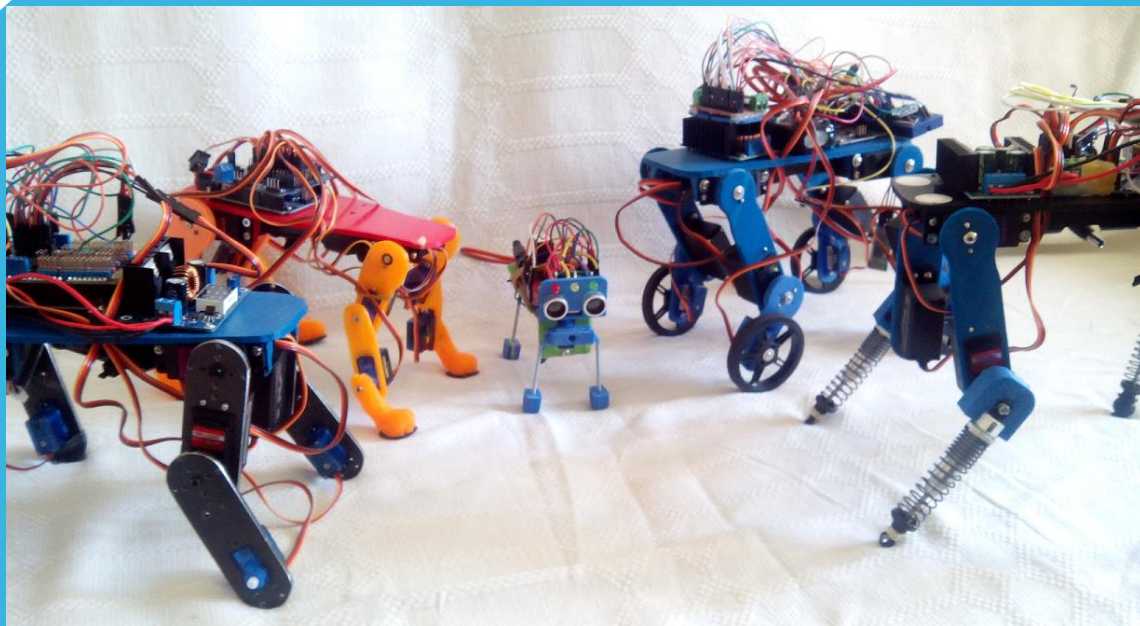


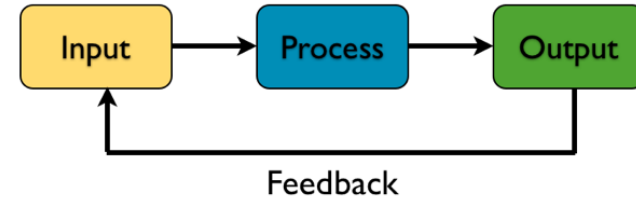
TALLER DE INICIACIÓN A LA ROBÓTICA CON ARDUINO

MARCOS SÁNCHEZ GARCÍA

WWW.3DSOLID.ES



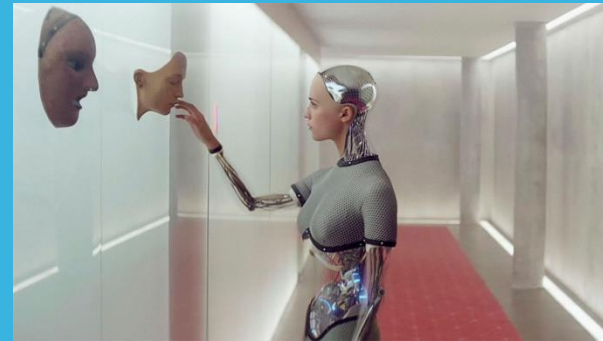
1. ¿Qué entendemos por «robot»?



Tradicionalmente: «Máquina programable capaz de hacer determinadas acciones/trabajos de forma autónoma»

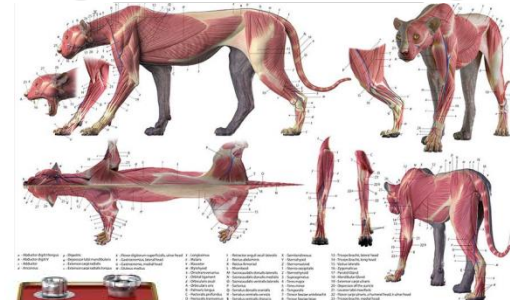
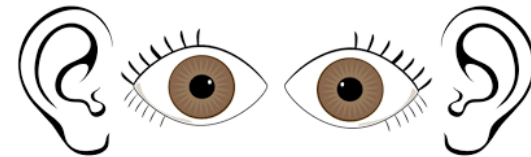
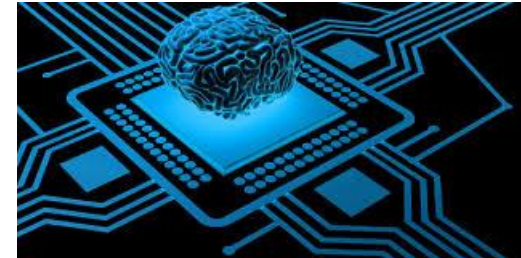
Hoy en día se espera que se aproxime más a un ser vivo:

1. Que sea capaz de **percibir** su entorno (sensores).
2. Que sea capaz de **reaccionar** adecuadamente a las situaciones que sobrevengan. Iniciando o deteniendo un movimiento (motores), dando avisos luminosos (diodos led) o acústicos (zumbadores), o hablando (sintetizadores de voz, altavoces) o realizando cualquier otra acción o «interacción» conveniente. Esto , a su vez ,podría permitirle **comunicarse** con otros robots o con personas.
3. Que sea capaz de **aprender** de situaciones pasadas para perfeccionar su «comportamiento»: inteligencia artificial, **redes neuronales** etc



1.¿Qué elementos necesitamos para hacer un robot?

1. **«Cerebro»:** **placa controladora** que albergue el programa que dirija al robot y que gestione los componentes electrónicos: Arduino, Raspberry etc
2. **Sensores:** que reciban la información del entorno (sensores de luz, temperatura, ultrasónicos etc).
3. **Actuadores:** que permitan dar una respuesta: motores, diodos led, zumbadores, mosfet etc
4. **Fuente de alimentación**, que aporte energía al sistema. En robots pequeños muy simples nos puede bastar una simple pila de 9V alcalina o recargable. En robots más grandes se aconseja usar baterías LiPo y separar la alimentación de la placa controladora (Arduino) de los motores , ya que estos requieren un amperaje mucho más alto que el que puede suministrar la placa controladora. Algunas shields permiten esta doble alimentación (5 V para el Arduino y 6V o más para los motores).
5. Un **esqueleto** o **chasis** (plástico, metal, mixto...)

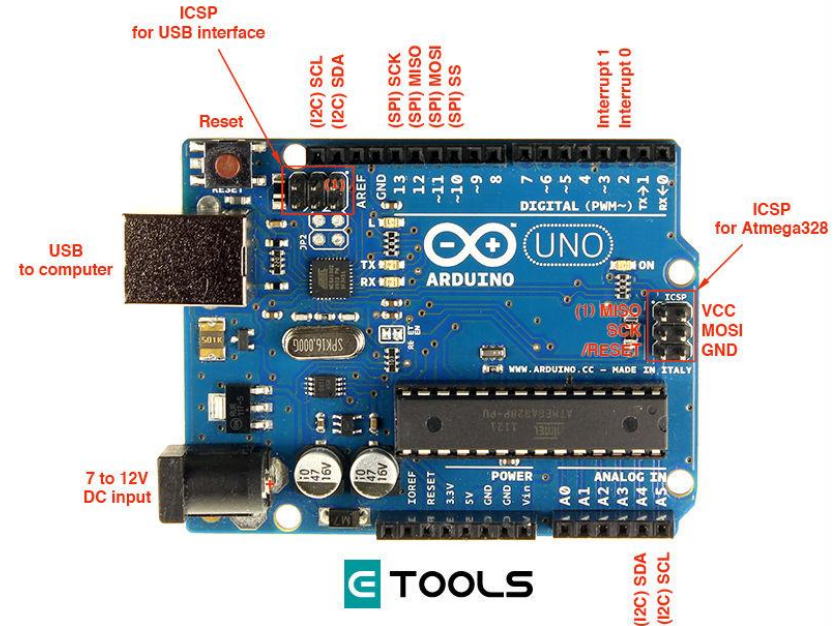


2.1. ¿Qué es Arduino?

Es una tarjeta electrónica programable que permite construir dispositivos interactivos que pueden controlar objetos del mundo real.

1. Es una plataforma de hardware libre, sus especificaciones y esquemas son de acceso público.
2. Proporciona también un software consistente en un entorno de desarrollo (IDE), con el que se puede programar fácilmente (deriva de C++) y que se puede bajar gratuitamente de internet:

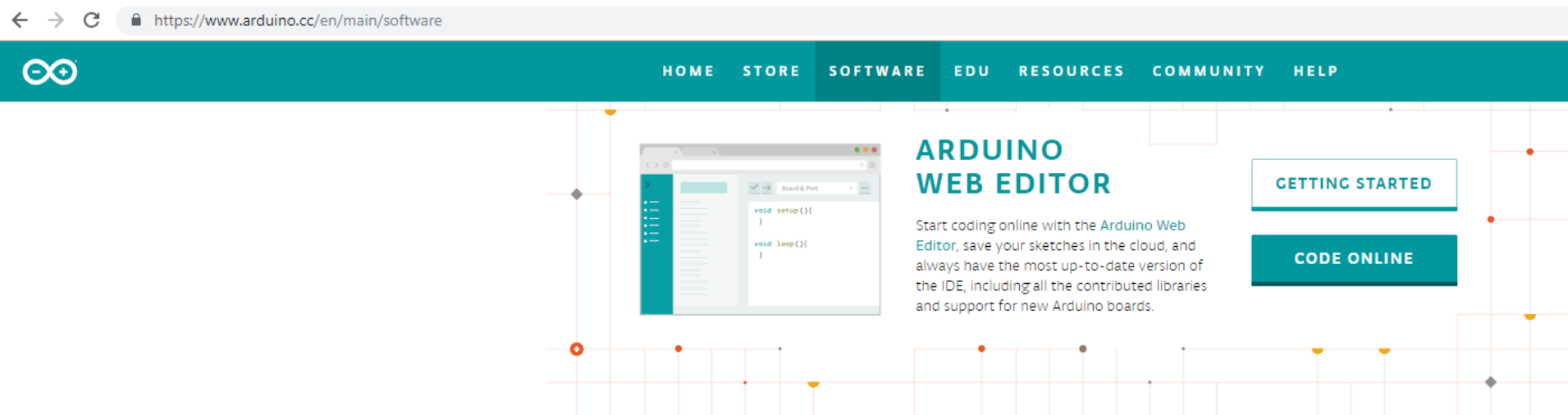
<https://www.arduino.cc/en/main/software>



Resumen de características Técnicas

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Limite)	6 – 20V
Pines para entrada- salida digital.	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica.	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

2.2. Descarga del entorno de programación (IDE):



Download the Arduino IDE

The screenshot shows the 'ARDUINO 1.8.8' download page. On the left is the Arduino logo. The text describes the IDE as open-source software that runs on Windows, Mac OS X, and Linux. A red arrow points from the 'ARDUINO 1.8.8' section to the 'Windows Installer, for Windows XP and up' option. Other options include 'Windows ZIP file for non admin install', 'Windows app' (requiring Win 8.1 or 10), 'Mac OS X 10.8 Mountain Lion or newer', 'Linux 32 bits', 'Linux 64 bits', and 'Linux ARM'.

ARDUINO 1.8.8

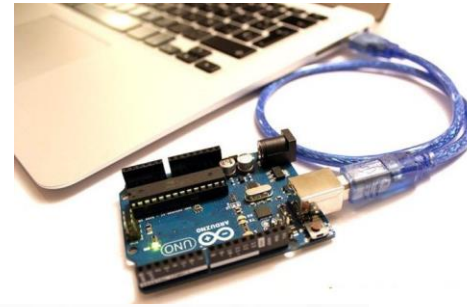
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install
Windows app Requires Win 8.1 or 10
Mac OS X 10.8 Mountain Lion or newer
Linux 32 bits
Linux 64 bits
Linux ARM

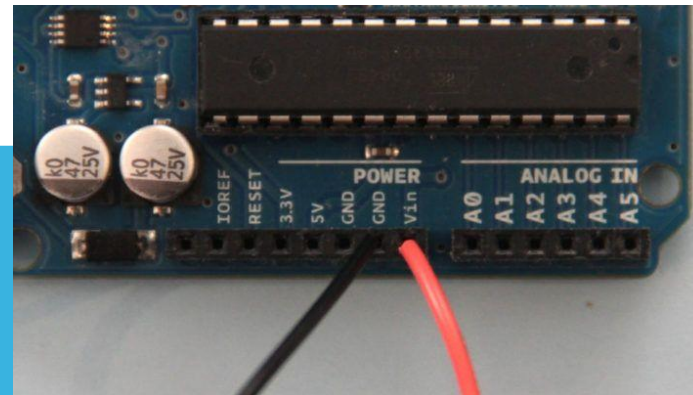
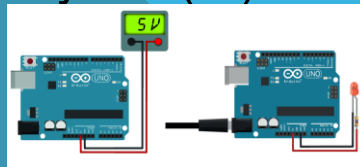
2.3. ¿Cómo se alimenta un Arduino?



1. Mediante el conector USB (5V).
2. Mediante el jack de alimentación (entre 7 y 12 V). La ideal sería con salida (output) de 7 V, 1 A y DC (corriente directa, no AC)
3. Entrada Vin y GND (entre 6 y 12 V). ¡No conectar a la vez que el jack!



4. Entrada 5V y GND (5V).



2.4. Esquema de conexiones de Arduino:

Las **entradas** (**input**) permiten recibir información del entorno y; las **salidas** (**output**) sirven para dar órdenes adecuadas a esa información recibida.

1. Entradas **analógicas**.

Son los pines (clavijas, conectores) que van desde A0 hasta A5.

El rango de entrada va desde el valor 0 (0v) hasta 1024 (5V) , detecta variaciones de 4,8 mV.

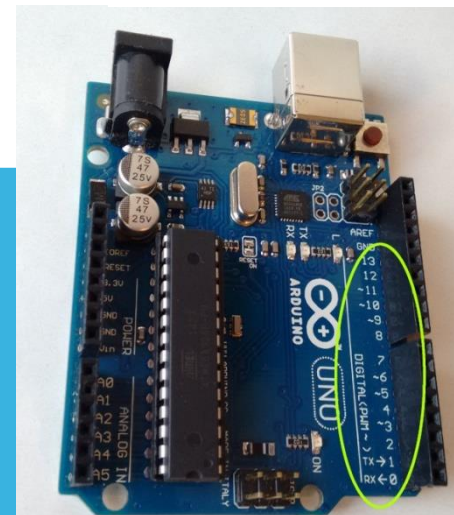
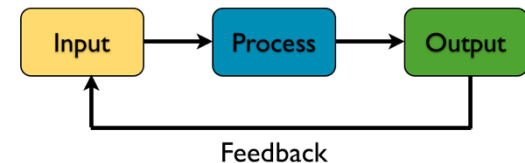
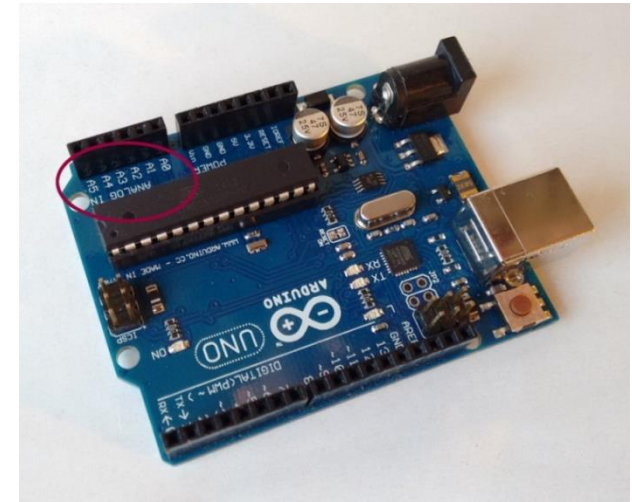
2. Entradas y salidas **digitales**.

Van desde D0 a D13.

Si los queremos usar como salidas hay que **declararlos** como tal , al inicio de nuestro programa de Arduino (en el **setup**).

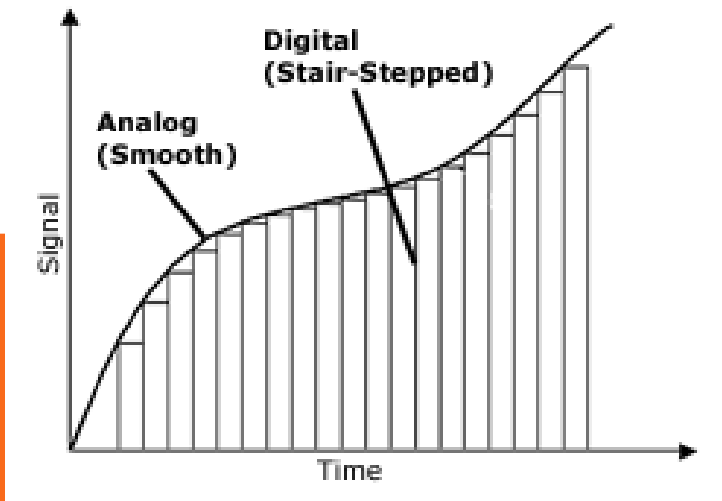
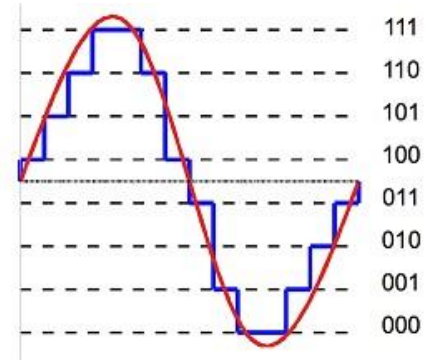
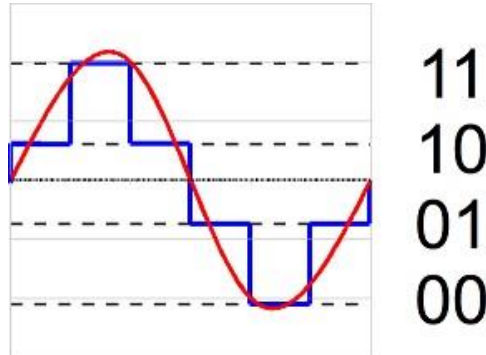
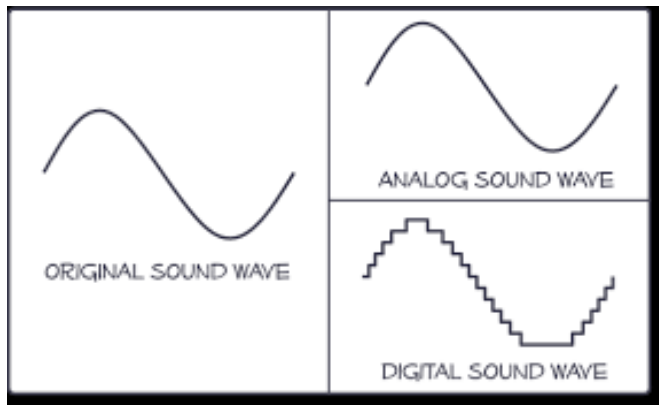
*Podemos utilizar los pines digitales 3,5,6,9,10,11 como salidas analógicas (PWM: ~, con rango desde 0 a 255),

+ info: <http://arduino.synchroproiekt.com>






















2.4. Esquema de conexiones de Arduino:

Diferencia «analógico» frente a «digital»



2.4. Opciones/tipos de Arduino para robótica:

Arduino 	Processor 	Flash KiB 	EEPROM KiB 	SRAM KiB 	Digital I/O pins 	...with PWM 	Analog input pins 	USB Interface type 	Dimensions inches 	Dimensions mm 
Diecimila 	ATmega168	16	0.5	1	14	6	6	FTDI	2.7" x 2.1"	68.6mm x 53.3mm
Duemilanove 	ATmega168/328P	16/32	0.5/1	1/2	14	6	6	FTDI	2.7" x 2.1"	68.6mm x 53.3mm
Uno 	ATmega328P	32	1	2	14	6	6	ATmega8U2	2.7" x 2.1"	68.6mm x 53.3mm
LilyPad 	ATmega168V or ATmega328V	16	0.5	1	14	6	6	None	2" ø	50mm ø
Fio 	ATmega328P	32	1	2	14	6	8	None	1.6" x 1.1"	40.6mm x 27.9mm
Nano 	ATmega168 or ATmega328	16/32	0.5/1	1/2	14	6	8	FTDI	1.70" x 0.73"	43mm x 18mm
Mega 	ATmega1280	128	4	8	54	14	16	FTDI	4" x 2.1"	101.6mm x 53.3mm
Mega2560 	ATmega2560	256	4	8	54	14	16	ATmega8U2	4" x 2.1"	101.6mm x 53.3mm

LilyPad



Nano



Uno



Mega 2560



The diagram shows the top of an Arduino Uno R3 PCB. The pins are arranged in two rows. The top row contains pins 0-13, A0-A5, and various power pins. The bottom row contains pins 16-25, A6-A9, and various power pins. A large black box with the text "Shield 'Guts' Go Here" is placed in the center of the board.

Pin Locations:

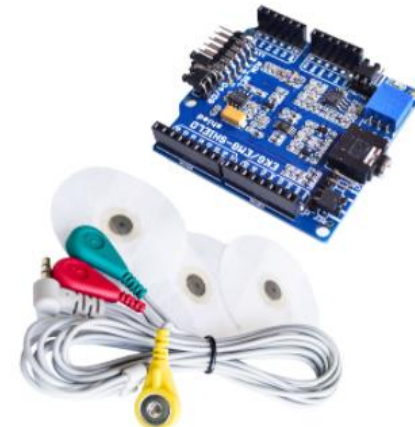
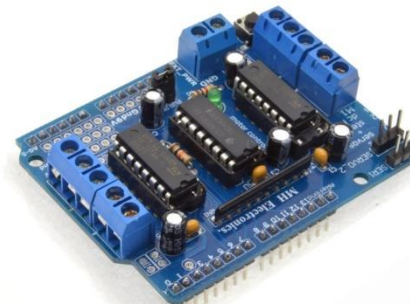
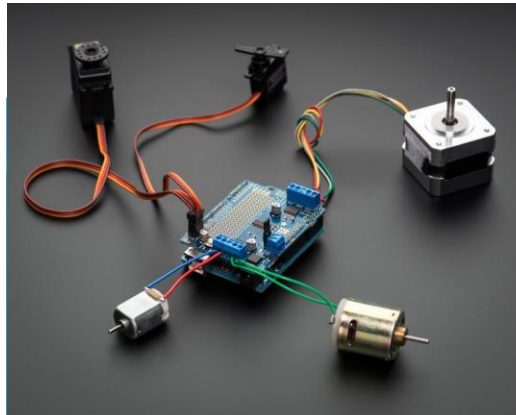
- Top Row (Left to Right):**
 - 0 (blue), 1 (blue), 2 (blue), 3 (blue), 4 (blue), 5 (blue)
 - 6 (blue), 7 (blue), 8 (blue), 9 (blue), 10 (blue), 11 (blue)
 - 12 (blue), 13 (blue), A0 (green), A1 (green), A2 (green), A3 (green)
 - A4 (green), A5 (green), GND (blue), 5V (red), 3.3V (red), IOREF (yellow)
- Bottom Row (Left to Right):**
 - 16 (blue), 17 (blue), 18 (blue), 19 (blue), 20 (blue), 21 (blue)
 - 22 (blue), 23 (blue), 24 (blue), 25 (blue), A6 (green), A7 (green)
 - A8 (green), A9 (green), GND (blue), 5V (red), 3.3V (red), IOREF (yellow)

Legend:

- GND:** Blue circle
- Reset:** Yellow circle
- MOSI:** Red circle
- SCK:** Green circle
- VCC:** Red circle
- MISO:** Blue circle

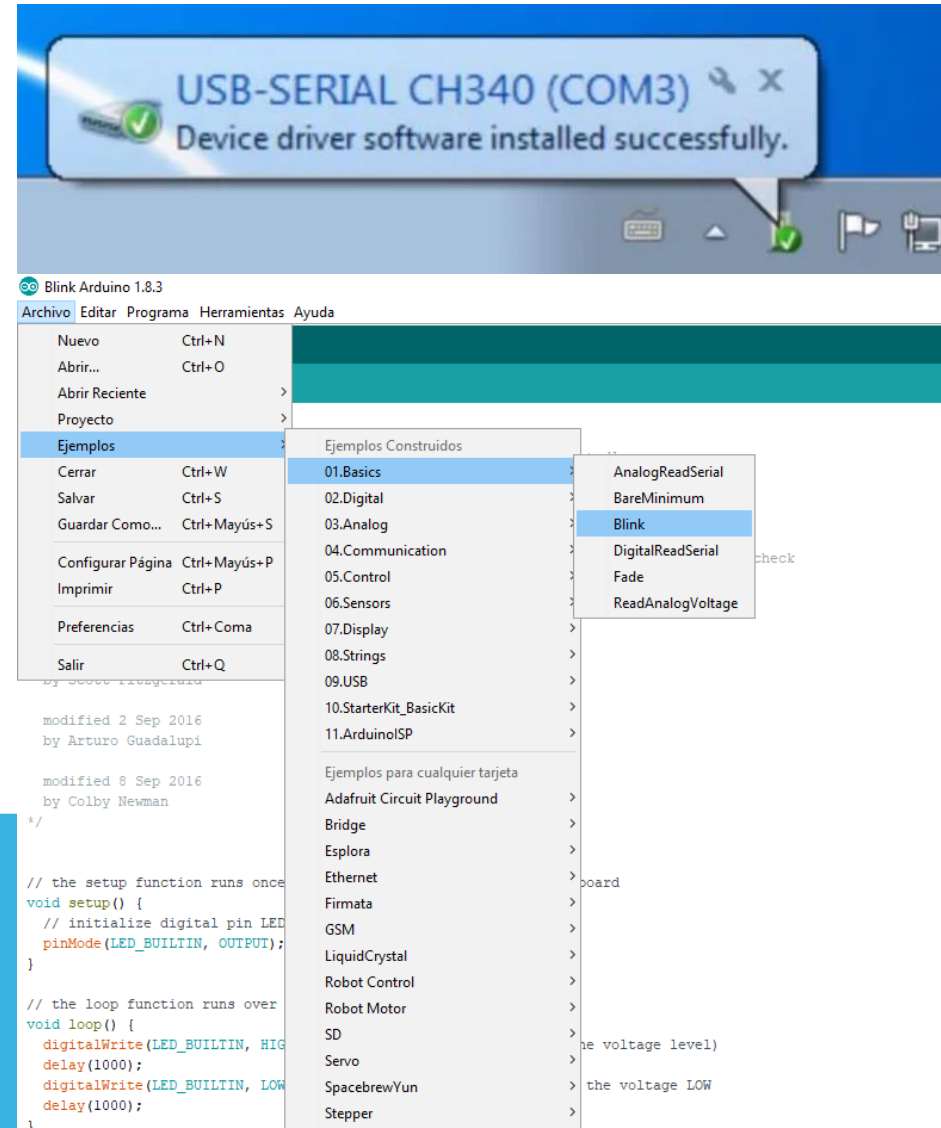
-
- A blue breadboard populated with various electronic components. It includes several resistors of different values (e.g., 10k, 1k, 100k), two electrolytic capacitors (one 100µF, one 10µF), and several integrated circuits (ICs) such as a 741 op-amp, a 555 timer, and a 7401 NAND gate. There are also two push buttons, a 9V battery connector, and a 5-pin D-sub connector. The breadboard is densely packed with components, demonstrating a complex circuit setup.

-
- A red PCB-based module, likely a sensor or interface board. It features several pin headers with blue and black pins. A black connector is visible on the right side. The board includes various electronic components such as resistors, a capacitor, and a small integrated circuit.



3.1. Nuestra primera prueba de Arduino: Blink

1. Conectamos la placa de Arduino al puerto USB de nuestro Arduino UNO (cable de conexión A-B).
2. Buscamos e instalamos los driver del arduino chino (chip CH340G).
3. Pulsamos en el icono del programa para abrir el IDE (entorno de programación): «Archivo «-> «Ejemplos «-> «Basics «-> «Blink»
4. Seleccionamos placa (Arduino UNO) y puerto COM3 o superior en: «Herramientas» ->»Placa» y «Herramientas» ->»Puerto»
5. Subimos el sketch (programa) Blink , comprobamos su funcionamiento.



3.1. Nuestra primera prueba de Arduino: Blink

🔗 Blink Arduino 1.8.3

Archivo Editar Programa Herramientas Ayuda

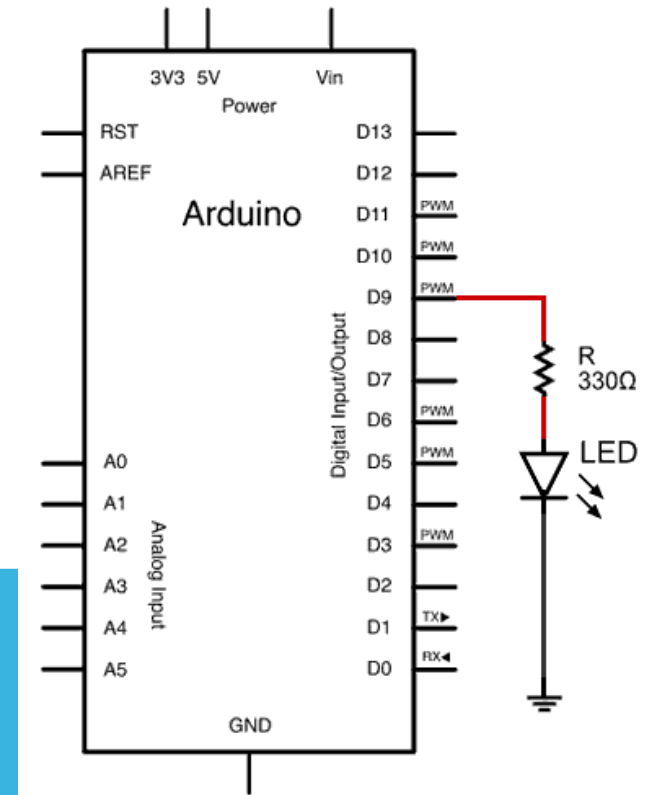
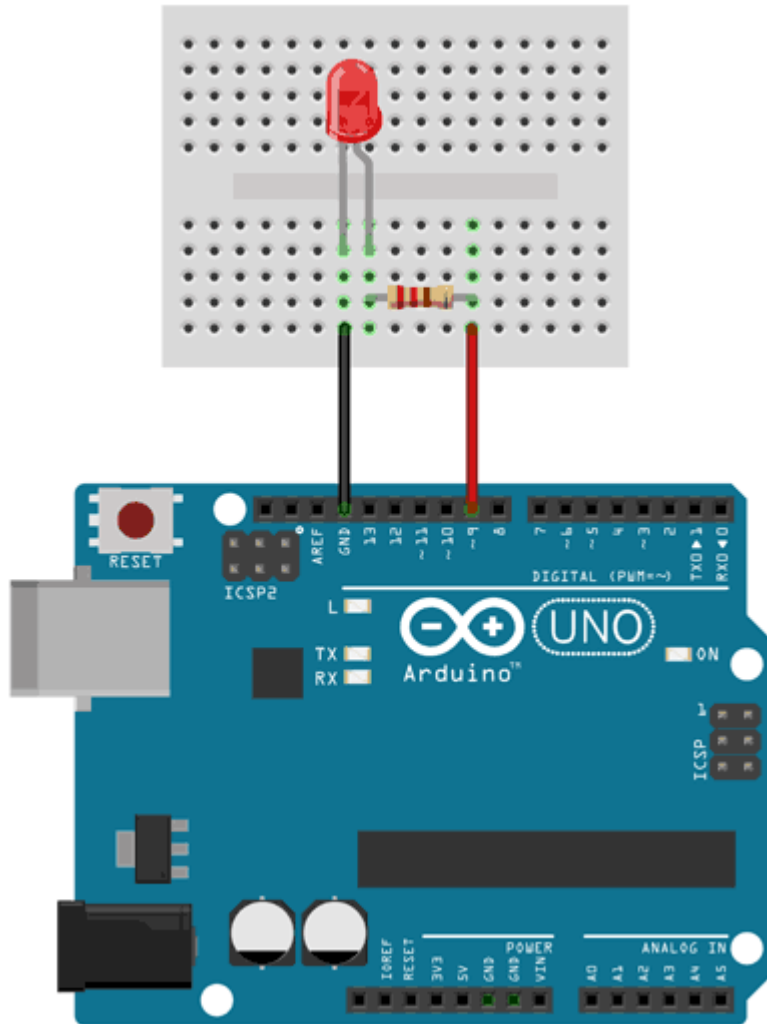


```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO  
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to  
  the correct LED pin independent of which board is used.  
  If you want to know what pin the on-board LED is connected to on your Arduino model, check  
  the Technical Specs of your board at https://www.arduino.cc/en/Main/Products  
  
  This example code is in the public domain.  
  
  modified 8 May 2014  
  by Scott Fitzgerald  
  
  modified 2 Sep 2016  
  by Arturo Guadalupi  
  
  modified 8 Sep 2016  
  by Colby Newman  
*/
```

```
// the setup function runs once when you press reset or power the board  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
  delay(1000); // wait for a second  
}
```

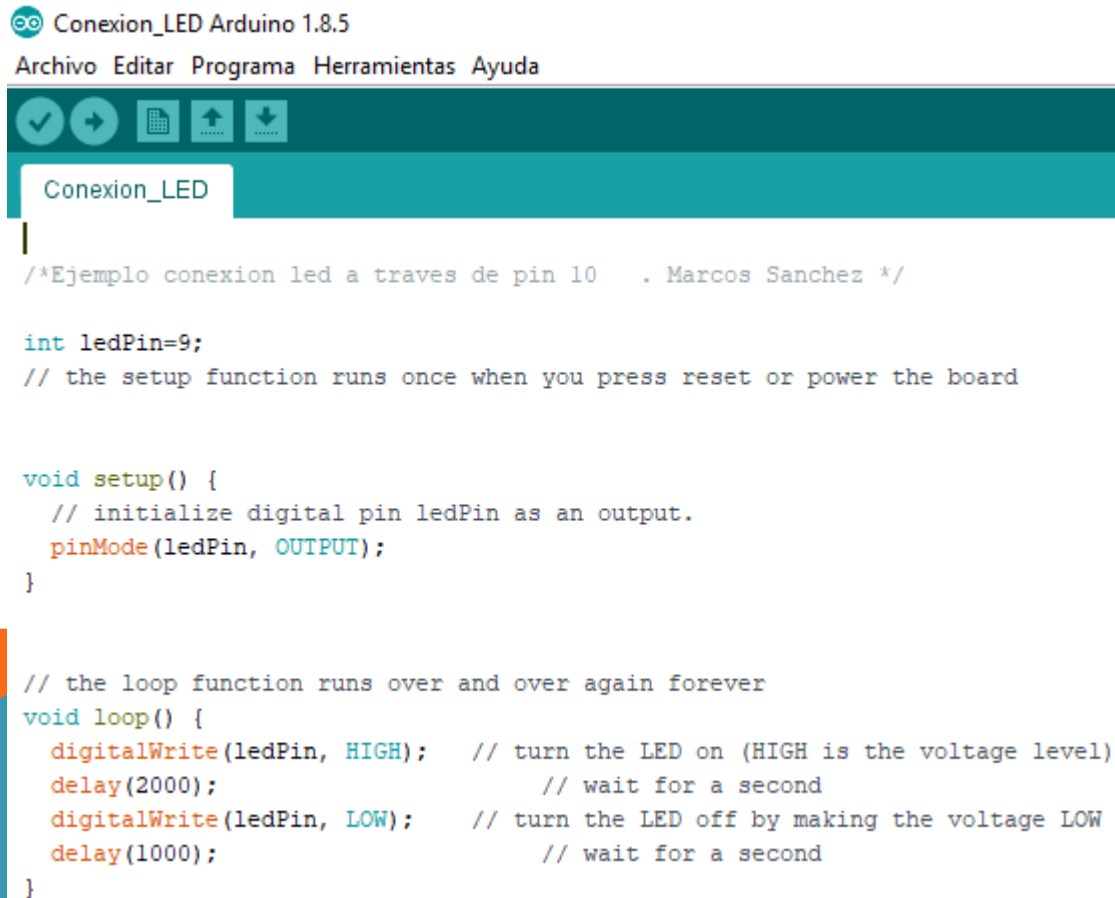

3.2. Conexión del Arduino a un LED :

1. Conectamos la placa de Arduino al puerto USB de nuestro Arduino UNO (cable de conexión A-B). Usaremos una resistencia de entre 700 Oh (Ω) y 1KOh (Ω).



3.2. Conexión de un LED al Arduino:

1. Conectamos la placa de Arduino al puerto USB de nuestro Arduino UNO (cable de conexión A-B). Usaremos una resistencia de entre 700 Oh (Ω) y 1KOh (Ω).



The screenshot shows the Arduino IDE interface with the 'Conexion_LED' sketch loaded. The menu bar includes 'Archivo', 'Editar', 'Programa', 'Herramientas', and 'Ayuda'. The toolbar contains icons for checking, running, saving, and uploading. The sketch name 'Conexion_LED' is displayed in the top bar. The code is as follows:

```
Conexion_LED Arduino 1.8.5
Archivo Editar Programa Herramientas Ayuda

/*Ejemplo conexion led a traves de pin 10 . Marcos Sanchez */

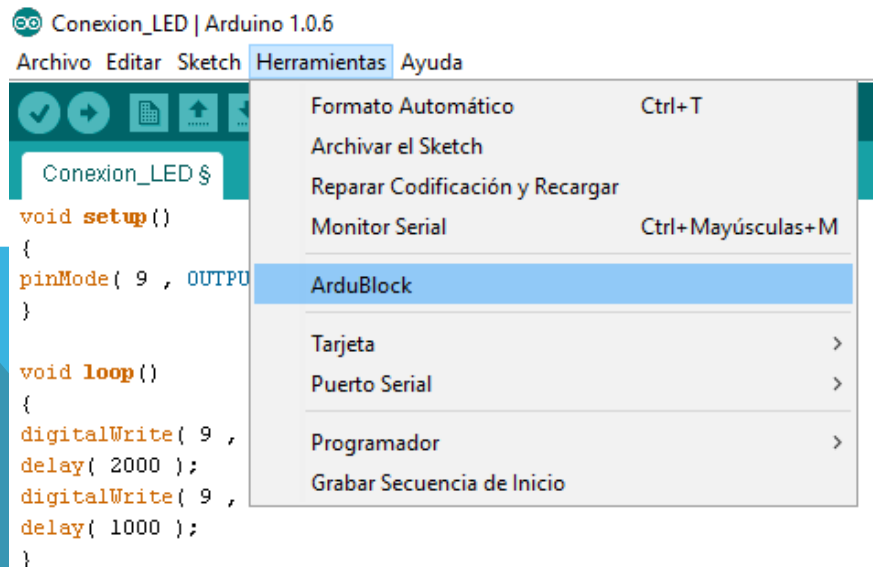
int ledPin=9;
// the setup function runs once when you press reset or power the board

void setup() {
  // initialize digital pin ledPin as an output.
  pinMode(ledPin, OUTPUT);
}

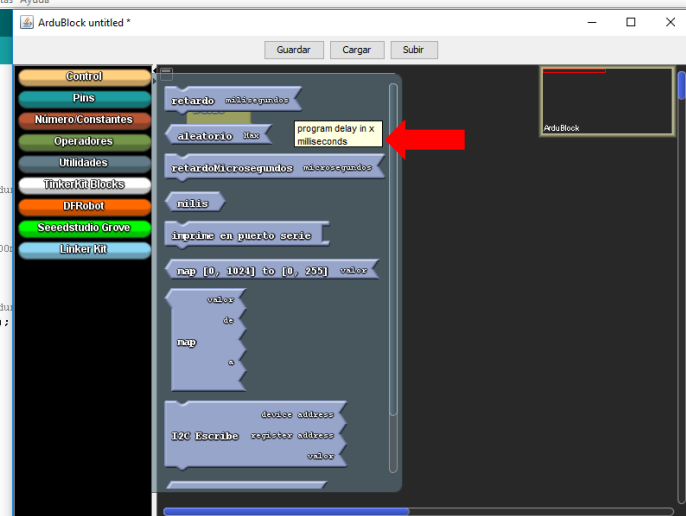
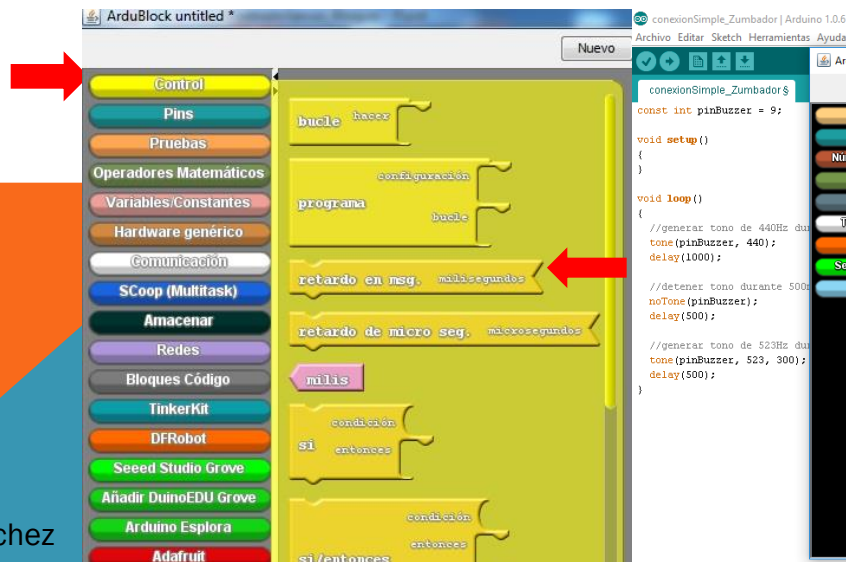
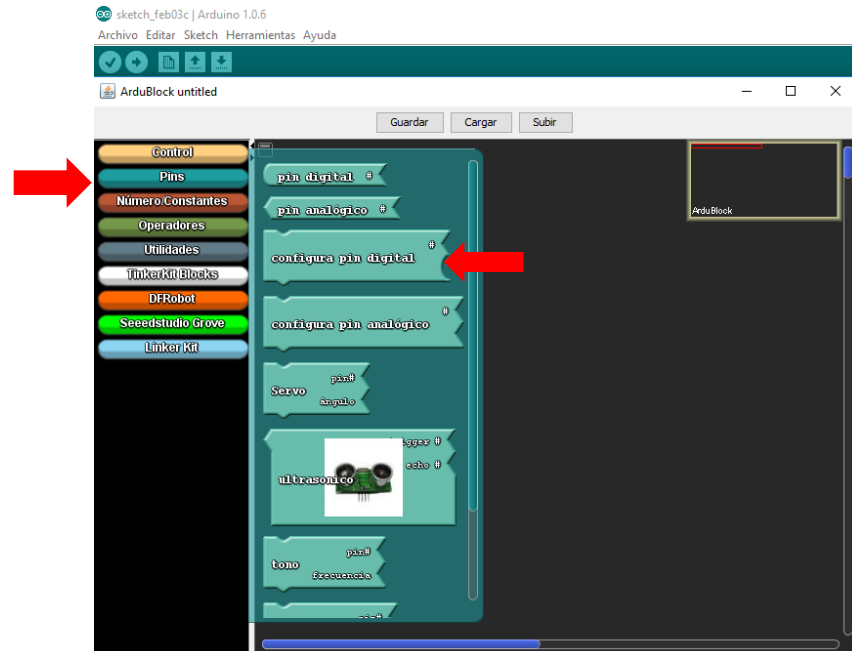
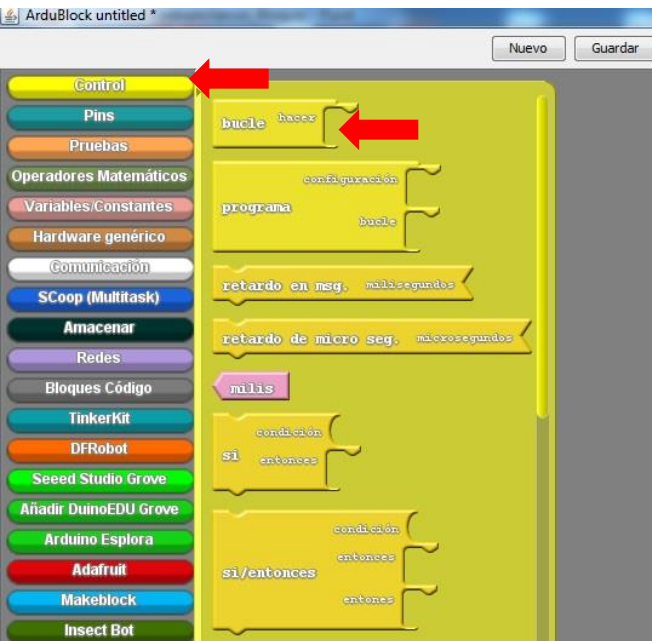
// the loop function runs over and over again forever
void loop() {
  digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(2000);                // wait for a second
  digitalWrite(ledPin, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                // wait for a second
}
```

3.2. Conexión de LED con ArduBlock:

1. Ardublock es un entorno de programación gráfica, que usa bloques funcionales.
2. La versión «oficial» de ArduBlock solo funciona con Arduino 1.0.6, que se puede descargar de: <https://www.arduino.cc/en/Main/OldSoftwareReleases#previous>
3. El archivo Ardublock se descarga de : <https://sourceforge.net/projects/ardublock/>
4. Luego se crean la carpeta «ArduBlockTool» y «tool» en la carpeta de instalación del Arduino de manera que quede la siguiente ruta, donde se copia el archivo descargado Ardublock en la carpeta «tool»: C:\Program Files (x86)\Arduino\tools\ArduBlockTool\tool
5. Después se abre el IDE de Arduino y en el menú “Herramientas “ está el enlace para Ardublock:



3.2. Conexión de LED con ArduBlock:



3.2. Conexión de LED con ArduBlock:

sketch_feb03a | Arduino 1.0.6

Archivo Editar Sketch Herramientas Ayuda



sketch_feb03a \$

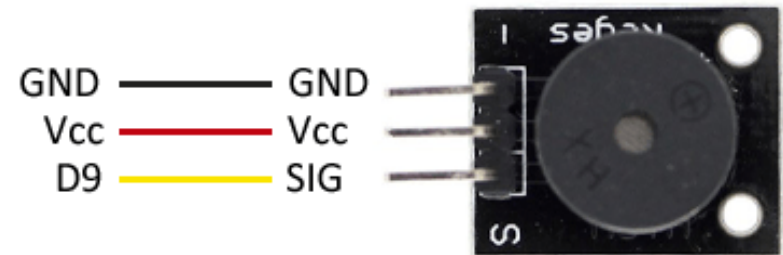
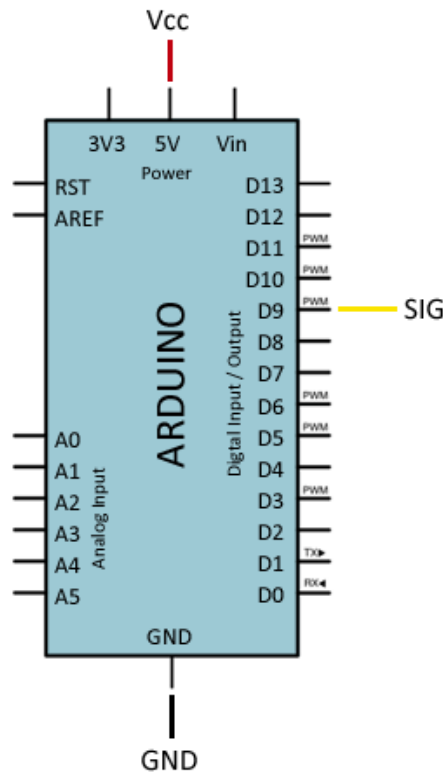
```
void setup()
{
  pinMode( 9 , OUTPUT);
}

void loop()
{
  digitalWrite( 9 , HIGH );
  delay( 2000 );
  digitalWrite( 9 , LOW );
  delay( 1000 );
}
```

The screenshot shows the ArduBlock software interface. On the left is a sidebar with a category menu: Control, Pins, Número/Constantes, Operadores, Utilidades, TinkerKitBlocks, DFRobot, Seeedstudio Grove, and Linker Kit. The main workspace is titled 'ArduBlock untitled *' and contains a 'do' loop block. Inside the loop, there are four blocks in sequence: 'configura pin digital' with pin #9 and mode HIGH, 'retardo' for 2000 milliseconds, 'configura pin digital' with pin #9 and mode BAJO, and 'retardo' for 1000 milliseconds. At the top of the workspace are buttons for 'Guardar', 'Cargar', and 'Subir'. A small preview window on the right shows a dark area with the text 'ArduBlock'.

3.3. Conexión del Arduino a un zumbador pasivo :

1. Los buzzer son transductores piezoeléctricos , es decir, tienen una membrana que vibra al ser atravesada por una corriente eléctrica y esto genera los sonidos. Son muy fáciles de conectar porque solo llevan 3 pines: GND (se conecta al GND del Arduino), VCC (se conecta al positivo de Arduino: 5V o 3,3V), y «S» (signal) que transmite la señal y se conecta habitualmente a un pin digital.



3.3. Conexión del Arduino a un zumbador pasivo :

1. Mientras esté funcionando no podemos usar las salidas PWM en los pines 3 y 11 en Arduino Nano y Uno (9 y 10 en caso del Arduino mega). Tampoco podemos usar dos tone a la vez, deberemos apagar uno con la función noTone () antes de usar el otro.
2. Los rangos de la función tone son de 61 a 65535 Hz.



```
conexionSimple_Zumbador | Arduino 1.0.6
Archivo Editar Sketch Herramientas Ayuda

void setup()
{
}

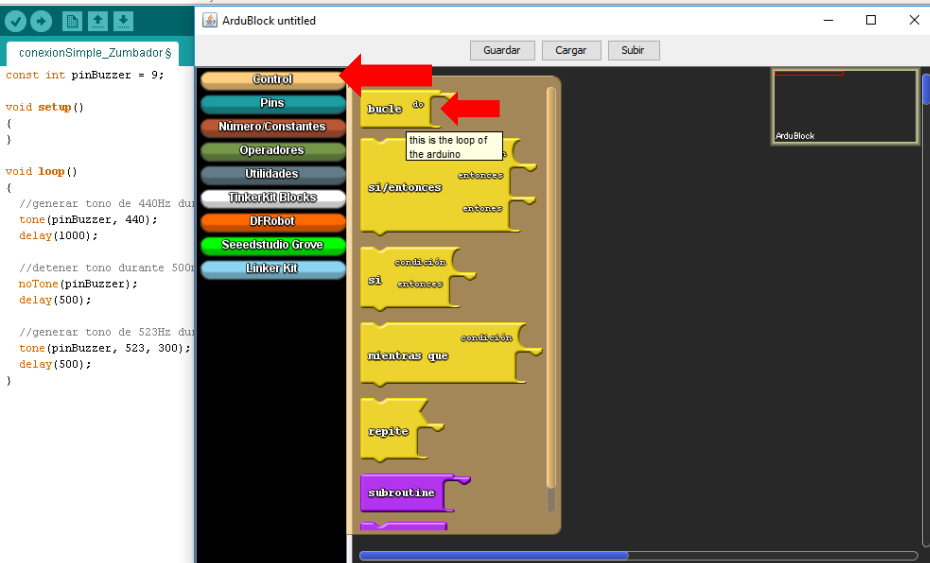
void loop()
{
    //generar tono de 440Hz durante 1000 ms
    tone(pinBuzzer, 440);
    delay(1000);

    //detener tono durante 500ms
    noTone(pinBuzzer);
    delay(500);

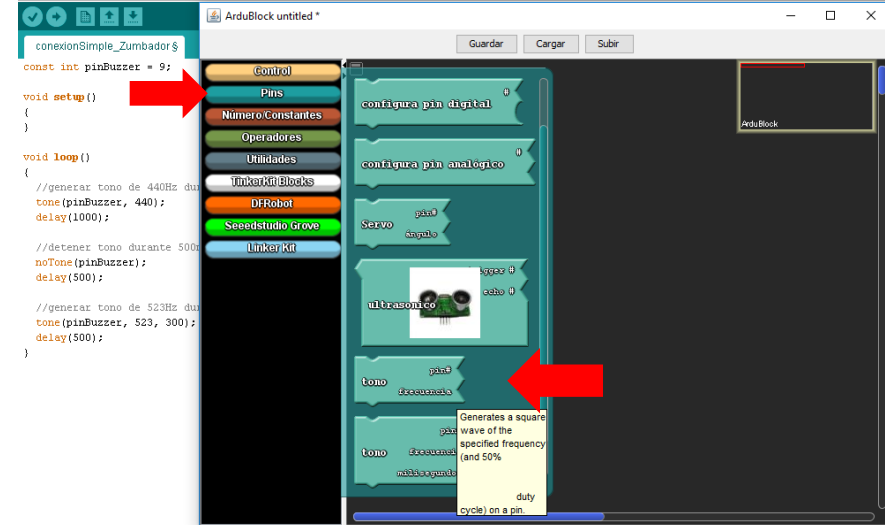
    //generar tono de 523Hz durante 500ms, y detenerlo durante 300ms.
    tone(pinBuzzer, 523, 300);
    delay(300);
}
```

3.3.Conexión de zumbador pasivo con ArduBlock :

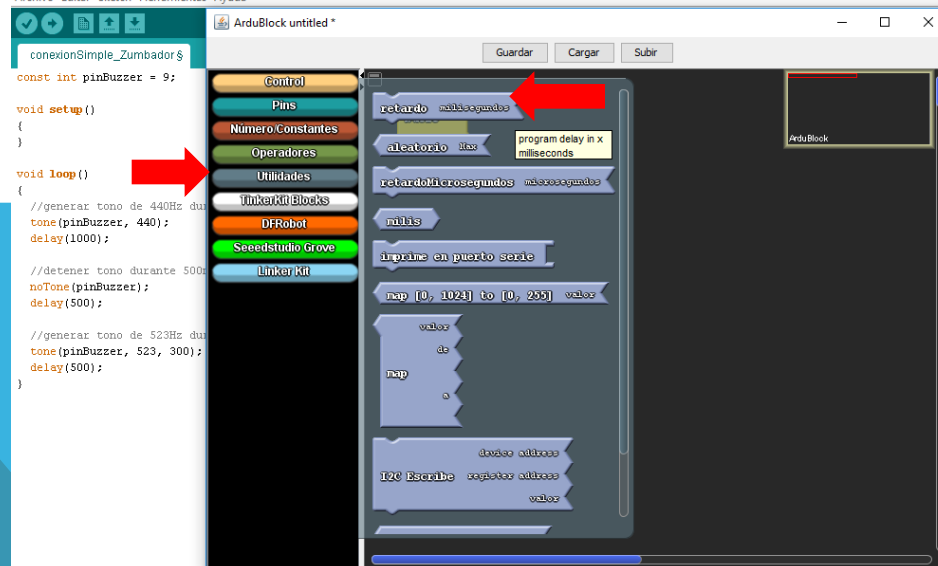
conexionSimple_Zumbador | Arduino 1.0.6
Archivo Editar Sketch Herramientas Ayuda



conexionSimple_Zumbador | Arduino 1.0.6
Archivo Editar Sketch Herramientas Ayuda



conexionSimple_Zumbador | Arduino 1.0.6
Archivo Editar Sketch Herramientas Ayuda



3.3. Conexión de zumbador pasivo con ArduBlock :

conexionSimple_Zumbador | Arduino 1.0.6
Archivo Editar Sketch Herramientas Ayuda

ArduBlock conex2_Zumbador_Ardublock.abp *

Guardar Cargar Subir

```
conexionSimple_Zumbador $  
  
void setup()  
{  
  
}  
  
void loop()  
{  
  tone(9, 440);  
  delay( 1000 );  
  noTone(9);  
  delay( 500 );  
  tone(9, 523, 300);  
  delay( 500 );  
}
```

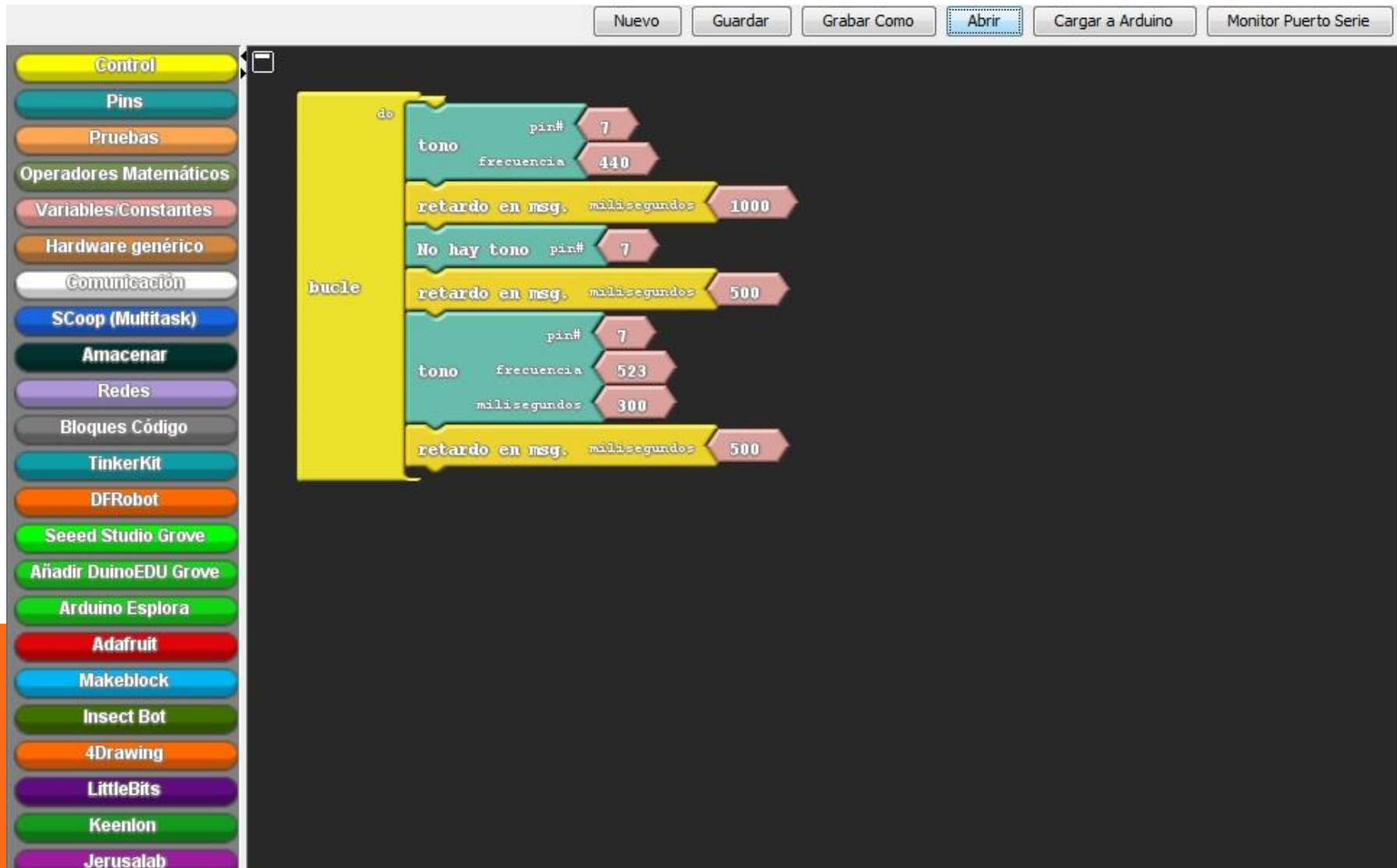
Control
Pins
Número/Constantes
Operadores
Utilidades
TinkerKit Blocks
DFRobot
Seedstudio Grove
Linker Kit

retardo milisegundos 1000
pin# 9
tono frecuencia 440
retardo milisegundos 1000
sin tono pin# 9
retardo milisegundos 500
pin# 9
tono frecuencia 523
milisegundos 300
retardo milisegundos 500

bucle

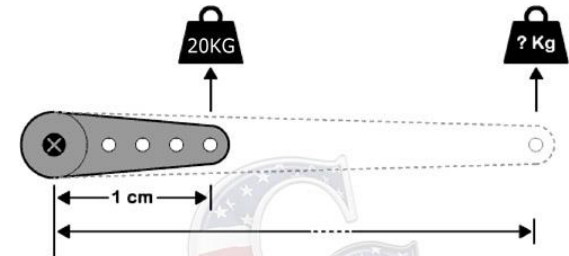
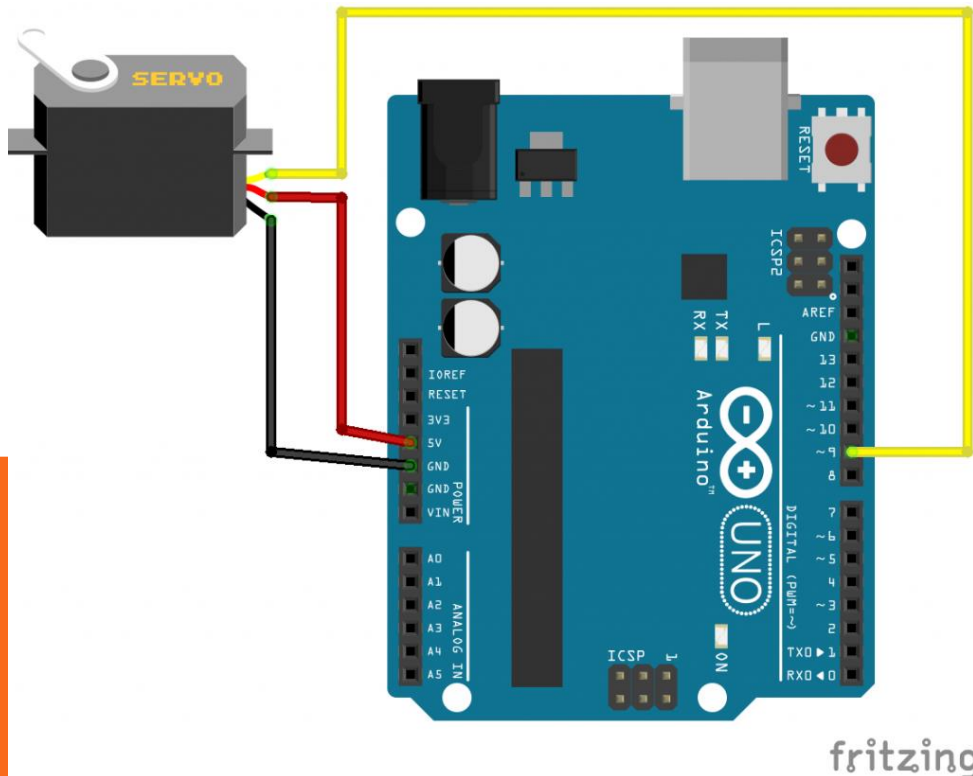
Ardu Block

3.3. Conexión de zumbador pasivo con ArduBlock :



3.4. Conexión del Arduino a un motor servo:

Fabricante	Duración del pulso [ms]			Frec. [Hz]	Color de los cables		
	Mínima (0°)	Neutral (90°)	Máxima (180°)		Positivo	Negativo	Control
Futaba	0.9	1.5	2.1	50	Rojo	Negro	Blanco
Hitech	0.9	1.5	2.1	50	Rojo	Negro	Amarillo
Graupner/Jr	0.8	1.5	2.2	50	Rojo	Marrón	Naranja
Multiplex	1.05	1.6	2.15	40	Rojo	Negro	Amarillo
Robbe	0.65	1.3	1.95	50	Rojo	Negro	Blanco
Simprop	1.2	1.7	2.2	50	Rojo	Azul	Negro



This servo motor's torque is 20kg with 6.7 Volt input.

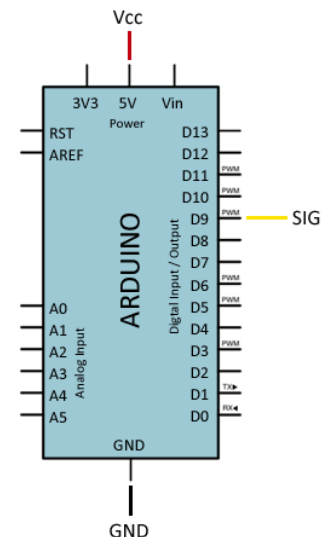
$$1\text{cm} = 20\text{kg} / 1\text{cm} = 20 \text{ kg}$$

$$2\text{cm} = 20\text{kg} / 2\text{cm} = 10\text{kg}$$

$$3\text{cm} = 20\text{kg} / 3\text{cm} = 6.66\text{kg}$$

and so on.

The torque number (force times distance) is a constant.



3.4. Conexión de servos: tipos y características.

1. **Micro servo SG-90**. Engranajes: nylon. Peso : 9 gramos.

Voltaje: 4,8 - 6 V. Torque: 1kg/cm. Tamaño: 23 x 12 x 29 mm

Velocidad :0,12s/ 60 grados (4,8V) sin carga.



2. **Microservo MG90S**: Engranaje: metal. Peso: 12 gramos.

Voltaje : 4,8 – 6 V. Torque: 1,6 (4,8V), 1,8 kg/cm (6V).

Tamaño: 22,6 x 12 x 22,5 mm.

Velocidad :0,1 s/ 60 grados (4,8V) ,0,09s/60 grados (a 6 V)



3. **Servo MG996R**: Engranaje: metal. Peso: 60 gramos aprox.

Voltaje : 4,8 – 7 V. Torque: 13 kg/cm(4,8V), 15 kg/cm (6V).

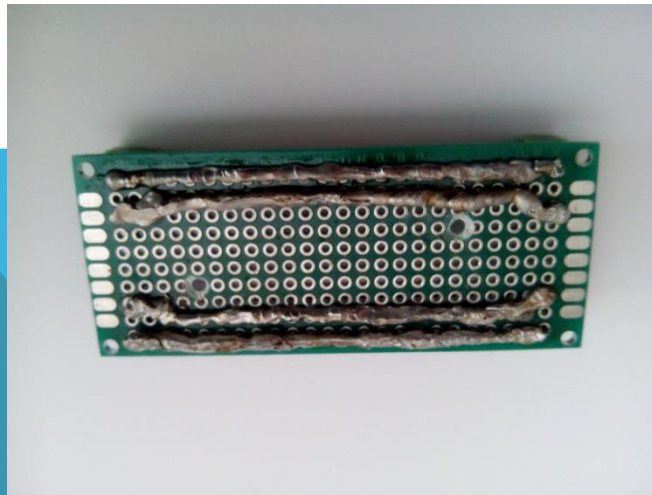
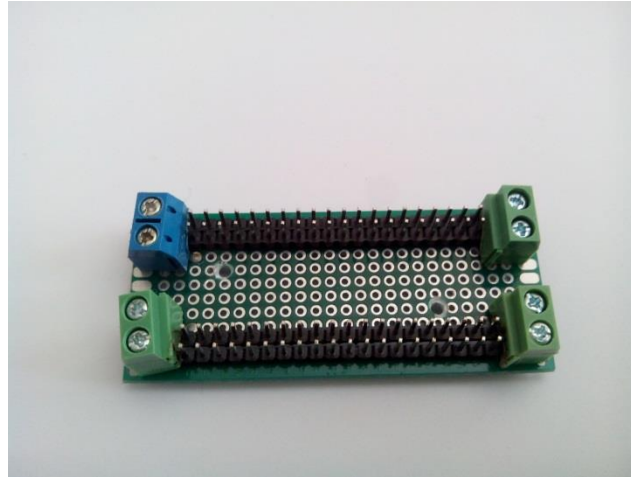
Tamaño: 40 x 19 x 43 mm aprox.

Velocidad :0,17 s/ 60 grados (4,8V) ;0,14s/60 grados (a 6 V)



3.4. Conexión de servos: tipos y características.

PCBs «caseras» para alimentación de doble voltaje (se pueden usar diodos rectificadores como unión de las dos líneas de pines, cada uno baja el voltaje unos 0,35 V):



3.4. Conexión del Arduino a un motor servo:

testServo_simple Arduino 1.8.5

Archivo Editar Programa Herramientas Ayuda



```
#include <Servo.h>    //se incluye la libreria servo que facilita mucho el control de los motores servo

Servo myservo;    // create servo object to control a servo // twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

int ajusteFino = 5;    //grados de ajuste fino para que el engranaje del servo quede a 90 perfectos

void setup() {
  myservo.attach(2);    // attaches the servo on pin 2 to the servo object
}

void loop() {

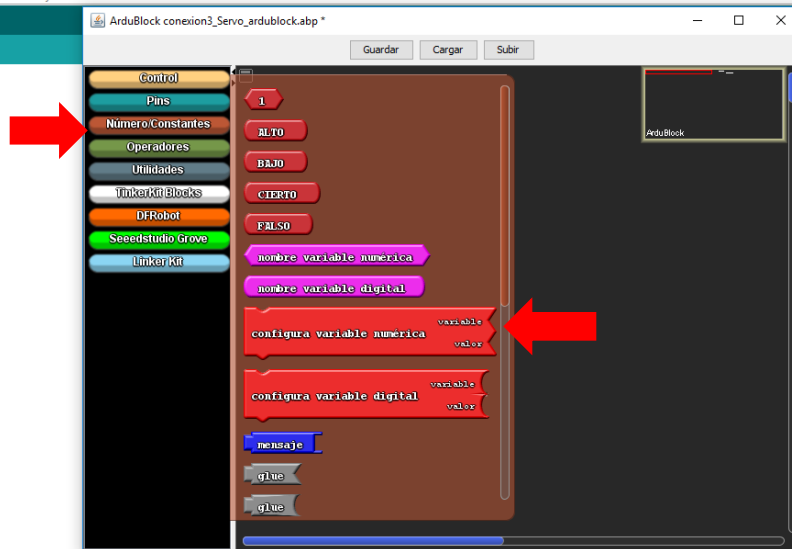
  myservo.write(60 + ajusteFino);    // tell servo to go to position in variable 'pos'
  delay(500);    // waits 15ms for the servo to reach the position

  myservo.write(120 + ajusteFino);    // tell servo to go to position in variable 'pos'
  delay(500);    // waits 15ms for the servo to reach the position

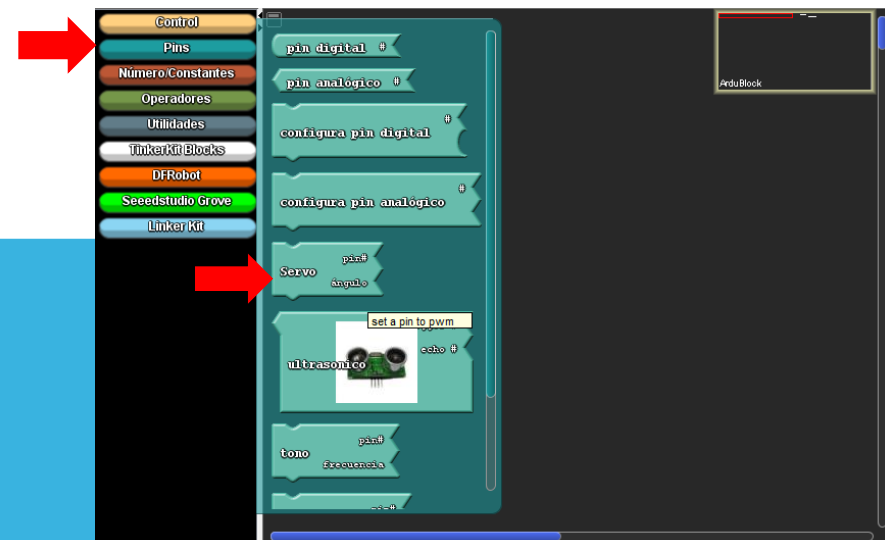
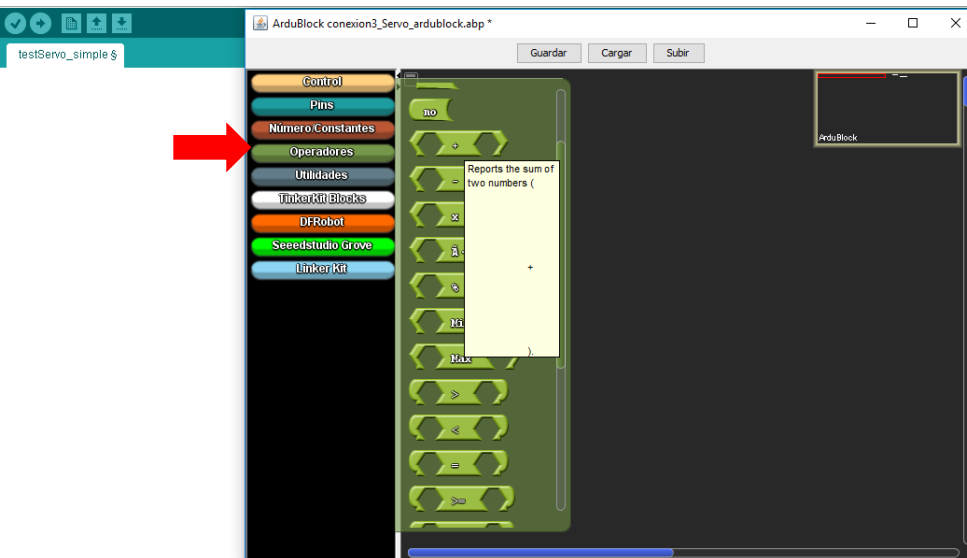
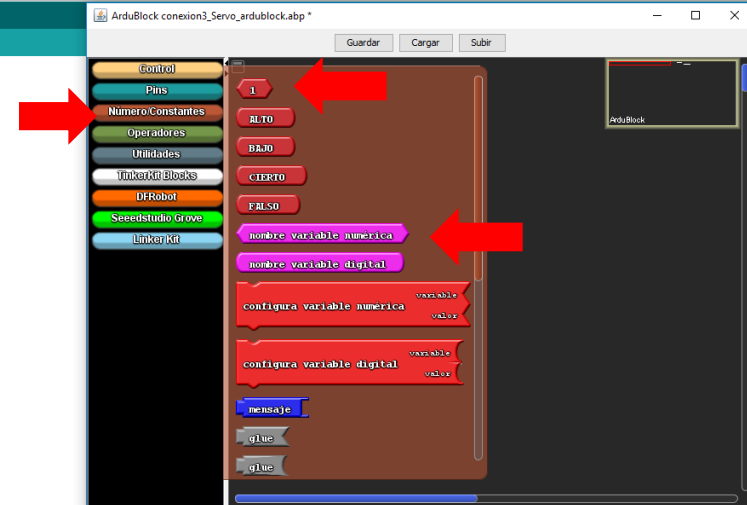
  myservo.write(90 + ajusteFino);    // tell servo to go to position in variable 'pos'
  delay(3000);
}
```

3.4.Conexión a un motor servo usando Ardublock:

testServo_simple | Arduino 1.0.6
Archivo Editar Sketch Herramientas Ayuda



testServo_simple | Arduino 1.0.6
Archivo Editar Sketch Herramientas Ayuda



3.4. Conexión a un motor servo usando Ardublock:

1. Creamos la variable numérica «ajusteFino» para que el motor servo quede perfectamente centrado a 90 grados. En este ejemplo la corrección necesaria para centrar era de 5°.

ArduBlock CentradoServosActual.abp

Nuevo Guardar Grabar Como Abrir Cargar a Arduino Monitor Puerto Serie

Control Pins Pruebas Operadores Matemáticos Variables/Constantes Hardware genérico Comunicación SCoop (Multitask) Amacemar Redes Bloques Código TinkerKit DFRobot Seeed Studio Grove Añadir DuinoEDU Grove Arduino Esplora Adafruit Makeblock Insect Bot

hacer

configura variable integer variable ajusteFinoFront valor 5

configura variable integer variable ajusteFinoBack valor 5

bucle

servo pin# 5 ángulo 90 + ajusteFinoFront

retardo en msg. milisegundos 100

servo pin# 4 ángulo 90 + ajusteFinoBack

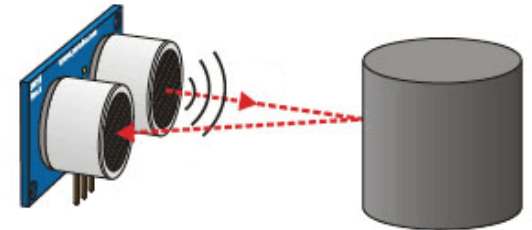
retardo en msg. milisegundos 100

3.4. Conexión a un motor servo usando Ardublock:

1. Ejemplo de marcha hacia delante: combinando la inclinación lateral de las patas delanteras (cambio de peso de un lado a otro) con el empuje del servo trasero.

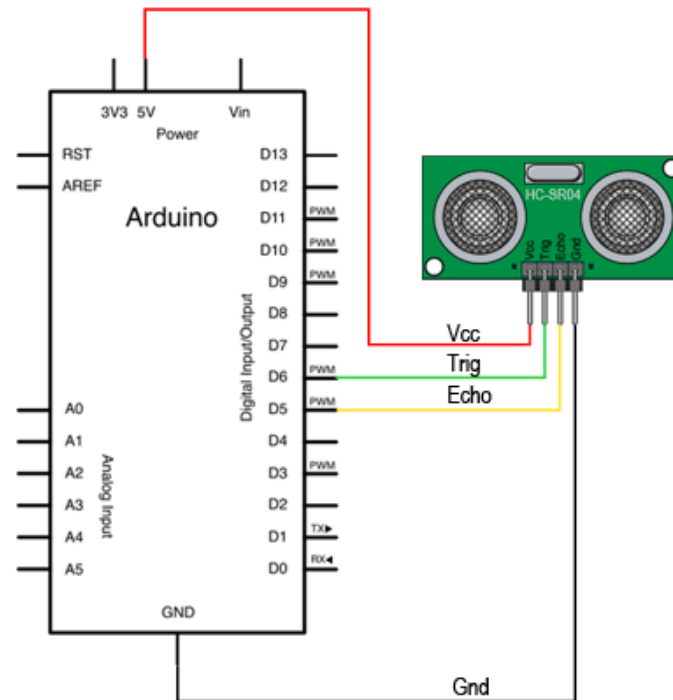
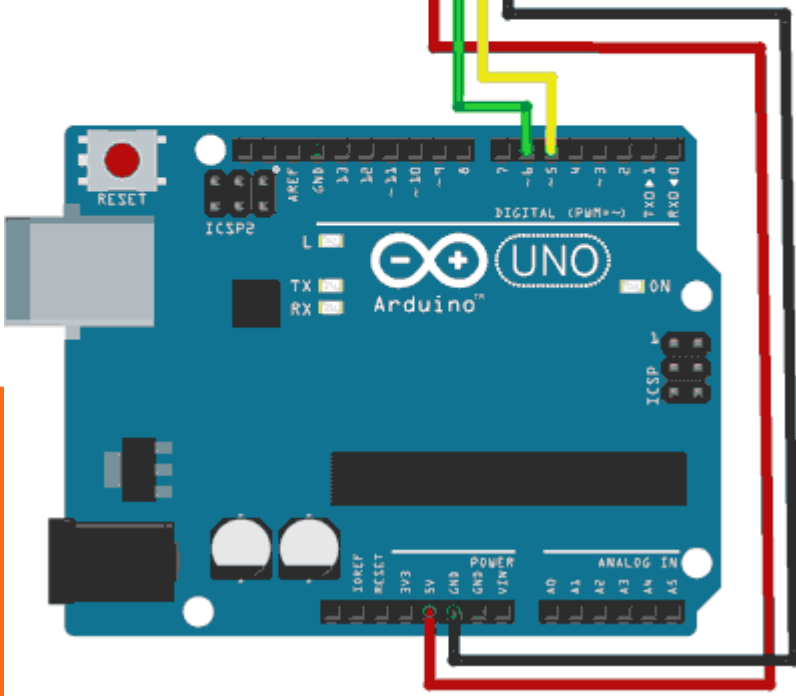
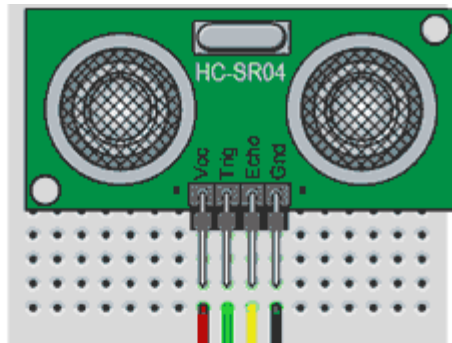


3.5. Conexión del Arduino a un sensor ultrasónico:



$$\text{Tiempo} = 2 * (\text{Distancia} / \text{Velocidad})$$

$$\text{Distancia} = \text{Tiempo} \cdot \text{Velocidad} / 2$$



3.4. Conexión sencilla a un sensor de ultrasonido:

```
programaSensorUS_guay | Arduino 1.0.6
Archivo Editar Sketch Herramientas Ayuda

programaSensorUS_guay$
// 3DSolidBot by Marcos Sanchez , enero 2018 :   www.3dsolid.es

#include <NewPing.h>
|
int ledPin = 7; // para blink de prueba

/* CODIGO SENSOR ULTRASONICO : Aqui se configuran los pines donde debemos conectar el sensor*/
#define TRIGGER_PIN 13 // Arduino pin tied to trigger pin on the ultrasonic sensor. Cable MORADO
#define ECHO_PIN 12 // Arduino pin tied to echo pin on the ultrasonic sensor. Cable GRIS
#define MAX_DISTANCE 100 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.

/*Crear el objeto de la clase NewPing*/
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() { ///////////////////////////////////////////////////SETUP////////////////////////////////////SETUP////////////////////////////////
  Serial.begin(9600); // set up Serial library at 9600 bps

  pinMode(ledPin, OUTPUT);

}

void loop() { ///////////////////////////////////////////////////LOOP////////////////////////////////////LOOP////////////////////////////////
  delay(100);
  Serial.print("Ping: ");
  Serial.print(sonar.ping_cm()); // Send ping, get distance in cm and print result (0 = outside set distance range)
  Serial.println("cm");

  if ((sonar.ping_cm())<25&&(sonar.ping_cm())>0) { //si está muy cerca : se enciende el LED //DESCARTAMOS EL VALOR 0 porque ouede dar errores
    digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(400);
    digitalWrite(ledPin, LOW); // turn the LED off by making the voltage LOW
    delay(300);
  }

  else{ // SI NO HAY CERCA UN OBSTACULO...
    // ACCION A DETERMINAR, POR EJEMPLO, ANDAR HACIA DELANTE
  }
}
```

3.5. Conexión a un sensor US con Ardublock:

The screenshot displays the Ardublock IDE interface. The left sidebar contains a category list: Control, Pins, Pruebas, Operadores Matemáticos, Variables/Constantes, Hardware genérico, Comunicación, SCoop (Multitask), Amacemar, Redes, Bloques Código, TinkerKit, DFRobot, Seeed Studio Grove, Añadir DuinoEDU Grove, Arduino Esplora, Adafruit, Makeblock, Insect Bot, 4Drawing, LittleBits, Keenlon, and Jerusalab. The main workspace shows a program with the following blocks:

- do** loop containing:
 - configura variable integer** block with **variable** set to **distancia** and **valor** set to the **ultrasonico** sensor block.
 - impresion serial** block with **message** set to **glue** and **distancia**.
 - condición** block with an **and** condition:
 - distancia** **>** **0**
 - distancia** **<** **25**
 - entonces** block containing:
 - configura pin digital** block with **#** set to **10** and **config** set to **HIGH**.
 - retardo en msg. milisegundos** block with **milisegundos** set to **300**.
 - configura pin digital** block with **#** set to **10** and **config** set to **BAJO**.
 - retardo en msg. milisegundos** block with **milisegundos** set to **200**.

The **ultrasonico** sensor block is configured with **trigger #** 13 and **echo #** 12. The **do** loop is labeled **bucle** on the left. The **condición** block is labeled **si** on the left.

3.5. Conexión avanzada a un sensor ultrasónico:

```
programaSensorUS_guay$
// 3DSolidBot by Marcos Sanchez :   www.3dsolid.es
//Nov 2018

#include <NewPing.h> // cargamos libreria "NewPing" para facilitar el uso del sensor de ultrasonidos

#include <Servo.h>    //cargamos libreria "Servo" para facilitar el control de motores servo

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

Servo servoBack; // create a servo object  SERVO DELANTERO
Servo servoFront; // create a servo object  SERVO TRASERO

int ledPin = 7; // para blink de prueba
/* CODIGO ZUMBADOR
const int pinBuzzer = A4; // pin for buzzer(zumbador)

////L0 siguiente es para tonos del zumbador

    int numTones = 10;
    int tones[ ] = {261, 277, 294, 311, 330, 349, 370, 392, 415, 440,466, 494};
                    // mid C C# D D# E F F# G G# A
*/

/* CODIGO SENSOR ULTRASONICO : Aqui se configuran los pines donde debemos conectar el sensor*/
#define TRIGGER_PIN 13 // Arduino pin tied to trigger pin on the ultrasonic sensor.Cable MORADO
#define ECHO_PIN    12 // Arduino pin tied to echo pin on the ultrasonic sensor. Cable GRIS
#define MAX_DISTANCE 100 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm

/*Crear el objeto de la clase NewPing*/
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

int rotBack; // ROTACION motor de atras
int rotFront; // ROTACION motor de adelante

int d; // delay corto
int dMedium; // delay mediano
int dLong; //delay largo

int ajusteFinoF = 4; //grados de ajuste fino para que el engranaje del servo quede a 90 perfectos F = FRONT = servo delantero
int ajusteFinoB = 11; //grados de ajuste fino para que el engranaje del servo quede a 90 perfectos B= BACK = servo trasero
```


3.5. Conexión avanzada a un sensor US (Cont.):

```
void setup() {          //////////////////////////////////////SETUP////////////////////////////////////SETUP////////////////////////////////////
    Serial.begin(9600);    // set up Serial library at 9600 bps

    servoBack.attach(3); // servo de atras vinculado a la salida 2
    servoFront.attach(2); // servo de adelante vinculado a la salida 3

    // codigo ZUMBADOR /// pinMode(pinBuzzer, OUTPUT);
    pinMode(ledPin, OUTPUT);
    d = 25;
    dMedium = 100;
}

void loop() {          //////////////////////////////////////LOOP////////////////////////////////////LOOP////////////////////////////////////
    delay(100);
    Serial.print("Ping: ");
    Serial.print(sonar.ping_cm()); // Send ping, get distance in cm and print result (0 = outside set distance range)
    Serial.println("cm");

    if ((sonar.ping_cm() < 20) && (sonar.ping_cm() > 0)) { //si está muy cerca...se enciende el led
        digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
        delay(400);
        digitalWrite(ledPin, LOW); // turn the LED off by making the voltage LOW
        delay(300);
    }

    else{ // SI NO HAY CERCA UN OBSTACULO, ANDA HACIA DELANTE: FRONTWALK
        frontWalk(30);
        frontWalk(30);
    }
}

void frontWalk (int anguloGiro)
{
    servoFront.write(120 + ajusteFinoF);
    delay(dMedium);
    servoBack.write(120 - anguloGiro - 5 + ajusteFinoB);
    delay(dMedium);

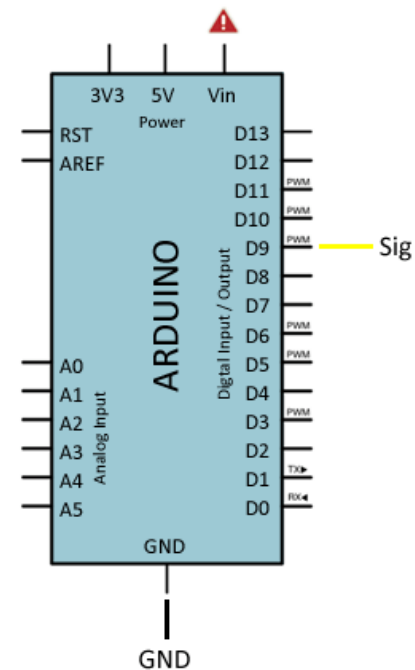
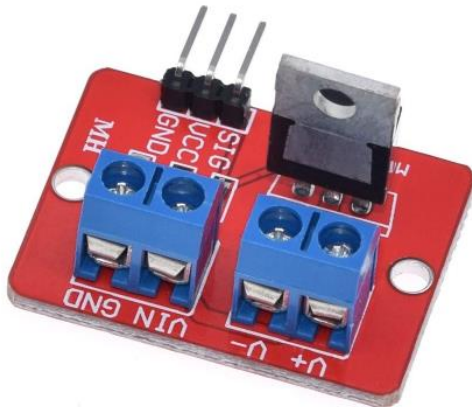
    servoFront.write(120 - anguloGiro - 10 + ajusteFinoF);
    delay(dMedium);
    servoBack.write(120 + 5 + ajusteFinoB);
    delay(dMedium);
}
```

3.6. Conexión del Arduino a un mosfet:

1. Se puede definir como un interruptor controlado eléctricamente que permite al Arduino controlar dispositivos con necesidades de tensión o intensidad muy superiores a las que él les podría suministrar con sus salidas.

Por ejemplo, el siguiente código simplemente enciende y apaga la carga en intervalos de 5 segundos.

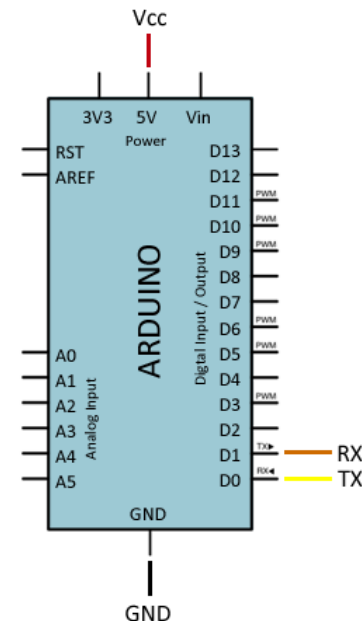
```
1  const int pin = 9;
2
3  void setup() {
4      pinMode(pin, OUTPUT); //definir pin como salida
5  }
6
7  void loop(){
8      digitalWrite(pin, HIGH); // poner el Pin en HIGH
9      delay(5000);             // esperar un segundo
10     digitalWrite(pin, LOW);  // poner el Pin en LOW
11     delay(5000);             // esperar un segundo
12 }
```



Si usáis alimentación externa al sensor, cuidado con la alimentación de Arduino

3.7. Conexión del Arduino a un módulo Bluetooth:

1. La conexión Bluetooth nos permitirá una **comunicación inalámbrica** con nuestro Arduino, usando un puerto serie de nuestro Arduino. Mientras cargamos un programa en nuestro Arduino Uno o Nano debemos recordar desconectar el módulo BT dado que los programas se cargan a través del puerto serie (salvo que usemos Arduino Mega que tiene 4 puertos serie).
2. Hay dos tipos de módulos BT: **hc-05**, de 6 pines, puede enviar y recibir (master), **hc-06**, de 4 pines solo puede recibir (slave).
3. Al conectar es **IMPORTANTE** cruzar los conectores Tx y Rx, con los del Arduino: esto es, conectar el Tx del módulo con el Rx del Arduino y viceversa.



3.7. Conexión del Arduino a un módulo Bluetooth:

1. Ejemplo de programa para encender y apagar un LED conectado a D8:

The screenshot shows the Arduino IDE interface with the following code blocks:

```
configuración
  configura pin digital # D8 BAJO
  cabecera #include <SoftwareSerial.h>
  cabecera 2 SoftwareSerial BTSerial(16,17);
  configuración BTSerial.begin(9600);
  configuración BTSerial.flush();
  retardo en msg. milisegundos 200

bucle
  bucle ABVAR 1 a = BTSerial.available();
  condición a > 0
  entonces
    bucle char ABVAR 2 command = BTSerial.read();
    condición command == F
    entonces
      configura pin digital # D8 HIGH
      impresión serie message2 pin 8 ON
      nueva línea true
    condición command == B
    entonces
      configura pin digital # D8 BAJO
      impresión serie message2 pin 8 OFF
      nueva línea true
```

A yellow note box on the right side of the code area contains the text: "16,17 son A2 y A3".

4. Código final del robot: Gamakerbot.

_3DSolidBot_V9_20181216

```
}

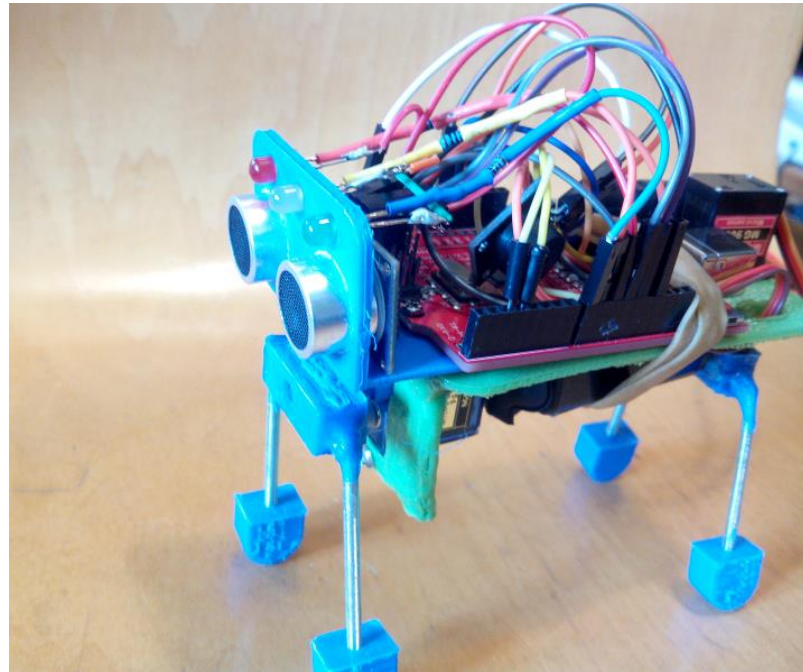
void loop() {          ////////////////////////////////////LOOP//////////////////////////////////////LOOP//////////////////////////////////////

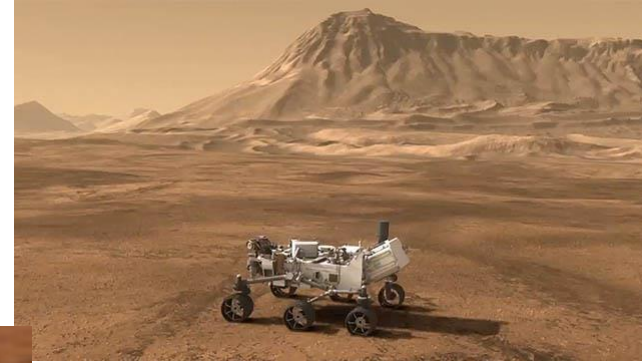
    digitalWrite(yled, HIGH); //enciende amarillo
    digitalWrite(rled, LOW);  //apaga rojo
    digitalWrite(gled, LOW);  // apaga verde

    // delay(200);          // Wait 200ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
    Serial.print("Ping: ");
    Serial.print(sonar.ping_cm()); // Send ping, get distance in cm and print result (0 = outside set distance range)
    Serial.println("cm");

    if ((sonar.ping_cm()<10)&&(sonar.ping_cm()>0)) {          //si está muy cerca, el obstaculo se enciende red= ROJO
        digitalWrite(yled, LOW); //apaga led amarillo
        digitalWrite(rled, HIGH); // se enciende led rojo
        zumbidol();
        centradoServos();
        backWalk(30); // no poner mas de 30 en anguloGiro backWalk
        backWalk(30); // no poner mas de 30 en anguloGiro backWalk
        backWalk(30); // no poner mas de 30 en anguloGiro backWalk
        backWalk(30); // no poner mas de 30 en anguloGiro backWalk
        backWalk(30); // no poner mas de 30 en anguloGiro backWalk
        delay (50);
        //zumbidol();
        centradoServos();
    // la parte de turnLeft aun esta por pulir...
        turnLeft(20);
        turnLeft(20);
        turnLeft(20);
        turnLeft(20);
        turnLeft(20);
        turnLeft(20);
        zumbidol();
        delay (50);
    }

    else{
        digitalWrite(yled, LOW);
        digitalWrite(gled, HIGH); // se enciende el led verde
        frontWalk(30);
        //delay (200);
    }
}
```





www.gamaker.org
www.3dsolid.es