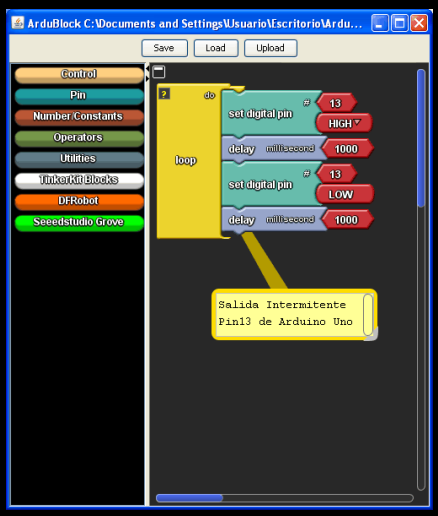
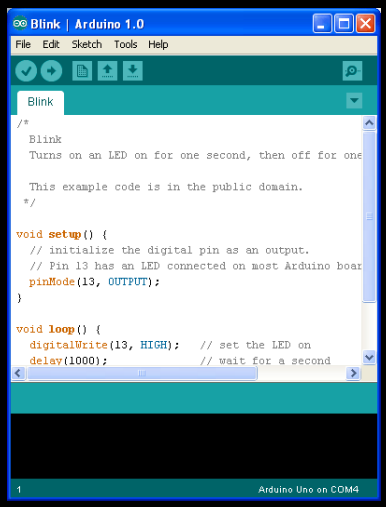
3

IDE Arduino + Ardublock

Utilización de Ardublock para la generación en modo gráfico de aplicaciones para el IDE Arduino





Ver. 1.0

José Manuel Ruiz Gutiérrez

Serie: Herramientas Gráficas para la programación de Arduino

Índice

Diciembre de 2011 Versión de Documento: V1.0
José Manuel Ruiz Gutiérrez j.m.r.gutierrez@gmail.com
Blog de referencia: http://josemanuelruizgutierrez.blogspot.com/

1. Objetivo de este trabajo.

Con el presente trabajo práctico pretendo dar a conocer las posibilidades de la herramienta Ardublock en conjunción con el IDE Arduino. Se trata de un plugin que permite la elaboración del programa para Arduino sin necesidad de escribir el código con la sintaxis de sus órdenes.

Esta posibilidad gráfica de elaborar programas es muy útil cuando estamos utilizando la Plataforma Arduino en niveles educativos. La facilidad de realizar la aplicación gráficamente permite que el alumno se dedique a pensar en el algoritmo más que en la corrección del código escrito.

En este trabajo aporto una colección de ejemplos que permitirán al lector comprender las posibilidades de esta poderosa conjunción Arduino + Ardublock y le animarán a continuar facilitándole el conocimiento de una de las plataformas Open Hardware más interesantes y difundidas en el mundo.

Ardublock es una aplicación libre que se puede obtener en:

http://blog.ardublock.com/

2. Una introducción general a Ardublock

(ArduBlock)

Descripción general

Esta herramienta está basada en la tecnología de programación mediante bloques funcionales tan extendida en la actualidad. Realmente se distribuye como un applet de java que se añade a las herramientas del IDE Arduino.

Realmente Ardublock es una utilidad gráfica cuya misión es generar código compatible con el entorno IDE Arduino. Sus ventajas son:

- Es una herramienta gratuita.
- Facilita la creación de sketch para Arduino.
- Genera código directamente.
- Ofrece una colección de bloques funcionales muy básicos que facilitan la comprensión de la programación.
- Esta muy indicado para aplicarlo en niveles educativos básicos en donde el usuario no necesita tener conocimientos de programación.
- Es una aplicación muy sencilla de instalar.
- Es muy sencillo de utilizar

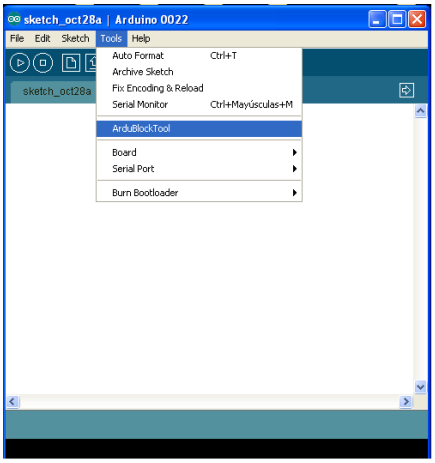
Procedimiento e instalación

1. Descarguar *ardublock-all.jar* ArduBlock.

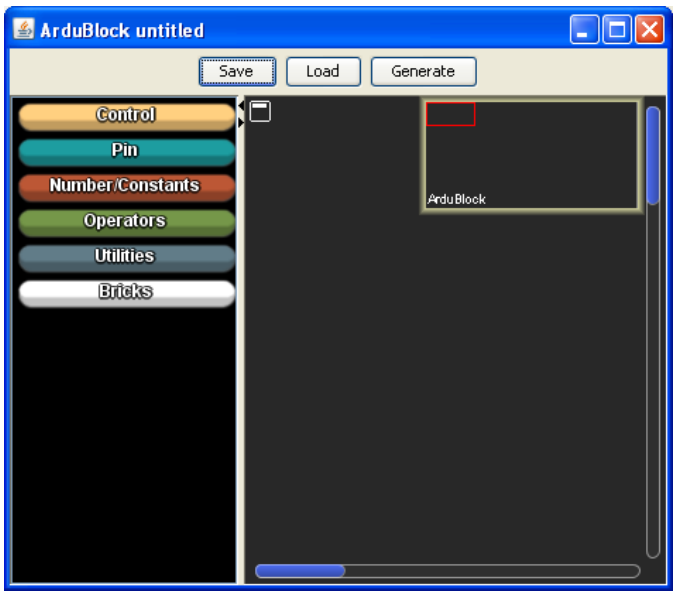
2. En la carpeta en donde este instalado el IDE Arduino debemos incluir el fichero *ardublock-all.jar* "…./arduino-022/tools/ArduBlockTool/tool/ardublock-all.jar"

Modo de trabajo

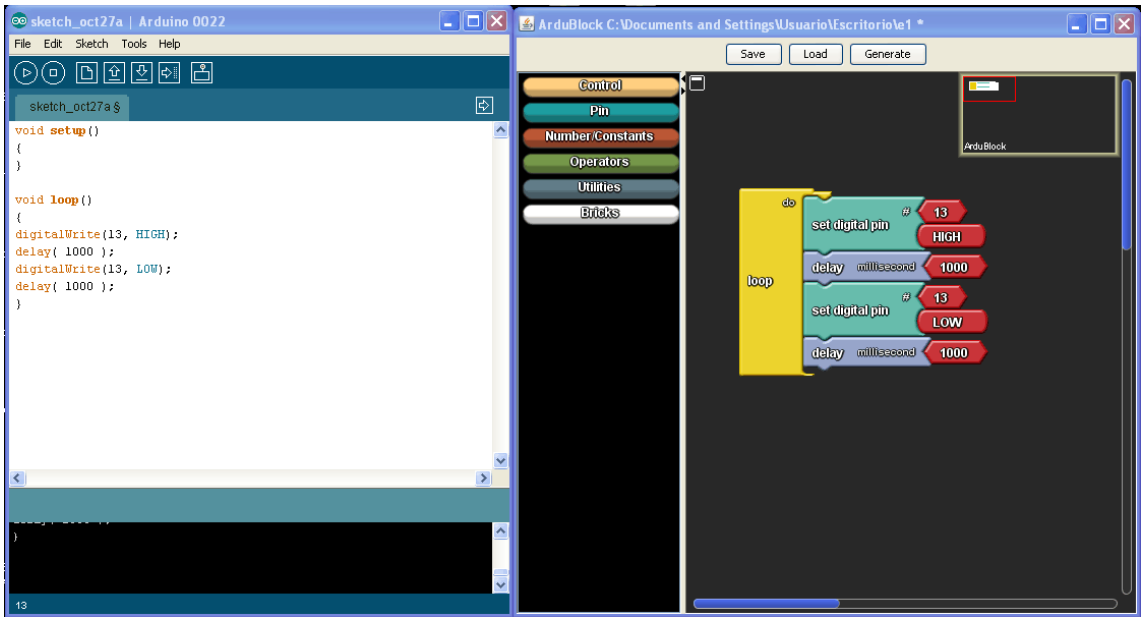
Una vez instalado el fichero ardublock-all.jar en la correspondiente carpeta se arranca el IDE de Arduino y para invocar Ardublock basta que seleccionemos *Tools-> Ardublock Tool*



Después de seleccionar la herramienta aparece la pantalla de programación gráfica de la figura.

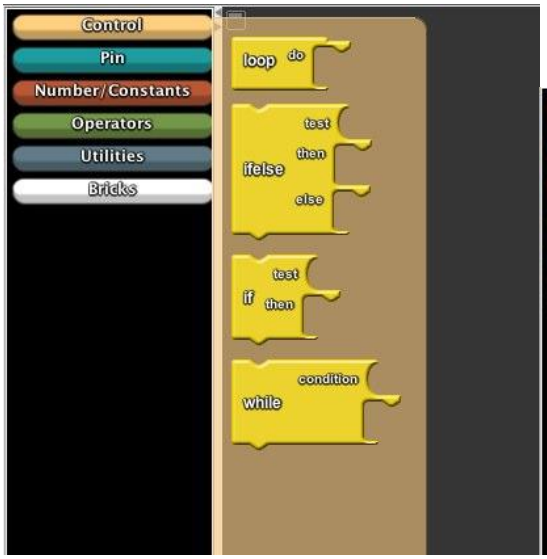


Se realiza la programación gráfica con la ayuda del entorno seleccionando los bloques correspondientes y una vez termina se activa sobre el botón “Generate “ y aparece en el el IDE de Arduino el código escrito de la aplicación listo para ser transferido a la tarjeta Arduino.



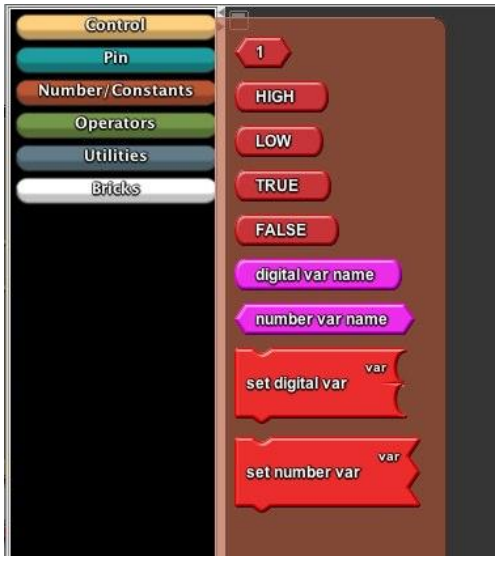
Librerías de Ardublock.

A continuación se muestran las librerías de bloques con las que se cuenta en el entorno.

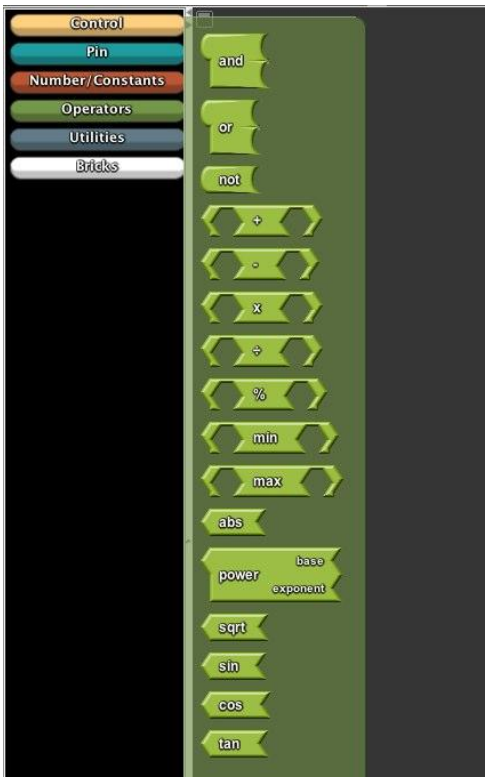


Control

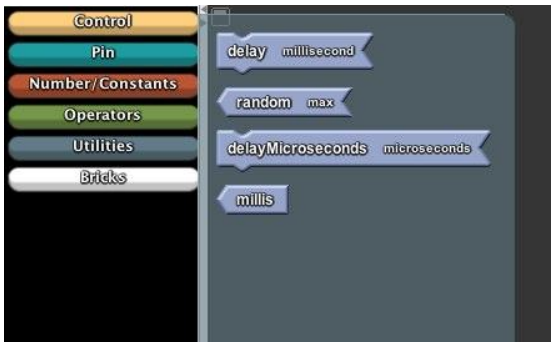
Pin



Números/Constantes



Operadores

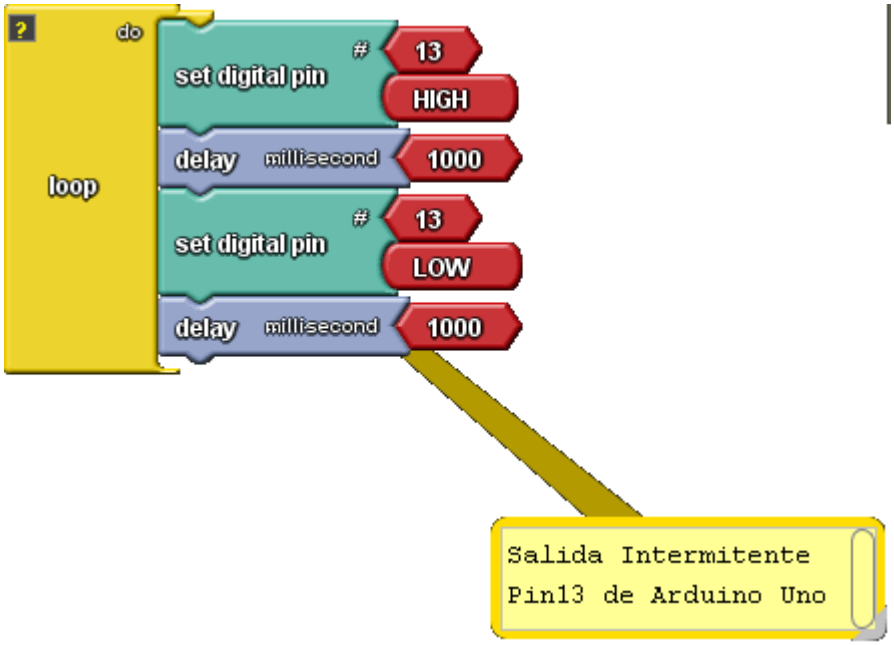


Utilidades

Bloques de Hardware

3. Salida Intermitente

La siguiente aplicación es la más sencilla de todas y suele servir para probar que las cosas funcionan. Se trata de encender y apagar la salida digital establecida en el **Pin 13**.



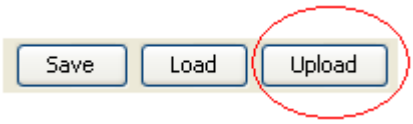
Dieño gráfico del algoritmo con Ardublock

Dentro del bloque **"loop"** que siempre se ha de colocar en las aplicaciones y que eauivale al bloque loop del programa que se escribe de manera convencional ocn el IDE Arduino **"void loop"**

Los bloques **"set digital pin"** sirven para forzar el estado de una salida

El bloque **"delay"** sirve para establecefr un retardo en msg

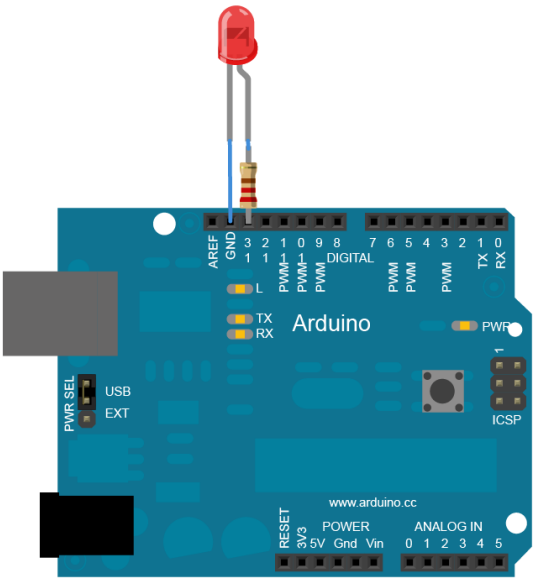
Una vez que hemos realizado el esquema grafico pulsamos el botón **Upload**



Y el programa se excribe de manera automatica en el IDE Arduino y podremos proceder a su carga en la tarjeta Arduino Uno

```
void setup()
{
pinMode( 13 , OUTPUT);
}

void loop()
{
digitalWrite( 13 , HIGH );
delay( 1000 );
digitalWrite( 13 , LOW );
delay( 1000 );
}
```

Colocamos un led para mostrarnos informacion del estado de la señal en el lado fisico de la trajeta Arduino (Pin Digital 13)



Montaje de la aplicación

4. Funciones lógicas

En el siguiente ejemplo se implemntaran en la tarjeta Arduino cuatro funciones lógicas distintas que activaran 4 salidas respectivamente que se alimentaran a través de dos entradas.

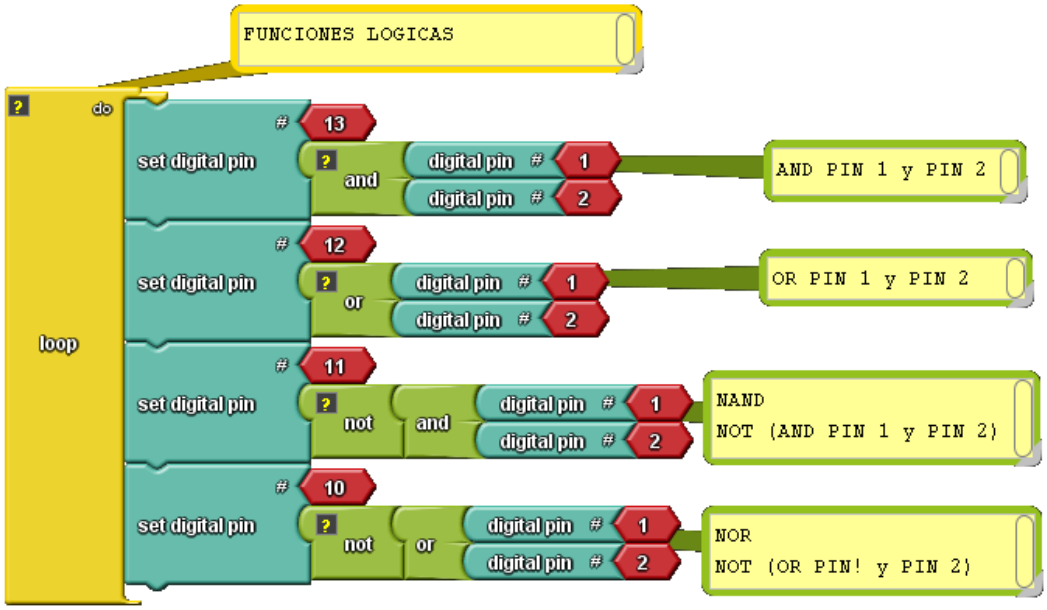
Designacion de entradas:

Entrada 1 **PIN 1**
Entrada 2 **PIN 2**

Designacion de salidas:

Salida función	AND	(Entrada1 ,Entrad2)	**PIN 13**
Salida función	OR	(Entrada1 ,Entrad2)	**PIN 12**
Salida función	NAND	(Entrada1 ,Entrad2)	**PIN 11**
Salida función	NOR	(Entrada1 ,Entrad2)	**PIN 10**

Para la implemntacion del programa se ha cosntruido el algoritmo a base de utilizar funciones con operadores booleanos: AND, OR, NOT



El código generado para el IDE Arduino es el siguiente.

```
void setup()
{
pinMode( 11 , OUTPUT);
pinMode( 10 , OUTPUT);
pinMode( 2 , INPUT);
pinMode( 12 , OUTPUT);
pinMode( 13 , OUTPUT);
pinMode( 1 , INPUT);
}

void loop()
{
digitalWrite( 13 , ( digitalRead( 1) && digitalRead( 2) ) );
digitalWrite( 12 , ( digitalRead( 1) || digitalRead( 2) ) );
digitalWrite( 11 , !( ( digitalRead( 1) && digitalRead( 2) ) ) );
digitalWrite( 10 , !( ( digitalRead( 1) || digitalRead( 2) ) ) );
}
```

5. Sistema Combinacional.

En este ejemplo vamos a realizar un sistema de control combinacional muy sencillo qe consitira en el encendido de tres lamaparas: Salida 1, Salida 2 y Salida 3 mediente tres interruptores Entrada 1, Entrada 2 y Entrada 3.

Designacion de Salidas

- L1 Lámpara 1: Salida 1: **PIN 13**
- L2 Lámpara 2: Salida 1: **PIN 12**
- L3 Lámpara 3: Salida 1: **PIN 11**

Designacion de Entradas:

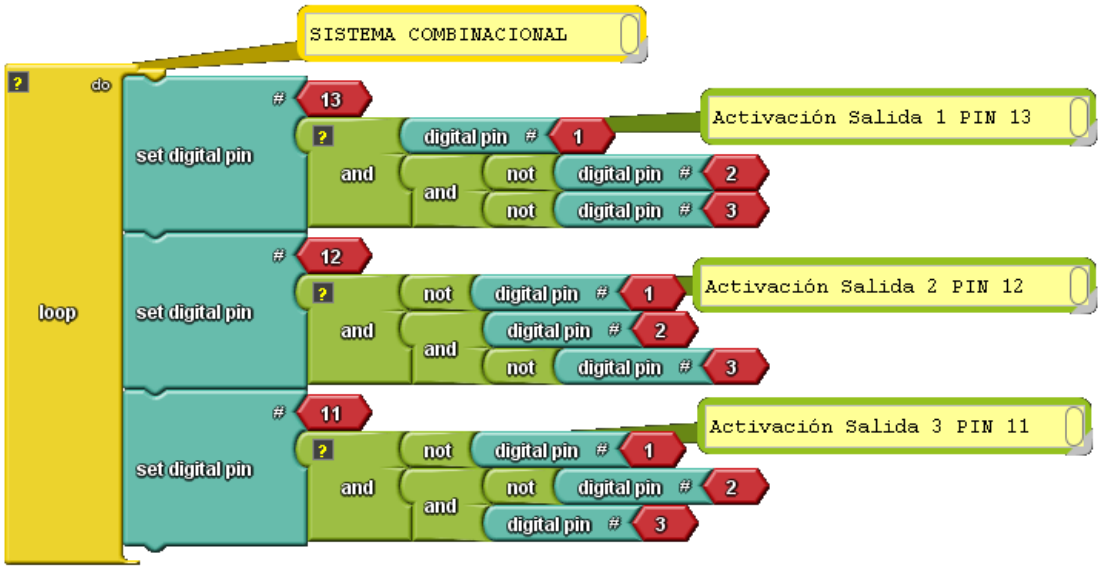
- E1 Entrada 1: **PIN 1**
- E2 Entrada 2: **PIN 2**
- E3 Entrada 3: **PIN 3**

Funcionameinto:

- Si activamos Interruptor 1 se activa la lamapara 1: Salida 1
- Si activamos Interruptor 2 se activaran las lamapara 1y 2: Salida 1 y Salida 2
- Si activamos Interruptor 3 se activa la lamapara 1, 2 y 3: Salida 1, Salida2 y Salida 3
- A continuacion escribimos la tabla de verdad de este seistema combinacional.

Entradas			**Salidas**		
E3	**E2**	**E1**	**L1**	**L2**	**L3**
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	1	0
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	1		0	0	0
1	1	1	0	0	0

A continuación se muestra el algoritmo implementado con Ardublock



El siguiente es el código generado para el IDE Arduino.

```
void setup()
{
pinMode( 11 , OUTPUT);
pinMode( 2 , INPUT);
pinMode( 3 , INPUT);
pinMode( 12 , OUTPUT);
pinMode( 13 , OUTPUT);
pinMode( 1 , INPUT);
}

void loop()
{
digitalWrite( 13 , ( digitalRead( 1) && ( !( digitalRead( 2) ) && !( digitalRead( 3) ) ) ) );
digitalWrite( 12 , ( !( digitalRead( 1) ) && ( digitalRead( 2) && !( digitalRead( 3) ) ) ) );
digitalWrite( 11 , ( !( digitalRead( 1) ) && ( !( digitalRead( 2) ) && digitalRead( 3) ) ) );
}
```

6. Comparador de Entrada analógica con una constante.

En esta aplicación se trata de realizar la comparación de una de las señales de entrada analógica "A2" de la tarjeta Arduino (procedente de un sensor) con un valor constante (en este caso 100). En función del resultado de la comparación se activará la salida PIN 13 de Arduino (1 si A2 > 100 y 0 en caso contrario)

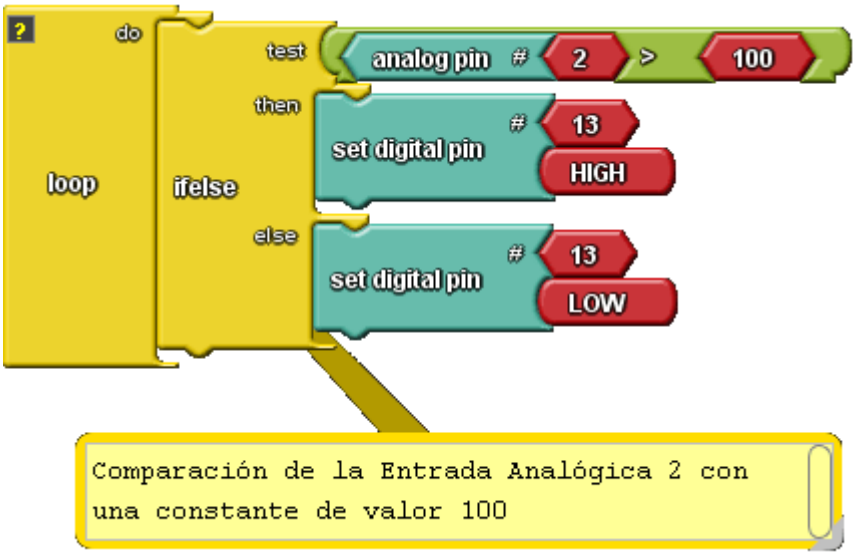
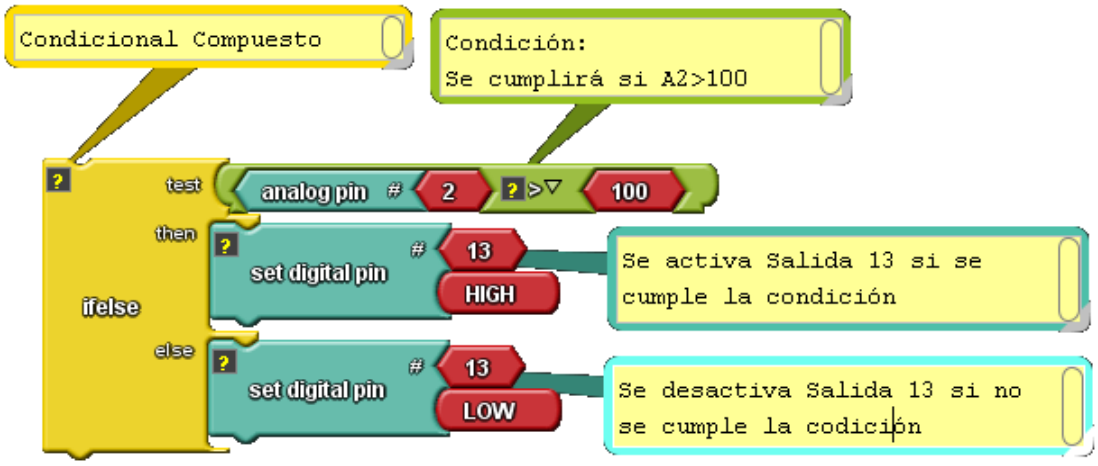


Diagrama realizado con Ardublock

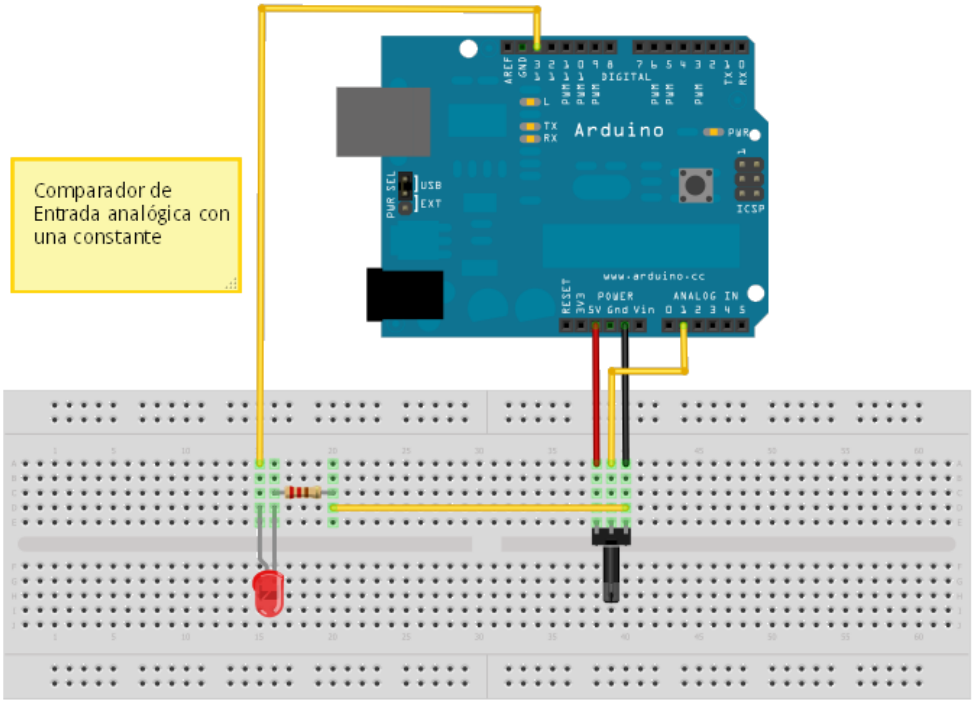
```
void setup()
{
pinMode( 13 , OUTPUT);
}

void loop()
{
if (( ( analogRead(A2) ) > ( 100 ) ))
{
digitalWrite( 13 , HIGH );
}
else
{
digitalWrite( 13 , LOW );
}
}
```

Código generado para el IDE Arduino

En este ejemplo hemos utilizado un nuevo bloque de función **"Ifelse"**



A continuación se muestra el montaje de la aplicación.



7. Generador de Impulsos en el PIN 13 de Arduino

En esta aplicación se trata de generar por una salida de la tarjeta Arduino un número determinado de impulsos en este caso serán 20. La orden para realizar el envío de los impulsos se dará mediante la **Entrada 10** (un pulsador)

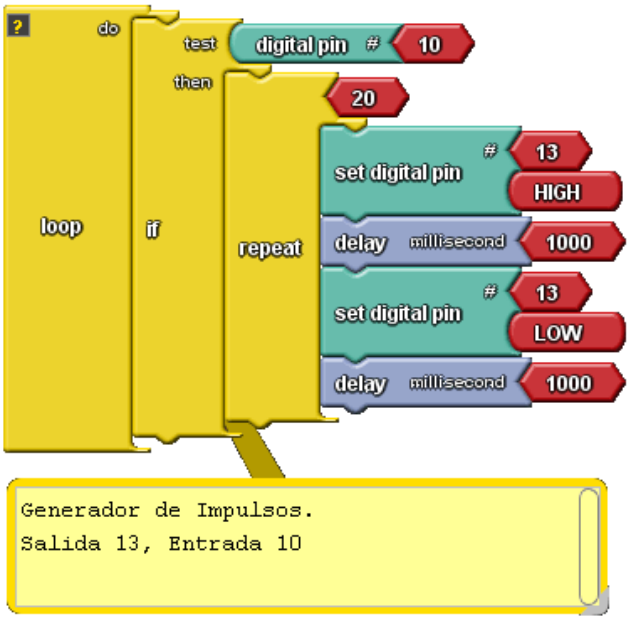


Diagrama de la aplicación

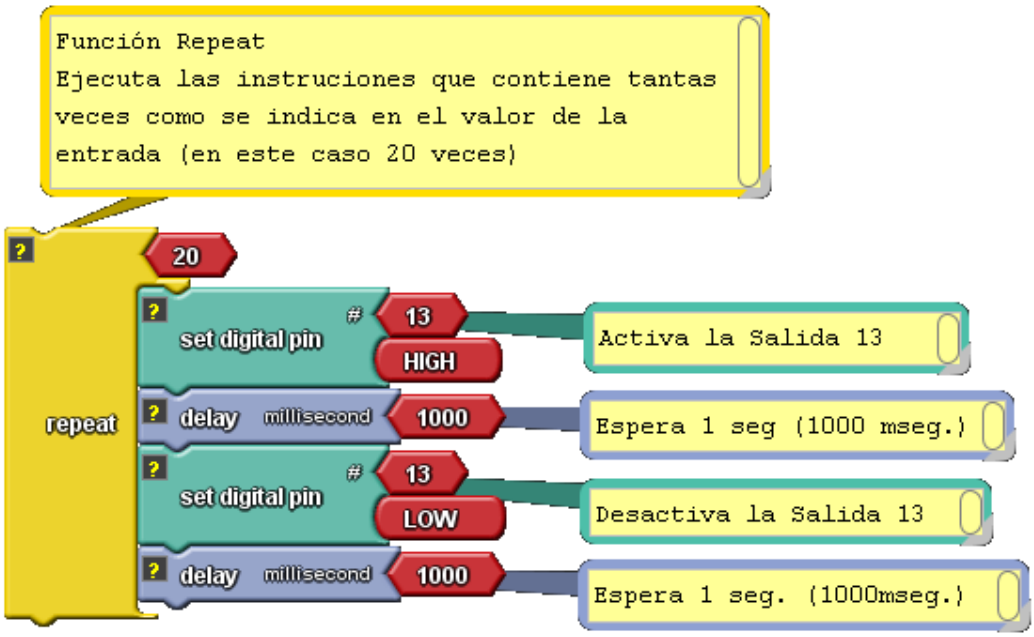
Los impulsos se darán a través de la **Salida 13** (PIN 13 digital de Arduino Uno)

La duración del impulso será 1 seg activado y 1 seg desactivado.

```
int _ABVAR_1_;

void setup()
{
pinMode( 13 , OUTPUT);
pinMode( 10 , INPUT);
}

void loop()
{
if (digitalRead( 10))
{
for (_ABVAR_1_=0; _ABVAR_1_< ( 20 ); ++_ABVAR_1_ )
{
digitalWrite( 13 , HIGH );
delay( 1000 );
digitalWrite( 13 , LOW );
delay( 1000 );
}

}
}
```

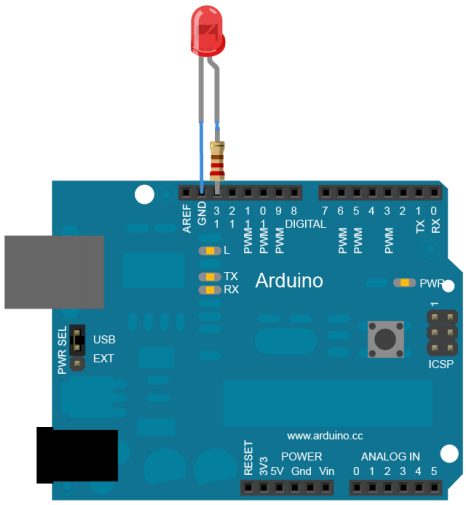
Código para el IDE Arduino

La función **“repeat”** ejecuta el código que contiene tantas veces como se indica en su parámetro de entrada



Función repeat

Montaje a realiza con la tarjeta Arduino Uno

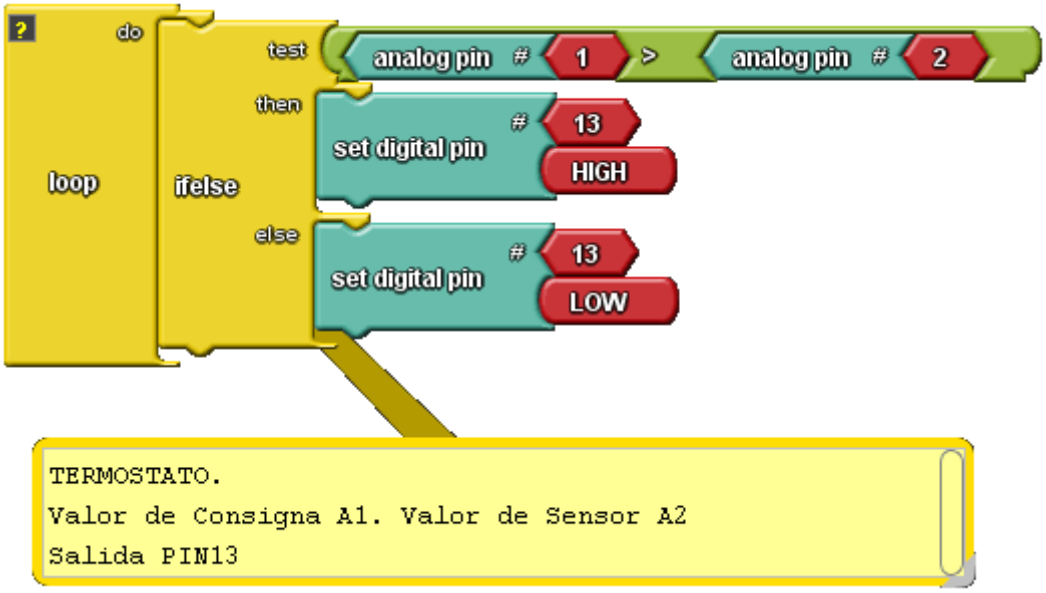


8. Termostato

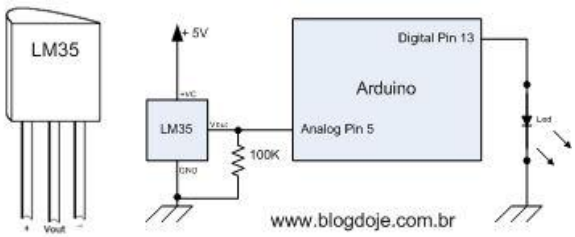
Con este ejemplo se pretende controlar el encendido de la calefacción de una casa teniendo en cuenta la temperatura a la que queremos que este la casa (temperatura de consigna) y el valor de la temperatura ambiente.

- El valor de consigna se introduce en el sistema a través de la entrada analógica **A1.**
- El valor del sensor de ambiente se introduce a través de la entrada analógica **A2.**
- La salida que gobierna el calefactor será la correspondiente al **PIN 13**

El algoritmo es muy sencillo, basta que incluyamos un bloque condicional en el que se pregunte si el valor de la *tconsigna* > *tambiente* si se cumple la condición se activara el calefactor y en caso contrario se desactivara.



Para medir la temperatura se puede utilizar un sensor como el que se muestra a continuación.



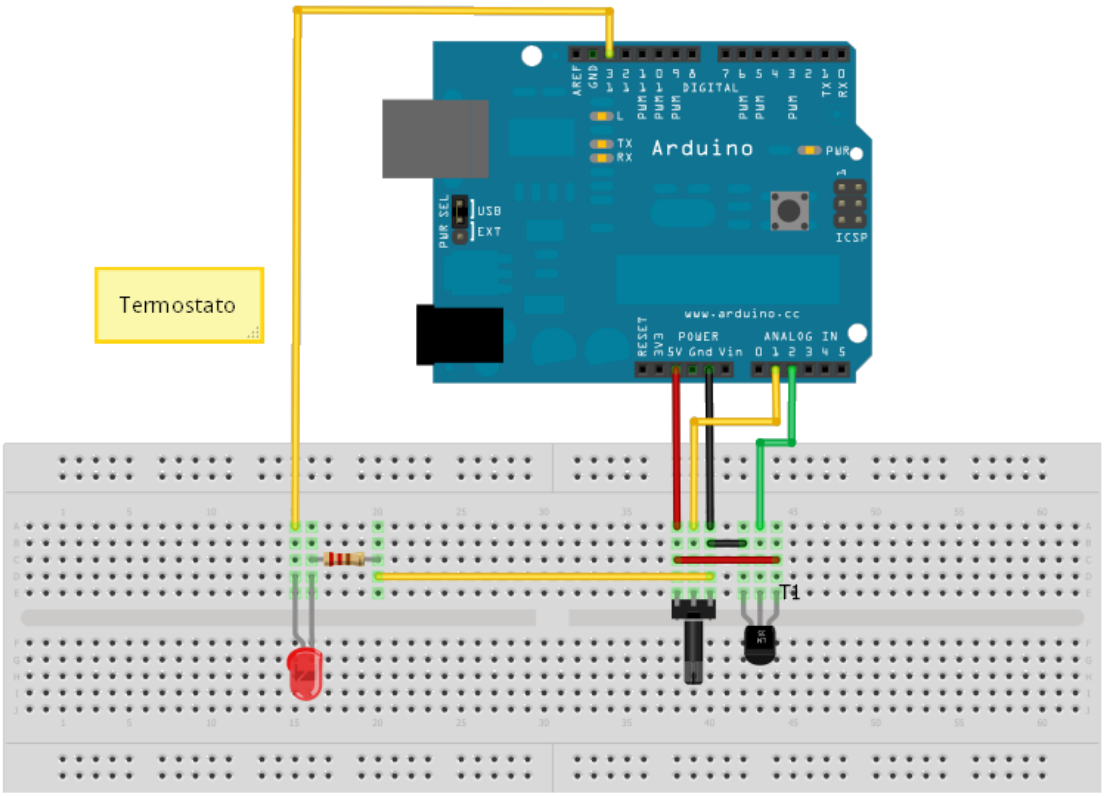
Sensor de temperatura

A continuación se muestra el código correspondiente generado.

```
void setup()
{
pinMode( 13 , OUTPUT);
}

void loop()
{
if (( ( analogRead(A1) ) > ( analogRead(A2) ) ))
{
digitalWrite( 13 , HIGH );
}
else
{
digitalWrite( 13 , LOW );
}
}
```

Código para el IDE Arduino

La imagen siguiente corresponde al montaje de la aplicación con Fritzing



9. Semáforo

Se trata de realizar un semáforo que gobierne tres salidas en forma de diodos led (rojo, ámbar y verde)

Señales de salida:

rojo: PIN 13
ambar: PIN 12
verde: PIN 11

Parámetros:

tiempo_rojo=1 seg. Tiempo_ambar=1 seg. Tiempo_verde=1seg.

El algoritmo es muy sencillo. Se trata de activar las señales correspondientes a las tres lámparas del semáforo con intervalos de tiempo tipo **"delay"**.

El código generado para el IDE Arduino es:

```
void setup()
{
pinMode( 11 , OUTPUT);
pinMode( 12 , OUTPUT);
pinMode( 13 , OUTPUT);
}

void loop()
{
digitalWrite( 13 , HIGH );
digitalWrite( 12 , LOW );
digitalWrite( 11 , LOW );
delay( 1000 );
digitalWrite( 13 , LOW );
digitalWrite( 12 , HIGH );
digitalWrite( 11 , LOW );
delay( 1000 );
digitalWrite( 13 , LOW );
digitalWrite( 12 , LOW );
digitalWrite( 11 , HIGH );
delay( 1000 );
}
```

A continuación se diagrama de bloques para el algoritmo.

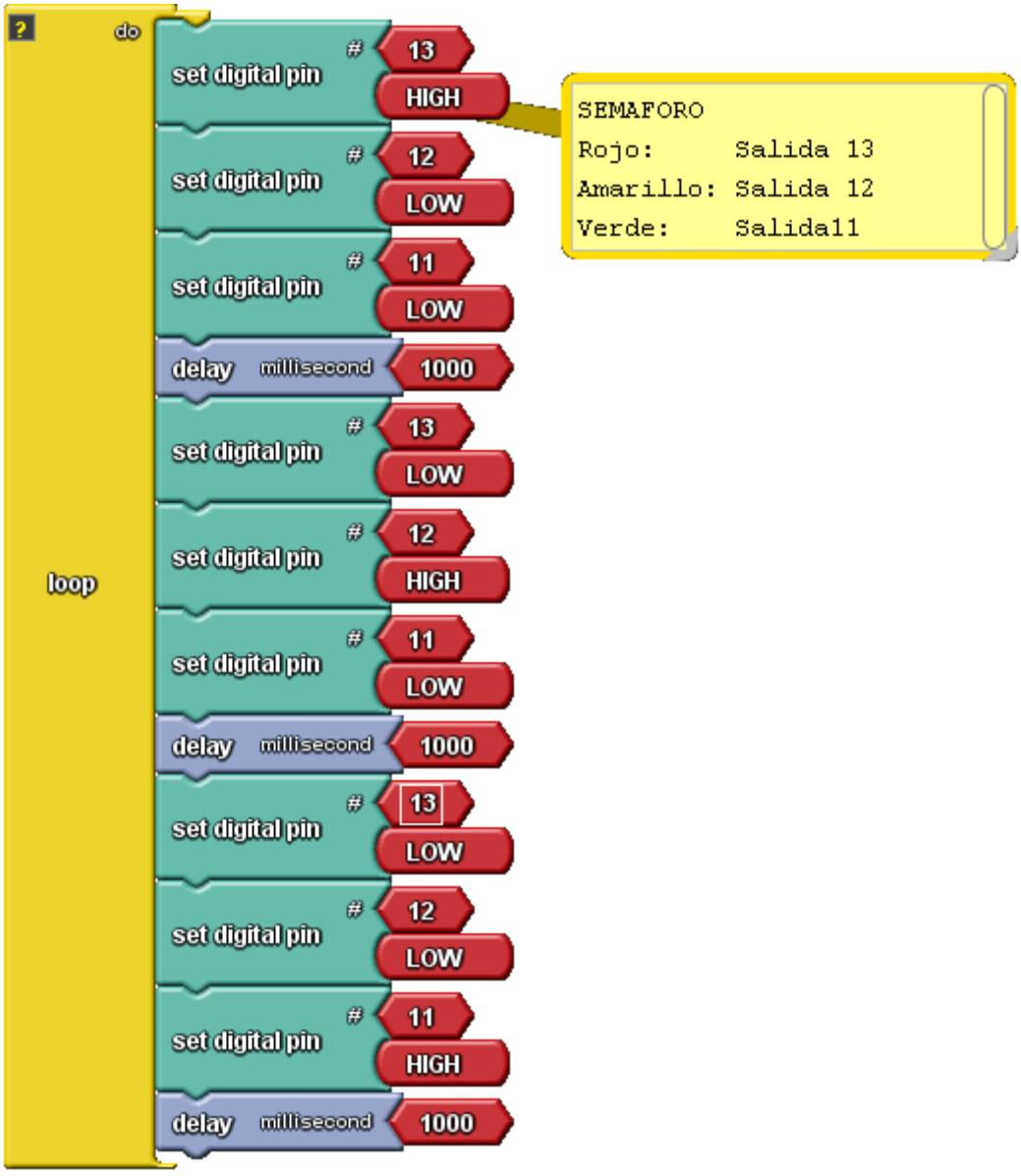
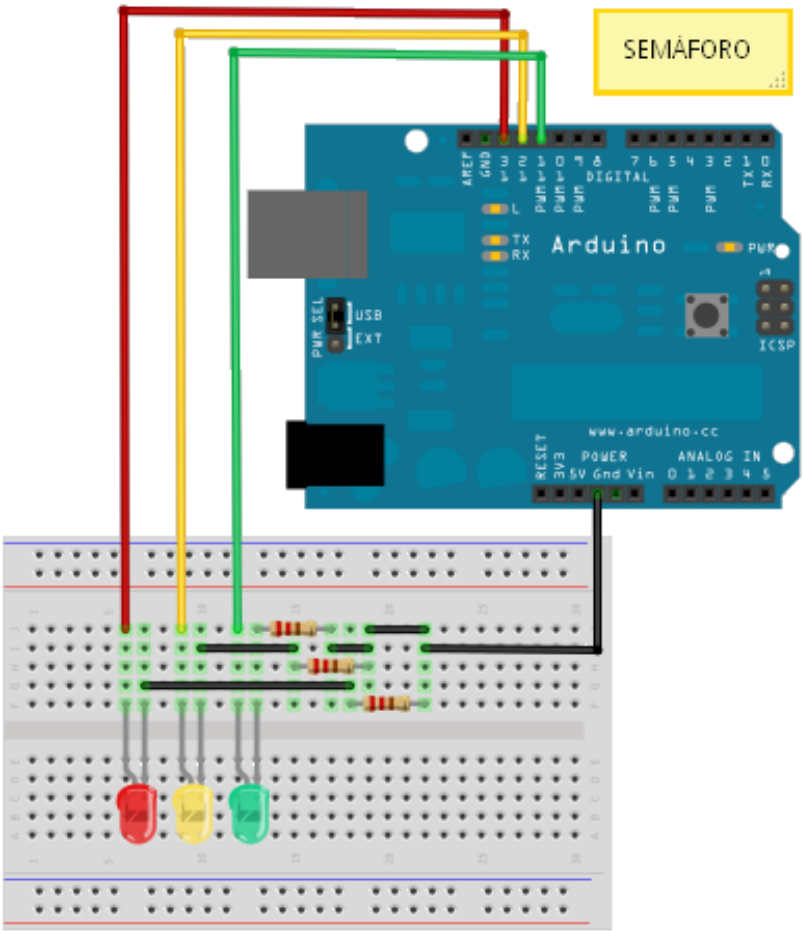


Diagrama realizado con Ardublock

En la siguiente figura se muestra el montaje de la aplicación en protoboard.



Montaje sobre protoboard

10. Confort

Con esta práctica nos introducimos en el mundo de la Domótica. Se trata de poder controlar la activación de tres lámparas en un dormitorio en función de la cantidad de luz que midamos mediante un sensor de luz y por otro lado controlar el encendido de un radiador eléctrico también haciendo uso de un sensor, en este caso de temperatura.

Las señales que debemos definir y manejar son las indicadas en la figura siguiente.

Los señales analógicas de entrada son:

stem= Sensor de temperatura (entrada analógica **A1**)

sluz= Sensor de luz (entrada analógica **A2**)

Las señales digitales de salida son:

Lampara1= Salida Digital **PIN 13**

Lampara2=Salida Digital **PIN 12**

Lampara3= Salida Digital **PIN 11**

Ventilador= Salida Digital **PIN 5**

Calefactor: Salida Digital **PIN 4**

El algoritmo de control:

En la figura siguiente se muestra el esquema del algoritmo de control que hay que implementar. Las lámparas se encienden de acuerdo a las siguientes condiciones

Si stemp **A2 < 100** entonces lampara1 se enciende **PIN 13= HIGH**

Si sluz **A1<150** entonces lamapra2 se enciende **PIN 12=HIGH**

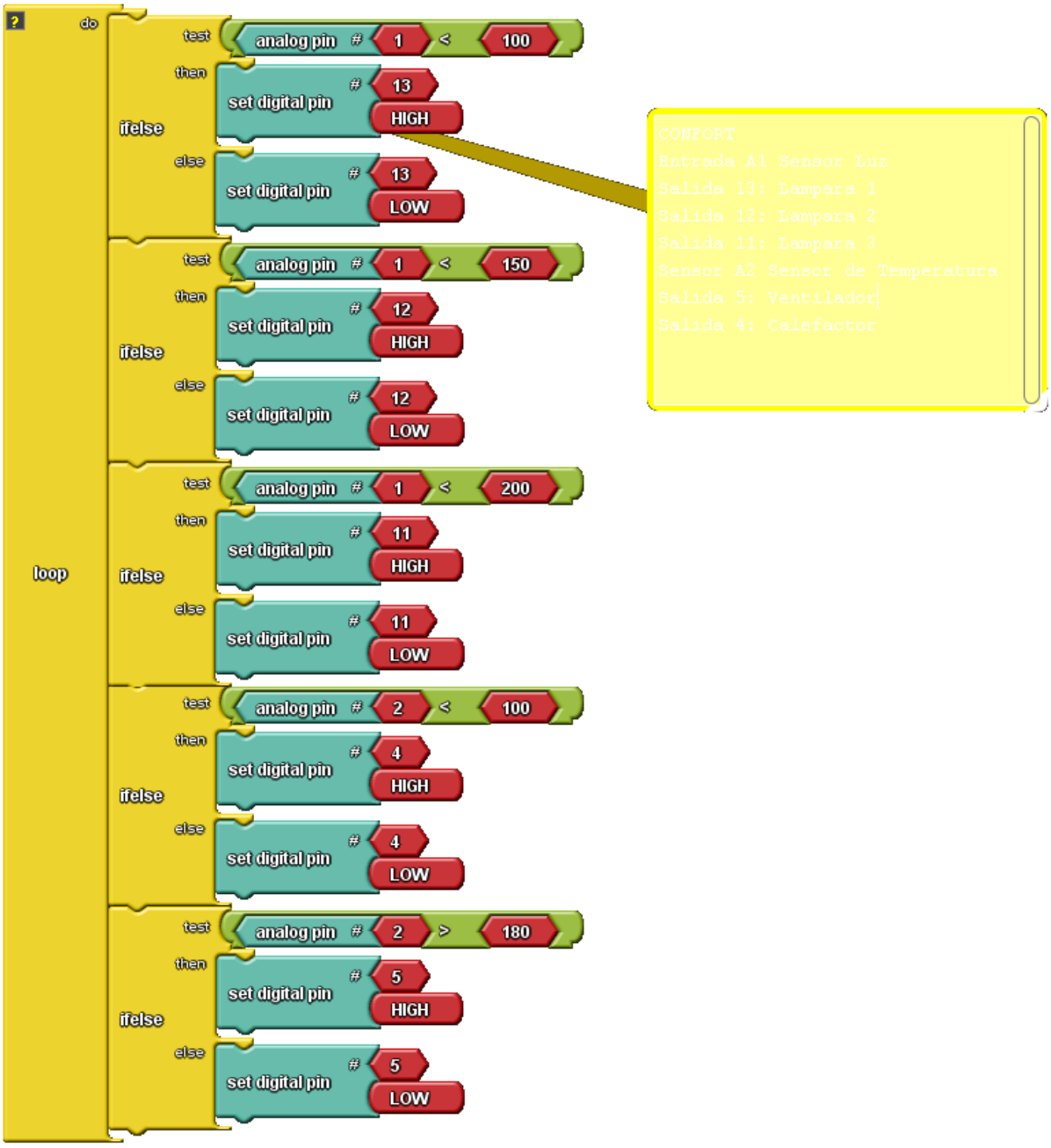
Si sluz **A1<200** entonces lámpara3 se enciende **PIN 11=HIGH**

Para el control de la calefacción y el ventilador se han utilizado de nuevo bloques de función **ifelse.** Las condiciones son:

Si stemp **A2 < 100** entonces calefactor se enciende **PIN 4= HIGH**

Si stemp **A2 > 180** entonces ventilador se enciende **PIN 5=HIGH**

El Algoritmo a realizar con Ardublock es el siguiente:

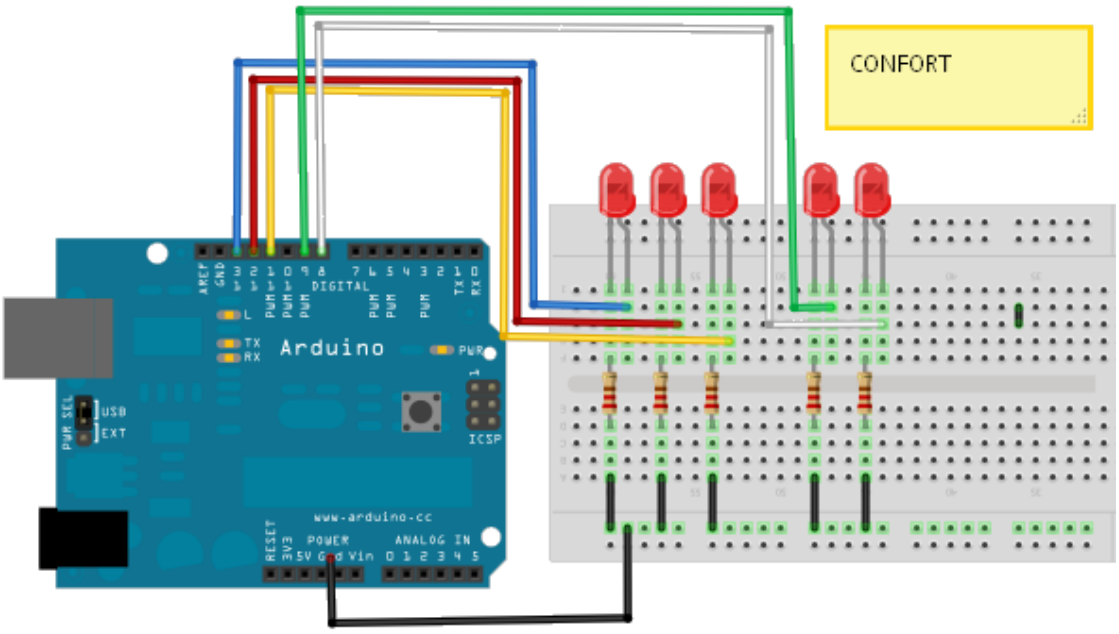


El código generado para el IDE ARDUINO es el siguiente

```
void setup()
{
pinMode( 11 , OUTPUT);
pinMode( 4 , OUTPUT);
pinMode( 12 , OUTPUT);
pinMode( 13 , OUTPUT);
pinMode( 5 , OUTPUT);
}

void loop()
{
if (( ( analogRead(A1) ) < ( 100 ) ))
{
digitalWrite( 13 , HIGH );
}
else
{
digitalWrite( 13 , LOW );
}
if (( ( analogRead(A1) ) < ( 150 ) ))
{
digitalWrite( 12 , HIGH );
}
else
{
digitalWrite( 12 , LOW );
}
if (( ( analogRead(A1) ) < ( 200 ) ))
{
digitalWrite( 11 , HIGH );
```

```
}
else
{
digitalWrite( 11 , LOW );
}
if (( ( analogRead(A2) ) < ( 100 ) ))
{
digitalWrite( 4 , HIGH );
}
else
{
digitalWrite( 4 , LOW );
}
if (( ( analogRead(A2) ) > ( 180 ) ))
{
digitalWrite( 5 , HIGH );
}
else
{
digitalWrite( 5 , LOW );
}
}
```

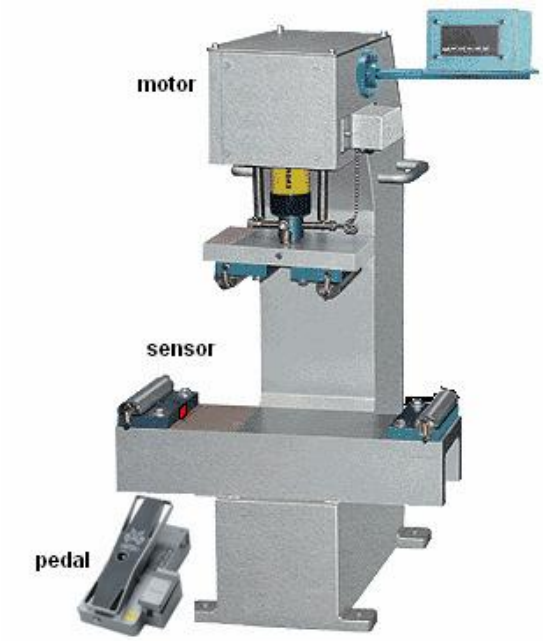
El montaje sobre protoborad es el siguiente



11. Prensa Hidráulica

Disponemos de una prensa hidráulica que se acciona mediante un pedal de tal manera que cuando lo accionamos baja el émbolo de la prensa y permanece bajado durante un tiempo de 0,8 seg. Al cabo del cual sube el cilindro y se vuelve a su posición de reposo para quedar en situación de volver a realizar otra operación de prensado.

La prensa dispone de un sensor en la mesa de tal manera que si el operario tiene la mano sobre esta se interrumpe la barrera del sensor y esta señal impide que baje el cilindro. Al activarse el sensor se encenderá una lámpara roja de alarma.



El motor se gobierna mediante dos señales *"bajamotor"* y *"subemotor"*

Señales a tener en cuenta:

- **bajamotor:** Acciona el motor para que baje el cilindro (digital)(Salida PIN1
- **subemotor**: Acciona el motor para que suba el cilindro (booleana)RLS (Salida PIN 2)
- **pedal:** Orden de actuación al pulsar el pedal(digital) (Entrada PIN 5)
- **sensor :** Sensor de seguridad de la barrera fotoeléctrica (digital) (Entrada PIN 4)
- **alarma:** Señalización de alarma para el caso de que el sensor este activado (digital) (Salida PIN 3)

Funcionamiento

- Cuando se active el pedal la prensa pedal=true, PIN 5=true, deberá bajar e cabezal bajamotor=true, PIN 1= true, siempre y cuando la señal que

llegue del sensor de la mesa sea sensor=false, PIN 4=false, en caso contrario no bajara la prensa. Se dispondrá de un indicador de la señal del sensor que nos pondrá en aviso de que hay una alarma (alarma=true).

- La prensa una vez que llega abajo permanecerá allí 0,8 seg. Para después retornar (subemotor=true, PIN 2=true, y bajamotor=false, PIN 1=false) y de nuevo el sistema vuelve a reposo.

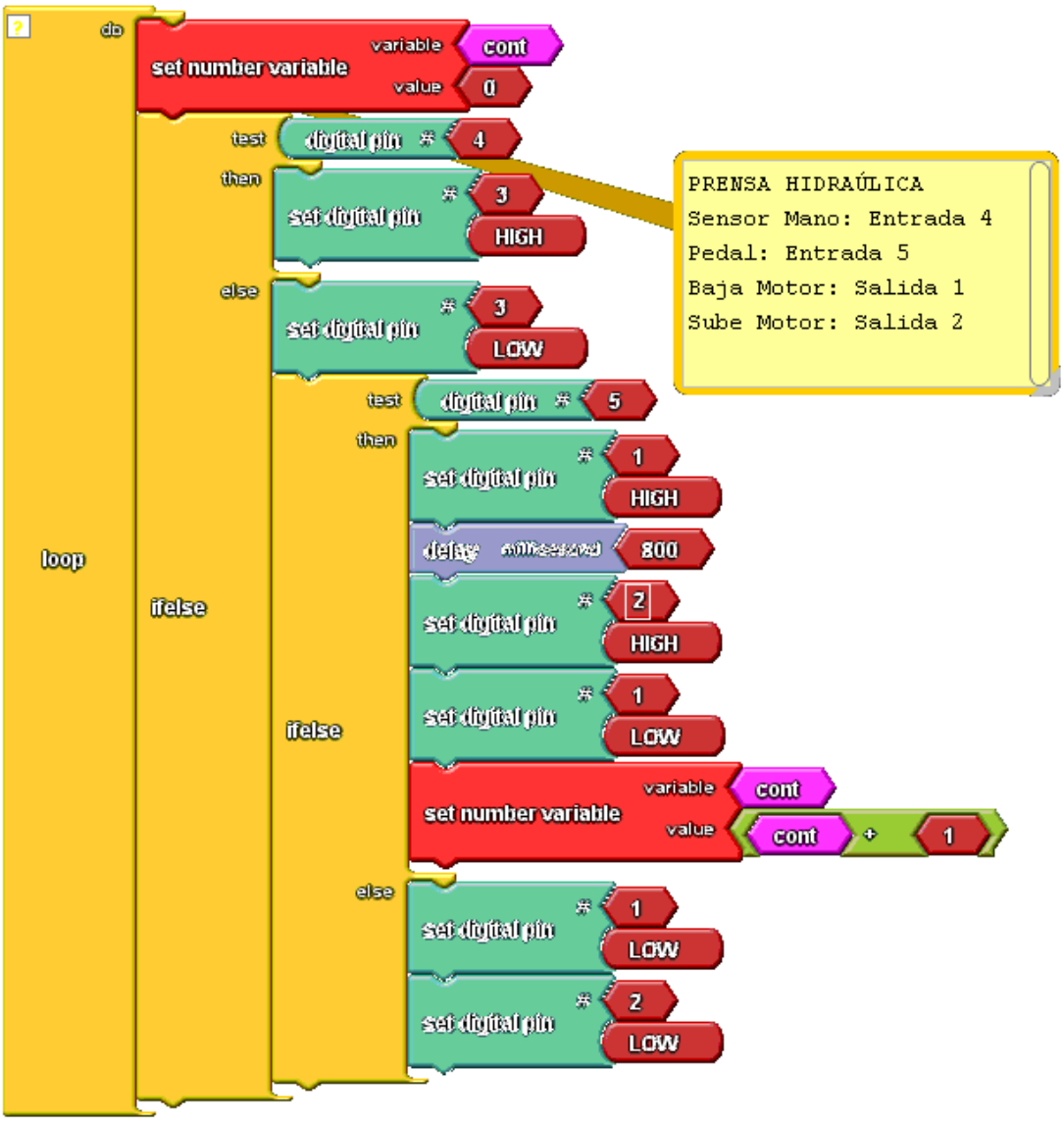


Diagrama grafico creado con Ardublock

A continuación se muestra el código generado para el IDE Arduino

```
int _ABVAR_1_cont;

void setup()
{
pinMode( 3 , OUTPUT);
pinMode( 2 , OUTPUT);
_ABVAR_1_cont = 0;
pinMode( 1 , OUTPUT);
pinMode( 4 , INPUT);
pinMode( 5 , INPUT);
}

void loop()
{
_ABVAR_1_cont = 0 ;
if (digitalRead( 4))
{
digitalWrite( 3 , HIGH );
}
else
{
digitalWrite( 3 , LOW );
if (digitalRead( 5))
{
digitalWrite( 1 , HIGH );
delay( 800 );
digitalWrite( 2 , HIGH );
digitalWrite( 1 , LOW );
_ABVAR_1_cont = ( _ABVAR_1_cont + 1 ) ;
}

 }
 else
 {
 digitalWrite( 1 , LOW );
 digitalWrite( 2 , LOW );
 }
 }
 }
```

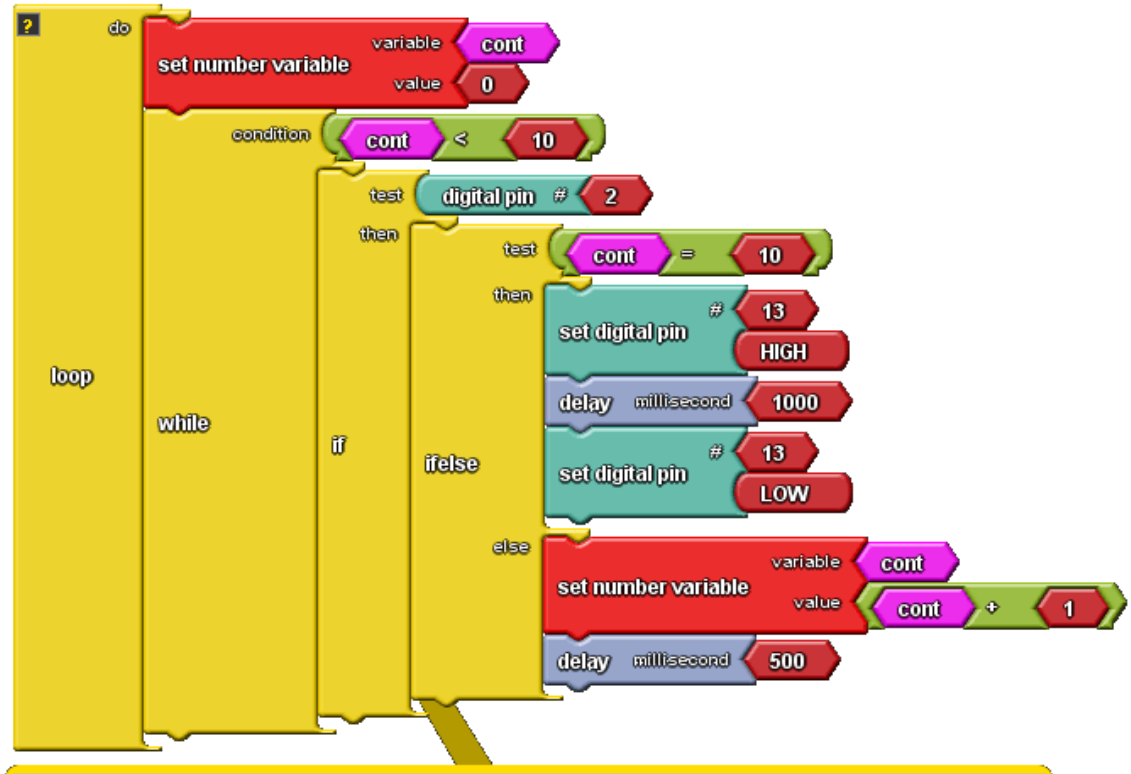
12. Contador de impulsos de entrada

Se trata de contar hasta 10 impulsos que procedan de una de las entradas **PIN 2** de Arduino

Se definirán una variable: **cont** que recoge el valor del número de impulsos que entran por la entrada digital habilitada y los va contando hasta llegar a 10.

Una vez que se llega a 10 impulsos se deberá activar una salida **PIN 13** durante un tiempo de **1 seg**. Y de nuevo se volverá a iniciar la cuenta de impulsos.

El algoritmo es muy sencillo. Se trata de recoger la variable de entrada (estado de la entrada sensor) estableciendo la condición de que **cont=10**. Como acción del **then** se activara la salida **PIN 13** y en caso contrario se incrementa la variable **cont=cont+1.**




```
CONTADOR DE IMPULSOS DE ENTRADA
Cuenta hsta 10 impulosos en la Entrada 2
Se activa durante 1seg.la Salida 13 y se vuelve a repetir la cuenta
La lectura de la entrad se retarda 0.5 seg. para evitar rebotes
```

El código generado para el IDE Arduino es:

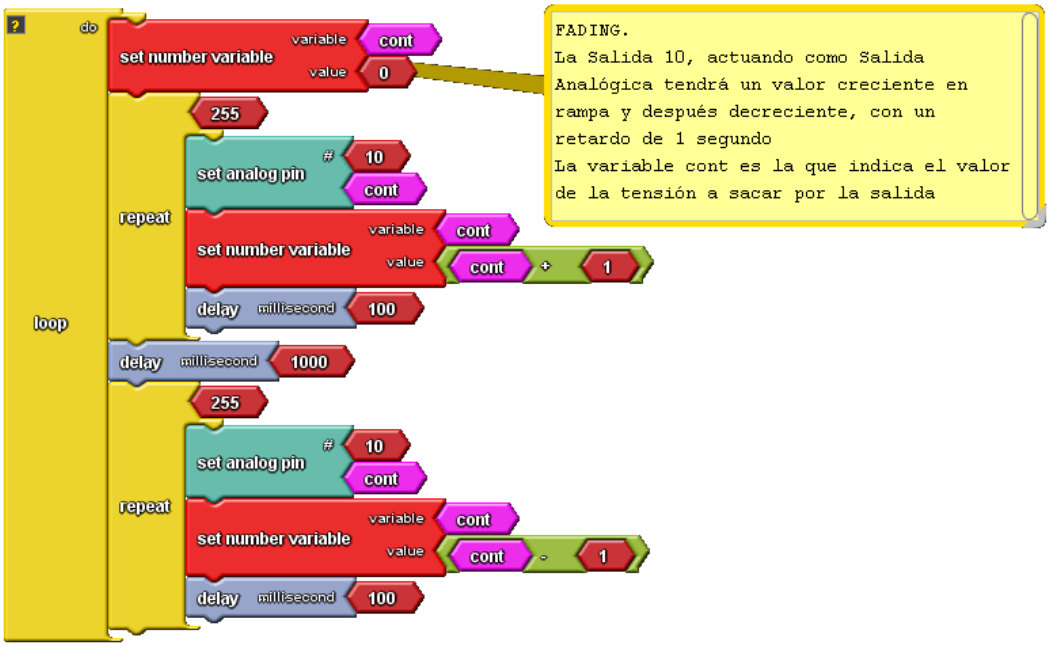
```
int _ABVAR_1_cont;

void setup()
{
pinMode( 2 , INPUT);
_ABVAR_1_cont = 0;
pinMode( 13 , OUTPUT);
}

void loop()
{
_ABVAR_1_cont = 0 ;
while ( ( ( _ABVAR_1_cont ) < ( 10 ) ) )
{
if (digitalRead( 2))
{
if (( ( _ABVAR_1_cont ) == ( 10 ) ))
{
digitalWrite( 13 , HIGH );
delay( 1000 );
digitalWrite( 13 , LOW );
}
else
{
_ABVAR_1_cont = ( _ABVAR_1_cont + 1 ) ;
delay( 500 );
}
}
}
}
```

13. Encendido y apagado progresivo de un led

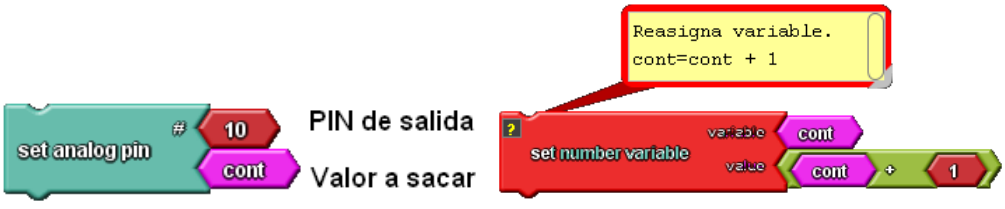
Se trata de enviar valores de 0 a 255 a una de las salidas **PIN10** primero en sentido ascendente y después de 1 segundo en sentido descendente.

Se define una variable que llamaremos cont y que será su valor el que utilicemos para enviar a una salida, en este caso considerada analógica, el correspondiente valor

En el gráfico de la figura vemos el algoritmo empleado e implementado con Ardublock.



Se utiliza la función **set analog pin** que es la que saca el valor al pin de salida. Se ha recurrido al bloque de función **repeat** que ya hemos utilizado en otro ejemplo.



El incremento o decremento de la variable **cont** se realiza haciendo uso del bloque **set number variable** que reasigna el valor cont **cont=cont+1.** Los valores se enviaran con un retardo de 0,1 seg.

El código para el IDE Arduino generado es el siguiente:

```
int _ABVAR_1_cont;
int _ABVAR_2_;
int _ABVAR_3_;

void setup()
{
_ABVAR_1_cont = 0;
}

void loop()
{
_ABVAR_1_cont = 0 ;
for (_ABVAR_2_=0; _ABVAR_2_< ( 255 ); ++_ABVAR_2_ )
{
analogWrite(10, _ABVAR_1_cont);
_ABVAR_1_cont = ( _ABVAR_1_cont + 1 ) ;
delay( 100 );
}

delay( 1000 );
for (_ABVAR_3_=0; _ABVAR_3_< ( 255 ); ++_ABVAR_3_ )
{
analogWrite(10, _ABVAR_1_cont);
_ABVAR_1_cont = ( _ABVAR_1_cont - 1 ) ;
delay( 100 );
}

}
```

14. Gobierno de un motor con cuatro velocidades.

Se trata de gobernar un motor de bajo consumo en cc. Que se conectara a una de las salidas analógicas de la tarjeta Arduino.

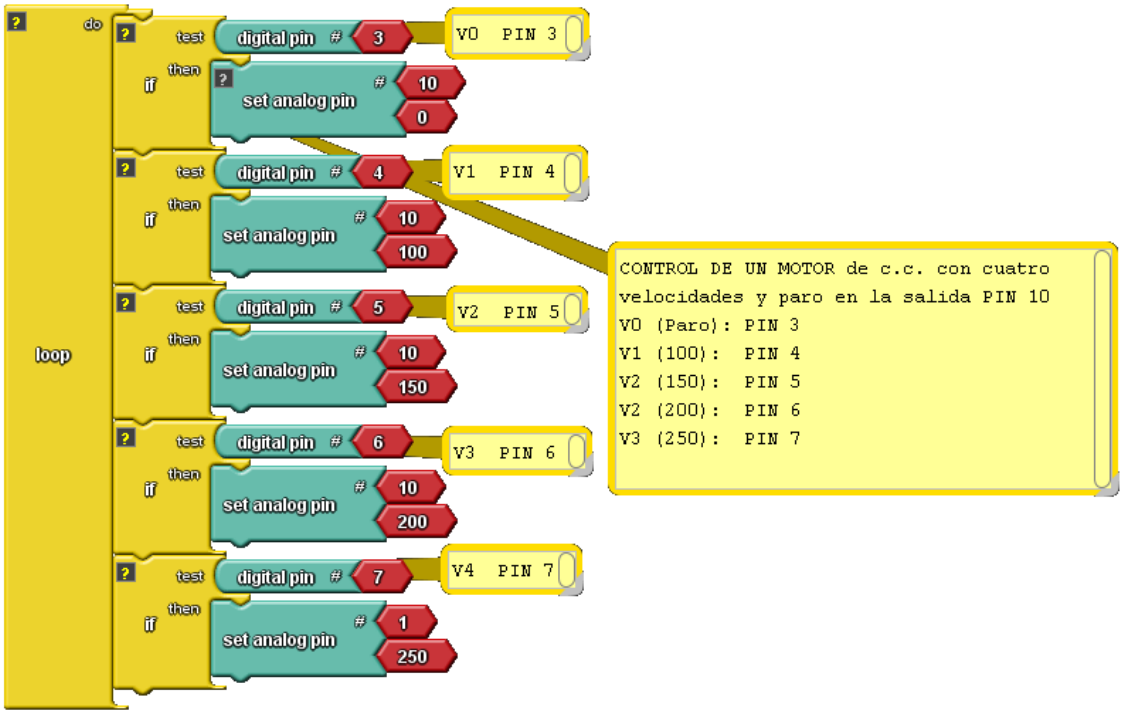
Se establece que la salida será el **PIN 10** y las velocidades serán las siguientes:

V0 equivalente a un valor de 0 (motor parado)
V1 equivalente a un valor de 100
V2 equivalente a un valor de 150
V3 equivalente a un valor de 200
V3 equivalente a un valor de 250

Las velocidades se conmutaran mediante 4 pulsadores que se conectaran como entradas un pulsador de paro que detendrá el motor en los pines :

P0 paro V0 **PIN 3**
P1 para V1 **PIN 4**
P2 para V2 **PIN 5**
P3 para V3 **PIN 6**
P4 para V4 **PIN 7**

La interface para la conexión del motor será mediante un transistor.



Código generado para el IDE Arduino

```
void setup()
{
pinMode( 3 , INPUT);
pinMode( 7 , INPUT);
pinMode( 4 , INPUT);
pinMode( 5 , INPUT);
pinMode( 6 , INPUT);
}

void loop()
{
if (digitalRead( 3))
{
analogWrite(10, 0);
}
if (digitalRead( 4))
{
analogWrite(10, 100);
}
if (digitalRead( 5))
{
analogWrite(10, 150);
}
if (digitalRead( 6))
{
analogWrite(10, 200);
}
if (digitalRead( 7))
{
analogWrite(1, 250);
}
}
```