Scratch 2.0 + Arduino
Snap! + Arduino

Librería **2sa_fm**

Comunicación y Programación Gráfica de la Plataforma Arduino con la ayuda de las últimas herramientas software Inspiradas y Desarrolladas en el Laboratorio de Medios del MIT

Manual y Guía de Aplicaciones

José Manuel Ruiz Gutiérrez

Enero 2014 Ver.1

Índice

1. Introdución

Poder realizar el control de una tarjeta Arduino sin necesidad de escribir código puede resultar muy interesante. Con Scratch ahora es muy fácil, gracias a la incorporación de nuevos bloques de función en el paquete de librerías que viene con Scratch.

La librería s2a_fm nos va a permitir justamente esto: Gobernar Arduino desde Scratch 2.0 y también desde la variante de Scratch llamada Snap! basada en el Byob

Estamos ya en disposición de controlar dispositivos físicos tales como sensores de temperatura, potenciómetros, sensores de luz, sensores de distancia, movimiento, etc. y receptores como Leds, Motores, servos, etc.

Realmente resulta fascinante la idea.

¡Vamos a ello!

2. ¿Que es s2a_fm?.

s2a_fm es una librería, extensión de hardware conectable con Scratch, escrito en Python, que permite la comunicación entre un microcontrolador Arduino y Scratch sin problemas.

s2a_fm se compone de tres componentes principales de software:

1. Un servidor HTTP que se comunica con Scratch. El estándar de Python **BaseHTTPServer** se utiliza por simplicidad y la compatibilidad.
2. El convertidor de protocolos traduce los datos entre los protocolos HTTP y los protocolos **Firmata**. Todo esto se hace sin problemas y permite a Scratch y Snap! abordar proyectos con Arduino.
3. PyMata , una biblioteca de Python que se comunica con el Arduino utilizando el protocolo estándar de Arduino Firmata . Esto hace que las comunicaciones sean rápidas, eficientes y consistentes. PyMata se encarga de todos los detalles del protocolo Firmata.

Para iniciar un nuevo proyecto de Scratch, sólo se debe cargar el archivo de script **s2a_fm_base.sb2** suministrado con esta versión. Para iniciar un proyecto con Snap! Se debe importar **s2a_fm_Snap_base.xml**. Estos archivos contienen todos los bloques necesarios para realizar el gobierno de Arduino (entradas, salidas, analógicas y digitales) listos para ser usados. Después de cargar el proyecto los bloques se encuentran en la librería de bloques de extensión de Scratch o Snap! y están listos para ser usados en sus proyectos. Para Snap! Los bloques se pueden encontrar en el grupo de librerías de *movimiento, sonido* y s*ensores* de la paleta de bloques.

3.- Instalación de s2a_fm

Archivos Phyton necesarios:

Antes de instalar s2a_fm, debe tener los siguientes componentes Python instalados en el equipo:

Python versión 2.7 o superior. Puede descargar Python desde: (http://python.org/)
PySerial (http://pyserial.sourceforge.net/)
PyMata (https://github.com/MrYsLab/PyMata)

Para instalar PySerial y PyMata, puede utilizar el programa de instalación de loibrreias y anexos para Phyton llamado "**pip**" (http://www.pip-installer.org/en/latest/) si lo tiene instalado en su equipo. Para usarlo, abra una ventana de comandos y escriba:

PIP install "nombre_paquete"

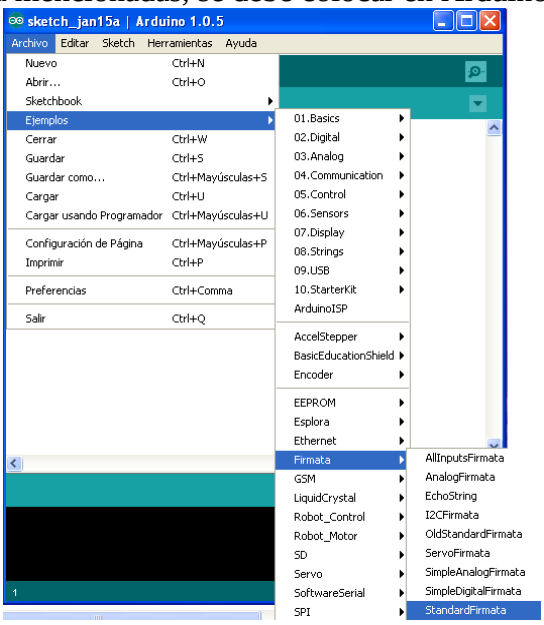
Como alternativa, puede descargar los paquetes utilizando los enlaces de arriba, PySerial esta contendio en un archivo ejecutable setup y en el caso de del paquete PyMata se deberá escribir:

python setup.py install.

Nota: es posible que tenga privilegios de administrador para realizar la instalación.

Arduino Sketch:

Para realizar la conexión e intercambio de datos entre Arduino y Scrtach, estando en medio Phyton con las librerías ya mencionadas, se debe colocar en Arduino el firmware Firmata.



Bastará con cargar el skecth llamado **"StándardFirmata**" en la tarjeta Arduino, que se encuentra en la librería Firmata incluida con el IDE de Arduino.

Si desea utilizar las instrucciones Tono y/o SONAR (ping) proporcionadas por S2a_fm , debe cargarse el sketch **FirmataPlus** incluida en la distribución de la libreraia s2a_fm. Asegúrese de que está utilizando PyMata 1.54 o posterior con este sketch.

Tendrás que añadir la librería **NewPing** para Arduino que compila correctamente este sketch. Se puede descargar en: https://code.google.com/p/arduino-new-ping/downloads/detail?name = NewPing_v1.5.zip

NOTA : Después de instalar NewPing , tendrá que modificar uno de sus archivos . Siga las instrucciones que se facilitan en este enlace:

https://code.google.com/p/arduino-new-ping/wiki/HELP_Error_Vector_7_When_Compiling

A pesar de que el StandardFirmata ha sido reemplazado por FirmataPlus, pero todavía se proporciona como parte de la distribución para aquellos que deseen utilizarla.

Se recomienda el uso de Arduino IDE versión 1.5.5 o superior. El uso de una versión anterior puede dar errores de compilación.

Ejecutando s2a_fm

Es el momento de conectar Arduino a un puerto USB de su ordenador.

Vaya al directorio donde ha instalado el paquete s2a_fm y escriba:

python s2a_fm.py COM_PORT_ID

Se ha incluido un fichero de ejecución por lotes (tipo bat) en la carpeta *..\s2a_fm\extra_goodies\windows* que tiene ya creada la instrucción de arranque del driver de comunicación con el puerto.

COM_PORT_ID es el nombre del puerto serie que se utiliza para comunicarse con el Arduino y será el mismo nombre que el que se utiliza cuando se conecta el IDE de Arduino. Por ejemplo, para Windows , podría ser COM3, para linux, podría ser / dev/ttyACM0 .

Después de ejecutar el comando, debería ver algo como lo siguiente aparecen en la ventana de terminal:

s2a_fm version 1.2 Copyright(C) 2013-2014 Alan Yorinks All Rights Reserved
Opening Arduino Serial port /dev/ttyACM0
Please wait while Arduino is being detected. This can take up to 30 seconds ...
Board initialized in 1 seconds

Total Number of Pins Detected = 20
Total Number of Analog Pins Detected = 6
Please wait for Total Arduino Pin Discovery to complete. This can take up to 30 additional seconds.
Arduino Total Pin Discovery completed in 1 seconds
Starting HTTP Server!
Use <Ctrl-C> to exit the extension

Por favor, inicie Scratch o Snap!

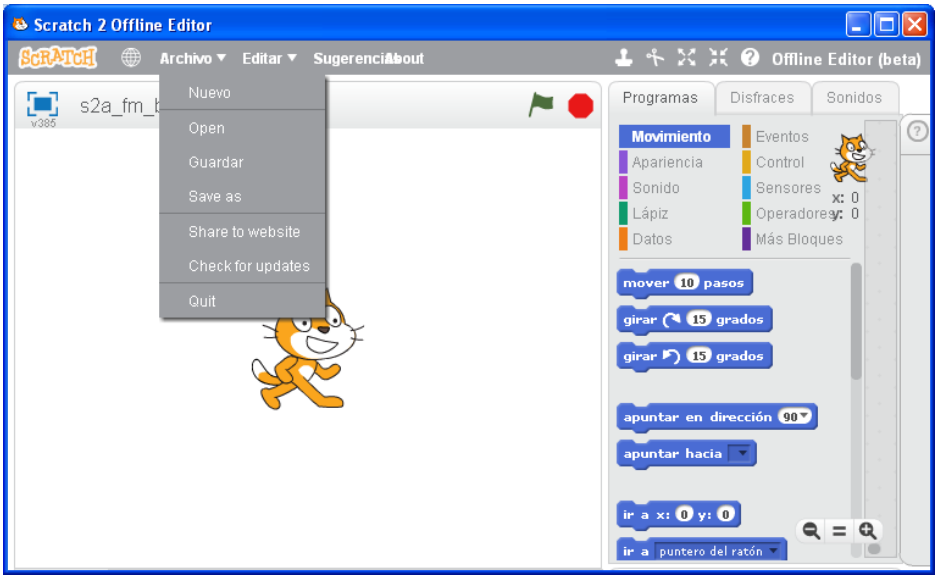
Si es así ¡enhorabuena! Acaba de abrir la vía por la que se podrá comunicar Arduino y Scratch

NOTA: Si trabaja con un PC antiguo, con sistema operativo Windows XP o inferior es fácil que se ralentice bastante la ejecución de las ordenes de control desde Scratch sobre todo si utiliza Scratch 2.0. Mejor trabajar con Snap! si no le importa, ira mas rápido.

4. Creación de un proyecto.

4.1. Con Scratch 2.0

Ahora inicie Scratch 2.0. Cree un nuevo proyecto y seleccione Archivo/Cargar desde el ordenador y seleccione **s2a_fm_base_Es.sb2** desde el directorio/ScratchFiles/ScratchProjects incluido en la distribución s2a_fm



Cuando Scratch se detectado aparece la siguiente línea de texto en la consola:

Scratch detected! Ready to rock and roll...

Y si vas a la pestaña de librería “Mas bloques” se debe ver: El indicador circular junto a *"s2a_fm - Scratch para Arduino"* en color verde, lo que indica que la conexión ha sido establecido con éxito.

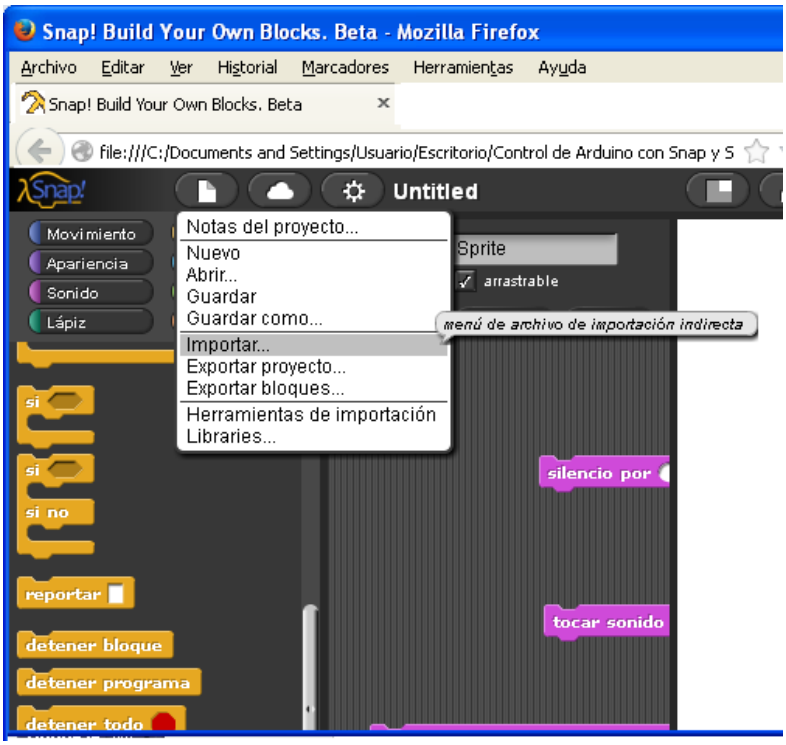
Lo que procede ahora es realizar el programa con el entorno Scrtach arrastrando y parametrizando cada bloque.

4.2. Con Snap!

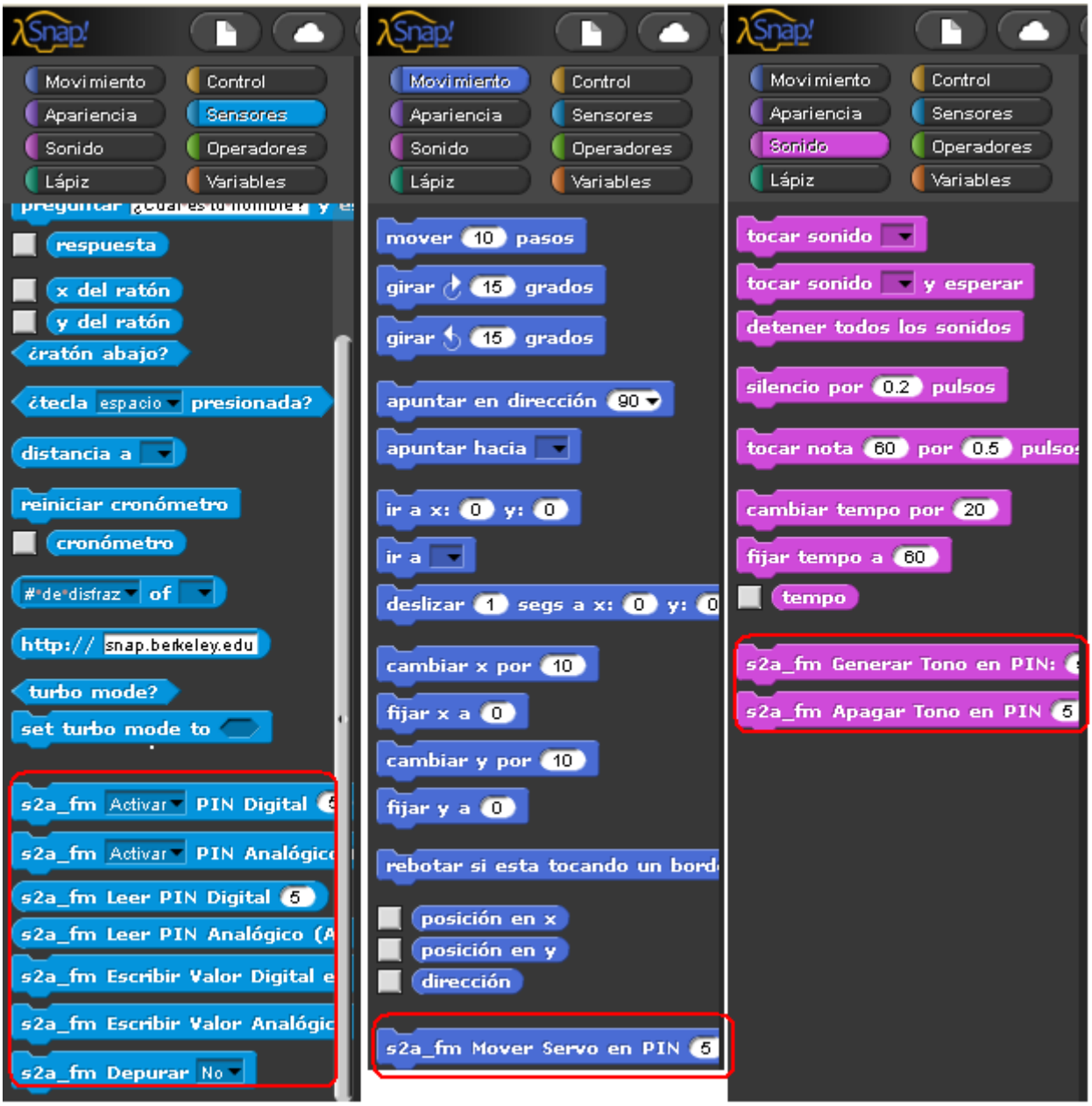
Una vez iniciado Snap!, vamos al menu Archivo y seleccione Importar.

Vaya al directorio Snap! Archivos y seleccione el archivo " **s2a_fm_Snap_base_Es.xml** " . Se trata de un proyecto en blanco que carga todos los bloques de la librearía s2a_fm .

Ahora ya podremos realizar nuestra aplicación arrasytrando los bloques necesarios para componer el “algoritmo” del programa que se comunicara con arduino.



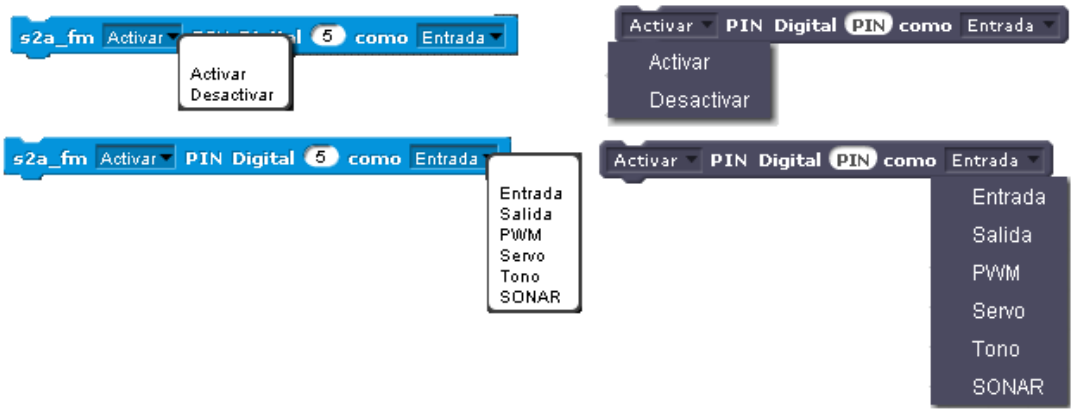
Los bloque añadidos son los que figuran en la siguiente imagen y como vemos se han colocado en las librerías Sensores, Movimiento y Sonido respectivamente.


Snap!
Movimiento
Control
Apariencia
Sensores
Sonido
Operadores
Lápiz
Variables
respuesta
x del ratón
y del ratón
¿ratón abajo?
¿tecla espacio presionada?
distancia a
reiniciar cronómetro
cronómetro
#de disfraz of
http:// snap.berkeley.edu
turbo mode?
set turbo mode to
s2a_fm Activar PIN Digital
s2a_fm Activar PIN Analógico
s2a_fm Leer PIN Digital 5
s2a_fm Leer PIN Analógico (A
s2a_fm Escribir Valor Digital e
s2a_fm Escribir Valor Analógic
s2a_fm Depurar No
mover 10 pasos
girar 15 grados
girar 15 grados
apuntar en dirección 90
apuntar hacia
ir a x: 0 y: 0
ir a
deslizar 1 segs a x: 0 y: 0
cambiar x por 10
fijar x a 0
cambiar y por 10
fijar y a 0
rebotar si esta tocando un bord
posición en x
posición en y
dirección
s2a_fm Mover Servo en PIN 5
tocar sonido
tocar sonido y esperar
detener todos los sonidos
silencio por 0.2 pulsos
tocar nota 60 por 0.5 pulso
cambiar tempo por 20
fijar tempo a 60
tempo
s2a_fm Generar Tono en PIN:
s2a_fm Apagar Tono en PIN 5

5. Descripción de los bloques de la librería

Los bloques de color blanco y negro son los de Scratch. Los bloques de color son la Snap ! *Los bloques en* Snap! empiezan con el nombre de identificación " s2a_fm ".

5.1. Establecer Pin en modo digital



Estas son las imágenes de los bloques con sus pestañas de configuración para Snap! y para Scratch 2.0

Para Snap! este bloque se encuentra en la ficha Sensores de la paleta de bloque y para Scratch 2.0 en la ficha "Mas bloques"

Este bloque habilita o deshabilita un pin digital Arduino (Activar, Desactivar) y permite definir la forma de trabajo del Pin: Entrada, Salida , PWM, Servo o Tono o SONAR.

Activar

Antes de acceder a un pin de entrada o de salida tiene que estar habilitado . Para habilitarla, seleccione Activar desde el primer menú desplegable. Si se elige el modo de entrada (véase más adelante), lo que permite el pasador instruirá automáticamente la placa Arduino se notifican los cambios de valor del Pin.

Desactivar

Si un pin ha sido previamente "Activar", seleccionando "Desactivar" se inactivará el pin de su modo y naturalmente también su modo de trabajo . Normalmente, esta opción de Desactivar no se usa, pero se proporciona para ofrecerle una completa flexibilidad en la escritura de guiones de Scratch.

Numero de Pin

El valor introducido debe estar dentro del rango de números de pines para la tarjeta Arduino. El número de pines detectados para el Arduino se muestra en la ventana de la

consola al iniciar el programa por primera vez, y esta información también se coloca en el archivo de registro. Si el número PIN se encuentra fuera del rango de la tabla, un mensaje de error se envía a la consola y se registra un error. El comando se ignora si hay un error detectado.

Modo Digital

Hay 6 modos digitales compatibles.

- **Entrada** para la conexión a un dispositivo de entrada , como un interruptor .
- **Salida** para la conexión a un dispositivo tal como un LED.
- **PWM** es un modo de salida se utiliza para controlar un pasador con un analogWrite. Esto se puede utilizar a desvanecerse el nivel de luz en un LED.
- **Servo** configura el pin para hacer funcionar un motor servo.
- **Tono** configura el pin a obtener de la biblioteca Tone través Firmata .
- **SONAR** configura el pin para trabajar con la biblioteca NewPing . Hasta un máximo de 6 dispositivos pueden ser controlados simultáneamente.

NOTA : El dispositivo de tipo HC- SR04 está configurado para funcionar en modo de pin único (gatillo y eco conectados entre sí) . Consulte la documentación para obtener detalles NewPing : https://code.google.com/p/arduino-new-ping/wiki/NewPing_Single_Pin_Sketch

NOTA ESPECIAL: Algunas placas Arduino proporcionan resistencias pull-up internas Para activar el modo de pull-up de un alfiler, consulte la documentación de Arduino para la instrucción.

5.2. Establecer Pin en modo Analógico

Los pines analógicos son siempre pines de entrada y sólo pueden ser activados o desactivados Se fijan siempre como entradas. Para Snap! este bloque se encuentra en la paleta de bloques Sensores.

Activar

Antes de acceder a un pin analógico de entrada debe estar habilitado. Para habilitarla, seleccione Activar desde el primer menú desplegable. Habilitar el pin fijara automáticamente la placa Arduino para leer y enviar los valores de tensión que se entreguen en la entrada correspondiente. .

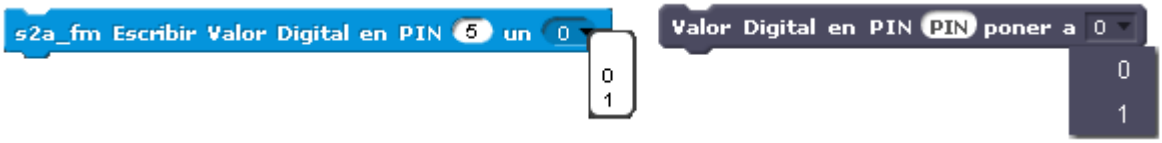
Desactivar

Si un pin se ha Activado previamente, seleccionando Desactivar se inactivará el Pin desde su modo previamente habilitado. Los datos ya no se enviarán mientras está desactivado .

Número de Pin

El valor introducido debe estar dentro del rango de números de los pines analógicos para la tarjeta Arduino. El número de pines detectados por el software de control de puerto es expresado en la ventana de la consola al iniciar el programa por primera vez. Si el número de pin indicado está fuera del rango de la tabla, un mensaje de error se envía a la consola y se registra el error . Si se detecta un error, se ignora el comando. El número de pin usao es el mismo que figura en la tarjeta . Por ejemplo, para leer el pin analógico A3 , ajuste el número de pin en el bloque será 3.

5.3. Escribir valor digital



Para Snap! este bloque se encuentra en la ficha de bloques Sensores.

Número de Pin

El número de pin debe estar ajustado a un pasador que fue habilitado previamente como de salida y está en el intervalo de valores de pin digital . Si la clavija no está habilitado como un pin de salida o si está fuera de rango , la petición es ignorada y un mensaje de error se escribe en la consola y el archivo de registro.

Valor de salida en el Pin

Si un pin digital se ha habilitado para la salida , se puede establecer el nivel de salida en un uno o un cero. Seleccione el valor de la lista desplegable en el bloque.

5.4. Escribir valor Analógico en salida tipo (PWM)

para Snap ! este bloque se encuentra en la ficha Detección de la paleta de bloque.

Número de Pin

El número PIN se debe establecer en un pin que se habilitó previamente como PWM y es un pin digital PWM. Si la clavija no está habilitado como un pin PWM o si está fuera de rango , la petición es ignorada y un mensaje de error se escribe en la consola y el archivo de registro.

Valor PWM

Establecer el valor de salida pin para estar entre 0-255 . Si el valor está fuera del rango de un mensaje de error se escribe en la consola y el archivo de registro. Se tendrá en cuenta la solicitud.

5.5. Generar tono

Este bloque indica al Arduino para reproducir un tono en el pin designado. Para Snap! este bloque se encuentra en la ficha Sonido paleta de bloque.

Número de Pin

El número de PIN debe ser por un pasador que fue habilitado previamente para el tono y se encuentra en el rango de los números de los pines digitales. Si la clavija no está habilitado como un alfiler Tono o si está fuera de rango, la petición es ignorada y un mensaje de error se escribe en la consola y el archivo de registro.

Frecuencia (HZ)

El tono se reproducirá con la frecuencia especificada . No hay validación de datos para esta entrada.

Duración (ms)

El número de milisegundos para mantener el tono. Si este valor establece en cero, el tono se reproducirá indefinidamente. Puede que esté de mi usando el Tono Turn Off bloque.

5.6. Desactivar Tono

Este bloque permite el apagado de un tono en el Pin indicado. Se utiliza sobre todo si la duración de reproducción del tono se ajusta a cero (un tono continuo) .

Para Snap! este bloque se encuentra en la librería de bloques de Sonido.

Número de Pin

El número de PIN debe ser por un Pin que fue habilitado previamente para **Tono**. Si el Pin no está habilitado como un Pin "**Tono**" o si está fuera de rango, la petición es ignorada y un mensaje de error se escribe en la consola y el archivo de registro.

5.7. Mover Servo

Este bloque fijará una posición del servo en el valor especificado en el Pin indicado.

Para Snap! este bloque se encuentra en la librería de bloques Movimiento.

Número de Pin

El número de PIN debe ser sido habilitado previamente para **Servo** y se encuentra en el rango de valores de pines digitales. Si el Pin no está habilitado como un Pin **Servo** o si está fuera de rango, la petición es ignorada y un mensaje de error se escribe en la consola y el archivo de registro.

Posición (Deg)

Esto ajusta la posición del motor expresada en grados . El rango es de 0 a 180. Un valor fuera de este será ignorado y un mensaje de error se escribe en la consola y el archivo de registro.

5.8. Leer Pin digital

Este bloque permite leer el estado de un Pin digital que previamente se ha configurado como **Entrada** digital. Se utiliza para obtener acceso al valor actual para el número pin especificado Devuelve un cero o un uno.

Para Snap! este bloque se encuentra en la pestaña Detección paleta de bloque.

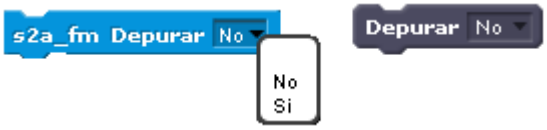
5.9. Leer Pin analógico

Este bloque permite leer el estado de un Pin que previamente se ha configurado como una entrada analógica. Contendrá el valor actual para el número pin especificado en el rango de 0

a 1.023. Introduzca el número de pin analógico Arduino en el campo PIN. Por ejemplo, para leer A3 pin analógico, ajuste el número de pin en el bloque 3.

Para Snap! este bloque se encuentra en la pestaña Sensores de los boques de librería.

5.10. Depurador



Activa o desactiva la función de depuración (el valor predeterminado es Off) de la lista desplegable. Si está habilitada la depuración de todos los comandos de Scratch se registran tanto en el registro como en la consola. Cada comando lleva una marca de tiempo con la hora local. Los bloques de lectura de valores (información del sensor) no aparecen como parte del informe de depuración porque las lecturas de Scratch para los datos es de aproximadamente 30 veces por segundo y esto inundaría el registro.

Para Snap! este bloque se encuentra en la pestaña Sensores en la palleta de librerias. Aquí está una muestra del registro de depuración:

DEBUG: 2013-12-19 14:19:06.733833: debugger On
DEBUG: 2013-12-19 14:19:06.768232: digital_pin_mode Enable 48 Output
DEBUG: 2013-12-19 14:19:06.800449: digital_pin_mode Enable 51 Output
DEBUG: 2013-12-19 14:19:06.849955: digital_write 48 1
DEBUG: 2013-12-19 14:19:06.872217: digital_write 51 1
DEBUG: 2013-12-19 14:19:06.906062: digital_write 48 0

5.11. El botón rojo de parada

Al pulsar el botón rojo de parada en el reproductor de Scratch enviará una orden de reposición a la Arduino y restablecerá todas las estructuras de datos internos a sus valores iniciales.

5.12. Archivo de registro

Un nuevo archivo de registro se crea cada vez que se inicia **s2a_fm**, y el archivo de registro anterior se descarta. El nombre del archivo de registro es " **s2a_fm_debugging.log** " y se encuentra en un directorio llamado "log" de la carpeta que contiene s2a_fm. El archivo de registro contiene registros "info ", que se considera parte de la operación normal y que pueda contener registros de depuración que le ayudará en la depuración de un programa de Scratch cuando se conecte con Arduino .

He aquí parte de un registro típico:

INFO:root:s2a_fm version 1.2 Copyright(C) 2013-2014 Alan Yorinks All Rights Reserved
INFO:root:com port = /dev/ttyACM0
INFO:root:20 Total Pins and 6 Analog Pins Found
INFO:root:Scratch detected! Ready to rock and roll...
DEBUG:root:digital_pin_mode: The pin number must be set to a numerical value
DEBUG:root:analog_write: The value field must be set to a numerical value

En este ejemplo, los registros INFO indican un comportamiento normal y proporcionan datos de información, y los Registros de DEPURACIÓN indican un problema. En la primera línea de depuración, el número de pin no se ha introducido correctamente la hora de establecer un modo de pin digital, y la segunda DEBUG indica que el campo VAL debe ser introducido correctamente al ejecutar un bloque de escritura analógica.

6. Ejemplos de Programas.

A continuación se exponen algunos ejemplos con los que he probado esta librería. Los he realizado en Snap! Dado que en Scratch 2.0 la ejecución, en maquinas mas antiguas co0n Windows XP, se ralentiza mucho la CPU .

6.1. Blink.

En este ejemplo activamos y desactivamos cada segundo un led que colocamos en el **PIN3** de Arduino poniéndose en modo intermitente esta salida digital.

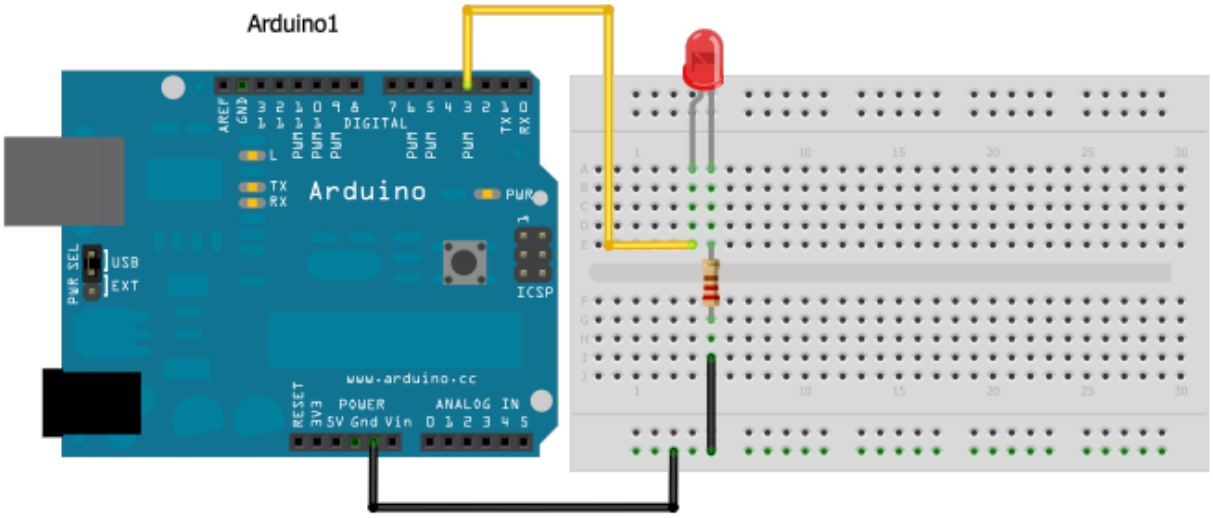
Para realizar el ejemplo basta que seleccionemos los dos bloques de Snap! "*Al Presionar*" y "*por siempre*" que permitirán el inicio de la ejecución y su mantenimiento en un bucle continuo.

En lo que se refiere a los bloques a utilizar de la librería **s2a_fm** haremos uso del bloque de "*Configuración del Pin Digital PIN3*" como salida y seguidamente, ya dentro del bucle de "*repetición*", pondremos las órdenes de "*Escribir Valor Digital en PIN*" y la orden "*esperar*".

Para probar el funcionamiento del ejemplo conectamos Arduino al puerto USB (debemos saber el número de puerto) y después ejecutamos el ficheror **s2a_fm.bat** en el que la orden a ejecutar será:

c:\python27\python s2a_fm.py com6

Este sería el montaje a realizar



6.2. Blink Tiempo Variable.

Este ejemplo es similar al anterior, queremos realizar el encendido y apagado de un led conectado en una salida digital PIN3 pero hacienda que en este caso el tiempo de encendido y apagado se pueda modificar mediante un potenciómetro conectado en la entrada analógica A0.

Los bloques a utilizar son los ya mencionados en el ejemplo anterior "*Al Presionar*" y "*por siempre*" que permitirán el inicio de la ejecución y su mantenimiento en un bucle continuo.

En lo que se refiere a los bloques a utilizar de la librería **s2a_fm** haremos uso del bloque de "*Configuración del Pin Digital PIN3*" como salida y seguidamente, ya dentro del bucle de "*repetición*", pondremos las órdenes de "*Escribir Valor Digital en PIN*" y la orden "*esperar*".

Se debe crear una variable que recogerá el valor analógico leído en el canal A0 de Arduino.

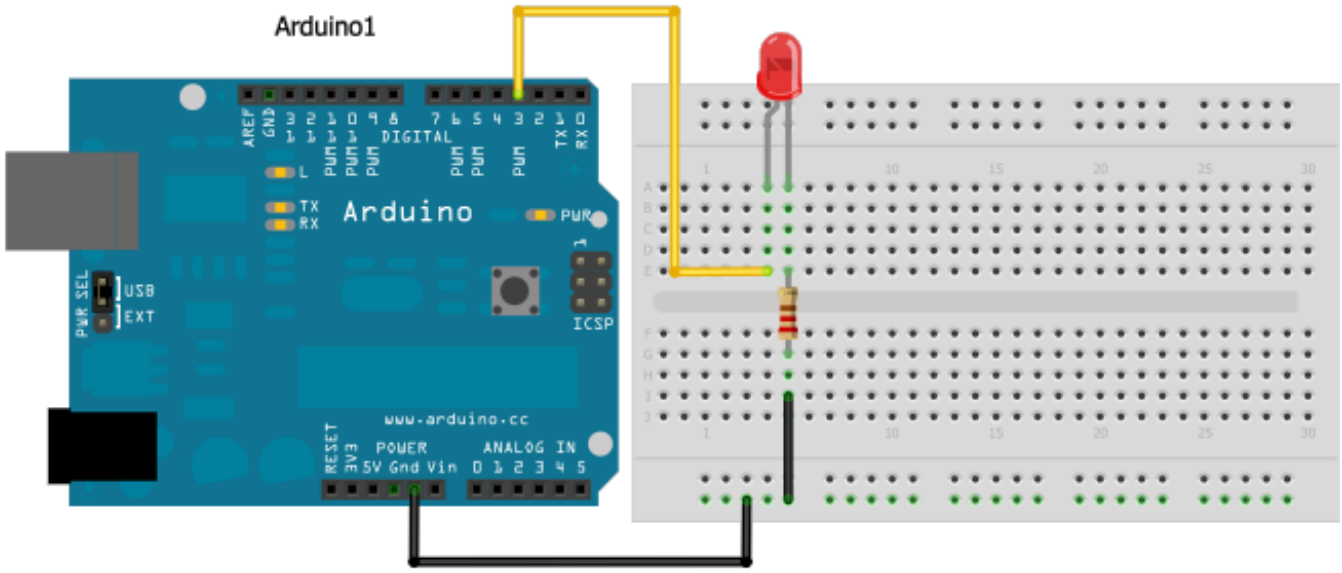
La variable se debe asociar con el valor analógico haciendo uso de un bloque de tipo "*Fijar valor"* pero no debemos hacer la asignación directa sino que hemos dividió el valor leído (max de 1024) por 1024 para acotar el tiempo máximo de variación a 1 seg. 1024/1024=1. También tomaremos la precaución de redondear el valor de este cociente con el fin de no asignar un valor a la instrucción "*Esperar*" que contenga decimales.

Lo primero que se debe hacer, fuera del bucle "*por siempre*" es configurar el pin analógico A0 como activándolo mediante el bloque "*Activar PIN Analógico (A)0 como Entrada* ", el pin

digitalPIN 3 actuará como salida y debemos configurarlo con la instrucción " *Activar PIN Digital 3 como Salida*"

Una vez realizada la configuración entramos en el bucle "por siempre" , fijamos el valor de a al canal A0 y pasamos a activar la salida digital PIN3 con la instrucción *"Escribir Valor Digital en PIN 3 un 1", a continuación temporizamos "esperar tiempo segs"* para seguidamente desactivar la salida PIN3 *"Escribir Valor Digital en PIN 3 un 0",* y volver a temporizar *"esperar tiempo segs".*

Este sería el montaje sobre protoboard.



6.3. Botón

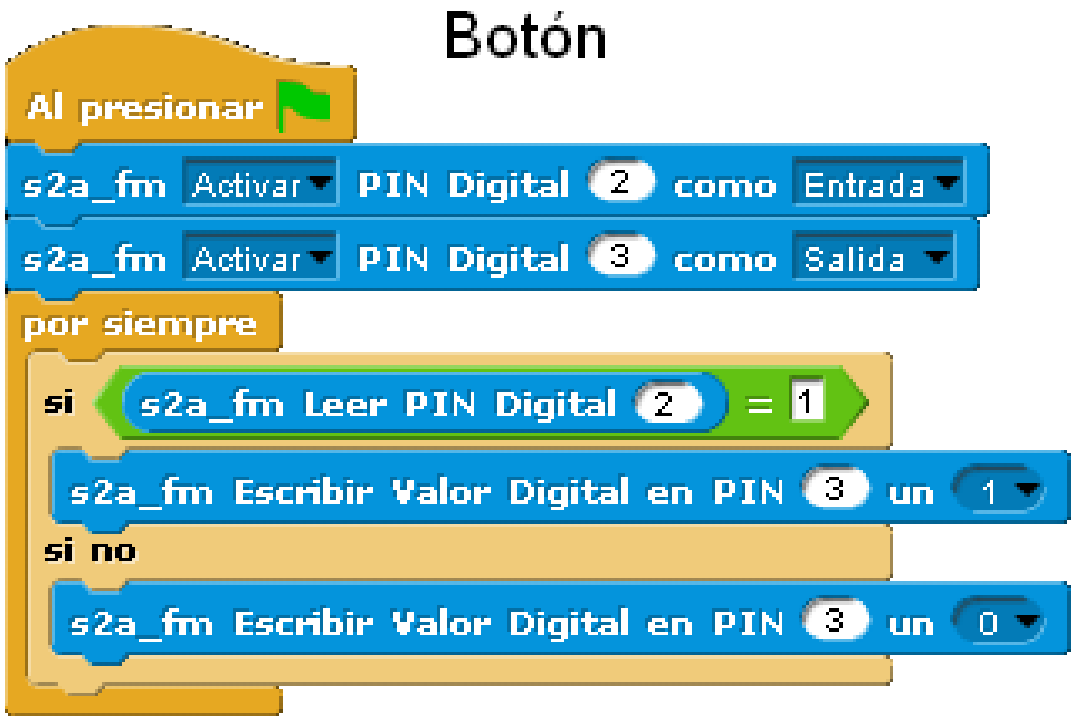
En este ejemplo vamos a leer el estado de un Pin digital configurado como entrada PIN2 y dependiendo de su valor, 0 o 1, activaremos o desactivaremos una salida digital ubicada en el PIN3.

Se utilizan los bloques "Al presionar" y "por siempre". Los Pines se deben configurar, como siempre fuera del bucle mediante los bloques "*Activar PIN Digital 2 como Entrada*" y "*Activar PIN Digital 3 como Salida*".

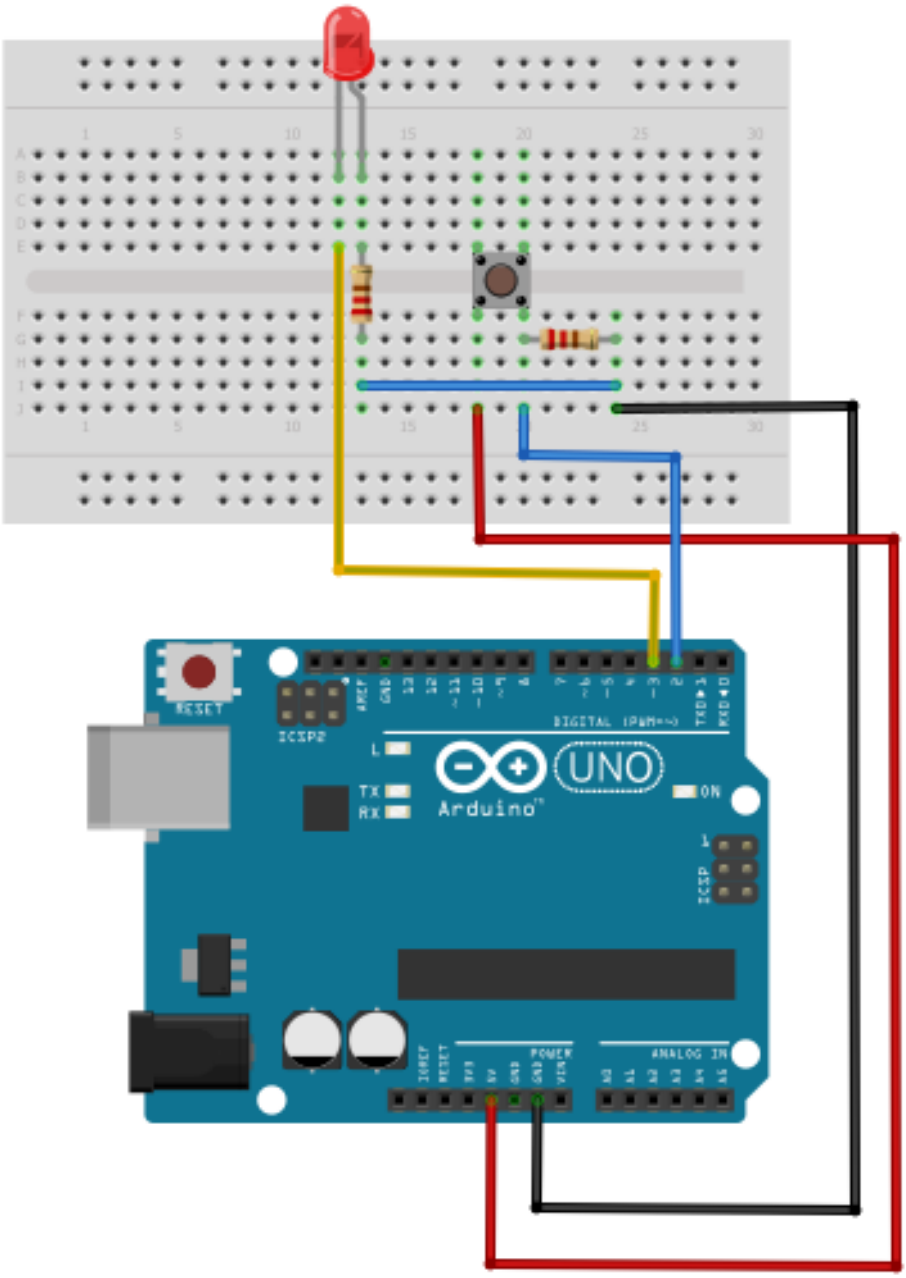
Dentro del bucle "por siempre" ponemos una estructura condicional compuesta "*si… si no*" en la que la condición es el estado del PIN2 que se realiza con un bloque de tipo Operadores con condición "=" cuya variable se el estado del PIN2 que se recoge mediante el bloque "*Leer PIN Digital 2*" y en el segundo miembro ponemos directamente el valor "1".

Si la condición se cumple activaremos la salida digital PIN3 "*Escribir Valor Digital PIN 3 un 1*"

Si la condición no se cumple desactivaremos la salida digital PIN3 "*Escribir Valor Digital PIN 3 un 0"*



Este sería el montaje sobre protoboard.



6.4. Contador

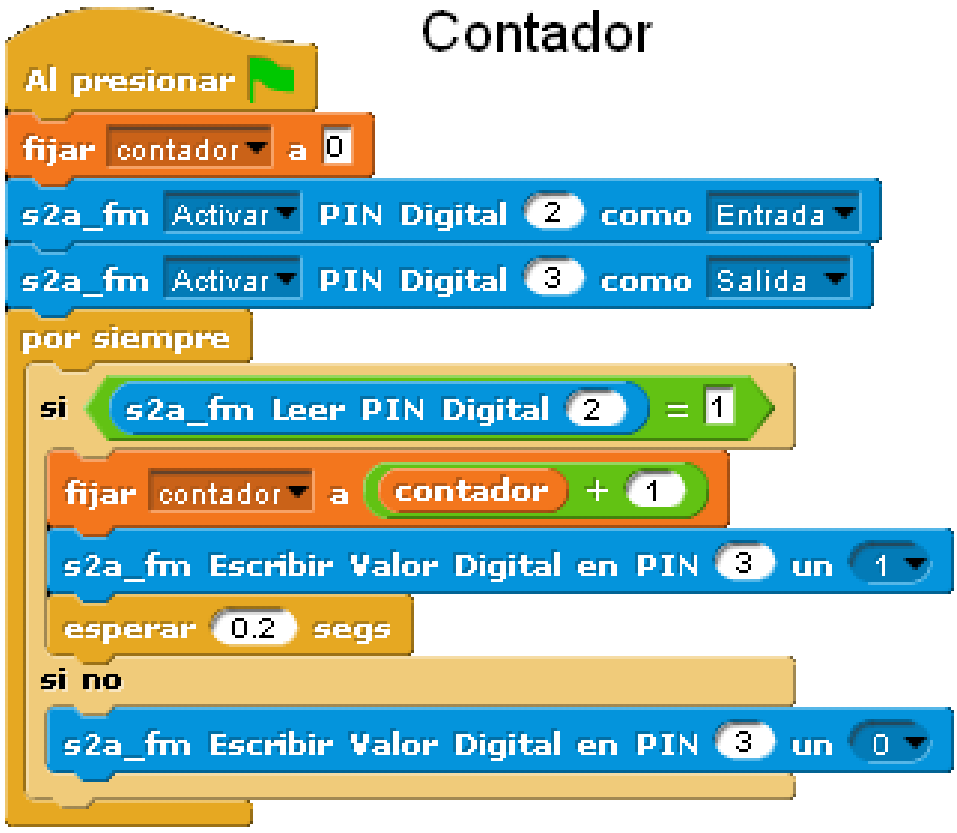
En este ejemplo se trata de recoger los cambios de valor de la entrada digital **PIN2** y realizar la cuenta de los impulsos que llegan. Se habilitará el **PIN3** como salida que refleje estos impulsos que llegan. Crearemos una variable llamada contador en la que se almacenará el valor del estado de cuenta y que se mostrará en todo momento en la pantalla.

La forma de realizar el programa seria, una vez configuradas la entrada y salida digitales que hemos mencionado **PIN2** y **PIN3** se entraría en el bucle "*por siempre*" y dentro de él preguntaríamos por el estado de la entrada mediante una instrucción condicional "*si si no*".

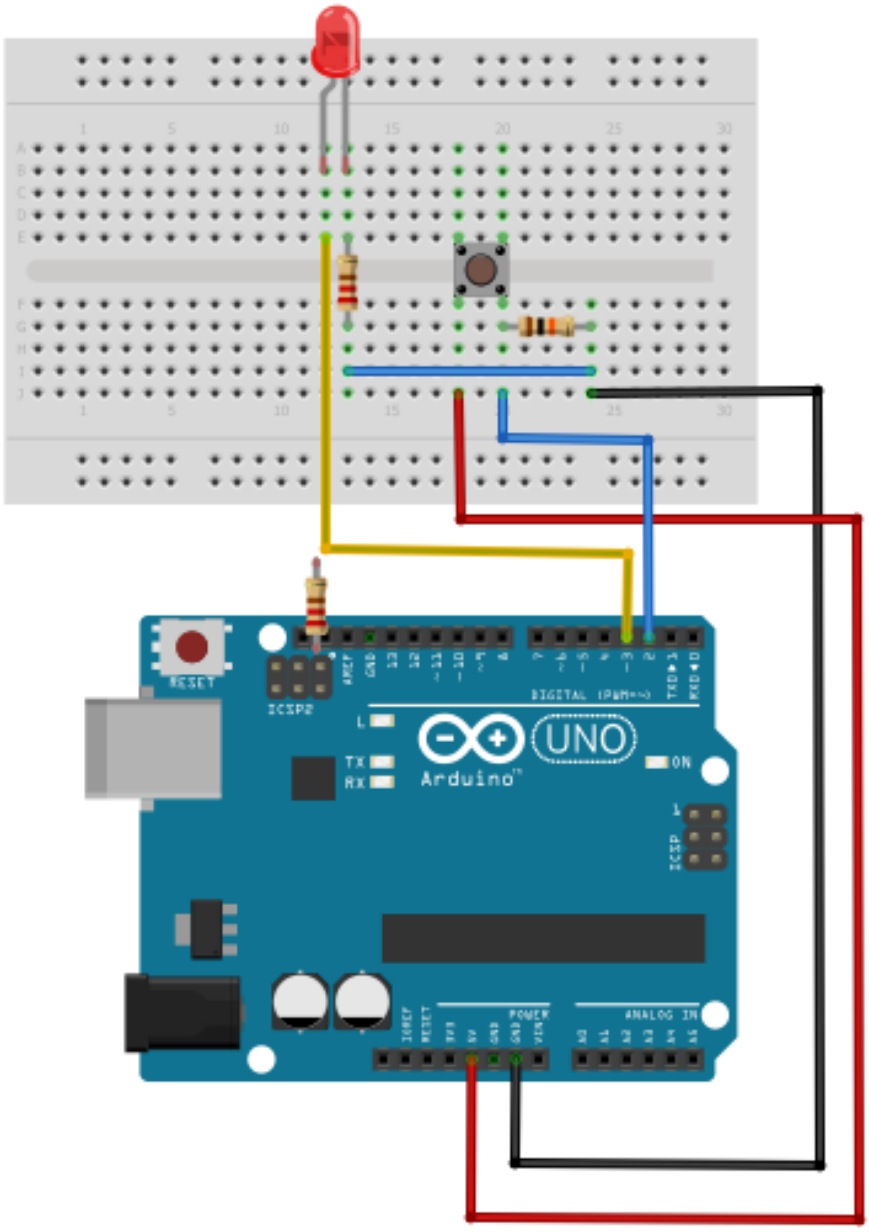
Si se cumple la condición de que el estado de la entrada **PIN2** vale **1** "*Leer PIN 2 =1*"entonces se incrementará la variable **contador** (contador=contador+1) y se activará la salida **PIN3**.

Se ha puesto un bloque "*esperar 0.2 seg*" que introduce una espera de 0.2 seg. Con el fin de evitar que los posibles "rebotes" en el pulsador induzcan errores en la cuenta.

Si no se cumple la condición simplemente mantendremos desactivada la salida PIN3.



Este sería el montaje en protoboard.



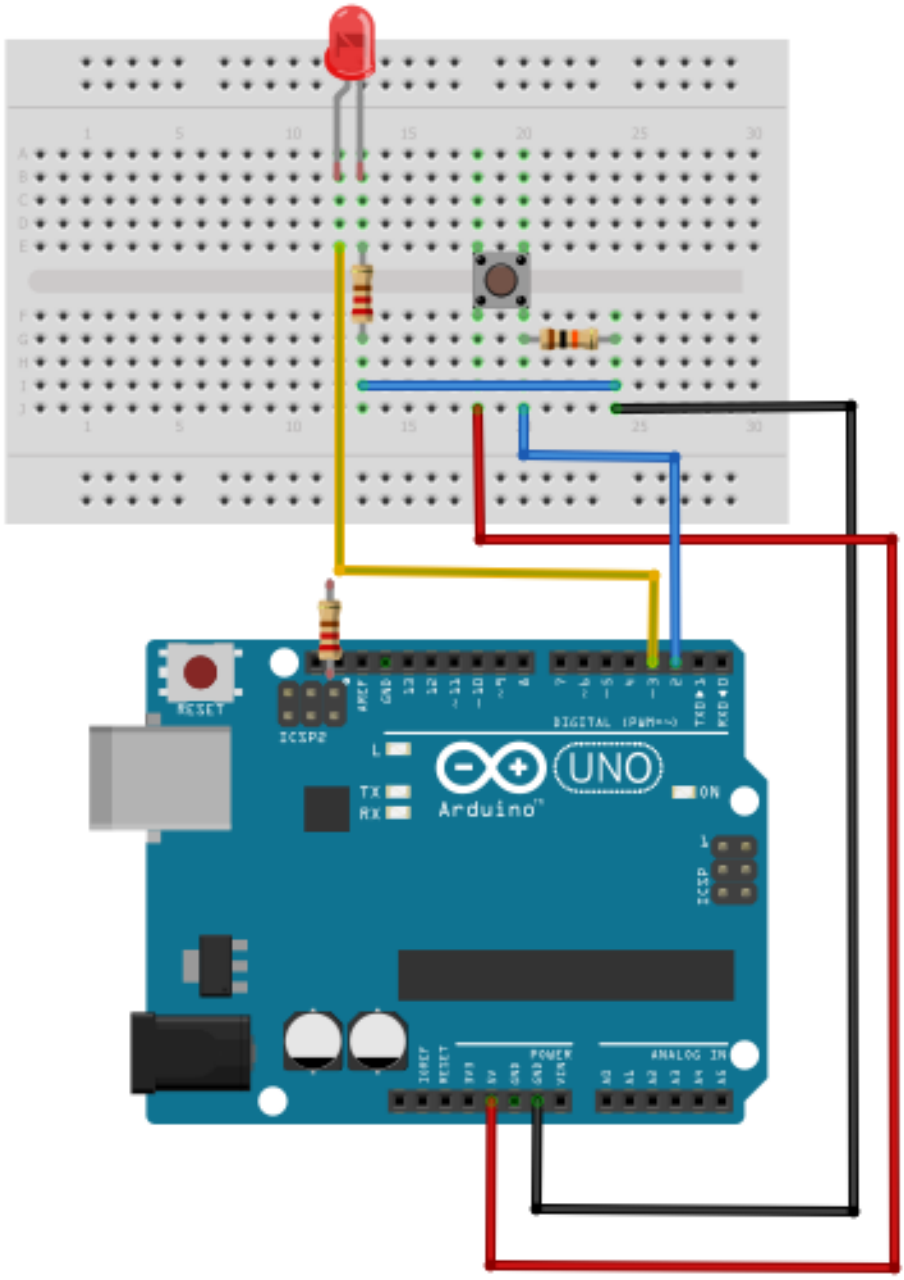
6.5. Contador con puesta a cero

En este ejemplo, muy similar al anterior, hemos decidido realizar la puesta a cero del contador mediante la tecla "espacio". De la misma manera qe en el anterior se trata de recoger los cambios de valor de la entrada digital **PIN2** y realizar la cuenta de los impulsos que llegan. Se habilitará el **PIN3** como salida que refleje estos impulsos que llegan. Crearemos una variable llamada contador en la que se almacenará el valor del estado de cuenta y que se mostrará en todo momento en la pantalla.

La forma de realizar el programa seria sería muy similar al anterior únicamente que debemos testear el estado de la tecla espacio para ello colocaremos un bloque de tipo "si" en el que la condición de ejecución sea precisamente el pulsar la tecla espacio. La instrucción que colocamos dentro de la condición es, *"tecla espacio presionada"*.

Si se cumple la condición de de puesta a cero (tecla espacio pulsada) se pondrá a cero la variable contador.

Este sería el montaje para realizar en protoboard.



6.6. Semáforo

El ejemplo siguiente simula el comportamiento de un sencillo semáforo.

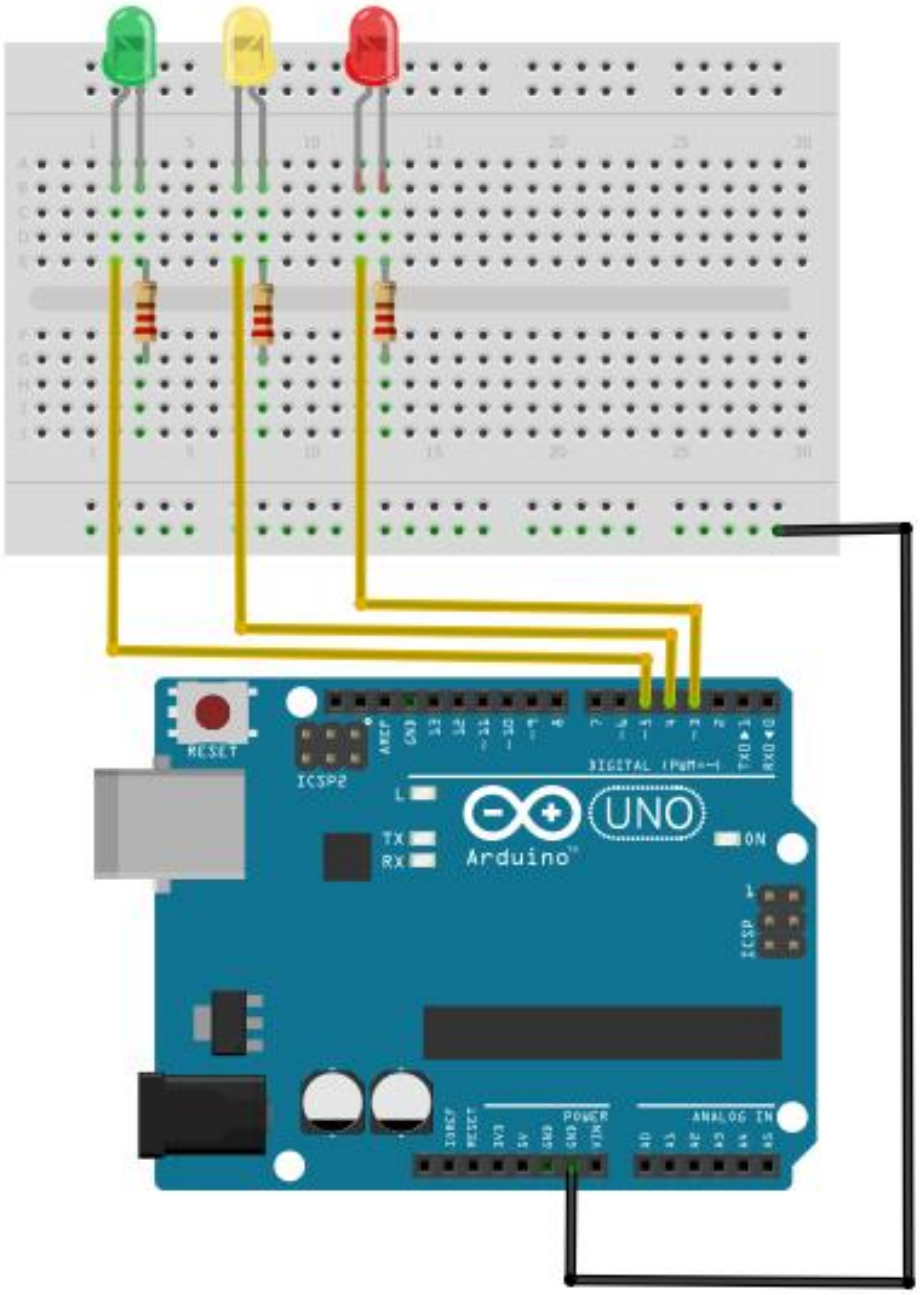
Se designan como salidas digitales los pines PIN3, PIN4 y PIN5.

La programación consiste en realizar la secuencia de encendido y apagado de las salidas con las temporizaciones correspondientes entre cada uno de los tres estados: Estado 1 (PIN3=1, PIN4=0,PIN5=0), Estado 2 (PIN3=0, PIN4=1,PIN5=0) y Estado 3 (PIN3=0, PIN4=0,PIN5=1).

El valor de la temporización se obtendrá a través de la lectura del canal analógico A0. Como la señal varía entre 0 y 1023 debemos hacer un escalado, en nuestro caso dividimos por 1000, lo cual significa que tendremos un tiempo que podrá variar entre 0 y 1,024 seg.

En la siguiente figura se muestra el esquela de bloques del programa.

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



6.7. Servo1.

El control de un servo es una aplicación muy común dentro de las posibles a realizar con Arduino. En este ejemplo vamos a mover un servo que conectaremos en el PIN3. Lo que haremos es crear dos bucles tipo "*for* "que realizaran el movimiento entre 0º y 180º en pasos de 4. El Angulo de giro se asignara mediante una variable que crearemos llamada a que pondremos a cero en el momento de incio para asegurarnos de que comenzamos en la posición de ángulo=0. Los parámetros que debemos colocar al bloque *for* son *i=0 to 45* ya que si los saltos son de 4 grados, para alcanzar los 180 debemos dar 45 pasos

Para consignar el PIN3 como salida servo lo hacemos mediante la instrucción *"Enable PIN 3 como Servo"*.

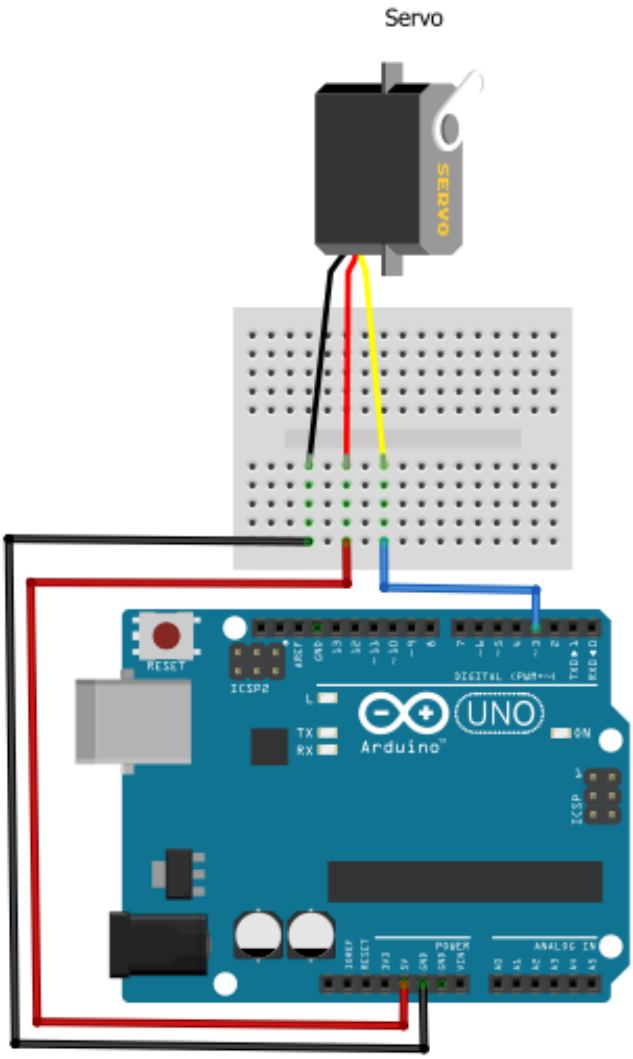
Se establecerán dos estructuras *for* con el fin de que el motor gire en un sentido hasta 180º y

después retorne hacia atrás hasta llegar de nuevo a 0º. La instrucción para mover el servo dentro de la librería s2a_fm es *"Mover Servo en PIN 3 Grados a"*

Se han colocado dos bloques de retardo *"esperar 0.3 seg"* con el fin de poder ver con mas claridad el giro del motor.

Dentro de cada bucle for se debe fijar la variable a a un valor con el fin de que se incremente o decremente: "*fijar a* a *a+4"* y "*fijar a* a *a-4"*para el bucle de incremento de 0º a 180º y para el bucle de decremento de 180º a 0º respectivamente

Este sería el montaje sobre protoboard.



6.8. Servo2

En este montaje el ángulo de giro del servo lo tomaremos de la variable analógica leída en el POIN A0 de Arduino.

Crearemos una variable llamada a y a ella le asociaremos el valor leído del canal A0 mediante la instrucción "*fijar a (Leer PIN(A)0)/8"*. Este valor será el que pondremos en el lugar del ángulo a girar en la instrucción que mueve el servo que tendremos conectado al PIN3 de Arduino *"Mover Servo PIN 3 (redondear a))"*.

Se divide el valor de la lectura analógica por 8 dado que el valor máximo a poner en el ángulo a girar es de 180º de esta manera limitamos el valor a 125º ya que aproximadamente 1023/8 es igual a 125. Después se redondea el valor para que el valor del ángulo sea entero

El montaje en protoboard de este ejemplo es el mismo que el del ejemplo anterior.

6.9. Botón con imagen.

En este ejemplo hemos introducido una imagen de fondo de un shield de Arduino con el que se ha realizado la practica.

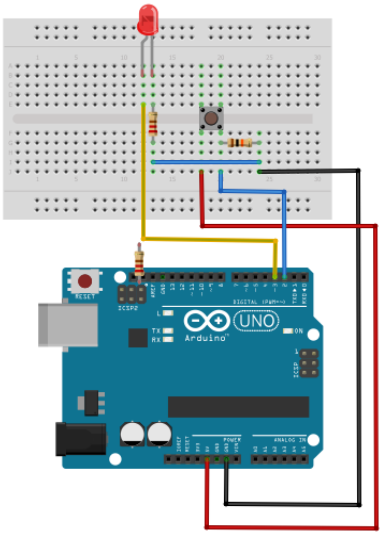
Se trata de activar uno de los Leds de esta tarjeta , concretamente amarillo que está conectado al PIN3.

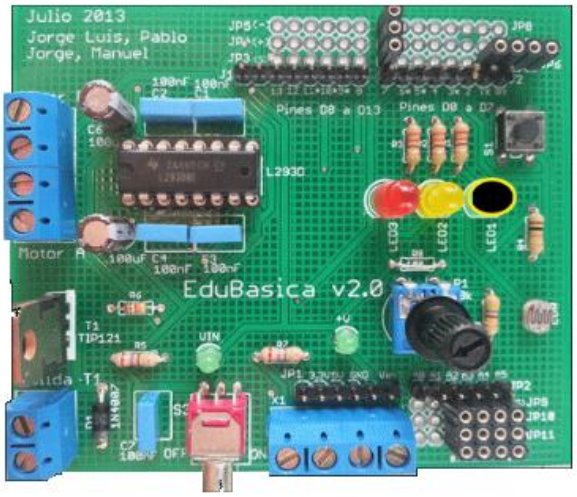
Queremos, además, que cuando se active se cambie de color un circulo que hemos colocado sobre la imagen de fondo que simulara el encendido y apagado de led (de color rojo o negro).

Gobernaremos la salida digital PIN3 en función del estado de la entrada digital PIN2. Esto ya se hizo en el ejemplo “botón”

Se ha creado una variable de nombre led que cambia si valor entre 1 y o dependiendo de si esta activada la salida PIN3 (disfraces on y off) del objeto correspondiente al indicador.

En la siguiente figura se muestra el montaje de la aplicación





7. Referencias

Arduino:	http://arduino.cc/
Snap !	http://snap.berkeley.edu/
Scratch	http://scratch.mit.edu/Scratch
Arduino estándar Firmata	http://playground.arduino.cc/Interfacing/Firmata
PyMata	https://github.com/MrYsLab/PyMata
NewPing Arduino Biblioteca	https://code.google.com/p/arduino-new-ping/
Librería s2a_fm	https://github.com/MrYsLab/s2a_fm

NOTA:

Este manual está basado en el manual de la librería **S2a_fm** realizado por el autor