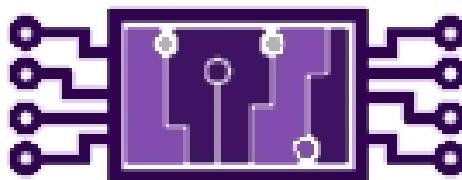
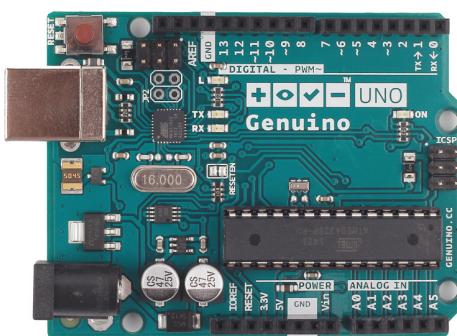
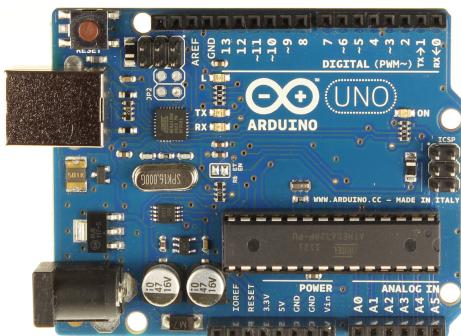


# Genuino Day 2016

Taller  
"Arduino con Ardublock"  
*100% Practico*



MacMaker





## MANUAL DEL TALLER



# Noroeste



Propósito .....	4
¿Qué voy a aprender? .....	5
1. Introducción.....	6
1A. Computación física.....	7
2. Para tener en cuenta .....	8
2A. Arduino Mazatlán.....	8
2B. Patrocinadores .....	8
2C. Hardware libre .....	9
2D. Software libre .....	9
2E. Creative commons .....	9
2F. Licencia de la obra .....	10
2G. Referencia rápida para programar.....	11
3. Arduino .....	13
3A. Proyecto Arduino .....	15
3B. Placa Arduino Uno y sus partes .....	14
3C. Instalando drivers.....	16
3D. Conociendo el software Arduino.....	19
3E. Cargando mi primer programa.....	20
4. Tutoriales .....	23
T1. Hola Mundo - LED intermitente.....	23
T2. Encender un LED con un pulsador .....	27
T3. Lectura serial de una entrada digital .....	31
T4. Lectura serial de una entrada analógica .....	35
T5. Control de un Servo Motor .....	59
5. Ardublock .....	74
5A. Introducción a ArduBlock.....	74





Genuino  
DAY 2016

## PROPOSITO

Conocer el funcionamiento de las cosas es algo que nos hemos planteado desde el inicio de los tiempos; hoy en día nos enfrentamos a una realidad donde abundan la automatización, la domótica (automatización de las casas y edificios), la interacción de las personas con las máquinas, la electrónica, la mecánica y la programación.

Casi cualquier proceso que nos podamos imaginar tiene un porcentaje de dependencia de estas máquinas, por ejemplo: Tu despertador sonó a las 6am para que vinieras a la escuela o fueras al trabajo, esa máquina, reloj, trabajó durante toda la noche para al final avisarte que era hora de despertar.

El propósito de esta guía es abordar el concepto de computación física que es la capacidad de interacción y comunicación de una máquina con los humanos, usando sensores y actuadores. Las decisiones de esto las va a tomar un microcontrolador que se encuentra ubicado en la placa Arduino. La tarjeta Arduino es el corazón de la presente guía.

# ¿QUÉ VOY A APRENDER?

Muchas veces pensamos que los temas tecnológicos requieren de gran habilidad técnica y de un gran conocimiento, pero esto no es cierto. Queremos que con el desarrollo de esta guía entiendas que muchos de esos procesos tecnológicos son simples de entender y aquellos que son complejos son la unión de muchos procesos simples.

En esta guía vas a aprender a imaginar y aterrizar todas ideas a conceptos tangibles de los cuales te puedes sentir orgulloso, ya que fue tu idea y tu lo desarrollaste ;)

F  
U  
E  
N  
T  
E  
  
D  
E  
  
I  
N  
F  
O  
R  
M  
A  
C  
I  
O  
N

Wikipedia es una de las enciclopedia en la nube más grande que pueden existir, puedes encontrar gran variedad de información en distintos idiomas y eres libre de usarla para aprender.



La presente guía incorpora contenido de Wikipedia (texto e imágenes) con el ánimo de explicar los diversos conceptos que se enuncian. El contenido de la Wikipedia tomado en esta guía ha sido transcrita textualmente en algunos casos, en otros casos los conceptos se han reeditado para poder comprender más fácilmente la idea.

Para referenciar que hemos tomado contenido de Wikipedia, al lado de cada concepto técnico vas a encontrar el logo de Wikipedia de esta manera podrás leer más contenido si buscas ese mismo concepto en la Wikipedia.

Wikipedia es de contenido libre, de manera que todo el texto está disponible bajo la [Licencia Creative Commons-Atribución-Compartir Igual 3.0\(CC-BY-SA\)](#). La mayor parte del contenido también está disponible bajo la [Licencia de Documentación Libre GNU \(GFDL\)](#). Esto significa que el contenido de Wikipedia se puede distribuir y enlazar de acuerdo con lo establecido en estas licencias.

# 1. INTRODUCCIÓN

MazMaker, presentan este manual de prácticas que abordan el aprendizaje sobre el concepto DIY (Do it yourself) o en español —Hazlo tú mismo”. Luego de una cuidadosa selección de componentes electrónicos y apoyados en la placa Arduino se crea el producto Kit Básico de Arduino apoyado de esta guía.

Abordamos temas fundamentales como el hardware y software libre, revisando de manera cuidadosa el proyecto Arduino y apoyándonos en el Software Fritzing para lograr montajes muy llamativos y semejantes a la realidad. No es necesario que sepas de electrónica y programación

Finalmente llegamos a una parte muy especial, al capítulo de los tutoriales, donde paso a paso se explican los ejemplos, durante el recorrido de aprendizaje te encontrarás con preguntas, tips y ejercicios.





Genuino  
DAY 2016

## Computación física

1A

La **Computación física**, significa la construcción de sistemas interactivos físicos mediante el uso de software y hardware que pueden sentir y responder al mundo analógico. Si bien esta definición es suficientemente amplia para abarcar aspectos como los sistemas inteligentes de control de tráfico de automóviles o los procesos de automatización de fábricas, en un sentido más amplio, la computación física es un marco creativo para la comprensión de la relación de los seres humanos en el mundo digital. En la práctica, a menudo el término describe el arte hecho a mano, diseño de proyectos DIY o pasatiempos que utilizan sensores y microcontroladores para traducir entradas analógicas a sistemas basados en software, y/o controlar dispositivos electromecánicos como motores, servos, iluminación u otro hardware.

Otras implementaciones de computación física trabajan con el reconocimiento de la voz, la cual se capta e interpretan sus ondas sonoras a través de micrófonos u otros dispositivos de detección de ondas sonoras, también la visión por computador, que aplica algoritmos a los videos detectados por algún tipo de cámara. Interfaces táctiles son también un ejemplo de la computación física.

El prototipado (crear montajes rápidos con ayuda de una proto-board y componentes básicos de electrónica) juega un papel importante en la computación física. Herramientas como Arduino y Fritzing son útiles para diseñadores, artistas, estudiantes y entusiastas porque ayudan a elaborar prototipos rápidamente.



## 2. PARA TENER EN CUENTA

Te presentamos una información de interés que te recomendamos la tengas en cuenta para el desarrollo de la presente guía. Conoce más acerca de los desarrolladores y los pilares de este excelente material.

### MazMaker



### MazMaker

Electrónica, Informática, Refrigeración, etc. de la ciudad y Puerto de Mazatlán, Sinaloa, México.

La responsabilidad social que el mundo nos exige hace que por medio del sitio Web: <http://www.mazportal.com> podamos compartir: noticias, eventos, tutoriales, proyectos, etc. Así mismo proximamente se abrirá La Tienda virtual. Agradecemos de antemano tu preferencia y que compres nuestros productos con el mejor precio del mercado. Contáctanos para conocer más de nosotros.

2A

### PATROCINADORES



**Noroeste**



2B

## Hardware libre



Se llama hardware libre a los dispositivos de hardware cuyas especificaciones y diagramas esquemáticos son de acceso público, ya sea bajo algún tipo de pago o de forma gratuita. La filosofía del software libre (las ideas sobre la libertad del conocimiento) es aplicable a la del hardware libre. Se debe recordar en todo momento que libre no es sinónimo de gratis. El hardware libre forma parte de la cultura libre.

Dado que el hardware tiene asociados a él costos variables directos, ninguna definición de software libre se puede aplicar directamente sin modificación. En cambio, el término hardware libre se ha usado principalmente para reflejar el uso del software libre con el hardware y el lanzamiento libre de la información con respecto al hardware, a menudo incluyendo el lanzamiento de los diagramas esquemáticos, diseños y montajes.



## Software libre



El software libre (en inglés free software, aunque esta denominación también se confunde a veces con "gratis" por la ambigüedad del término "free" en el idioma inglés, por lo que también se usa "libre software" y "logical libre") es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado, y redistribuido libremente. Según la Free Software Foundation, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el software y



distribuirlo modificado.

## Creative commons



Creative Commons (CC) es una organización no gubernamental sin ánimo de lucro que desarrolla planes para ayudar a reducir las barreras legales de la creatividad, por medio de nueva legislación y nuevas tecnologías. Las licencias Creative Commons o CC están inspiradas en la licencia GPL (General Public License) de la Free Software Foundation, compartiendo buena parte de su filosofía. La idea principal detrás de ellas es posibilitar un modelo legal ayudado por herramientas informáticas, para así facilitar la distribución y el uso de contenidos.

Existe una serie de licencias Creative Commons, cada una con diferentes configuraciones, que permite a los autores poder decidir la manera en la que su obra va a circular en internet, entregando libertad para citar, reproducir, crear obras derivadas y ofrecerla públicamente, bajo ciertas diferentes restricciones.

La licencia de la presente obra se expone en la página 11.



## Licencia de la obra

### Usted es libre de:

Compartir - copiar, distribuir, ejecutar y comunicar públicamente la obra  
hacer obras derivadas

### Bajo las condiciones siguientes:



**Atribución** — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciatario (pero no de una manera que sugiera que tiene su apoyo o que apoyan el uso que hace de su obra).



**No Comercial** — No puede utilizar esta obra para fines comerciales.



**Compartir bajo la Misma Licencia** — Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

### Entendiendo que:

**Renuncia** — Alguna de estas condiciones puede **no aplicarse** si se obtiene el permiso del titular de los derechos de autor

**Dominio Público** — Cuando la obra o alguno de sus elementos se halle en el **dominio público** según la ley vigente aplicable, esta situación no quedará afectada por la licencia.

**Otros derechos** — Los derechos siguientes no quedan afectados por la licencia de ninguna manera:

- Los derechos derivados de **usos legítimos** u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.
- Los derechos **mORALES** del autor;
- Derechos que pueden ostentar otras personas sobre la propia obra o su uso, como por ejemplo **derechos de imagen** o de privacidad.

**Aviso** — Al reutilizar o distribuir la obra, tiene que dejar muy en claro los términos de la licencia de esta obra. La mejor forma de hacerlo es enlazar a esta página.

[www.creativecommons.org/licenses/by-nc-sa/3.0/deed.es](http://www.creativecommons.org/licenses/by-nc-sa/3.0/deed.es)



El programa de Arduino se puede dividir en tres partes principales: la estructura, las variables (valores y constantes) y funciones.

## ESTRUCTURA

- setup()
- loop()

### +Estructuras de control

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

### +Sintaxis

- ; (punto y coma)
- {} (llaves)
- // (comentario de una sola línea)
- /\* \*/ (comentario de varias líneas)
- # define
- # include

### +Operadores matemáticos

- = (operador de asignación)
- + (suma)
- - (resta)
- \* (multiplicación)
- / (división)
- % (módulo)

### +Operadores de comparación

- == (igual que)
- != (diferente de)
- < (menor que)
- > (mayor que)
- <= (menor o igual a)
- >= (mayor o igual a)

### +Operadores booleanos

- && (y)
- || (o)
- ! (no)

### +Acceso con apuntadores

- \* eliminar la referencia del operador
- & operador de referencia

### +Operadores bit a bit

- & (bit a bit AND)
- | (bit a bit OR)
- ^ (bit a bit XOR)
- ~ (bit a bit NOT)
- << (a la izquierda BitShift)
- >> (a la derecha BitShift)

### +Operadores compuestos

- += (compuesto adición)
- -= (compuesto substracción)
- \*= (compuesto multiplicación)
- /= (compuesto división)
- &= (compuesto bit a bit AND)
- |= (compuesto bit a bit OR)

## VARIABLES

### +Constantes

- HIGH | LOW
- INPUT | OUTPUT
- true | false
- Constantes enteras
- Constantes flotante

### +Tipos de datos

- void
- boolean
- char
- byte
- int
- word
- long
- unsigned long
- float
- double
- string - arreglo char
- String - objeto
- array

### +Utilidades

- sizeof()
- char()
- byte()
- int()
- word()
- long()
- float()

### +Conversión

**+Digital I/O**

- pinMode()
- digitalWrite()
- digitalRead()

**+Analog I/O**

- analogReference()
- analogRead()
- analogWrite() - PWM

**+Avanzadas I/O**

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

**+Tiempo**

- millis()
- micros()
- delay()
- delayMicroseconds()

**+Matemáticas**

- min()
- max()
- abs()
- constrain()
- map()
- pow()
- sqrt()

**+Trigonométricas**

- sin()
- cos()
- tan()

**+Números aleatorios**

- randomSeed()
- random()

**+Bits y Bytes**

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

**+Interrupciones externas**

- attachInterrupt()
- detachInterrupt()

**+Interrupciones**

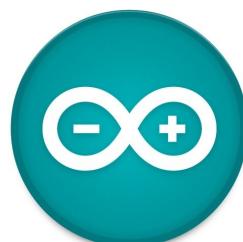
- interrupts()
- noInterrupts()

**+Comunicación**

- Serial
  - begin()
  - end()
  - available()
  - read()
  - peek()
  - flush()
  - print()
  - println()
  - write()

- EEPROM - leer y escribir
- Ethernet - conectarse a Internet
- Cristal líquido - control de LCD
- SD - lectura y escritura de tarjetas SD
- Servo - control de servomotores
- SPI - comunicación por el bus SPI
- Paso a paso - control de motores
- Wire - enviar y recibir datos TWI/I2C

... y muchas más visitas



[arduino.cc](http://arduino.cc)  
[genuino.cc](http://genuino.cc)

### 3. ARDUINO

Con las capacidades de Arduino solo debemos tener nociones básicas de electrónica y programación, eso es suficiente para comenzar a desarrollar nuestros proyectos. Arduino cuenta con una gran comunidad donde se comparte todo lo desarrollado y es una gran ventana para que puedas ver todo lo que es posible desarrollar.

#### Proyecto Arduino



El proyecto comenzó en Ivrea, Italia (el sitio de la compañía de computadoras Olivetti), en el año 2005 con el fin de crear un dispositivo para estudiantes para el control integrado de proyectos de diseño e interacción, con la finalidad de que fuera más barato que los sistemas de creación de prototipos disponibles en ese entonces. A partir de mayo de 2011, más de 300.000 unidades de Arduino han sido distribuidas. Los fundadores Massimo

Banzi y David Cuartielles nombraron el proyecto como Arduino de Ivrea, un protagonista histórico de la ciudad. En primer lugar "Arduino" es un término masculino italiano, que significa "gran amigo".

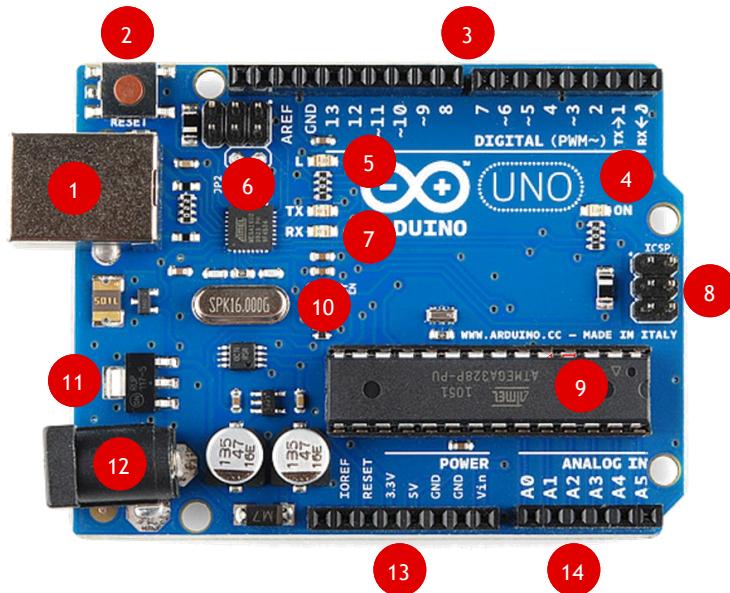
El proyecto Arduino es un fork (en la ingeniería de software, un fork es un proyecto que sucede cuando los desarrolladores tienen una copia legal del código fuente y empiezan el desarrollo independiente de ella, creando una obra distinta de software) de la plataforma Wiring de código abierto. Wiring fue creado por el artista colombiano y programador Hernando Barragán como una tesis de maestría en el Instituto de diseño e interacción Ivrea, bajo la supervisión de Massimo Banzi y Casey Reas. Por otra parte, Wiring se basa en Processing y su entorno de desarrollo integrado creado por Casey Reas y Ben Fry.



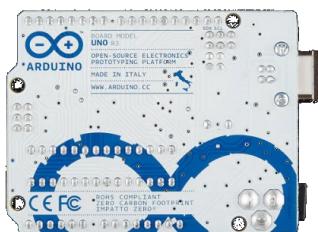
"Arduino fue construido en torno al proyecto Wiring de Hernando Barragán. Wiring fue el proyecto de tesis de Hernando en el Instituto de diseño e interacción Ivrea. Fue pensado para ser una versión electrónica de Processing que utiliza nuestro entorno de programación y fue modelado para la sintaxis de Processing. Fue supervisado por mí mismo y Massimo Banzi, un fundador de Arduino. No creo que Arduino existiría sin Wiring y no creo que Wiring existiría sin Processing. Y sé que Processing sin duda no existiría sin Design By Numbers y John Maeda<sup>1</sup>"



## Vista frontal



## Vistas auxiliares



- 1 Conector USB para el cable Tipo AB
- 2 Pulsador de Reset
- 3 Pines de E/S digitales y PWM
- 4 LED verde de placa encendida
- 5 LED naranja conectado al pin13
- 6 ATmega 16U2 encargado de la comunicación con el PC
- 7 LED TX (Transmisor) y RX (Receptor) de la comunicación serial
- 8 Puerto ICSP para programación serial
- 9 Microcontrolador ATmega 328, cerebro del Arduino
- 10 Cristal de cuarzo de 16Mhz
- 11 Regulador de voltaje
- 12 Conector hembra 2.1mm con centro positivo
- 13 Pines de voltaje y tierra
- 14 Entradas análogas



## MAC y LINUX

Si tu computadora tiene de sistema operativo alguna versión de Mac o una distribución de LINUX, lo único que debes hacer es:

- 1 Conectar la placa Arduino Uno al PC
- 2 Descargar el software de [arduino.cc/en/Main/Software](http://arduino.cc/en/Main/Software)
- 3 Listo para trabajar y cargar programas

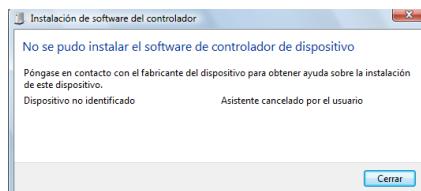


## WINDOWS 8.1, 7, Vista y XP

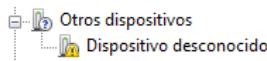
Si tu computadora tiene de sistema operativo Windows en versión 8.1, 7, Vista o XP, debes realizar la siguiente sucesión de sencillos paso.

- 1 Descargar el software de [arduino.cc/en/Main/Software](http://arduino.cc/en/Main/Software) para Windows  

- 2 Descomprimir la carpeta de Arduino en una ubicación de fácil acceso
- 3 Conectar la placa Arduino Uno al PC y ver este aviso. No nos debemos preocupar

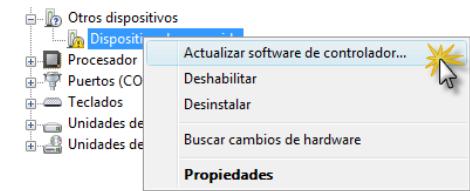


- 4 Visitar Panel de control y luego Administrador de dispositivos, allí buscar la siguiente opción

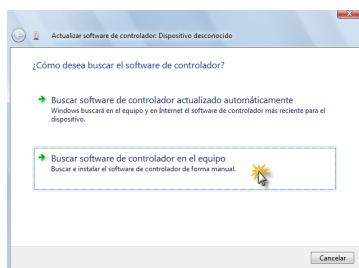


## WINDOWS 8.1, 7, Vista y XP

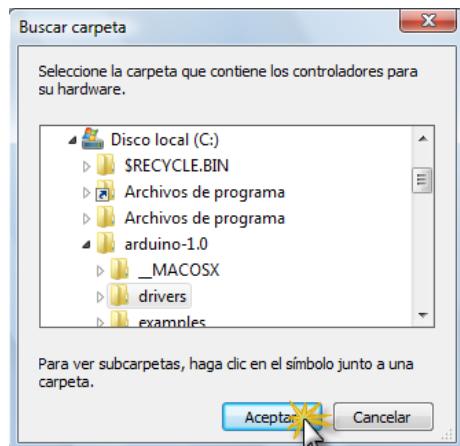
- 5 Click derecho sobre Dispositivo desconocido y luego sobre la opción Actualizar software del controlador.



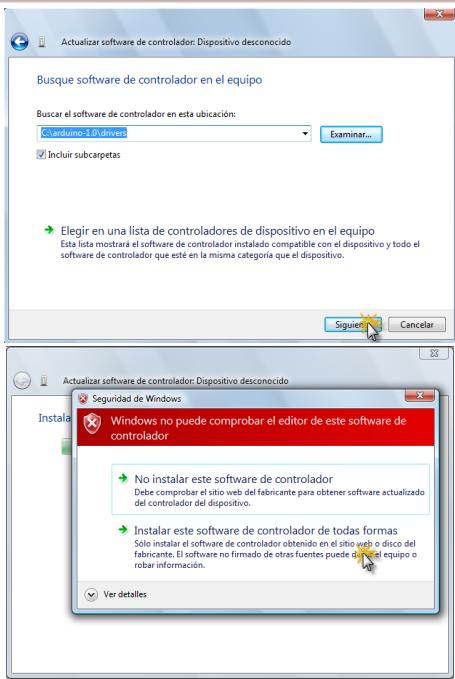
- 6 Ingresar a la opción Buscar software de controlador en el equipo



- 7 Examinar y buscar la carpeta de Arduino previamente descomprimida en el paso 2. Dentro de esa carpeta acceder a la carpeta Drivers y dar Aceptar

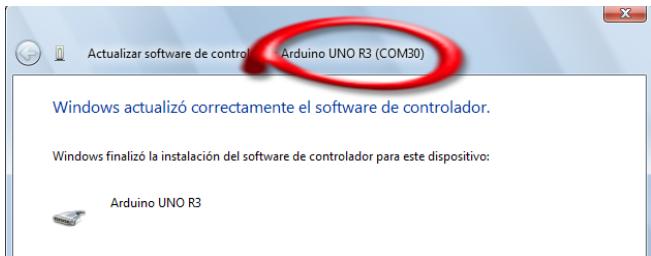


- 8 Una vez buscamos la carpeta de Drivers le damos Siguiente



9

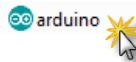
Recibimos la confirmación del Puerto COM asignado, este número de Puerto COM es muy importante tenerlo en cuenta a la hora de programar.



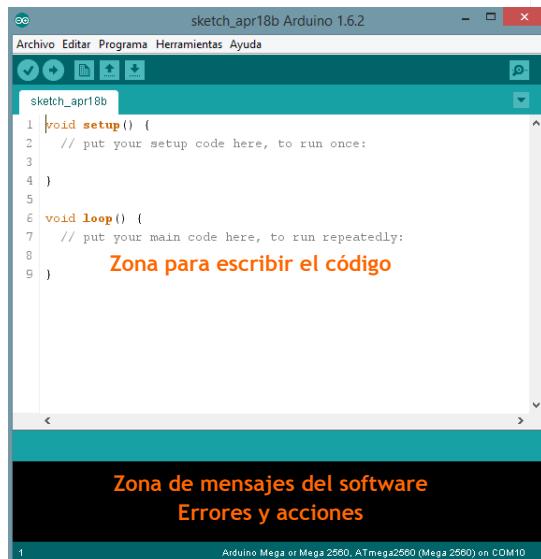
## Conociendo el software Arduino



Para ejecutar el programa Arduino, ingresamos a la carpeta de Arduino y allí buscamos el ícono de Arduino y le damos doble click

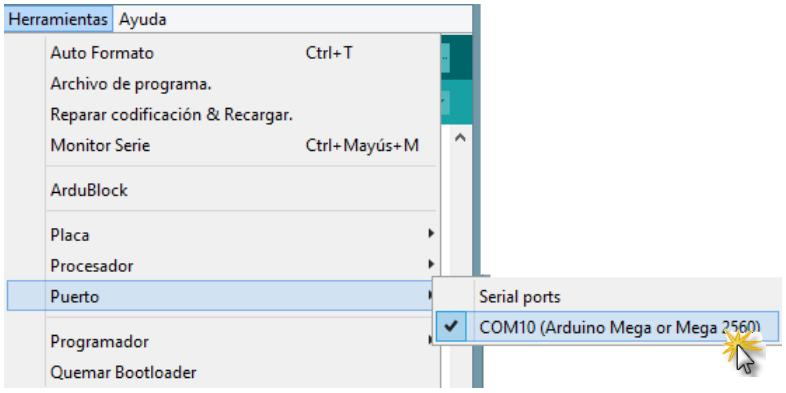


3D



1

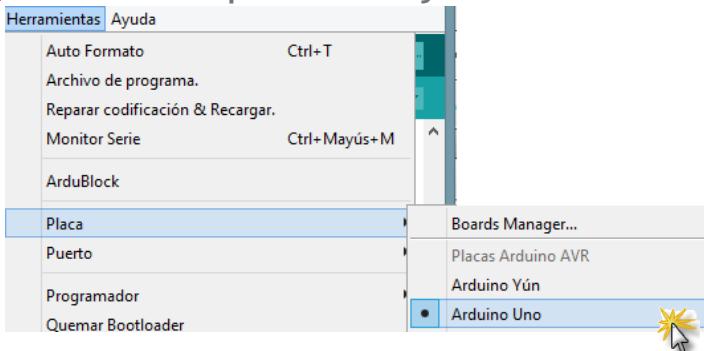
## Puerto COM



19

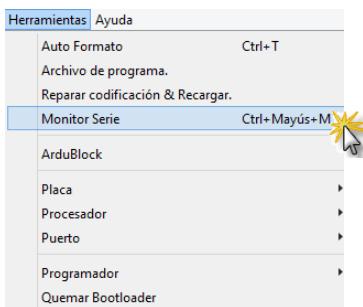
2

## Seleccionar la placa a trabajar



3

## Consola serial

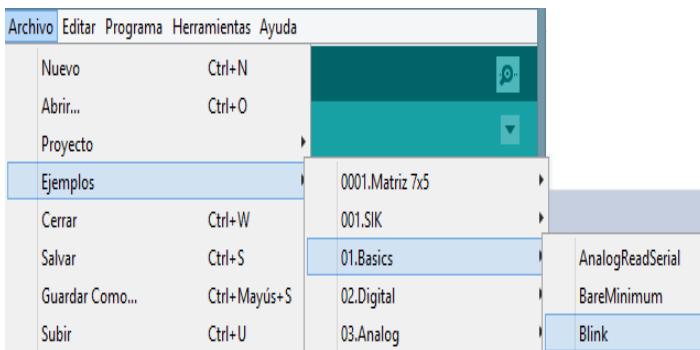


## Cargando mi primer programa

Al final de este ejercicio ya vas a tener la capacidad de cargar programas a tu placa Arduino. Para ello abre el software de Arduino y realiza lo que se indica en la siguiente imagen.

3E

1



## Cargando mi primer programa

Antes de continuar con el siguiente paso asegúrate de configurar de manera correcta:

A- Puerto COM, revisa el **Paso 1** de la sección **6F**, recuerda que el valor del puerto COM lo obtuvimos en el **Paso 9** de la sección **6E**

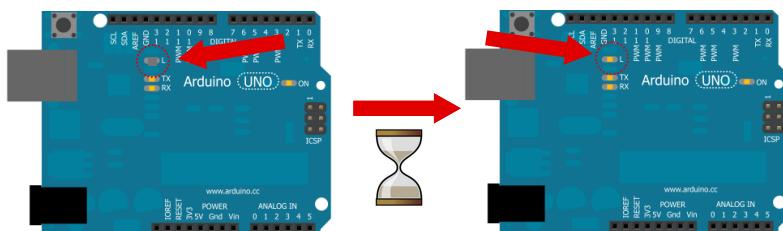
B- Placa, revisa el **Paso 2** de la sección **6F**, recuerda que para este caso la placa es Arduino UNO

2

```
/*  
Blink  
Turns on an LED on for one second, then off for one second, repeating.  
  
This example code is in the public domain.  
*/  
  
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH); // set the LED on  
  delay(1000); // wait for a second  
  digitalWrite(13, LOW); // set the LED off  
  delay(1000); // wait for a second  
}  
1  
Arduino Uno on COM32
```

3

El programa de intermitencia (Blink) que acabas de cargar en electrónica se llama “Hola mundo”, consiste en prender y apagar un LED en intervalos de un segundo. El LED que prende y apaga es la parte 5 según la sección **6D** o el marcado con la letra L según la imagen de abajo. Ahora te podemos dar la bienvenida al mundo de Arduino :D!

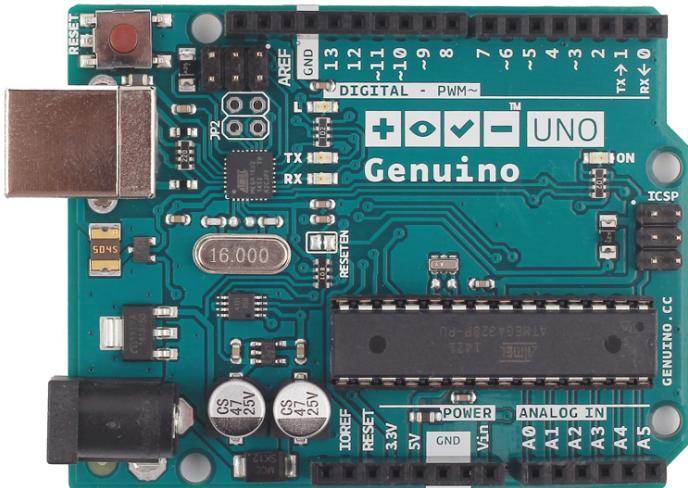


## Ejercicio

A partir del ejemplo cargado en la sección anterior, ahora te proponemos que modifiques un poco el programa, para ello en las dos líneas de código donde dice:

```
delay(1000);
```

Cambia el valor de 1000 por 2000 y vuelve a cargar el programa a tu placa Arduino Uno, ¿que observas?



**¿Qué aprendo?**

- Activar una salida digital
- Encender un LED en ON/OFF
- Temporizar una señal de salida
- Sintaxis de un programa en Arduino

**Conocimientos previos**

- Señal digital
- Función digitalWrite()
- Polaridad de un LED
- Conexión de la placa Arduino al computador

**Materiales**

Arduino UNO



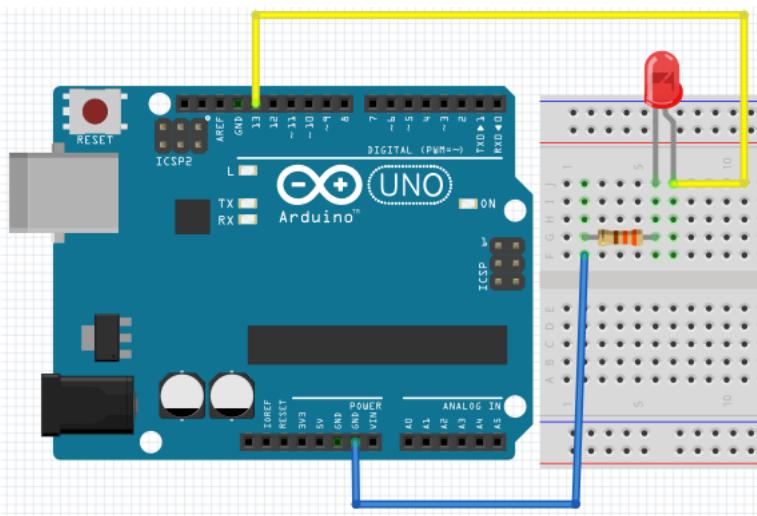
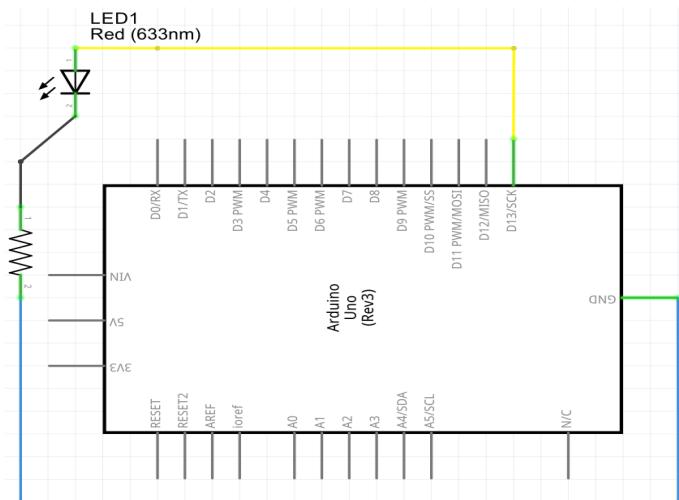
LED Verde



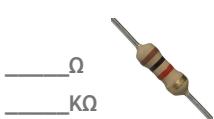
Cable USB Tipo AB



**Genuino  
DAY 2016**



1— ¿Cuál es el valor de esta resistencia?



2— ¿Qué hace esta función?

`digitalRead()`

---



---

3— Completa

$$I = \frac{?}{R}$$

C  
Ó  
D  
I  
G  
O

```
/*
-----  

  Hola Mundo  

-----  

  Enciende un LED por un segundo y lo apaga por el mismo tiempo  

*/
```

D  
E

```
//-----  

//Función principal  

//-----
```

P  
R  
O  
G  
R  
A  
M  
A  
C  
I  
O  
N

```
void setup()      // Se ejecuta cada vez que el Arduino se inicia
{
    pinMode(13,OUTPUT); // Inicializa el pin 13 como una salida
}

//-----  

//Función cíclica  

//-----
```

```
void loop()      // Esta función se mantiene ejecutando
{
    digitalWrite(13,HIGH); // Enciende el LED
    delay(1000);          // Temporiza un segundo (1s = 1000ms)
    digitalWrite(13,LOW); // Apaga el LED
    delay(1000);          // Temporiza un segundo (1s = 1000ms)
}
```

// Fin del programa

T  
I  
P  
S

1- El `//` en programación se utiliza para hacer comentarios, es muy útil para que puedas explicar algo acerca de la sintaxis de una línea de código. Un ejemplo de su uso:

`digitalWrite(13,LOW); // Apaga el LED`

2- Las señales digitales (Encendido o apagado) están muy presentes en todos los sistemas, y muchos sensores trabajan sobre este principio, te invitamos a conocer algunos:



Sensor PIR

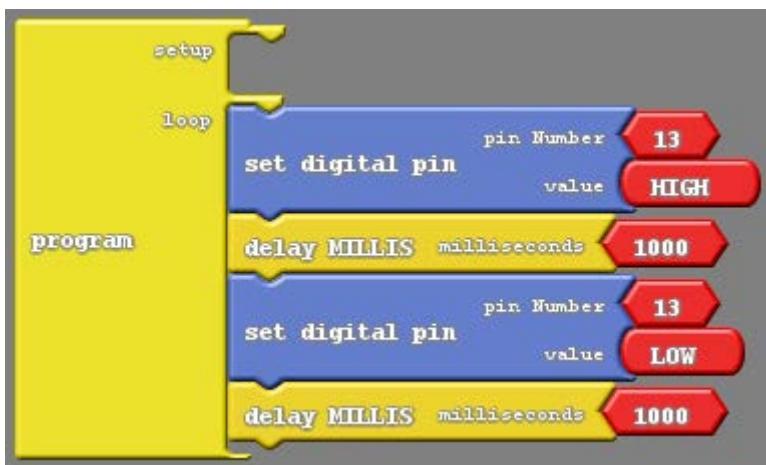
Detecta movimiento



Sensor óptico SHARP

Detecta la presencia de algún objeto en un rango de 5cm

```
/*
-----
 Hola Mundo
-----
 Enciende un LED por un segundo y lo apaga por el mismo tiempo
*/
```



**¿Qué aprendo?**

- Cablear un circuito
- Condicional If/else
- Estado de un pulsador
- Leer una entrada digital y escribir una salida digital

**Conocimientos previos**

- Señal digital
- Función digitalWrite() y digitalRead()
- Divisor de voltaje
- Condicional y operadores de comparación

**Materiales**

1



Arduino UNO

1



LED Amarillo

1



Protoboard

1



Cable USB Tipo AB

1



Pulsador

1



Resistencia 1K

4

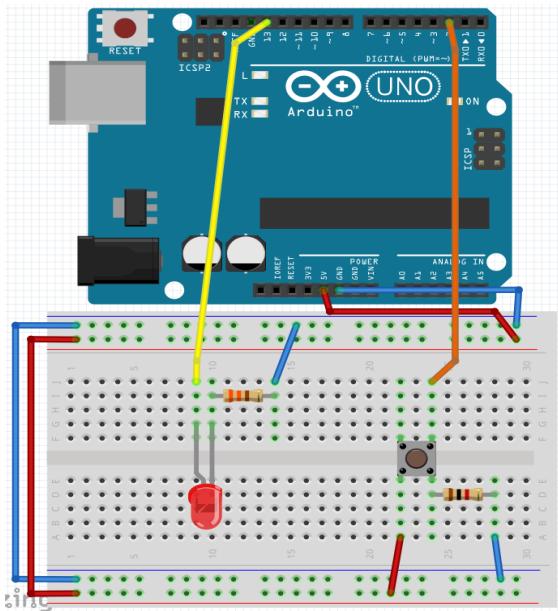
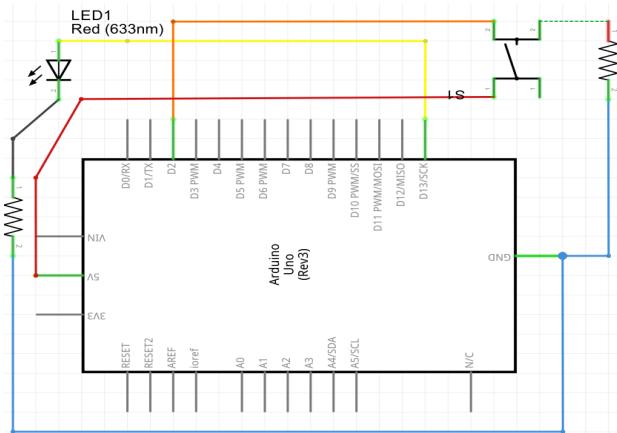


Conectores MM

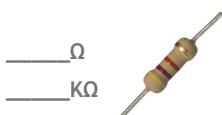


MacMaker

**Genuino**  
**DAY 2016**



**1—** ¿Cuál es el valor de esta resistencia?



**2—** ¿Qué hace esta función?

`digitalWrite()`

**3—** Un ejemplo de un lenguaje de alto nivel

C  
Ó  
D  
I  
G  
O

D  
E

P  
R  
O  
G  
R  
A  
M  
A  
C  
I  
Ó  
N

```
/*
-----  

  Encender LED con un pulsador  

-----  

  Oprimir un pulsador y mientras este se mantenga accionado  

    un LED se enciende  

*/  

//-----  

//Declara puertos de entradas y salidas  

//-----  

int pulsador=2;           //Pin donde se encuentra el pulsador, entrada  

int led=13;                //Pin donde se encuentra el LED, salida  

//-----  

//Funcion principal  

//-----  

void setup() // Se ejecuta cada vez que el Arduino se inicia  

{  

  pinMode(pulsador, INPUT); //Configurar el pulsador como una entrada  

  pinMode(led,OUTPUT);     //Configurar el LED como una salida  

}  

//-----  

//Funcion ciclica  

//-----  

void loop() // Esta funcion se mantiene ejecutando  

{  

  //Condicional para saber estado del pulsador  

  if (digitalRead(pulsador)==HIGH)  

  {  

    //Pulsador oprimido  

    digitalWrite(led,HIGH); //Enciende el LED  

  }  

  else  

  {  

    //Pulsador NO oprimido  

    digitalWrite(led,LOW); //Apaga el LED  

  }  

}  

//Fin programa
```

T  
I  
P  
S

1- Cuando estés programando en el Software de Arduino, muchas cosas de las que escribes son palabras reservadas por el lenguaje, todas las palabras reservadas las puedes encontrar en la sección **5S**, al escribirlas éstas se colocan en un color diferente, este es un truco para saber que esta bien, por ejemplo:

```
void loop() {  

  digitalWrite(13, HIGH);  

  delay(1000);  

  digitalWrite(13, LOW);  

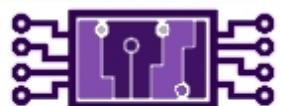
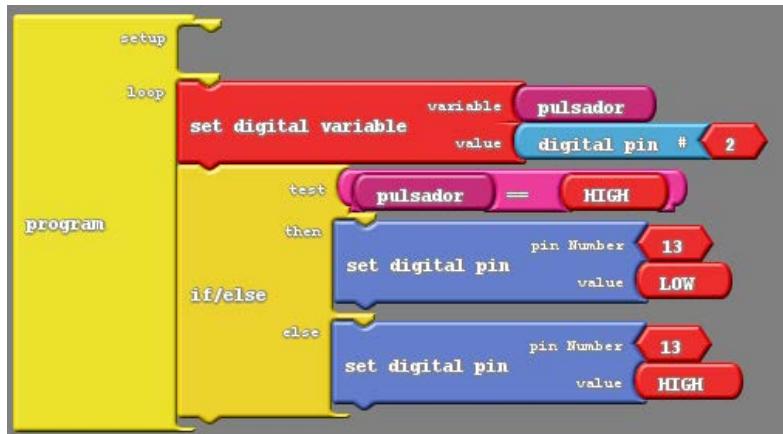
  delay(1000);  

}
```

2- Todas la instrucciones de programación para Arduino, se encuentran totalmente documentadas con claros ejemplos de cómo se utilizan, te invitamos a que visites: [arduino.cc/en/Reference/HomePage](http://arduino.cc/en/Reference/HomePage)



```
/*
-----  
Encender LED con un pulsador  
-----  
Oprimir un pulsador y mientras este se mantenga accionado  
un LED se enciende  
*/
```



MazMaker

Genuino  
DAY 2016

**¿Qué aprendo?**

- Manejar una entrada digital
- Ver datos por la pantalla del computador
- Consola serial
- Leer una entrada digital y escribir por consola serial

**Conocimientos previos**

- Señal digital
- Función `digitalRead()` y `Serial.println()`
- Opción de Consola serial, ver **6F** (paso 3)

**Materiales**

1



Arduino UNO

1



Pulsador

1



Protoboard

1



Cable USB Tipo AB

1



Resistencia 1K

4

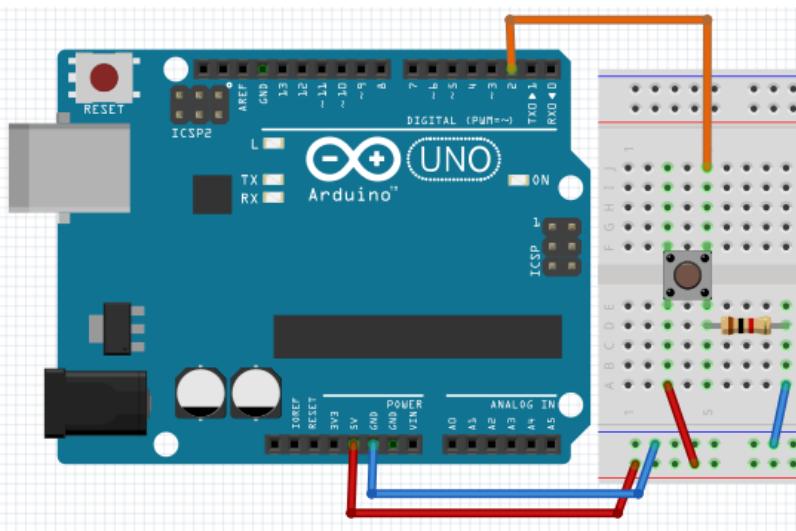
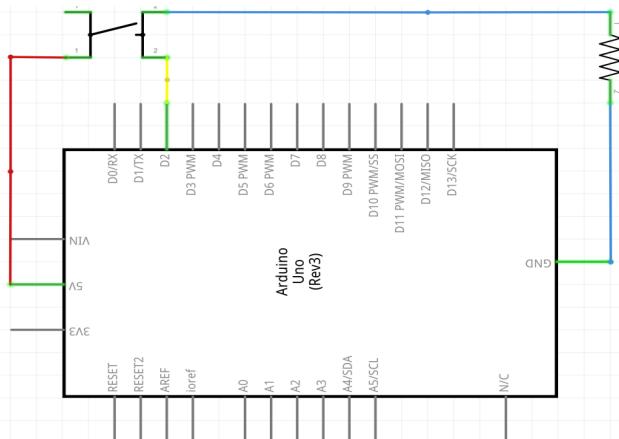


Conectores MM

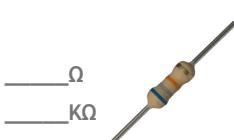


MaxMaker

**Genuino**  
**DAY 2016**



1— ¿Cuál es el valor de esta resistencia?

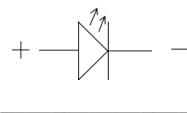


2— ¿Qué hace esta función?

**Serial.println()**



3— Este símbolo a que corresponde



```

C /*-----*/
Ó Lectura serial de una entrada digital
D -----  

I Leer una entrada digital y mostrar por la pantalla del
G computador (consola serial) el estado del pulsador
O cuando es oprimido
D */
E
P //-----  

R //Declarar puertos de entradas y salidas  

A //-----  

O int boton=2; //Pin donde se encuentra el pulsador, entrada
D //-----  

E //Funcion principal  

P //-----  

R void setup() // Se ejecuta cada vez que el Arduino se inicia
A {  

    //Configuración  

    pinMode(boton,INPUT); //Configurar el boton como una entrada  

    Serial.begin(9600); //Inicia comunicación serial
    }  

G //-----  

R //Funcion ciclica  

A //-----  

O void loop() // Esta funcion se mantiene ejecutando
M // cuando este energizado el Arduino
A
C //Guardar en una variable entera el valor del boton 0 ó 1
I int estado = digitalRead(boton);
O //Condicional para saber estado del pulsador
N if (estado==0)
{
    // Pulsado
    Serial.println("Pulsado"); //Imprime en la consola serial
    "Pulsado"
}
else
{
    // No esta pulsado
    Serial.println("NO Pulsado"); //Imprime en la consola serial
    "NO Pulsado"
}
delay(100); //Retardo para la visualización de datos en la consola
}
//Fin programa

```

## TIPS

1- La codificación binaria es muy importante para transmitir datos entre dispositivos, son las largas cadenas de 0 y 1, por ejemplo 00011101010101 esto podría ser un mensaje que contiene información referente a una clave personal para acceder a un edificio. Los números en base 10 se pueden representar como valores binarios:

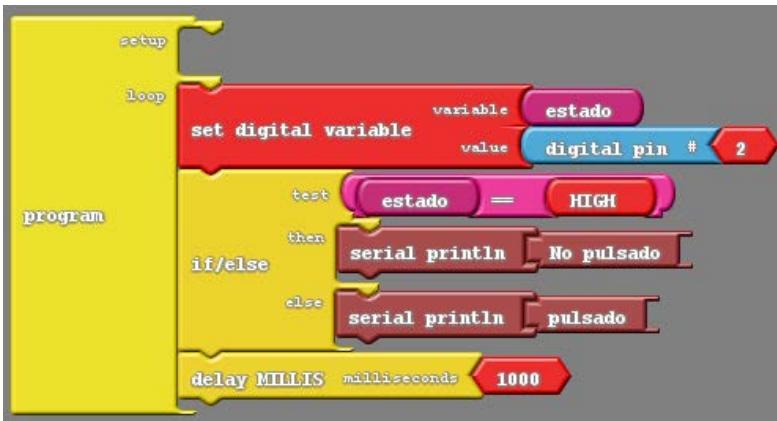
2- Para recordar



- Para leer una señal digital usa: **digitalRead(numeroPin);**
- Para escribir una señal digital usa: **digitalWrite(numeroPin, valor);**
- Una salida o entrada digital siempre es **HIGH** o **LOW**

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

```
/*  
Lectura serial de una entrada digital  
-----  
Leer una entrada digital y mostrar por la pantalla del  
computador (consola serial) el estado del pulsador  
cuando es oprimido  
*/
```



**¿Qué aprendo?**

- Manejar una entrada analógica
- Ver datos por la pantalla del computador
- Múltiples estados de un potenciómetro
- Leer una entrada analógica

**Conocimientos previos**

- Señal analógica
- Función `analogRead()` y `Serial.println()`
- Opción de Consola serial, ver **6F** (paso 3)

**Materiales**

1



Arduino UNO

1



1



Protoboard

1



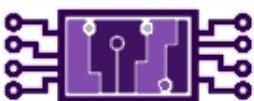
Cable USB Tipo AB

Potenciómetro 10K

3

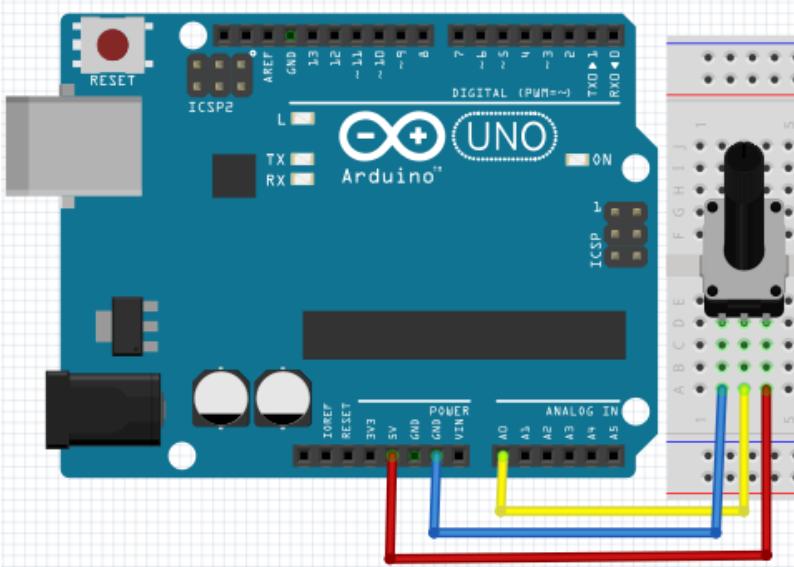
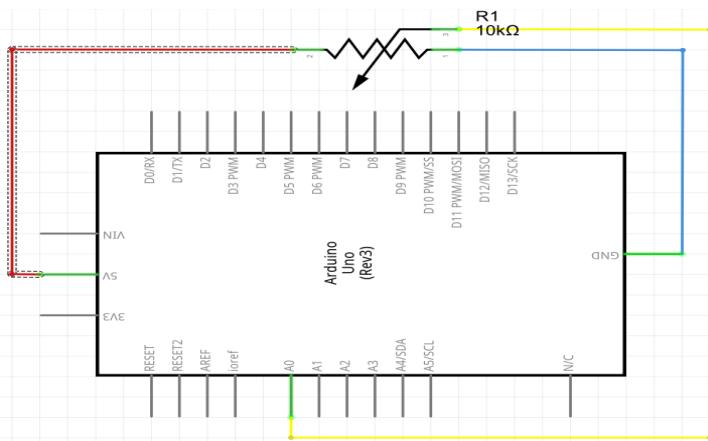


Conectores MM

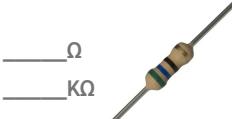


MazMaker

**Genuino**  
**DAY 2016**



1— ¿Cuál es el valor de esta resistencia?

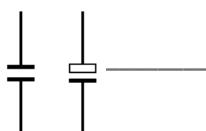


2— ¿Qué hace esta función?

**analogRead()**



3— Este símbolo a que corresponde



```
/*
-----  
Lectura serial de entrada analoga  
-----  
  
Leer una entrada analoga y mostrar por la pantalla del  
computador (consola serial) el valor luego de girar  
el potenciómetro  
  
*/  
  
//-----  
//Funcion principal  
//-----  
void setup() // Se ejecuta cada vez que el Arduino se inicia  
{  
    Serial.begin(9600); //Inicia comunicación serial  
}  
  
//-----  
//Funcion ciclica  
//-----  
void loop() // Esta funcion se mantiene ejecutando  
{  
    // cuando este energizado el Arduino  
  
    //Guardar en una variable entera el valor del potenciómetro 0 a 1024  
    int valor= analogRead(A0);  
  
    //Imprime en la consola serial el valor de la variable  
    Serial.println(valor);  
  
    //Retardo para la visualización de datos en la consola  
    delay(100);  
}  
  
//Fin programa
```

## 1- Te invitamos a que conozcas algunos tipos de potenciómetros



SoftPot  
Sistema touch



Trimmer  
Alta precisión



Encoder RGB  
Giro continuo

## 2- Para recordar



- Para leer una señal analoga usa: `analogRead(numeroPin);`
- Para escribir una señal analoga de PWM usa: `analogWrite(numeroPin, valor);`
- Una entrada analoga va de **0** o **1023**
- Una salida analoga de PWM va de **0** o **255**

C  
Ó  
D  
I  
G  
O

/\*

-----  
Lectura serial de entrada analoga  
-----

Leer una entrada analoga y mostrar por la pantalla del computador (consola serial) el valor luego de girar el potenciómetro



COM7

Valor0

Valor7

Valor59

Valor59

Valor300

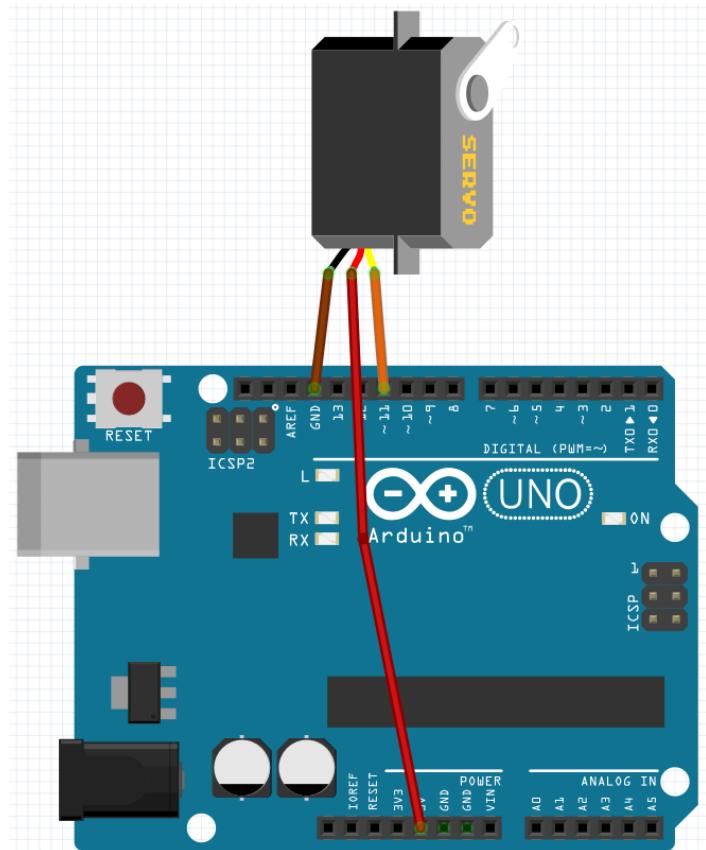
Valor597

Valor1023



MazMaker

Genuino  
DAY 2016



Nota:

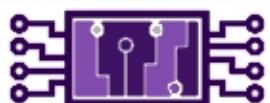
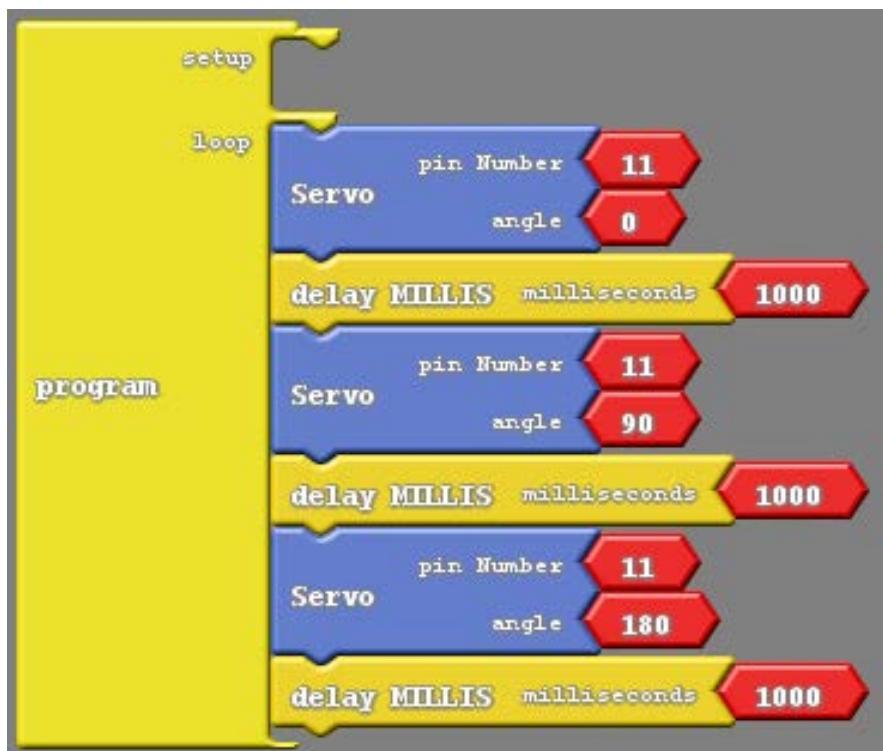
Cafe = GND

Rojo= 5V

Anaranjado = pin# PWM



Genuino  
DAY 2016



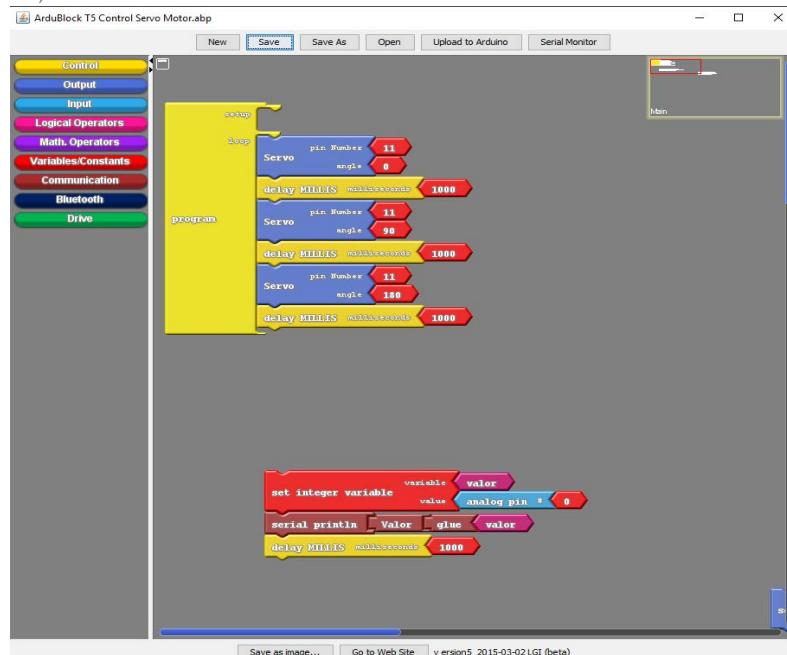
MazMaker

Genuino  
DAY 2016

# Introducción a Ardublock

## ¿Qué es Ardublock?

Es una herramienta de programación visual por bloques para Arduino, funciona como un rompecabezas de bloques funcionales de distintos colores y genera de forma fácil y sencilla el código en el entorno de programación de Arduino (Arduino IDE).



Ardublock es una herramienta que acerca el mundo de la programación de Arduino a estudiantes y aficionados de una forma fácil y sencilla.



Hola, queremos conocer tus opiniones referente a este material, son de gran ayuda con el ánimo de seguir mejorando los contenidos y haciendo éstos más claros.

Estamos atentos a recibir todo tipo de comentarios que nos sirvan de retroalimentación y nos fortalezcan más.

Escríbenos a:

[gustavo.reynaga@gmail.com](mailto:gustavo.reynaga@gmail.com)

Muchas gracias por tus aportes :D



Este manual fue adaptado de la obra "Guía Básica de Arduino", elaborado por:

La Tienda de Robótica y el Equipo de Cosas de Mecatrónica.

<http://www.tiendaderobotica.com/>

<http://www.cosasdemelectronica.com/>





MazMaker

Genuino  
DAY 2016

# ¡GRACIAS!

## PATROCINADORES

Noroeste



Tecnika®