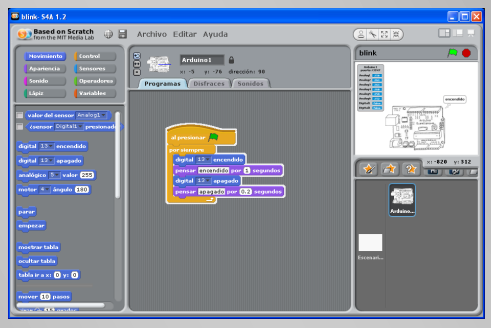
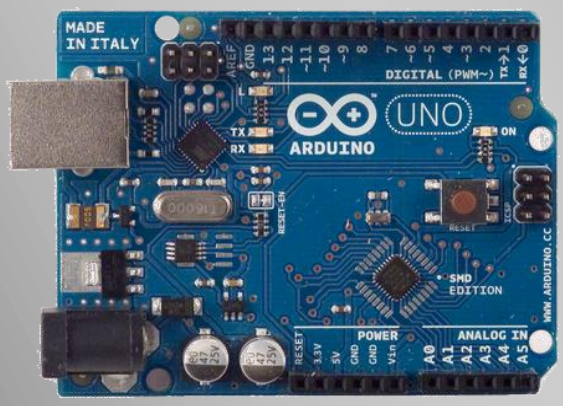
4

S4A (Scratch) + Arduino

Utilización de S4A (Scratch) más la tarjeta Arduino en un ambiente de programación gráfica orientado a la educación



+



José Manuel Ruiz Gutiérrez

Ver. 1.0

Serie: Herramientas Gráficas para la programación de Arduino

Índice

Enero de 2012 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 S4A en conjunción con la tarjeta Open Hardware Arduino UNO.

S4A es un entorno desarrollado a partir del prestigioso entorno Scratch desarrollado en el MIT Media Lab de muy amplia difusión en el mundo de la educación.

La intención de este trabajo es facilitar a los profesores, estudiantes e interesados en la programación grafica y en los entornos Open Hardware una primera iniciación practica abordando la resolución de sencillos ejemplos en los que se hace uso de la librería de objetos que se encargan de la comunicación con Arduino.

Es evidente que ya se terminaron aquellos entornos de programación en los que había que "pelearse" con una sintaxis de comandos en ocasiones diabólica. En este caso S4A realiza esa tarea tediosa y permite al usuario dedicarse a la creación y perfeccionamiento del algoritmo que resolverá su problema.

En este trabajo aporto una colección de ejemplos que permitirán al lector comprender las posibilidades de esta poderosa conjunción Arduino + S4A y le animarán a continuar facilitándole el conocimiento de una de las plataformas Open Hardware más interesantes y difundidas en el mundo.

Poner en la comunidad internacional Arduino este trabajo es para mí una satisfacción porque con ello creo aportar un "pequeño grano de arena" al conocimiento y a su pública y libre difusión a través de herramientas públicas y gratuitas.

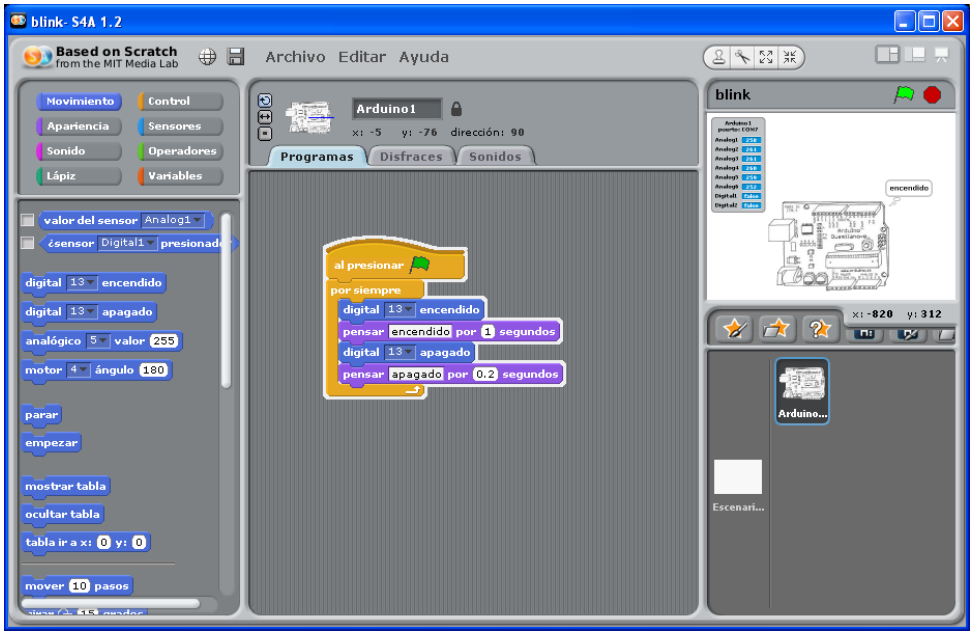
Agradezco sinceramente a todos cuantos, antes de mi, trabajaron en este campo y valoro sus aportaciones, especialmente a los creadores de S4A que han contribuido de una manera notoria a la difusión de Arduino como herramienta de aprendizaje.

2. Una introducción a S4A

Citilab

Introducción

Citilab pone a disposición de la comunidad Arduino una aplicación basada en Scratch para programar de manera gráfica Arduino. La aplicación se llama S4a (ficheros para descargar S4A y Firmware)



S4A es una modificación de Scratch que proporciona una programación sencilla de la plataforma abierta de hardware Arduino. Incluye nuevos bloques para controlar sensores y actuadores conectados a Arduino. También hay una tabla que informa del estado de los sensores similar a la PicoBoard.

Ha sido desarrollada para atraer a la gente al mundo de la programación. Su objetivo es también proporcionar una interfaz de nivel alto para programadores de Arduino con funcionalidades como la interacción de varias placas a través de eventos de usuario.

Características

- Los objetos de la librería Arduino ofrecen bloques para funcionalidades básicas del microcontrolador, escrituras y lecturas tanto analógicas como digitales, además de alguna de más nivel.

Puedes encontrar bloques para controlar servomotores de rotación continua Parallax.

- Se pueden crear objetos Arduino a través de 3 maneras distintas en el entorno de Scratch. Puedes elegir entre crear una nueva conexión o usar una ya existente. Esta característica permite a los objetos virtuales Arduino funcionar de forma colaborativa usando la misma conexión (el objeto físico). El objeto Arduino encontrará él mismo el puerto USB donde la placa esté conectada.

- S4A interactúa con Arduino enviando el estado de los actuadores y recibiendo el de los sensores cada 75 ms, por lo tanto el ancho de pulso ha de ser mayor que este período. Este intercambio de información se efectúa usando el protocolo de la PicoBoard, el cual ya está implementado en un programa específico (llamado firmware) en la placa. Encontrareis instrucciones de cómo cargarlo a través del entorno Arduino.

- Funciona con las versiones Duemilanove/Diecimila y Uno, quizás también con otras pero no las hemos testeado aún. También se puede controlar una placa de manera inalámbrica si se añade un módulo RF como por ejemplo XBee. Una característica importante es que se puede crear un proyecto utilizando tantas placas como puertos USB haya disponibles.

- S4A es compatible con Scratch, se puede trabajar con proyectos Scratch y también con la PicoBoard. Sin embargo, no se pueden compartir proyectos ya que va en contra de los términos de uso de Scratch.

- Por otra parte, la configuración de entrada/salida aún está siendo desarrollada, así que por ahora los componentes tienen que conectarse de una forma concreta. Dicha configuración ofrece 6 entradas analógicas (pines analógicos), 2 entradas digitales (pines digitales 2 y 3), 3 salidas analógicas (pines digitales 5, 6 y 9), 3 salidas digitales (pines 10, 11 y 13) y 4 salidas especiales para conectar servomotores de rotación continua Parallax (pines digitales 4, 7, 8 y 12).

Creadores: S4A ha sido desarrollado por Marina Conde, Víctor Casado, Joan Güell, José García y Jordi Delgado con la ayuda del Grupo de Programación Smalltalk del Citilab. Para informar de errores o simplemente darnos sugerencias podéis escribir a: scratch@citilab.eu.

Modo de instalación.

1. Para instalar S4A +Arduino se deberán seguir los siguientes pasos.
2. Primero instalar el software S4A una vez que se haya descargado del su lugar de origen.
3. Se deberá instalar en la tarjeta Arduino el Firmware correspondiente que facilita la comunicación con S4A. Esto se realiza cargando el fichero firmware en el IDE de Arduino y después descargándolo sobre la tarjeta.
4. Finalmente se ejecuta S4A y de realiza el diseño haciendo uso de las librerías de bloques correspondientes una parte de las cuales se encarga de la lectura y escritura de datos en la tarjeta de acuerdo siempre con la configuración que establezca el firmware. A continuación se detallan estas configuraciones de E/S que no olvidemos que no se pueden modificar desde S4A.

ENTRADAS/SALIDAS

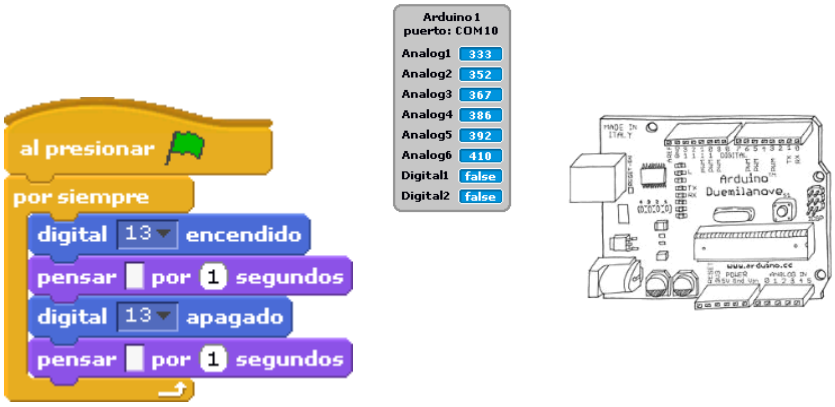
- *salidas digitales (pines digitales 10,11 y 13)*
- *salidas analógicas (pines digitales 5, 6 y 9)*
- *entradas analógicas (todos los pines analógicos de entrada)*
- *entradas digitales (pines digitales 2 y 3)*
- *servomotores RC (pines digitales 4, 7, 8 y 12)*

3. Salida intermitente

Nuestro primer ejercicio será el encendido y apagado de un diodo led conectado en la salida PIN 13 de la Tarjeta Arduino.

En la siguiente imagen se ve el gráfico correspondiente a esta sencilla aplicación.

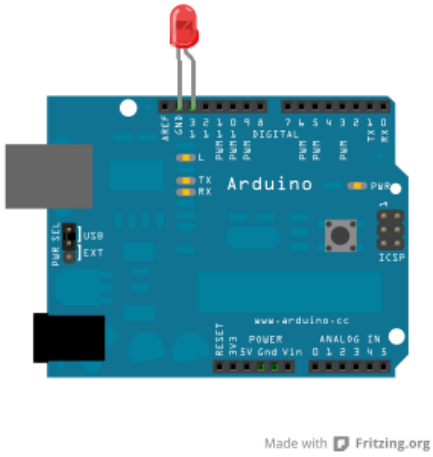
Se han utilizado dos funciones **“digital”** de la librería **“movimiento”** asociadas a la salida **PIN 13** una en estado **“encendido”** y otra en estado **“apagado”**



Las temporizaciones se hacen con los bloques **“pensar”** de la librería **“apariencia”** a los que les hemos quitado el texto que muestran por defecto y en los que se ha colocado el valor del tiempo en segundos, se pueden realizar también con el bloque **“esperar … segundos”** de la librería **“control”**

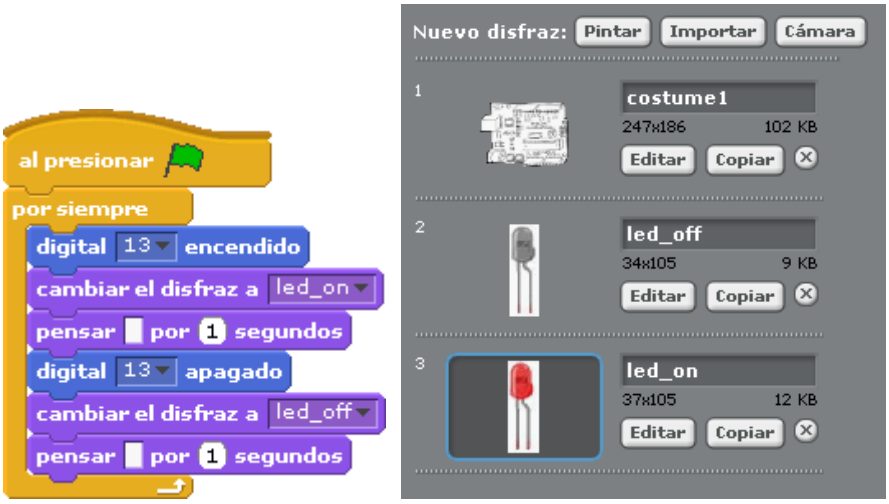
En la pantalla “escenario no se ha puesto ningún objeto, por lo que aparece la que muestra por defecto.

El montaje de este primer ejemplo es muy sencillo y se muestra en la siguiente imagen.



4. Salida intermitente con visualización de estado en pantalla.

Si queremos que en la pantalla del escenario aparezca una imagen de un diodo led que simula encendido y apagado debemos crear con la herramienta **“disfraces”** del entorno dos imágenes una que llamados **led_off** y otra **led_on** que muestran el estado.



Después en el programa lo que hacemos es utilizar el bloque de función **“cambiar el disfraz”** perteneciente a la librería **“apariencia”** que permite mostrar en el escenario una u otra imagen.

Los bloques **“al presionar”** y **“por siempre”** pertenecen a la librería **“control”**

En la siguiente figura se muestra el programa ya elaborado.

La siguiente figura es el aspecto del escenario

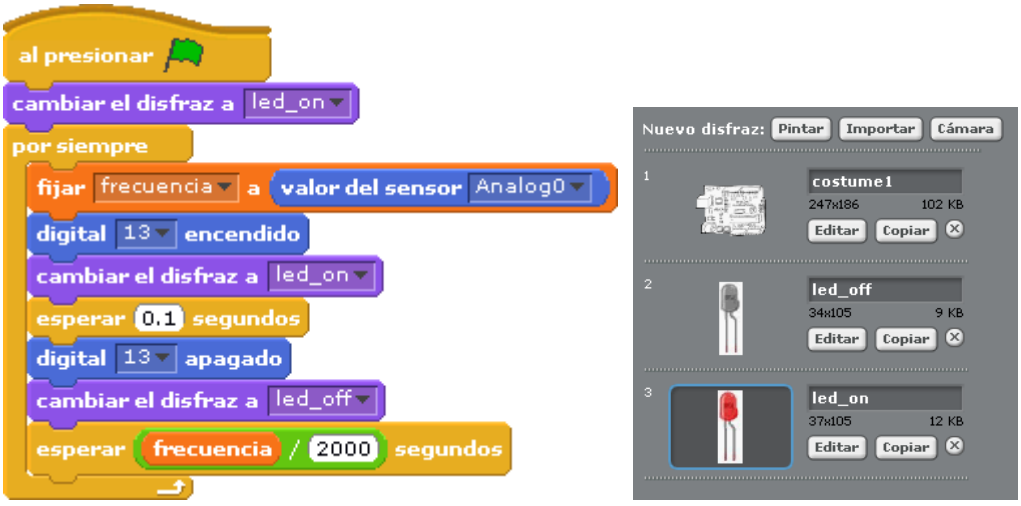
Led_off led_on

5. Salida intermitente con visualización de estado en pantalla y control de frecuencia mediante un canal de entrada analógica.

El siguiente ejemplo nos muestra la forma de poder asociar una variable al valor de una entrada analógica **Analog0** con el fin de poder variar el retardo en el encendido y apagado de una salida digital **PIN 13.**

Comenzamos definiendo la variable **"frecuencia"** que mediante la función **"fijar frecuencia a valor del sensor Analog0"** asociamos a ella el valor leído del canal analógico.

El tiempo de encendido lo fijamos a **0.1 segundo**s y en donde actuamos es en el tiempo de apagado. El tiempo de apagado los configuramos con la función **"esperar frecuencia/2000 segundos"**



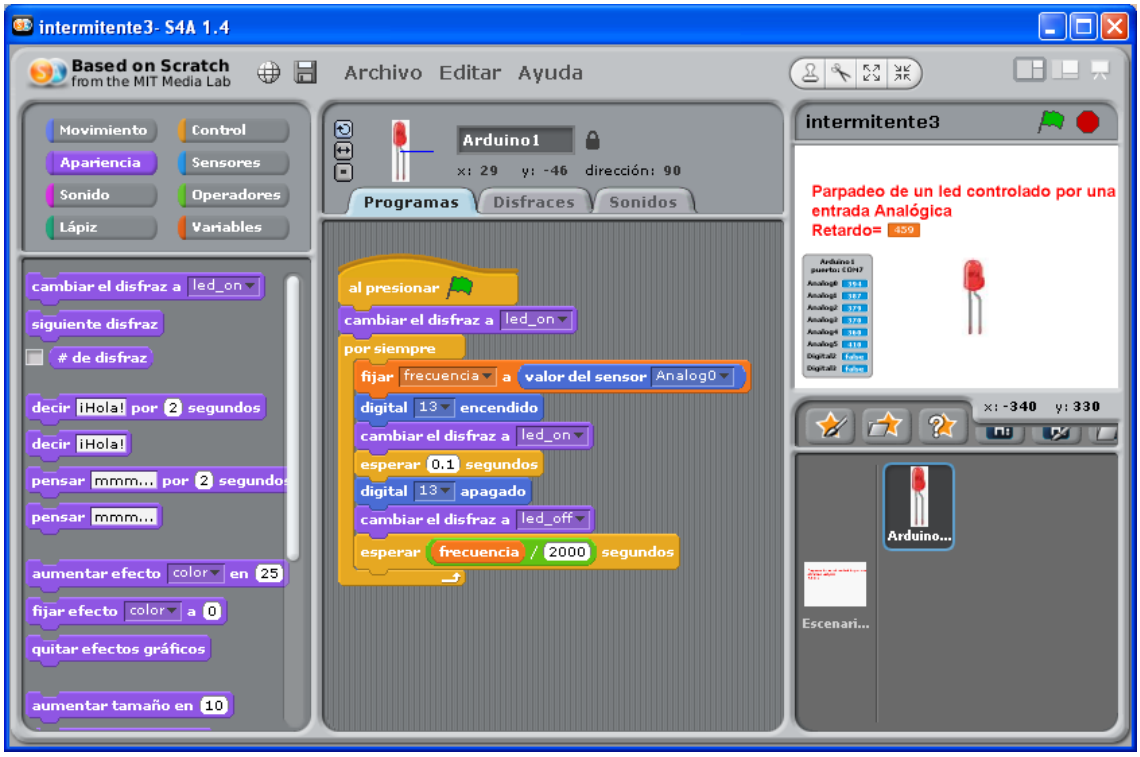
Hemos recurrido a la opción de mostrar en la pantalla "escenario" la imagen de un diodo led encendido y apagado con los disfraces **led_off** y **led_on.** Mostramos estas imágenes mediante la función **"cambiar el disfraz a led_on" y "cambiar el disfraz a led_off".**

Una vez activada la simulación podemos observar como al modificar el valor del potenciómetro varía el tiempo de apagado del led.

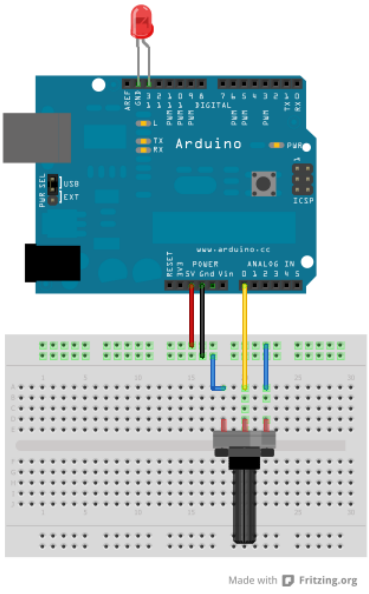
El valor leído del canal **Analog0** se divide por **2000** con el fin de reducir el rango de variación del la variable frecuencia. En este caso podemos deducir que el valor de frecuencia seria:

- **Frecuencia** varía entre **0 y 0,512 seg.**
- **Analog0** varía entre **0 y 1024**

En la figura siguiente vemos el aspecto de la aplicación



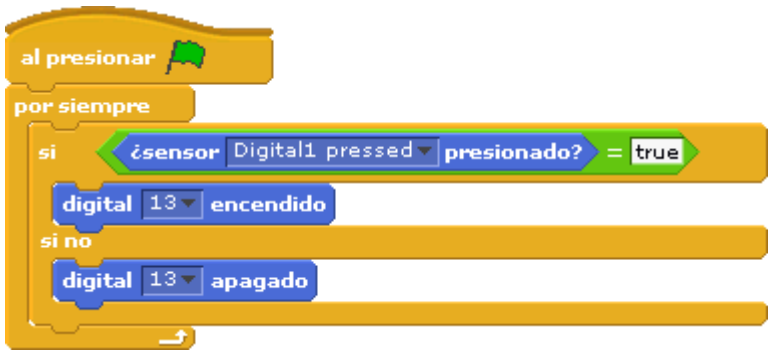
A continuación se muestra el esquema de conexión de los componentes



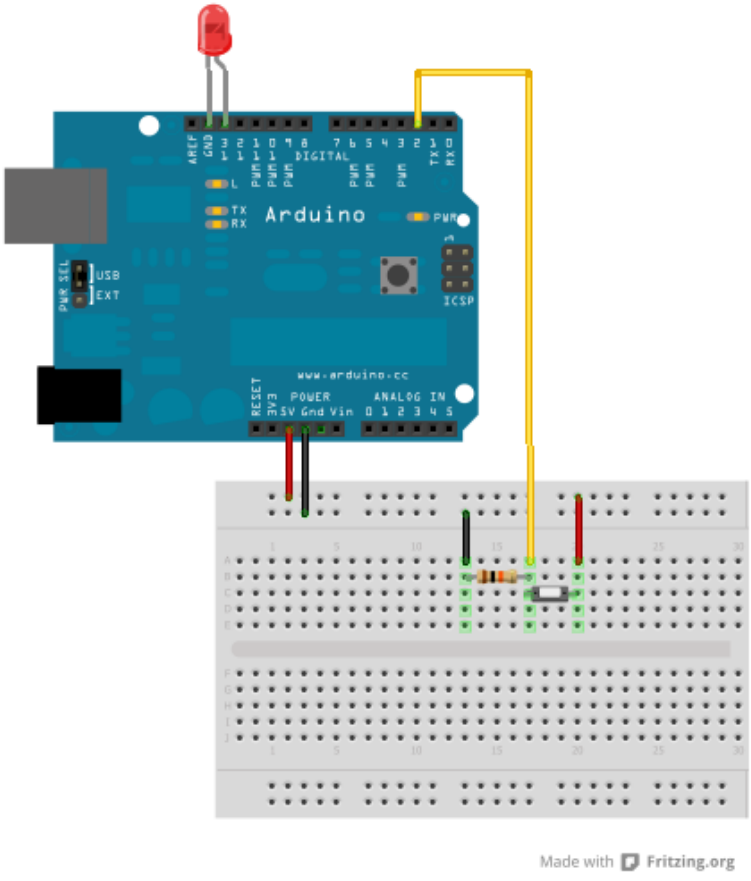
6. Gobierno de una salida mediante un pulsador

Queremos gobernar una salida digital **PIN 13** mediante el accionamiento de una entrada digital **PIN 2** a la que le hemos colocado un pulsador.

El programa es muy sencillo. Mediante el bloque de **"sensor… presionado"** averiguamos si el valor de la entrada es **"true"** o **"false"** y en función de este, mediante el bloque de función **"si… si no"** perteneciente la librería **"control"**, conseguimos la función de gobierno deseada: Activar o desactivar la salida **PIN 13**



El esquema del montaje es el de la siguiente figura.



7. Gobierno de salida con pulsador en pantalla: Pulsador virtual

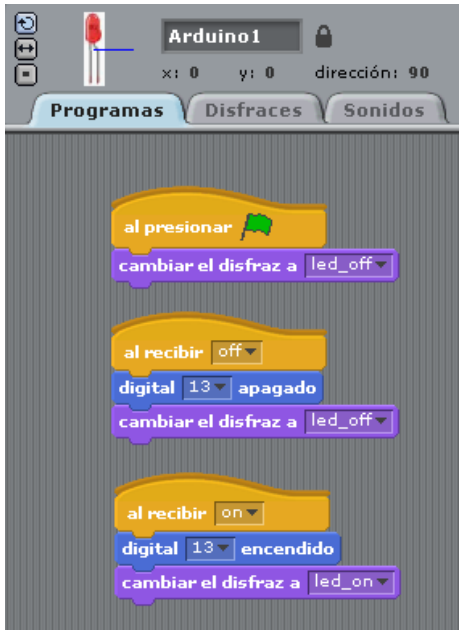
En este caso lo que vamos a añadir a la versión anterior es la visualización en la pantalla escenario de un led y un pulsador, pero en este caso el pulsador será "virtual", es decir, que el diodo led de la salida PIN 13 se activará gobernado desde la pantalla del ordenador.

Se trata de activar desde la pantalla "escenario" un botón para activar el **PIN 13** y otro para desactivarlo.

- La activación y desactivación se lleva a cabo por la acción sobre dos objetos: **led_off** y **led_on** y la activación propiamente dicha es la que se asocia al objeto **Arduino1** que tiene también integrados los objetos **led_on** y **led_off**

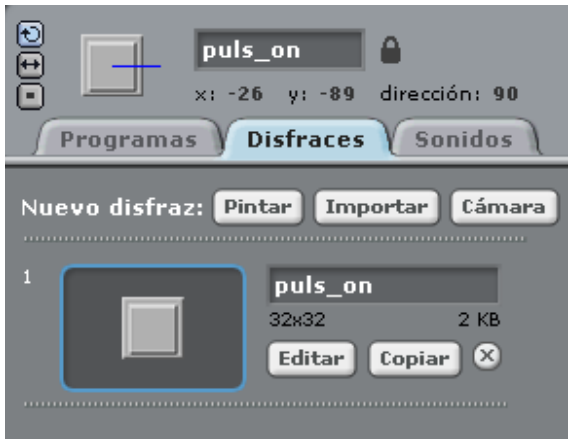
- Cuando se presiona sobre el objeto **botón_on** se envía al **"mensaje" on**, orden **"enviar a todos"** que al recibirse en el script correspondiente al programa **Arduino1** activa la salida **PIN 13** mediante la orden **"digital 13 encendido"**

- Cuando se presiona sobre el objeto **botón_off** se envía al **"mensaje" off**, orden **"enviar a todos"** que al recibirse en el script correspondiente al programa **Arduino1** desactiva la salida **PIN 13** mediante la orden **"digital 13 apagado"**

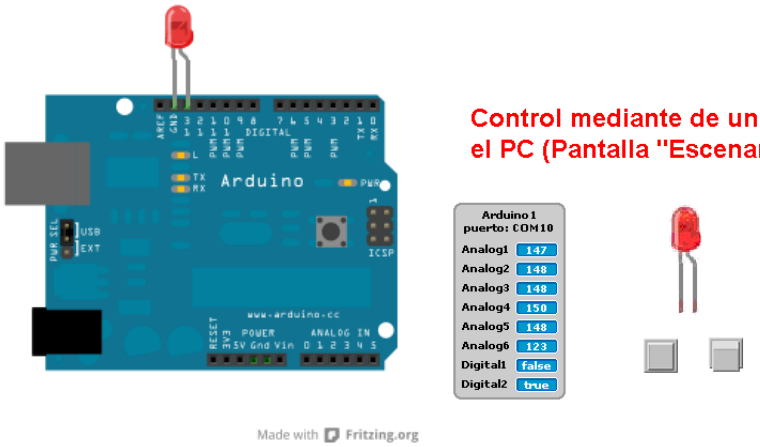


En las figuras vemos los objetos del programa Arduino1 y el programa correspondiente.

Los programas **puls_off** y **puls_on** se muestran a continuación junto a los "disfraces de cada uno"



La pantalla "escenario" de esta aplicación es la que se muestra a continuación, en ella vemos los objetos descritos anteriormente: **led, puls_off** y **puls_on**



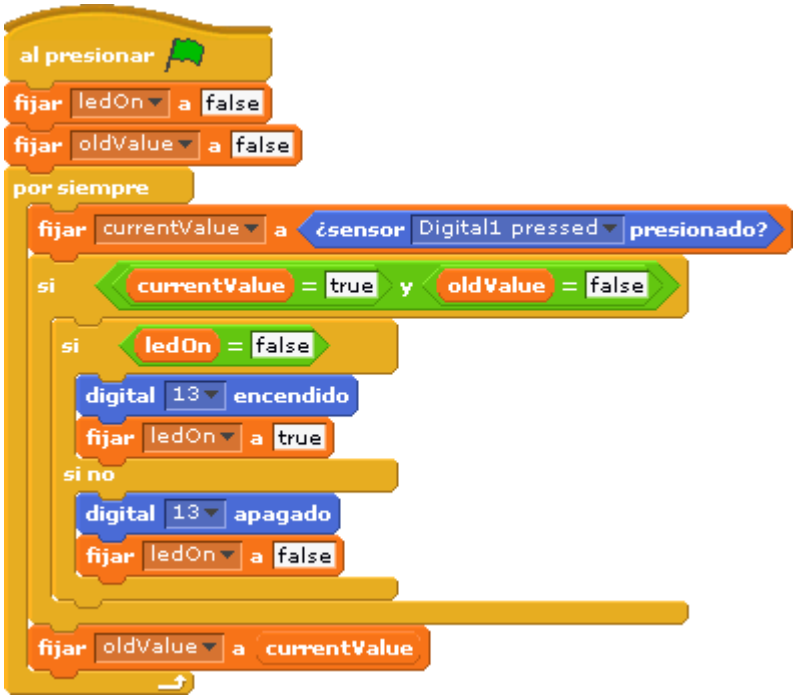
8. Gobierno de salida en modo biestable "memoria"

Ahora vamos a controlar la salida **PIN 13** con un pulsador desde la tarjeta Arduino pero para el gobierno se podrá hacer simplemente pulsando, actuando el sistema como un biestable (una pulsación enciende, la siguiente pulsación apaga y así sucesivamente).

En la figura se muestra el algoritmo.

En este caso, previamente se crean tres variables:

led_on: indica el estado de la salida **PIN 13**
oldValue: Indica el estado anterior de la salida **PIN 13**
currentValue: Indica el estado actual de la salida **PIN 13**

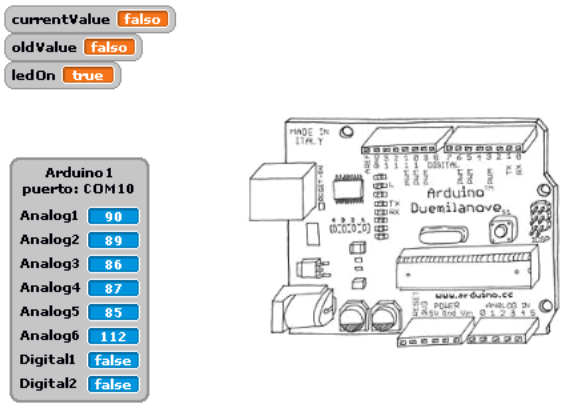


El funcionamiento es el siguiente:

- Se fijan los valores **ledOn** y **OldValue** a **false** ("cero") **"fijar ledOn false"** y **"fijar oldValue false"**
- Se lee el valor de la entrada **Digital1** y se asocia a la variable currentValue
- Se pregunta si el valor **currentValue** es **true** y el **OldValue** es **falso**, si es así entonces se pregunta si el led está apagado **ledOn=false** y si es asi de activa la salida **PIN13** **"digital 13 encendido"** y se fija el

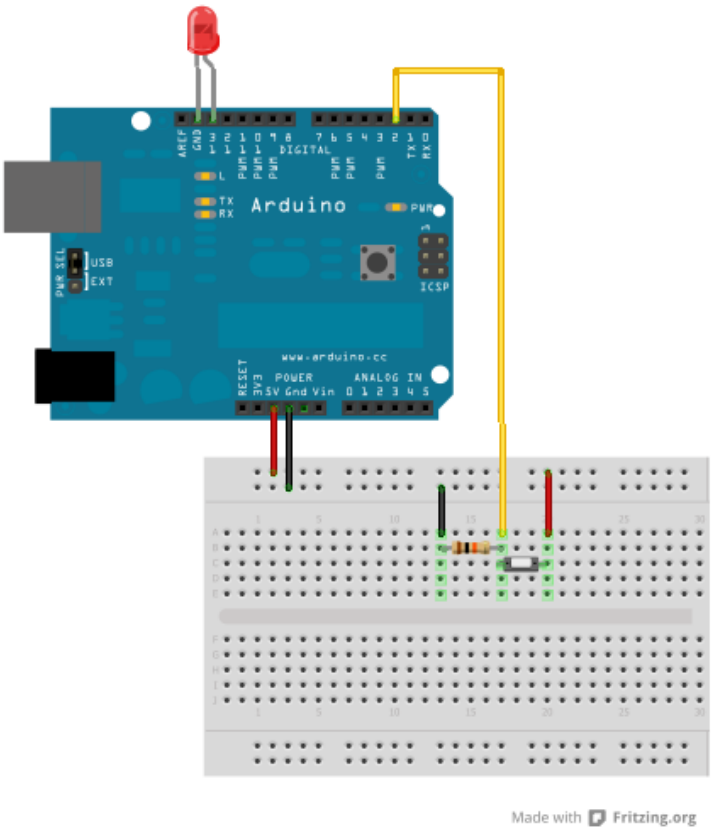
valor de **ledOn** a true **fijar ledOn true”**, si no , se apaga el led **“digital 13 apagado”** y se pasa pone el valor **“fijar ledOn false”**

- Finalmente se fija **oldValue** al mismo valor que **currentValue “fijar oldValue a currentValue”**



Este es el aspecto que presentaría la pantalla “escenario”. Vemos que se muestran los valores de las variables **currentValue, oldValue y ledOn.**

Este sería el montaje del circuito.



9. Control de una salida mediante el teclado

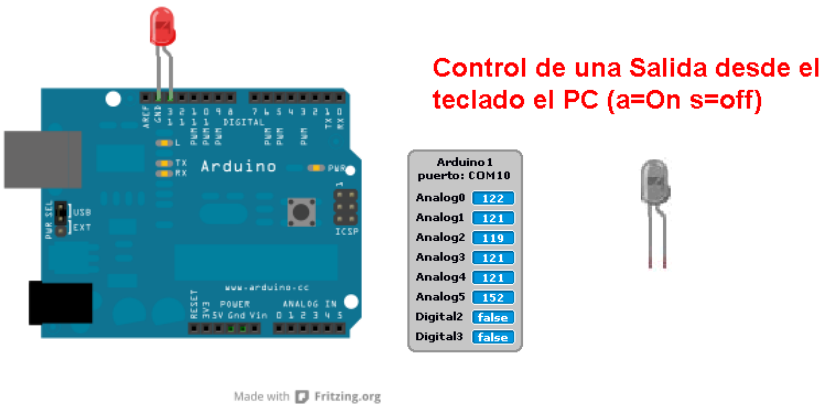
Se trata de gobernar una salida (**PIN 13**) mediante el teclado con la letra **"a"** la activamos y con la letra **"s"** la desactivamos.

El algoritmo es muy sencillo. Se han creado dos **"disfraces"** uno **led_off** y el otro **led_on** que se asociarán al estado de apagado y encendido de la salida. Se ha recurrido al bloque de función **"al presionar tecla"** que se activa cuando se detecta que una tecla del teclado se ha pulsado.

Si se pulsa la tecla **a** la señal se activa **PIN 13 =true**
Si se pulsa la tecla **s** la señal se desactiva **PIN 13=false**

Al comienzo del programa para asegurarnos de que la salida esta apagada ponemos el disfraz de led apagado **"cambiar el disfraz a led_off"**

En la siguiente imagen vemos el estado de la pantalla **"escenario"**



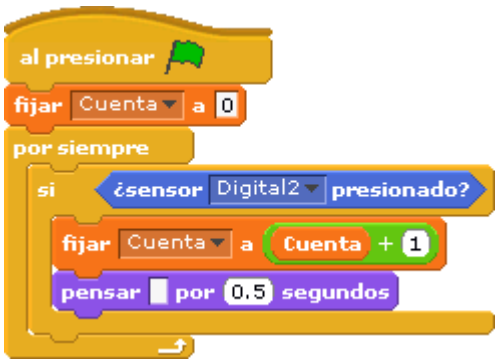
10. Contador Sencillo

Contar es una función muy útil en los sistemas, es por ello por lo que a continuación vamos a estudiar el siguiente ejemplo.

Se trata de contar los impulsos que van entrando por una de las entradas digitales **PIN 2**

Definimos una variable a la que llamaremos **"cuenta"** en ella se acumulará el valor del contador que vamos a implementar. Luego la mostraremos en la pantalla "escenario".

- El programa debe comenzar poniendo a **"0"** el valor de la variable **"cuenta",** lo hacemos con la instrucción **"fijar Cuenta a 0"**

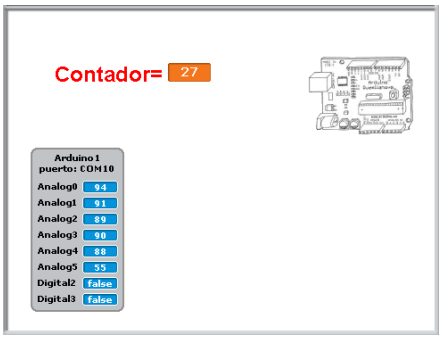


- En el bucle principal del programa **"por siempre"** integremos un bucle cndicional "s" en el que la condición que se testea es si la entrada digital **PIN 2** ha sido activada **"¿sensor Digital2 presionado"**

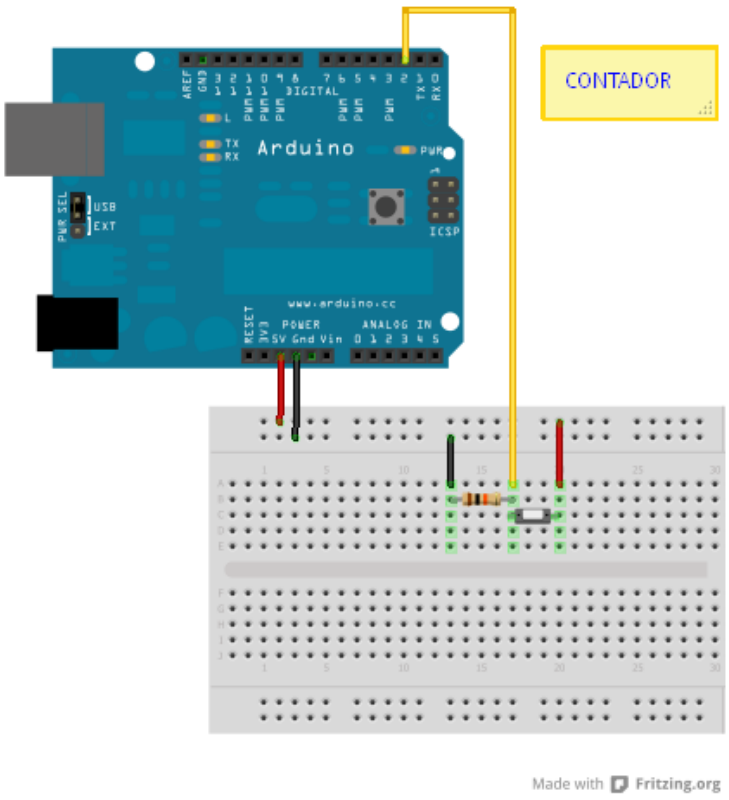
- Si se cumple la condición lo que se hace es incrementar en "1" el valor del contador **"fijar Cuenta a Cuenta+1"**

- Se establece a continuación un retardo de 0.5 seg. Con el fin de evitar que cuente más de uno por el tiempo que este presionado aunque sea muy poco. **"pensar por 0.5 seg"** . También se podría haber realizado el retardo con la función **"esperar… segundos"** de la librería de **Control**

En la siguiente imagen vemos el aspecto de la pantalla "escenario"



Se muestra a continuación el montaje de la aplicación.



11. Contador Adelante/atrás

Esta es una variante del ejercicio anterior en la que deseamos poder contar hacia adelante o hacia atrás haciendo uso de dos entradas digitales **Digilal2** y **Digital3** correspondientes a los pines **PIN 2** y **PIN 3** de la tarjeta Arduino respectivamente.

De la misma manera que hemos hecho en el ejemplo anterior definimos la variable **Cuenta** que almacenará el valor de contador.

Esta vez dispondremos de dos bucles tipo "si" uno para cada una de as dos operaciones **"contar"** y **"descontar"**

Contar:

Para el bucle contar testeamos el estado de la variable de entrada digital **Digital2 PIN 2** y si se cumple que esta activada incrementamos el contador **"fijar Cuenta a Cuenta+1"**

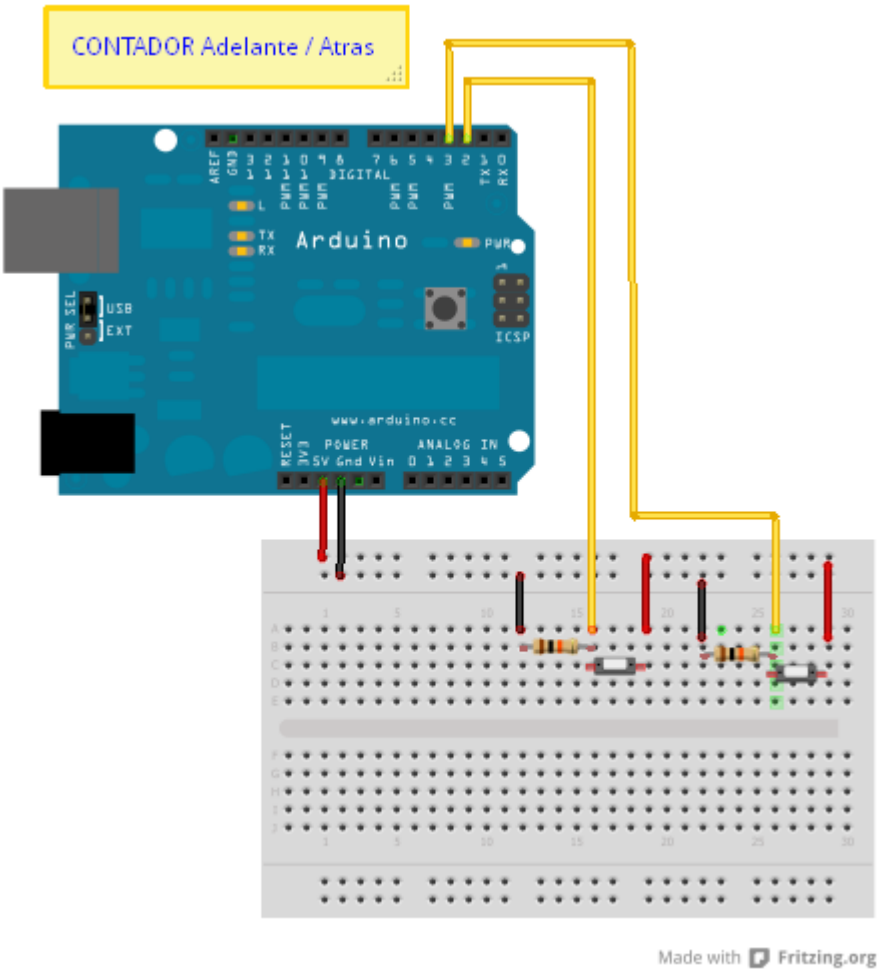
Descontar:

Para el bucle descontar testeamos el estado de la variable de entrada digital **Digital3 PIN 3** y si se cumple que esta activada incrementamos el contador **"fijar Cuenta a Cuenta-1"**

Se han colocado igualmente los retardos para evitar "rebotes" en la cuenta

En la figura siguiente vemos el aspecto de la pantalla "escenario".

Este es el montaje que deberíamos realizar para probar la aplicación.



12. Contador con puesta a “cero”

Esta este ejemplo deseamos poder realizar la “puesta a cero” del valor del contador. Para este montaje dispondremos de dos pulsadores conectados a las entradas digitales **Digilal2** y **Digital3** correspondientes a los pines **PIN 2** y **PIN 3** de la tarjeta Arduino respectivamente.

Digital2 (**PIN 2**)	Será la para la entrada de impulso de cuenta
Digital3 (**PIN 3**)	Será la entrada para la puesta a cero

De la misma manera que hemos hecho en el ejemplo anterior definimos la variable **Cuenta** que almacenará el valor de contador.

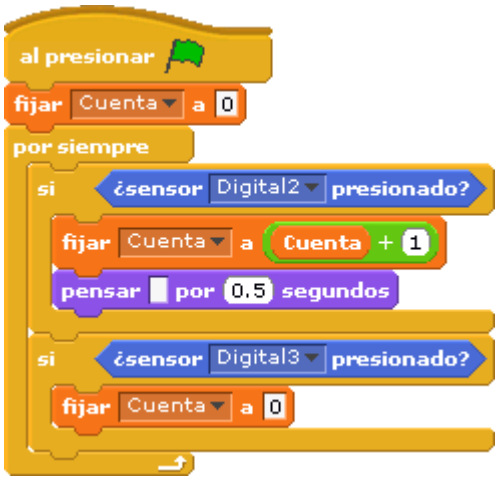
Esta vez dispondremos de dos bucles tipo **“si”** uno para cada una de las dos operaciones **“contar”** y **“poner a cero”**

Contar:

Para el bucle contar testeamos el estado de la variable de entrada digital **Digital2 PIN 2** y si se cumple que esta activada incrementamos el contador **“fijar Cuenta a Cuenta+1”**

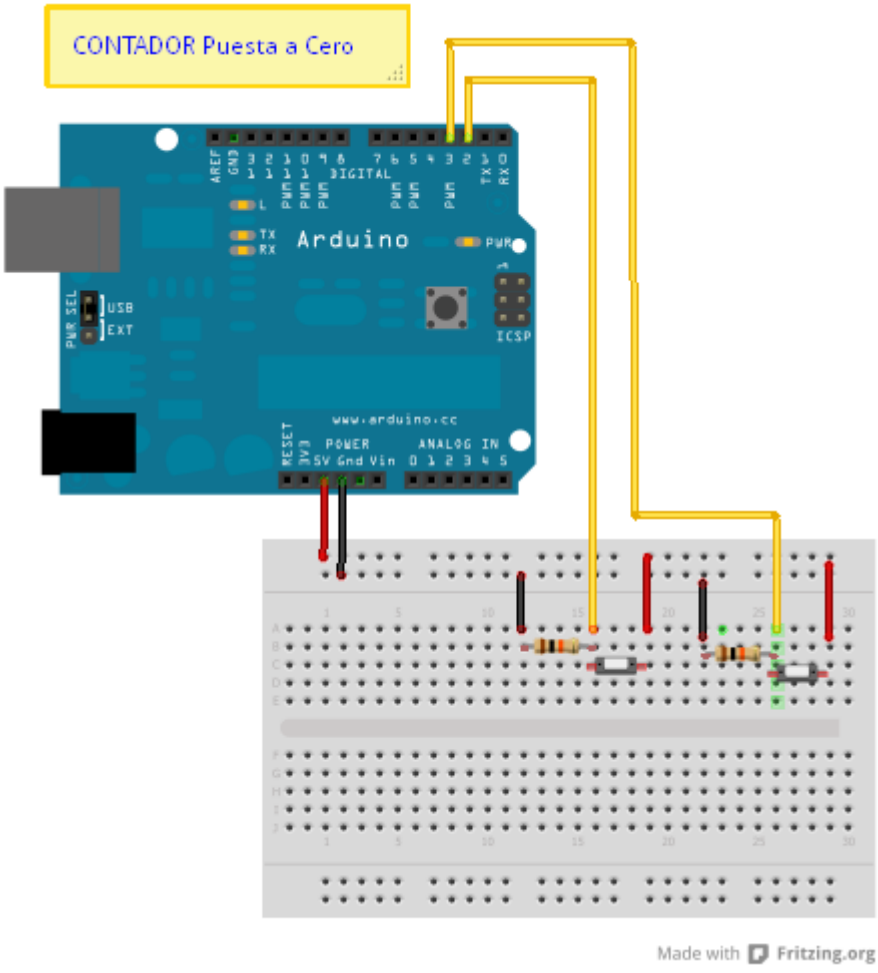
Poner a cero:

Con la entrada **Digital3** ponemos a cero e valor Cuenta **“fijar Cuenta a 0”**



A continuación vemos el aspecto de la pantalla “escenario”

El siguiente es el montaje de la aplicación.



13. Semáforo

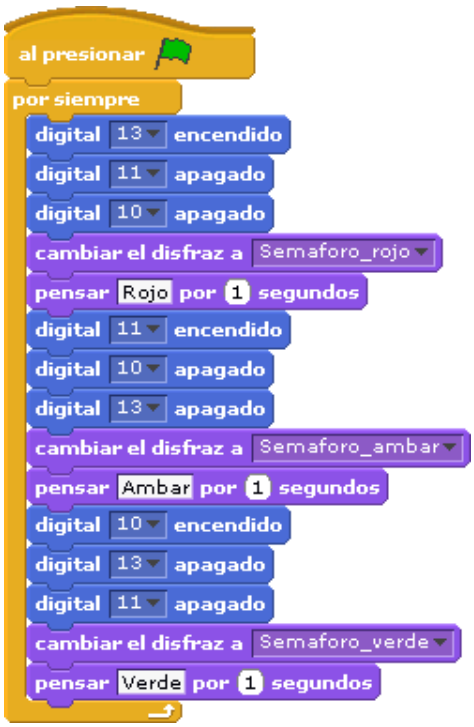
En este ejemplo vamos a realizar un semáforo. Utilizaremos las siguientes salidas

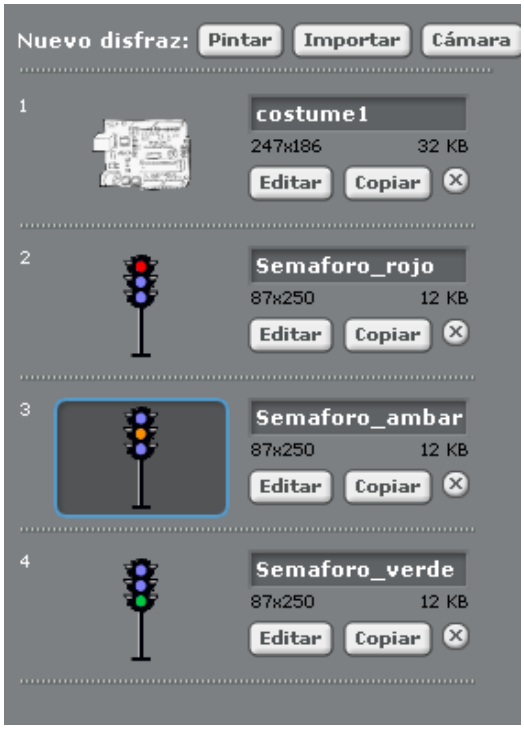
Lámpara Roja	**PIN 13**
Lámpara Ámbar	**PIN 11**
Lámpara Verde	**PIN 10**

Se han creado tres disfraces para representar los tres estados del semáforo que se irán mostrando de acuerdo a la secuencia de encendido de las salidas:

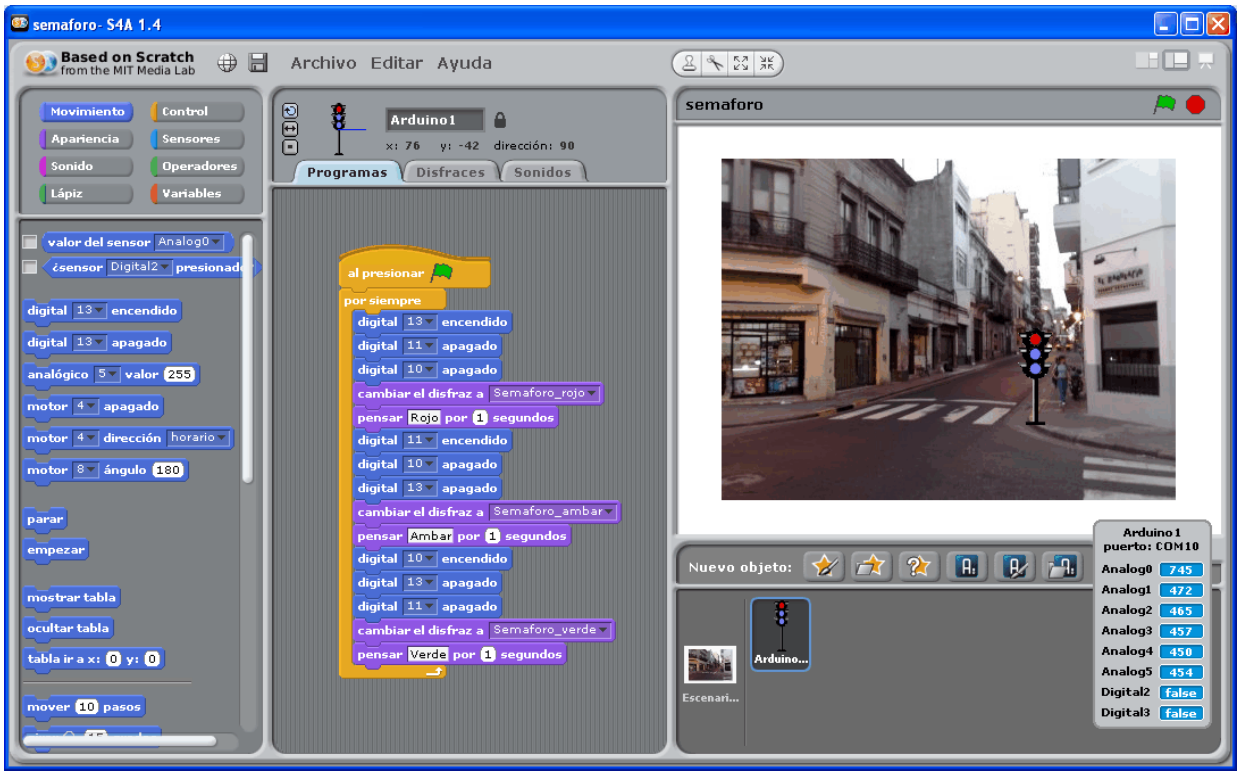
"cambiar disfraz a Semáforo_rojo"
"cambiar disfraz a Semáforo_ambar"
"cambiar disfraz a Semáforo_verde"

Los tiempos de espera se han definido mediante la función **"Pensar … por 1 segundo"**

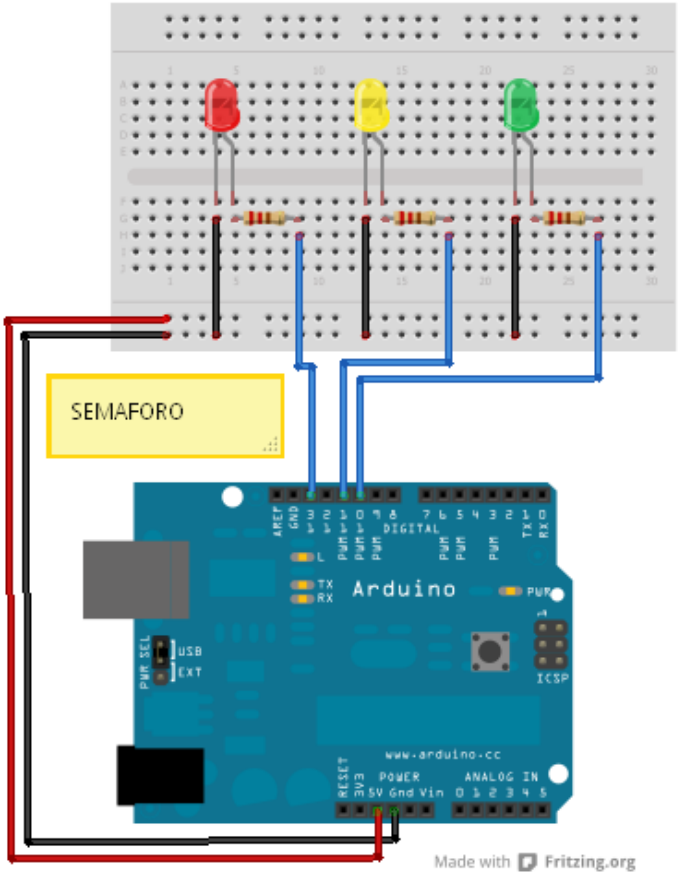




En la pantalla de **"escenario"** se ha colocado de fondo la imagen de un cruce de calles, lo cual le da cierto realismo a la aplicación.



En la siguiente imagen se muestra el circuito de montaje con protoboard.



14. Control de un motor

El entorno S4A dispone de dos salidas para el control de un motor mediante las instrucciones:

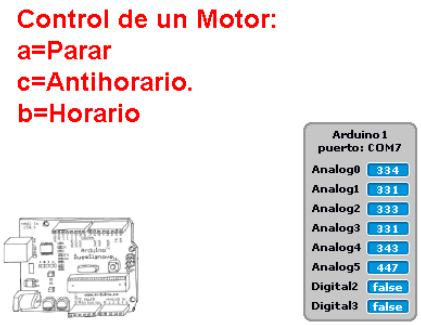
“motor … apagado”
“motor … dirección horario/antihorario”

En este montaje se trata sencillamente de asociar el mando del motor a tres letras del teclado del ordenador:

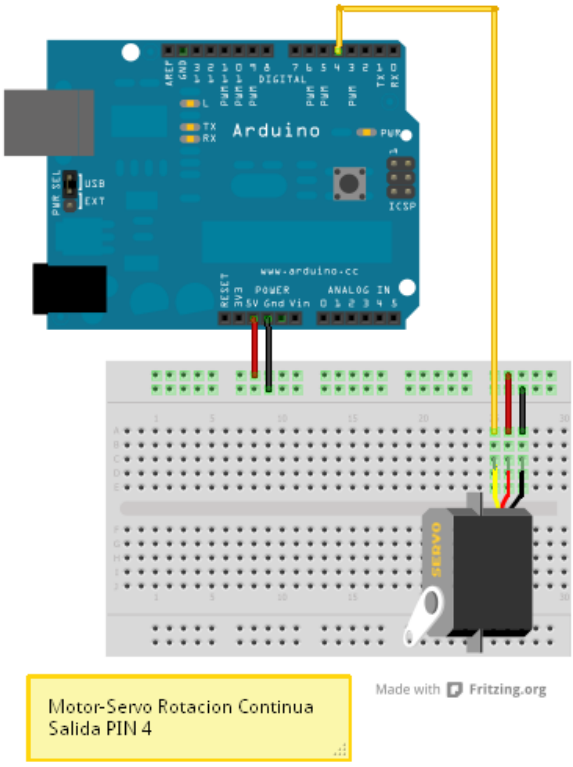
Tecla **a**	apagado
Tecla **b**	Giro en sentido horario
Tecla **c**	Giro en sentido antihorario

Se ha dispuesto que al iniciar el programa el motor se fuerce al estado apagado y eso se hace con la función **“motor … apagado”**

En la pantalla del escenario se ha escrito un texto explicativo.



El esquema de montaje es el siguiente.



15. Control de un servomotor (giro 180º)

Para controlar un servomtor del tipo 180º de giro S4A dispone e la instrucción **"motor … angulo"** en la que ángulo es un valor entero que pude estar comprendido entre 0 y 180.

Crearemos dos variables: **angulo y sensor**

angulo es el valor que colocaremos en la instrucción de control del motor **"motor … ángulo…"** El valor ángulo se obtiene redondeando el valor medido en **Analog0** dividido por **0.18**, dado que el valor máximo que se puede poner como ángulo es **180** y sin embargo el valor máximo que leemos del canal es **1024**. Por eso **0.18=180/1024** El redondeo es porque la función motor no admite ángulos que se den con valores decimales solo admite valores enteros.

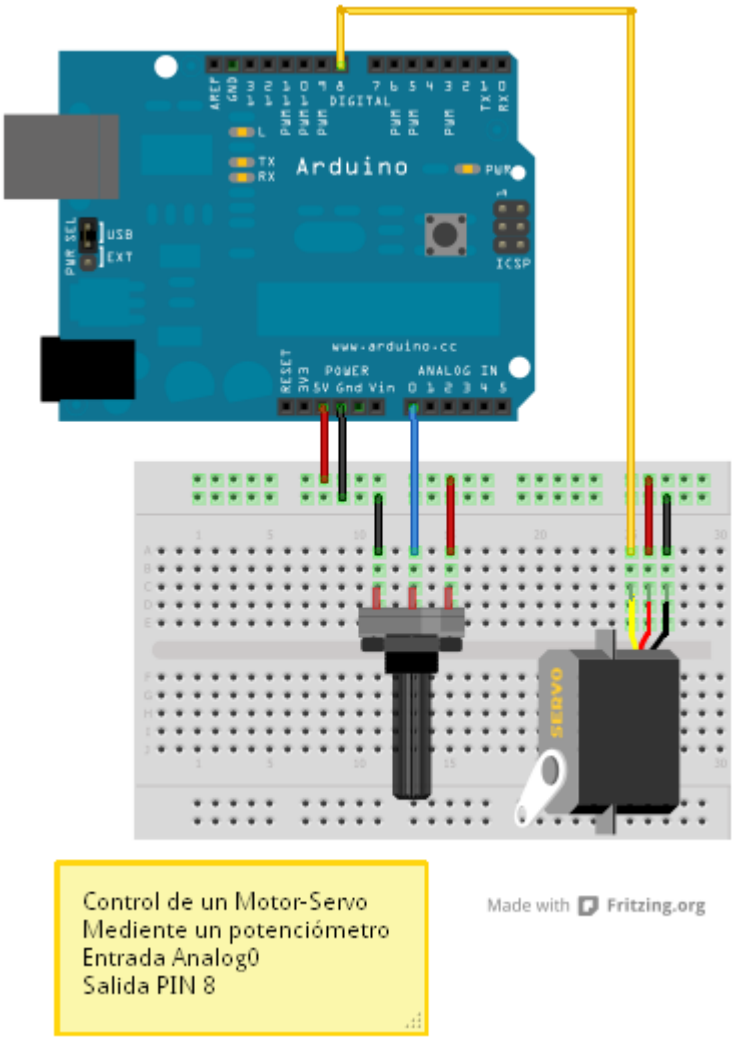
sensor Es el valor leído del canal analógico de entrada **"fijar sensor a valor sensor Analog0"**

La idea en esta aplicación es que el ángulo de giro se obtenga a través de una de la entradas analógicas de la tarjeta Arduino, en este caso lo haremos a través de **Analog0 "valor sensor Analog0"** este valor leído lo asignamos a la variable

El algoritmo es muy sencillo. En primer lugar fijamos la posición de reposo en 90º **"motor 8 ángulo 90".** Seguidamente, dentro de un bucle de repetición continua "por siempre" se fijan las variables a sus valores y para terminar se activa el motor con la variable ángulo **"motor 8 ángulo angulo"**

En la pantalla **"escenario"** colocamos para su visualización los valores de las variables sensor y angulo.

En la siguiente figura vemos el esquema de montaje de la aplicación.



16. Lectura de un canal analógico de entrada

Con este ejemplo queremos leer el valor de uno de los canales de entrada analógica de la Tarjeta Arduino y mostrar su valor en la pantalla **"Escenario"** del entorno **S4A**.

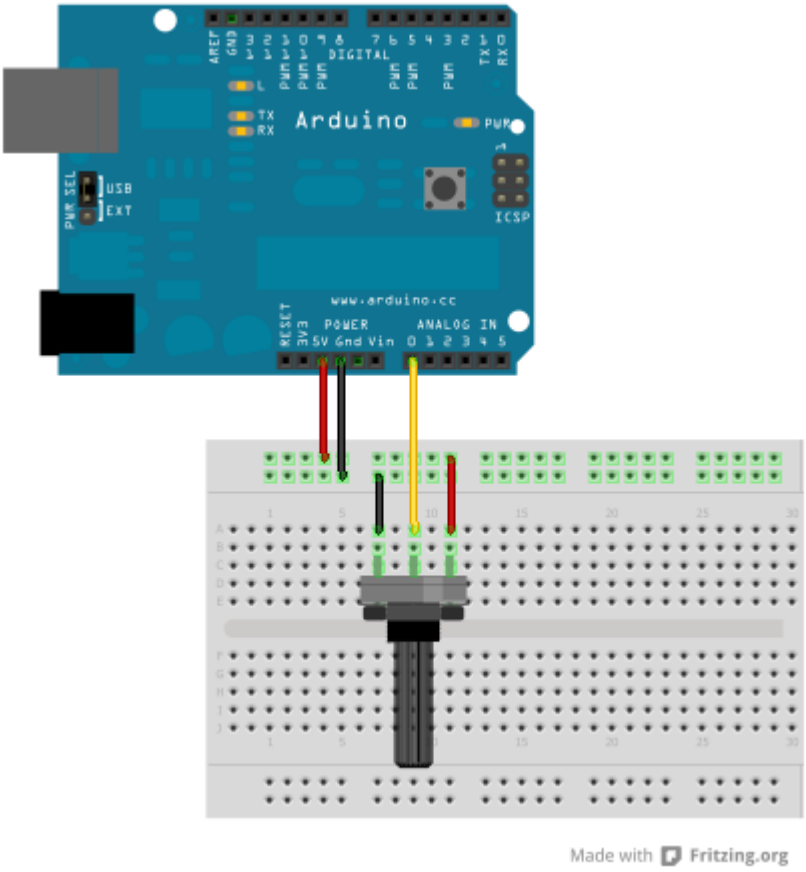
La solución es muy sencilla, basta recurrir al bloque de función **"fijar.."**. que colocamos dentro del bucle **"por siempre"**

Previamente debemos definir una variable seleccionando la librería **"variables"** y marcando nueva variable: le pondremos el nombre **Analogica1**

La variable **Analogica1** la asociaremos al **"valor del sensor Analog1"** bloque de función que se encuentra en la librería **"movimiento"**

A continuación se muestra el aspecto de la pantalla **"Escenario"** en el que hemos colocado el valor de la variable **Analog1** con un texto qu se ha colocado editando simplemente el fondo del escenario **"Lectura de un canal analógico Analog1="**

En la figura vemos el montaje en protoboard de la aplicación.



17. Simulación de un Termostato

En muchos automatismos y sistemas de control el estado de un valor digital en una salida depende del valor de una variable analógica de entrada. Este es el típico caso de un "termostato"

Esa vez definiremos la variable analógica de entrada como **value** y la asignaremos al canal analógico Analog1 mediante la función **"fijar value valor del sensor analog1"**

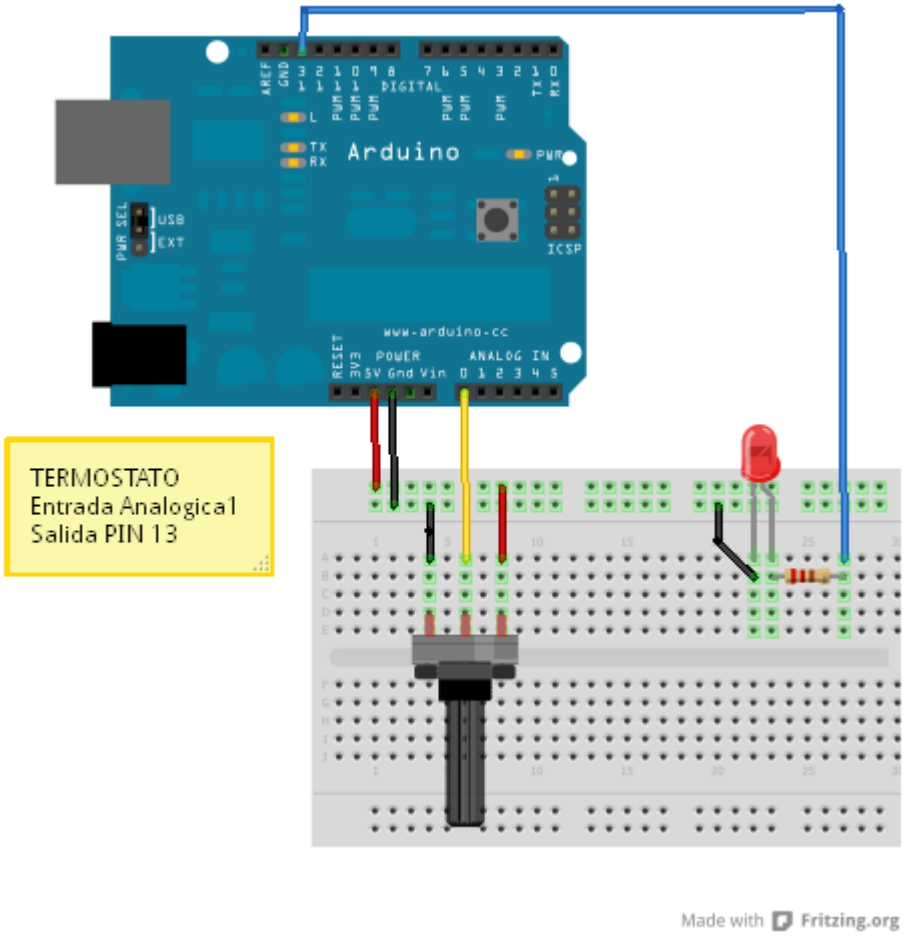
Lo que procederá a continuación es establecer un condicional compuesto **"si ,si no"** en el que se interrogue por el valor de "value", estableciéndose que:

si **value <800** entonces salida digital **PIN 13=true "digital 13 encendido"**

y en caso de no cumplirse la condición **PIN 13=false "digital 13 apagado"**

Se han establecido dos "disfraces" para representar la estufa encendida y apagada. El aspecto de la pantalla "escenario" es el siguiente.

El montaje del circuito es el siguiente.



18. Traspaso de un valor analógico de entrada a una salida analógica

En el ejemplo siguiente se recogerá un valor de un canal de entrada analógica **"Analógica1"** de la tarjeta Arduino y se dirigirá a una de las salidas analógicas **PIN 5**

Se definirían dos variables:

- **Analogica1** para el canal leído **"valor del sensor Analog1"**
- **Sal_Analogica1** para el valor que sacaremos por el **PIN 5** que actuará en este caso como salida analógica.

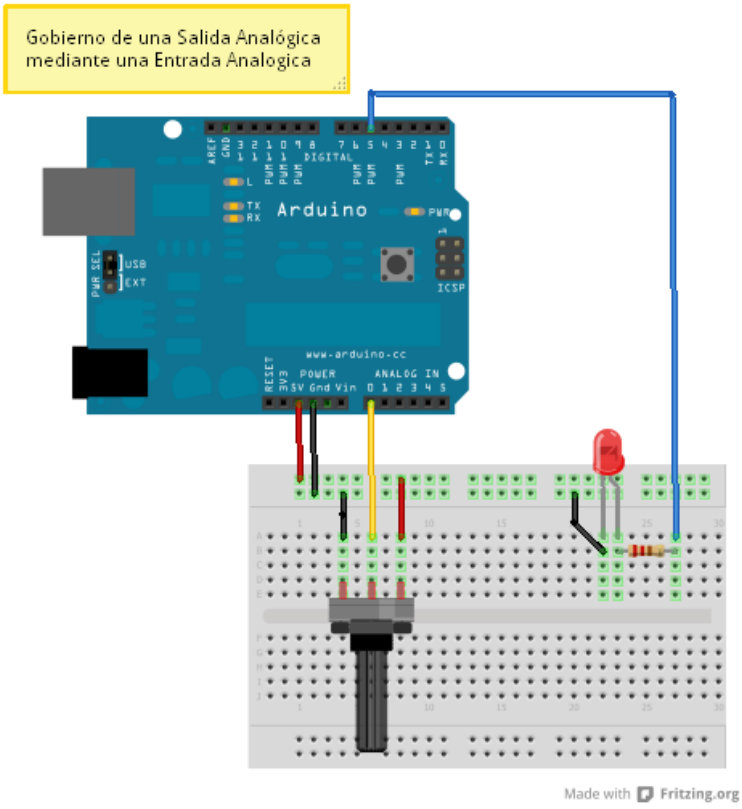
Las instrucciones para fijar los valores son:

"fijar Analogica1 a valor del sensor Analog1"
"fijar Sal_Analogica1 a redondear Analogica1/(1024/255)"

Es muy importante que se tenga en cuenta que en las salidas solo se puede sacar un valor entre **0 y 255**, sin embargo, la entrada analógica leído puede oscilar entre **0 y 1024**. Para resolver este problema lo que hacemos es aplicar un factor de reducción a la señal Analógica1 **K=1024/255**, por eso en la orden fijar para sacar el valor Sal_Analogica1 se utiliza este factor **K** y, además, se redondea el valor porque Arduino no permite sacar por sus salidas analógicas un valor que tenga decimales, deben ser valores enteros

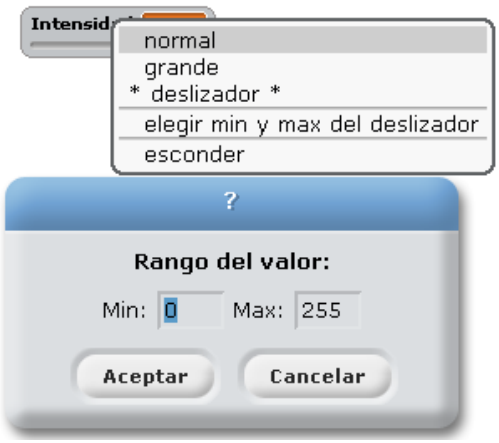
En la pantalla **"escenario"** se muestra los dos valores tanto la lectura analógica como el valor de escritura en la salida. Se ha editado el **"fondo del escenario"** y se ha rotulado el texto en rojo que aparece.

El siguiente es el esquema de la aplicación



19. Gobierno de una salida analógica desde la pantalla del Escenario

Se definirá una variable a la que llamamos, por ejemplo, **intensidad**. Seleccionamos en el elemento de presentación de la variable (en el escenario) que se muestre en modo **deslizador** y también se define el valor **mínimo y máximo de la variable (0 a 255).** En la figura vemos como hacerlo mediante el botón derecho del ratón

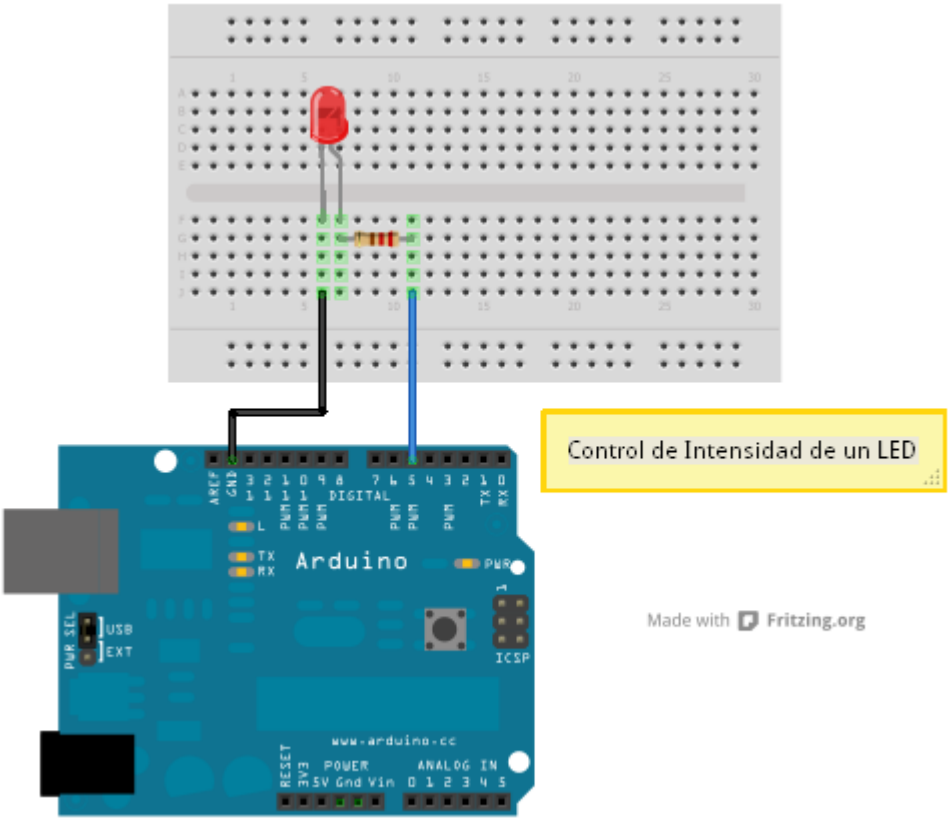


El programa es tan sencillo como el que sigue. Basta que la función de salida de valores analógicos se alimente con el valor intensidad: **"analógico 5 valor intensidad"**

La pantalla **"escenario"** es la que se muestra a continuación. Lo que hemos hecho es editar el fondo y poner el texto en rojo que aparece.

Bastará mover el cursor de la variable Intensidad para hacer que la intensidad del diodo led conectado al **PIN 5** varíe.

El montaje sobre protoboard es el siguiente.



20. Gobierno de una salida analógica mediante el valor de la posición x del ratón.

Con esta aplicación vamos a descubrir otra manera de interacción que facilita S4A con el mundo exterior.

En el ejemplo veremos cómo es posible cambiar la intensidad de un led colocado en la salida analógica **PIN 5** tomando en cuenta la **posición x** del ratón sobre la pantalla "escenario"

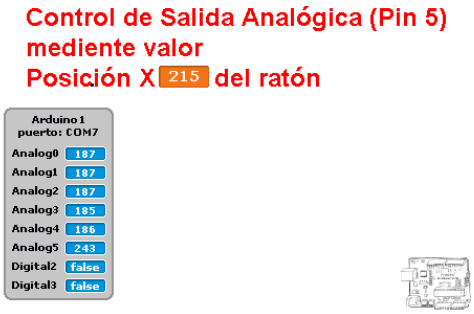
Definimos una variable a la que le ponemos el nombre de "intensidad".

Asociamos al valor de intensidad la posición del ratón tomada de la función **"x del ratón" .** Se ha redondeado el valor porque la función analógica de salida recoge solo valores enteros comprendidos entre **0 y 255** también se ha utilizado la función **"abs"** que transforma el valor en valor absoluto

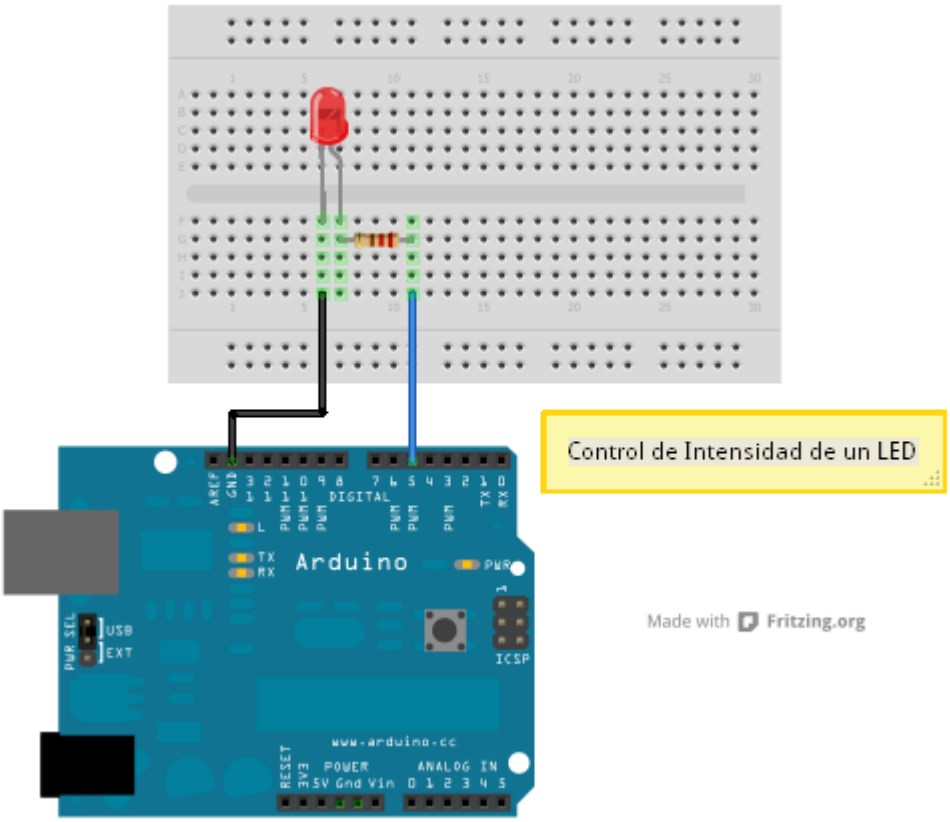
Básicamente el algoritmo es muy sencillo.

- Se fija el valor de la intensidad **"fijar Intensidad a abs de redondear x del ratón"**
- Se envía el valor intensidad a la salida **PIN 5 "analógico 5 valor intensidad"**

A continuación se muestra la pantalla "escenario".



Este es el esquema de montaje a realizar.



21. Gobierno de una salida analógica mediante un bucle secuencial continuo

En la práctica anterior hemos gobernado una salida analógica con una variable desde la pantalla. Ahora vamos a hacer lo mismo pero en lugar de asignar nosotros el valor con el cursor movible de la variable lo hará solo el sistema creado dos bucles.

En primer lugar definimos una variable que denominamos **intensidad** y que al comienzo del programa la ponemos a cero **"fijar intensidad a 0"**

Se establecerán dos bucles uno de **ascenso** de valor y otro de **descenso** lo cual significa que el Led de la salida **analogica5 PIN 5,** mediante **"analógico 5 valor intensidad"** se irá iluminando de menos a más hasta alcanzar un valor máximo y a partir de ese instante entraremos en un segundo bucle que será todo lo contrario, de decremento, hasta llegar a un valor mínimo.

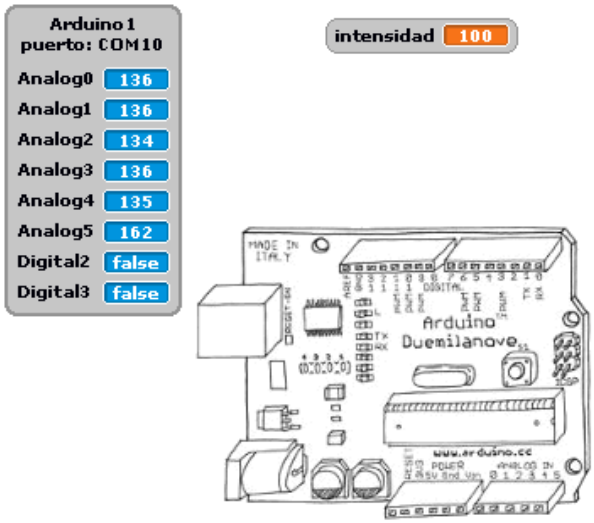
Los saltos de valor se han establecido de 10 en 10, tanto para el ascenso como para el descenso. Se ha dispuesto un retardo de 0.2 seg. para que el efecto se perciba mejor **"pensar por 0.2 segundos"**

Los bucles se han montado con la función **"repetir hasta que"** con la condición de valores máximo y mínimo a alcanza en cada bucle **intensidad >240** e **intensidad<20.**

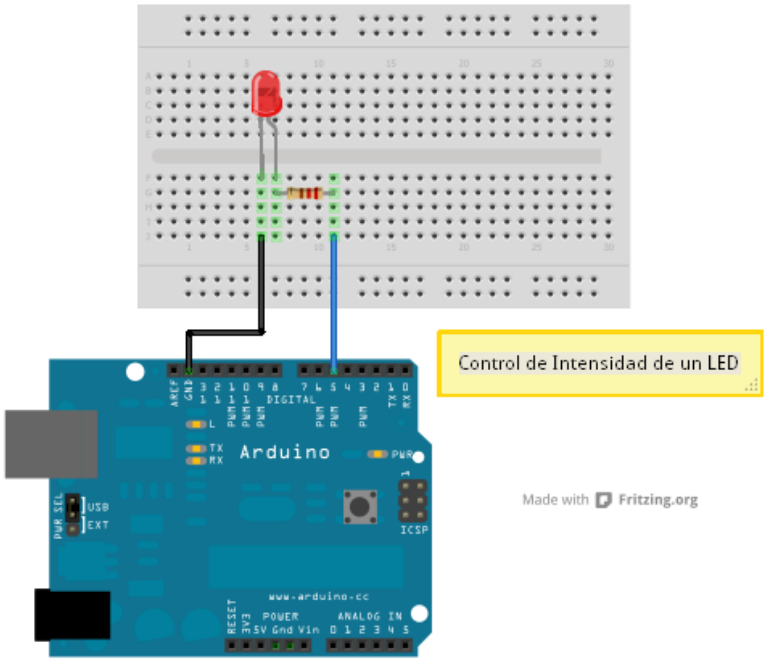
Las funciones encargadas de incrementar y decrementar los valores de intensidad son:

“fijar intensidad a intensidad + 10” y **“fijar intensidad a intensidad - 10”**

El aspecto de la pantalla **“escenario”** es el que vemos a continuación



El esquema del montaje es el siguiente.



22. Termómetro con leds y sensor LM35

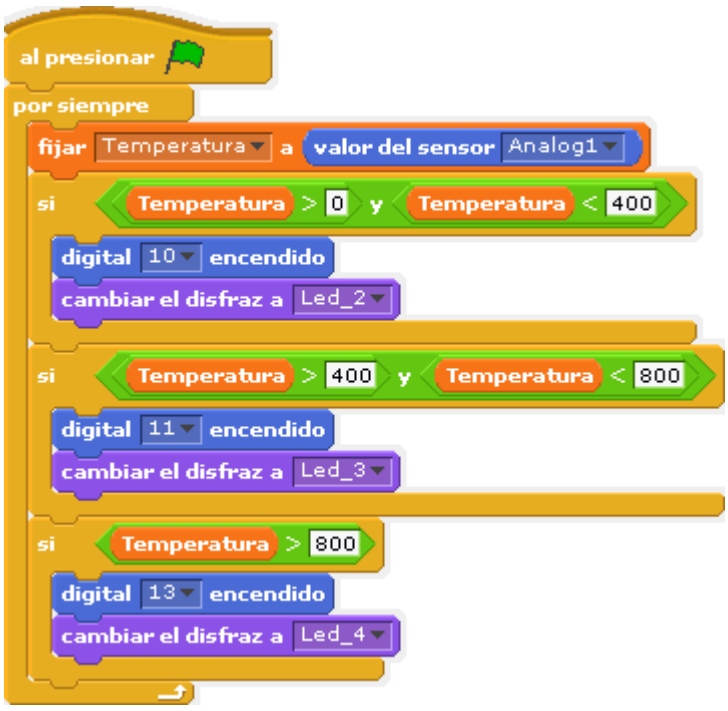
La siguiente aplicación permite la lectura de un canal de entrada analógico **Analog1** y la posterior comparación con unos rangos de valor que permitieran el gobierno de 3 salidas digitales.

En primer lugar se definirá una variable analógica que llamaremos **Temperatura** y cuyo valor se asignará al canal **Analog1** de la Tarjeta Arduino mediante el bloque de función **"fijar Temperatura a valor del sensor Analog1"**

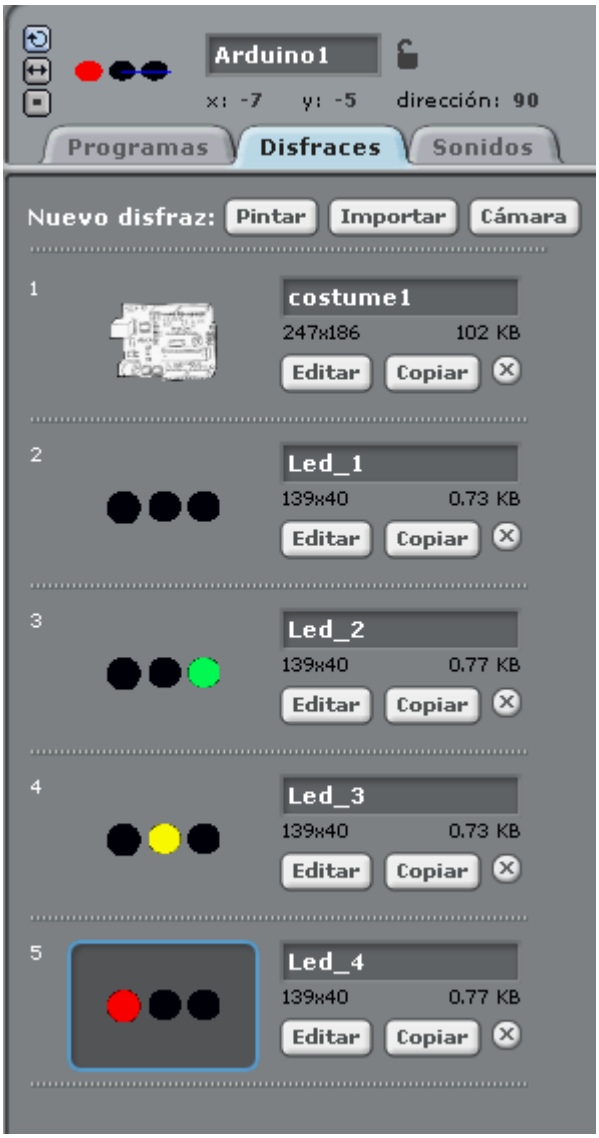
Ls salidas a gobernar serán 3**: PIN 10, PIN 11** y **PIN13.**

Las condiciones que se establecen para el gobierno de las salidas vienen dadas por los rangos que figuran en la cabecera de las funciones condicionales **"si"**

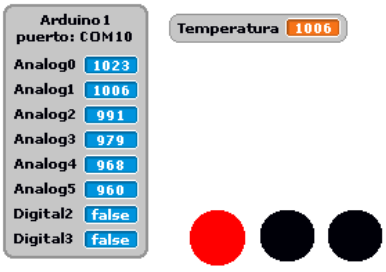
Si **400 > Temperatura >0** entonces **PIN 10 =true**
Si **800 > Temperatura >400** entonces **PIN 11 =true**
Si **Temperatura >800** entonces **PIN 13 =true**



Se han creado cuatro **"disfraces"** que permiten indicar en le pantalla **"escenario"** el estado real de las salidas, en la figura se pueden ver: **Led_2, Led_3 y Led_4** el disfraz **Led_1** se ha creado por si deseáramos un cuarto rango en el que no se activase ninguna salida

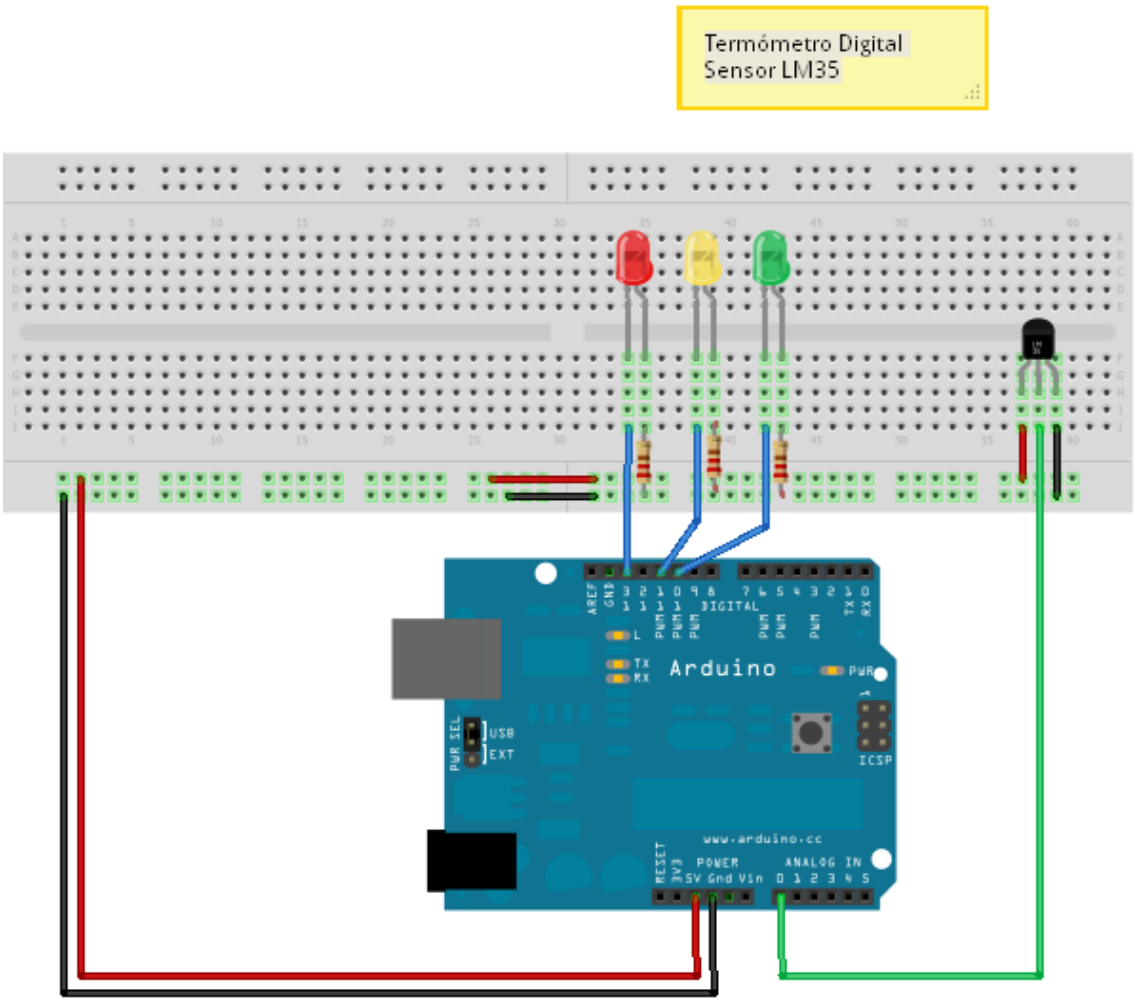


La figura siguiente muestra el estado de la pantalla “escenario”



A continuación se muestra el esquema de montaje en en protoboard de esta aplicación.

Se ha tomado como sensor de temperatura un sensor de semiconductor LM35



23. Instalación Domótica

Para finalizar esta colección de ejemplos se realizara una aplicación que permite el control de dos aspectos básicos en una instalación de "domótica": La iluminación y la calefacción.

Se cuenta con la utilización de los disfraces que representen las lámparas encendidas y apagadas y también un radiador encendido y apagado.

Se ha colocado una imagen de una habitación en la parte de fondo de la aplicación. Con todo buscamos la aproximación al realismo de la instalación.

La aplicación tiene tres objetos:

El objeto principal **Arduino**
El objeto **Iluminación** que se corresponde con la iluminación
E objeto **Calefacción** que se corresponde con la calefacción

Designación de variables:

Las variables en este ejemplo son las siguientes:

Consigna_luz	Permite seleccionar desde pantalla el valor de la luz deseada
Consigna_temp	Permite seleccionar desde pantalla el valor de la temperatura deseada.
Luz	Mide a través del canal analógico **Analog1** el valor de la temperatura ambiente
Temperatura	Mide el valor de la temperatura en el canal de entrada **Analog2**
Lámpara	Indica el estado de la lámpara (1= encendida 0=apagada)
Radiador	Indica el estado del radiador (1=encendido 0=apagado)

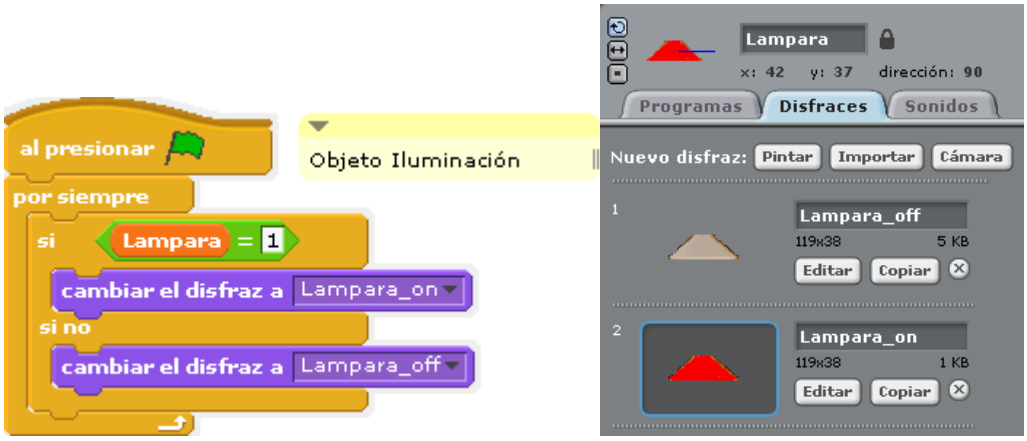
El objeto Iluminación tiene asociado un programa muy sencillo que se encarga de controlar los disfraces **"Lampara_off"** y **"Lámpara_on".** Para ello, mediante un condicional **"si, si no"** testea la variable **Lampara** y si esta es 1 se pasa al disfraz lámpara_on **"Cambiar el disfraz lámpara_on"**

Sensores analógicos de entrada:

Canal **Analo0 PIN 0 analógico** para medir la luz
Canal **Analog1 PIN 1 analógico** para medir la temperatura

Salidas digitales:

PIN 13 salida de activación de lámpara
PIN 11 salida de activación de lámpara
PIN 10 salida de activación de radiador



El objeto Calefacción tiene asociado el programa que se encarga de controlar los disfraces **"radiador_off"** y **"radiador_on".** Para ello, mediante un condicional **"si, si no"** testea la variable **Radiador** y si esta es 1 se pasa al disfraz radiador_on **"Cambiar el disfraz radoador_on"**

El programa principal perteneciente al objeto Arduino es el que se muestra en la siguiente figura.

- Dentro de un bucle de repetición "por siempre" como todos los programas lo primero que se hace es fijar las variables a sus fuentes de dato:

"fijar Temperatura a valor del sensor Analog1"
"fijar Luz a valor del sensor Analog0"

- Seguidamente se pasa a una función condicional tipo **"si, si no"** que se encargará de comprobar si el valor de la variable **Luz** es menor del valor de la variable **Consigna_luz.**

- Si se cumple la condición eso significa que hay poca luz y debemos encender las lámparas:

"digital 13 encendido" y "digital 11 encendido"

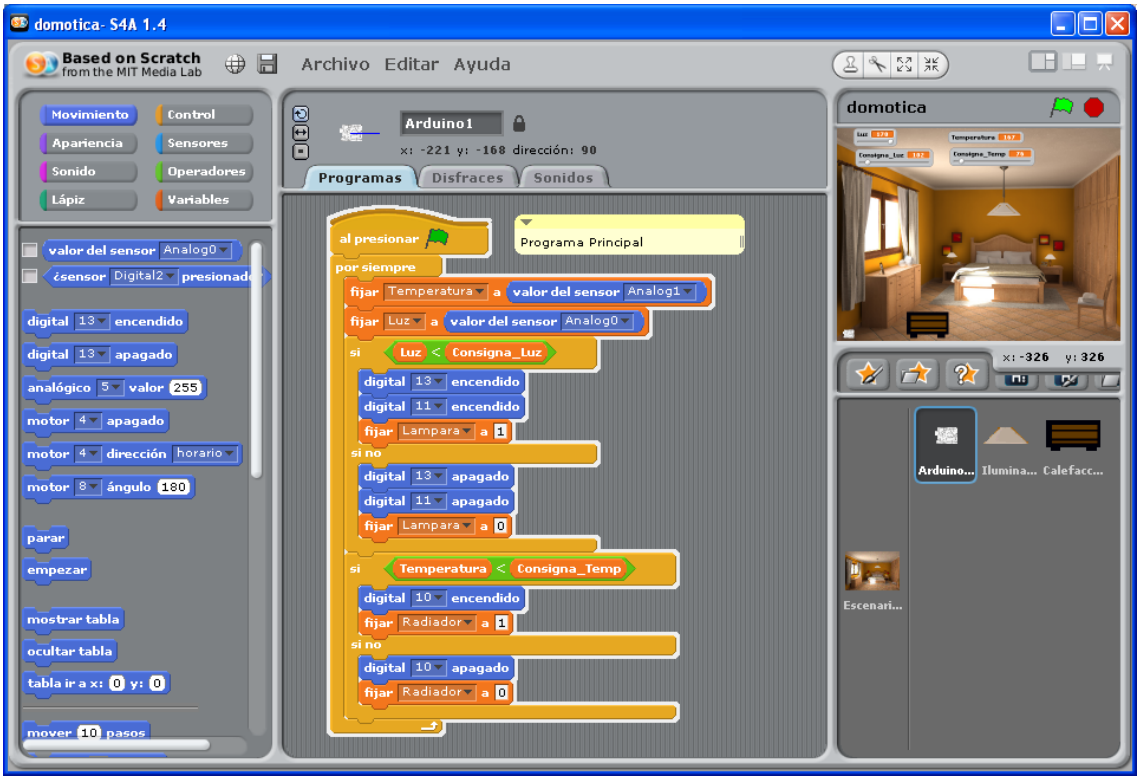
- El siguiente bucle condicional tipo **"si, si no"** se encargará de comprobar si el valor de la variable **Temperatura** es menor del valor de la variable **Consigna_temp.**

- Si se cumple la condición eso significa que hace frio y debemos encender el radiador:

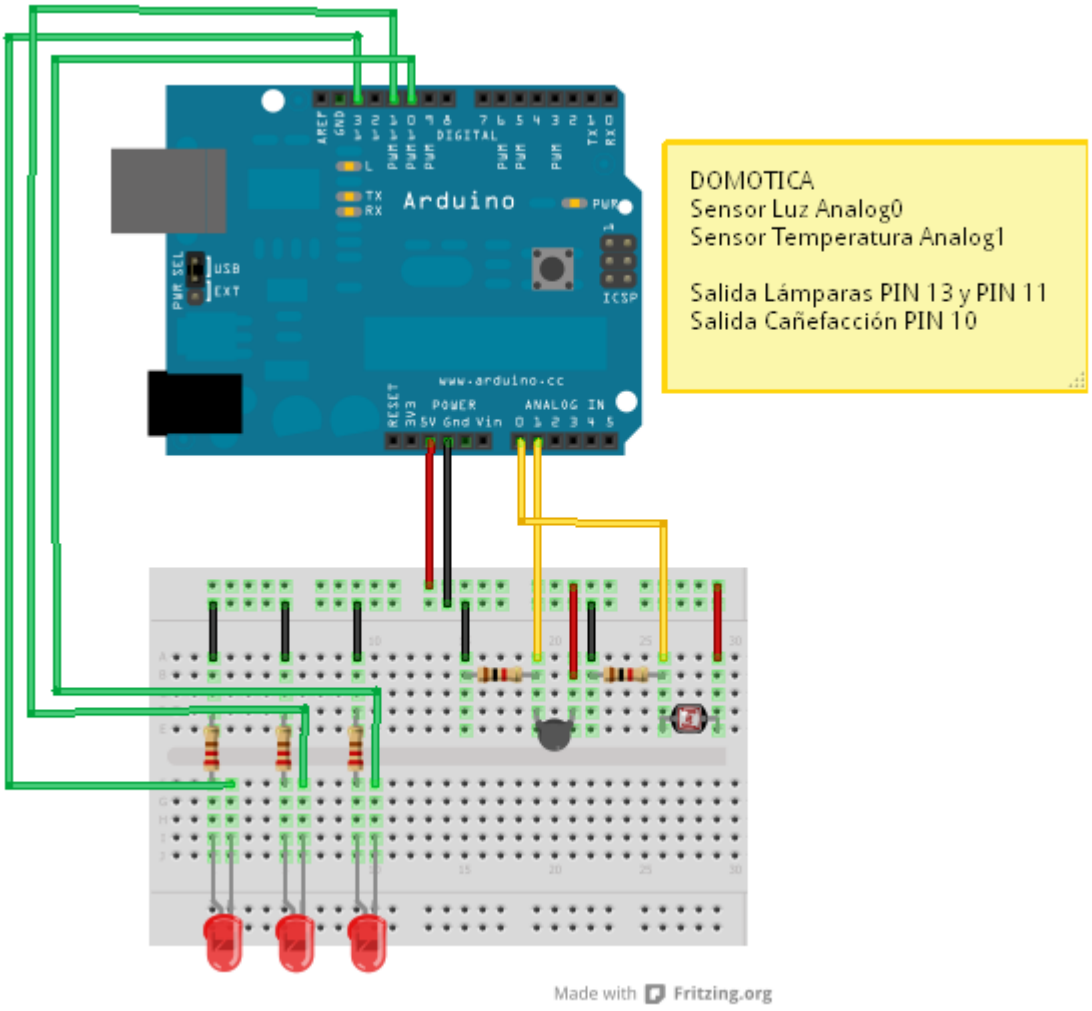
“digital 10 encendido”

La siguiente figura muestra la pantalla de 2escenario de la aplicación”

Vemos una imagen del entorno con el programa cargado.



Este sería el montaje de ñpa aplicación.



Bibliografía y lugares para ampliar información.

Lugar oficial de Citalb Projecte Scratch: http://s4a.cat/index_es.html

Lugar oficial de la plataforma Arduino:
http://www.arduino.cc/es/Main/Software

Pagina oficial de Scratch:
http://scratch.mit.edu/

Documentación variada y muy útil EDUTEKA.
http://www.eduteka.org/modulos.php?catx=9&idSubX=278#!modulo-Scratch