

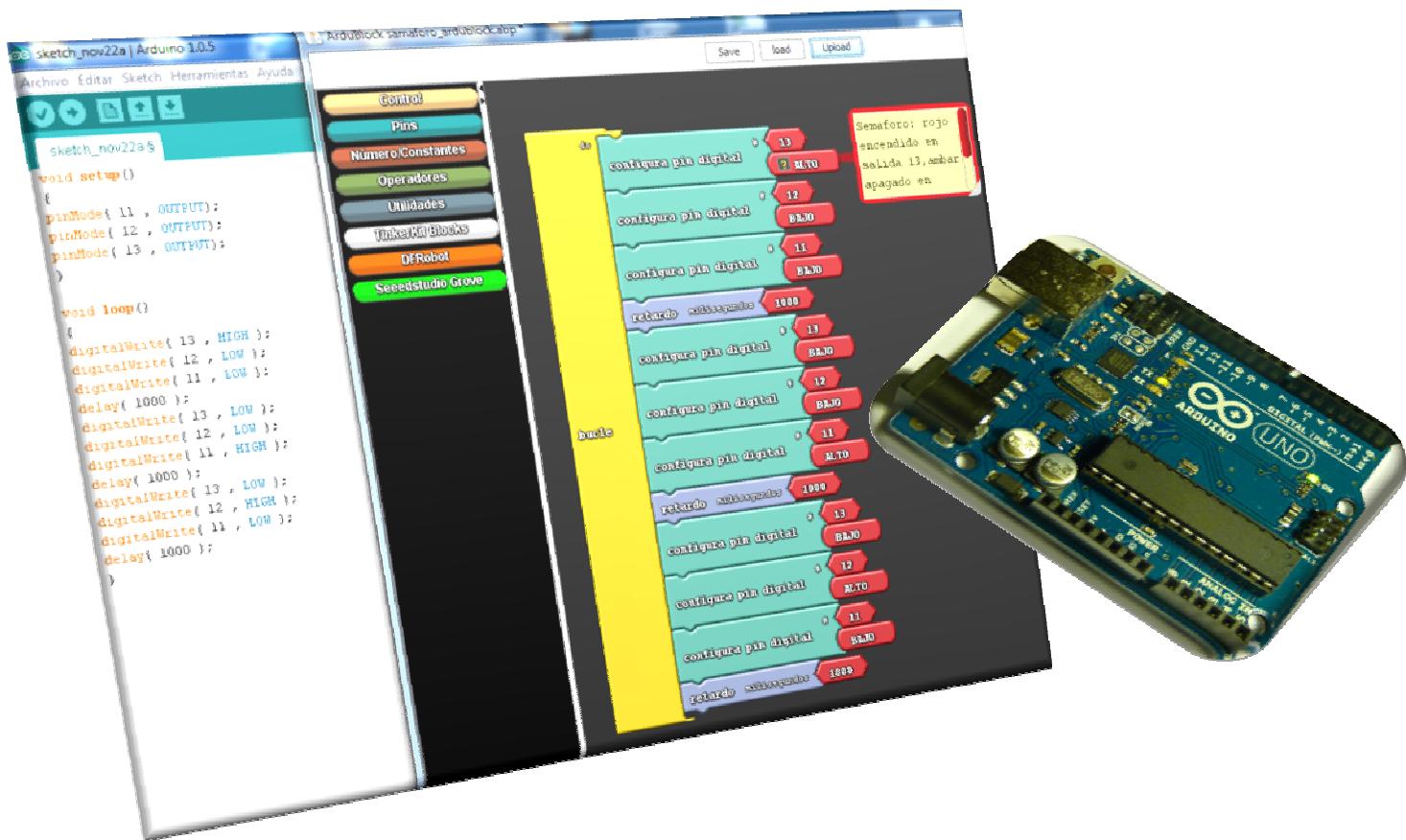


GOBIERNO
de
CANTABRIA

CONSEJERÍA DE EDUCACIÓN,
CULTURA Y DEPORTE



ARDUBLOCK



MANUAL BÁSICO DE ARDUBLOCK

Para programar Arduino sin saber Arduino

Por Ana Núñez Pérez





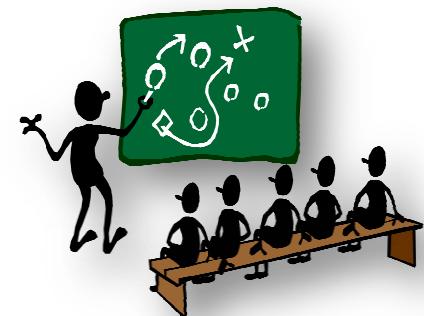
INDICE

1. PRIMERAS NOCIONES	3
2. EMPEZAMOS A TRABAJAR	6
3. POR DÓNDE EMPEZAMOS	10
4. NUESTRA PRIMERA PRÁCTICA: LED INTERMITENTE	14
5. FUNCIÓN REPITE	16
6. PRACTICA: SOS	17
7. PRACTICA: EL SEMAFORO	18
8. PRACTICA: GOBIERNO DEL ENCENDIDO DE UN LED (SALIDA) MEDIANTE UN PULSADOR (ENTRADA DIGITAL)	19
9. PRÁCTICA: LED (SALIDA) INTERMITENTE CON CONTROL DE FRECUENCIA MEDIANTE UN POTENCIOMÉTRO (ENTRADA ANALÓGICA)	22
10. PRÁCTICA: CONECTAR UNA SALIDA DIGITAL CON UN PULSADOR (ENTRADA DIGITAL) Y APAGARLA CON OTRA ENTRADA DIGITAL DIFERENTE. FUNCIÓN WHILE (MIENTRAS QUE)	24
11. PRÁCTICA: CONTROL DE UN MOTOR	27
12. PRÁCTICA 8: CONTROL DE UN SERVOMOTOR (giro 180º)	29
13. PRÁCTICA: SERVO CONTRALADO POR SEÑALES EXTERNAS	35
14. PRÁCTICA: SENSOR DE LUZ 1(LDR)	37
15. PRÁCTICA: SENSOR DE LUZ 2 (EL FOTOTRANSITOR).	40
16. PRÁCTICA: SENSOR IR, SENSOR DE LUZ INFRARROJA.	41
17. PRÁCTICA: ULTRASONIDOS SR04	43
18. PRÁCTICA: ROBOT MINISKYBOT (ARDUINO)	50
19. PRÁCTICA: CÓDIGO ESCOBA (ARDUINO)	54
20. PRÁCTICA: SIGUELINEAS CON ARDUBLOCK	56
21. DOCUMENTACIÓN – BIBLIOGRAFÍA – WEBGRAFÍA	57

1. PRIMERAS NOCIONES

⌚ ARDUBLOCK

¿Qué es Ardublock?: Ardublock **es un lenguaje gráfico de programación** que se distribuye como un plugin de java y que se añade a las herramientas del IDE de Arduino, con lo que permite la elaboración de programas para Arduino sin necesidad de escribir el código con la sintaxis de sus órdenes.



Se basa en el uso de una sencilla interfaz gráfica utilizando un sistema de bloques, que simbolizan diferentes elementos de programación, por ejemplo: instrucciones, condiciones, variables, bucles, etc. Estos bloques de programación se van ensamblando, como las piezas de un puzzle hasta formar programas, lo que hace que esta herramienta sea de gran utilidad para iniciar a nuestros alumnos en la programación con Arduino ya que tiene la posibilidad de pasar “nuestro programa-puzzle” a lenguaje Arduino permitiendo ver a nuestros alumnos a qué sentencias equivalen cada pieza del puzzle. . Además se trata de una aplicación libre y es muy fácil de instalar y manejar.

⌚ LA PLACA ARDUINO UNO

Arduino UNO es una tarjeta electrónica sencilla y de bajo precio diseñada para el aprendizaje de electrónica y la creación de prototipos.

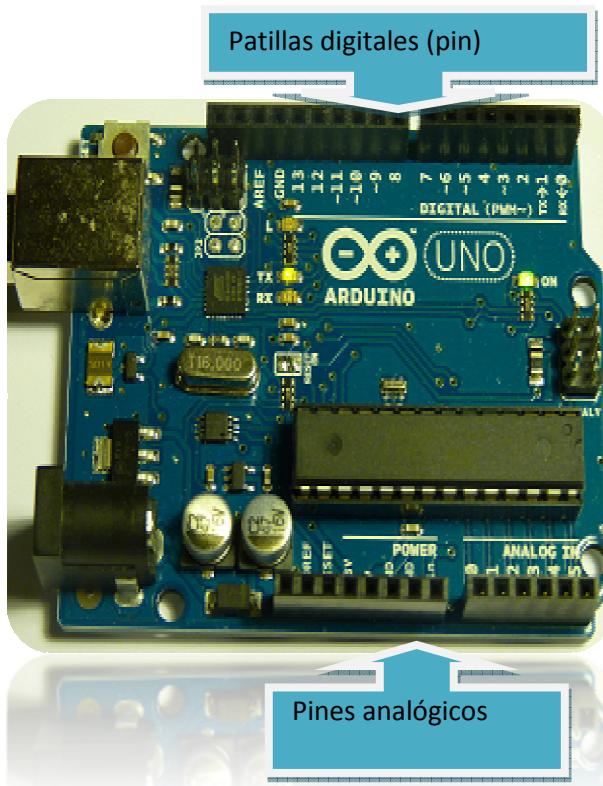
Qué nos vamos a encontrar en esta tarjeta:

- **Pines digitales:** la tarjeta Arduino UNO tiene 14 patillas digitales que pueden actuar tanto como entradas de datos (E) como de salidas (S), dependiendo de lo que nosotros queramos conectar en ellas, elementos de control, o actuadores. Por ejemplo, podríamos conectar un pulsador (elemento de control) en la patilla 7 (pin 7), con lo que la estaríamos usando como entrada y un LED (actuador) en el pin 13, con lo que la patilla 13 la estaríamos utilizando como salida. Nota: en el Pin 13 hay un LED y una resistencia asociada soldados a la placa, cuando el pin es de alto valor, el LED está encendido, cuando el valor está bajo, es apagado.
- **Entradas analógicas:** En estos pines podemos conectar distintos sensores como sensores de luz, temperatura, etc. Esta placa tiene 6.
- **Memoria Flash:** En ella se grabará nuestro programa, la de nuestra placa es de 32Kb.



➤ **Memoria RAM:** dispone de una RAM de 2 Kb

➤ **Memoria auxiliar:** de un 1Kb



Esta es la placa con la que trabajaremos, y las patillas digitales pueden verse en la parte superior (P0,1,2,3,4,5,6,7,8,9,10,11,12,13) y en la inferior las analógicas, numeradas como A0, A1,.... También en la parte inferior se encuentra el botón Reset, que permite reiniciar el bootloader y cargar un nuevo programa, también se puede hacer desde el IDE de Arduino.

En la parte derecha podemos ver el puerto USB, a la izquierda la entrada de corriente. Seis de las entradas digitales se pueden emplear como entradas PWM. Esto quiere decir que permiten simular un comportamiento analógico enviando señales de valor variable, de cara a modificar la intensidad de iluminación de un LED o la velocidad de giro de un motor, se corresponden a los pines 3, 5, 6, 9, 10 y 11, y permiten enviar señales de 256 valores distintos entre 0 y 5 voltios.

La alimentación de Arduino UNO se realiza a través de la entrada USB, o bien a través de la entrada de corriente SHIELDS DE ARDUINO



Los shields son placas que se empotran encima de una placa Arduino y que ofrecen funcionalidades complementarias a las de esta evitándonos tener que montar circuitos para probar nuestros programas, una de ellas es la “*Arduino Basic I/O*”

⌚ ARDUINO BASIC I/O

Para poder realizar montajes de electrónica de control con Arduino deberíamos trabajar con nuestros propios circuitos eléctricos y electrónicos empleando placas Proto Board, pero para simplificarnos algunas prácticas también podemos usar una placa hecha exprofeso para entornos docentes por la empresa vasca MSE Bilbao, la placa *BASIC I/O*, que podemos ver en la imagen.

Las entradas digitales se identifican con una D y un número, y ya van asociadas a LEDs pines 11, 10, 9 y 6, tiene también 4 pulsadores (pines 12, 8, 7 y 4), un altavoz (pin 2), conectores para motores (pines M+ y M-), que corresponden a dos tornillos de la parte superior derecha, y conexiones para servos (SV1 y SV2).

Las entradas analógicas se identifican con una A y entre ellas tenemos: dos potenciómetros (pines 0 y 1, situados en la esquina inferior izquierda de la imagen), a su derecha, de color blanco, un fotodiodo (pin 2), siguiendo hacia la derecha de la imagen, un sensor IR (pin 3), a su derecha, un sensor de temperatura (pin 4) y en la esquina inferior derecha de la imagen tenemos un transistor. Por último arriba a la izquierda se encuentra el botón de RESET.



Con esta placa ya tenemos creados los circuitos eléctricos y electrónicos necesarios para poder empezar a programar en Arduino.



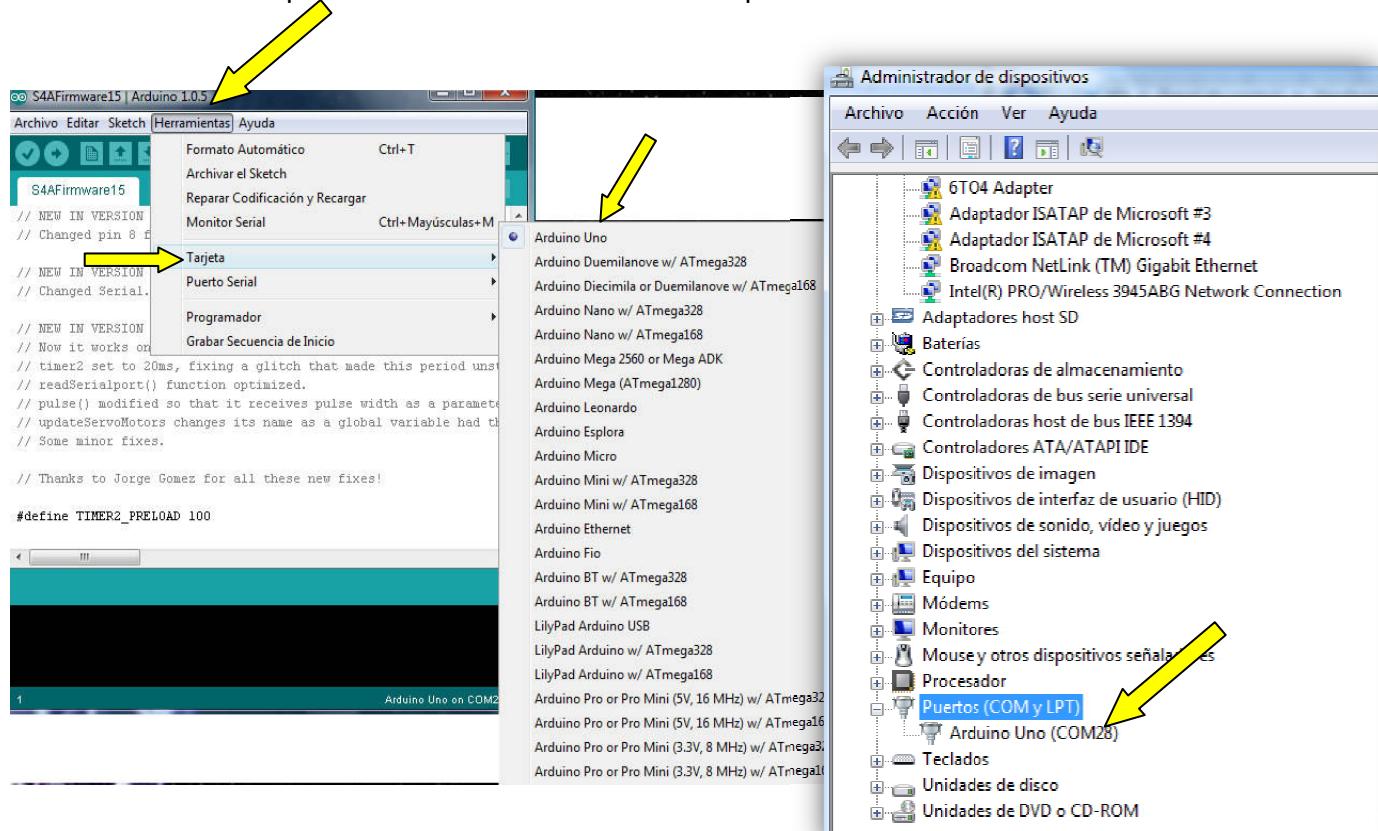
2. EMPEZAMOS A TRABAJAR

Para empezar a trabajar lo primero que tenemos que hacer es instalar el software necesario y para ello deberás seguir los pasos siguientes:

- **1º paso:** Descargad e instalad el entorno Arduino siguiendo las instrucciones de <http://arduino.cc/en/Main/Software>

- **2º paso (Seleccionar la placa Arduino que vamos a utilizar):** Conectad la placa Arduino a un puerto USB de vuestro ordenador, entonces se instalará el nuevo hardware y fígaros en qué puerto está conectada. Siempre podréis comprobarlo en el administrador de dispositivos-puertos COM y LPT.

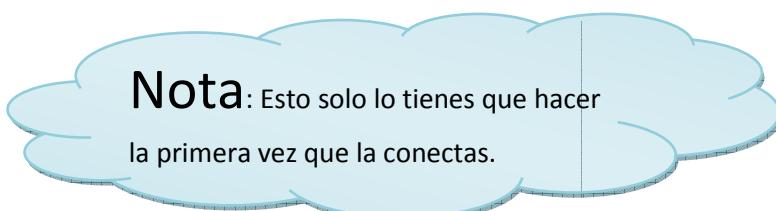
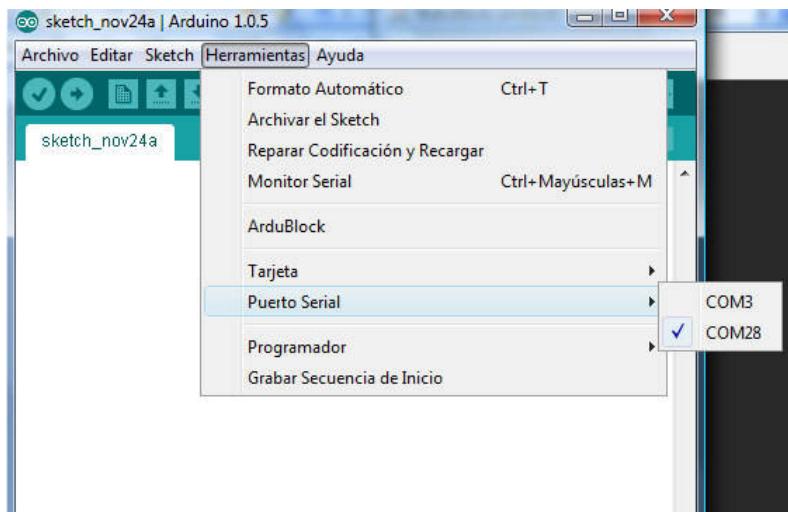
En mi caso la placa Arduino se ha instalado en el puerto COM 28



A continuación en el menú **Herramientas - Tarjeta**, seleccionamos la versión de la placa que vamos a utilizar (en mi caso veréis que se trata de la placa Arduino Uno).

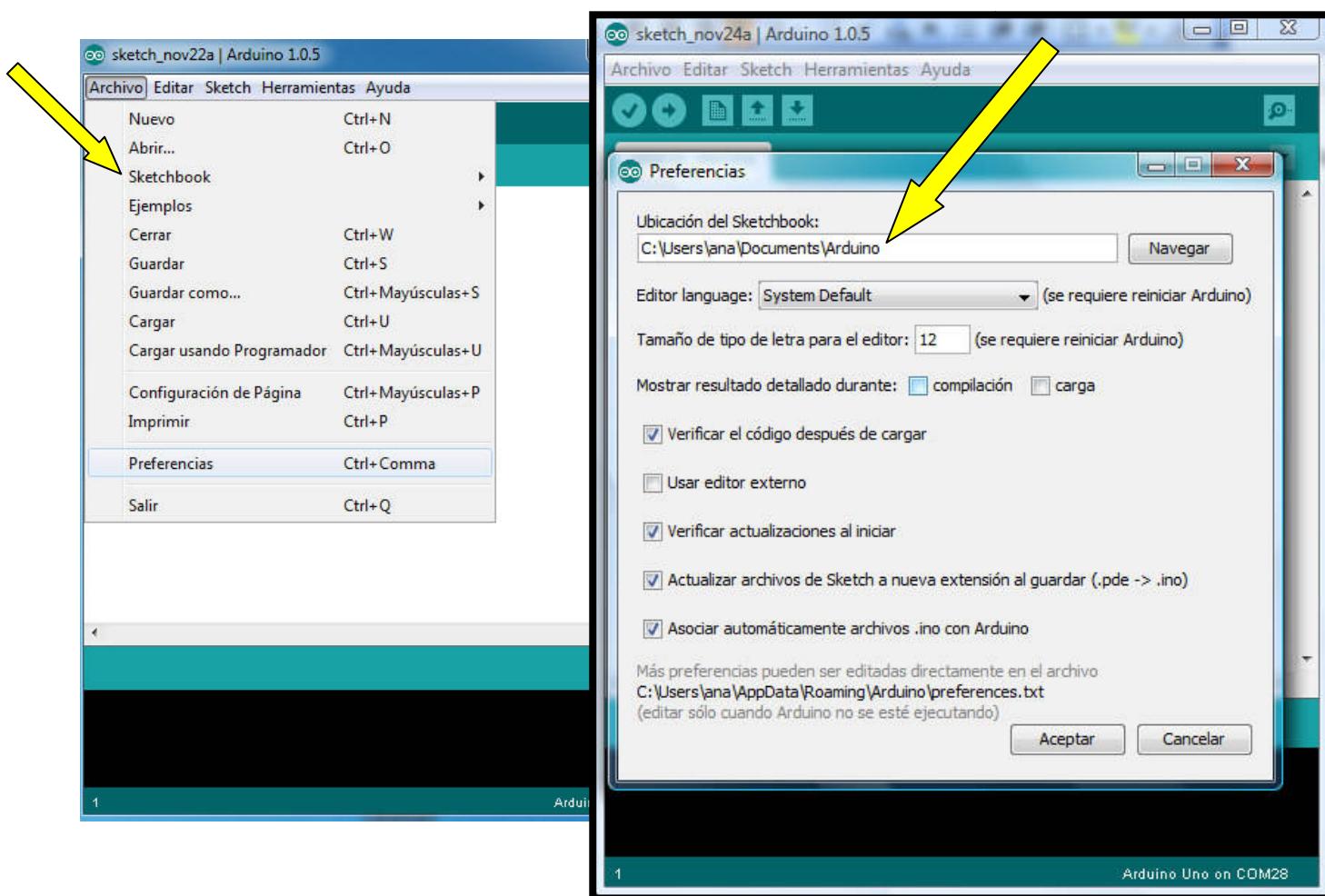


Ahora debemos seleccionar el canal el canal de comunicaciones que va a emplear nuestro ordenador para comunicarse con la tarjeta. Para ello en el menú **Herramientas – Puerto Serial**, seleccionad el puerto serie en qué esté conectada la placa.



- ☞ **3º paso:** Para instalar Ardublock tenemos que ir primero a la página de referencia: <http://blog.ardublock.com>. Luego allí ir donde pone download y descargar el archivo .jar, o ir directamente al enlace [descarga](http://sourceforge.net/projects/ardublock/files/) (<http://sourceforge.net/projects/ardublock/files/>). Si prefieres otra opción puedes es descargartelo directamente en el fichero *ardublock-all.jar* desde la página ArduBlock, <http://cloud.github.com/downloads/taweili/ardublock/ardublock-all.jar>

- ☞ **4º paso:** En carpeta en donde esté instalado el IDE Arduino debemos incluir el fichero *ardublock-all.jar*. Para asegurarnos dónde lo tenemos instalado hacemos lo siguiente: Abrimos Arduino y verificamos la ruta del Sketchbook, eso lo hacemos el menú de Arduino - “Archivo” -> “Preferencias” clicamos y nos aparecerá la ventana con la ubicación del **Sketchbook**. En mi caso: C:\Users\ana\Documents\Arduino .



Ahora que ya sabemos donde está instalado, cerramos el IDE de Arduino y lo abriremos de nuevo cuando tengamos el archivo .jar copiado.

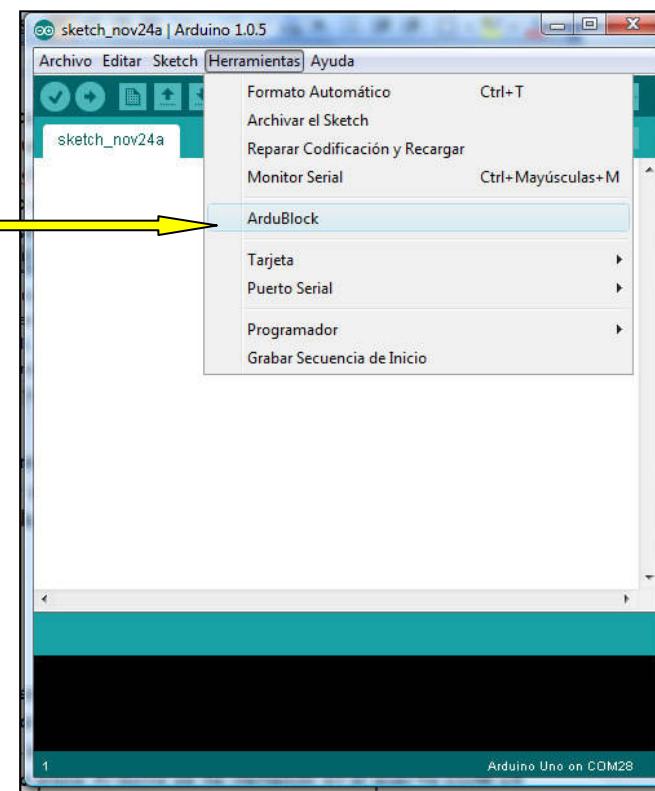
Nos vamos a la carpeta de Arduino (a la dirección que acabamos de verificar) y en ella creamos una carpeta “tools”, luego dentro de ésta creamos otra a la que llamamos “ArduBlockTool”, y dentro de ésta, otra carpeta que se llame “tool”. Finalmente, dentro de “tool” copiamos el archivo .jar que hemos descargado.

En mi caso:

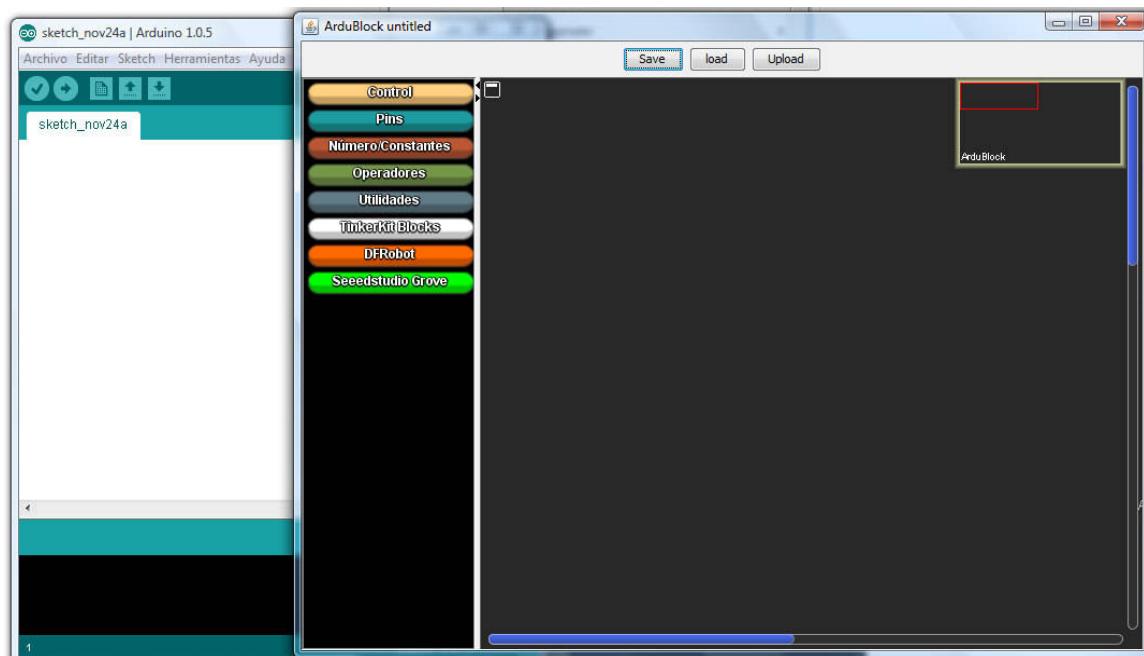




Arrancamos de nuevo el IDE de Arduino y vamos al menú de Herramientas, donde ahora debe aparecer Ardublock.

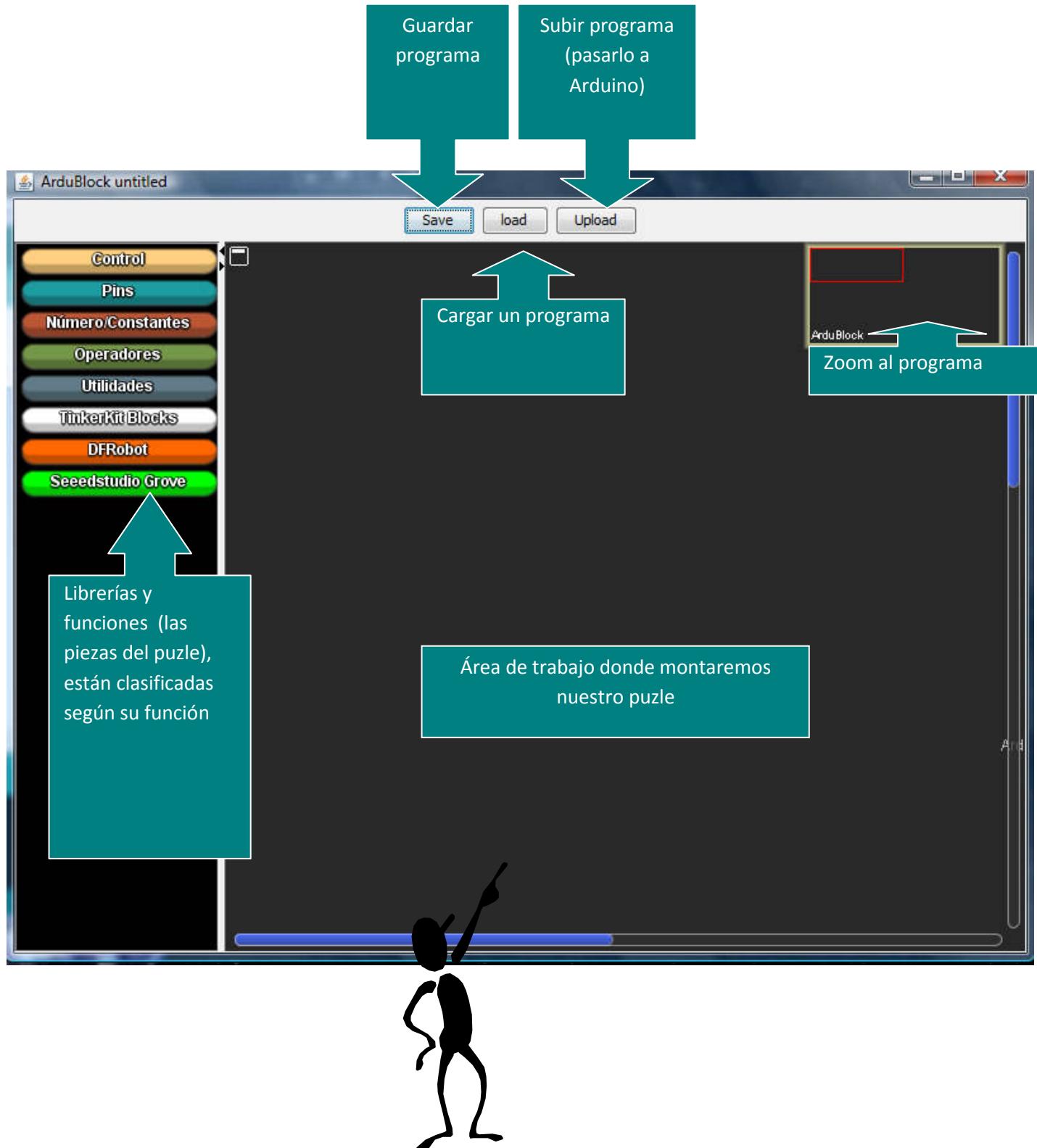


Hacemos clic, se nos abrirá una nueva pantalla de trabajo, la de Ardublock como la de Arduino



3. POR DÓNDE EMPEZAMOS

Ya con nuestra pantalla de Ardublock abierta vamos a ver con que nos encontramos.



Ahora vamos a analizar un poco las librerías de Ardublock. Como verás están clasificadas por colores según su funcionalidad, son muy intuitivas y sencillas de manejar. Como ya hemos dicho van encajándose unas en otras como las piezas de un puzzle, si colocamos una función o pieza con otra cuya forma no encaja al igual que ocurre en un puzzle la unión no quedará bien realizada, sin embargo si la función si se corresponde con el tipo de piezas que deben encajar entre sí se oirá un “clac” al unirlas, esto es muy útil para ir aprendiendo que en programación hay ciertas reglas que se deben cumplir.

Control



Funciones de Control, estas piezas permiten realizar:

- Bucle: ejecuta la acción de forma indefinida.
- Si/entonces: ejecuta otras instrucciones de manera condicional (está tiene tres enganches) en condición, colocamos la instrucción que establece la condición, en el *primer entonces* la acción que se ejecutará si la condición es **verdadera**, y en el *segundo entonces* la que la acción que se ejecutará si la condición es **falsa**.
- Mientras que: al establecer una condición se mantiene la acción encajada en la parte inferior de la ficha del puzzle, mientras la condición establecida siga siendo verdadera.
- Repite: En la parte superior aparecerá el número de veces que se repite la acción colocada en hueco inferior de la pieza.



Funciones de Pines, con ellas se lleva a cabo la gestión de las entradas y salidas tanto analógicas como digitales.

Por ejemplo:

- Permite leer el estado de una entrada digital, al colocarla en el área de trabajo nos aparecerá asociada al un número 1 al clicar sobre el número escogeremos el número del pin que queramos leer. Lo mismo para pin analógico.

Con esta función podemos darle el valor “*alto*” o “*bajo*” a la entrada o salida elegida.

- con esta función gestionaremos los valores de los servos como el ángulo inicial.

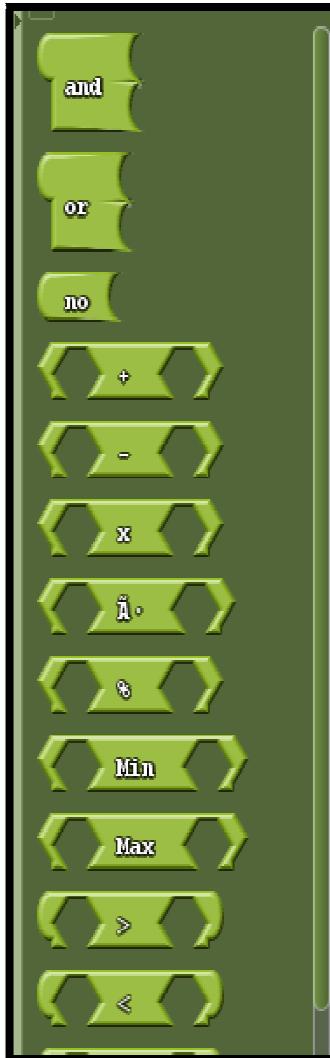
Número/Constantes

Aquí encontramos fichas muy utilizadas en los proyectos para temporizar como por ejemplo la función retardo, que detiene el progreso del programa los milisegundos que indiquemos.



Operadores

Estos operadores permiten realizar operaciones lógicas y aritméticas, como por ejemplo las funciones lógicas AND, NOT, OR...



Utilidades

Con ellas podemos dar valores para operar con ellos (1,2,...) o constantes (alto, bajo, verdadero...) a las entradas y salidas. Fíjate que sus formas encajan con las funciones de pines vistas anteriormente.



TinkerKit Blocks

DFRobot

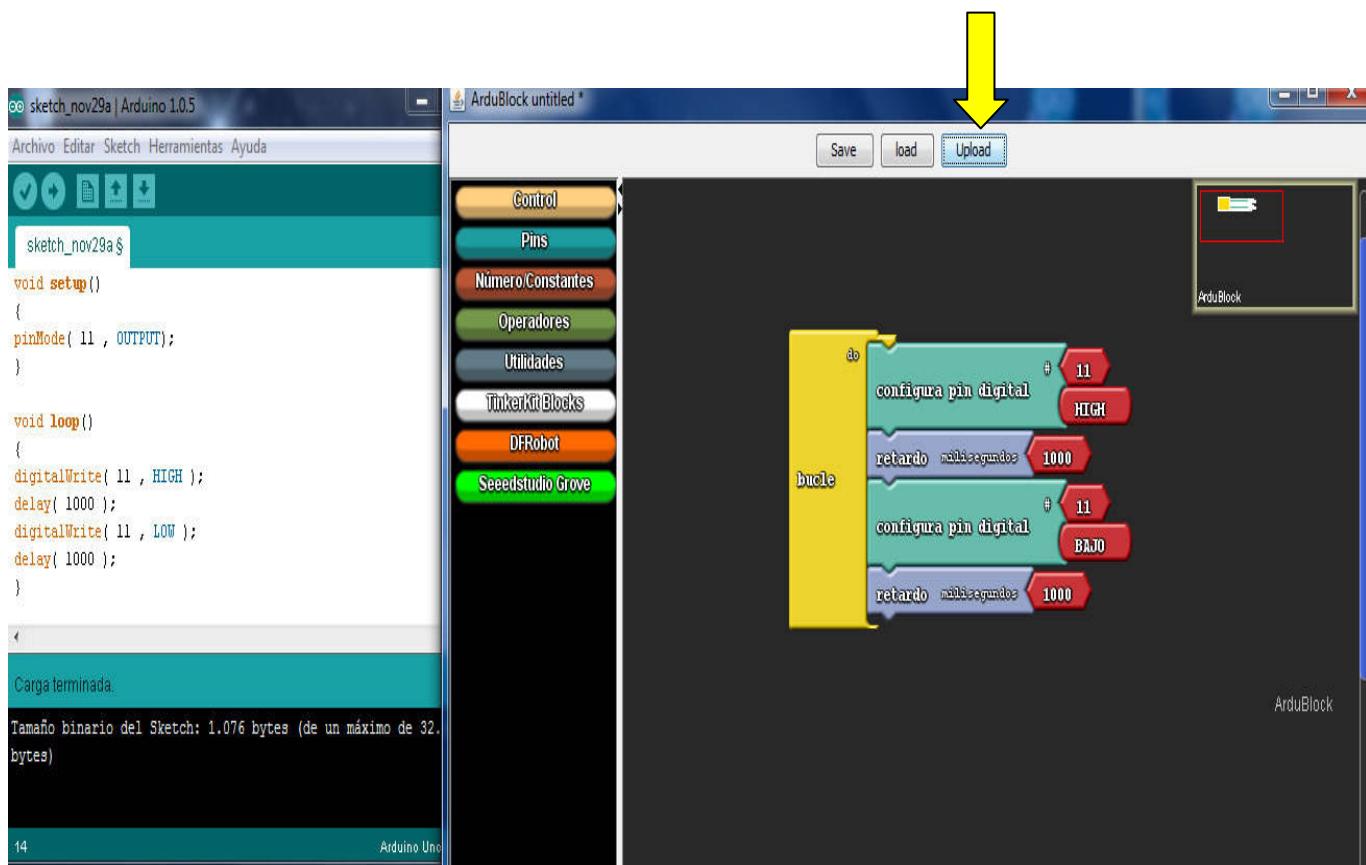
Seeedstudio Grove

Están dedicados al control de periféricos concretos, como LED, Relés, Sensores analógicos de sonido, etc.

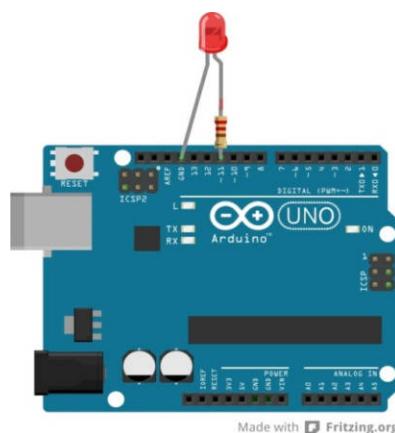


4. NUESTRA PRIMERA PRÁCTICA: LED INTERMITENTE

Nuestro primer ejercicio será el apagado y encendido del led rojo de la tarjeta de experimentación ARDUINO BASIC I/O, este led se corresponde con la salida D11. Y el programa por lo tanto quedará como se ve en la figura. Fíjate que una vez formado nuestro puzzle daremos a la tecla “**Upload**” y automáticamente nos aparecerá en la pantalla de Arduino el mensaje “*compilando el sketch*” y en unos segundos el programa estará cargado en nuestra placa, entonces aparecerá el mensaje “*carga completada*”. Ahora ya puedes ver tu primer programa traducido a Arduino.



Si no dispones de la tarjeta de experimentación ARDUINO BASIC I/O, pues no pasa nada, basta con que realices este sencillo montaje en tu placa Arduino UNO:



Made with Fritzing.org

Si no usamos el pin digital 13 de Arduino para conectar el LED necesitaríamos una **resistencia** de unos 200 ohmios (el valor estándar más cercano es de 220 ohmios), este valor lo obtenemos con un sencillo cálculo:

- Un LED normal necesita de entre 5 y 20 mA para encenderse, por lo que necesitaremos unos 15mA para que el LED luzca bien.
- La alimentación del Arduino UNO es de 5V.
- En el LED caerán 2 V
- En la resistencia deben caer unos 3 V

Por lo tanto:

$$5 - 2 = 3 \text{ V}$$

Aplicando la ley de Ohm (Tensión = Intensidad * Resistencia):

$$V = I * R$$

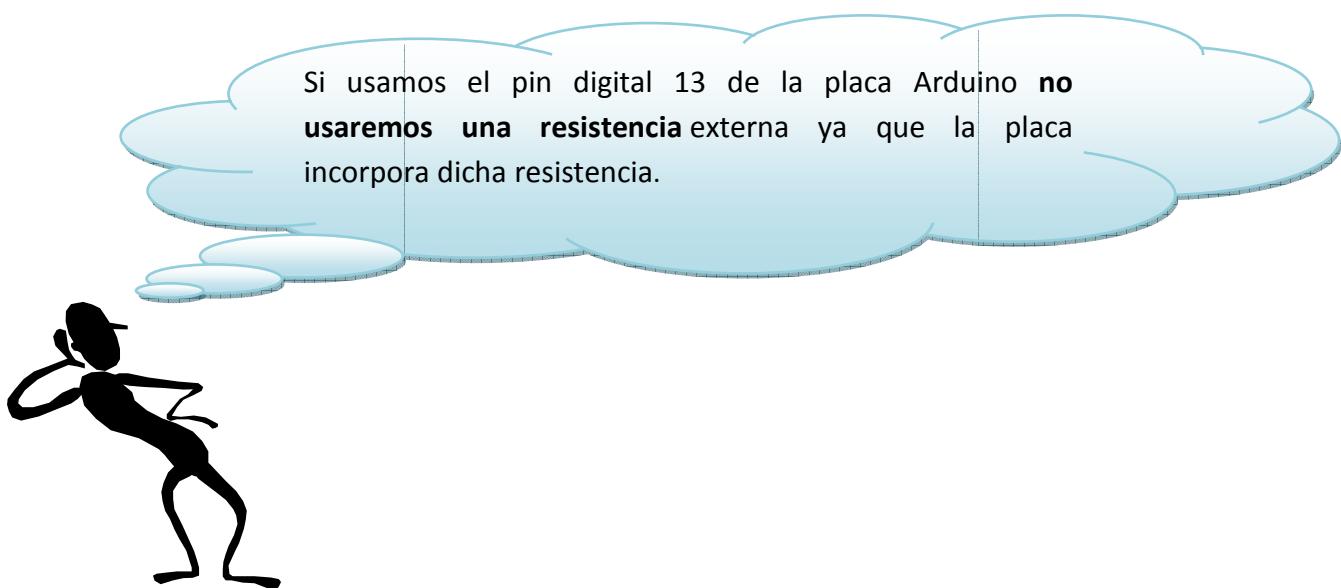
Puesto que conocemos V e I podremos calcular R:

$$R = V / I$$

Con los valores:

$$R = 3 / 0.015 = 200 \Omega$$

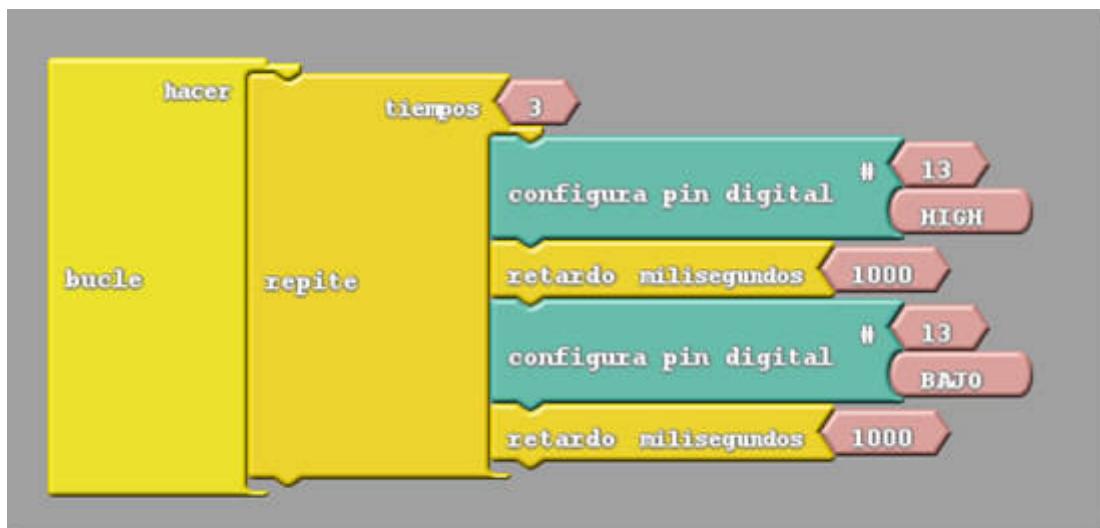
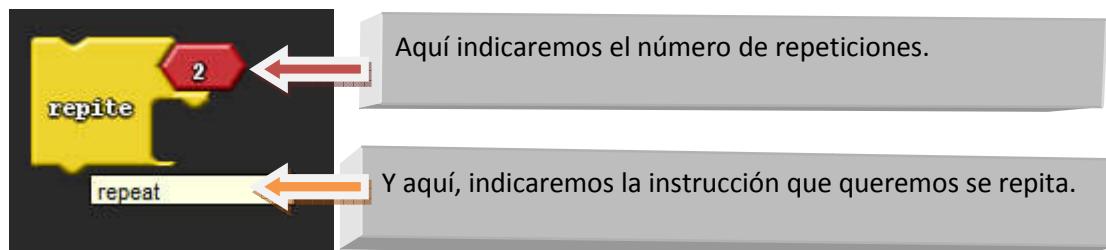
Por lo tanto necesitaremos la resistencia del mercado que más se aproxime a 200 Ω , que es una de 220 Ω .





5. FUNCTION REPITE:

Dentro la biblioteca de control una de las funciones que podemos encontrar es la función repite. Esta función nos puede ser de utilidad para ahorrarnos el repetir un número de veces determinadas órdenes en nuestros programas.



Este sería un ejemplo de cómo utilizar esta función. Pero como observarás utilizar la función repite así, como única instrucción de nuestro programa no tiene mucho sentido, ya que como los programas empiezan con un bucle, después de repetir 3 veces el apagado y encendido de lo que hayamos conectado en la salida 13, por ejemplo un LED, el programa volverá a empezar, por lo que no apreciaremos esa repetición.

A continuación vamos a realizar una práctica en la que esta función repite si nos será de utilidad.

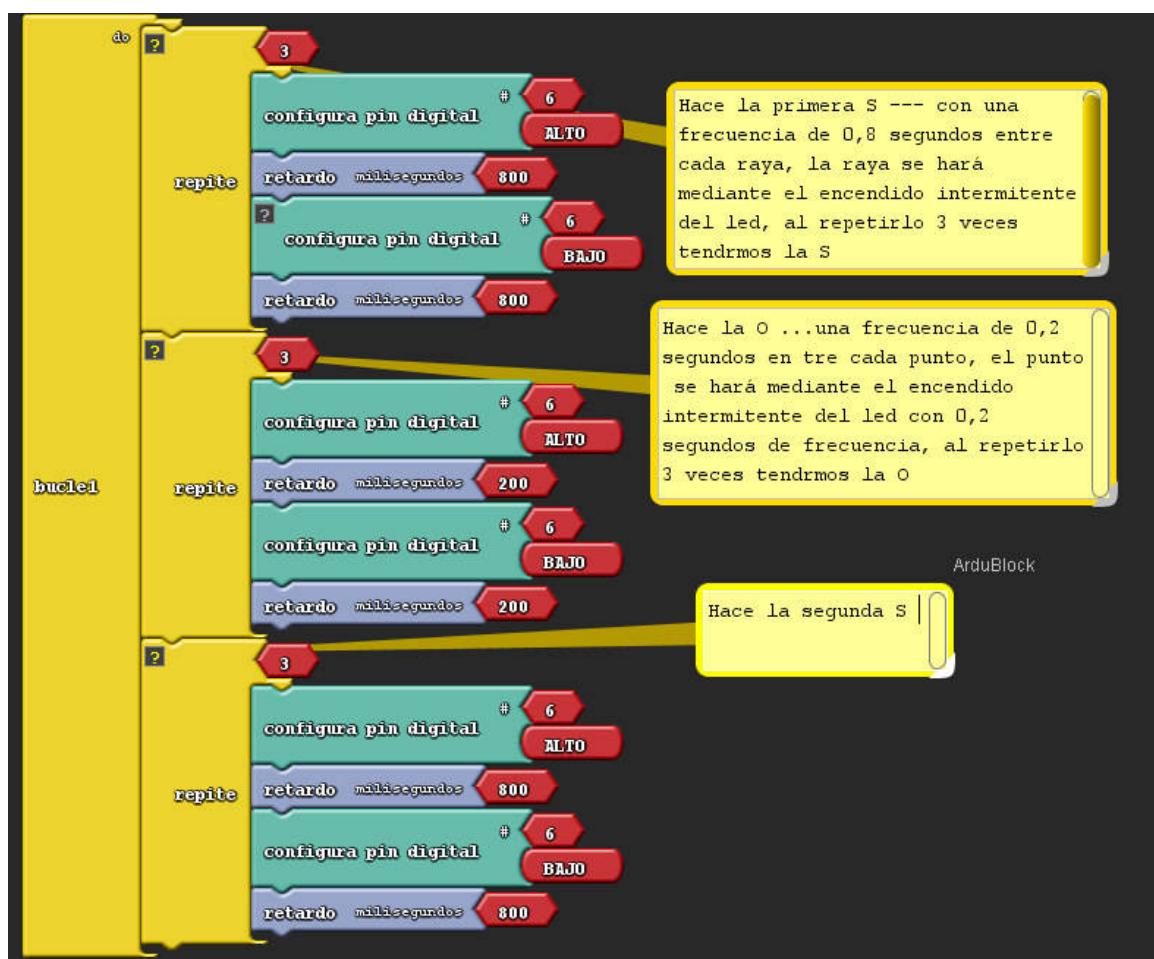


6. PRACTICA: SOS

La siguiente práctica consistirá en realizar una señal de SOS con el led blanco de la tarjeta de experimentación ARDUINO BASIC I/O que se corresponde con la salida D6. Si no, puedes utilizar el mismo montaje del ejercicio anterior y por lo tanto el **pin 11** en lugar del 6, o utilizar la salida 13, pin 13 en ella puedes colocar directamente el LED sin necesidad de usar una resistencia de protección.

Pues vamos allá, ahora te toca a ti.

Hay muchas posibilidades para realizarlo por ejemplo ésta sería una:



Recuerda si usas el montaje de la práctica anterior en la placa ARDUINO UNO, en el programa **cambia el 6** en la configuración PIN digital por un **11**.



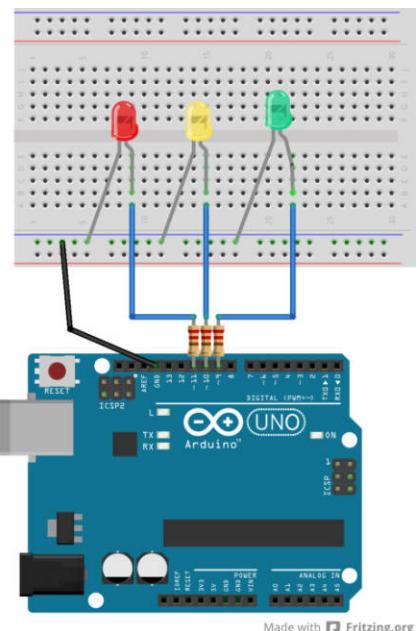


7. PRACTICA: EL SEMAFORO

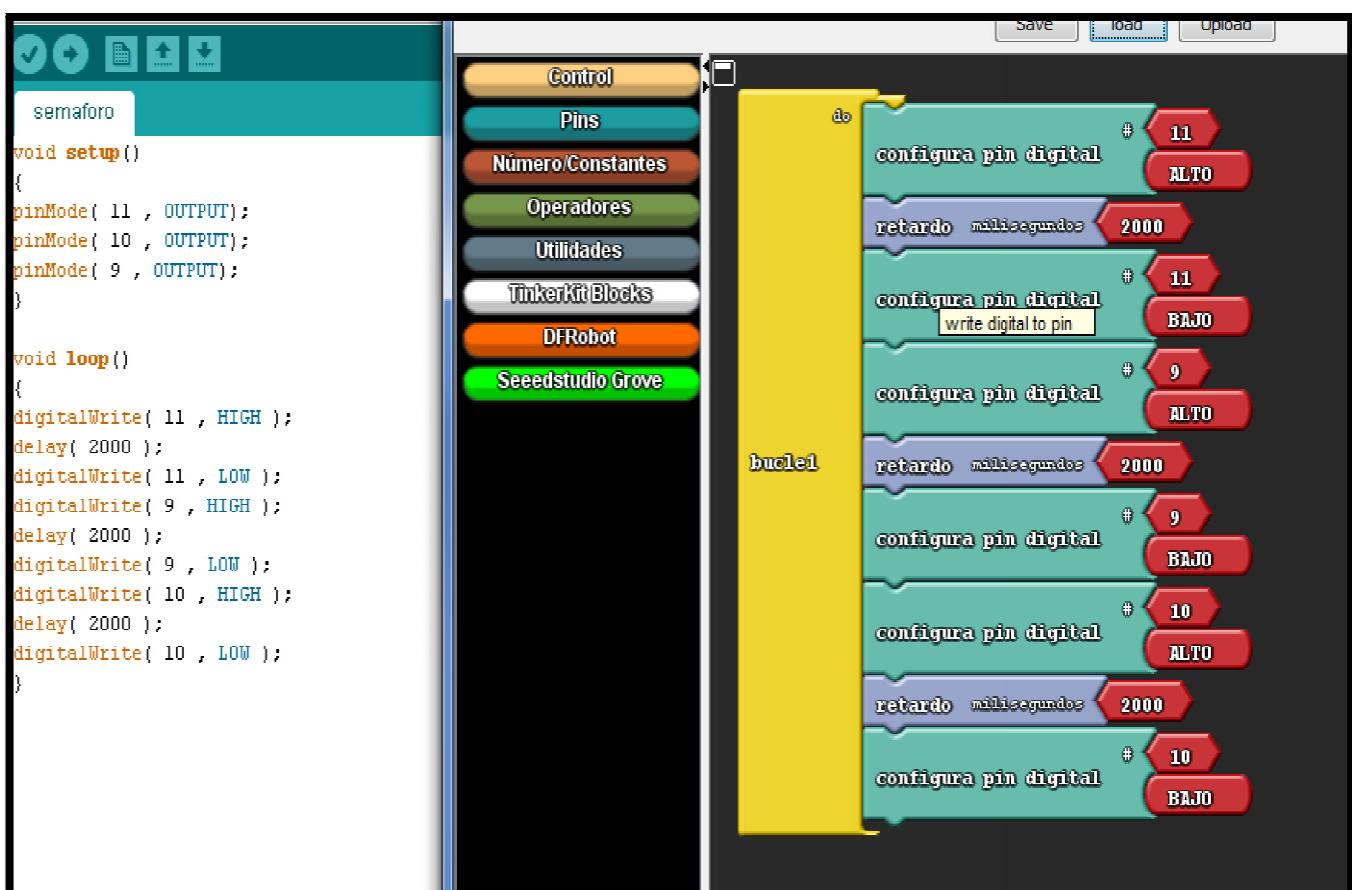
Con lo aprendido hasta ahora la siguiente práctica nos resultará muy sencilla, pues se trata de programar un semáforo. Utilizaremos las siguientes salidas:

- Pin 11: Led rojo. - Pin 10: Led ámbar - Pin 9: Led verde.

Así que siguiendo la misma lógica que aplicamos en las prácticas anteriores el programa nos quedará como observa en la imagen.



Made with Fritzing.org



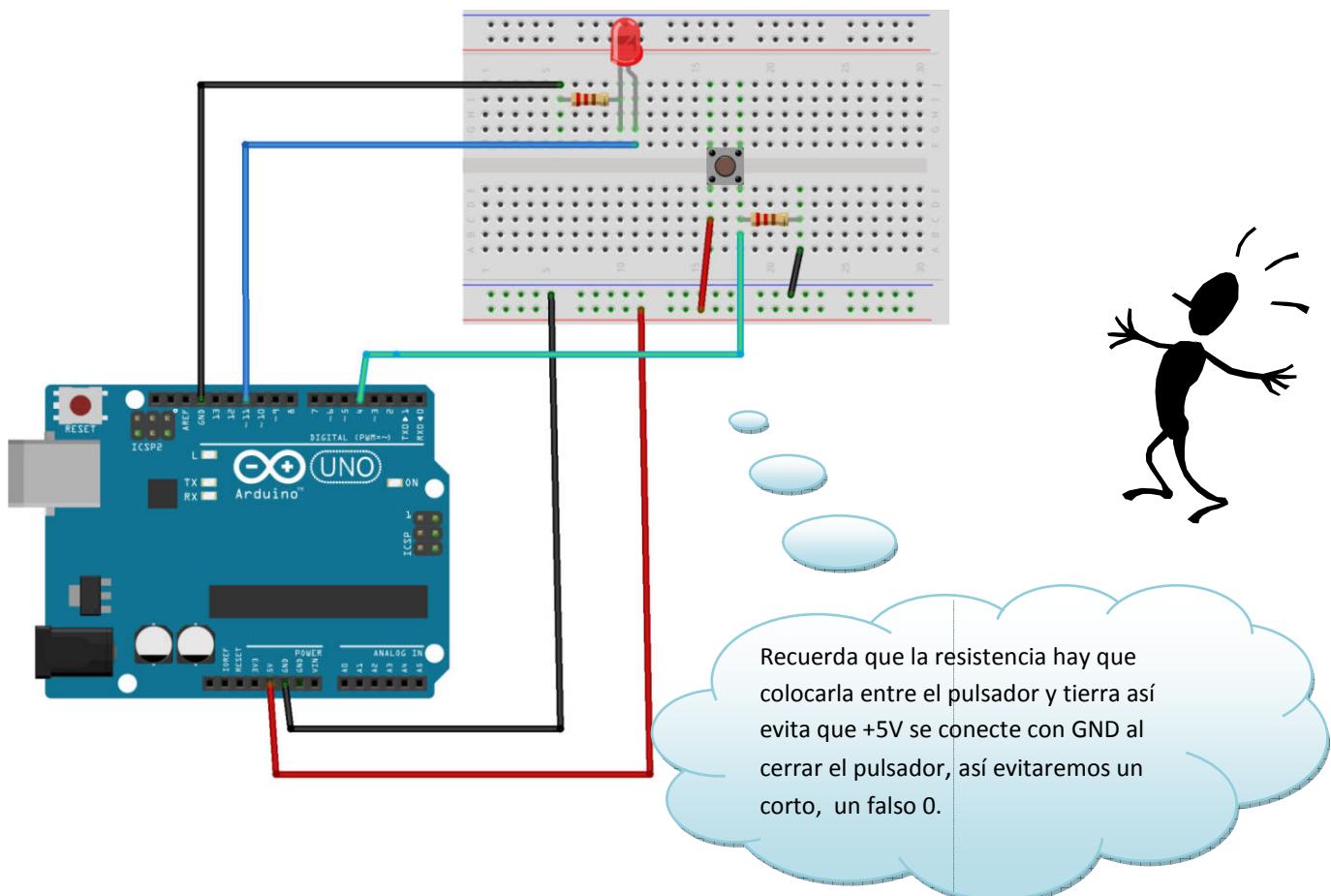
Partiendo de éste ahora puedes modificarlo tú, jugando con las frecuencias, añadiendo señal para peatones etc. Es tu turno.



8. PRACTICA: GOBIERNO DEL ENCENDIDO DE UN LED (SALIDA) MEDIANTE UN PULSADOR (ENTRADA DIGITAL)

Con esta práctica queremos gobernar el encendido y apagado de nuestro LED rojo, colocado en salida digital 11, PIN 11, mediante el accionamiento de un pulsador, conectado en PIN 4, que será nuestra entrada digital. La lógica del programa es sencilla: si el pulsador está presionado, es decir, es igual a 1 (true), la salida digital 11 (LED rojo) estará encendido; si no, salida digital 11 (LED rojo) apagado.

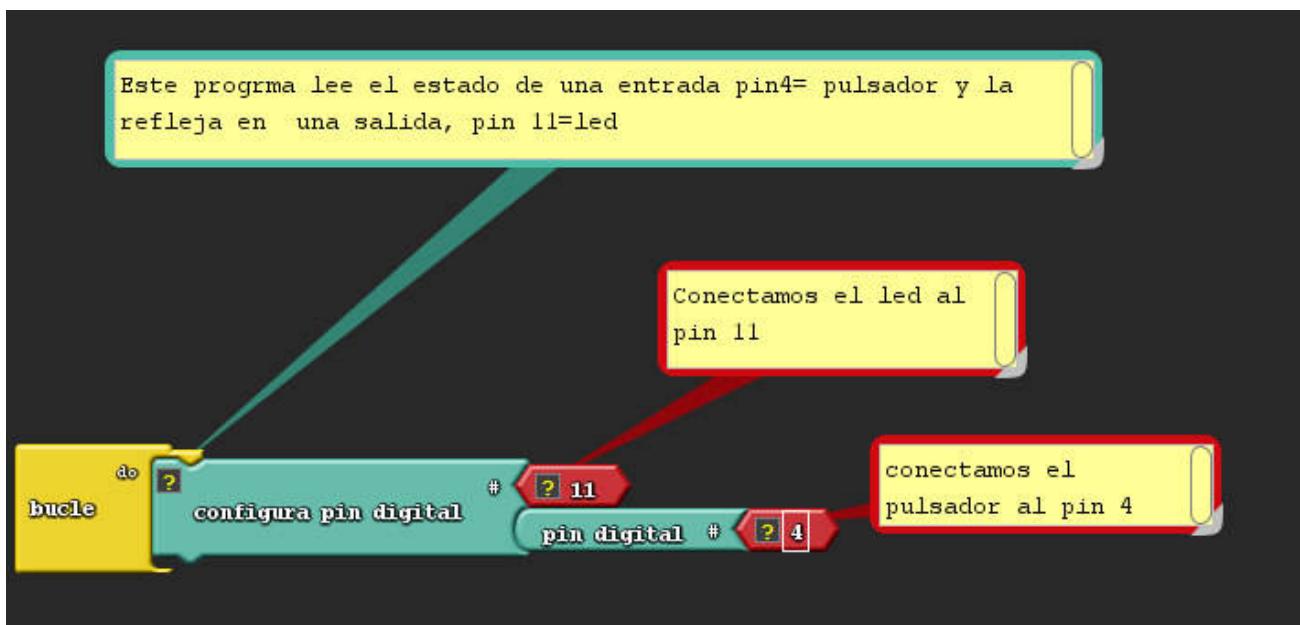
El montaje del circuito sería el siguiente:





Y el programa lo podemos hacer de muchas maneras, yo os voy a dejar alguna, pero seguro que a vosotros se os ocurrirán otras.

Ejemplo 1: Asociando el valor de una salida al estado de una entrada, lo más



sencillo.

El IDE Arduino quedará:

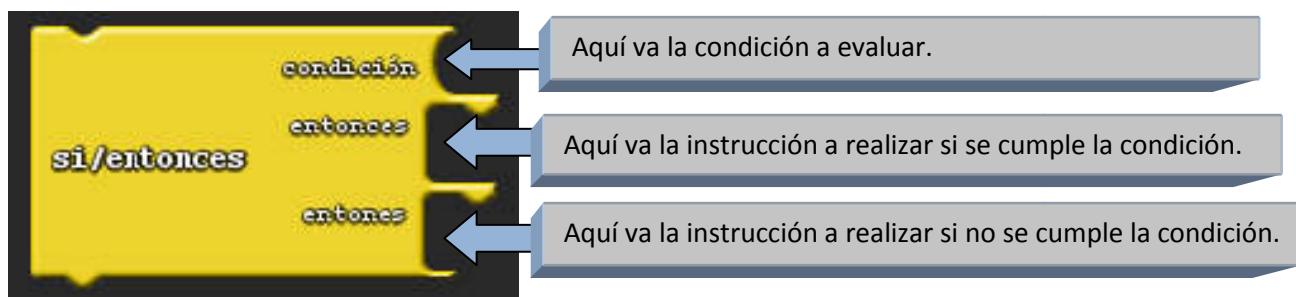
```
void setup()
{
    pinMode( 6 , OUTPUT);
    pinMode( 4 , INPUT);
}

void loop()
{
    digitalWrite( 6 , digitalRead( 4 ) );
}
```



Ejemplo 2: utilizando la FUNCIÓN CONDICIONAL SI/ ENTONCES

Esta función opera la siguiente manera: establecemos una condición a evaluar, es decir, Arduino “mirará” si eso se cumple o no, si se cumple realizará lo instrucción que colocamos primero y si no realizará la segunda orden.



Y así quedaría el programa en Ardublock.



Que se corresponde con el siguiente código:

```
void setup()
{
    pinMode( 11 , OUTPUT);
    pinMode( 4 , INPUT);
}

void loop()
```

```

if (digitalRead( 4))

{

    digitalWrite( 11 , HIGH );

}

else

{

    digitalWrite( 11 , LOW );

}

}

```

9. PRÁCTICA: LED (SALIDA) INTERMITENTE CON CONTROL DE FRECUENCIA MEDIANTE UN POTENCIOMÉTRO (ENTRADA ANALÓGICA)

En esta práctica vamos a ver como se asocia una variable numérica al valor de una entrada analógica, en nuestro caso un potenciómetro (Analog1) con el fin de poder variar el retardo en el encendido y apagado de una salida digital (led amarillo colocado en el PIN 10) al variar una entrada analógica, es decir al girar un potenciómetro.

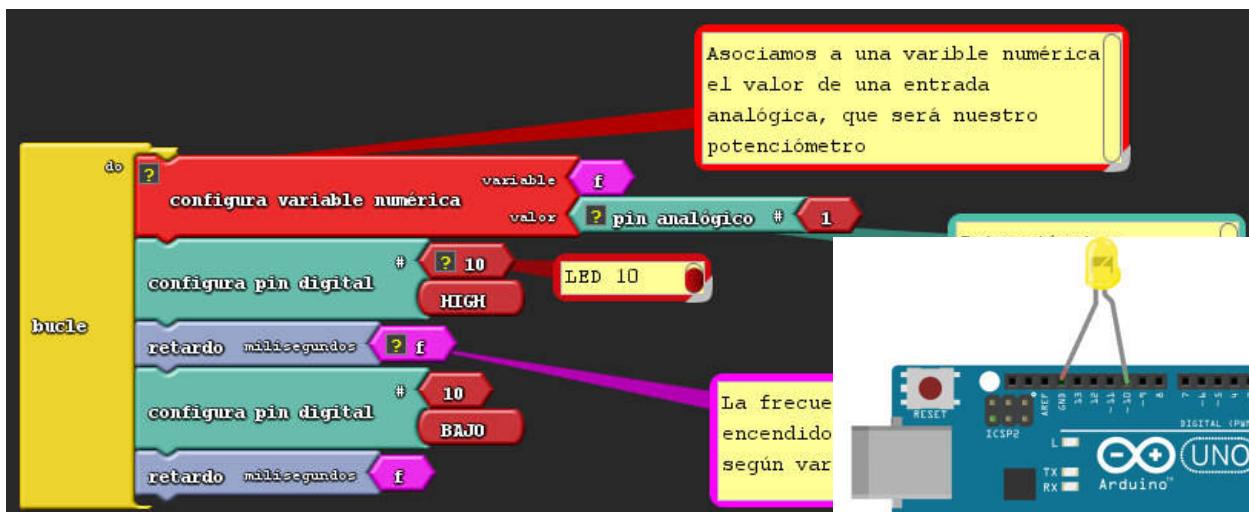
Comenzamos definiendo la variable “f”, y para ello utilizaremos la función:



Es aquí donde le daremos el nombre “f” a nuestra variable y la asociaremos al valor de la entrada analógica 1 (el potenciómetro)

Y luego en el retardo sustituiremos el valor numérico fijo, que si recordáis era de 1000ms en la primera práctica, por uno variable, “f”. Así según vayamos girando el potenciómetro la frecuencia de apagado y encendido del led irá variando.

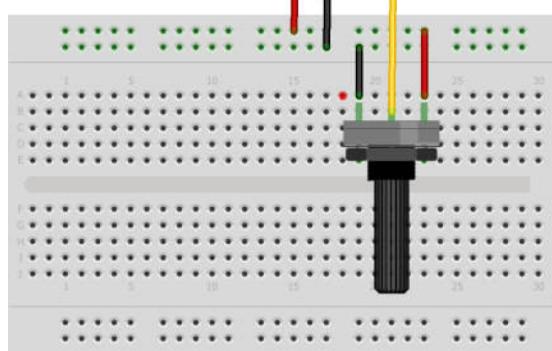
Y este será el resultado del puzzle



Y este sería el código generado para el ID de Arduino

```
int _ABVAR_1_f;  
  
void setup()  
{  
    pinMode( 10 , OUTPUT);  
    _ABVAR_1_f = 0;  
}  
  
void loop()  
{  
    _ABVAR_1_f = analogRead(A1) ;  
    digitalWrite( 10 , HIGH );  
    delay( _ABVAR_1_f );  
    digitalWrite( 10 , LOW );  
    delay( _ABVAR_1_f );  
}
```

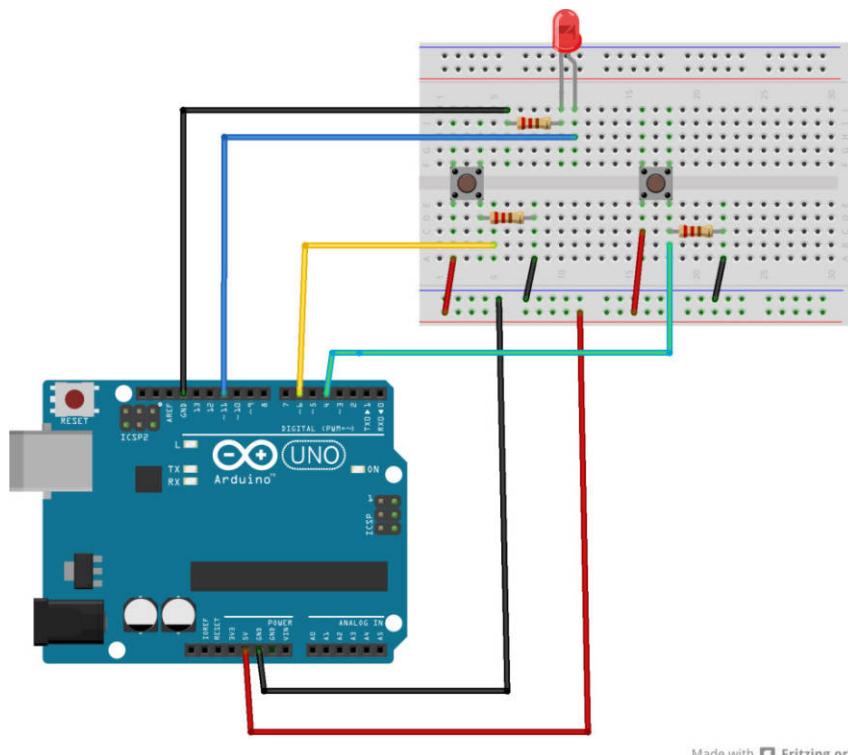
Y así es como
de los compo





10. PRÁCTICA: Conectar una salida digital con un pulsador (entrada digital) y apagarla con otra entrada digital diferente. FUNCIÓN WHILE (MIENTRAS QUE)

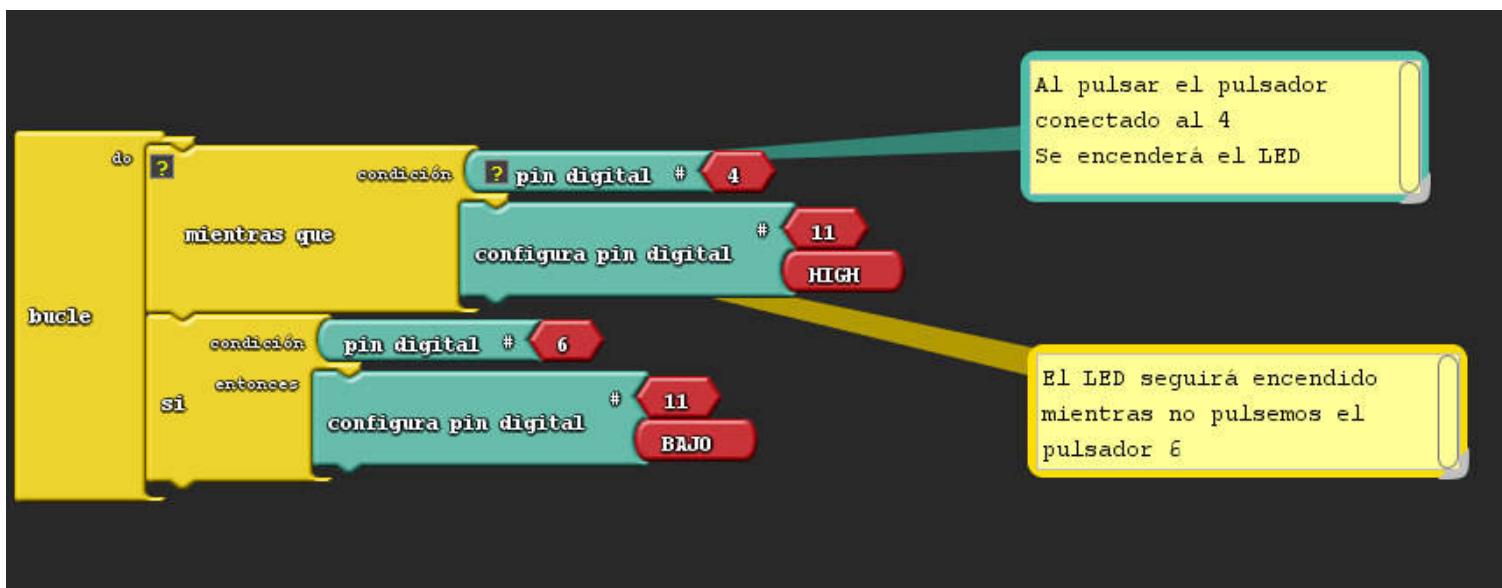
En primer lugar vamos a ver cuál será el montaje electrónico que deberíamos realizar.



Made with Fritzing.org

Vamos a utilizar esta práctica para introducir la FUNCIÓN WHILE (mientras que). Con esta función se establece una condición a evaluar y una instrucción a realizar mientras se cumpla la condición establecida.

La lógica por lo tanto de nuestro programa será muy sencilla: La salida 11 estará activada, es decir el LED estará luciendo al activar la entrada 4, al pulsar el pulsador 4, y permanecerá así mientras no pulsemos el pulsador conectado en la línea 6.



Y este sería el ID de Arduino.

```
Archivo Editar Sketch Herramientas Ayuda
sketch_feb23b §
pinMode( 11 , OUTPUT);
pinMode( 4 , INPUT);
pinMode( 6 , INPUT);
}

void loop()
{
while ( digitalRead( 4) )
{
digitalWrite( 11 , HIGH );
}

if (digitalRead( 6))
{
digitalWrite( 11 , LOW );
}
}
```

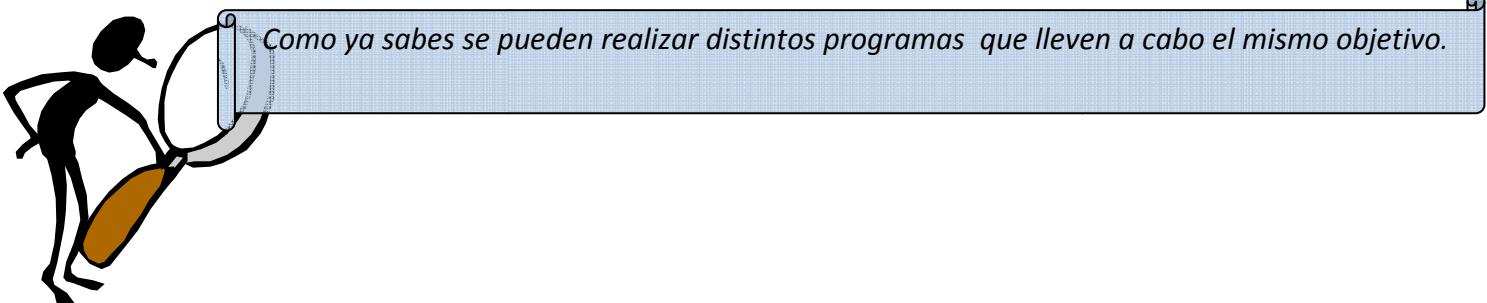
Seguro que a ti se te han ocurrido otras formas de hacer el programa, sin usar la función while, por ejemplo:



```

sketch_feb23c §
{
pinMode( 13 , OUTPUT);
pinMode( 4 , INPUT);
pinMode( 6 , INPUT);
}

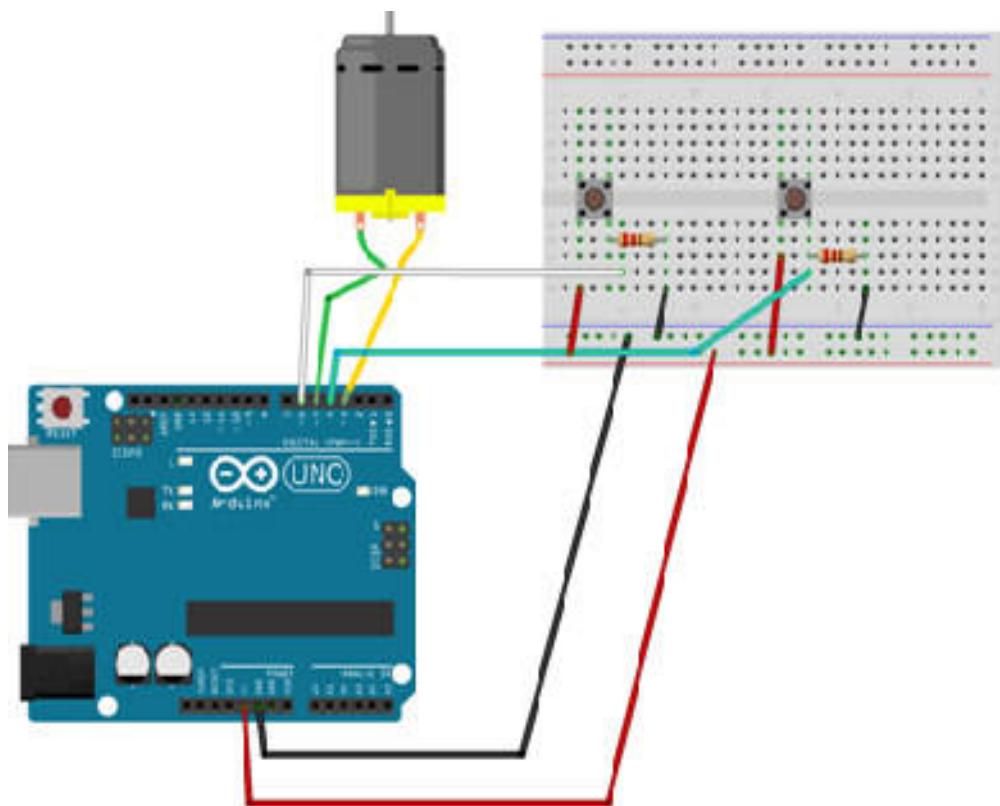
void loop()
{
if (digitalRead( 4))
{
digitalWrite( 13 , HIGH );
}
if (digitalRead( 6))
{
digitalWrite( 13 , LOW );
}
}
  
```





11. PRÁCTICA: CONTROL DE UN MOTOR

Con esta práctica vamos a llevar a cabo el cambio de giro de un motor de cc de 5V. Para ello vamos a utilizar dos pulsadores como entradas digitales, la 4 y la 6, y para conectar el motor utilizaremos los pins 3 y 5, ya que como dijimos al inicio de este manual estas entradas digitales se pueden emplear como entradas PWM, es decir, permiten simular un comportamiento analógico enviando señales de valor variable, de cara a modificar la velocidad de giro de un motor.





El motor girará en un sentido al pulsar 4 y al contrario al pulsar 6

Uno de los conectores del motor se conecta al pin 3

El motor girará en un sentido al pulsar el pulsador 4

El otro conector del motor se conecta al pin digital 5

El motor cambiará de giro al pulsar el pulsador 6

ArduBlock

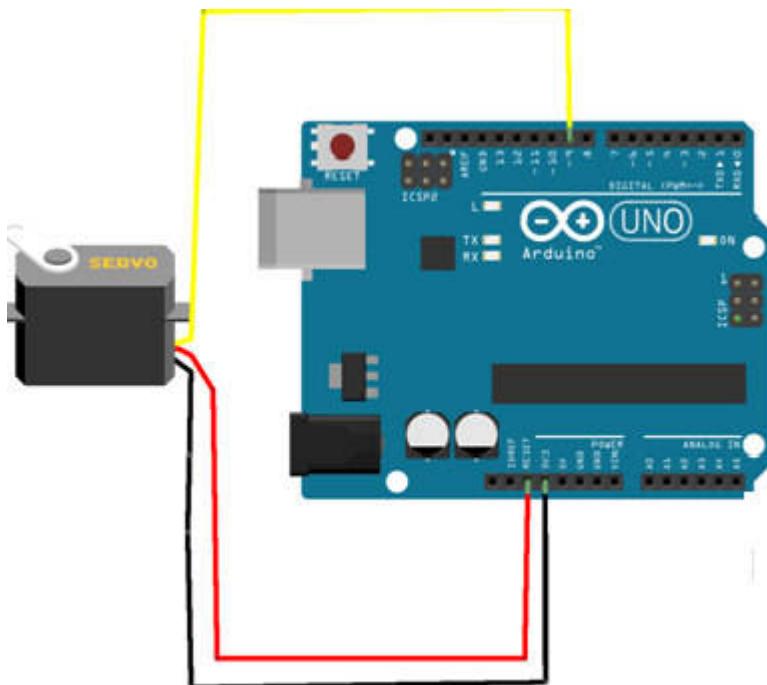
Y el código de ARDUINO quedará:

```
void setup()
{
    pinMode( 3 , OUTPUT);
    pinMode( 6 , INPUT);
    pinMode( 4 , INPUT);
    pinMode( 5 , OUTPUT);
}

void loop()
{
    digitalWrite( 3 , digitalRead( 4 ) );
    digitalWrite( 5 , digitalRead( 6 ) );
}
```

12. PRÁCTICA 8: CONTROL DE UN SERVOMOTOR (giro 180º)

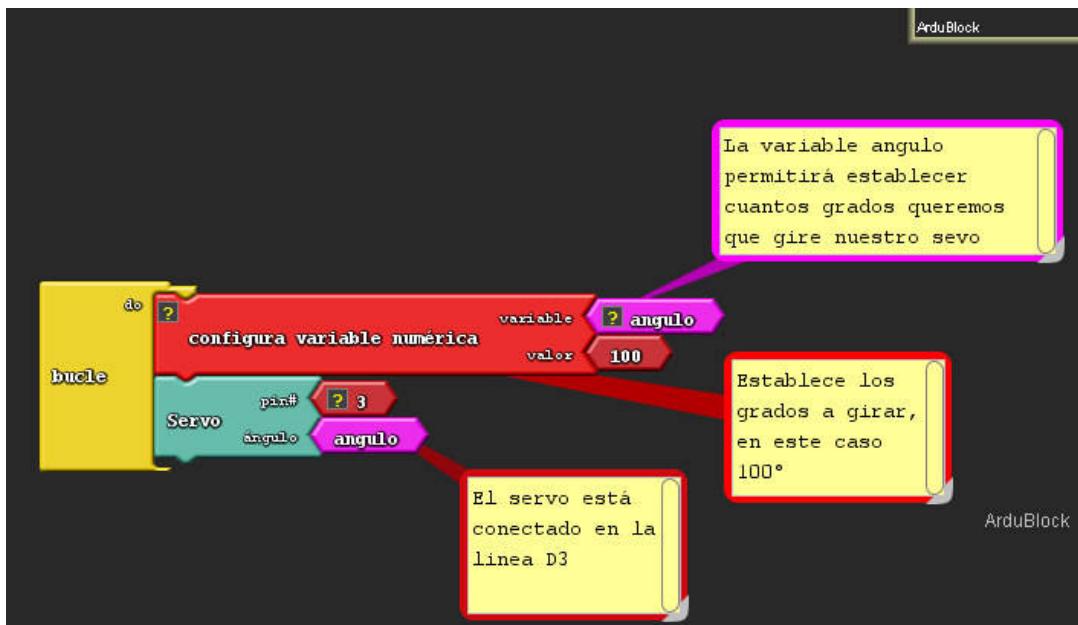
Los servos motores poseen tres terminales: alimentación, masa y señal. El terminal de alimentación, es normalmente rojo, y se debe conectar al pin 5V de la placa Arduino. El terminal de masa el cual es normalmente negro debe de conectarse a el pin de masa del la placa Arduino. El cable de señal suele ser amarillo, naranja o blanco y debe conectarse al pin 9 de la placa Arduino, y es la que recibe la señal que le indica el ángulo que debe girar, esta señal está formada por un pulso que el circuito incluido en el servo interpreta para girar el motor y dejarlo fijo en el ángulo indicado.



Ardublock tiene una ficha para el control de un servo en la librería PINS,



Este sería un ejemplo en Ardublock de cómo utilizar esta “ficha”.



Cuyo código será:

```

Archivo Editar Sketch Herramientas Ayuda
servo1
#include <Servo.h>

int _ABVAR_1_angulo;
Servo servo_pin_3;

void setup()
{
  _ABVAR_1_angulo = 0;
  servo_pin_3.attach(3);
}

void loop()
{
  _ABVAR_1_angulo = 100 ;
  servo_pin_3.write( _ABVAR_1_angulo );
}

```

Si lo que queremos es hacer girar un servo de 0º a 180º continuamente, el código Arduino más sencillo sería utilizando la función for, y podría ser así:

Código en Arduino para controlar un servo que va de 0 a 180º en pasos de un 1º:

```
#include <Servo.h>

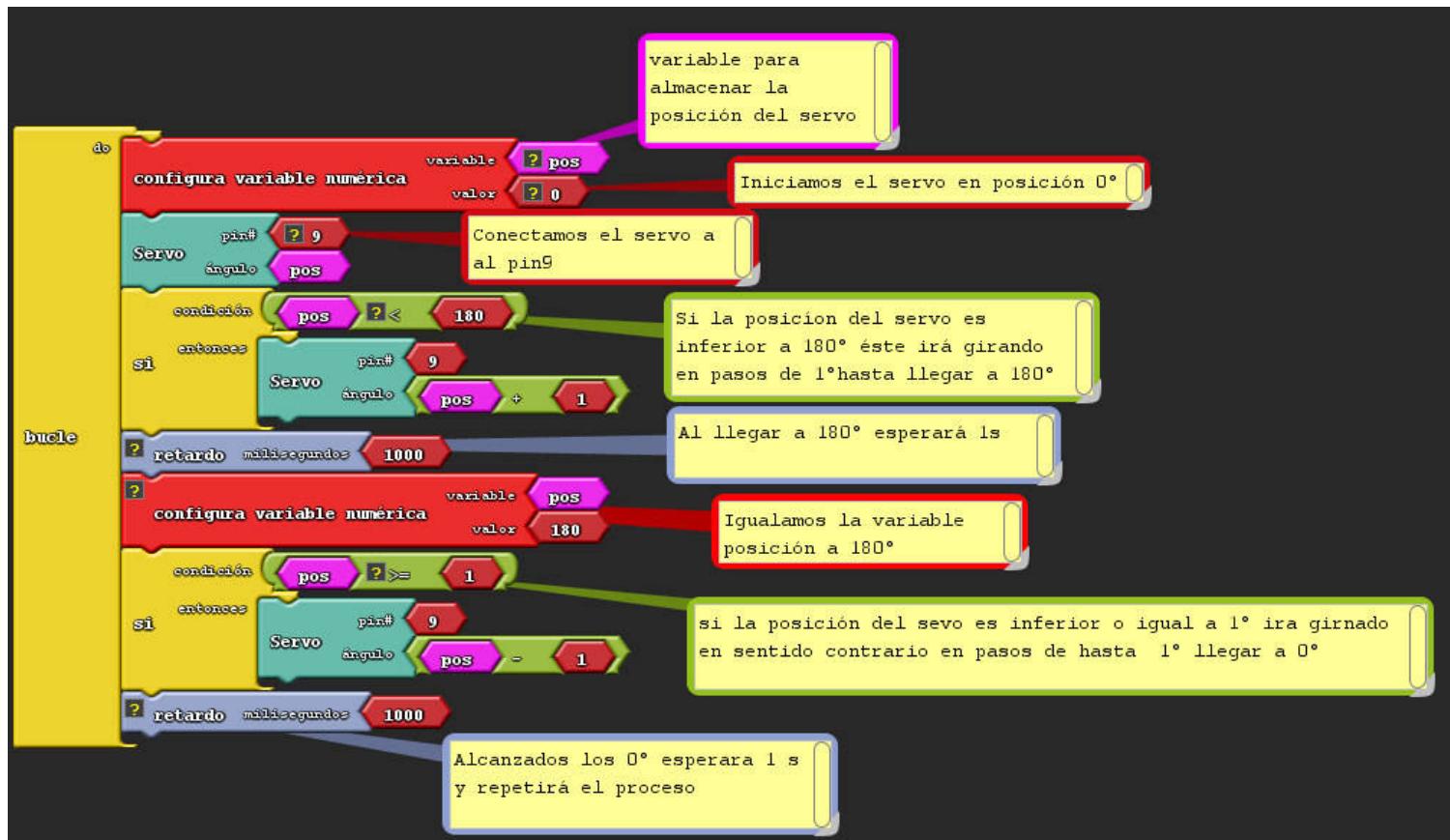
Servo myservo; // crea un objeto tipo servo para controlar el servo

int pos = 0; // variable para almacenar la posición del servo

void setup()
{
    myservo.attach(9); // liga el servo conectado en el pin 9 al objeto servo
}

void loop()
{
    for(pos = 0; pos < 180; pos += 1) // va de 0 a 180 grados
    {
        myservo.write(pos); // dice al servo que se posicione en la posición indicada
        por la variable 'pos'
        delay(15); // espera 15 ms para dar tiempo al servo a llegar a la nueva
        posición
    }
    for(pos = 180; pos>=1; pos-=1) // va de 180 a 0 grados
    {
        myservo.write(pos); // dice al servo que se posicione en la posición indicada
        por la variable 'pos'
        delay(15); // espera 15 ms para dar tiempo al servo a llegar a la nueva
        posición
    }
}
```

Pero si lo queremos hacer con Ardublock nos encontramos con un pequeño problema, que es que Ardublock no tiene una ficha para la función “for”, así que tendremos que buscar una alternativa, yo os propongo hacerlo usando la función “condicional”, con lo que nuestro programa en Ardublock quedaría así:





Y el IDE de Arduino quedaría:

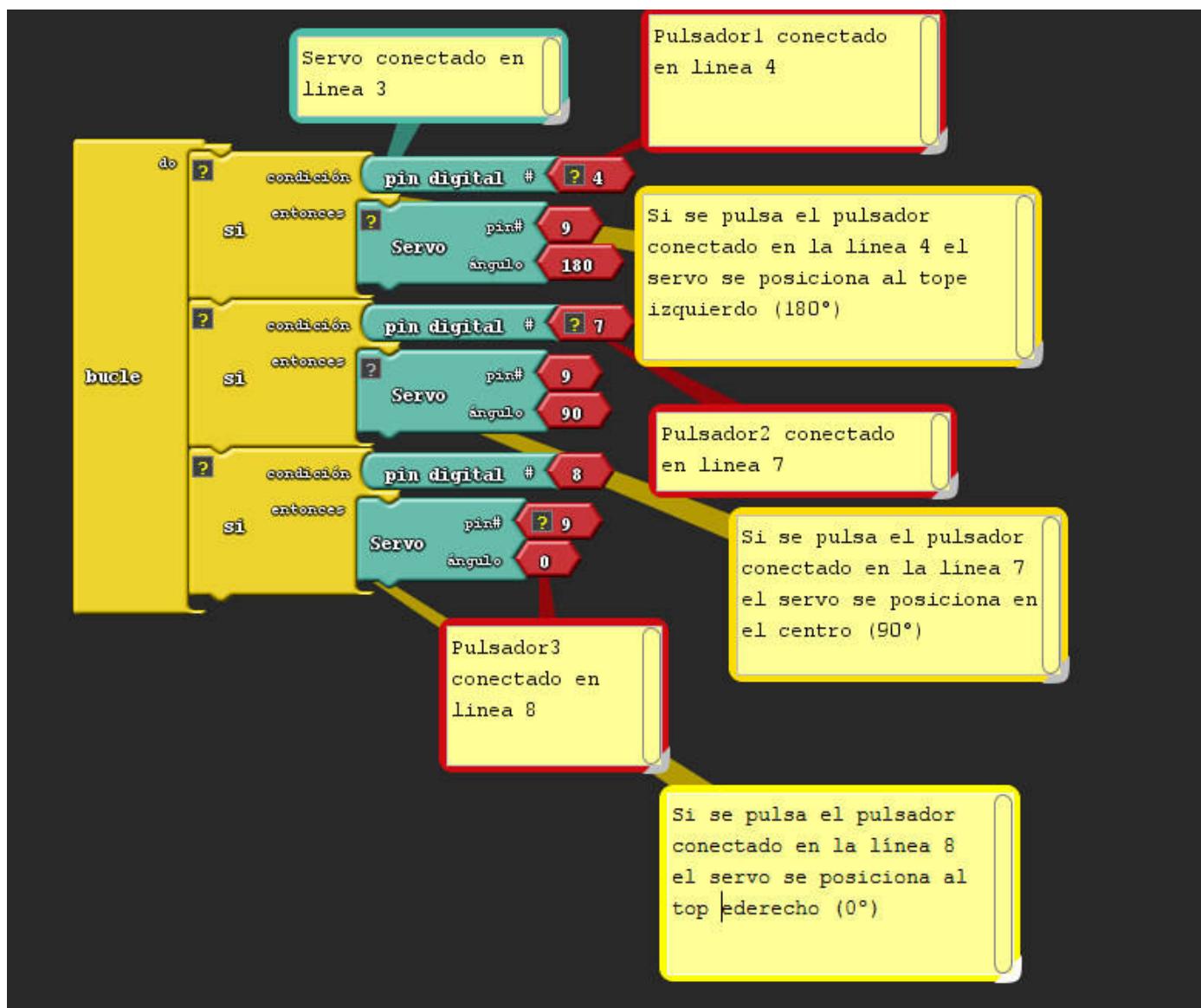
```
Archivo Editar Sketch Herramientas Ayuda
servo_girando §
{
  servo_pin_9.attach(9);
  _ABVAR_1_pos = 0;
}

void loop()
{
  _ABVAR_1_pos = 0 ;
  servo_pin_9.write( _ABVAR_1_pos );
  if (( ( _ABVAR_1_pos ) < ( 180 ) ))
  {
    servo_pin_9.write( ( _ABVAR_1_pos + 1 ) );
  }
  delay( 1000 );
  _ABVAR_1_pos = 180 ;
  if (( ( _ABVAR_1_pos ) >= ( 1 ) ))
  {
    servo_pin_9.write( ( _ABVAR_1_pos - 1 ) );
  }
  delay( 1000 );
}
```



13. PRÁCTICA : SERVO CONTRALADO POR SEÑALES EXTERNAS

Un servo también se puede controlar por señales externas como pulsadores, aquí tenéis un ejemplo de un servo controlado por tres pulsadores que harán de entradas digitales, conectadas en los pines 4, 7 y 8. Cuando pulsamos el 4 el servo girará a 180º al pulsar el 7 se situará en el centro, 90º, y al pulsar el 8 volverá a 0º.





```
servo servo_pin_9;

void setup()
{
pinMode( 8 , INPUT);
pinMode( 7 , INPUT);
servo_pin_9.attach(9);
pinMode( 4 , INPUT);
}

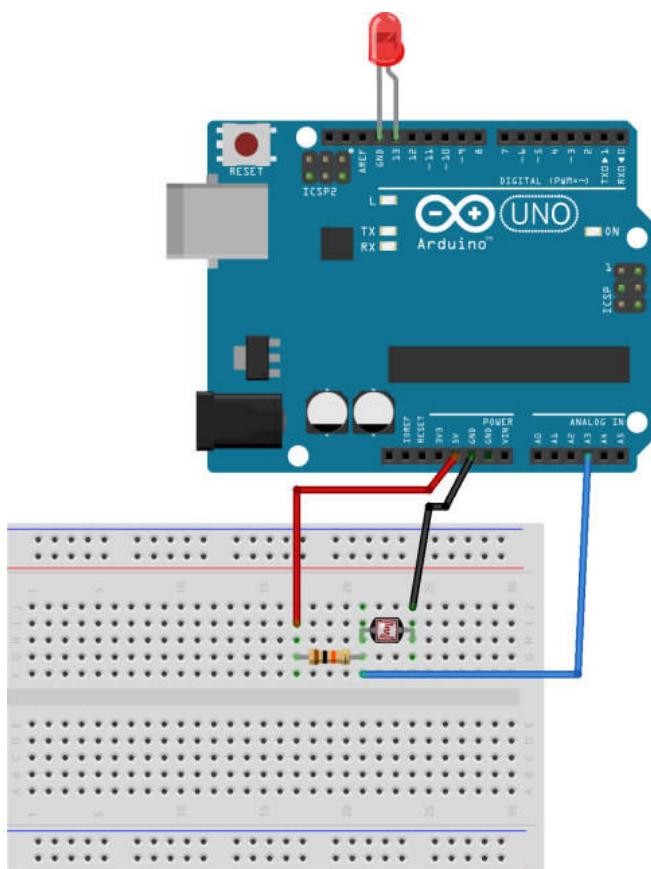
void loop()
{
if (digitalRead( 4))
{
servo_pin_9.write( 180 );
}
if (digitalRead( 7))
{
servo_pin_9.write( 90 );
}
if (digitalRead( 8))
{
servo_pin_9.write( 0 );
}
}
```



14. PRÁCTICA: SENSOR DE LUZ 1(LDR)

Este es un ejercicio muy sencillo con el que queremos activar una salida digital (un LED) conectado en el pin 13, cuando no detecte luz, y deberá apagarse cuando la LDR detecte luz, algo similar a un alumbrado público que se activa cuando oscurece y se desactiva cuando amanece.

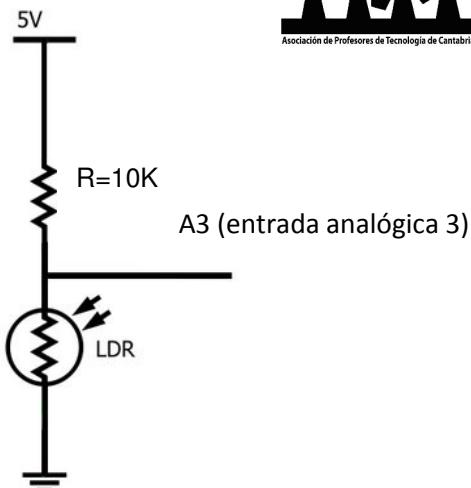
El esquema será el siguiente:



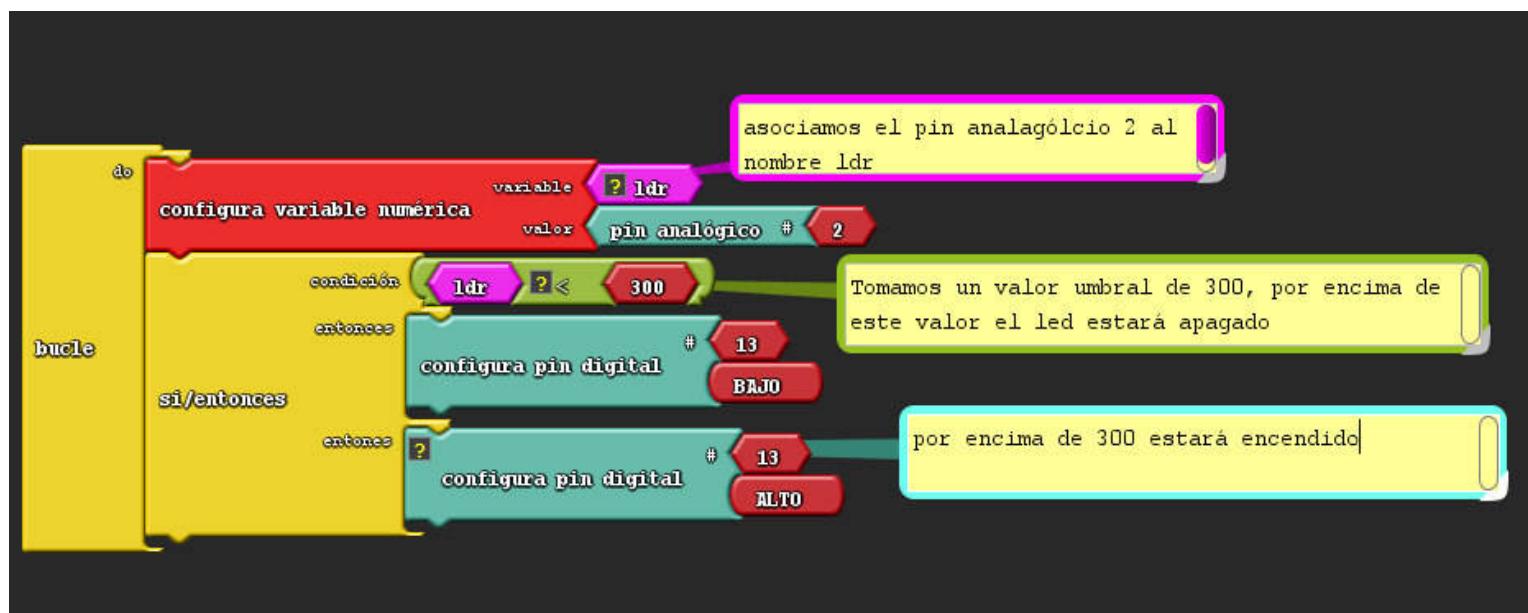
Made with Fritzing.org

Usaremos el sensor de luz conectándole a la entrada analógica 2 y tomaremos un valor umbral de 300, cuando estemos por debajo se activará el LED conectado en la salida 13 que simulará el alumbrado automático, en caso contrario, el habrá suficiente iluminación y por lo tanto el LED estará apagado.

Puedes variar el valor umbral para hacer que el LED se encienda con más o menos intensidad de luz.



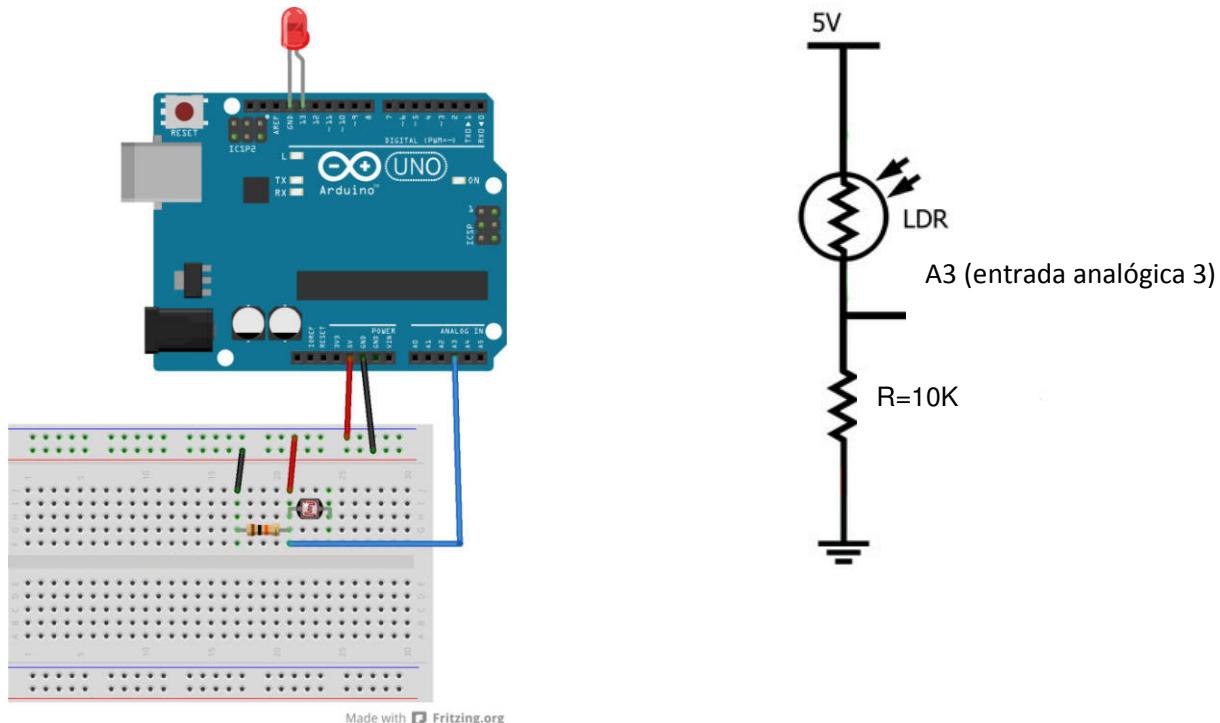
EL I programa nos quedará así:



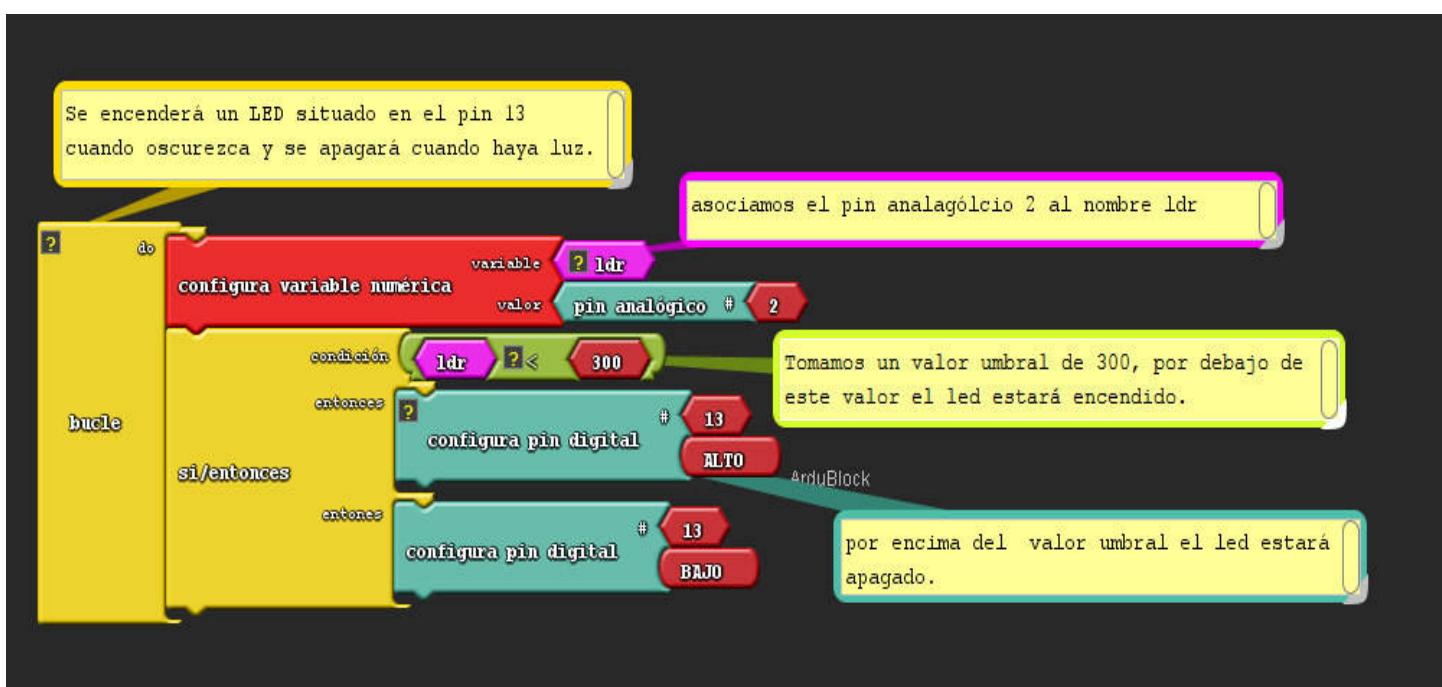
Ahora puedes probar a variar tú los valores umbral, si por ejemplo en este último caso cambias el 300 por 500 ¿el LED se iluminará con más o menos oscuridad que si tomamos un valor umbral de 300?



Comprueba que ocurre si invertimos el divisor de tensión:



Si queremos que el LED se siga enciendo cuando baje la luz la lógica del programa será ahora:

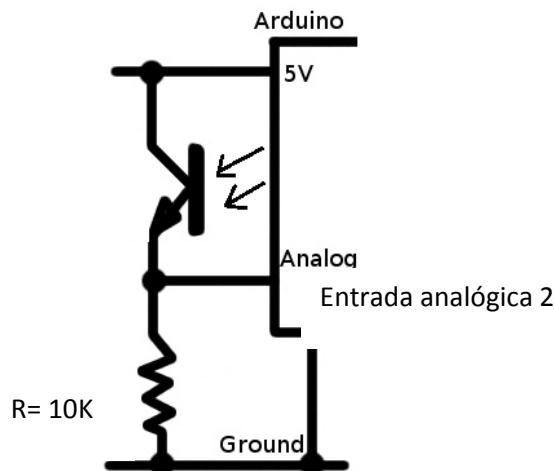




15. PRÁCTICA: SENSOR DE LUZ 2 (EL FOTOTRANSISTOR).



El fototransistor es un dispositivo capaz de dejar pasar más o menos intensidad a su través en función de la luz que incide sobre él, es decir a más luz más corriente.



Se activará un led, conectado en el pin 13, cuando el sensor de luz conectado en la entrada analógica 2 detecte un valor de lumonisidad que se corresponda con una señal umbral por debajo de 200, si el sensor detecta luz, el led se apagará





```
fototransistor

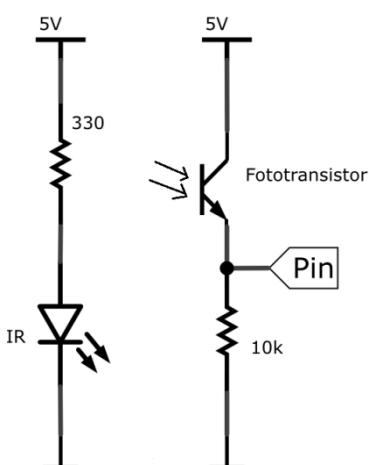
int _ABVAR_1_sensor;

void setup()
{
    _ABVAR_1_sensor = 0;
    pinMode( 13 , OUTPUT);
}

void loop()
{
    _ABVAR_1_sensor = analogRead(A2) ;
    if ( ( _ABVAR_1_sensor ) < ( 200 ) )
    {
        digitalWrite( 13 , HIGH );
    }
    else
    {
        digitalWrite( 13 , LOW );
    }
}
```

16. PRÁCTICA: SENSOR IR, SENSOR DE LUZ INFRARROJA.

Se trata de un sensor de luz infrarroja formado por dos componentes, un emisor E, alimentado a 5 V, que está continuamente emitiendo luz en la frecuencia de infrarrojos. Cuando ésta se refleja sobre un objeto rebota y la recibe el fototransistor receptor R



Según la cantidad de luz reflejada dejará pasar más o menos intensidad.

Nosotros lo vamos a utilizar para detectar una línea negra, para ello conectaremos un led rojo y otro verde como salidas digitales en los pines 11, y 9. Si el IR detecta una línea negra se encenderá el led rojo, si por el contrario no la detecta o detecta



blanco, se encenderá el led verde.

El programa será:



Que se corresponde con este código:

```
sketch_feb23c §
int _ABVAR_1_IR;

void setup()
{
pinMode( 11 , OUTPUT);
_ABVAR_1_IR = 0;
pinMode( 9 , OUTPUT);
}

void loop()
{
_ABVAR_1_IR = analogRead(A3) ;
if (( ( _ABVAR_1_IR ) < ( 300 ) ))
{
digitalWrite( 11 , HIGH );
digitalWrite( 9 , LOW );
}
else
{
digitalWrite( 9 , HIGH );
digitalWrite( 11 , LOW );
}
}
```



17. PRÁCTICA: ULTRASONIDOS SR04

Midiendo distancias con un sensor de ultrasonidos

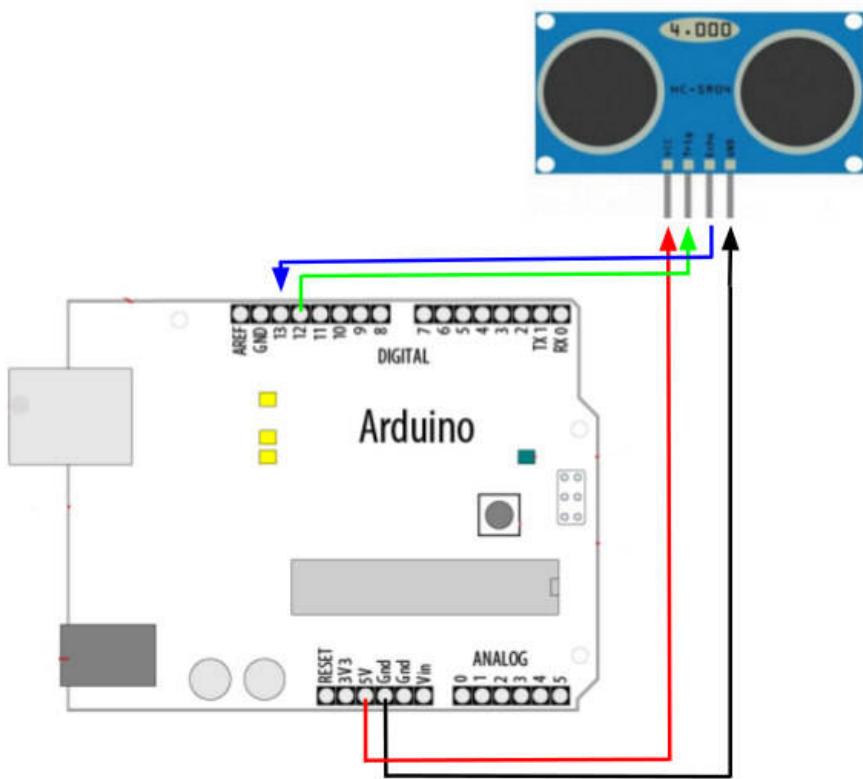
Un complemento imprescindible para muchos de robots o vehículos controlados a distancia es un sensor que nos permita saber la distancia libre de obstáculos para movernos. Si las distancias van a ser pequeñas podemos emplear sensores de infrarrojos, pero si queremos movernos en áreas grandes y poder medir distancias en un rango de varios metros el complemento perfecto es un [sensor de ultrasonidos](#). Funciona exactamente igual que un radar, de hecho es un pequeño radar. Emite un pulso de sonido a una frecuencia tan alta que es imperceptible para el oído humano y cronometra el tiempo que el sonido tarda en llegar a un obstáculo, rebotar y volver al sensor. Como la velocidad de propagación del sonido en el aire es un dato bien conocido (343.2 m/s) echamos mano de una conocidísima formula ($e = v * t$) y calculamos la distancia recorrida por el sonido..

El sensor ultrasónico HC-SR04 utiliza el sonar para determinar la distancia a un objeto como los murciélagos o los delfines hacer. Ofrece una excelente precisión del alcance y lecturas estables en un paquete fácil de usar.

Sus características son:

- Fuente de alimentación: 5V DC
- Corriente en reposo: <2 mA
- Ángulo Eficaz: <15 °
- Que van Distancia: 2 cm - 500 cm / 1 "- 16 pies
- Resolución: 0,3 cm

Su conexión a la placa Arduino se realiza así:



Un código sencillo para medir distancias sería:

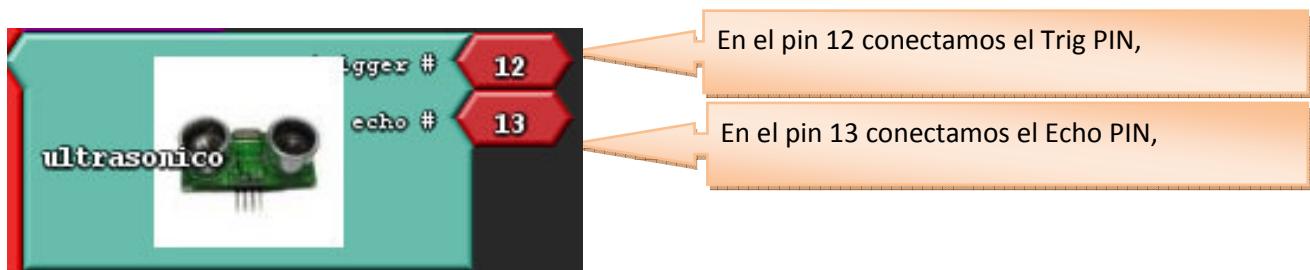
```
#include <Ultrasonic.h>

Ultrasonic ultrasonic(12,13); // (Trig PIN,Echo PIN)

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

void loop()
{
  Serial.print(ultrasonic.Ranging(CM)); // CM or INC
  Serial.println(" cm" );
  delay(100);
}
```

Pero Ardublock tiene una ficha para este sensor:



Y un programa con Ardublock para medir distancias podría ser:



Cuyo código IDE es:

```
int ardublockUltrasonicSensorCodeAutoGeneratedReturnCM(int trigPin, int echoPin)
{
    int duration;
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(5);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    duration = duration / 59;
    return duration;
}

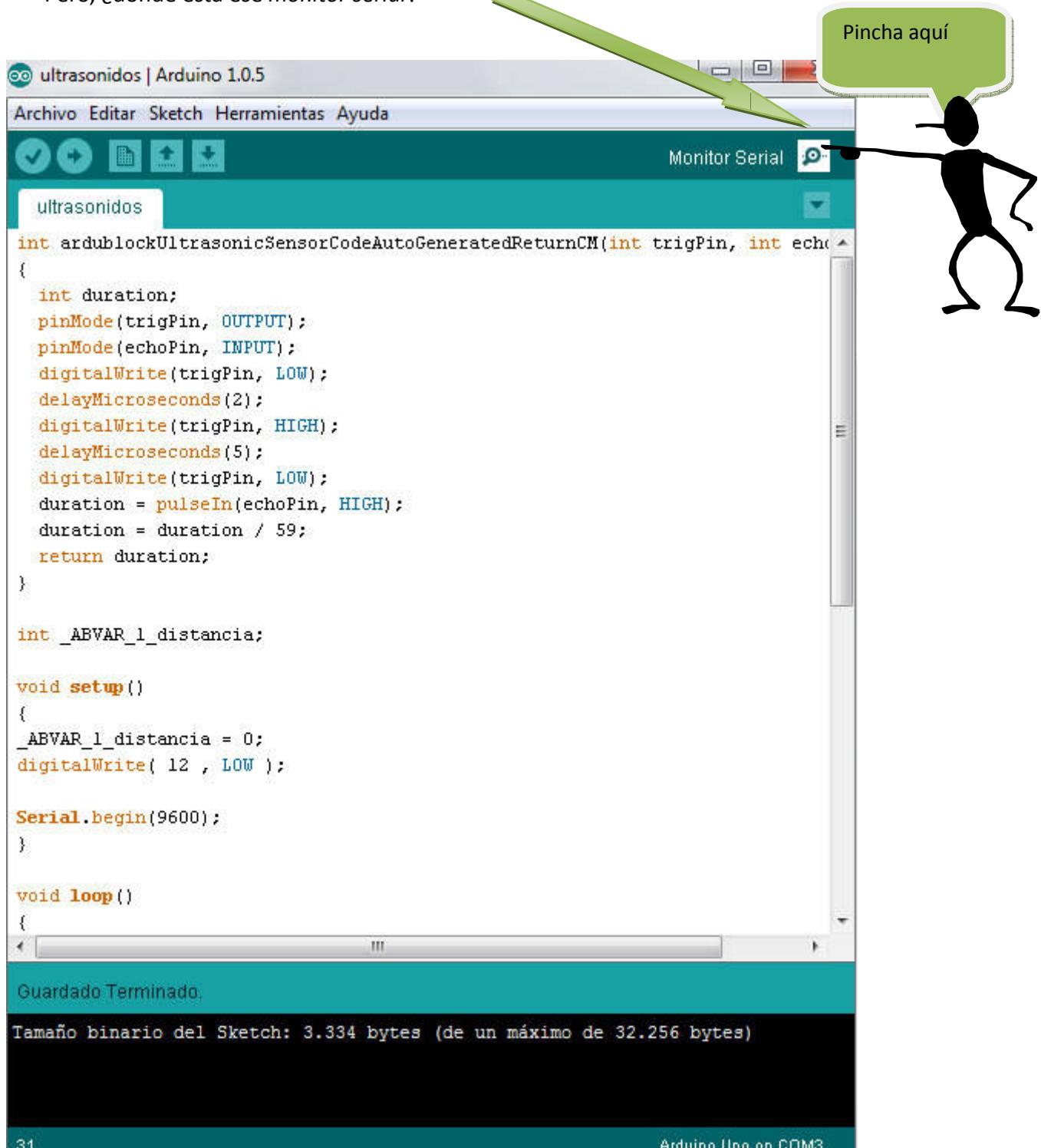
int _ABVAR_1_distancia;
void setup()
{
    _ABVAR_1_distancia = 0;
    digitalWrite( 12 , LOW );
    Serial.begin(9600);
}

void loop()
{
    _ABVAR_1_distancia = ardublockUltrasonicSensorCodeAutoGeneratedReturnCM( 12 , 13 ) ;
    Serial.print( "distancia" );
    Serial.print( _ABVAR_1_distancia );
    Serial.println("");
    delayMicroseconds( 1000 );
}
```

Este programa funciona de la siguiente manera:

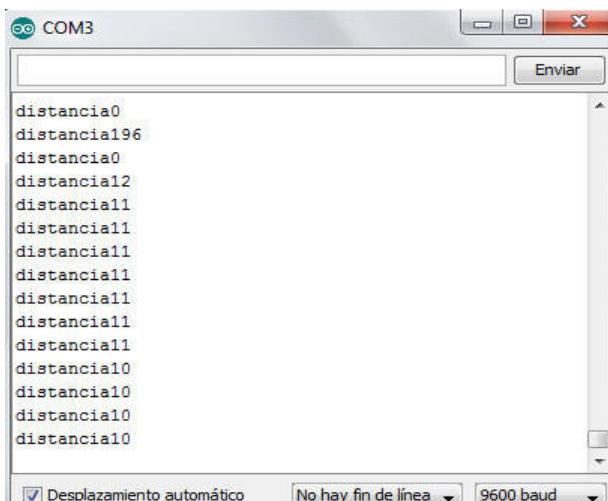
El sensor mide la distancia a la que se encuentra un objeto y cada segundo nos presenta el resultado en el *monitor serial* de Arduino.

Pero, ¿dónde está ese *monitor serial*?



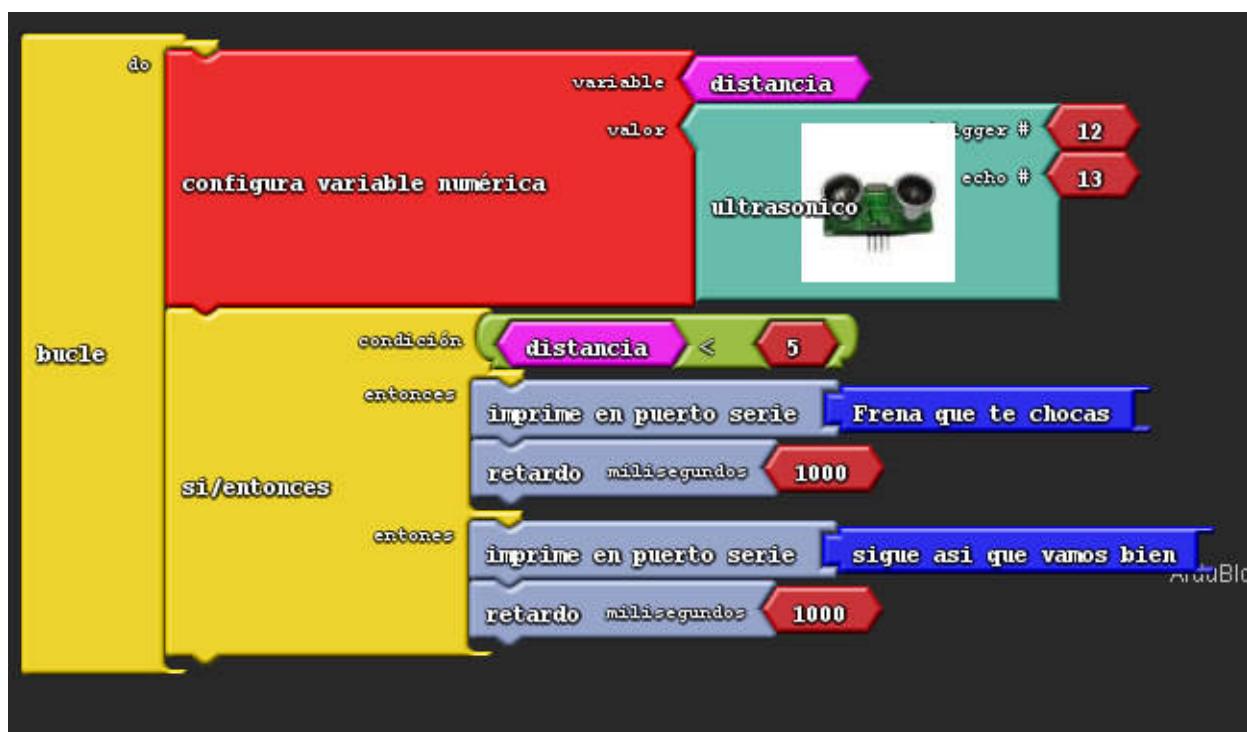


Al pinchar sobre la lupa (monitor serial) se te despliega una pantalla donde se presentan las distancias mediadas, en mi caso ésta:



A continuación te propongo una práctica: realiza un programa en el que nos aparezca en el monitor serial el mensaje: "*frena que te chocas*", cuando un objeto se encuentre a menos de 5 cm de nuestro sensor, y si no es así que nos diga "*sigue así que vamos bien*".

La solución podría ser:

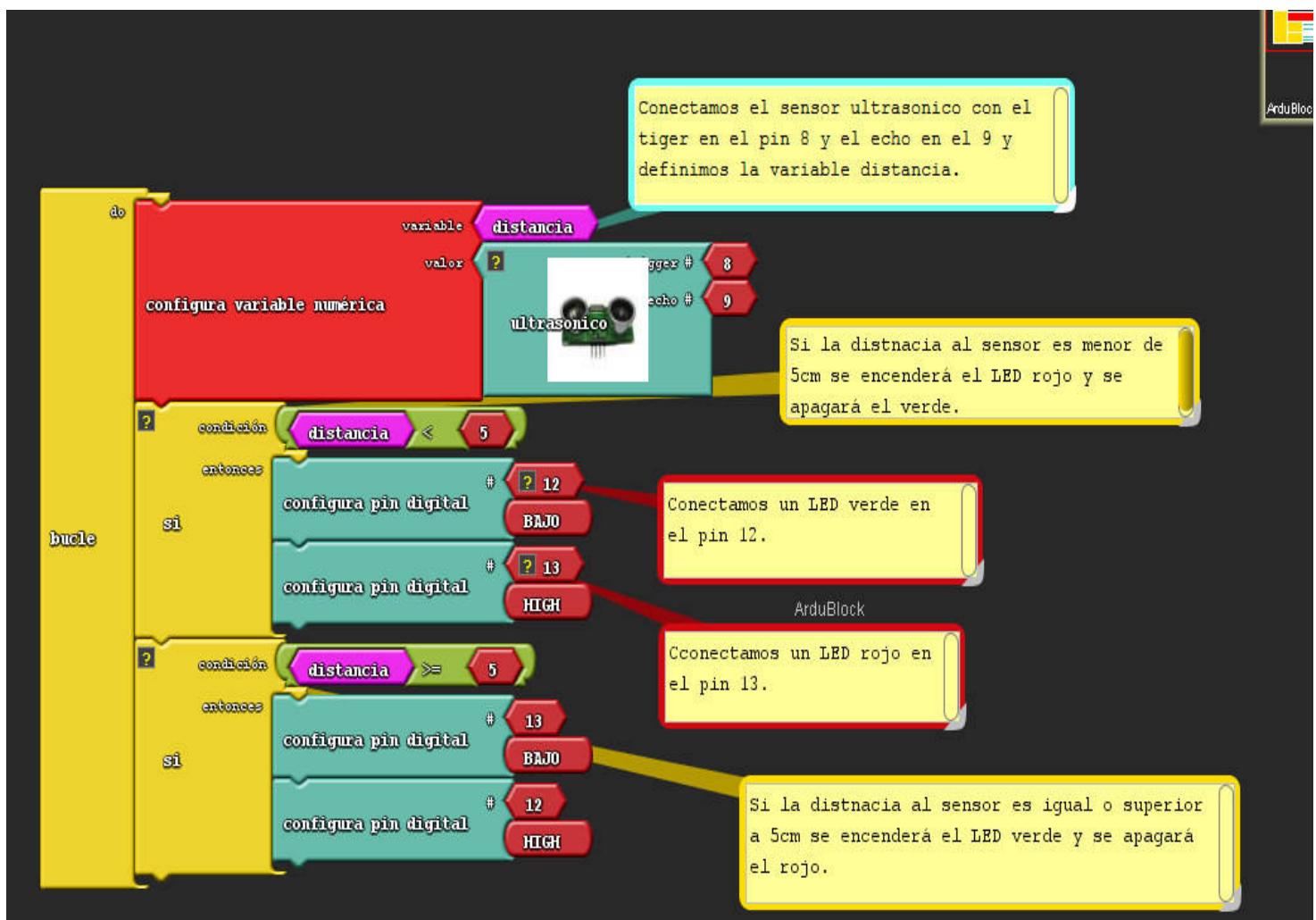


Otra propuesta:

Realiza un programa ahora para que se encienda un LED rojo cuando la distancia de un objeto al sensor sea inferior a 5 cm y un LED verde si es superior.

En este caso vamos a utilizar los pines 8 y 9 para conectar el sensor ultrasónico, y los pines 13 y 12 para conectar los LED rojo y verde respectivamente, es decir, las salidas.

Y éste sería el posible programa.



Si ahora, utilizas también motores como salidas puedes diseñar un coche que gire al detectar un obstáculo, pero eso ya te lo dejo a ti.

18. PRÁCTICA: ROBOT MINISKYBOT (ARDUINO)

Para finalizar vamos a montar un robot MinSKyBot siguiendo las intrucciones de Borjaper en su tutorial:

<http://www.arduteka.com/2013/04/robot-miniskybot-2-como-siguelineas-y-detector-de-obstaculos/>

Aunque aquí os lo dejo copiado.



[Arduino](#), [Impresion](#) [3D](#), [IR](#), [MiniskyBot](#), [Robot](#), [Robótica](#), [sensor ultranosidos](#), [TUTORIAL](#) by Borja

¿QUÉ VAMOS A HACER?

Vamos a ver varios sketch de **Arduino** para que el robot ejecute las siguientes funciones:

- Seguir una línea negra sobre un fondo blanco
- Esquivar objetos que encuentre en su recorrido
- Empujar objetos fuera de un área limitada por una circunferencia negra

¿Qué necesitamos?

En primer lugar necesitamos que alguien nos imprima en 3D las piezas del robot, si no conocéis a nadie, os podéis poner en contacto con nosotros e intentaremos hacer que no os quedéis sin ellas!

El resto del material:

- 2 servomotores
- Arduino UNO
- 2 led IR
- 2 receptores IR
- 4 resistencias (dos de 210Ω y dos de $10k\Omega$)
- 1 sensor ultrasonidos (HC-SR04)
- 1 interruptor
- tiras de pines
- placas donde soldar los componentes.

Para el montaje del robot, podéis seguir [estas instrucciones](#) creadas por los padres del proyecto.

• Caracterización de los servos

Una vez [modificados los servos](#) para que se comporten como motores que pueden girar continuamente tenemos que identificar el valor para el cual el motor está parado, en este video de [Ioleando.es](#) podéis ver en detalle cómo realizarlo:

Para saber la posición en la que está, se utiliza un potenciómetro que gira simultáneamente con el servo y que indica el ángulo en el que se encuentra. Con la modificación quitamos el potenciómetro y lo sustituimos por dos resistencias iguales con lo que la información que le llega al servo es que se encuentra siempre en la posición de 90° , si le indicamos al servo que se coloque en una posición mayor de 90° , por ejemplo 180° , girará de forma continua en un dirección, si le indicamos al servo que se coloque en una posición inferior, por ejemplo 0° , girará de forma continua en el otro sentido. Si le indicamos 90° el servo se parará.

Como las resistencias no son perfectas y tienen cierta diferencia respecto al valor nominal (aunque pongamos resistencias con 1% de tolerancia) se genera un cierto error respecto a la posición central. Así que 90° puede que no pare el motor y tendremos que buscar un ángulo cercano a 90° para que el motor quede parado. Necesitaremos identificar este dato para cada uno de los dos servos.

En el caso de los sketch indicados en este tutorial utilizo la librería Servo.h y el método writeMicroseconds para controlar los servos.

Los valores que utilzo en los dos servos son:

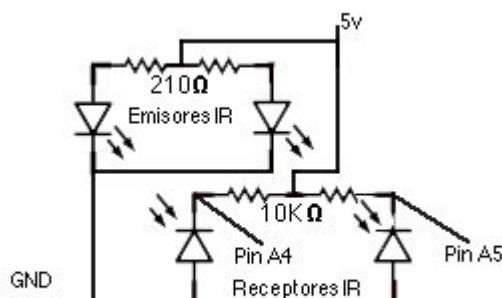
- #define derechoParada 1450
- #define izquierdoParada 1390
- #define derechoAdelante 2000
- #define izquierdoAdelante 1000
- #define derechoAtras 1000
- #define izquierdoAtras 2000

La posición a 0º suele indicarse con 1000μs y la posición 180º con 2000μs, al tener orientaciones opuestas para cada servo hay que asignar valores para que cada servo gire en un sentido diferente.

Las paradas corresponden a 1450 μs y 1390 μs.

- **Circuito IR siguelineas y sensor ultrasonidos**

El siguiente paso será soldar los emisores y receptores IR junto con sus resistencias según el siguiente esquema:



También es necesario caracterizar estos sensores para encontrar un umbral entre blanco y negro que podamos usar en el sketch. En las pruebas que he realizado al poner el sensor sobre fondo negro da un valor de 600 y al ponerlo sobre fondo blanco da 30, así que he escogido como umbral 200 o 300 para que cuando la señal sea menor de este umbral entienda que el sensor está sobre fondo blanco y corrija la trayectoria.

También incluiremos una tira de 4 pines hembra para alojar el sensor de ultrasonidos.

Finalmente se han soldado los seis hilos del trozo cable paralelo para conectar los sensores y la alimentación (5v, GND, Trig, Echo, IR izquierdo, IR derecho) con la placa Arduino.

Los receptores IR están cubiertos en la parte lateral con plástico (un pequeño trozo de globo fino del que se usa para hacer figuras) para que sólo reciban luz infrarroja por el extremo, es decir la que rebota desde el suelo.

- **Conexión con la placa Arduino.**

Usaremos el trozo de cable paralelo para conectar los sensores con la placa Arduino. El cable va soldado a otra placa en la que también hemos soldado 9 pines macho en un extremo que encajarán con las entradas analógicas, más Vin y GND y un pin que queda entre los dos bloques de pines (en total 9 pines). En el otro extremo de la placa hemos soldado 8 pines macho que encajarán con las entradas digitales (de 0 a 7) de la placa Arduino. De esta forma para conectar y desconectar la placa Arduino del robot sólo hay que sacar la placa que tiene los pines soldados.

En esta placa también colocamos dos tiras de tres pines machos (en la parte superior) donde conectaremos los cables de los servos. Por último para conectar las líneas de control de los servos usaremos dos cables que se conectarán directamente a los pines 8 y 12 de la placa Arduino.

Como vemos en la imagen hemos añadido un interruptor en el cable de alimentación que va desde las pilas a la placa para poder parar el robot.

- **Código siguelineas**

Con este código el robot es capaz de seguir un recorrido marcado por una línea negra sobre un fondo blanco. La línea debe ser lo suficientemente ancha para que los dos sensores quepan dentro de ella.



```
#include <Servo.h>

#define derechoParada 1450
#define izquierdoParada 1390
#define derechoAdelante 2000
#define izquierdoAdelante 1000

Servo derecho;
Servo izquierdo;

int iriPin=A5;
int irdPin=A4;
int irderecho,irizquierdo,derechoMovimiento,izquierdoMovimiento;

void setup()
{
  derecho.attach(8);
  izquierdo.attach(12);
}

void loop() {

  irderecho=analogRead(irdPin);
  irizquierdo=analogRead(iriPin);

  derechoMovimiento=derechoAdelante;
  izquierdoMovimiento=izquierdoAdelante;

  if (irderecho < 200){
    izquierdoMovimiento=izquierdoParada;
  }
  if (irizquierdo < 200){
    derechoMovimiento=derechoParada;
  }

  derecho.writeMicroseconds(derechoMovimiento);
  izquierdo.writeMicroseconds(izquierdoMovimiento);

}
```

La lógica es muy simple:

- Si los sensores “ven” negro (están sobre la línea) los dos motores giran hacia delante.
- Si uno de ellos “ve” blanco parará el motor del otro lado con lo que el robot girará hacia la línea negra.
- Si los dos sensores ven blanco el robot se para.

Al principio del sketch se han definido las constantes que hacen girar (o parar) a los motores.

19. Práctica: Código Escoba (Arduino)

Con este código queremos que el robot busque objetos cercanos (a menos de 50 cm) y cuando los encuentre los empuje fuera de un círculo limitado por una línea negra.



```
#include <Servo.h>
#include <Ultrasonic.h>

#define derechoParada 1450
#define izquierdoParada 1390
#define derechoAdelante 2000
#define izquierdoAdelante 1000
#define derechoAtras 1000
#define izquierdoAtras 2000

#define iriPin A5
#define irdPin A4

Servo derecho;
Servo izquierdo;

Ultrasonic ultrasonic(4,A0); // (Trig PIN, Echo PIN)

int irderecho,irizquierdo;

void setup()
{
    derecho.attach(8);
    izquierdo.attach(12);
}

void loop() {

    if (ultrasonic.Ranging(CM) < 50){
        derecho.writeMicroseconds(derechoAdelante);
        izquierdo.writeMicroseconds(izquierdoAdelante);
    }else{
        derecho.writeMicroseconds(derechoAdelante);
        izquierdo.writeMicroseconds(izquierdoParada);
    }

    irderecho=analogRead(irdPin);
    irizquierdo=analogRead(iriPin);

    if ((irderecho > 300) ||(irizquierdo > 300)) {
        derecho.writeMicroseconds(derechoAtras);
        izquierdo.writeMicroseconds(izquierdoAtras);
        delay(1000);
    }
}
```

Utilizamos las librerías Servo.h y [Ultrasonic.h](#) para manejar los servos y el sensor de ultrasonidos respectivamente.

La lógica se basa en mantener al robot girando (sólo se mueve un motor) hasta que el sensor de distancia detecte algo a menos de 50 cm. Cuando detecta un objeto avanza moviendo los dos motores.

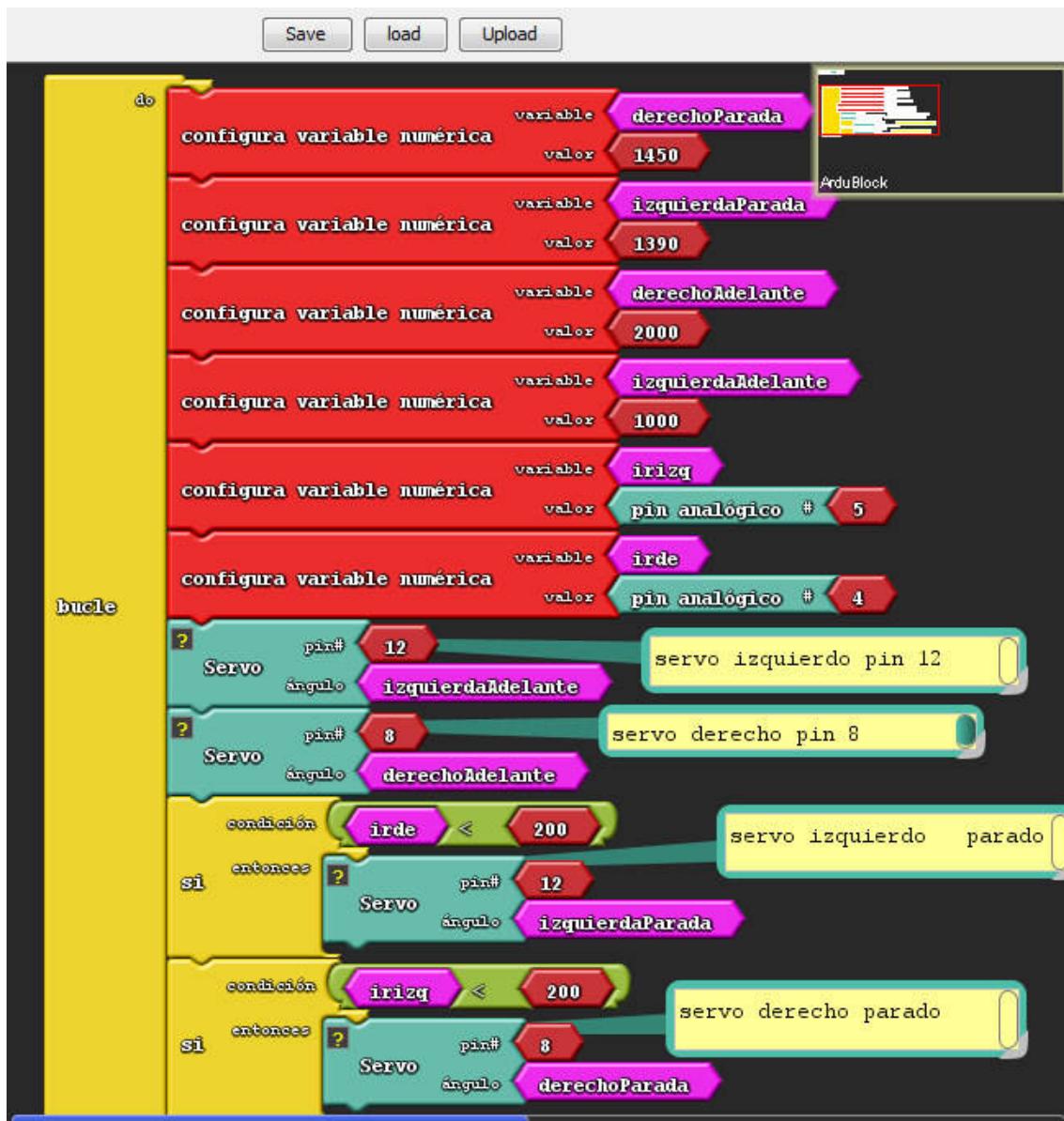
Cuando los sensores IR “ven” la línea negra el robot retrocede y el objeto queda fuera de la línea negra. Es necesario quitar el objeto ya que sino el robot estará intentando llegar al objeto y retrocediendo de forma continua.

La superficie donde están los objetos puede estar elevada y cuando el robot los saca se caen.



20. PRÁCTICA : SIGUELINEAS CON ARDUBLOCK

Si en lugar de hacerlo directamente en Arduino como hace Borja en Arduteka, queremos hacerlo con Ardublock el programa para el **SIGUELINEAS** nos quedaría así:



Cuyo código correspondiente será:

```
#include <Servo.h>

int _ABVAR_1_derechoParada;
Servo servo_pin_12;

int _ABVAR_2_izquierdaParada;
int _ABVAR_4_izquierdaAdelante;
int _ABVAR_6_irde;
int _ABVAR_5_irizq;
int _ABVAR_3_derechoAdelante;
Servo servo_pin_8;

void setup()
{
    _ABVAR_4_izquierdaAdelante = 0;
    _ABVAR_3_derechoAdelante = 0;
    _ABVAR_5_irizq = 0;
    servo_pin_8.attach(8);
    _ABVAR_2_izquierdaParada = 0;
    servo_pin_12.attach(12);

    _ABVAR_1_derechoParada = 0;
    _ABVAR_6_irde = 0;
}

void loop()
{
    _ABVAR_1_derechoParada = 1450 ;
    _ABVAR_2_izquierdaParada = 1390 ;
    _ABVAR_3_derechoAdelante = 2000 ;
    _ABVAR_4_izquierdaAdelante = 1000 ;
    _ABVAR_5_irizq = analogRead(A5) ;
    _ABVAR_6_irde = analogRead(A4) ;
```

```

servo_pin_12.write( _ABVAR_4_izquierdaAdelante );

servo_pin_8.write( _ABVAR_3_derechoAdelante );

if ( ( _ABVAR_6_irde ) < ( 200 ) )

{
    servo_pin_12.write( _ABVAR_2_izquierdaParada );

}

if ( ( _ABVAR_5_irizq ) < ( 200 ) )

{
    servo_pin_8.write( _ABVAR_1_derechoParada );
}
}

```

Puedes compararlo con el que nos propone Borja en Arduteka.

Ahora ya os doy el relevo a vosotros: ¡A diseñar vuestros propios programas!



DOCUMENTACIÓN – BIBLIOGRAFÍA - WEBGRAFÍA:

- Arduino la tecnología al alcance de todos de Mikel Etxebarria Isuskiza
- [Ardublock + Arduino tutorial](#) de José Manuel Ruiz Gutiérrez (Diciembre de 2011)
Versión de Documento: V1.0 José Manuel Ruiz Gutiérrez. Blog de referencia:
<http://josemanuelruizgutierrez.blogspot.com/33>
- <http://www.arduino.cc/>
- Ángela Bravo (2012). [Ardublock: Entorno de programación gráfica para Arduino](#). Recuperado 12 de noviembre de 2013, desde
<http://solorobotica.blogspot.com.es/search?q=ardublock>
- Borja (2013), [Robot miniskybot 2 como siguelíneas y detector de obstáculos](#).
Recuperado 11 de enero de 2014 desde:
<http://www.arduteka.com/2013/04/robot-miniskybot-2-como-siguelineas-y-detector-de-obstaculos/>
- [Opiron Electronics](#) (2013). [Todo sobre Ardublock by Opiron](#). Recuperado el 12 de noviembre de 2013 desde : <http://www.opiron.com/portfolio/tutorial-sobre-ardublock>

- Pablo Murillo(2011). [Tutorial Arduino # 0004 – Sensor LDR](#). Recuperado 9 de enero de 2014 desde: <http://www.arduteka.com/2011/11/tutorial-arduino-0004-sensor-ldr/>
- <http://www.ardublog.com/>

- Antony García González (2013). HC-SR04: *Sensor ultrasónico para Arduino*. Recuperado desde <http://panamahitek.com/hc-sr04-sensor-ultrasonico-para-arduino/>