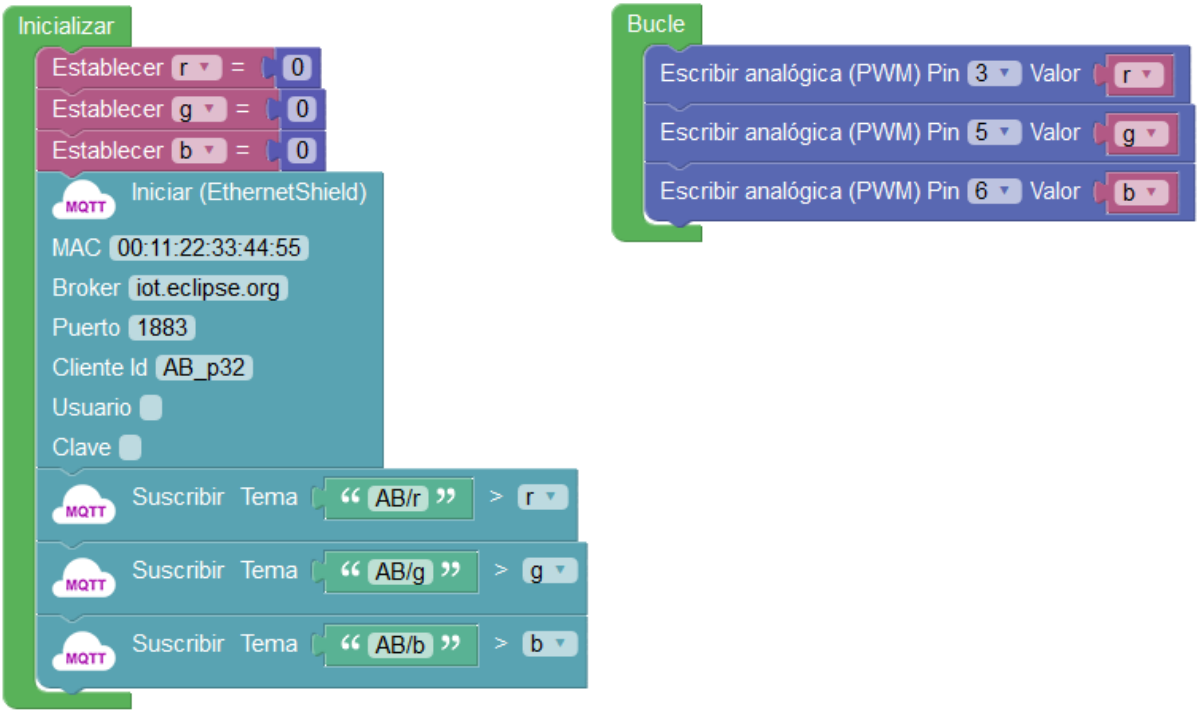




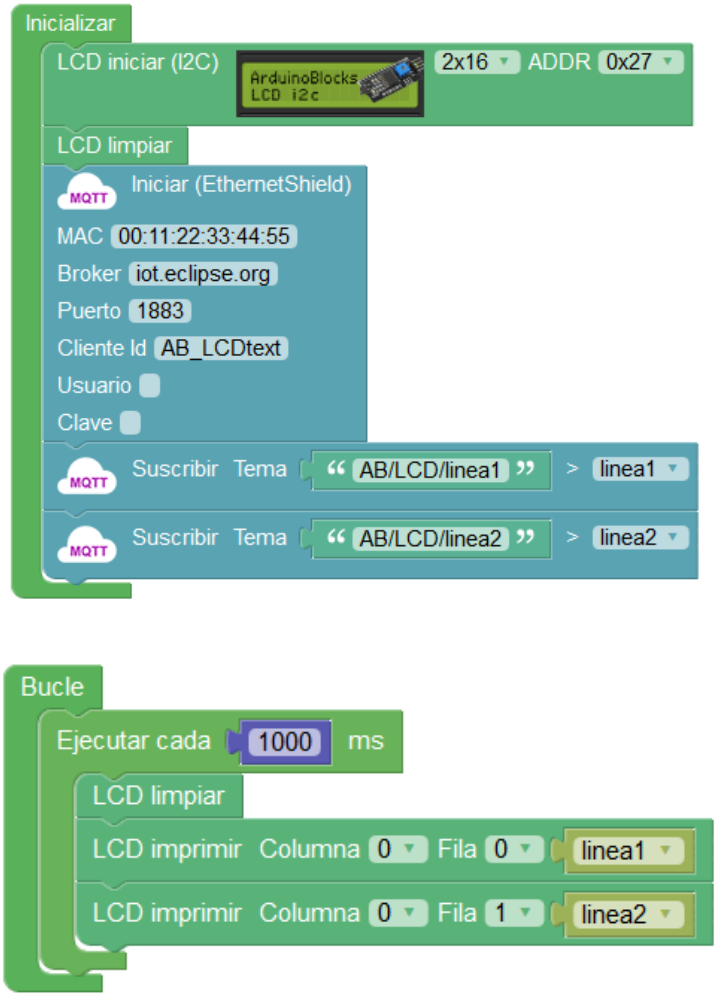
Ejemplo: Suscribirse a tres “topics” para controlar un led RGB.



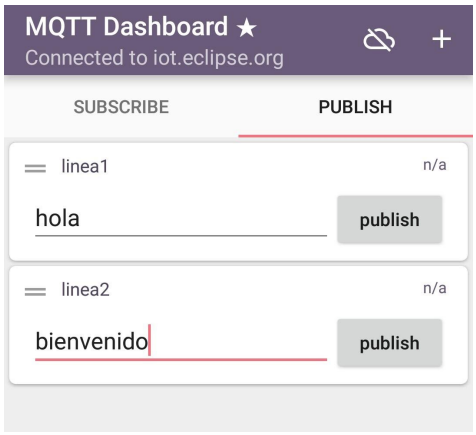




Ejemplo: Recibir un texto via MQTT para mostrar en un display LCD



Desde la aplicación MQTT Dashboard (Android) podemos modificar los valores de los textos que se visualizan en cada línea del LCD:



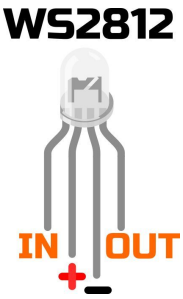
+Información sobre MQTT y ejemplos:

http://arduinoblocks.didactronica.com/tag/mqtt/

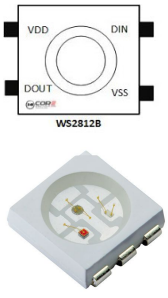
3.3.17 Neopíxel / WS2812

Los leds ws2812 o neopíxel son leds RGB inteligentes que incorporan su propio microcontrolador dentro de cada led. Normalmente se encuentran en forma de tira de leds conectados uno al siguiente , pudiendo realizar tiras de decenas o cientos de leds.

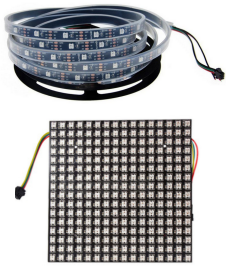
Led standard



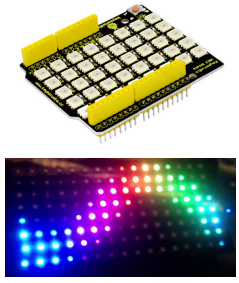
Led SMD



Tira / matrix leds



Shield nepíxel



Un led RGB ws2812 tiene normalmente 4 pins de contacto:

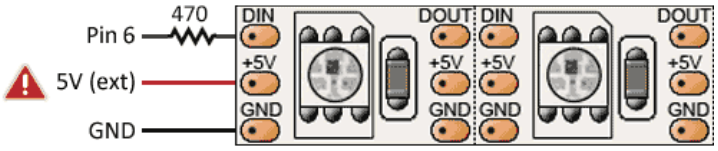
-*VCC*: alimentación 3 o 5v

-*GND*: 0v

-*DIN* (*DI*): Entrada de información. Conectado a un pin de Arduino.

-*DOUT* (*DO*): Salida de información. Conectado al siguiente led o tira de leds.

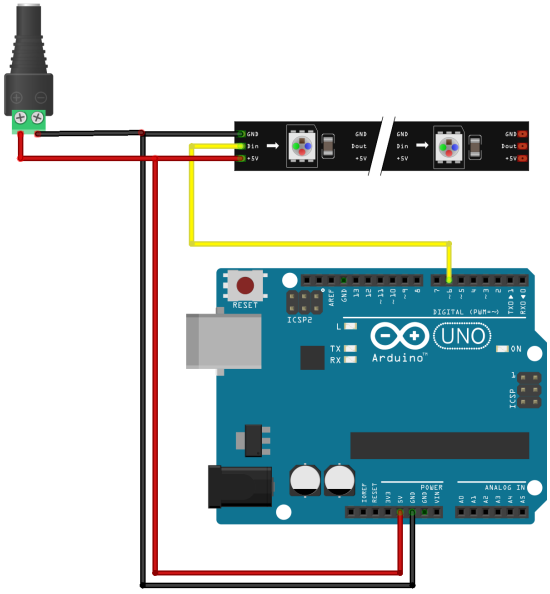
Ejemplo de conectores en Tira de leds neopíxel:



Las tiras de leds neopíxel necesitan normalmente una fuente de alimentación externa (si conectamos muy pocos leds, 5 o menos, no hará falta alimentación externa). En la mayoría de casos necesitaremos la fuente externa para suministrar suficiente corriente en caso de iluminar varios leds de la tira de leds simultáneamente.

Ejemplo: conexión tira de leds neopíxel con alimentación externa.

El GND de la fuente externa y el GND del Arduino siempre se deben unir.

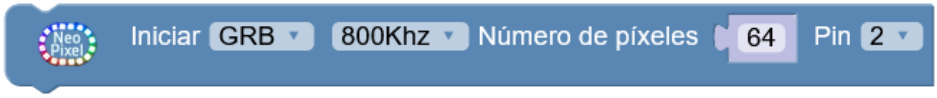


- **Iniciar:** Este bloque se debe definir dentro del "inicializar" o "setup" para indicar la configuración los leds neopíxel conectados. Indicando varios parámetros:

-**Frecuencia**: 800Khz / 400Khz, es la velocidad del "*bistream*" con los datos que se evían a través del pin DIN con los datos para cada neopíxel. (*Una tasa de 400khz ya permitiría controlar más de 1000 leds con un refresco de 30fps*)

-**Número de píxeles:** indica el número de píxeles conectados en serie, si conectamos una tira de leds de 100 leds lo indicaremos en este valor.

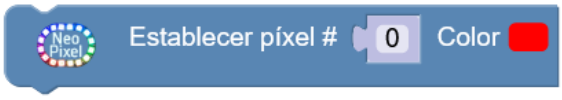
-Pin: Es el pin de control a través el cual Arduino controlará y enviará los datos a los leds.



- **Limpiar:** Apaga todos los neopíxels conectados en serie.

- **Mostrar**: Actualizar los datos enviados a los leds neopíxel. Cualquier operación de las siguientes no se reflejará hasta que se ejecute el bloque “mostrar”. Esto se realiza por una cuestión de optimización, así podemos realizar varias operaciones internamente y mostrarla a la vez en una sóla operación.

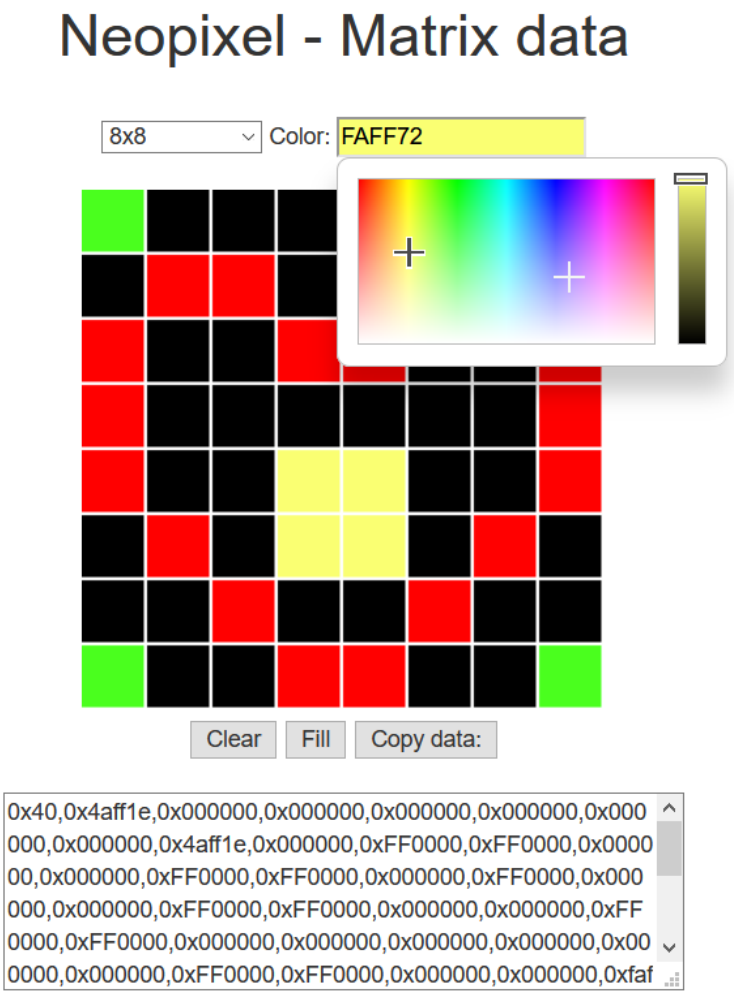
- **Establecer píxel:** Permite fijar un píxel (led) en concreto de toda la serie a un color, indicando el número de led (el primero es el 0) y los valores R,G y B (0...255) o seleccionando el color.



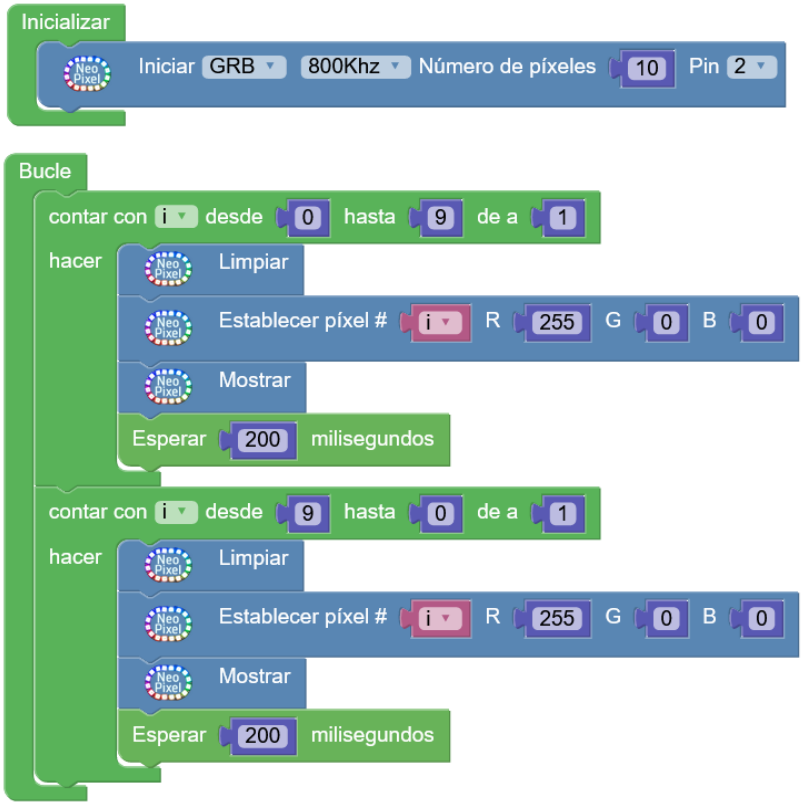
Ejemplo: iluminar una tira de 30 leds a rojo (255,0,0)

- **Establecer matriz:** Funciona de forma similar al anterior, pero en caso de usar una matriz de leds podemos indicar el píxel a modificar mediante sus coordenadas X,Y y los valores R,G,B (0...255)

- **Establecer datos:** Permite rellenar una matriz de leds a partir de un "*bitmap*" de datos. Con el botón derecho y la opción "*Ayuda*" podemos abrir el editor para obtener los datos del "*bitmap*"



Ejemplo: leds "coche fantástico" con neopíxel



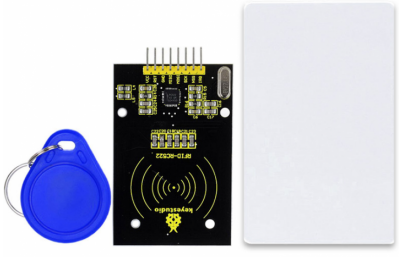
+Información y ejemplos:

http://arduinoblocks.didactronica.com/tag/neopixel/

3.3.18 RFID

La identificación mediante radio frecuencia, permite leer tarjetas o etiquetas sin contacto simplemente acercándolas. Al acercar una tarjeta o etiqueta RFID al lector podemos obtener el ID de cada una de ellas (que será única). Este tipo de dispositivos es perfecto para controles de presencia, controles de acceso, etc.

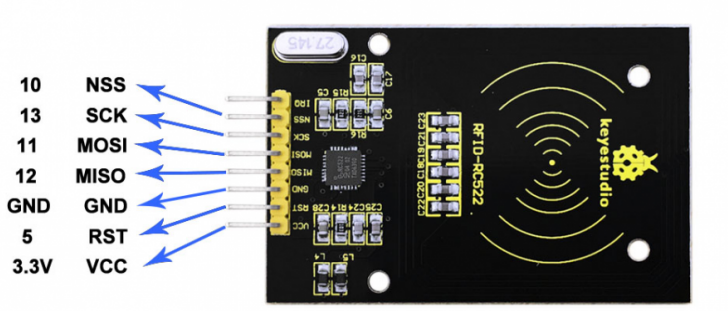
Llavero, lector y tarjeta RFID

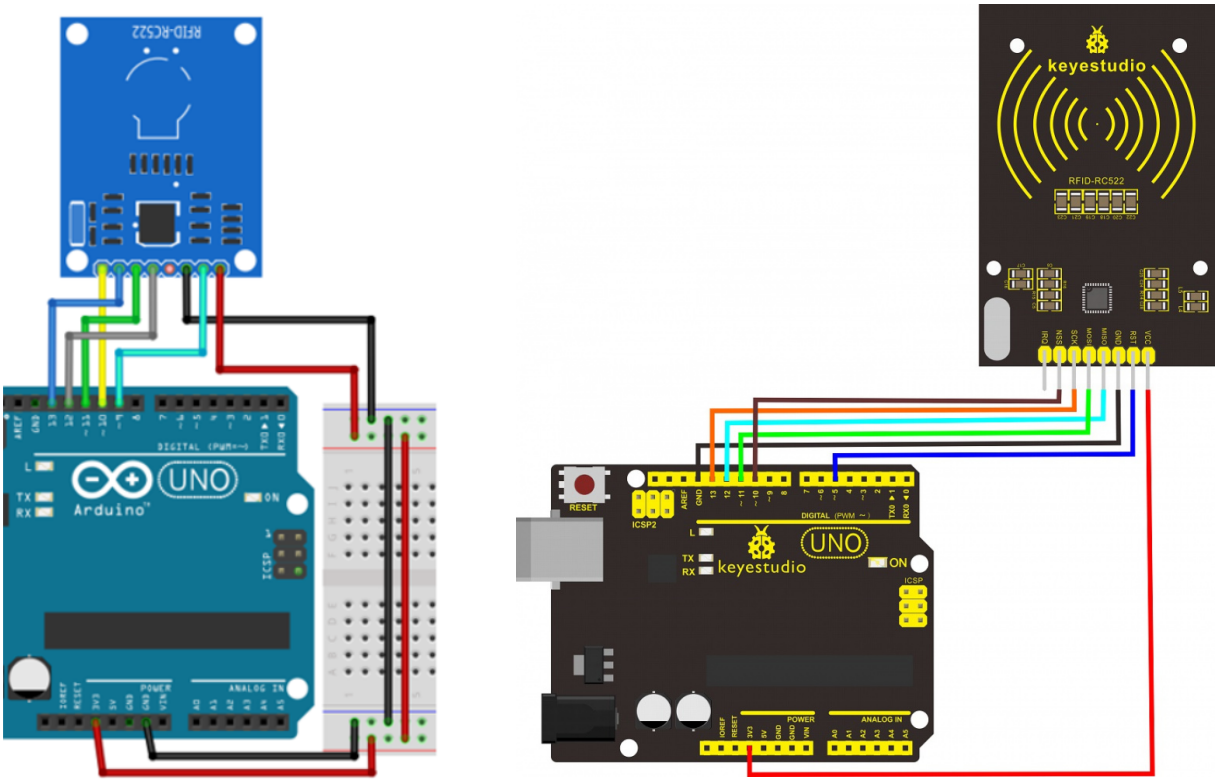


Etiquetas RFID

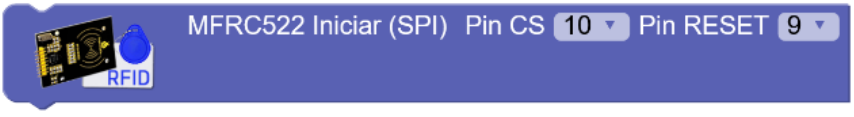
Lector RFID

El lector de tarjetas RFID se conecta mediante el bus SPI.

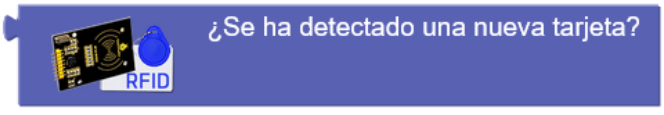




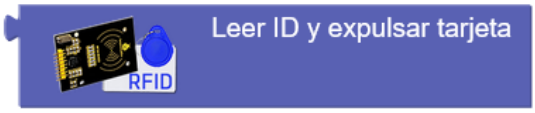
- Iniciar: A parte de la conexión SPI (pines 11,12,13) se debe indicar los pines para las conexiones CS (chip select) y RESET.



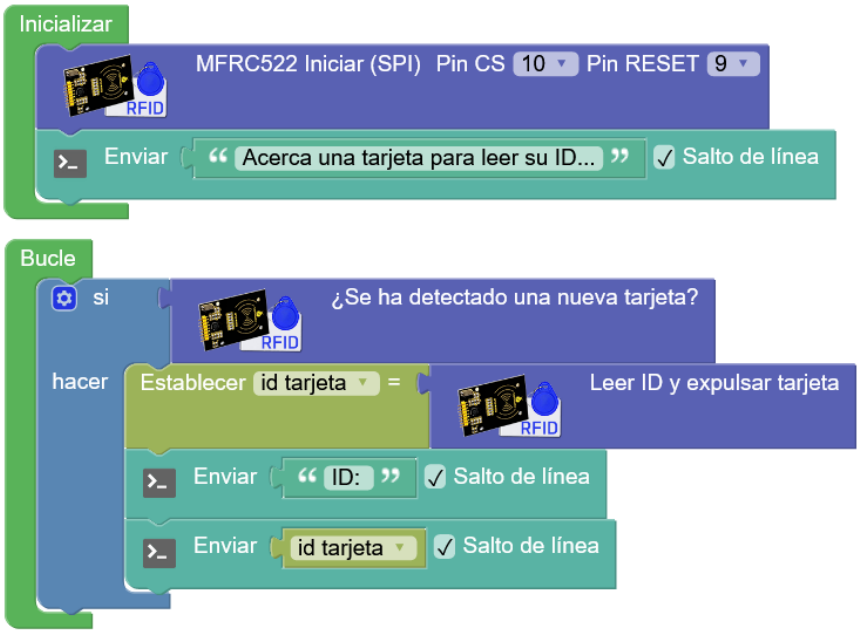
- **¿Se ha detectado una nueva tarjeta?** : Este bloque devuelve verdadero en caso de detectar una tarjeta cerca del lector pendiente de leer.



- **Leer y expulsar tarjeta:** Obtiene el código identificativo de la tarjeta o etiqueta. El código será una cadena de texto con representación hexadecimal de su identificador interno. Una vez leído el código, se deberá alejar y volver a acercar otra tarjeta para volver a poder leer (se expulsa virtualmente). Lo recomendable es almacenar este valor en una variable de texto para luego procesar el valor.

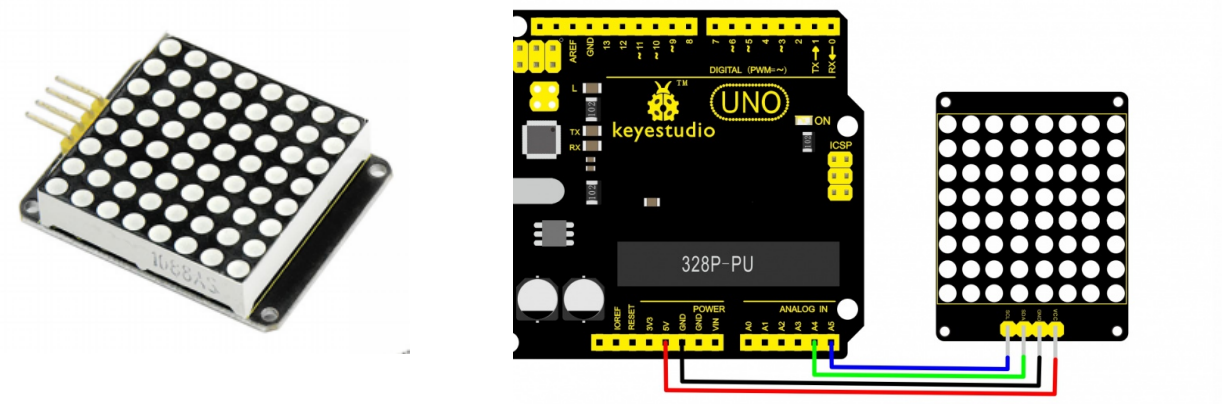


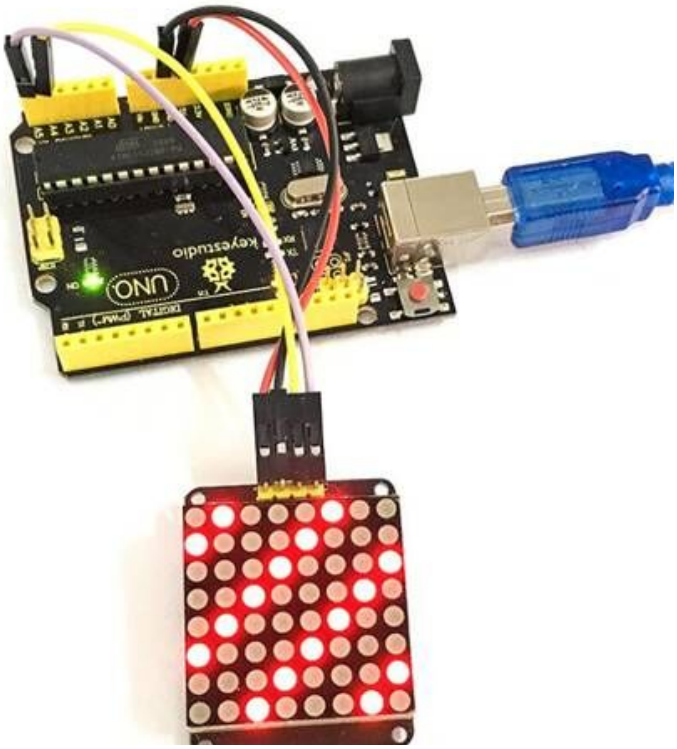
Ejemplo: Leer identificación de las tarjetas o etiquetas y mostrar el ID por la consola serie



3.3.19 Matriz de leds 8x8

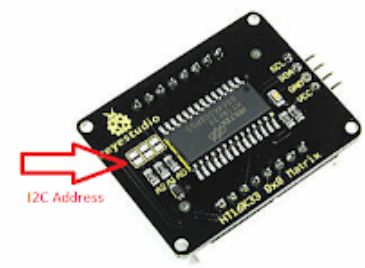
Las matrices de leds permiten controlas 64 leds monocromo (rojos normalmente) dispuestos en forma de matriz de 8 filas y 8 columnas. Para controlar los leds la matriz tiene un chip controlador (HT16K33) que gestiona los leds y se comunica con Arduino mediante bus I2C.



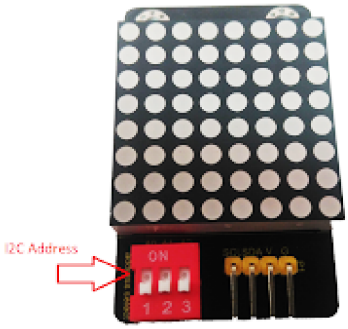


Podemos conectar varias matrices de leds en el mismo bus I2C, para controlarlas individualmente debemos cambiar la dirección de cada una de ellas.

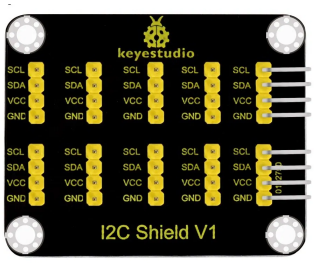
Configuración de la dirección I2C mediante pines soldados



Configuración de la dirección I2C mediante microswitchs



Hub I2C para ramificar y conectar varios dispositivos al mismo bus I2C

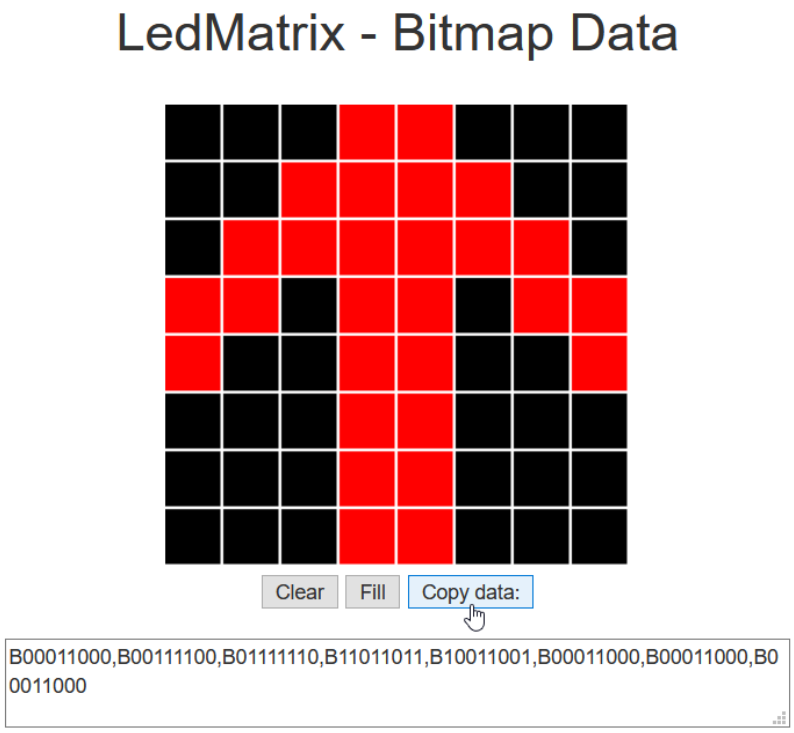


- **Iniciar:** Inicializa una matriz en una dirección en concreto. Posteriormente podremos referirnos a ella medienta el ID #

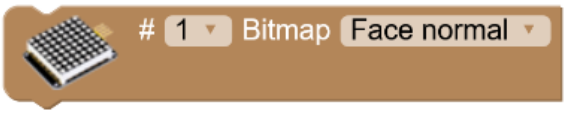
- **Rotación:** Permite ajustar la orientación de la matriz led para indicar en que posición se encuentra y funcione correctamente.

- **Limpiar:** Apaga todos los leds de la matriz

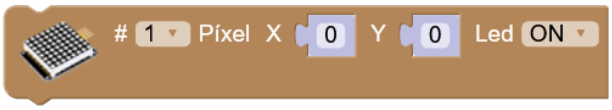
- **Bitmap:** Permite establecer un mapa de bits para dibujar en la matriz led. Mediante la opción “Ayuda” del bloque podemos acceder al editor de mapa de bits.



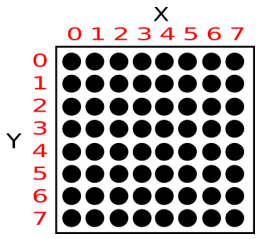
- **Bitmap (predefinidos):** Mediante este bloque podemos cargar bitmaps predefinidos de una forma muy sencilla.



- **Píxel:** Enciende o apaga un led en concreto indicando sus coordenadas.



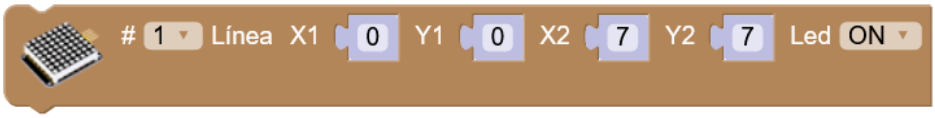
Sistema de coordenadas en la matriz led 8x8



- **Línea:** Permite dibujar una línea recta entre dos coordenadas en la matriz led.

  Punto origen: X1, Y1

  Punto destino: X2, Y2



- **Rectángulo:** Dibuja un rectángulo indicando las coordenadas una esquina y la contraria.

  Esquina 1: X1, Y1

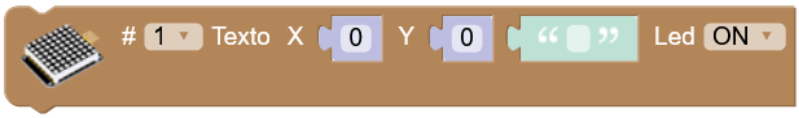
  Esquina 2: X2, Y2

- **Círculo:** Dibuja un círculo indicando las coordenadas del centro y el número de píxeles de radio.

  Centro: X, Y

  Radio: R

- **Texto:** Dibuja un texto en la matrix led. Usando coordenadas fuera de las visibles podemos realizar textos con desplazamiento con una única matriz led.

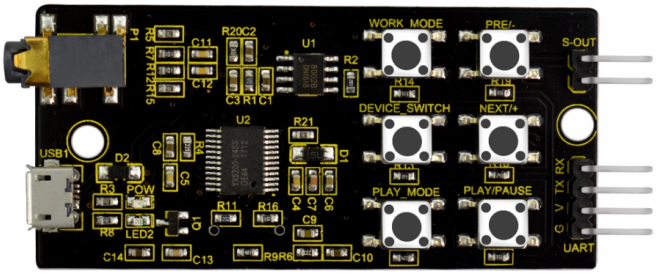


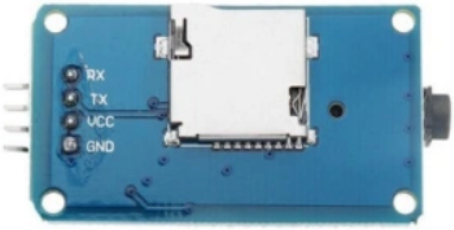
Ejemplo: Scroll de texto de izquierda a derecha

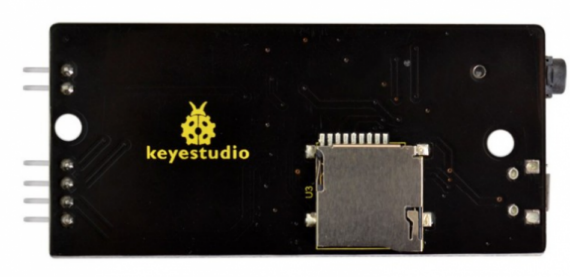
3.3.20 Reproductor Mp3

El módulo reproductor MP3 consta de un microcontrolador capaz de leer carpetas de archivos de sonido y decodificar el formato MP3, un amplificador con salida de audio para altavoz o auriculares y una interfaz serie para controlar el funcionamiento desde un microcontrolador como Arduino.

Estos módulos también conocidos como "*serial mp3*" están basados en el chip YX5300.





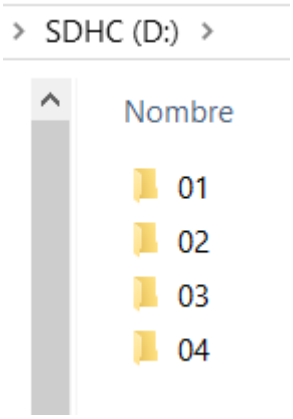


http://shop.innovadidactic.com/index.php?id_product=868&controller=product

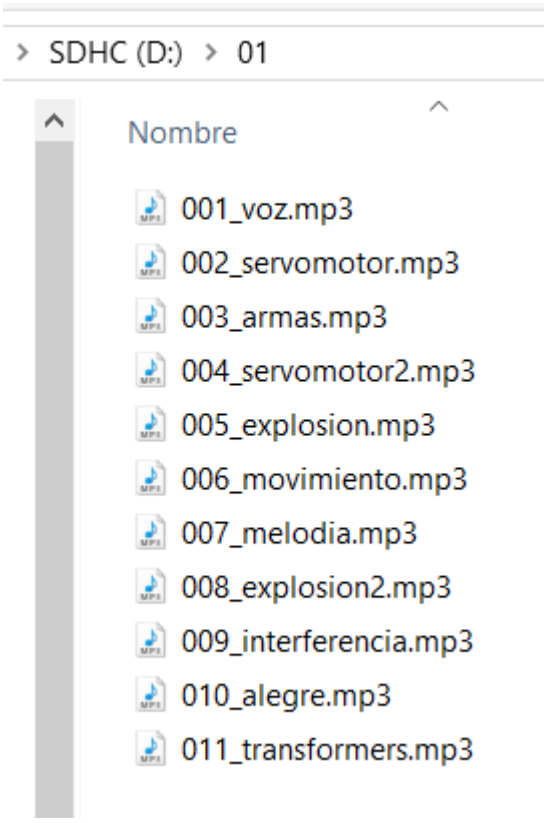
https://wiki.keyestudio.com/KS0387_keyestudio_YX5200-24SS_MP3_Module

El formato de la tarjeta micro SD debe ser FAT.

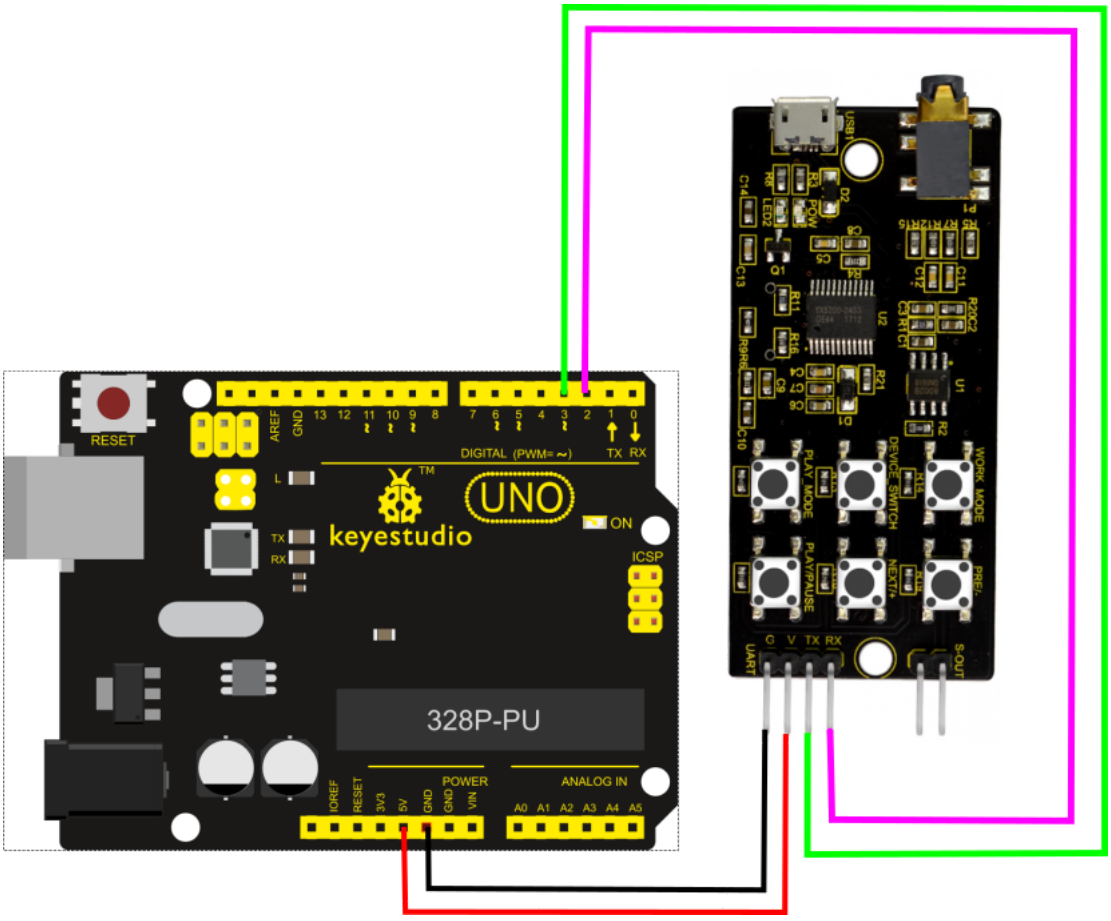
El nombre de las carpetas debe ser ***01, 02, 03… 99***



Y dentro cada canción o sonido debe empezar con 3 dígitos antes del nombre (**001...255**).



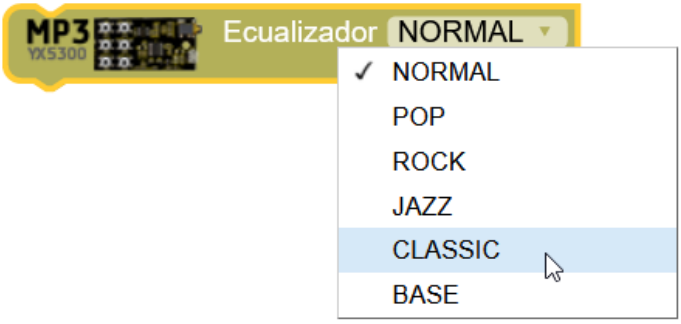
Ejemplo: conexión del módulo en los pines RX=2 / TX=3



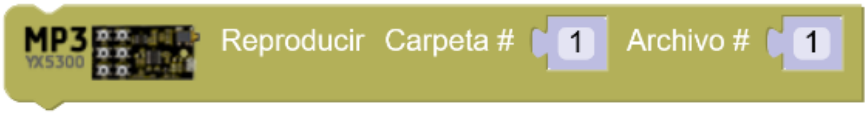
- **Iniciar**: Inicializa el módulo indicando los pines utilizados para la comunicación serie (RX/TX)

- **Volumen**: Establece el nivel de sonido del módulo (0...30)

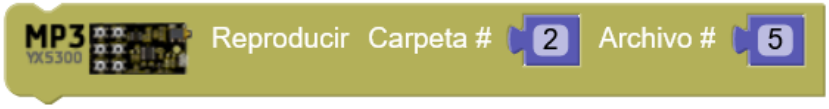
- **Ecualizador**: Selecciona uno de los ajustes predefinidos de ecualización del sonido.



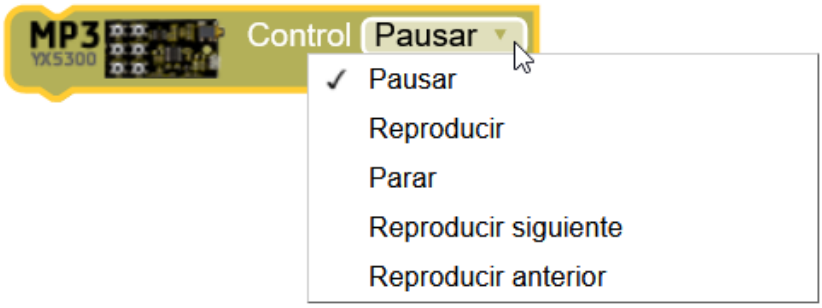
- **Reproducir**: Inicia la reproducción de un archivo de sonido. Se debe indicar el número de carpeta (0...99) y el número de archivo (0...255)



*Ejemplo: reproducir el sonido "**02/005_explosion.mp3**"*

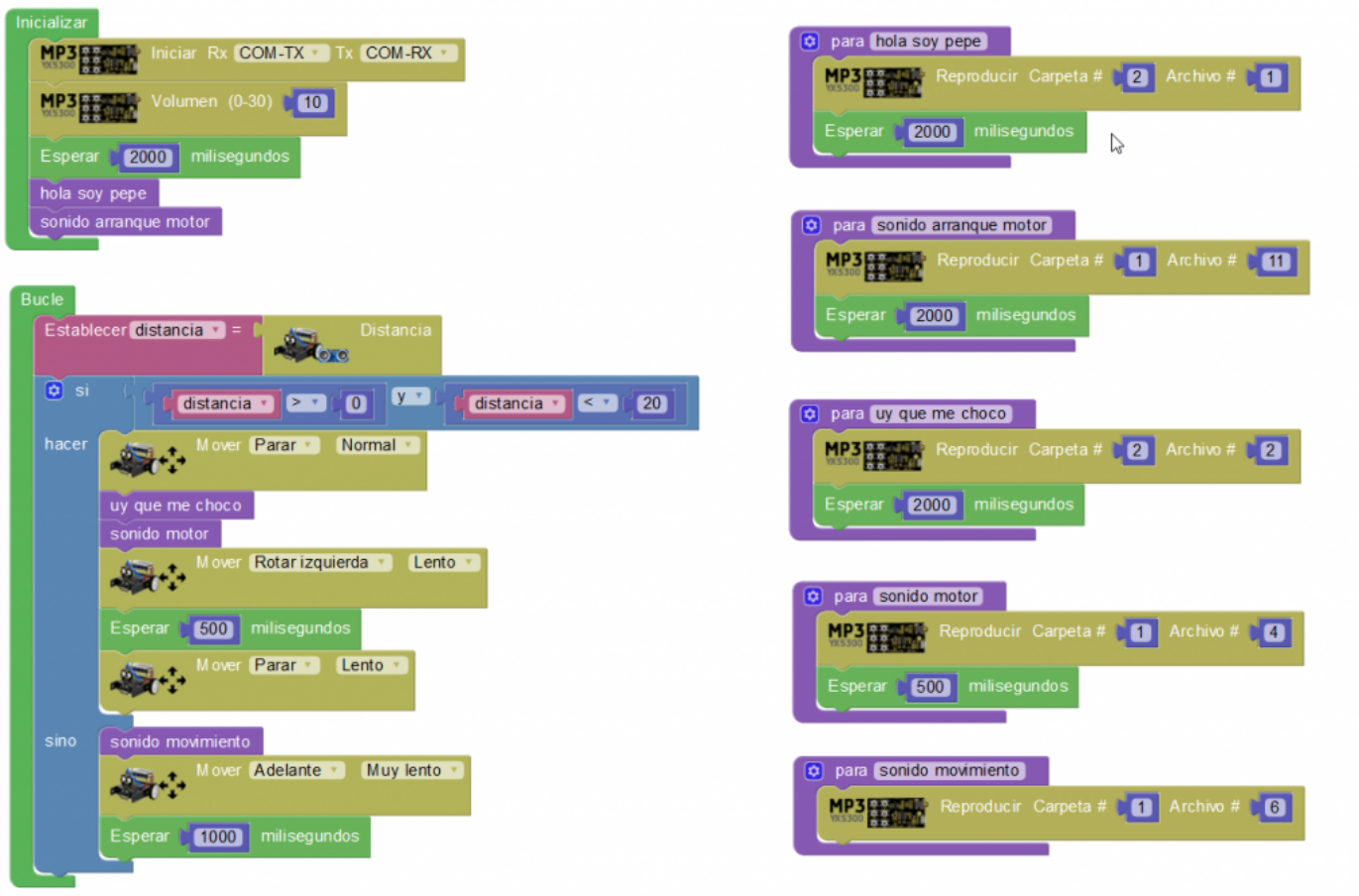


- **Control**: Cambia el estado del reproductor



- **Reiniciar**: Reinicia el módulo internamente.

Ejemplo: robot que "habla" y con efector especiales:



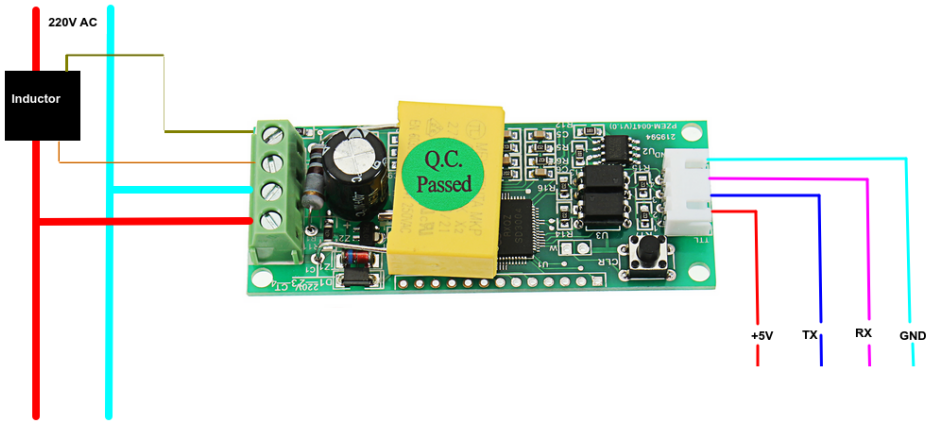
+Información y ejemplos:

http://arduinoblocks.didactronica.com/2019/12/modulo-reproductor-mp3-sd/

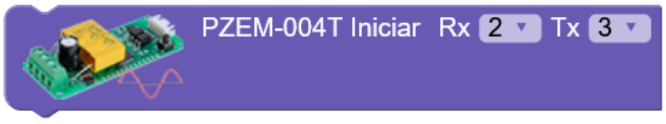
3.3.21 Domótica

Los sensores domóticos son sensores orientados a la industria, domótica y la automatización.

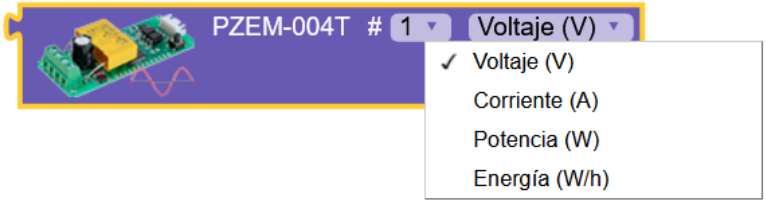
- **PZEM-004T:** es un sensor que permite medir voltaje, corriente y potencia de una instalación de corriente alterna. Los cables se deben atornillar para medir la tensión (V) y por otro lado deben pasar por la bobina toroidal para medir la corriente que circula (A).



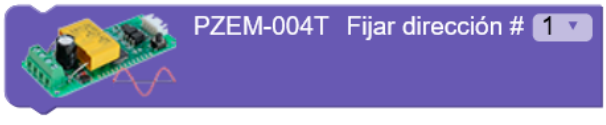
- **Iniciar**: Inicia el módulo indicando los pines de conexión para la comunicación serie con el módulo (RX/TX):



- **Obtener valor medido:** Leer el valor de Voltaje (Voltios), Corriente (Amperios) , Potencia (Watios) o Energía (W/h). Se debe especificar la dirección (#) del módulo, pues pueden conectarse varios sensores en paralelo a los mismos pines de comunicación.

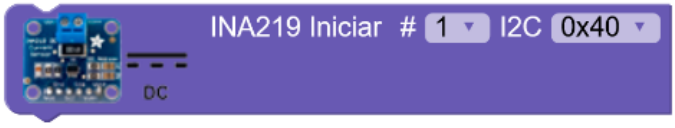


- **Fijar dirección:** El sensor PZEM-004T permite varios sensores conectados en paralelo en los mismo pines serie. Con este bloque podemos fijar la dirección del módulo conectado.

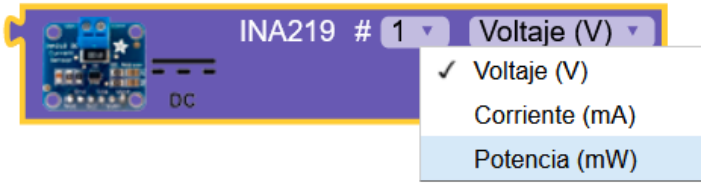


- **INA219:** Es un sensor para medir voltaje y corriente pero para corriente continua. Se conecta a Arduino mediante bus I2C.

    - **Iniciar**: Inicializar el módulo INA219 indicando la dirección I2C utilizada, de forma que permite utilizar varios módulos en el mismo bus con distintas direcciones.



    - **Obtener medición**: Obtiene los datos de Voltaje (Voltios) , Corriente (Amperios), Potencia (Watios)



3.3.22 Teclado / Ratón

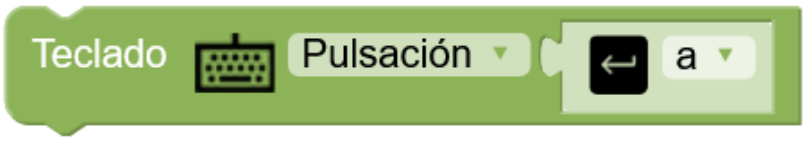
Los bloques de teclado / ratón solo están disponibles para Arduino UNO y Micro.

Estos dos modelos de Arduino son detectados como un dispositivo HID (Human Interface Device) , es decir, como un dispositivo de interfaz humana, lo que quiere decir que puede simular comportarse como un teclado o ratón. Esta funcionalidad nos permite aplicaciones tan interesantes como realizar un controlador de juegos con Arduino, enviar datos a una aplicación simulando la pulsación de teclas, mover el cursor del ratón del sistema operativo a partir de datos de sensores conectados a Arduino , etc.

- **Teclado**

    - **Enviar:** envía un texto a través del Arduino como si se hubiera tecleado desde un teclado conectado por USB, de forma que el texto aparecerá en la aplicación activa. Por ejemplo, si abrimos un procesador de texto irá apareciendo el texto enviado desde el Arduino como si lo hubiéramos tecleado.

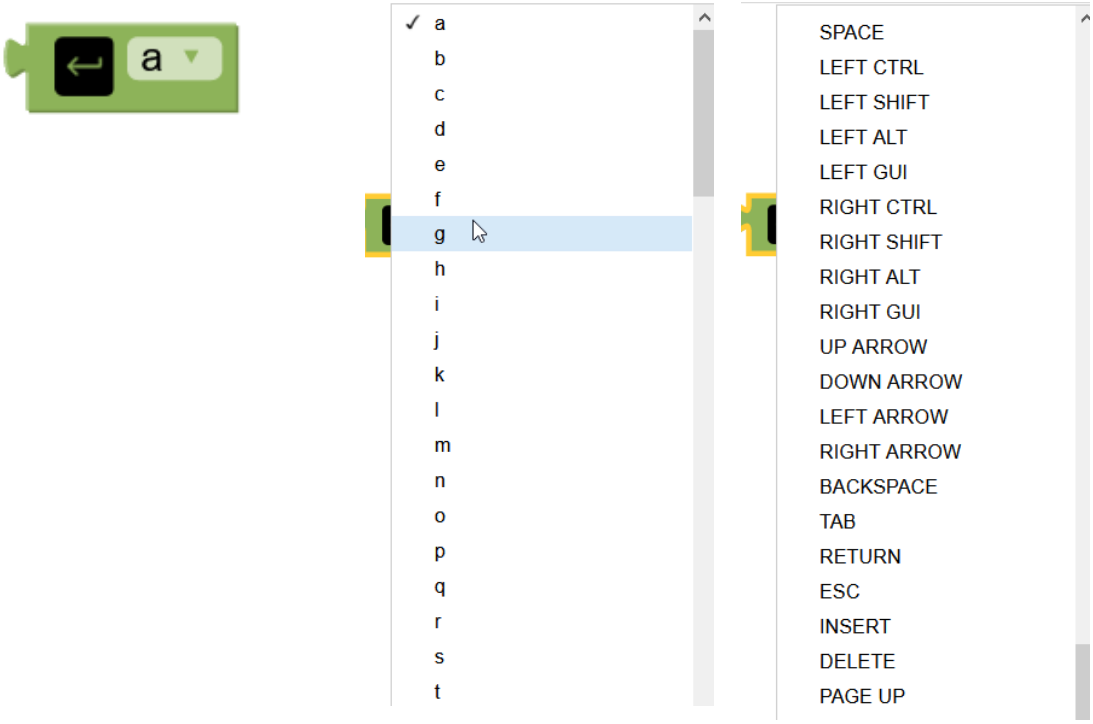
- **Pulsación tecla:** simula la pulsación de una tecla.



Hay 3 modos:

- *pulsación*: simula el proceso de pulsar y soltar la tecla
- *pulsar*: simula la pulsación (y se queda pulsada)
- *soltar*: simula el soltar la tecla

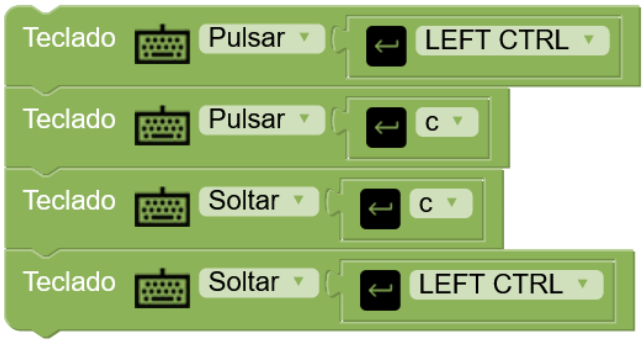
Tenemos un bloque con el listado de teclas posibles,tanto alfanuméricas como de códigos especiales:



Ejemplo: simular tecla de flechas izquierda/derecha del teclado

desde Arduino Leonardo con 2 pulsadores conectados

Ejemplo: simular la combinación de teclas CTRL+C (copiar)



Ejemplo: simular la combinación de teclas *CTRL+ALTt+SUPR*

Ejemplo: simular pulsación WIN+L (cerrar sesión windows)

cuando un sensor de distancia detecta que nos alejamos más de 2 metros

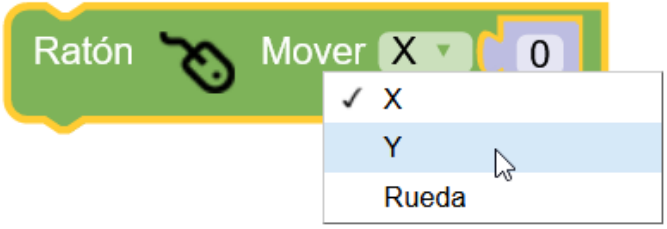
- **Ratón**

  - **Botón:** simula el click de uno de los botones del ratón

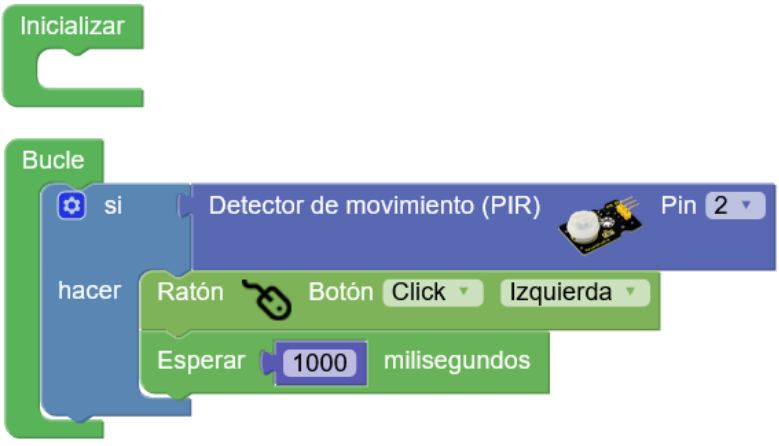
    Hay 3 modos:

    - *Click*: simula el proceso de pulsar y soltar el botón
    - *Pulsar*: simula la pulsación (y se queda pulsado)
    - *Solta*: simula el soltar el botón

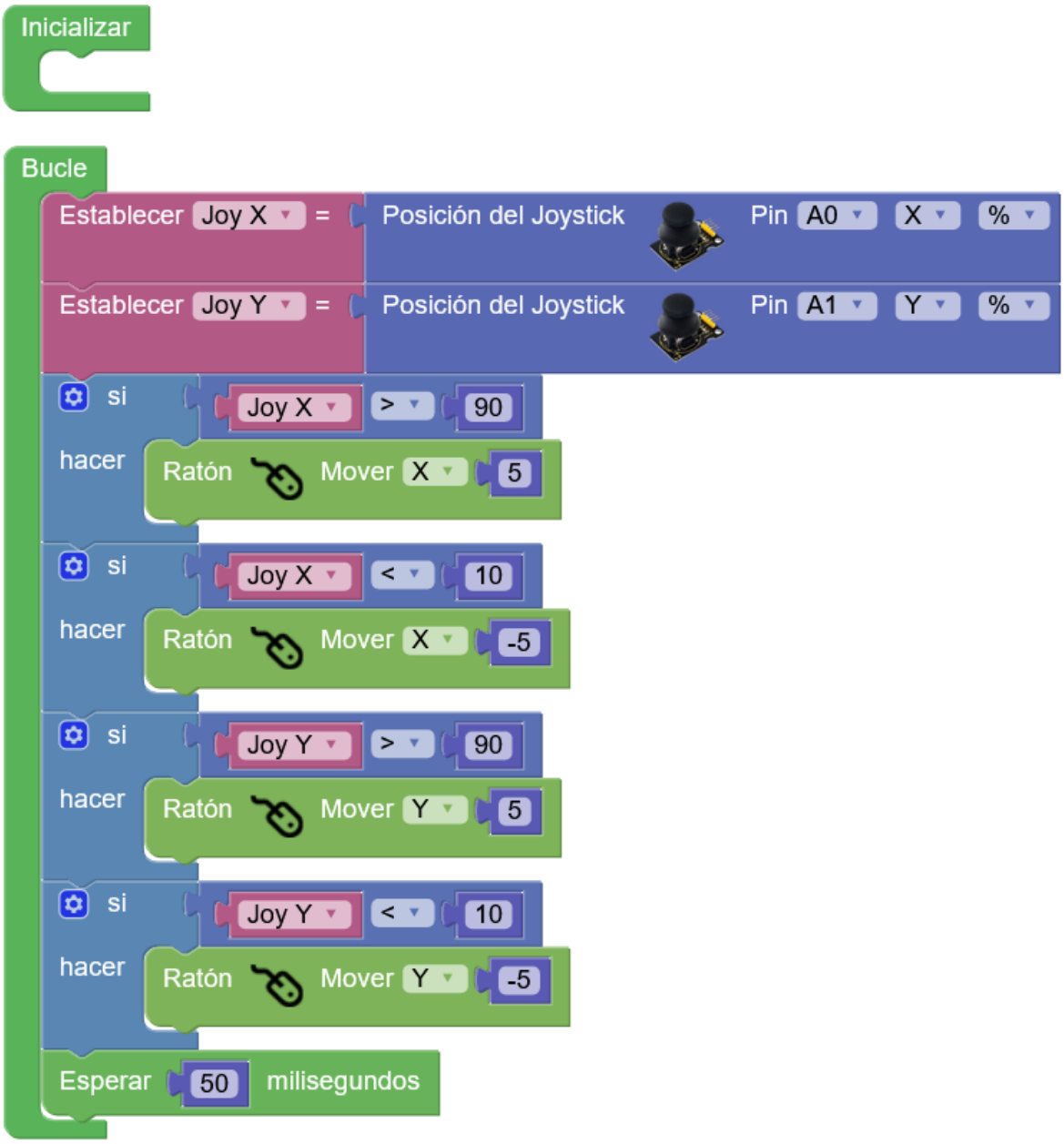
  - **Mover**: simula el desplazamiento del curso del ratón. Indicamos el valor positivo o negativo de incremento en píxeles. Siempre será un valor relativo a la posición actual.



Ejemplo: simular clicl del ratón al detectar presencia con un sensor PIR



Ejemplo: mover el ratón con un joystick conectado a Arduino



+Información y ejemplos:

http://arduinoblocks.didactronica.com/category/leonardo/

ANEXO I: Bloques incompatibles con bloqueos de tiempo.

Los siguientes bloques deben realizar tareas periódicas en segundo plano y por lo tanto debemos evitar situaciones en nuestro programa donde se bloquee la ejecución. Si un bloque o bloques consumen mucho tiempo de ejecución del procesador no dejarán realizar estas tareas en segundo plano y por lo tanto el funcionamiento no será correcto.

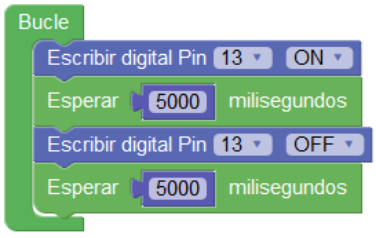
Los bloques GPS necesitan leer periódicamente los datos desde el módulo para obtener la información actualizada. Si utilizamos bloqueos en nuestro programa los datos GPS no serán válidos.

Los bloques MQTT gestionan la comunicación a través de la red Ethernet (TCP/IP) de forma continua en segundo plano, si bloqueamos la ejecución del programa no se realizará correctamente la comunicación.

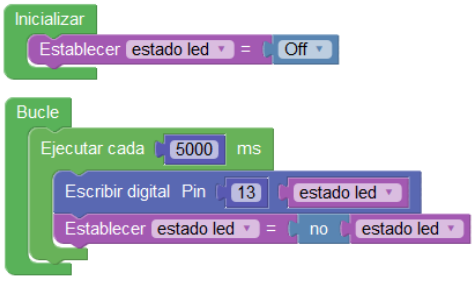
Por lo tanto en estos casos es recomendable seguir siempre un método de programación por tareas utilizando bloques del tipo “ejecutar cada” (ver apdo. 3.3.2)

Ejemplo: Parpadeo de led cada 5s con bloqueo y sin bloqueo

Utilizando bloqueo de tiempo:

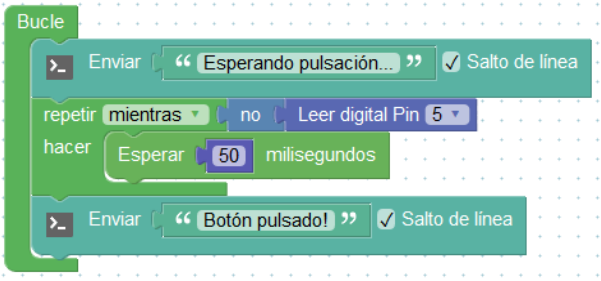


Solución sin bloqueos:

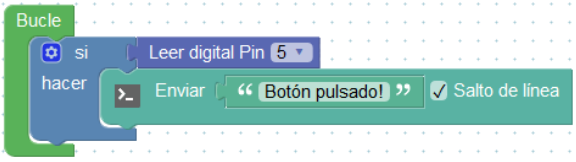


Ejemplo: Comprobar pulsación de un botón en el pin 5

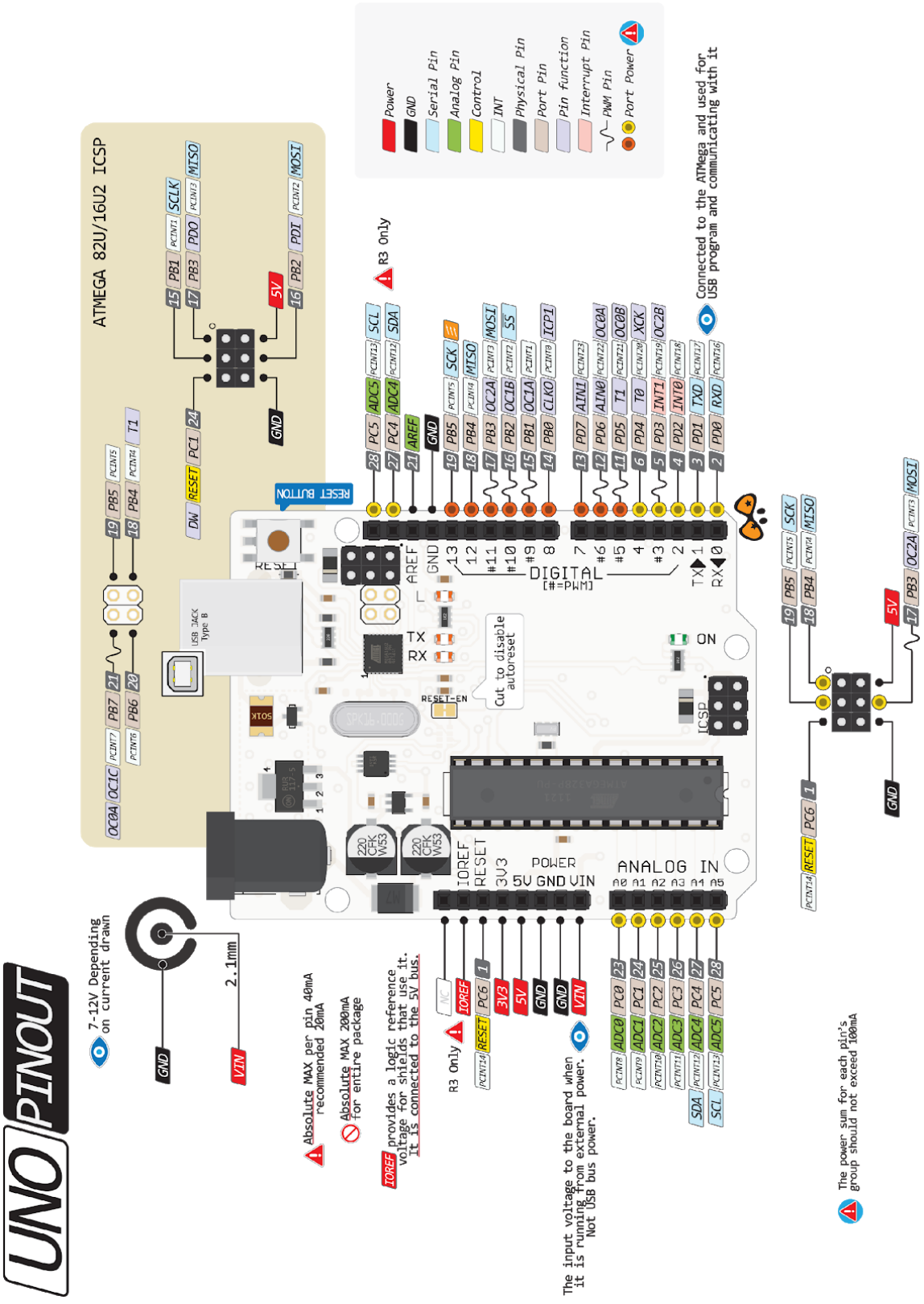
Utilizando bloqueo de tiempo:



Solución sin bloqueos:



ANEXO II: Arduino UNO R3 pinout



http://www.pighixxx.com/test/pinouts/boards/uno.pdf