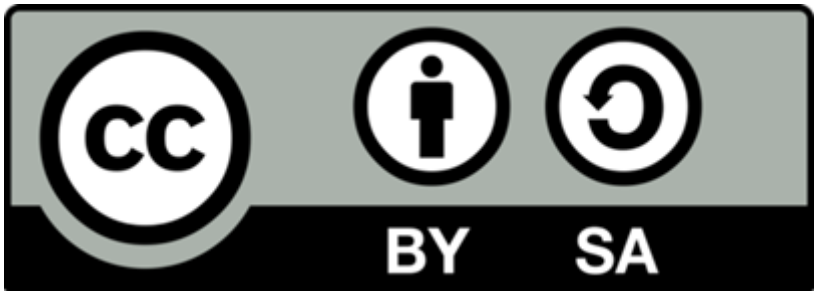
Introducción a la Robótica y la domótica con Arduino y Raspberry Pi

José Antonio Vacas Martínez

http://elCacharreo.com:

21 de abril de 2017



Índice general

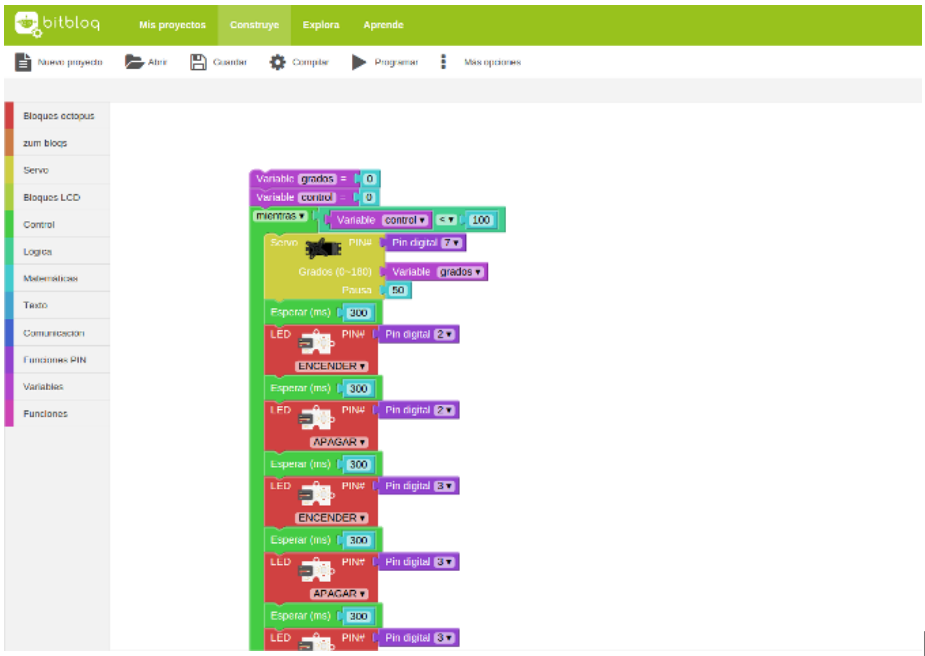
Programación de Arduino

Opciones de programación de Arduino

Veamos las diferentes opciones que tenemos para programar Arduino:

Arduino IDE

IDE Arduino|entorno multiplataforma que permite la edición, compilación y la programación de Arduino



|Bitbloq|
Entorno de desarrollo visual basado en la programación con bloques. Es muy sencillo e intuitivo de usar y solo necesitamos el navegador para usarlo. Existen alternativas offline como Visualino.

CodeBender

Codebender

Entorno de desarrollo totalmente basado en aplicaciones web, decir, sólo necesitamos un navegador web para poder desarrollar con Arduino. Permite la edición, compilación y la programación de Arduino. Podéis probarlo en https://codebender.cc/|

Figura 1: Arduino IDE

Figura 2: codebender

Entorno de programación Bitbloq

Bitbloq es un entorno de programación visual que nos permite crear programas para Arduino y placas compatibles y transferir los mismos a las placas de una forma sencilla.

Podemos acceder directamente desde su web http://bitbloq.bq.com/

Como verás cuando entres a su página, funciona con Chrome en todos los sistemas operativos, y el se encarga de avisarte si necesitas drivers o cambiar algo en tu sistema.

A lo largo de estos vídeos veremos algunas de sus características más importantes. Puedes encontrar más tutoriales en la página oficial de bitbloq

Introducción a la programación con Bitbloq

Bitbloq es un entorno de programación visual por bloques que nos permite programar nuestra placa Arduino o compatible de forma sencilla, evitando la complejidad de las sentencias C++.

Además nos permite programar nuestro Arduino sin instalar (prácticamente) nada en nuestro ordenador.

Veamos un ejemplo sencillo:

vídeo de ejemplo básico

ejemplo Parpadeo

vídeo “Cómo ver código C++ de un programa bitbloq”

Desde bitbloq siempre podemos ver el código Arduino generado. De momento no podemos modificar este código pero si copiarlo y llevarlo al IDE de Arduino

vídeo “Transfiriendo el programa bitbloq a Arduino”

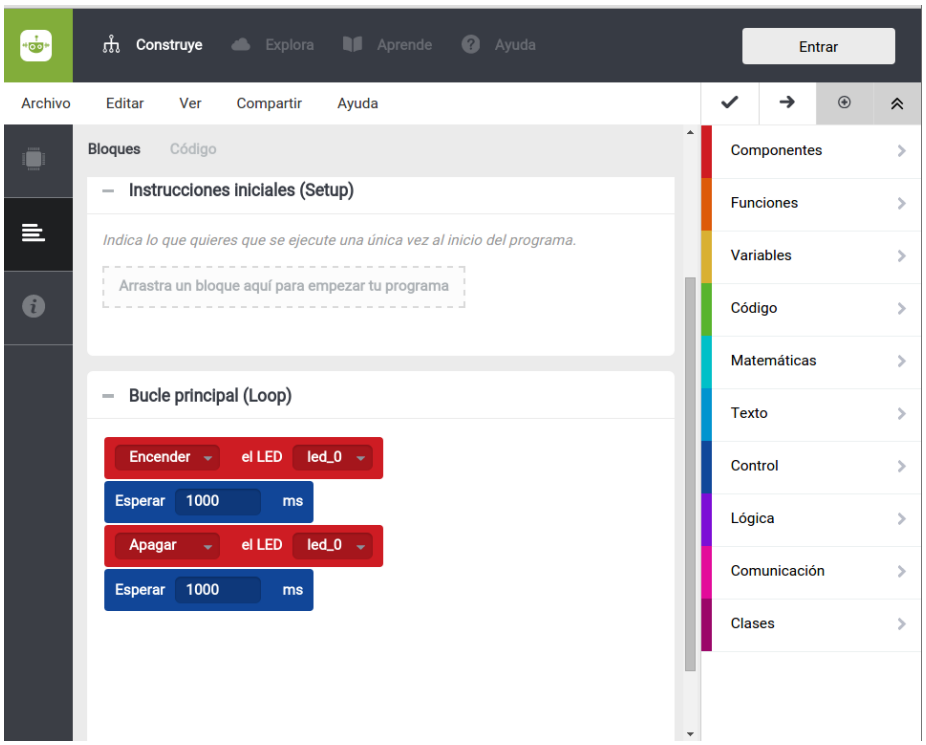


Figura 3: parpadeo

Bitbloq nos permite programar nuestro Arduino sin instalar (prácticamente) nada en nuestro ordenador. Sólo tenemos que pulsar sobre el botón cargar lo que hace que se compile el código, se detecte la placa y se envíe el programa a nuestro Arduino

Sentencias de control

video sobre sentencias de control

Las sentencias de control son aquellas que nos permite modificar el orden o el modo en el que se ejecutan los bloques de nuestro programa.

Variables

Para utilizar las sentencias de control necesitaremos el concepto de variables: que no es otra cosa que un lugar donde almacenar un valor que puede se modificar si así lo queremos

video

Con las variables podemos realizar operaciones matemáticas

video

ejemplo

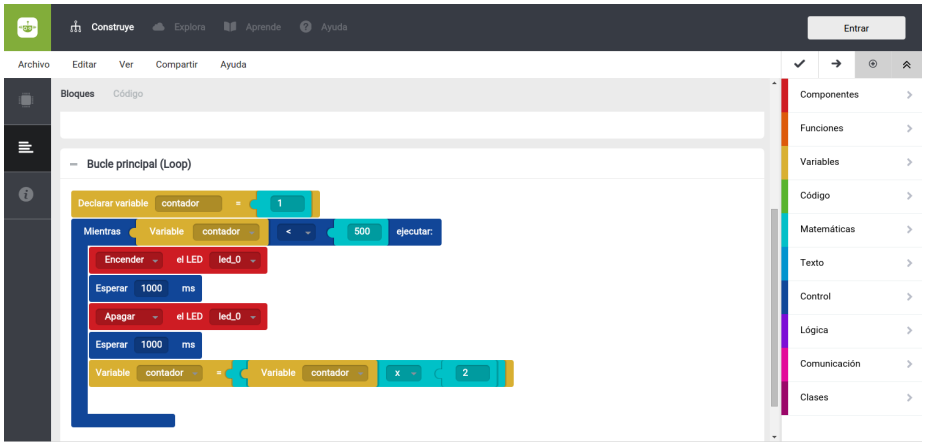


Figura 4: ejemplo

Bucle for

[vídeo] [ejemplo]

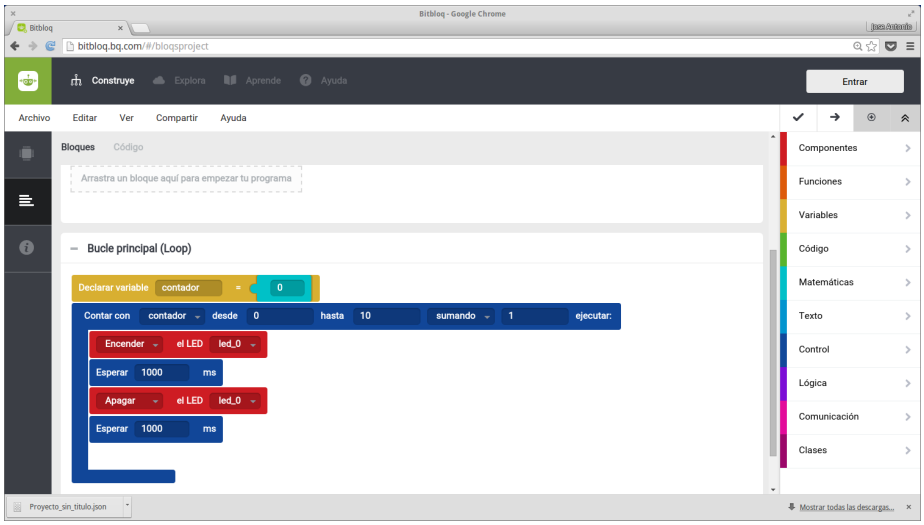


Figura 5: bucle for

El bucle **for** permite repetir un conjunto de pasos un número de veces determinado.

Necesitamos declarar una variables que actuará como contador y definir el valor inicial que tendrá la variable y el final, realizándose tantos como pasos como valores enteros haya entre ambas.

Bucle while

[vídeo]

[ejemplo]

Usaremos la sentencia de control **while** para los bucles donde el número de veces que se repite no está definido desde el principio

Bloque if : sentencias condicionales

Vídeo sobre sentencias condicionales

vídeo

ejemplo

Las sentencias condicionales permiten ejecutar un código y otro según se cumpla o no una determinada condición. Esta condición será una validación que

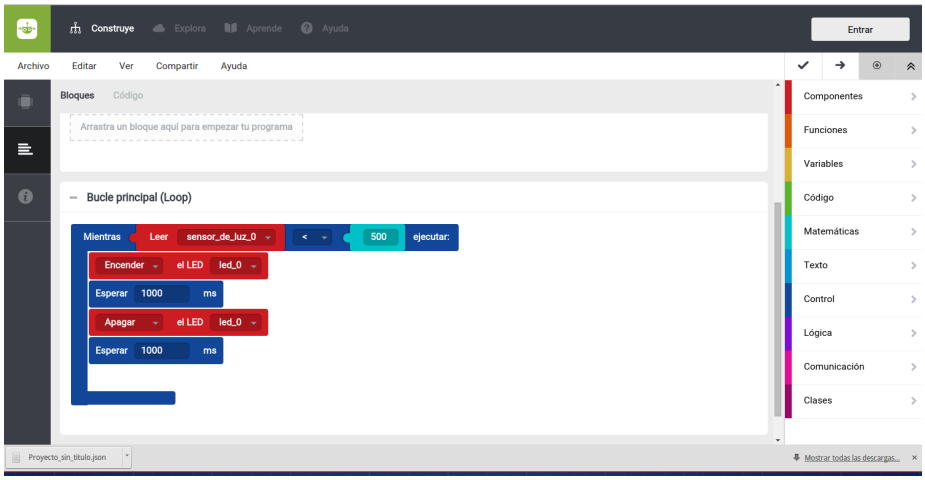


Figura 6: bucle while

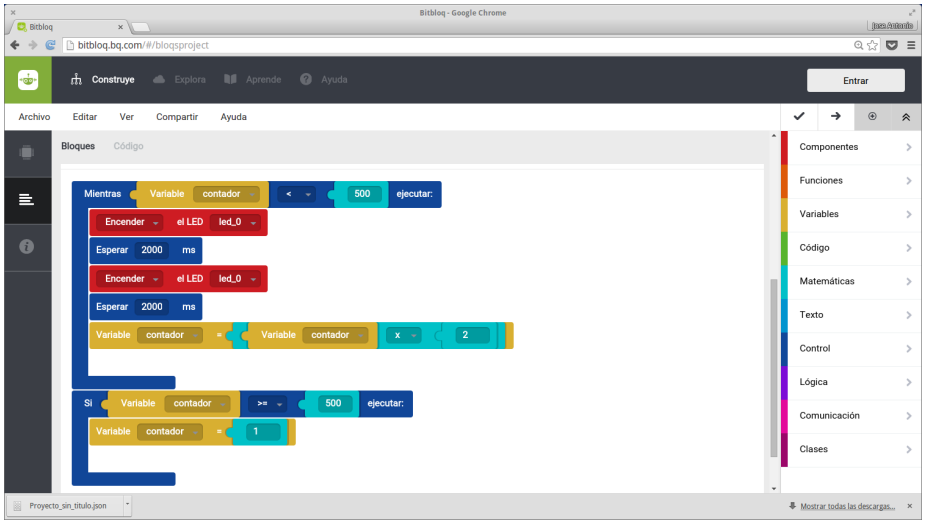


Figura 7: condicional

definiremos con operandos.

Podemos hacer que en caso de que se cumpla se ejecute un código (es el bloque if) y en caso de que no se cumpla la condición se ejecute otro (bloque else). Veamos un ejemplo

video ejemplo

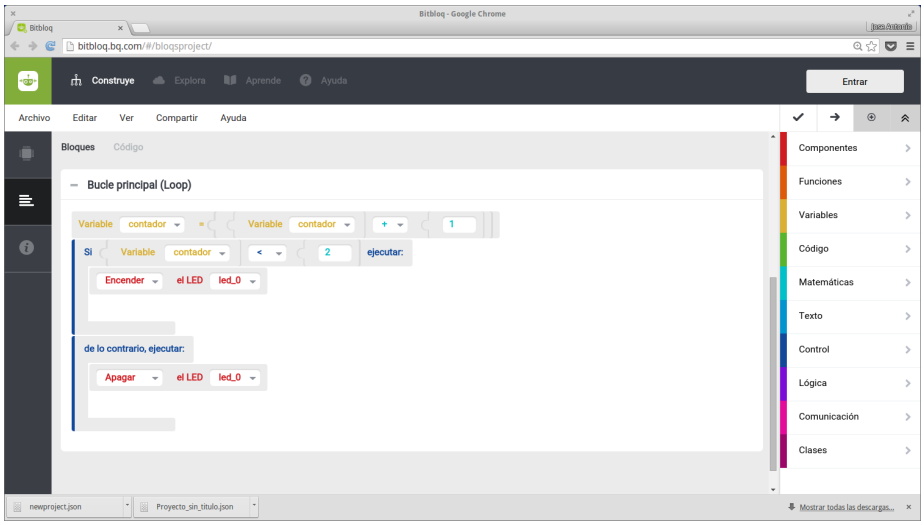


Figura 8: ejemplo

Condicionales-complejas

[vídeo] [ejemplo]

La condición que determina si se ejecuta un bloque u otro o si salimos de un bloque while puede contener varias comprobaciones.

Entre estas condiciones utilizaremos operadores lógicos que pueden ser AND o OR.

- Estas condiciones se tendrán que cumplir todas en el caso del operador AND.
- Con que se cumpla una de ellas se dará por válida toda la condición.

Envio de datos al pc

Video

[vídeo] [ejemplo]

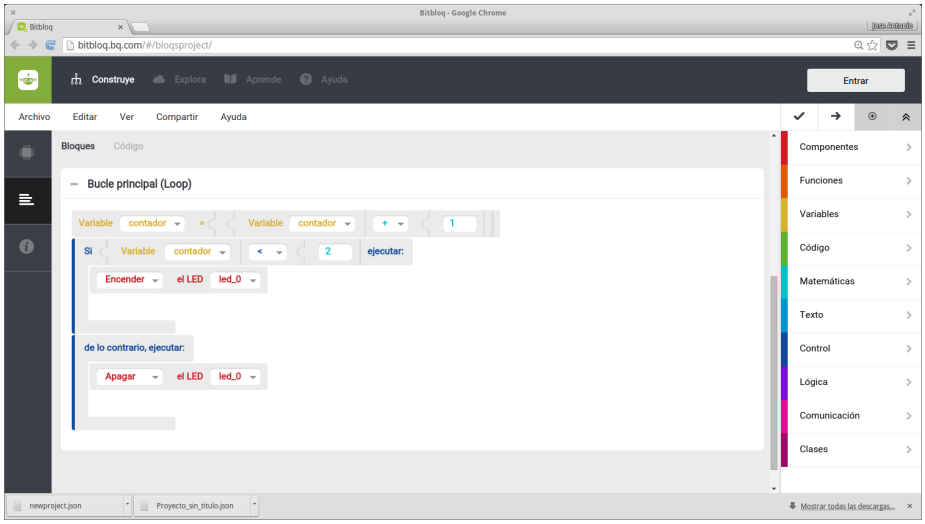


Figura 9: Condiciones_lógicas

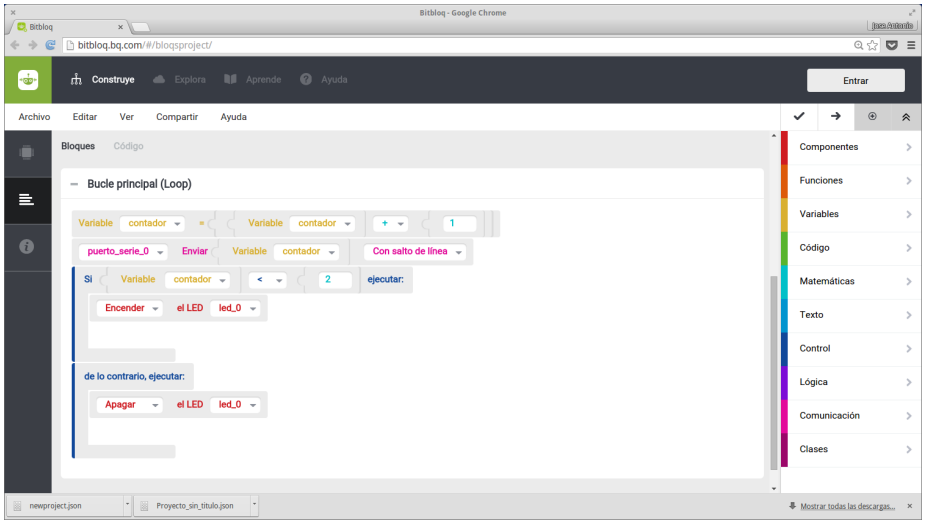


Figura 10: serial

Podemos enviar contenidos entre nuestra placa y el PC usando las sentencias de comunicaciones. Usaremos print para enviar algo (puede ser el valor de una variable o un texto) al PC o println para enviar y pasar a la siguiente línea.

Variables locales vs variables globales

Video

vídeo

ejemplo

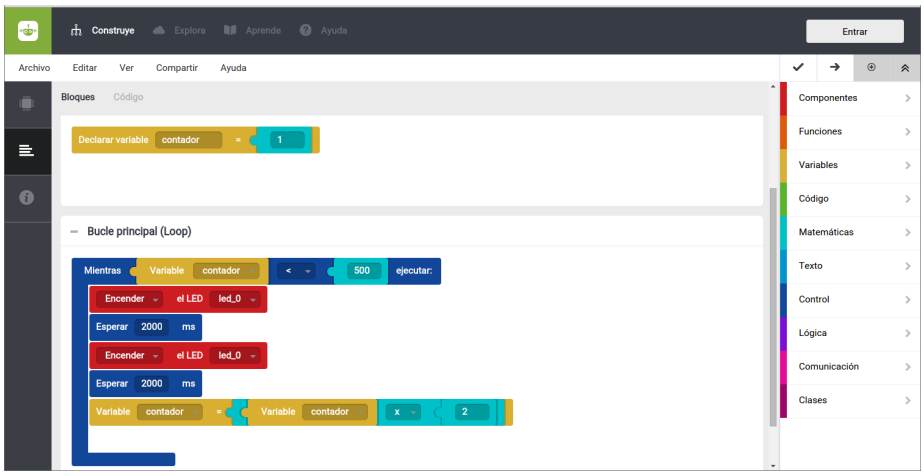


Figura 11: ejemplo

Podemos definir variables locales o globales. Una variable global estará definida y por tanto mantendrá su valor en todo el programa, mientras que una variable local solo se definirá donde se haya declarado.

Las variables globales mantienen su valor entre las distintas iteraciones que se realizan del programa.

Creando bucles sin sentencias de control

Podemos usar la forma cíclica (y unas variables globales) en la que se ejecutan los programas en Arduino para hacer un bucle sin más estructuras de control que una simple variable global

ejemplo

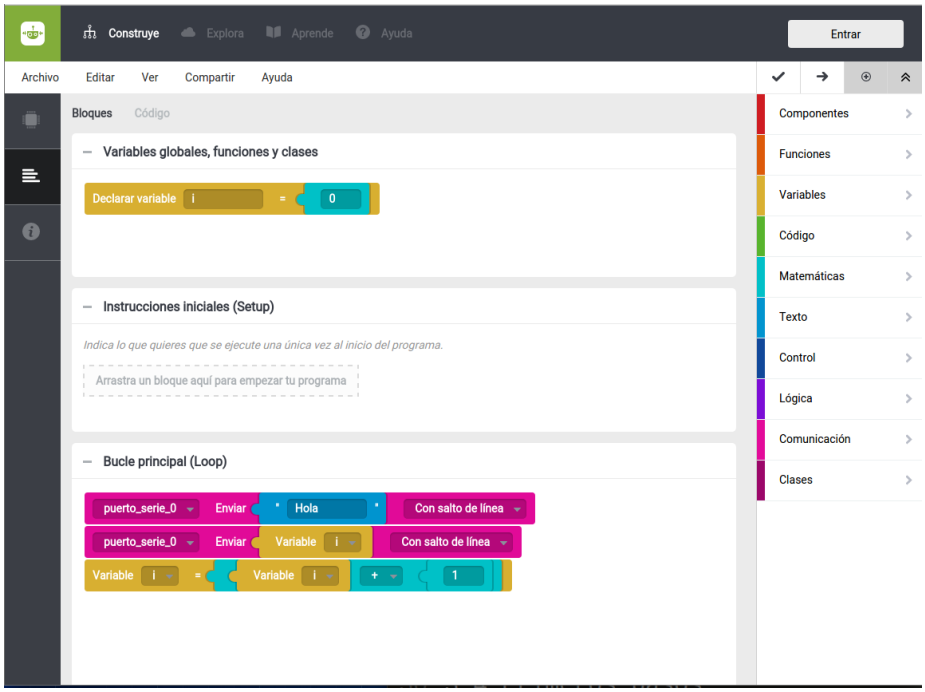
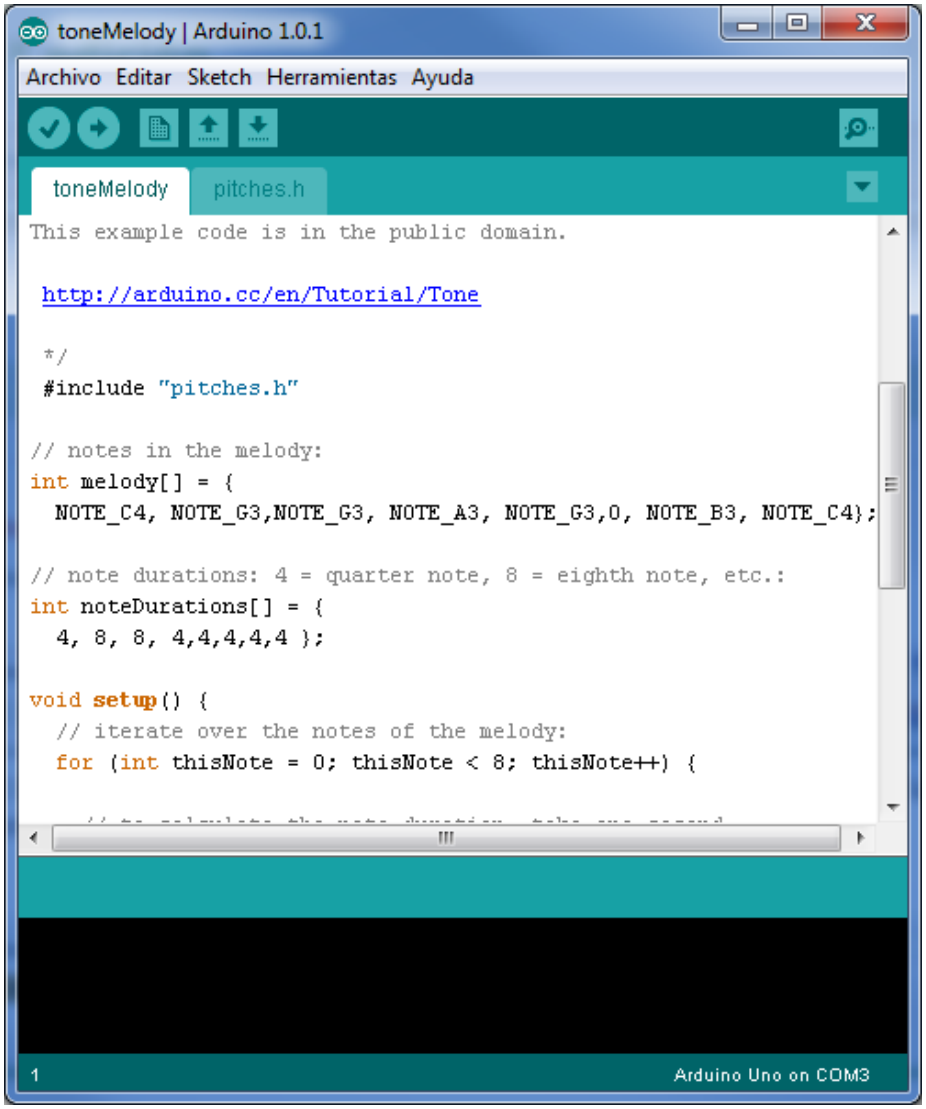


Figura 12: ejemplo

Usando el IDE de Arduino



El Entorno de Programación Integrado (IDE) de Arduino es sencillo y a la vez potente, incluyendo casi todas las funcionalidades que uno espera de un IDE actual.

Es un IDE multiplaforma, estando disponible para descargar directamente para Windows, Linux y MaxOS. Al ser una herramienta de código libre podemos descargar también el código fuente para compilarlo.

Está desarrollado en Java y deriva del IDE de Processing y de Wiring.

Ha sido diseñado teniendo en mente que sus usuarios no son expertos desarrolladores y que lo esencial es que los recien llegados sean capaces de familiarizarse con él.

Incluye un editor de código que resalta la sintaxis, indicando cuando es correcta, indicando si las llaves están bien emparejadas y organizando el código de forma clara.

Además el entorno incluye todo lo necesario para compilar y descargar las aplicaciones al microntrolador, sin necesidad de que el usuario conozca la complejidad de este proceso.

El entorno viene con una librería llamada Wiring (procedente de dicho proyecto) que facilita el programar pensando en las entradas y las salidas.

El código que se desarrolla es C/C++, aunque el usuario solo incluye las funciones setup y loop

Usaremos las versión 1.8.1 (o superior si se liberara)

El Entorno de Programación Integrado (IDE) de Arduino es sencillo y a la vez potente, incluyendo casi todas las funcionalidades que uno espera de un IDE actual.

Es un IDE multiplaforma, estando disponible para descargar directamente para Windows, Linux y MacOS. Al ser una herramienta de código libre podemos descargar también el código fuente para compilarlo.

Está desarrollado en Java y deriva del IDE de Processing y de Wiring.

Ha sido diseñado teniendo en mente que sus usuarios no son expertos desarrolladores y que lo esencial es que los recien llegados sean capaces de familiarizarse con él.

Incluye un editor de código que resalta la sintaxis, indicando cuando es correcta, indicando si las llaves están bien emparejadas y organizando el código de forma clara.

Además el entorno incluye todo lo necesario para compilar y descargar las aplicaciones al microcontrolador, sin necesidad de que el usuario conozca la complejidad de este proceso.

El entorno viene con una librería llamada Wiring (procedente de dicho proyecto) que facilita el programar pensando en las entradas y las salidas.

El código que se desarrolla es C/C++, aunque el usuario solo incluye las funciones setup y loop

Usaremos las versión 1.8.1 (o superior si se liberara)

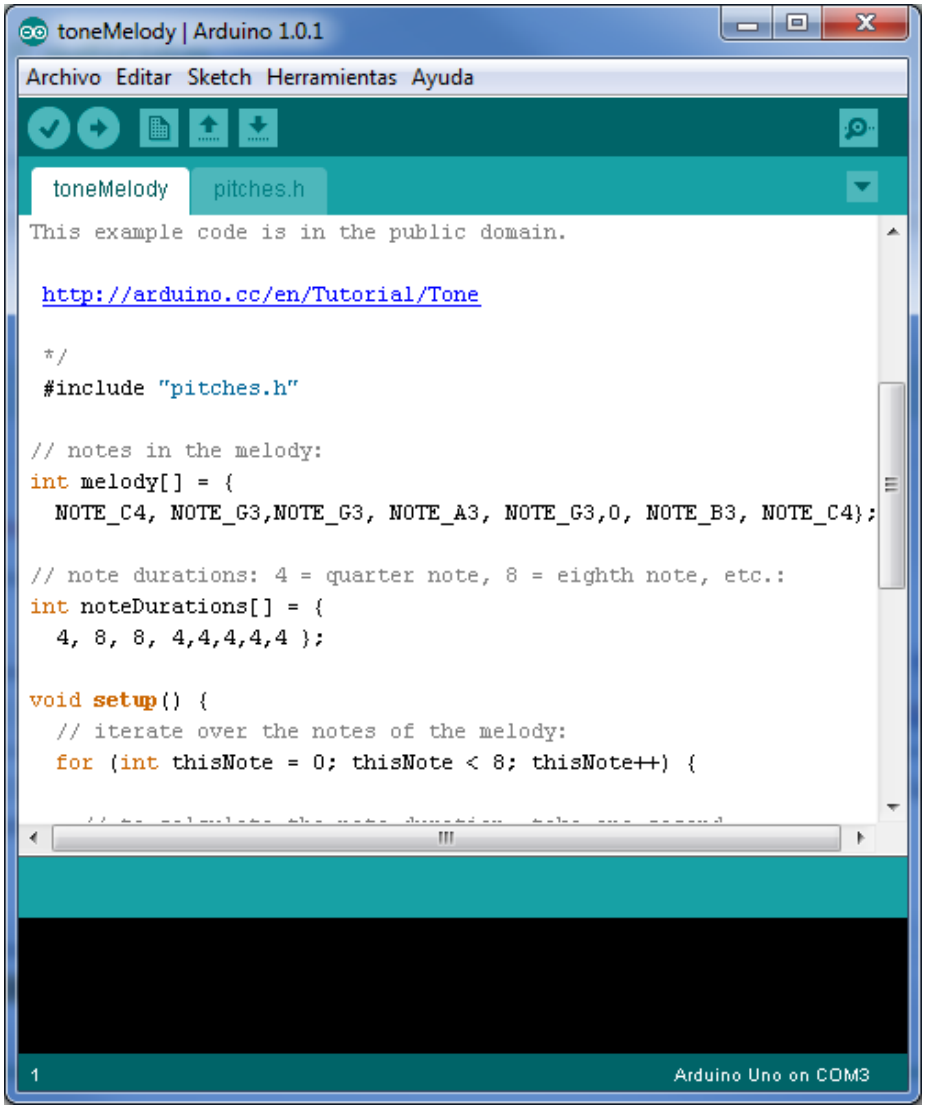


Figura 13: Arduino IDE

Instalación del entorno

Enlace para descarga

Descargando el programar a Arduino

Vamos a comenzar viendo paso a paso una descarga de un programa a Arduino.

Intentaremos hacer una lista paso a paso para poder detectar errores en el caso de que se produzcan

Enlace para descarga

En el siguiente vídeo podemos ver el paso a paso de la descarga de un programa a una placa Arduino

Comenzando a programar con Arduino

Comenzando a programar

Como hemos visto podemos encontrar multitud de ejemplos en el propio IDE de Arduino, de ellos podemos aprender la sintaxis y las posibilidades que el entorno tiene.

Vamos a hacer unos ejemplos nosotros que sin entrar en detalles técnicos nos van a permitir ir haciéndonos con el entorno.

Programa con contador

Introducimos el siguiente programa en el entorno. En el se irá incrementando la variable i y se enviará su valor al pc por el puerto de comunicaciones.

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

int i=0;
void loop() {
  Serial.print("hola ");
  Serial.println(i);
  i=i+1;
}
```

Vamos a analizar el programa:

- Todos los métodos que empiezan con Serial, se refieren a comunicaciones con el pc, en la línea Serial.begin(9600) estamos diciendo que vamos a iniciar (begin) las comunicaciones a una velocidad de 9600bps (bit por segundo). Serial.print lo que lleva a continuación al pc. Si utilizamos println le estamos indicando de después de imprimir salte a la siguiente línea.
- Sabemos el método setup se ejecuta sólo al principio una vez para configurar todo. El método loop en cambio se ejecuta de forma repetitiva, con lo que el resultado será que nuestra variable i se va incrementando a cada paso en una unidad (podíamos haber abreviado el programa sustituyendo la linea i=i+1; por i++ que significa lo mismo

Descargamos el programa  depués de haber comprobado que tenemos bien seleccionado nuestro modelo de placa y el puerto donde está conectada. Una vez descargada, abrimos la consola serie para ver el resultado

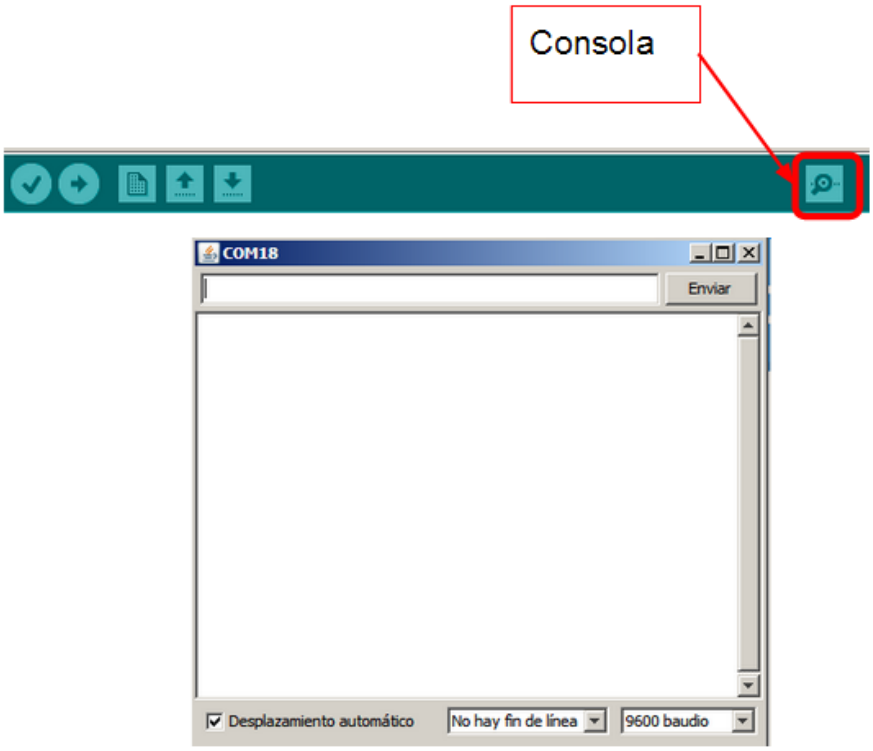


Figura 14: Consola

Cuidando de que la velocidad seleccionada será la misma que hemos indicado en el programa

Veremos como se producen una sucesión de línea con el texto “hola 1….”

En bitbloq hacemos el siguiente programa

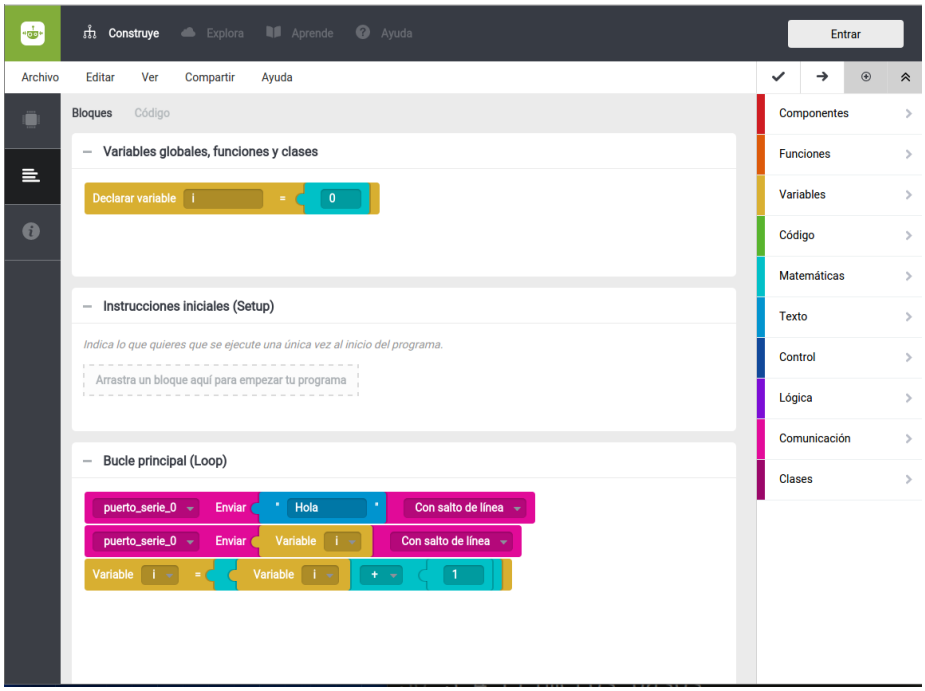


Figura 15: bucle variables globales

Tiempos

Podemos ver las líneas se suceden a toda velocidad en el programa anterior. Ahora mismo el micro está funcionando todo lo rápido que puede (la única limitación es la velocidad a la que se envían datos al pc)

Para introducir una pausa podemos usar las funciones delay y delayMicroseconds. Veamos su uso

- **delay(milisegundos)**: espera el número de milisegundos indicado
- **delayMicroseconds(microsegundos)**: espera los microsegundos indicados El número máximo que soporta es 16383

Así con solo introducir la siguiente línea detrás del incremento de la variable i, conseguiremos que el programa se retrase

```
delay(1000);
```

Ya que estamos hablando de tiempo veamos que otras funciones tiene Arduino relacionadas con el mismo:

- **long millis()** : devuelve el número de milisegundos desde que se encendió la placa. Se vuelve a poner a cero cada 50 días aproximadamente.
- **long micros()** : devuelve el número de microsegundos desde que se encendió la placa. Se vuelve a poner a cero cada 70 minutos aproximadamente. **Tiene una resolución de 4 microsegundos.**

Con todo esto vamos a hacer un programa que envie cada segundo el número de milisegundos que han pasado desde la ultima iteración. Intenta hacer el programa por ti mismo

Quedaría algo así:

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

long mAnteriores; // Lo usaremos para almacenar el valor anterior\

void loop() {
  long mActuales=millis(); // leemos el valor actual de millis

  Serial.println(mActuales-mAnteriores);
  // restamos el anterior del actual y esos son los que han pasado**

  mAnteriores=mActuales;
  // guardamos el dato actual para la siguiente iteración

  delay(1000);
}
```

Sonido

Arduino puede reproducir sonido sin más que conectar un altavoz (mejor un altavoz piezoelectrico de los que tienen los pcs)

Uno de los terminales del altavoz a la salida digital que usaremos y el otro a GND. Dependiendo de las características del altavoz pudiera ser conveniente usar una resistencia entre el altavoz y la salida digital.

Podemos hacer sonidos con las siguientes funciones:

```
tone(pin, frecuencia)
tone(pin, frequencia, duracion)
```

- Genera una señal cuadrada de la frecuencia y duración definidas.

Figura 16: buzzer

Para afinar las frecuencias a las notas podemos consultar tablas como esta:

Frecuencias (en hertzios) de las notas musicales

	Oc. 0	Oc. 1	Oc. 2	Oc. 3	Oc. 4	Oc. 5	Oc. 6	Oc. 7	Oc. 8
Do		32,70	65,41	130,81	261,63	523,25	1046,50	2093,00	4186,01
Do#		34,65	69,30	138,59	277,18	554,37	1108,73	2217,46	
Re		36,71	73,42	146,83	293,66	587,33	1174,66	2349,32	
Re#		38,89	77,78	155,56	311,13	622,25	1244,51	2489,02	
Mi		41,20	82,41	164,81	329,63	659,26	1318,51	2637,02	
Fa		43,65	87,31	174,61	349,23	698,46	1396,91	2793,83	
Fa#		46,25	92,50	185,00	369,99	739,99	1479,98	2959,96	
Sol		49,00	98,00	196,00	392,00	783,99	1567,98	3135,96	
Sol#		51,91	103,83	207,65	415,30	830,61	1661,22	3322,44	
La	27,50	55,00	110,00	220,00	440,00	880,00	1760,00	3520,00	
La#	29,14	58,27	116,54	233,08	466,16	932,33	1864,66	3729,31	
Si	30,87	61,74	123,47	246,94	493,88	987,77	1975,53	3951,07	

Figura 17: tabla frecuemcia

- Interfiere con algunas funciones de los pines el 3 y 11 (el pwm que veremos más adelante)

  `noTone(pin)`

- detiene la ejecución del tone actual

- se utiliza para usar varios pines generando sonido

Figura 18: como hacer sonido Arduino

Podemos ver cómo se usan estas funciones en el ejemplo incluido en Arduino sobre el tema. Accederemos a él según la imagen siguiente

En bitbloq existen 2 formas de generar sonidos

- Reproducir notas musicales: podemos escoger la nota que vamos a reproducir y su duración
- Seleccionar la frecuencia exacta que queremos reproducir y su duración

Se propone como ejercicio voluntario realizar alguna canción conocida, preferiblemente algo friki con Arduino. Comparte en los foros tu composición/interpretación.

Frecuencias (en hertzios) de las notas musicales

	Oc. 0	Oc. 1	Oc. 2	Oc. 3	Oc. 4	Oc. 5	Oc. 6	Oc. 7	Oc. 8
Do		32,70	65,41	130,81	261,63	523,25	1046,50	2093,00	4186,01
Do#		34,65	69,30	138,59	277,18	554,37	1108,73	2217,46	
Re		36,71	73,42	146,83	293,66	587,33	1174,66	2349,32	
Re#		38,89	77,78	155,56	311,13	622,25	1244,51	2489,02	
Mi		41,20	82,41	164,81	329,63	659,26	1318,51	2637,02	
Fa		43,65	87,31	174,61	349,23	698,46	1396,91	2793,83	
Fa#		46,25	92,50	185,00	369,99	739,99	1479,98	2959,96	
Sol		49,00	98,00	196,00	392,00	783,99	1567,98	3135,96	
Sol#		51,91	103,83	207,65	415,30	830,61	1661,22	3322,44	
La	27,50	55,00	110,00	220,00	440,00	880,00	1760,00	3520,00	
La#	29,14	58,27	116,54	233,08	466,16	932,33	1864,66	3729,31	
Si	30,87	61,74	123,47	246,94	493,88	987,77	1975,53	3951,07	

Figura 19: Tabla de frecuencias

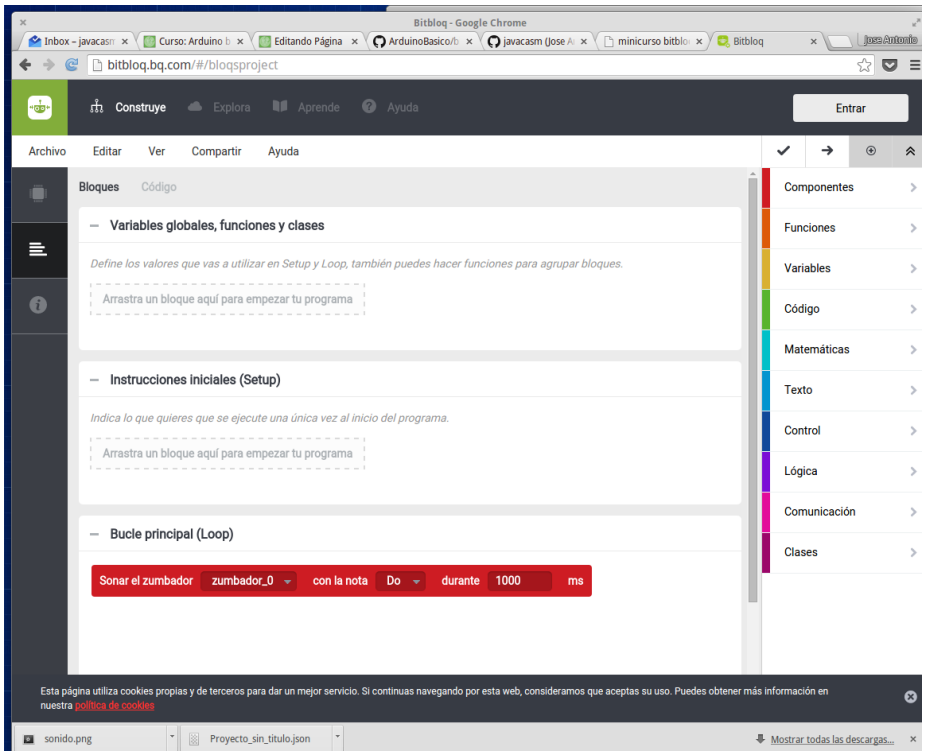


Figura 20: sonido

Librerías

Una librería no es más que un conjunto de código empaquetado y al que podemos llamar desde nuestro programa

Facilita la tarea de desarrollar pues aprovechamos un código ya existente y nos abstrae y encapsula la dificultad de la tarea.

Para usarlas solo tenemos que importarlas desde nuestro código con un #include <...>

Si queremos instalarlas sólo hay que descargarlas y copiarlas en la carpeta libraries del directorio donde se guardan nuestros sketches de Arduino. Creamos un directorio con el nombre de la librería.

En el caso de Bitbloq, ahora mismo no se pueden usar librerías, de forma manual.

Librerías básicas

- EEPROM - permite leer y escribir en almacenamiento duradero
- Ethernet - para conectar a internet
- Firmata - comunicaciones usando un protocolo concreto
- LiquidCrystal - manipulación de LCD
- SD - lectura y escritura en tarjetas SD
- Servo - control de Servos
- SPI - comunicaciones con dispositivos usando SPI
- SoftwareSerial - permite comunicaciones serie por otros pines
- Stepper - control de motores paso a paso
- Wire - comunicaciones I2C

Otras Librerías

Communication (networking and protocols):

- Messenger - para procesar mensajes de texto provenientes del PC
- NewSoftSerial - una versión mejorada de SoftwareSerial
- OneWire - permite la comunicación con dispositivos de Dallas (fabricante de dispositivos electrónicos) que usan este protocolo. Veremos un ejemplo en el módulo de comunicaciones.
- PS2Keyboard - Lee caracteres desde un teclado PS2.
- Simple Message System - facilita la tarea de enviar mensajes entre arduino y el PC
- SSerial2Mobile - envía mensajes de texto y email desde el movil (via comandos AT)
- Webduino - servidor web para usar con arduino ethernet
- X10 - Envia señales del protocolo X10**usando los cables electricos domésticos

- XBee - Comunicaciones XBees en modo API**
- SerialControl - Control remoto de otros Arduinos por medio de comunicaciones serie

Sensing:

- Capacitive Sensing - permite usar dos entradas como sensors capacitivos
- Debounce - para hacer debounce en entradas

Pantallas y LEDs:

- Improved LCD library mejora la libreria LCD library
- GLCD - permite manejar LCDs graficas basadas en el chip KS0108 o equivalente.
- LedControl - maneja matrices de led con chips MAX7221 o MAX7219.
- LedControl - librería alternativa para los chips anteriores.
- LedDisplay - controla pantallas basadas en HCMS-29xx con scroll.

Por estar basado en lenguajes estándar podemos portar (migrar) fácilmente librerías disponibles para otros sistemas

Estas librerías son compatibles con Wiring, los enlaces apuntan a la documentación. - Matrix - facilita el trabajo con Matrices de Leds - Sprite - Permite usar image básicas (sprite) en matrices leds

Frequency Generation and Audio:

- Tone - genera sonidos de la frecuencia dada

Motores y PWM:

- TLC5940 - facilita el usar el chip TLC594016 canales con PWM de 12 bit PWM.

Manejo del tiempo:

- DateTime - facilita el trabajar con fechas y horas.
- Metro - ayuda a medir intervalos temporales fijos
- MsTimer2 - permite ejecutar tareas cada N milisegundos (usa 2 temporizadores).

Utilidades:

- PString - clase ligera para imprimir
- Streaming - simplifica la manera de imprimir caracteres

Escribiendo nuestra propia librería

- Para crear nuestra librería tenemos que generar nuestro código en C++
- Crearemos una clase con nuestro código
- Usaremos un fichero "nuestralibreria.h" donde declararemos nuestro interface (obligatorio que exista el constructor)
- Un fichero "nuestralibreria.cpp" con nuestro código
- Incluiremos todos los ficheros en una carpeta "nuestralibreria" en la carpeta libraries del directorio de usuario
- Cerramos y abrimos el entorno Arduino para que la recompile y ya está disponible.

Ejemplo: Librería Servo

Nos permite controlar hasta 12 servos en un Arduino UNO y 48 en un Arduino Mega

- attach(pin) : conecta el objeto servo con el pin dado
- write(angul**0**) : establece la posición del servo
- read() : devuelve la posición del servo
- attached() : comprueba si está conectado
- detach(): desconecta el pin del servo

Veamos un ejemplo

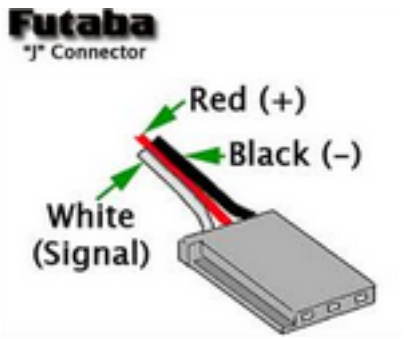


Figura 21: Servo Pinout

```
#include <Servo.h>

Servo myservo; // creamos un objeto servo
int potpin = 0; // pin donde está conectado el potenciómetro

void setup() {
  myservo.attach(9); // asignamos el pin 9 a nuestro servo
}
```

```
void loop() {
 int val = analogRead(potpin); // valor del potentiometro (entre 0 y 1023)
 int posicionServo**l = map(val, 0, 1023, 0, 179); // escalamos para el servo (entre 0 y 180)
 myservo.write(val); // envía la posición al servo
 delay(15);
}
```

Para usar la librería Servo con bitbloq podemos usar los bloques Servo. Existen 2 tipos de servos: los de rotación continua y los normales.

Veamos este ejemplo:

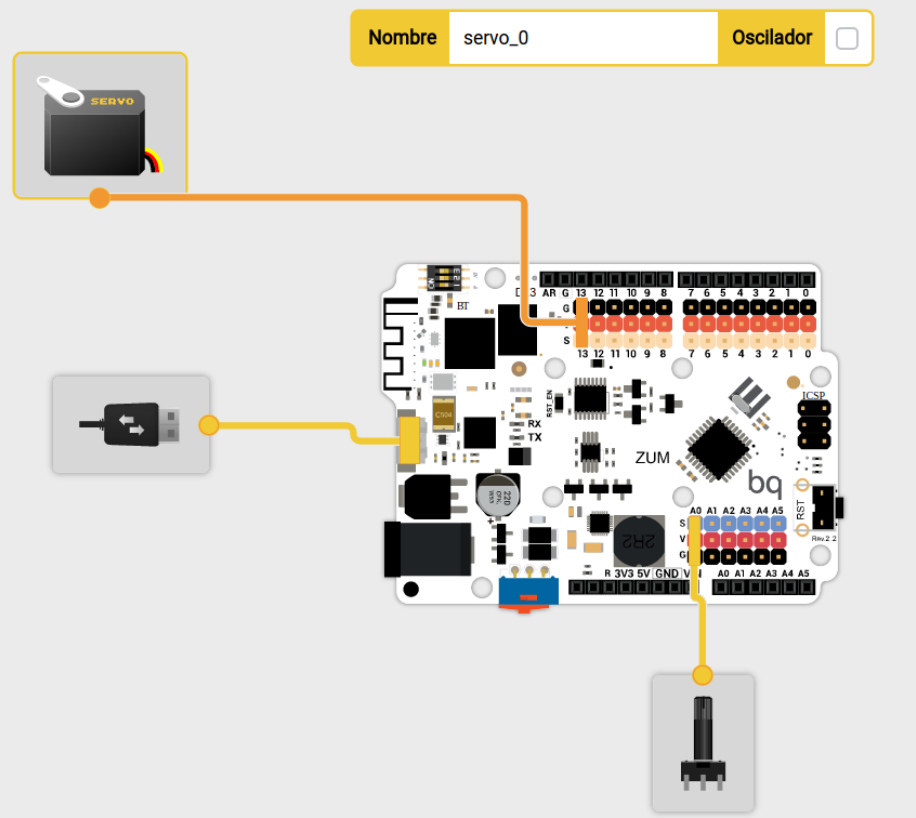


Figura 22: servo

Ejemplo: Librería EEPROM

Guarda datos en la memoria no volátil EEPROM:

- EEPROM.write(address, value) escribe el valor value en address.
- EEPROM.read(address): devuelve el valor de la posición address.

En el tema de almacenamiento veremos ejemplos de acceso a EEPROM