

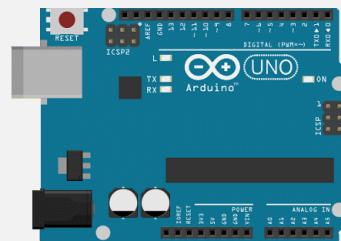
# Rhino-Grasshopper-Firefly + Arduino



+



+



Manual de usuario y practicas

Prof. José Manuel Ruiz Gutiérrez  
Febrero 2014

## Índice

1. Qué es Firefly?
2. Para empezar
3. Librerías de componentes de Firefly
4. Primeros pasos
5. Leer datos de Arduino a través del puerto USB
6. Enviar datos a la tarjeta Arduino a través del Puerto USB
7. Escritura de datos en ARDUINO.
8. Control de una salida de ARDUINO con una tecla, en modo biestable.
9. Lectura de datos de la tarjeta Arduino
10. Formas de tratamiento de una entrada digital de ARDUINO.
11. Lectura y escritura de un valor digital
12. Control de un Motor de CC.
13. Efecto Fading sobre un PIN de salida Analógica
14. Blink avanzado
15. Comparador
16. Control de salida PWM
17. Función AND
18. Leer entrada analógica
19. Leer entrada analógica y escalarla
20. Leer entrada analógica y escalar
21. Control, de circunferencia
22. ANEXO I. Principales bloques de Firefly



José Manuel Ruiz Gutiérrez

Febrero 2014. Ver1

Este trabajo está bajo licencia [Creative Commons Attribution 3.0 Unported License](https://creativecommons.org/licenses/by-nc-nd/3.0/))

## 1. ¿Qué es Firefly?



Imagen propiedad de [Firefly](#)

[Firefly](#) es un conjunto de herramientas de software dedicadas a la reducción de la brecha entre los mundos digital y físico. Permite el trabajo en tiempo real entre [Grasshopper](#) (un plug-in de modelado paramétrico para [Rhino](#) ) y el microcontrolador Arduino. También leerá / escribirá datos desde/hacia internet , trabajando con sensores remotos y otros elementos. Lo que hace único Firefly es su capacidad para convertir un modelo digital, tradicionalmente estático, en una interfaz "en vivo" que estará en comunicación constante con la compleja dinámica del mundo físico. Se pueden modular sensores y actuadores embebidos en un espacio arquitectónico y controlar prototipos utilizando la interfaz de Firefly / Grasshopper.

La barra de herramientas de Firefly Primer y varios archivos de ejemplo se publicaron inicialmente en conjunto con la Asociación de Arquitectos ( Londres) de Estructuras Biodinámicas en un Taller de Verano organizado por la CCA en San Francisco en julio de 2010 ( que también se celebra de nuevo en 2011 y 2012 ) . En el transcurso del taller Firefly era una beta probada por más de 40 participantes. En poco más de un año desde su lanzamiento Firefly se descargó más de 5000 veces por arquitectos, diseñadores, artistas e ingenieros de todo el mundo.

### ¿Cuáles son las principales características de Firefly)

#### **Arrastrar y Soltar:**

Firefly nos permite usar la interfaz de programación visual de Grasshopper, esto le da la capacidad de crear programas y dispositivos interactivos mediante la manipulación de elementos de forma gráfica en lugar de realizar una programación textual. Combina un conjunto especializado de componentes con un protocolo de comunicación nuevo (llamado Firefly Firmata, o firmware) que permiten una retroalimentación en tiempo

real entre los dispositivos de hardware. Todo esto sucede al instante - así que no hay compilación. Su programa se ejecuta en el instante que usted lo crea, por lo que el desarrollo y creación de prototipos de un proceso extremadamente rápido.

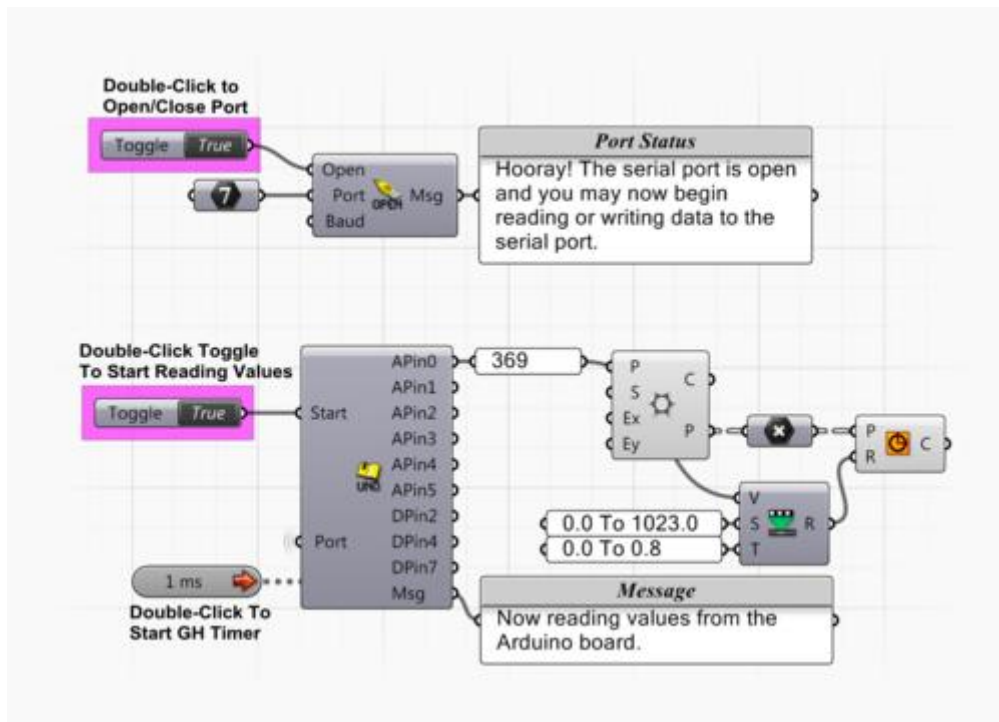


Imagen propiedad de [Firefly](#)

## Generación de Código.

Aprovechando la interfaz de programación visual de Grasshopper, Firefly le permitirá de forma rápida elaborar y probar la “maqueta e ideas de su prototipo” para objetos y dispositivos interactivos. Esto se logra principalmente mediante el envío de los datos de un lado a otro a través del puerto de serie de manera rápida y segura. Pero, ¿qué pasa si usted desea que su dispositivo sea autónomo ... lo que significa que no quiere estar atado a su ordenador mediante un cable USB molesto? Pues en este caso no tendrá problema en general el código para descargar en Arduino y que su aplicación pueda funcionar de manera autónoma sin conexión al puerto USB.

Afortunadamente, Firefly tiene una avanzada función de generación de código que traduce la representación espacial de su código Grasshopper directamente en código compatible con Arduino. Y lo hace todo sobre la marcha. Ahora, usted no tiene que preocuparse de escribir todo ese código manualmente. Simplemente, diseña su prototipo como lo haría normalmente y el generador de código realizara el tedioso trabajo de escribir el código.

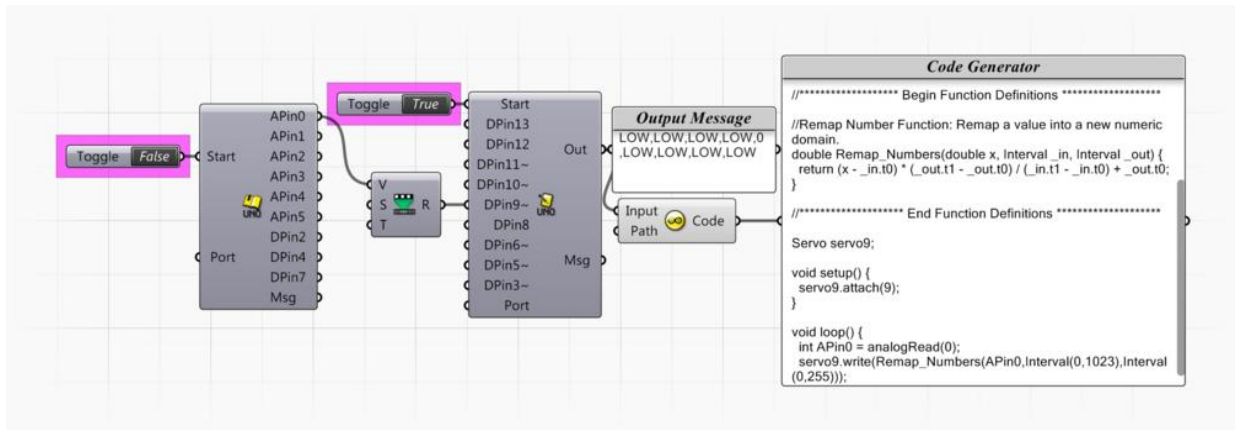


Imagen propiedad de [Firefly](#)

## Conexión a dispositivos físicos en tiempo real

Firefly permite la posibilidad de conectar dispositivos hardware externos e interactuar con ellos. El conjunto de librerías de Firefly tiene componentes que le permiten conectarse a muchos de los dispositivos de hardware más populares, incluyendo los dispositivos de adquisición de datos, teléfonos móviles, cámaras, dispositivos de juego (como el Nunchuck Wii y Microsoft Kinect) e interfaces de audio, por nombrar sólo algunos.



Imagen propiedad de [Firefly](#)

## Herramientas de visión por computador.

Firefly tiene una amplia gama de funciones, herramientas y efectos orientados al análisis de imágenes por computador. Citemos por ejemplo la posibilidad de integrar capturas de video (ahora con soporte para múltiples cámaras), cargar imágenes individuales (. Jpg., Tif., Png., Bmp, etc), reproducir archivos de vídeo y mucho más. Firefly también incluye varias herramientas para el filtrado, efectos gráficos y herramientas de composición para manipular los datos de imágenes en directo. Además, puede crear sus propios filtros personalizados utilizando núcleos de convolución,

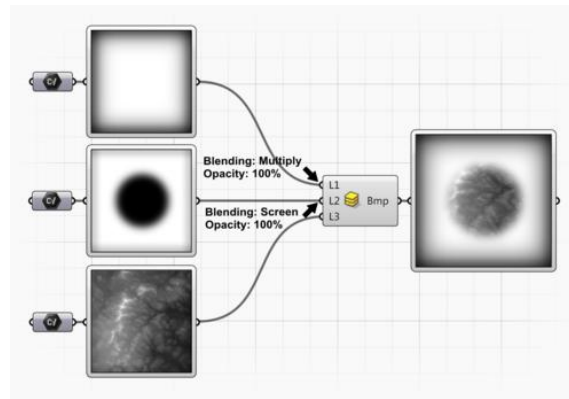
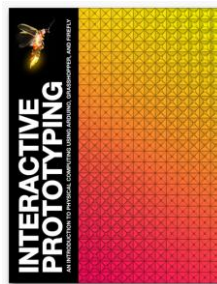


Imagen propiedad de [Firefly](#)

También puede aprovechar y utilizar los algoritmos de visión por ordenador, como el flujo óptico, vectores de gradiente, los vectores de contorno, y el análisis de color para hacer su próximo proyecto interactivo en un éxito visual.

### Guía de usuario

Firefly ofrece una [guía de usuario](#) muy completa que permite el conocimiento y utilización de la librería



En este manual se pretende realizar una aproximación practica al uso de la plataforma Arduino junto con Grasshopper con el fin de facilitar a quienes deseen realizar aplicaciones de interacción entre esta plataforma open hardware y Rhino puedan hacerlo.

La mayor parte del trabajo que aquí presento esta sacado de mis trabajo docente con alumnos de Grado de Diseño en la Escuela de Arte “Antonio López” de Tomelloso (Ciudad Real) España.

## 2. Para empezar

### Requisitos previos

Con el fin de comenzar a trabajar con Firefly, debemos asegurarnos en primer lugar de que en nuestra maquina este bien instalado el siguiente software:

- Rhinoceros 4.0 (o 5.0 WIP beta) <http://www.rhino3d.com/download.htm> Service Release 8 (para Rhino)

<http://download.rhino3d.com/en/Rhino/4.0/sr/download/>

- Grasshopper (versión 0.8 o superior)

<http://download.rhino3d.com/Grasshopper/1.0/wip/download/>

- Arduino IDE (Integrated Development Environment)

<http://arduino.cc/en/Main/Software>

### Paso 1

#### Instalar la librería de herramientas de Firefly.

Para instalar FireFly Build 1.0067 (se hará una sola vez los siguiente):

- Iniciar Rhino y escribir la palabra " grasshopper " en la línea de comandos . Esto abrirá el Editor de Grasshopper.
- Haga clic en Archivo> Carpetas especiales > Carpeta Componentes
- Eliminar cualquier versión previa de la Firefly.gha o Firefly\_X.gha construyen archivos o los archivos libTUIO.dll o C\_sawapan\_media.dll que pueden estar en esta carpeta.
- Abra la carpeta de instalación Firefy que se acaba de descargar (en la carpeta zip) y copiar todo el contenido de la carpeta Build Firefly (Firefly.gha , Firefly\_X.gha , Firefly\_Bridge.dll , libTUIO.dll y C\_sawapan\_media.dll ) en el Folder Los componentes que se abrió desde dentro de Rhino.
- A continuación, Reiniciar Rhino y Grashoopper.

NOTA : Si usted tiene problemas para activar Firefly en Rhino 5.0 beta Abra la carpeta Componentes Especiales (donde acaba de copiar los 5 archivos) y haga clic derecho en cada archivo y seleccione "Propiedades" . En la pestaña General, marque la casilla para " desbloquear " el archivo. Haga esto para cada uno de los tres archivos.

- A continuación , Reiniciar Rhino y Grasshopper

<http://www.fireflyexperiments.com/download/>

La primera cosa que usted deberá hacer es copiar y pegar los archivos Firefly.gha y Firefly\_X.gha (que se encuentran dentro del fichero Firefly zip ) y la librería TUIO.dll ponerlo en la correspondiente carpeta de Grasshopper

Carpetas y ficheros.

- En primer lugar, el lanzamiento de Rhino y escriba la palabra " Grasshopper " en la línea de comandos . Esto abrirá el editor de GH.
- Haga clic en Archivo> Carpetas especiales > Componentes de carpetas. Eliminar cualquier versión anterior de Firefly (ya sea mayor. Archivos gha o la libTUIO.dll ) si ya están instalados en su equipo.
- A continuación, abra la carpeta de instalación de Firefly (en el archivo zip. Que acaba de descargar ) y copiar los tres archivos ( Firefly.gha , Firefly\_X.gha y libTUIO.dll ) en la carpeta Componentes que se abrió desde dentro de Rhino.

Nota: Los tres de estos archivos tienen que copiarse y pegarse en ese directorio para que Firefly se ejecute correctamente. Este proceso sólo tiene que hacer una vez ! Asegúrese de eliminar todos los archivos de instalación de la luciérnaga anteriores ( si está actualizando desde una antes liberar y tenía los archivos de creación almacenados en esta carpeta).

Si había instalado una versión anterior de Firefly y usted había colocado los archivos de creación en el directorio C: Archivos de programa\(\x86)\Rhinos 4.0 directorio\Plug-ins\Grasshopper\Componentes ... por favor retirarlos.

- A continuación, Reiniciar Rhino y Grasshopper - Esta vez cuando inicie Grasshopper, debería ver un nuevo Firefly pestaña en la parte superior del Editor de Grasshopper.
- ¡Felicitaciones! Ahora ya ha instalado la barra de herramientas de Firefly.

Nota: Si usted está teniendo problemas para conseguir la luciérnaga para cargar correctamente en Rhino 5.0 beta, pruebe los siguientes pasos: Abra la carpeta Componentes especial y (en el que acaba de copiar los 3 archivos) haga clic derecho en cada archivo y seleccione "Propiedades". En la pestaña General, marque la casilla para " desbloquear " el archivo. Haga esto para cada uno de los tres archivos.

- Ahora , Reiniciar Rhino y Grasshopper Firefly Tab



## Paso 2.

**Cargar el firmware** “Firefly Firmata” para realizar la comunicación con Grasshopper en la tarjeta Arduino (esta operación solo se requiere hacer una vez):

Descargar e instalar el software IDE Arduino del correspondiente sitio de internet:  
<http://www.arduino.cc/en/Main/Software>

- En primer lugar, copiar/pegar el contenido de la carpeta ' Código Arduino ' (debería incluir 4 ficheros) desde el archivo zip en la carpeta de trabajo de Arduino (generalmente se encuentran en C :\nombre de usuario\Documents\Arduino ).
- Inicie la aplicación 1.0.5 IDE Arduino y abrir el fichero sketch Firefly Firmata: **File> Sketchbook > Firefly\_Uno\_Firmata > Firefly\_Uno\_Firmata.ino .**
- Hacer clic en **Herramientas > Tarjeta** y asegúrese de que ha asignado la tarjeta idónea.
- Hacer clic en **Herramientas> Puerto Serial** y asegúrese de que ha designado a la COM apropiada # (éstas se asignan automáticamente al conectar el tablero de su equipo).
- Tome nota de la **COM** asignado # ya que necesitaremos una vez que abramos Grasshopper.
- Seleccione el botón " Upload" (en la parte superior ).
- Espere unos segundos - usted debe ver los led RX y TX en el parpadeo bordo. Si la carga se realiza correctamente, el mensaje " uploading Done . " Aparecerá en la barra de estado.
- ¡Felicitaciones! Su placa Arduino ya está listo para " hablar" con Grasshopper.

### Créditos.

- La librería C\_sawapan\_media.dll fue creado por Panagiotis Michalatos y Sawako Kaijima - fundadores de Sawapan , un grupo de diseño y consultoría de computación con sede en Londres, Reino Unido . Para obtener más información sobre su trabajo, visite : [www.sawapan.eu](http://www.sawapan.eu)
- La biblioteca libTUIO.dll fue desarrollado por Martin Kaltenrunner y Ross Bencina en el Grupo de Tecnología Musical de la Universitat Pompeu Fabra de Barcelona, España .

## Paso 3

**Instalar WinAVR:** Necesario para Subir Sketches de Grasshopper) WinAVR ahora se requiere para ser instalado con el fin de utilizar el nuevo Subir al Componente de placa de E / S en Firefly.

WinAVR es una suite de herramientas de desarrollo de software de código abierto que incluye avr-gcc (compilador) y avrdude (programador) entre otras cosas. Para instalar la última versión de WinAVR: Ir a: <http://sourceforge.net/projects/winavr/files/WinAVR/> Haga clic en la carpeta llamada "20100110" y descargue el archivo WinAVR-20100110-install.exe. Ejecute el archivo ejecutable y siga las instrucciones.

## Paso 4

Ahora puede **empezar a trabajar** con la herramienta Firefly. Inicie Rhino, y escriba la palabra "Grasshopper" en el símbolo del sistema para iniciar el Editor del Grasshopper. Abra uno de los archivos de ejemplo se encuentran en el 'Grasshopper Ejemplos» carpeta (dentro del archivo zip) y empiece a explorar la “conexión entre el mundo físico y el mundo virtual”.

### Para la versión Firefly\_1.0068 for Rhino 5.0 (released July 17th, 2013)

Si deseamos instalar la última versión de Firefly **Firefly\_Installer\_1.0.0.68\_x86** o **Firefly\_Installer\_1.0.0.68\_x64** las cosas se simplificar dado que el fichero es un fichero instalable que realiza el proceso de manera automática, aunque debemos realizar igualmente la instalación del firmware tal como se ha explicado en el paso 2.

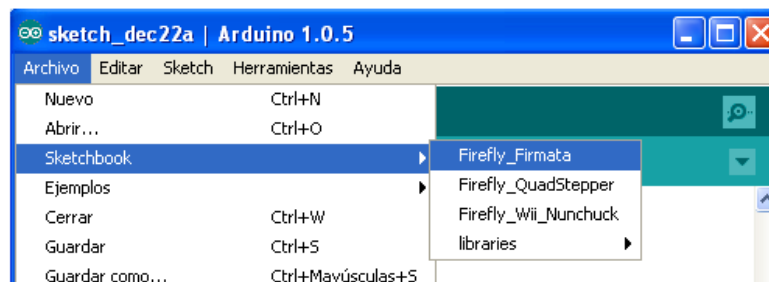
El firmware necesario se instalara en la carpeta ...\**Mis documentos\Arduino** que es en la que por defecto se instalan las librerías y aplicaciones del IDE Arduino. En la carpeta se crearan tres carpetas con tres ficheros correspondientes al firmware:

La carpeta de ejemplos de guardará en ...\**Mis documentos\Firefly\Examples**

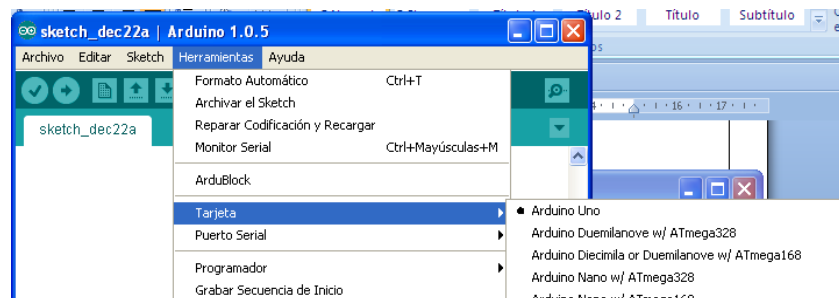
El resto de ficheros (librerías etc..) se instalan en la carpeta que Grasshopper designa para los plugins que se le instalen.

Es muy recomendable mirar los ejemplos que vienen con el software para comprender mejor las posibilidades de la librería Firefly.

## Carga del firmware en Arduino:



## Configuración de tarjeta y puerto.



## Designación de E/S en el firmware

Es importante que sepamos que el firmware que hemos cargado en Arduino designa de forma fija las siguientes entradas salidas:

### EN TARJETAS ESTÁNDAR (Uno, Diecimila , Duemilanove , Lillypad , Mini , etc ).

**ENTRADAS ANALÓGICAS PINS 0-5** Devuelven valores (de 0 a 1023) para sensores Analógicos.

**ENTRADAS DIGITALES PINS 2,4,7** Devuelven 0 o 1 para 3 posibles sensores digitales (botones, interruptores , encendido/apagado, verdadero/falso, etc ).

**SALIDAS DIGITALES PINS 3,5,6,8,9,10,11,12,13** Los **PINS 3,5,6,11** (marcados con ~) pueden actuar como **SALIDAS ANALÓGICAS** y se pueden utilizar para señales DIGITALES, **PWM** y **SERVO**, dependiendo del estado de la entrada de ese pin Firefly. **LAS SALIDAS DIGITALES PINS 8,9,10,12,13** se pueden utilizar para activar salidas digitales y además la 9 para Servo.

### EN TARJETAS DE MEGA (es decir ATmega1280 , ATmega2560 )

**ENTRADAS ANALÓGICAS PINS 0-15** devolverá los valores ( de 0 a 1023 ) para 16 sensores analógicos potenciales

**ENTRADAS DIGITALES PINS 34-41** volverá de 0 o 1 del ; para 8 posibles sensores digitales (botones, interruptores , encendido / apagado , verdadero / falso, etc )

**SALIDAS ANALÓGICAS PINES 2-13** pins se pueden utilizar para digitalWrite , analogWrite o Servo.write dependiendo del estado de la entrada de ese pin Firefly

**SALIDAS DIGITALS PINS 22-33** pueden utilizarse para digitalWrite , Servo.write o analogWrite dependiendo del estado de la entrada de ese pin Firefly

### EN TARJETAS DE LEONARDO

**ENTRADAS ANALÓGICAS PINS 0-5** se establecen para devolver valores (de 0 a 1023) para sensores analógicos

**ENTRADAS DIGITALES PINS 2,4,7** volverá de 0 o 1 del ; para 3 posibles sensores digitales (botones, interruptores , encendido / apagado , verdadero / falso, etc )

**SALIDAS ANALÓGICAS PINS 3,5,6,11** (marcado con un ~ ) se pueden utilizar para digitalWrite , analogWrite o Servo.write dependiendo del estado de la entrada de ese pin Firefly

**SALIDAS DIGITALES PINS 8,9,10,12,13** se pueden utilizar para digitalWrite, Servo.write o analogWrite dependiendo del estado de la entrada de ese pin Firefly

### 3. Librerías de componentes de Firefly.



Bloques principales de comunicación con Arduino



Bloques de sonido



Bloques de comunicación con redes e Internet



Bloques de utilidades




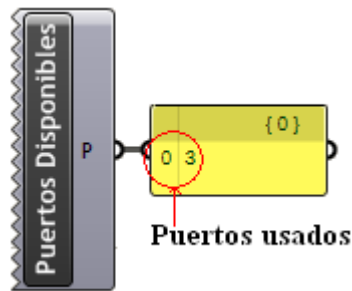
Bloques de visión e imagen.

Estas librerías son explicadas en el ANEXO de este manual.

## 4. Primeros pasos.

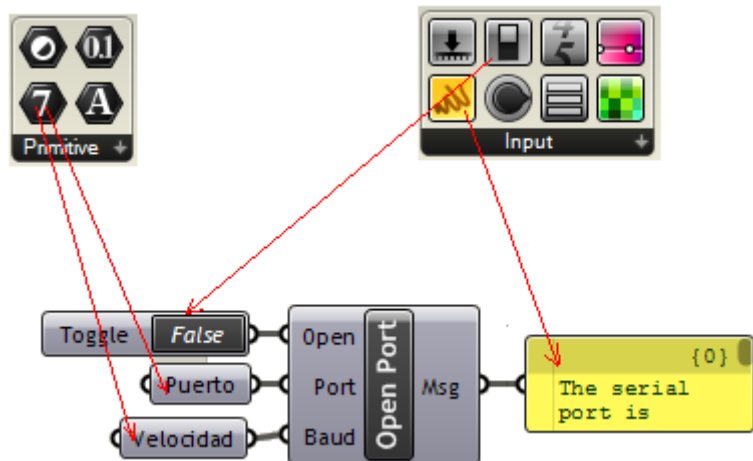
### Verificación de puertos.

La primera operación que debemos hacer una vez que hemos iniciado correctamente la aplicación Rhino+Grasshopper es verificar los puertos que tenemos ocupados en nuestra máquina. Para ello bastará que hagamos uso del bloque Port  que lo que entrega en su única salida son los valores de los puertos detectados.



### Apertura de Puerto






Con el fin de establecer la apertura del puerto USB con el que nos vamos a comunicar con Grasshopper debemos recurrir al bloque .

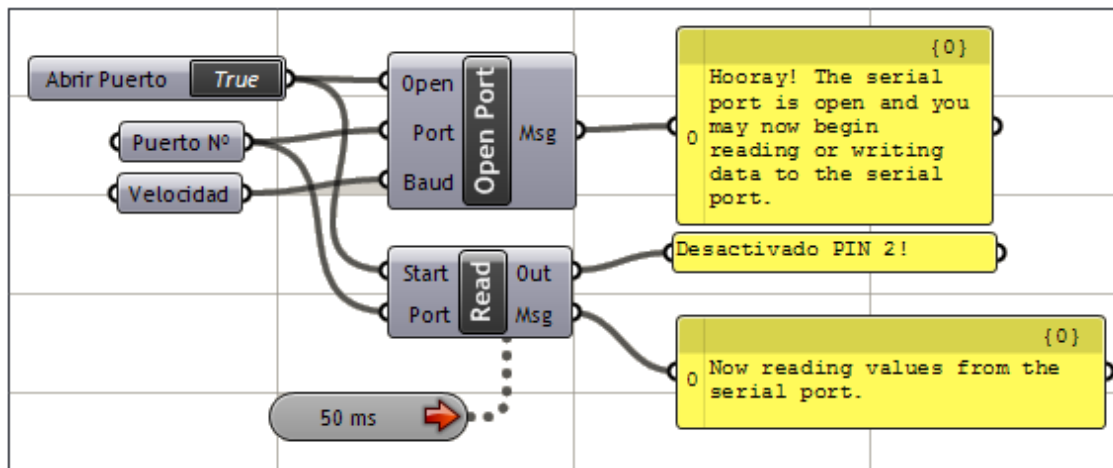


Estas dos operaciones siempre son básicas para realizar la conexión con Arduino, si bien la primera no lo es tanto, siempre que sepamos el puerto por el que nos comunicamos.

## 5. Leer datos de Arduino a través del Puerto USB

*Fichero: 00 leer puerto estado de entrada PIN*

Para leer datos del puerto procedentes de la tarjeta Arduino bastará con recurrir al bloque de apertura de puerto **Open/Close Port** , ya comentado anteriormente y luego al bloque **Serial Read (Generic)**  al que le debemos colocar como entradas un botón de activación  “Abrir Puerto”. Se ha conectado también en la entrada Port del bloque un bloque **Integer**  con el número de puerto “Puerto N°”. Finalmente se ha colocado también en el bloque de apertura de puerto **Open/Close Port** , en este caso un bloque de dato tipo **Integer** para la **Velocidad** de comunicación que estableceremos como 9600.



La lectura de datos se debe hacer cada cierto tiempo por lo que este bloque de lectura **Read** debe tener asociado un bloque Timer que es el que determina cada cuanto tiempo se leerá el puerto (normalmente este tiempo será 50 o 100 ms)



Para que funcione nuestro programa debemos cargar previamente en Arduino con la ayuda del IDE el programa que se encargara de mandar los datos para que Grashopper los lea. Este programa lo único que hará, porque así lo deseamos, es leer el estado de un pulsador colocado en el PIN de entrada PIN2 y en función de si esta activado o no enviar los mensajes:

“Activado PIN 2! “

Si esta activada la entrada (Valor 1)

“Desactivado PIN 2!”

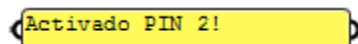
Si esta desactivada la entrada (Valor 0)

El programa será el siguiente:

```
/*  
  Lectura de estado de una entrada de Arduino  
*/  
const int buttonPin = 2;  
int buttonState = 0;  
  
void setup() {  
  Serial.begin(9600);  
  pinMode(buttonPin, INPUT);  
}  
void loop(){  
  buttonState = digitalRead(buttonPin);  
  if (buttonState == HIGH) {  
    Serial.println("Activado PIN 2!");  
  }  
  else {  
    Serial.println("Desactivado PIN 2!");  
  }  
}
```

Obsérvese que la velocidad de comunicación establecida es 9600 Baudios (*Serial.begin(9600);*)

La prueba consistirá en pulsar en el pulsador conectado a la entrada PIN 2 y observar cómo se lee el texto en el Panel correspondiente de la pantalla Grasshopper




Activado PIN 2!




## 6. Enviar datos a la tarjeta Arduino a través del Puerto USB

*Fichero: 01 escribir en puerto activar salida digital*

En este montaje vamos a realizar una prueba de cómo se pueden enviar datos desde Grasshopper a Arduino directamente a través del puerto USB en modo básico (envío de caracteres y valores numéricos).

Esta vez además del utilizar el bloque de apertura de puerto **Open/Close Port**  , que ya hemos comentado pondremos un bloque de escritura en el puerto **Serial Write**


**(Generic)**  . El bloque presenta tres entradas:

**Start** Iniciar y Activar

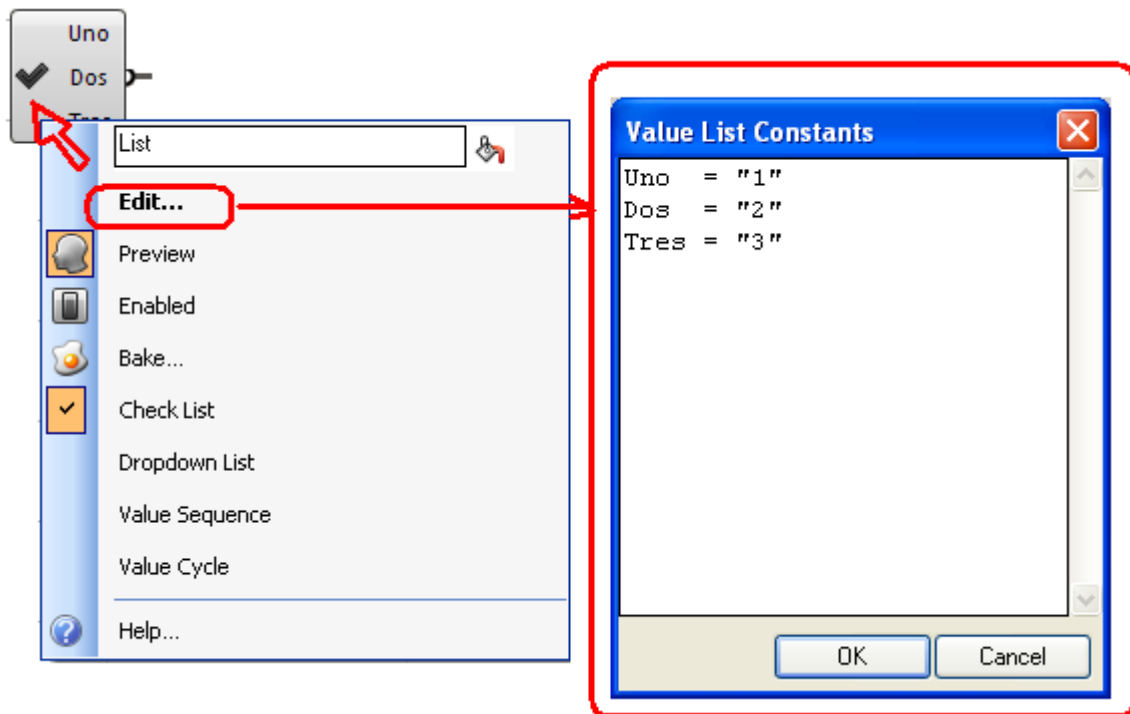
**Data** Entrada de datos a enviar al USB

**Port** Numero de puerto

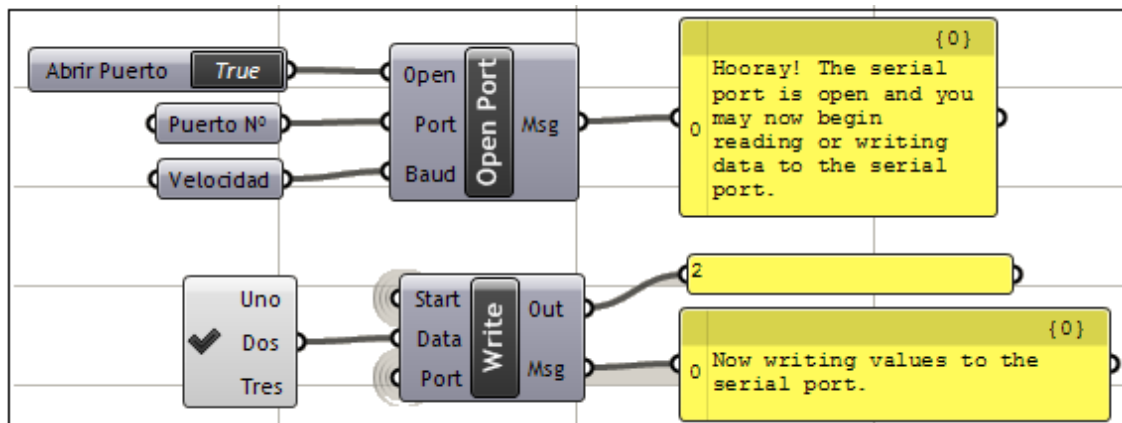
Las entradas **Start** y **Port** se han unido a los bloques “**Puerto N°**” y “**Abrir Puerto**” ocultando luego el cableado.

La entrada Data se ha cableado a un bloque del tipo “**Value List**”  en el que se han puesto las etiquetas Uno, Dos y Tres que se han asociado a los valores numéricos 1,2 y 3 y que dependiendo de las que pulsemos se activaran y apagaran en modo “biestable”

Pulsando con el botón derecho sobre el bloque se obtiene el menú contextual del que se selecciona **Edit** y se escriben los textos de etiquetas y valores que enviara el bloque a l seleccionar cada una de las posibilidades.



La idea es que en función de la selección que hagamos “Uno, Dos o Tres” se envíe a Arduino uno de los valores (1,2 o 3). En Arduino previamente colocaremos mediante el **IDE** el programa correspondiente que lo que hará será leer el puerto y en función de lo que se encuentre activará o desactivará en modo biestable las salidas **PIN3**, **PIN4** o **PIN5**





Programa a guardar en Arduino:

```
/* CONTROL DE SALIDAS A TRAVES DEL PUERTO USB */  
  
int ledPin1= 3, ledPin2= 4, ledPin3= 5;  
int status1 = HIGH, status2 = HIGH, status3 = HIGH;  
int val;  
void setup() {  
  Serial.begin(9600);  
  pinMode(ledPin1, OUTPUT);  
  pinMode(ledPin2, OUTPUT);  
  pinMode(ledPin3, OUTPUT);  
}  
void loop(){  
  val= Serial.read();// lee el valor del puerto  
  if(val!= -1) {  
    switch(val) {  
      case'1': status1 = !status1; break;  
      case'2': status2 = !status2; break;  
      case'3': status3 = !status3; break;  
    }  
  }  
  digitalWrite(ledPin1, status1);  
  digitalWrite(ledPin2, status2);  
  digitalWrite(ledPin3, status3);  
}
```

## 7. Escritura de datos en ARDUINO.

*Fichero: 02 escribir datos en Arduino*

Con esta opción vamos a ver la forma de escribir datos en la tarjeta Arduino. Para ello

recurriremos a la función  si manejamos Arduino UNO o a la función  si se trata de trabajar con Arduino MEGA. En cualquiera de los casos las cosas serán igual.

Veamos la escritura en Arduino UNO

Para escribir pondremos la función en el área de trabajo, suponiendo que ya tenemos realizada la operación de apertura de puerto. Lo que aremos es colocar un botón de Activación que puede ser el mismo que el que utilizamos para abrir el puerto “Activar Puerto”. Seguidamente debemos llevar la salida del bloque “Puerto N°” a la entrada Port del bloque.

Las salidas que tenemos útiles para poder gobernar son las indicadas en el bloque, en principio pueden ser todas digitales pero las salidas **DPin3, DPin5, DPin6, DPin9, DPin10 y DPin11** también podrá ser: **PWM** o **Servo**. La salida PWM acepta valores entre 0 y 255 y la salida Servo entre 0 y 180, (las salidas DPin9 y DPin10 no puede ser de tipo PWM)

Para realizar el gobierno de las salidas bastará poner en las entradas del bloque las señales adecuadas: *Un Number Slider, Boolean Toggle, Integer, Button, Kontrol Knob*, etc. De la librería **Params ->Inputs**.

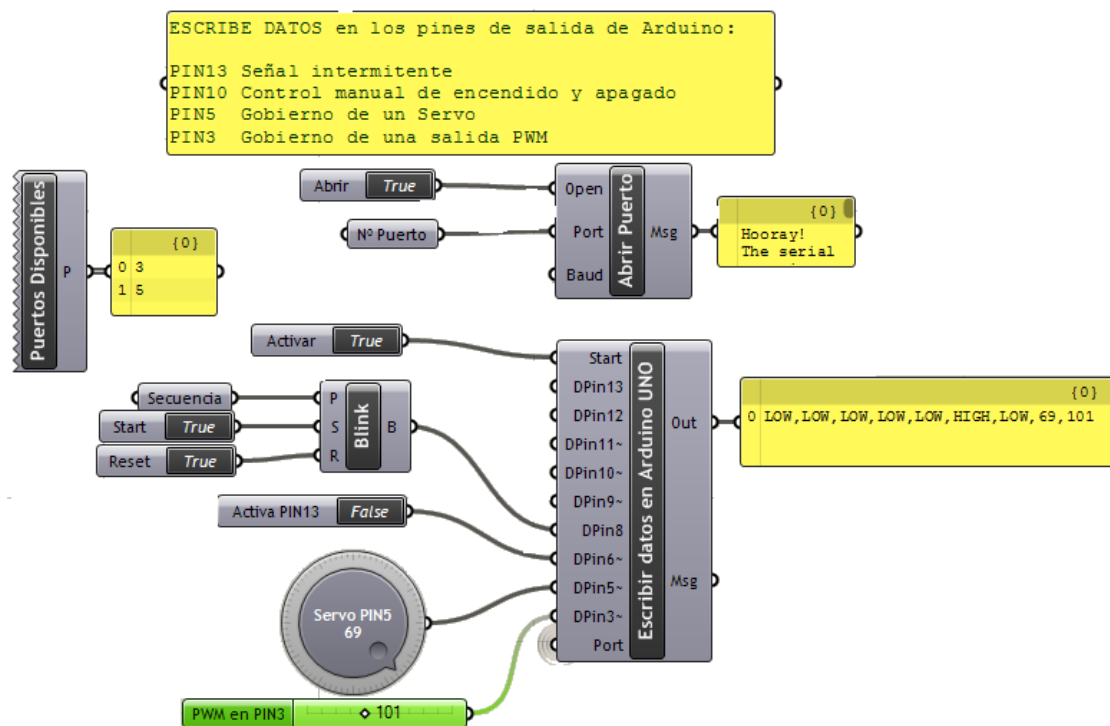


La escritura en la tarjeta Arduino se realiza siempre que cambie un valor de cualquiera de las entradas, por lo tanto no será necesario establecer un valor de tiempo de escaneo del puerto.

En el esquema vemos el montaje de los bloques y las señales de entrada para cada PIN, se ha colocado un Panel de texto de salida para ver el estado de las salidas: 1=HIGH, 0=LOW y los valores numéricos para las salidas 3 y 5.

Para la salida intermitente se ha utilizado un bloque de función de la librería que ya

implementa un oscilador llamado Blink .



El Bloque **Blink** es muy completo ya que podremos escribir una tabla de valores en su entrada **P** (**Secuencia**) que se repetirá siempre que la variable booleana **S** (**Start**) valga 1, con la entrada booleana **R** se reinicia la secuencia.

Por ejemplo podemos poner la secuencia simple de 100 ms en cuyo caso bastara dar a la variable Secuencia el valor **Set Integer=100**, si por ejemplo queremos que se encienda y apague 3 ves a 100 ms, 3 veces a 400 ms y 3 veces a 800ms deberíamos seleccionar el valor **Set Multiple Integers** = Una lista de *100,100,100,400,400,400,800,800,800*


La señal **PWM** que se ha colocado en el **PIN3** se simulara mediante un **Slider** configurado con  $V_{min}=0$  y  $V_{mx}=255$ .

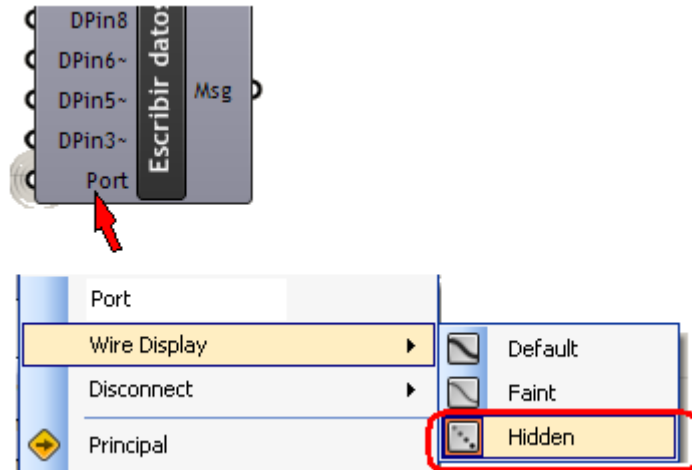
La señal **Servo** conectada al **PIN5** se puede suministrar con un componente **Control Knob** cuyo  $V_{min}=0$  y  $V_{mx}=180$  por tratarse de un servo.

Finalmente para activa de modo manual una salida lo hacemos con un control **Boolean Toggle** directamente conectado al PIN6.

Para realizar la prueba de esta aplicación montaremos el circuito con la ayuda de una protoboard y los componentes adecuados.



Para realizar el encendido y apagado del led colocado en el **PIN 8** mediante el objeto **Blink** podemos hacer uso del bloque **Gene Pol** facilitando la creación de patrones de salida de impulsos de duraciones variables. En la figura se muestra un esquema.

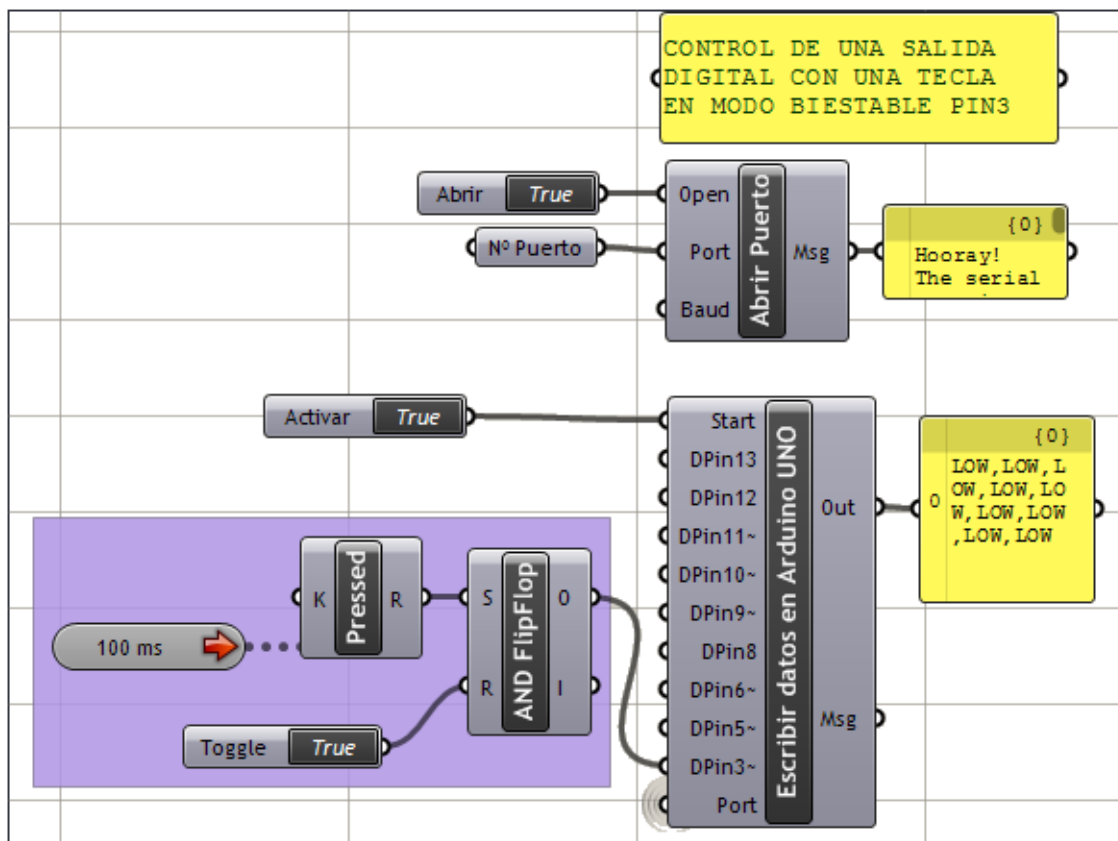
La señal Port del bloque **Uno Write**  está unida al bloque **Nº Puerto** aunque se ha colocado en modo “*oculto*”. Esta opción de ocultar las conexiones en el alambrado es útil para despejar y clarificar nuestro esquema. La opción se consigue en la opción “*Hidden*” de menú contextual que aparece al poner el ratón sobre la entrada en la que queremos realizar esta operación.





## 8. Control de una salida de ARDUINO con una tecla, en modo biestable.

*Fichero: 03 control salida con tecla biestable*

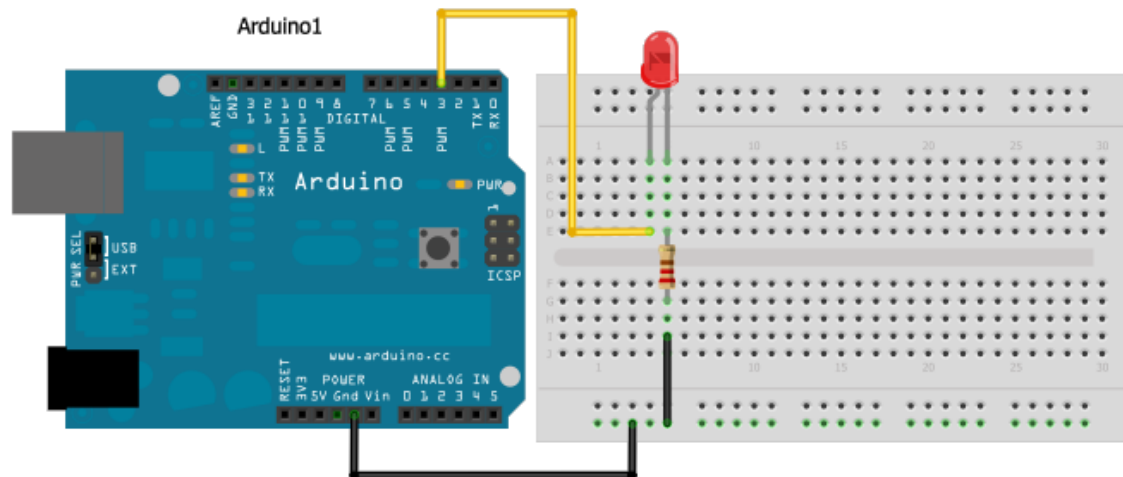
Es posible gobernar una salida digital cuando se pulse una determinada tecla del ordenador. Para esta función recurrimos al bloque de función “**Presed**”  cuya salida es True siempre que se haya pulsado una tecla. A este bloque debemos conectar el bloque de “escaneado temporal”  en el que seleccionaremos el tiempo en ms.



La función biestable que se ha utilizado en este montaje es **AND Flip Flop**  que entrega un valor booleano en función de las entradas que reciba por S. La entrada R puesta en estado False bloquea la entrada S.

No olvidemos que la entrada Port del bloque **Uno Write**  está unida al bloque **Nº Puerto** aunque se ha colocado en modo “oculto”.

Montaje en protoboard.





## 9. Lectura de datos de la tarjeta Arduino

*Fichero: 04 leer datos de Arduino*

Vamos a realizar la lectura de valores tanto analógicos como digitales de la tarjeta Arduino.

Sabemos que disponemos de 6 canales analógicos de entrada cuyo valor oscila entre 0 y 1023 y también disponemos de una serie de pines digitales que ya están configurados como tales en la librería Firefly.

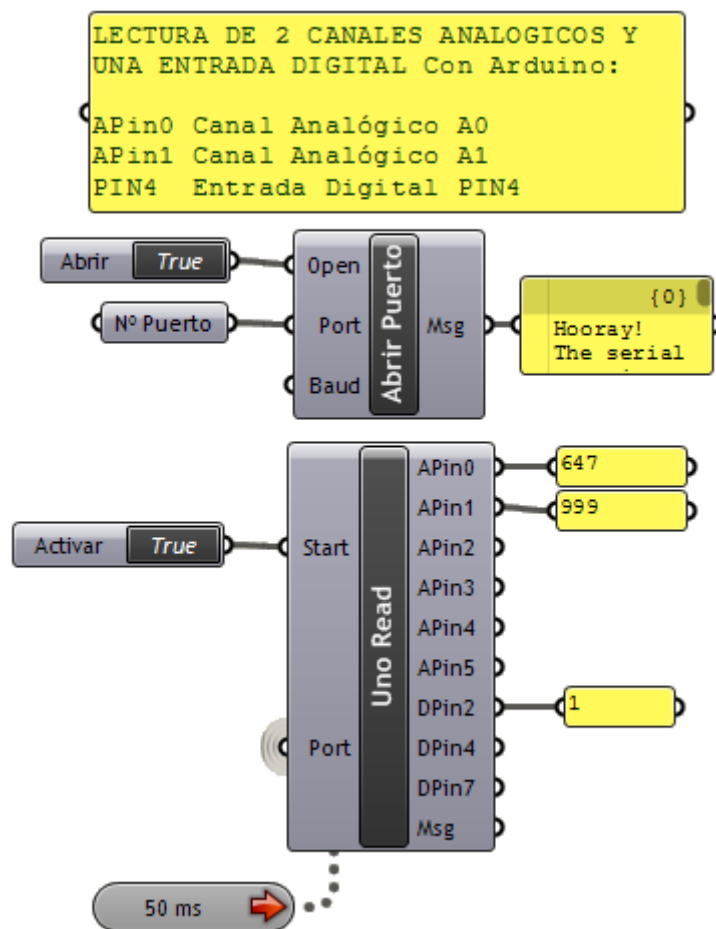
Veamos las señales posibles de entrada:

Señales de Entrada Digital:


**PIN2 PIN4 PIN7**

Señales de Entrada Analógica

**APin0, APin1, APin2, APin3, APin4, APin5**



Como condición previa siempre debemos abrir el puerto en el que se encuentra conectado Arduino. El bloque que se encarga de la lectura de datos de Arduino es el

bloque Uno Read . Este bloque tiene dos entradas, una de activación (booleana) y otra del número de puerto. En nuestro caso esta última entrada la hemos unido a la salida del bloque N° Puerto y hemos ocultado en cable de unión de la manera que ya hemos explicado anteriormente.

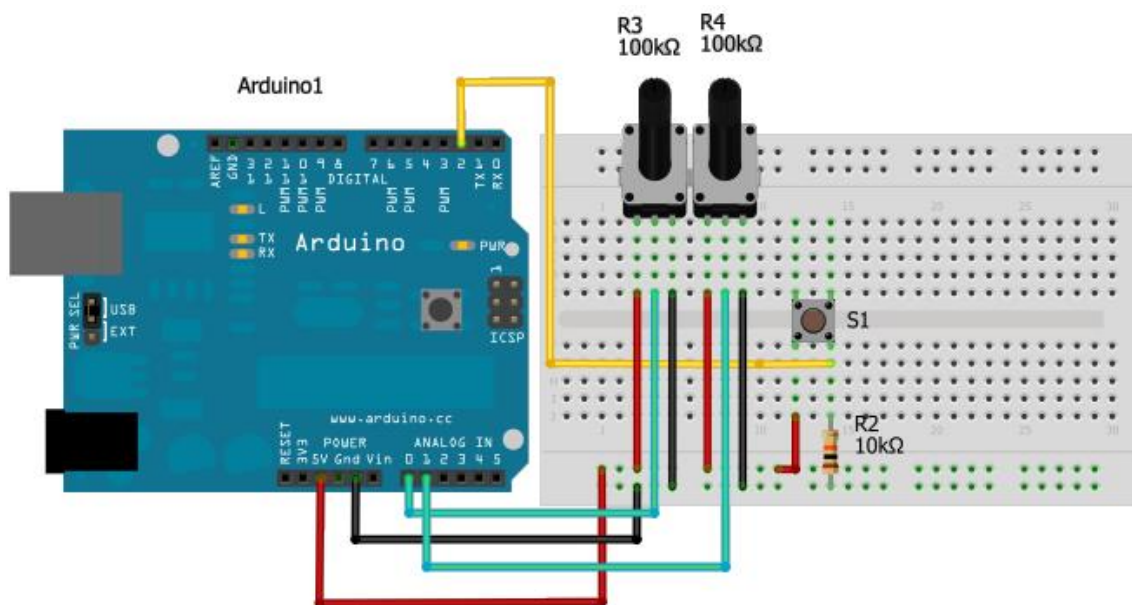
La lectura de datos se debe hacer cada cierto tiempo por lo que este bloque de lectura debe tener asociado un bloque Timer que es el que determina cada cuanto tiempo se leerá el puerto (normalmente este tiempo será 50 o 100 ms)



Las salidas que ofrece este bloque son cada una de las entradas que se leen en la tarjeta Arduino y que ya hemos enumerado.

Se han colocado bloque del tipo **Panel**  para visualizar los valores.

Montaje en protoboard.




## 10. Formas de tratamiento de una entrada digital de ARDUINO.

*Fichero: 05 tratamiento entradas digitales*

Cuando se lee una entrada digital que nos llega de Arduino podemos realizar un tratamiento de esta en distintos modos.

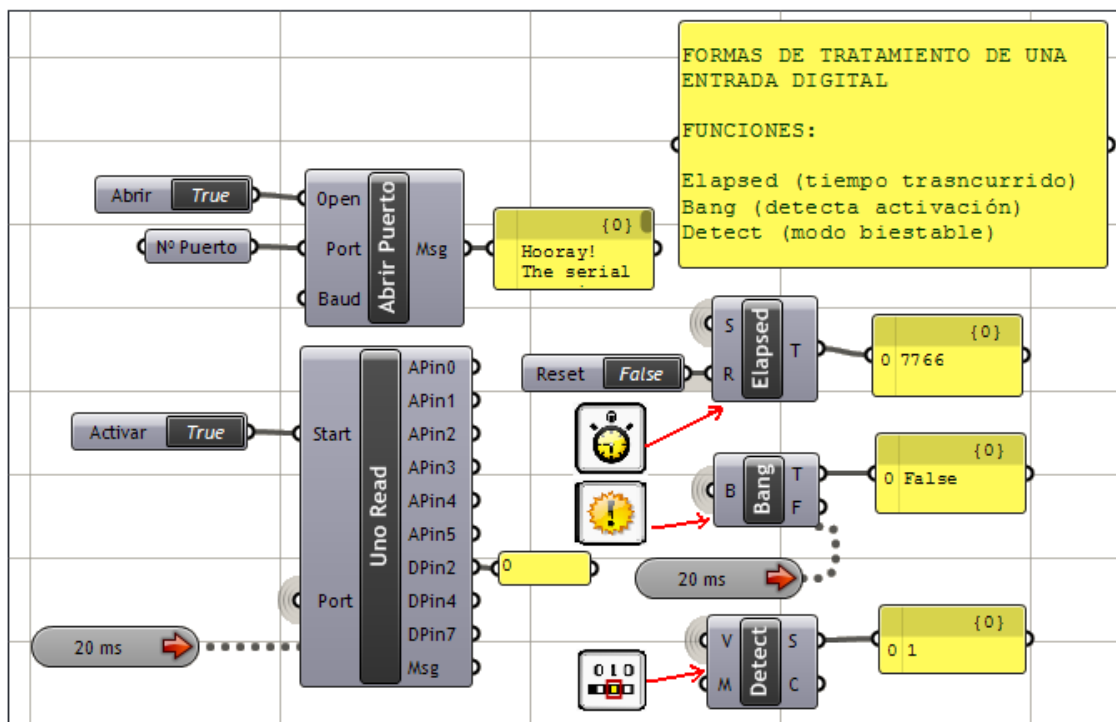
### Tiempo de activación transcurrido..

Con este método podemos contra los milisegundos que se mantiene activa una señal.


Para ello recurrimos al bloque **Stop Watch**  al que hemos etiquetado como **Elapsed**. Este bloque recibe el nivel alto del valor de la entrada **PIN 2** de Arduino y cuenta entregando el valor de la cuenta en su salida, el tiempo durante el que se ha mantenido activo. Con la entrada **R** se reinicia el contador de tiempo


### Detectar flanco de subida de una señal digital

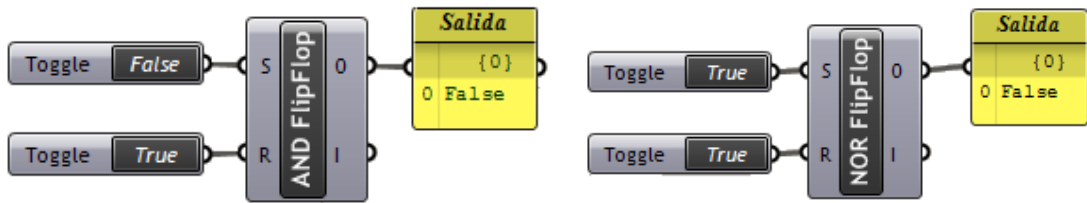
Este método, con la ayuda del bloque **Bang**  sirve para detectar que se ha activado la entrada PIN2 generándose un impulso corto




### Modo Biestable.

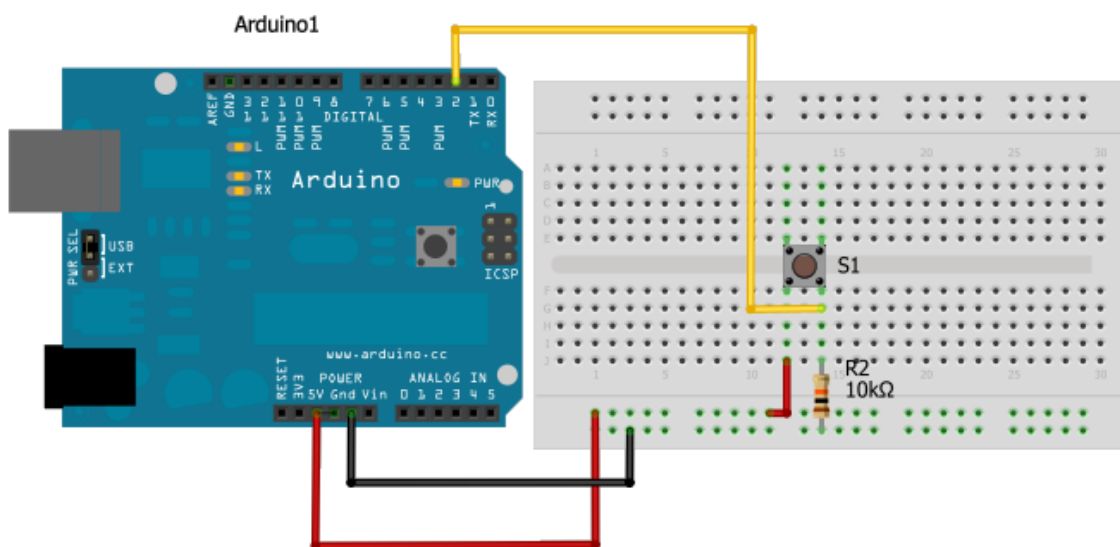
Con este bloque de detección de estado **State Detection**  detectamos que una señal se ha activado y actúa en modo “memoria” o **biestable**, de tal manera que si mandamos

un impulso de activa y si volvemos a mandar otro se desactiva (la activación es por flanco de subida). Este mismo efecto se puede conseguir con un **AND Flip-Flop**  en el que debemos poner en estado **True** la entrada **R** (*reset*)



Con la función **NOR Flip-Flop**  se consigue el mismo efecto pero esta vez se activara el biestable por la entrada **R** manteniendo en *True* la entrada **S**

Montaje en protoboard.

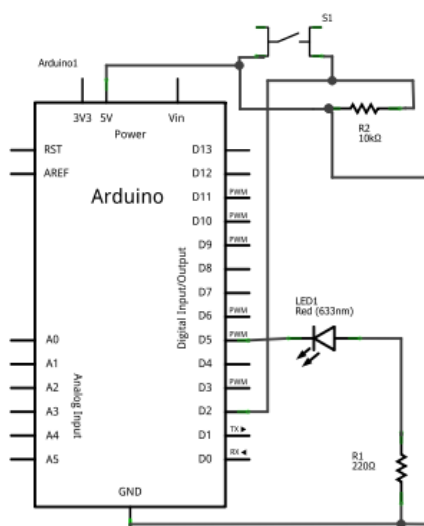
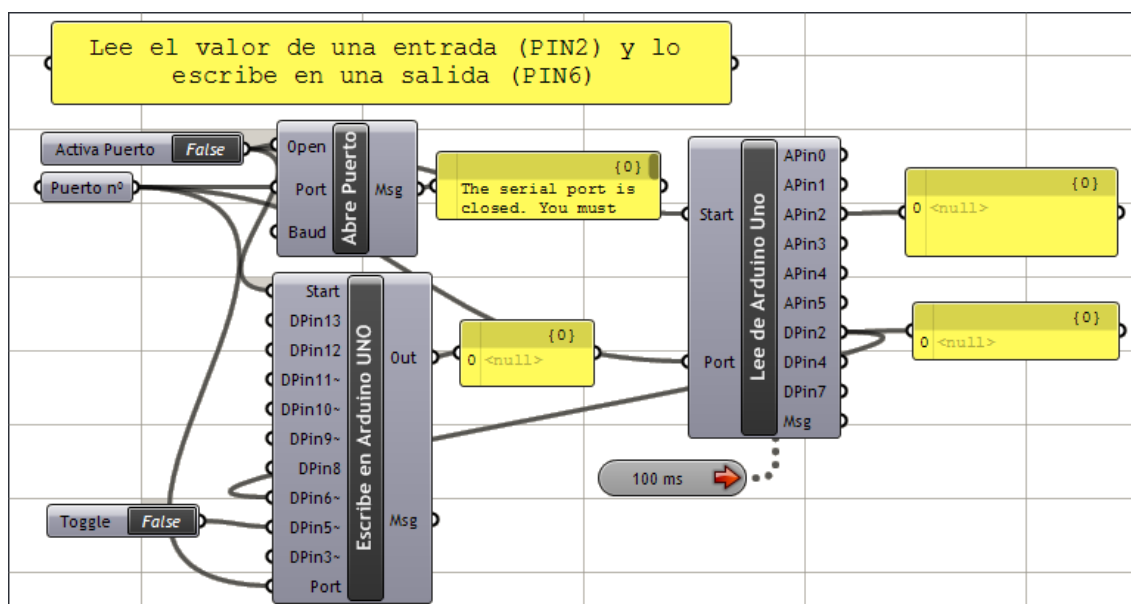


## 11. Lectura y escritura de un valor digital

*Fichero: 06 lee y escribe una señal digital*

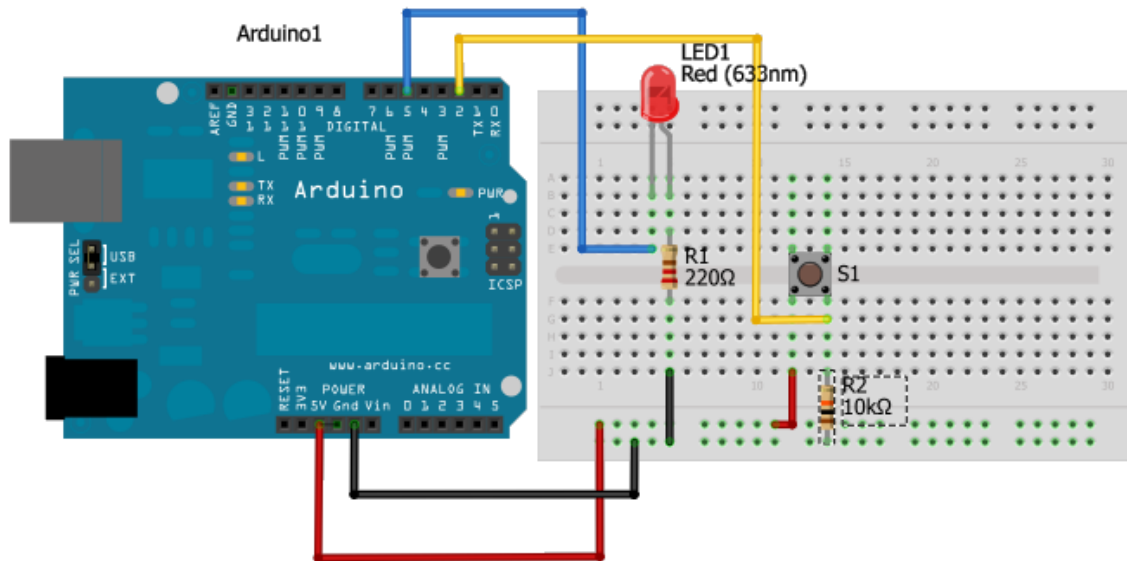
En este ejemplo se realizara la realimentación de una salida digital PIN2 sobre una entrada digital PIN 6.

El montaje es muy sencillo. Bastara colocar los bloques de lectura y escritura de Arduino y cablear la señal de salida PIN2 a la entrada PIN5. Se han colocado bloques Panel para mostrar los estados de las variables.



Esquema eléctrico.

Montaje en protoboard.



## 12. Control de un Motor de CC.

*Fichero: 07 control de un motor de cc*

No pocas veces tendremos que controlar un motor de cc, por ello se pone este ejemplo.

Las señales para realizar el control de un motor de cc. Son:

- Sentido de Giro Derecha PIN13
- Sentido de Giro Izquierda PIN12
- Velocidad PIN11 (En modo PWM)

Para realizar el montaje colocaremos además del necesario bloque de apertura de puerto

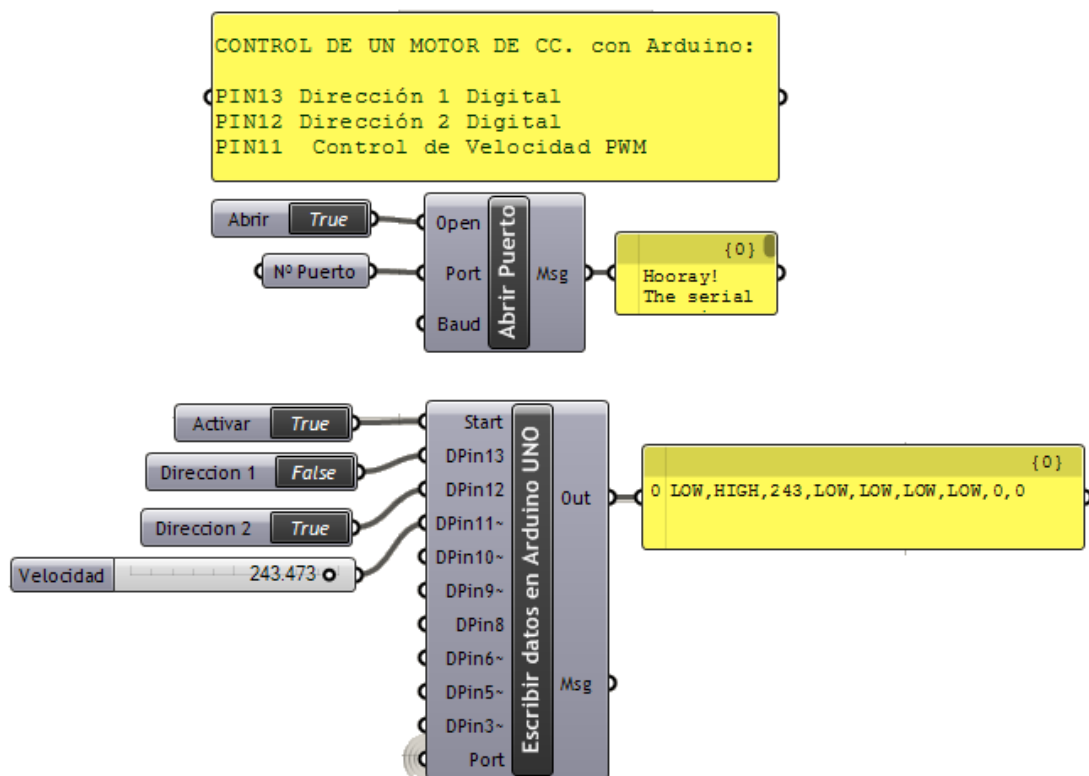


un bloque de escritura

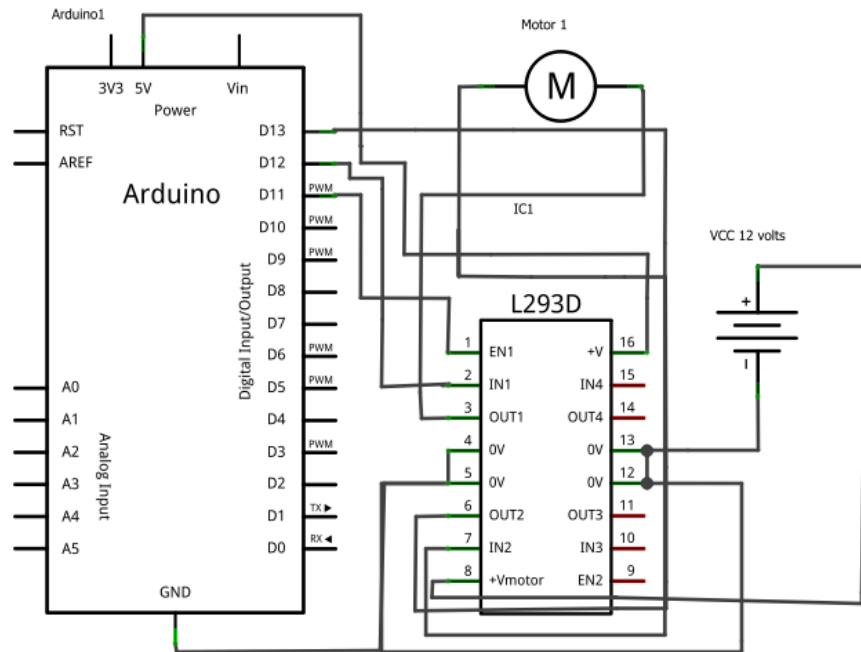


La velocidad será una señal de salida tipo PWM que en nuestro caso hemos seleccionado en el PIN11.

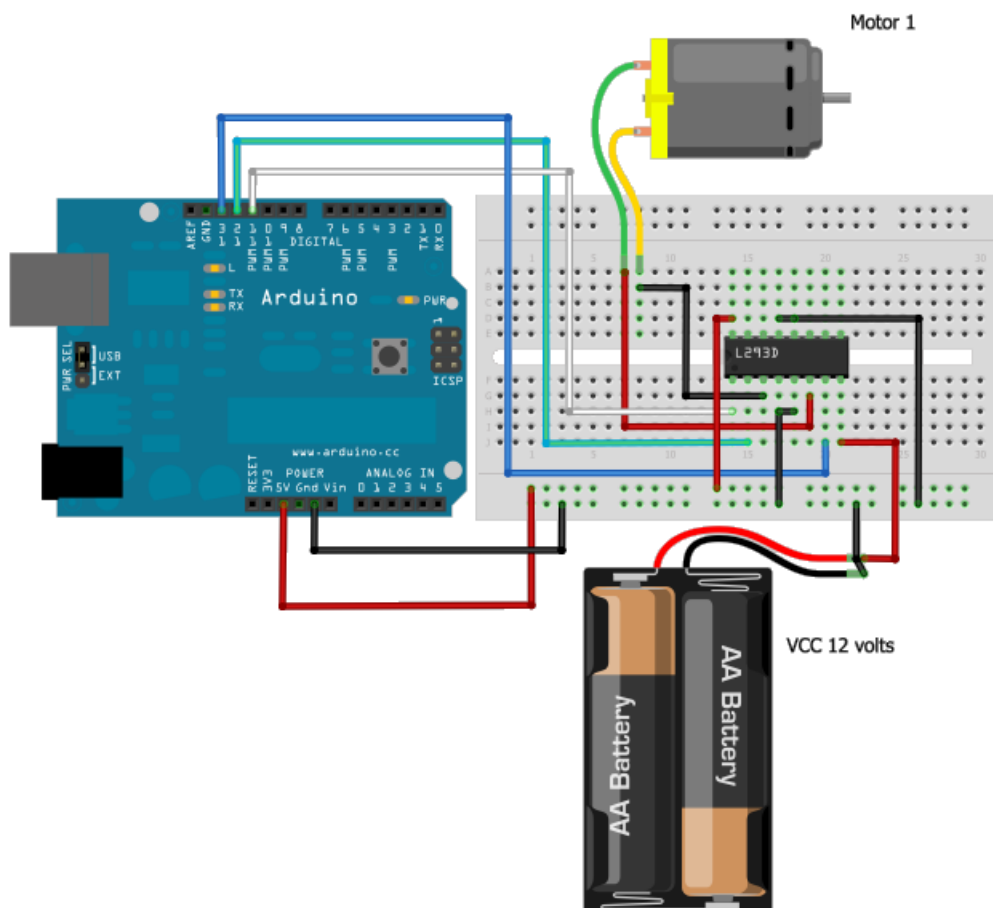
El montaje que se sugiere es mediante el circuito integrado de control de motores Driver L293 que incluye dos driver para control de hasta dos motores de cc.



Esquema eléctrico




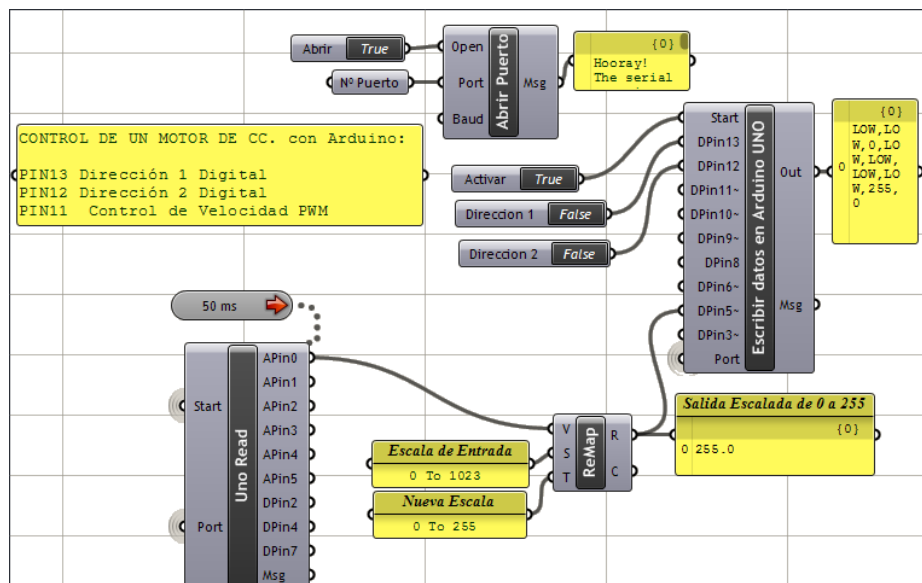
## Montaje en Protoboard



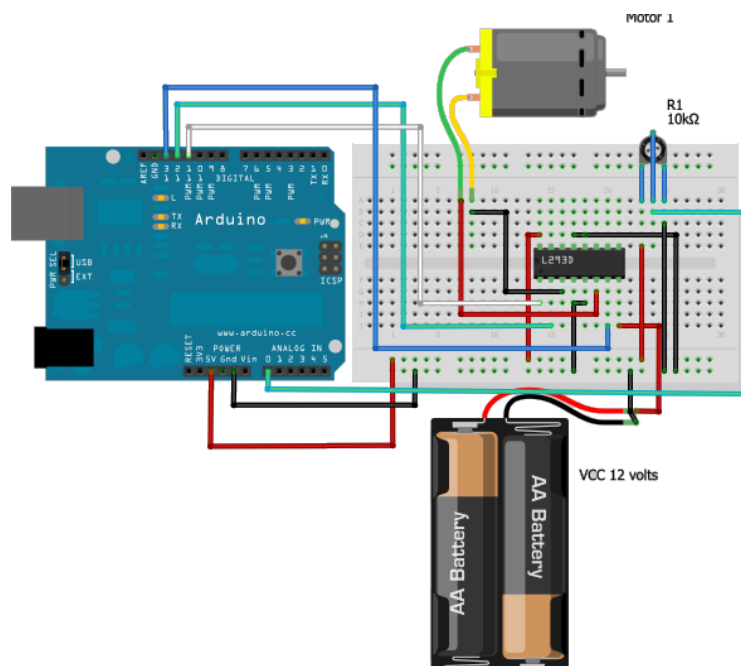


Una posible variación sería que la velocidad se pudiera recoger de un canal de entrada analógica de Arduino, por ejemplo el canal **AN0**. El esquema sería de la siguiente forma.

La señal que se lee del canal **AN0** oscila entre **0 y 1023**, sin embargo, la señal que se puede llevar a una salida **PWM** varia ente **0 y 255**, esto obliga a realizar un escalado de la señal. El escalado se consigue mediante el bloque **Remap Numbbbers**  que aparece en el esquema con el nombre ReMap. Este bloque lo que hace es escalar una señal que oscila entre los valores colocados en su entrada **S** a una señal que oscilara entre los valores señalados entre los colocados en la entrada **T**




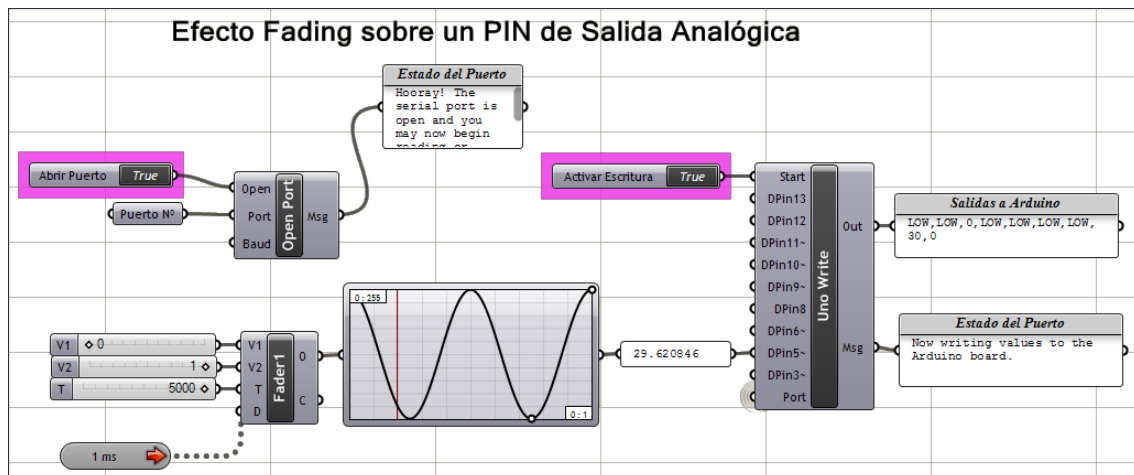
Este es el montaje sobre protoboard




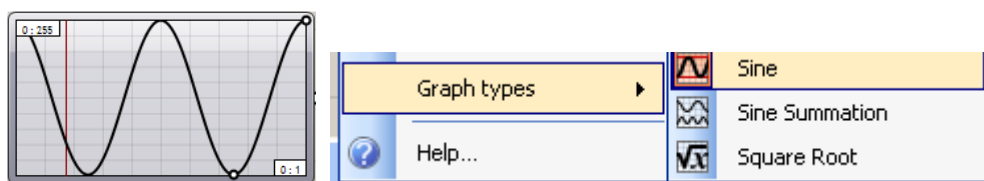
## 13. Efecto Fading sobre un PIN de salida Analógica

*Fichero: 08 Efecto fading*

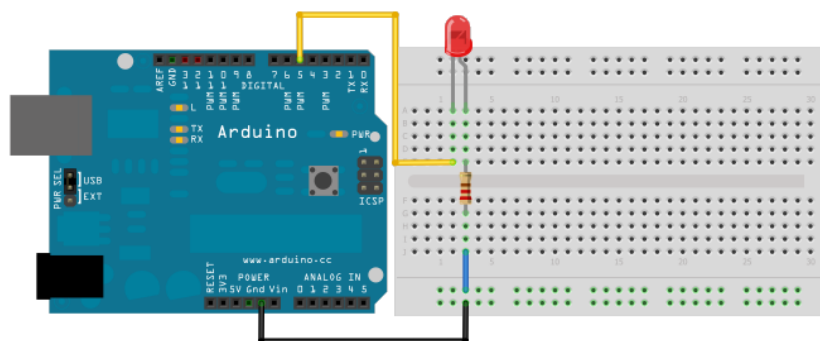
Con este ejemplo vamos a probar un interesante bloque de la librería Firefly que se denomina **Fader One Way** . Este bloque genera una señal en rampa comprendida entre dos valores **V1** y **V2** que va ascendiendo con una velocidad **T** (en ms)



En el montaje se genera la rampa entre los valores de **0** a **1** y luego se inyectan estos valores en una función **Graph Mapper**  que se ha configurado con la función seno con un valor oscilante entre **0** y **255** para desde la salida de este inyectar la señal en el **PIN 5** al que previamente le hemos configurado como PWM



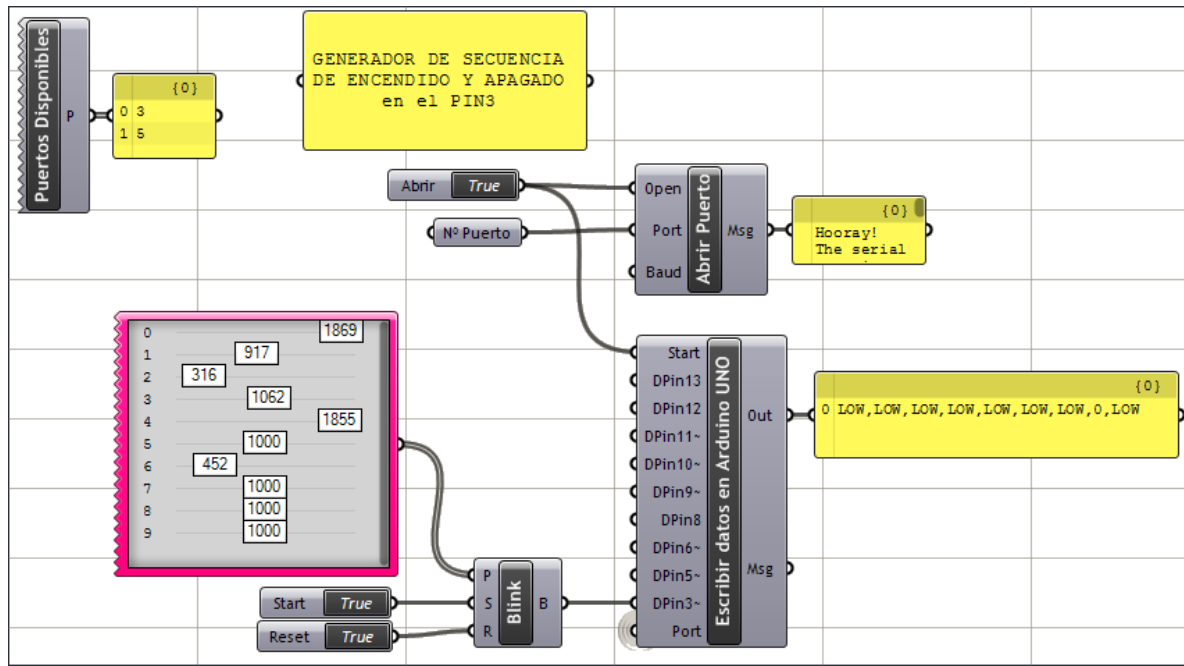
Este es el montaje de la práctica.



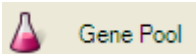
## 14. Blink Avanzado.

*Fichero: 09 blink avanzado*

En este ejemplo vamos a realizar el gobierno de una salida de Arduino **PIN3** con una secuencia de encendido y apagado configurable que se repetirá de modo continuo.

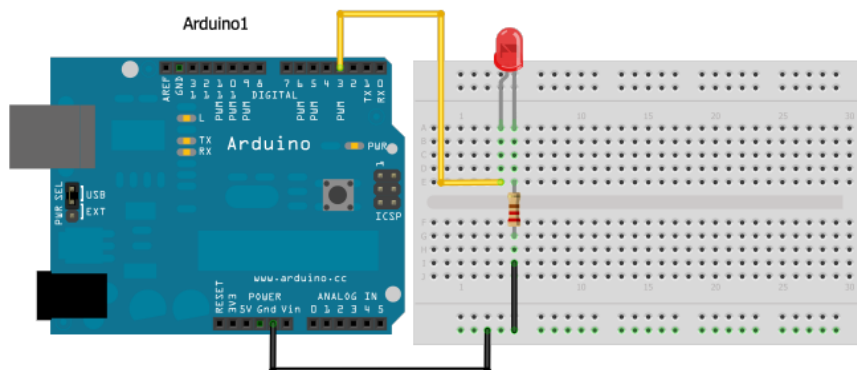


El bloque que utilizaremos para este montaje es un bloque **Blink** en el que en su entrada **P** se ha colocado un bloque **Gene Pool** que permite crear una lista de valores



en la que se establecerán los intervalos de encendido y apagado. En nuestro caso hemos creado una lista con 10 valores (de 0 a 9) Los valores son recogidos por el bloque **Blink** en forma de ms. Los valores pueden ser modificados simplemente moviendo el ratón sobre los cursores que indican el valor.

Realización del montaje en protoboard.

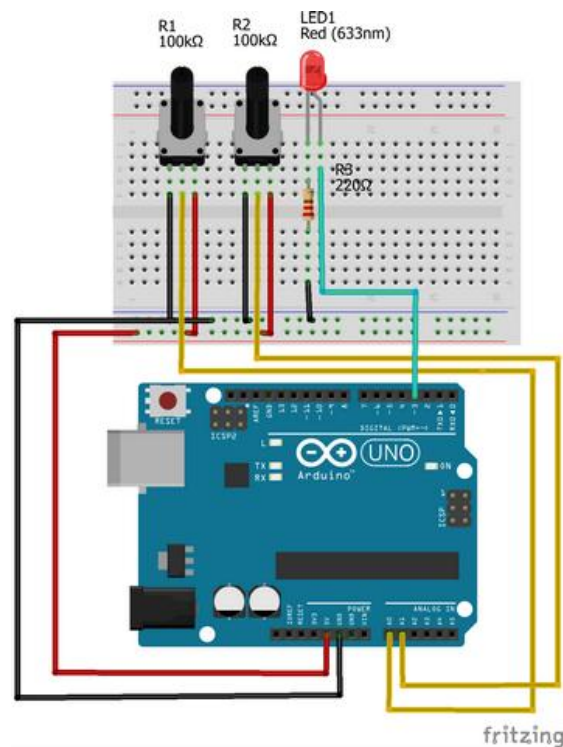
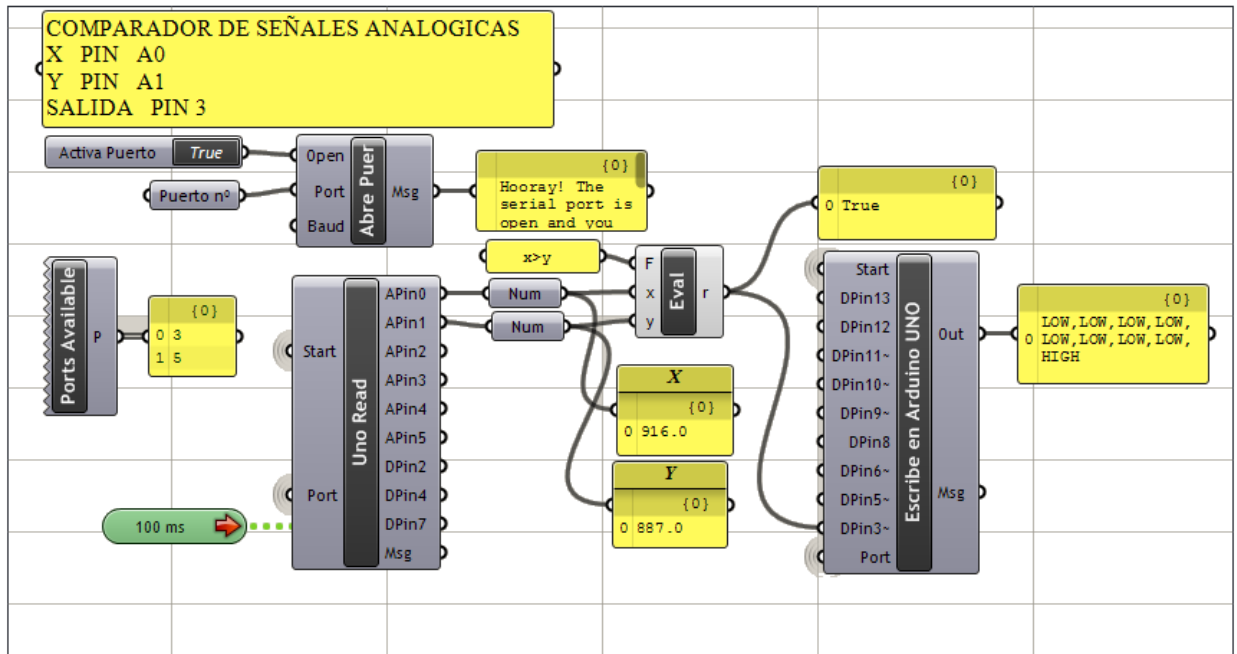


## 15. Comparador

*Fichero: 10 comparador*

En ocasiones necesitamos realizar una comparación de valores de señales analógicas para gobernar una salida. Este es el caso de un termostato.

Se recurre al bloque

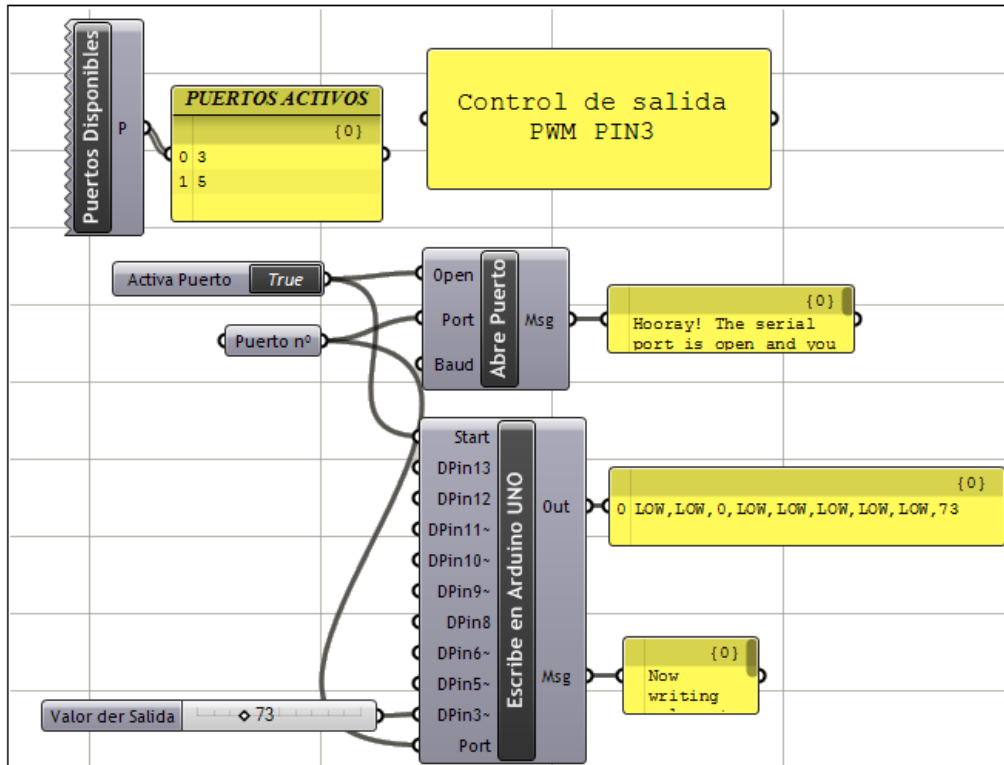


Montaje en protoboard

## 16. Control de salida PWM

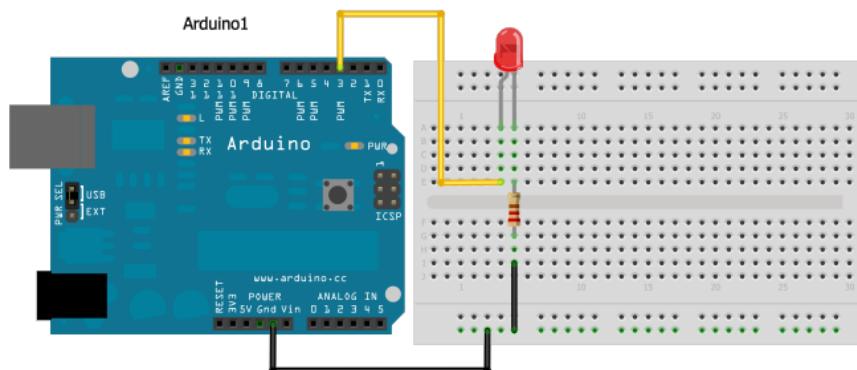
*Fichero: 11 control de salida PWM*

Este ejemplo muestra como realizar el control de una salida analógica en modo PWM. Ya hemos realizado alguna aplicación en la que se ha controlado una salida de esta manera, por lo tanto no merece mas comentarios.



Se realiza el montaje y no debe olvidarse configurar la salida PIN3 como salida PWM, para ello bastará con ponerse sobre la etiqueta de la señal en el bloque y con el botón derecho aparecerá un menú en el que seleccionamos el modo PWM para esta salida. El valor de entrada debe estar comprendido entre 0 y 255

Realización del montaje en protoboard.

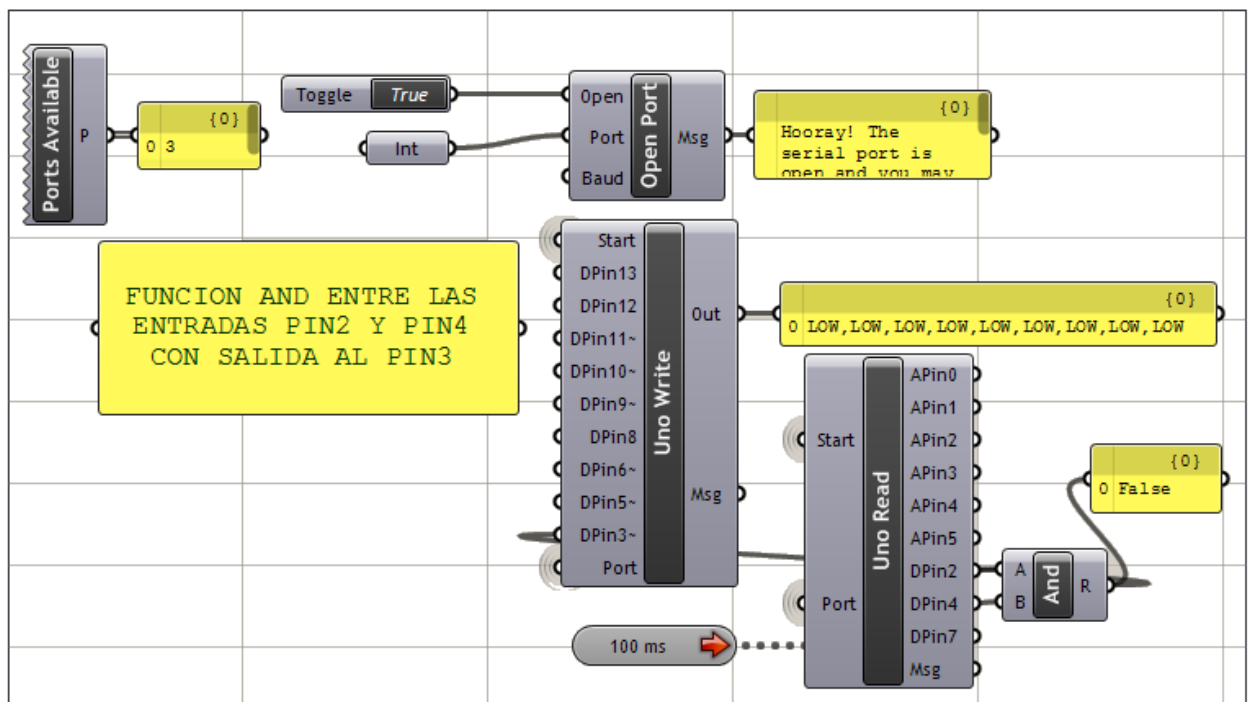


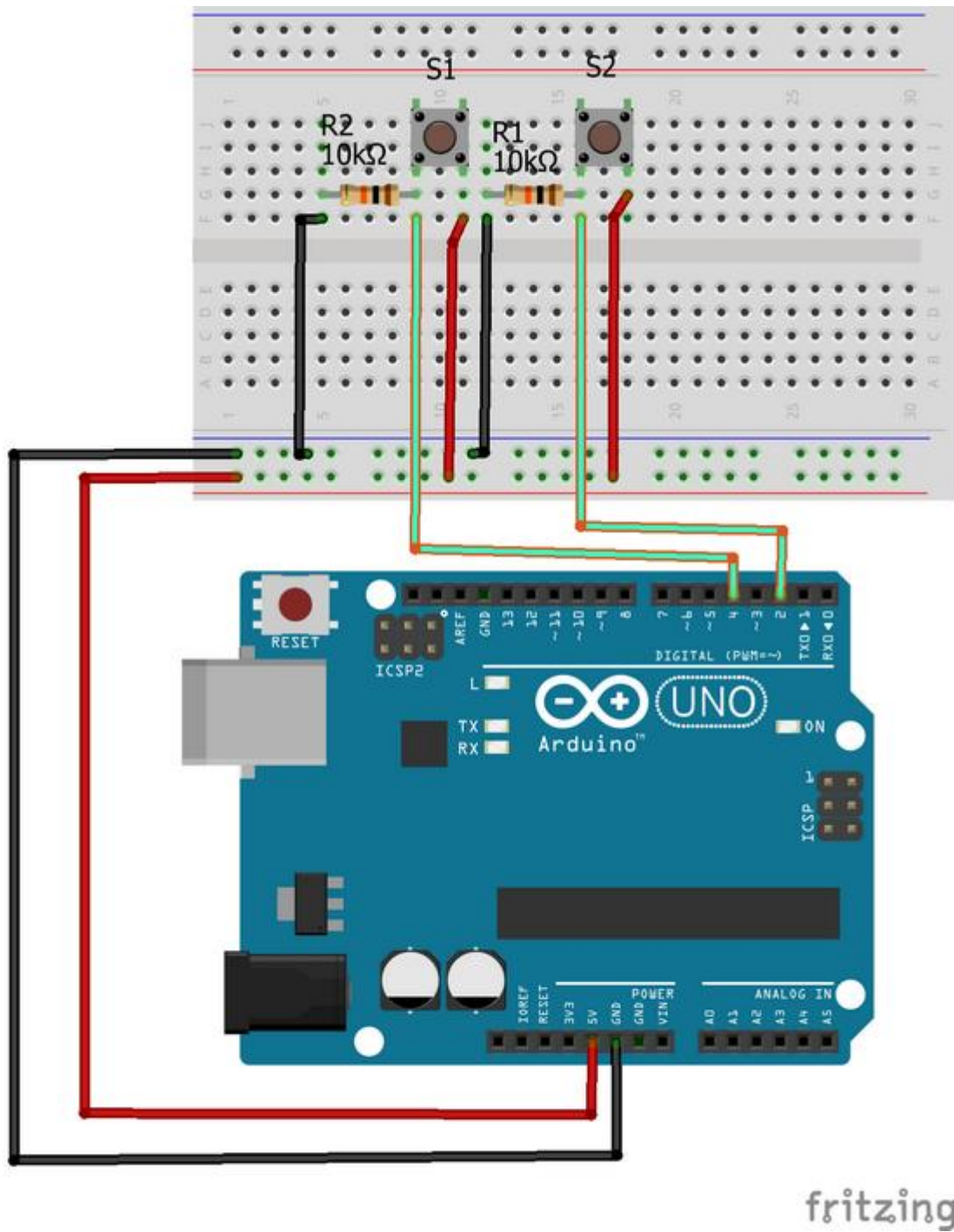
## 17. Función AND.

*Fichero: 11 función AND*

En el siguiente ejemplo se trata de recoger el estado de dos entradas de la tarjeta Arduino PIN2 y PIN3 y realizar con ellas la función AND.

Nos ayudaremos del bloque AND de la librería de funciones **Maths** (sección **Operators** -> **Gate And**).





Montaje en protoboard

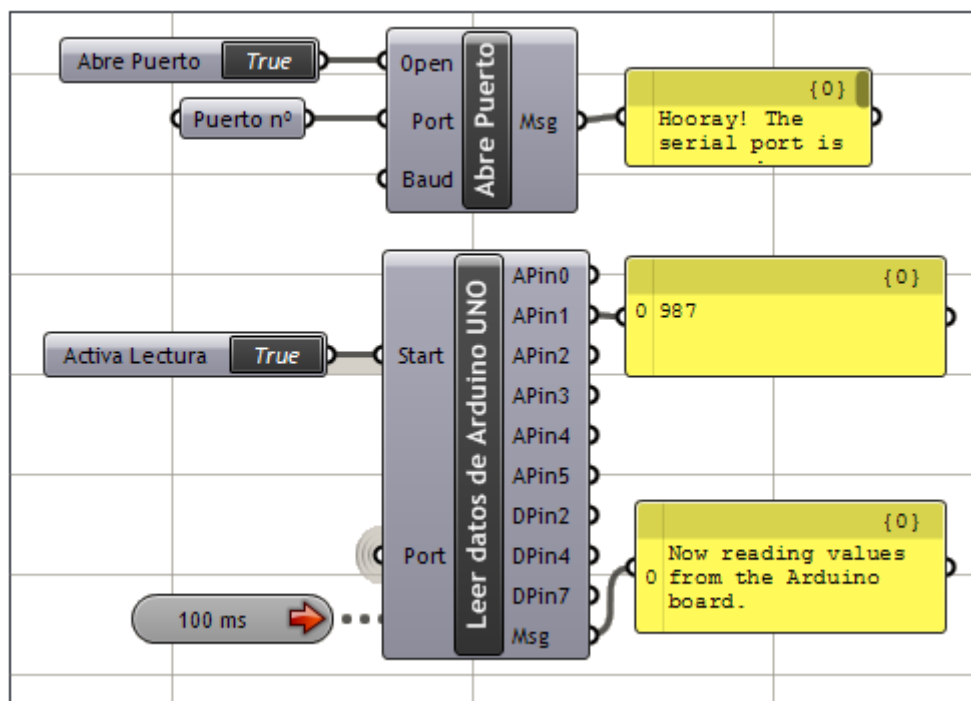
## 18. Leer entrada analógica.

*Fichero: 12 leer entrada analógica*

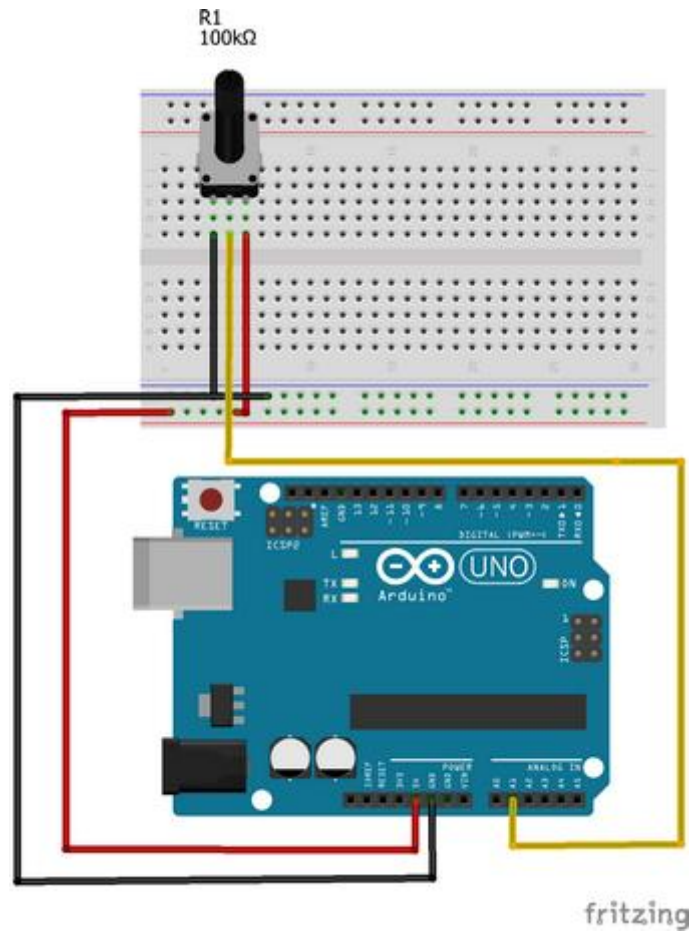
Con este ejemplo leemos un canal de entrada analógica de Arduino, concretamente el canal A1.

Debemos tener en cuenta la necesidad de realizar el scan del puerto de RS232 cada cierto tiempo, en nuestro caso cada 100ms. Sin este bloque no es posible la lectura.

Este bloque está en la librería **Params** (sección **Util -> Timer**)





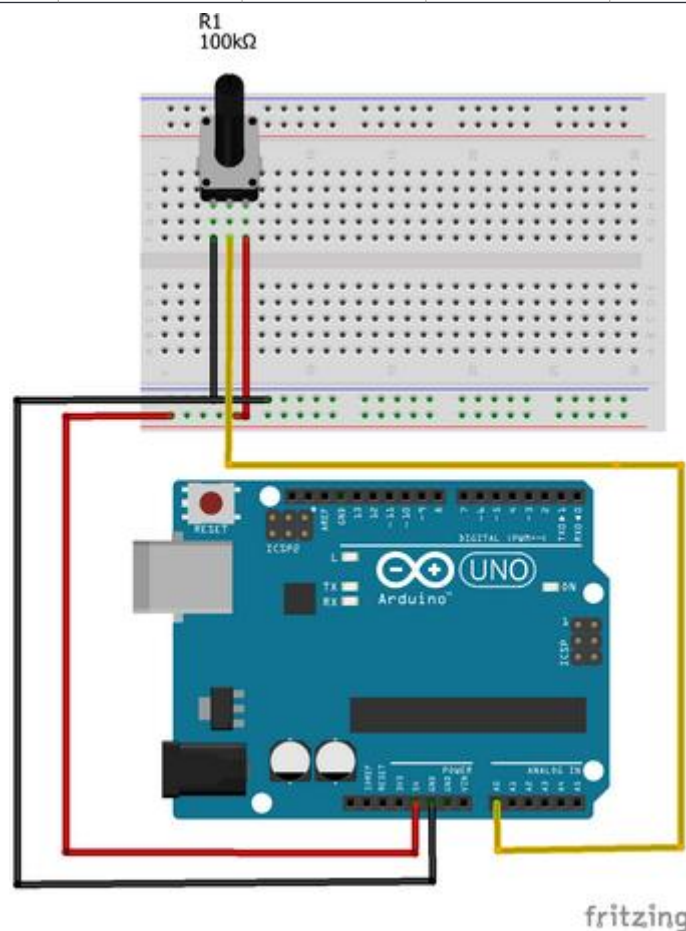
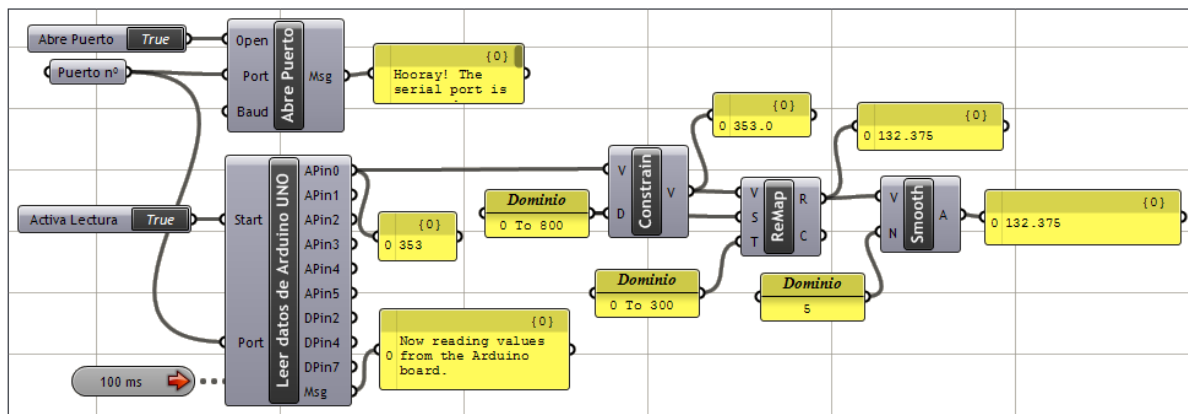


Montaje en protoboard

## 19. Leer entrada analógica y escalarla.

*Fichero 13\_1 leer entrada analógica y escalado*

En ocasiones es preciso realizar el escalado de una señal que se lee del puerto pasando sus valores otro rango. Para realizar esta operación se recurre al escalado de la señal. En este ejemplo hacemos precisamente el escalado de la señal que leemos desde el pin analógico de Arduino A0

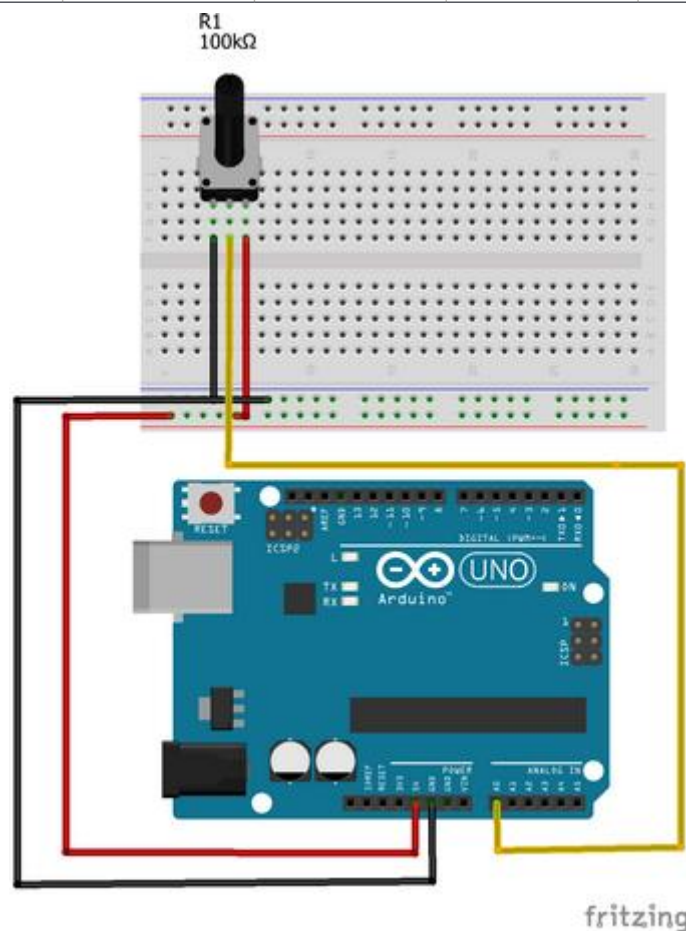
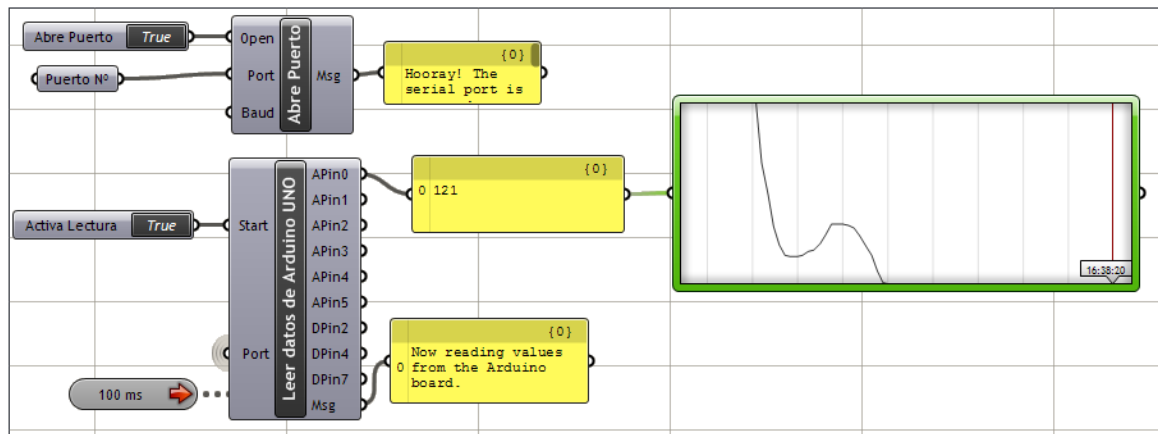


Montaje en protoboard.

## 20. Leer entrada analógica y representar.

*Fichero 13\_1 leer entrada analógica y representar*

En este último ejemplo se trata de utilizar la potencialidad de representar gráficamente una variable analógica haciendo uso del bloque de función **Value Tracker** de la librería **Params ->Util**



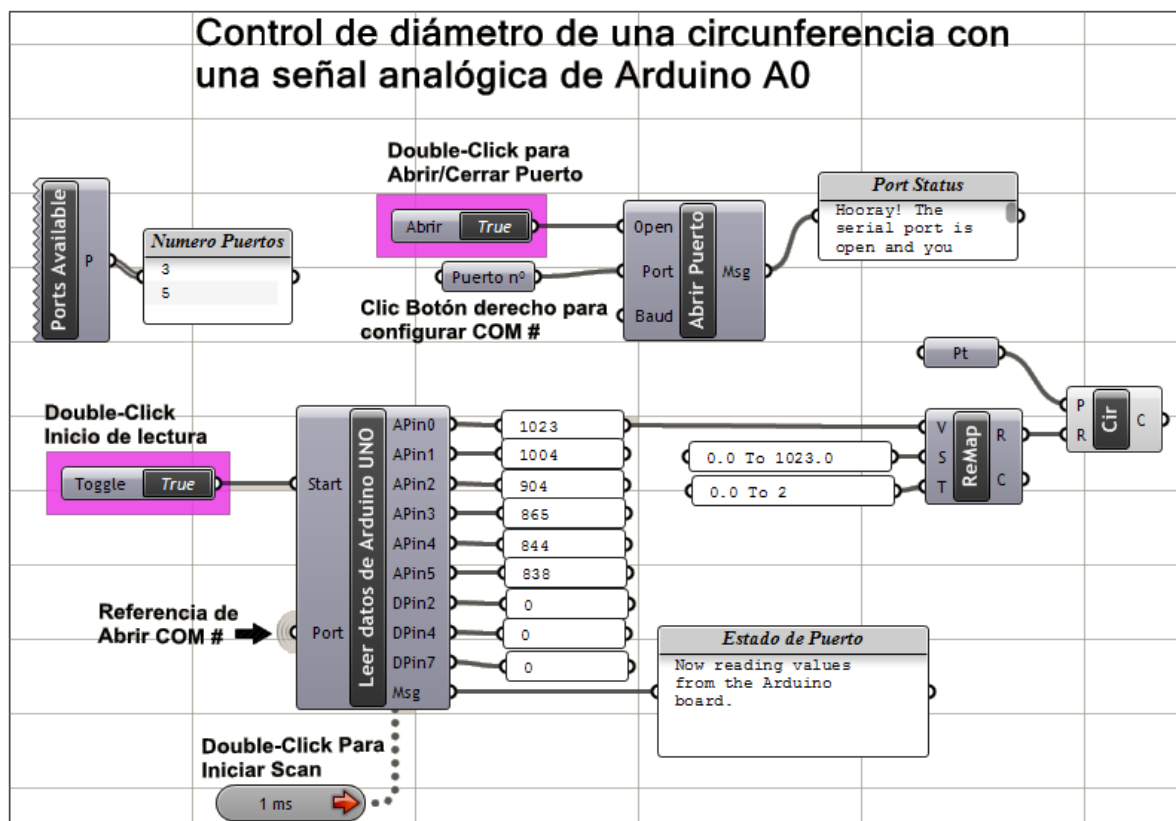
Montaje en protoboard

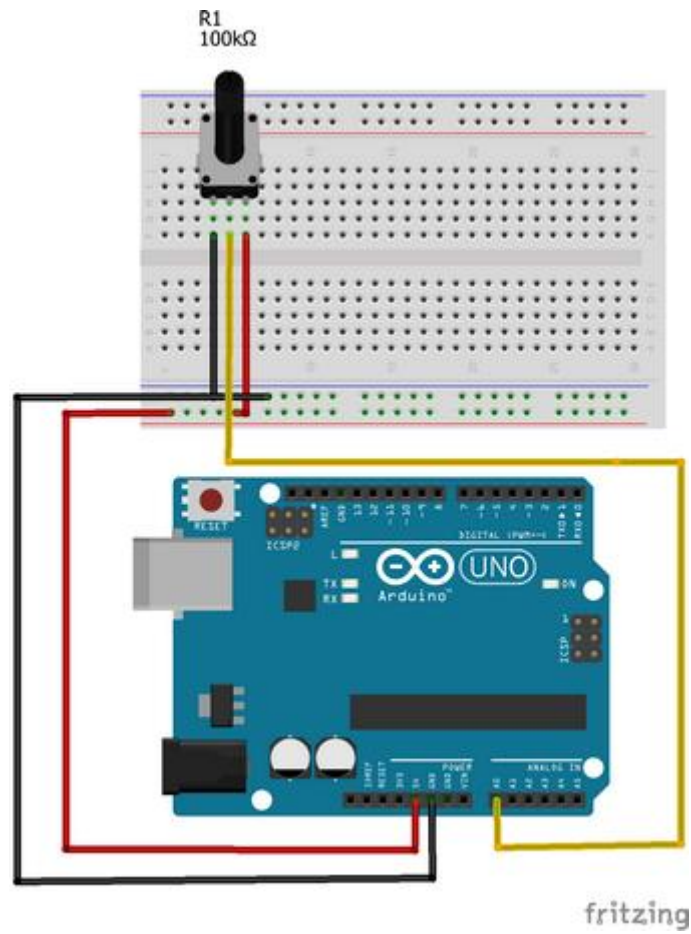
## 21. Control circunferencia.

*Fichero: 14 control circunferencia*

Con este ejemplo realizamos la interacción entre Arduino y Rhino controlando el radio de una circunferencia que se dibujara en la pantalla.

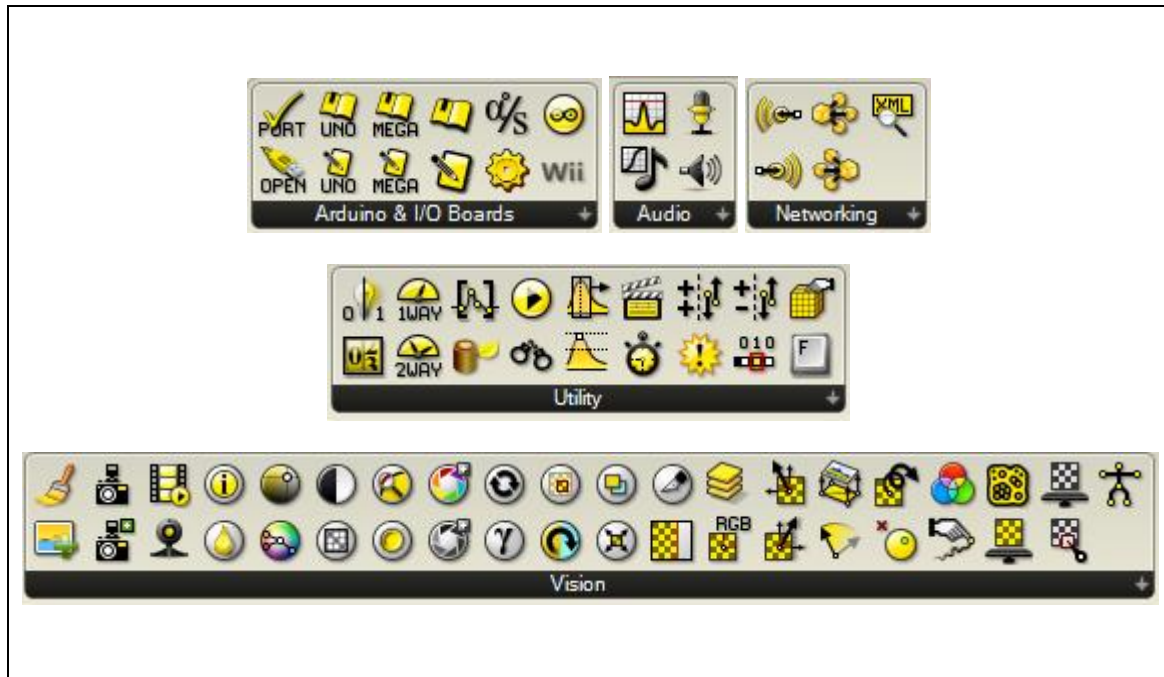
Tomamos el valor de la entrada analógica de la tarjeta Arduino A0 y lo llevamos al bloque **ReMap** que lo que hará será un escalado pasando el valor 0 a 1023 a 0 a 2 para acotar a este valor el valor del radio. Después se utilizara el bloque **Cir** que es el que dibuja la circunferencia. A este bloque le designamos un punto (centro de la circunferencia) simplemente marcándolo en la pantalla bloque **Pt**





Montaje en protoboard

## 22. Anexo I



### Descripción de librerías de Firefly

## Arduino Tools



### Puertos disponibles

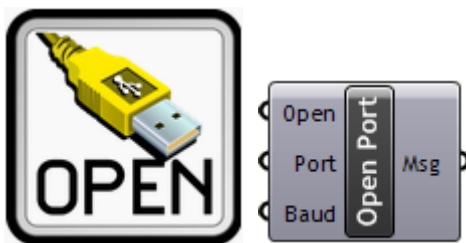
#### Descripción:

Comprueba los puertos COM que están disponibles. Si tenemos una o más tarjetas Arduino conectados al ordenador, éste devolverá el número asociado con cada puerto. Usted puede verificar el número de COM que es correcto, haga clic en **Herramientas > Puerto serie** dentro del IDE de Arduino (la tarjeta que debe estar conectado al ordenador).

**Entradas:** Ninguno

**Salidas:**

P (entero) - Lista de todos los puertos COM disponibles actualmente.



### Abrir /Cerrar Puerto

#### Descripción:

Abre o cierre la conexión de puerto serie. Se debe utilizar siempre que queramos

realizar un intercambio de datos entre Arduino y Grasshopper.

#### Entradas:

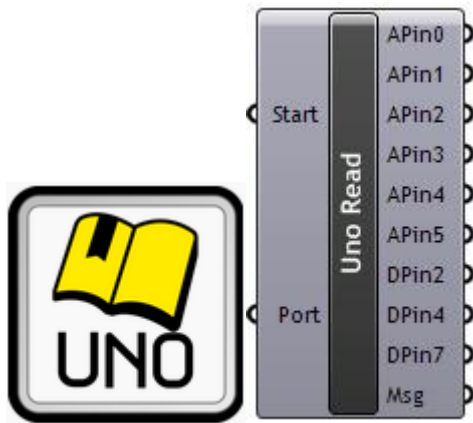
Open (boolean) - Abrir / Cerrar el puerto COM.

Port (entero) - El ID del puerto COM.

Baud (entero) - La velocidad de datos en bits por segundo para la transmisión de datos en serie. El valor por defecto si se utiliza el Firefly Firmata es 115200.

#### Salidas:

Msg. - Mensaje (string) - Estado actual del puerto serie.



### Leer datos de Arduino UNO

#### Descripción :

Utilizando el boceto Firefly Firmata , este componente leer los valores de todos los correspondientes pines digitales y analógicas de la Arduino Uno , Duemilanove , Diecimilla , o junta Lillypad u otros clones similares. Nota: Este componente está destinado a ser usado en conjunto con el Firefly Firmata Arduino Sketch .

#### Entradas:

Start (boolean ) - Empieza a leer los valores del puerto COM .

Port ( entero) - El ID del puerto COM para leer .

#### Salidas:

APin0 (entero ) - Valor del sensor leer del Pin de entrada analógica 0 (valor de 0 a 1.023 ) .

APin1 (entero ) - Valor del sensor leer del Pin de entrada analógica 1 (valor de 0 a 1.023 ) .

APin2 (entero ) - Valor del sensor leer del Pin de entrada analógica 2 (valor de 0 a 1.023 ) .

APin3 (entero ) - Valor del sensor leer del Pin de entrada analógica 3 (valor de 0 a 1.023 ) .

APin4 (entero ) - Valor del sensor leer del Pin de entrada analógica 4 (valor de 0 a 1.023 ) .

DPin2 (entero ) - Valor del sensor leer del pin de entrada digital 2 (valor de 0 - 1) .

DPin4 (entero ) - Valor del sensor leer del pin de entrada digital 4 (valor de 0 - 1) .

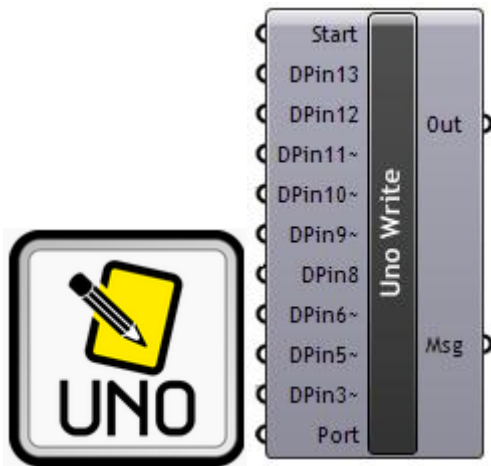


DPin7 (entero) - Valor del sensor leer del pin de entrada digital 7 (valor de 0 - 1) .

Msg. (cadena) - El estado actual del puerto serie.

### Notas:

Utilice el temporizador Grasshopper ( Params / Util) para provocar que el componente Uno Leer para actualizar la información del sensor a una frecuencia dada . Haga clic con el contador de tiempo del Grasshopper para establecer el intervalo del temporizador ( 1 milésima de segundo hará que el temporizador caduque tan pronto como sea posible).



## Escribir datos en Arduino Uno

### Descripción :

Utilizando el boceto Firefly Firmata , este componente escribir valores para todos los pines correspondientes de un Arduino UNO , Duemilanove , Diecimilla , o junta Lillypad u otros clones similares. Nota: Este componente está destinado a ser usado en conjunto con el Firefly Firmata Arduino Sketch .

### Entradas:

Inicie (boolean) - Empieza a leer los valores del puerto COM .

DPin13 (entero) - Valor de escribir en el pin (en función de estado del pin) .

DPin12 (integer) - Valor de escribir en el pin (en función de estado del pin) .

DPin11 (entero) - Valor de escribir en el pin (en función de estado del pin) .

DPin10 (entero) - Valor de escribir en el pin (en función de estado del pin) .

DPin9 (entero) - Valor de escribir en el pin (en función de estado del pin) .

DPin8 (entero) - Valor de escribir en el pin (en función de estado del pin) .

DPin6 (entero) - Valor de escribir en el pin (en función de estado del pin) .

DPin5 (entero) - Valor de escribir en el pin (en función de estado del pin) .

DPin3 (entero) - Valor de escribir en el pin (en función de estado del pin) .

Puerto (entero) - El ID del puerto COM a escribir.

**Salidas:**

Out ( cadena) - El mensaje de cadena enviada a la placa Arduino .

Mensaje ( cadena) - El estado actual del puerto serie.

**Notas:**

Haga clic en cada entrada para establecer el estado del pin . Cada estado pin incluye:

Digital: acepta un valor entre cero y uno .

PWM : Acepta un valor entre 0-255 y se utilizará para el ancho de pulso modulación (

PWM no está disponible en todos los terminales) .

Servo : acepta un valor entre 0-180 y se utiliza para ajustar la posición de un motor servo .

Para información adicional:

El control de un servo Pan / Tilt con Firefly Vídeo



## Leer Datos de Arduino Mega

**Descripción :**

Utilizando el boceto Firefly Firmata , este componente leer valores para los correspondientes pines digitales y analógicas en una placa Arduino Mega . Nota: Este componente está destinado a ser usado en conjunto con el Firefly Firmata Arduino Sketch .

**Entradas:**

Start (boolean ) - Empieza a leer los valores del puerto COM .

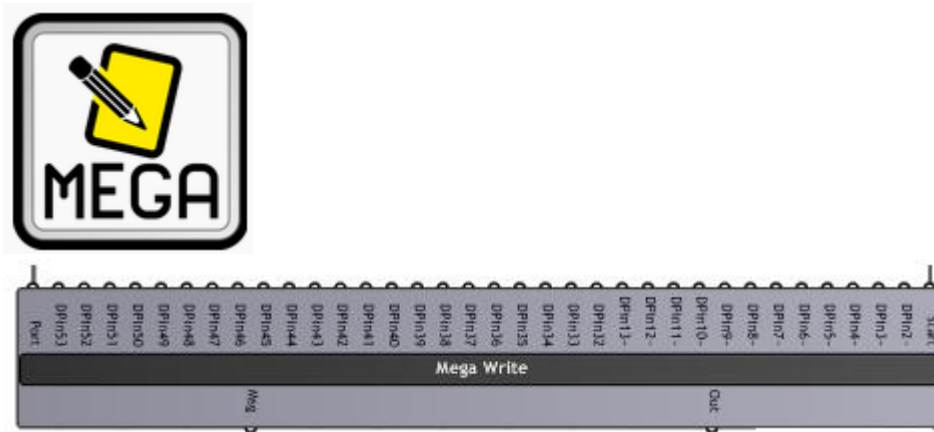
Port ( entero) - El ID del puerto COM para leer .

**Salidas:**

APin0 (entero) - Valor del sensor leer del Pin de entrada analógica 0 (valor de 0 a 1.023) .  
APin1 (entero) - Valor del sensor leer del Pin de entrada analógica 1 (valor de 0 a 1.023) .  
APin2 (entero) - Valor del sensor leer del Pin de entrada analógica 2 (valor de 0 a 1.023) .  
APin3 (entero) - Valor del sensor leer del Pin de entrada analógica 3 (valor de 0 a 1.023) .  
APin4 (entero) - Valor del sensor leer del Pin de entrada analógica 4 (valor de 0 a 1.023) .  
APin5 (entero) - Valor del sensor leer del Pin de entrada analógica 5 (valor de 0 a 1.023) .  
APin6 (entero) - Valor del sensor leer del Pin de entrada analógica 6 (valor de 0 a 1.023) .  
APin7 (entero) - Valor del sensor leer del Pin de entrada analógica 7 (valor de 0 a 1.023) .  
APin8 (entero) - Valor del sensor leer del Pin de entrada analógica 8 (valor de 0 a 1.023) .  
APin9 (entero) - Valor del sensor leer del Pin de entrada analógica 9 (valor de 0 a 1.023) .  
APin10 (entero) - Valor del sensor leer del Pin de entrada analógica 10 (valor de 0 a 1023)  
APin11 (entero) - Valor del sensor leer del Pin de entrada analógica 11 (valor de 0 a 1023)  
APin12 (entero) - Valor del sensor leer del pin análogo de entrada 12 (valor de 0 a 1023) .  
APin13 (entero) - Valor del sensor leer del Pin de entrada analógica 13 (valor de 0 a 1023)  
APin14 (entero) - Valor del sensor leer del Pin de entrada analógica 14 (valor de 0 a 1023)  
APin15 (entero) - Valor del sensor leer del Pin de entrada analógica 15 (valor de 0 a 1023)  
DPin22 (entero) - Valor del sensor leer del pin de entrada digital 22 (valor 0 - 1) .  
DPin23 (entero) - Valor del sensor leer del pin de entrada digital 23 (valor 0 - 1) .  
DPin24 (entero) - Valor del sensor leer del pin de entrada digital 24 (valor 0 - 1) .  
DPin25 (entero) - Valor del sensor leer del pin de entrada digital 25 (valor 0 - 1) .  
DPin26 (entero) - Valor del sensor leer del pin de entrada digital 26 (valor 0 - 1) .  
DPin27 (entero) - Valor del sensor leer del pin de entrada digital 27 (valor 0 - 1) .  
DPin28 (entero) - Valor del sensor leer del pin de entrada digital 28 (valor 0 - 1) .  
DPin29 (entero) - Valor del sensor leer del pin de entrada digital 29 (valor 0 - 1) .  
DPin30 (entero) - Valor del sensor leer del pin de entrada digital 30 (valor 0 - 1) .  
DPin31 (entero) - Valor del sensor leer del pin de entrada digital 31 (valor 0 - 1) .  
Msg (cadena) - El estado actual del puerto serie.

**Notas:**

Utilice el temporizador Grasshopper ( Params/Util) para provocar que el componente Uno Leer para actualizar la información del sensor a una frecuencia dada . Haga clic con el contador de tiempo del Grasshopper para establecer el intervalo del temporizador ( 1 milésima de segundo hará que el temporizador caduque tan pronto como sea posible).



## Escribir datos en Arduino Mega

### Descripción :

Utilizando el boceto Firefly Firmata , este componente escribir valores en todos los pines digitales correspondientes de un Mega placa Arduino . Nota: Este componente está destinado a ser usado en conjunto con el Firefly Firmata Arduino Sketch .

### Entradas:

Start (boolean ) - Empieza a leer los valores del puerto COM .

DPin2 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin3 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin4 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin5 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin6 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin7 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin8 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin9 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin10 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin11 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin12 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin13 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin32 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin33 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin34 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin35 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin36 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin37 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin38 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin39 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin40 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin41 (entero ) - Valor de escribir en el pin (en función de estado del pin ) .

DPin42 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin43 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin44 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin45 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin46 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin47 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin48 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin49 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin50 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin51 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin52 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
DPin53 (entero) - Valor de escribir en el pin (en función de estado del pin) .  
Puerto ( entero) - El ID del puerto COM a escribir.

**Salidas:**

Out ( cadena) - El mensaje de cadena enviada a la placa Arduino .

Msg ( cadena) - El estado actual del puerto serie.

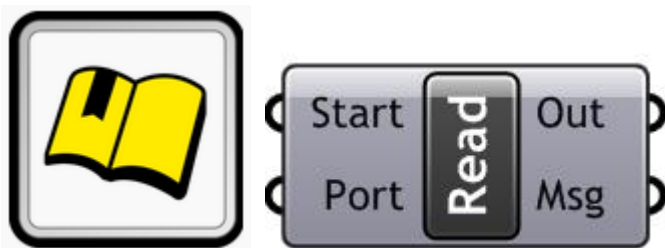
**Notas:**

Haga clic en cada entrada para establecer el estado del pin . Cada estado pin incluye:

Digital: acepta un valor entre cero y uno .

PWM : Acepta un valor entre 0-255 y se utilizará para el ancho de pulso modulación ( PWM no está disponible en todos los terminales).

Servo : acepta un valor entre 0-180 y se utiliza para ajustar la posición de un motor servo .



## Leer datos del Puerto Serie

**Descripción:**

Devuelve cualquier valor de cadena que viene sobre el puerto serie. Este componente no tiene ningún boceto Arduino específica asociada a ella, lo que significa que usted puede crear su propio código para enviar un valor de cadena de la Arduino con el comando () **Serial.println**.

**Entradas:**

Start (boolean) - Empieza a leer los valores del puerto COM.

Port (entero) - El ID del puerto COM para leer.

**Salidas:**

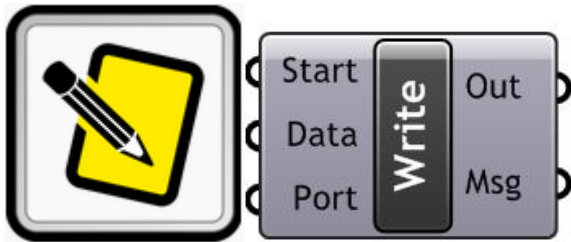
Out (cadena) - El valor de entrada de lectura de la conexión en serie.

Msg. (cadena) - El estado actual del puerto serie.

**Notas:**

Utilice el temporizador Grasshopper (Params/Util) para provocar que el componente Uno Leer para actualizar la información del sensor a una frecuencia dada. Haga clic con el contador de tiempo del Grasshopper para establecer el intervalo del temporizador (1 milésima de segundo hará que el temporizador caduque tan pronto como sea posible).

## Escribir datos en el Puerto Serie

**Descripción:**

Escribe cualquier valor de cadena al puerto serie. Usted tendrá que proporcionar también un código de Arduino para procesar el valor de la cadena que viene a través del puerto serie (normalmente manejada por el boceto Firefly Firmata).

**Entradas:**

Start (boolean) - Empieza a leer los valores del puerto COM.

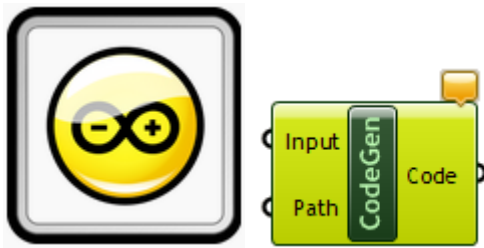
Data (cadena) - El valor de escribir en el puerto serie.

Port (entero) - El ID del puerto COM a escribir.

**Salidas:**

Out (cadena) - El mensaje de cadena enviada a la placa Arduino.

Msg. (cadena) - El estado actual del puerto serie.



## Generador de Código para Arduino

### Descripción :

Este componente intentará convertir cualquier definición de Grasshopper en código compatible Arduino (C + + ) sobre la marcha . Sólo tienes que conectar la cadena de salida desde cualquiera de éste con la salida de texto de los bloques Mega Uno a la entrada del generador de código y ver la acumulación de código a medida que crea su definición de Grasshopper . Su acción consiste en la detección de cualquier componente "conectado" en el componente de escritura Mega o Uno . Cuando se detecta un componente, se comprueba si tiene o no tiene una función para ese componente. Si lo hace, se añade la función de la parte superior del código , y llama correctamente a la función dentro de la configuración principal () o bucle (funciones) . Si la biblioteca de funciones no contiene una definición para un componente particular del Grasshopper , proporcionará una advertencia. Este código se puede guardar al mismo tiempo como un archivo pde . Para ser abierto en el IDE de Arduino .

### Entradas:

Input ( string) - Conecte el mensaje ' Out' de los componentes de escritura UNO / Mega a esta entrada para generar el código de Arduino .

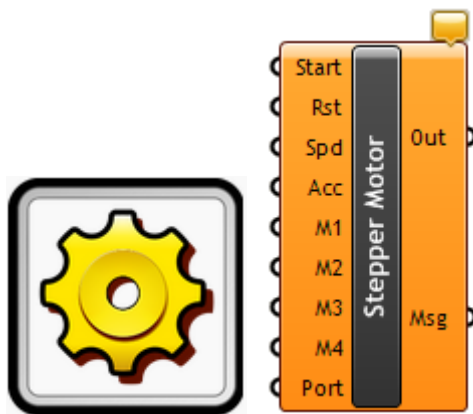
Path ( string) ( Opcional) - ruta del archivo opcional para guardar el código de Arduino ( extensión debe ser ino . ) .

### Salidas:

Code (cadena) - código compatible Arduino .

### Opciones:

Haga clic derecho en el componente generador de código para iniciar el Editor de código de Firefly.



## Control der Motores

### Descripción:

Este componente da formato a los datos de entrada para controlar hasta cuatro motores paso a paso . Nota: Este componente dará formato a los datos de acuerdo con el croquis Firefly Stepper Arduino incluido en la carpeta de código . Con este componente se puede controlar el número de pasos para mover los motores, la velocidad máxima de los motores, aceleración y calibración los valores establecidos.

### Entradas:

Start ( boolean ) - Indique si activar o no los motores.

Rst (boolean ) - Ajustar la posición actual del motor que es el punto de origen (utilizado para fines de calibración ) .

Spd (entero ) - La velocidad máxima de los motores.

Acc (entero ) - La cantidad de aceleración (y deceleración) a aplicar a los motores. Nota : La aceleración oscila desde 100,0 hasta 3200,0 donde el valor más bajo indica más rápido de aceleración / deceleración .

M1 ( largo ) - número de micropasos para mover el primer motor .

M2 ( largo ) - número de micropasos para mover el segundo motor .

M3 ( largo ) - número de micropasos para mover el tercer motor .

M4 (largo) - número de micropasos para mover el cuarto de motor.

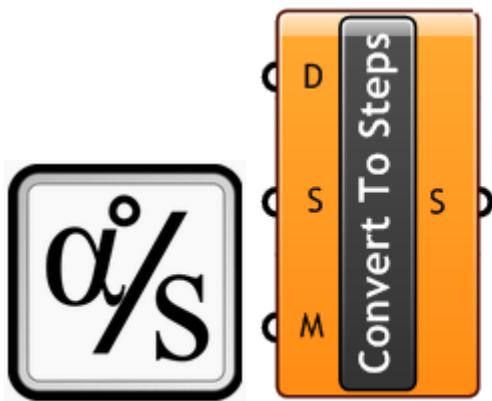
### Salidas:

S (cadena) - con formato de cadena para enviar a la placa Arduino .

### Notas:



El sketch Arduino utiliza para controlar estos motores paso a paso de tipo Stepper Biblioteca Accel construido por Adafruit Industries. Descargue la librería aquí: <https://github.com/adafruit/AccelStepper> Para instalar el Accel Stepper Arduino Biblioteca : Copiar / Pegar la carpeta AccelStepper en su carpeta ArduinoSketchDirectory / libraries . Esta es probable encontrarla en: C: \ Users\nombre de usuario\Documents\Arduino\libraries. Puede que tenga que hacer las bibliotecas subcarpeta primera vez. Luego se vuelve a reiniciar el IDE de Arduino . Para obtener más instrucciones , visite : <http://www.ladyada.net/library/arduino/libraries.html>



## Conversión de Grados en Pasos

### Descripción:

Este componente permite convertir un valor de ángulo en grados a la cantidad apropiada de pasos para mover un motor.

### Entradas:

D (doble) - Grados a convertir en micropasos.

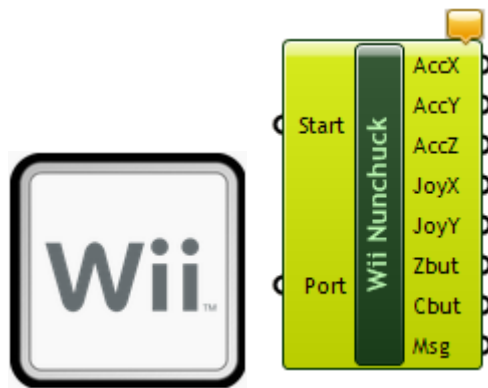
S (entero) - Pasos por revolución. Cada motor paso a paso es evaluado por el número de pasos que toma para hacer una vuelta completa. Un motor de 200 pasos, por ejemplo, tomaría 200 pasos para girar una vuelta completa, o aproximadamente 1.8deg por paso.

M (entero) - Muchos controladores paso a paso utilizan microstepping, que descompone el paso mínimo en pasos más pequeños micro. La mayoría de los controladores predeterminados para 8 micropasos. Así, para mover un motor paso a paso 200 una revolución completa que se necesita para mover 1.600 micropasos. Esta entrada le permite establecer cuántos micropasos utilizará su controlador.

Nota: Esta entrada sólo permitirá micropasos de 1, 2, 4, o 8.

**Salidas:**

S (largo) - El número de pasos para mover el motor.



## Control con el mando Wii Nunchuck

**Descripción:**

Este componente permitirá leer todos los valores de los sensores (acelerómetro de 3 ejes, palancas, C y Z botones) desde el mando Nunchuck Wii. Nota: Este componente está destinado a ser usado en conjunto con el Wii Nunchuck Arduino Sketch.

**Entradas:**

Start (boolean) - Empieza a leer los valores del puerto COM.

Port (entero) - El ID del puerto COM para leer.

**Salidas:**

ACCX (entero) - Acelerómetro valor X (0-1023).

ACCY (entero) - Acelerómetro valor Y (0-1023).

AccZ (entero) - Acelerómetro valor Z (0-1023).

JoyX (entero) - valor del eje X Joystick (0-1023).

Joyy (entero) - Joystick valor del eje Y (0-1023).

Cbut (entero) - Valor botón C (0 - 1).

Zbut (entero) - Valor botón Z (0 - 1).

Msg (cadena) - El estado actual del puerto serie.

## Utilidades



### AND Flip Flop

**Descripción:****Entradas:**

S (boolean) - Set.

R (boolean) - Reset.

**Salidas:**

O (boolean) - Salida.

I (boolean) - Salida de Inverse.



### Disparo

**Descripción:**

El componente de Bang registra las condiciones de borde para una entrada booleana. El T-salida se registrará un valor verdadero momentánea cuando la entrada cambia a un estado verdadero. La salida F registrará un valor verdadero

momentánea cuando la entrada cambia de nuevo a un estado falso. Este componente es el equivalente a un componente Bang en otros programas como Pd, Max / MSP, y VVVV. Nota: Este componente requiere el uso de un componente Timer (Parámetros / util).

**Entradas:**

B (boolean) - disparador de Boole.

**Salidas:**

T (boolean) - Devuelve una salida momentánea VERDADERO cuando el estado de entrada cambia a TRUE.

F (boolean) - Devuelve una salida momentánea VERDADERO cuando el estado de entrada cambia a FALSE.



## Binary Blink

**Descripción:**

Oscila de 0 y 1 de la base de un modelo de entrada de enteros. Se puede utilizar para crear intrincadas (ALTA / BAJA) patrones de parpadeo que se enviarán a los componentes de escritura en serie para controlar los LED, etc

**Entradas:**

P (entero) - Una lista de los números enteros que representan el patrón de oscilación en milisegundos. Por ejemplo, un patrón de parpadeo con dos valores (500, 1000) devolvería un cero (o baja) para la mitad de un segundo, y un uno (o alta) durante un segundo.

S (boolean) - Start / Stop de alternancia.

R (boolean) - El patrón se repite.

**Salidas:**

B (entero) - patrón binario de ceros o unos.



## Buffer

### Descripción:

Almacenar un conjunto de valores en la memoria intermedia basados en el dominio tampón. Por ejemplo, un dominio de amortiguamiento del 0 al 10 almacenará los últimos 10 valores. Sin embargo, si usamos un dominio de amortiguamiento del 5 al 15, entonces el búfer almacenará los valores de 10, pero sólo aquellos que se produjo cinco cuadros antes de que el valor de entrada actual.

### Entradas:

V (doble) - valor de entrada para almacenar en la memoria intermedia.

D (dominio) - Rango de valores para almacenar en la memoria intermedia.

### Salidas:

V (doble) - Lista de buffer.

Mn (doble) - El valor mínimo almacenado en el búfer.

Mx (doble) - El valor máximo almacenado en el búfer.



## Constreñir

### Descripción:

Restringe un número a un rango numérico específico (Domain).

### Entradas:

V (doble) - valor de entrada.

D (dominio) - Rango de restringir el valor de entrada.

**Salidas:**

V (doble) - valor restringida.



## Contador

**Descripción:**

Condes hacia arriba y hacia abajo. Puede establecer la dirección para contar, tamaño del paso, y límites de las restricciones para el contador. Nota: Este componente requiere el uso del componente GH\_Timer (Parámetros / util).

**Entradas:**

D (boolean) - Sentido de contaje. Un valor TRUE contará hacia arriba (positivo), mientras que un valor FALSO contará hacia abajo (negativo).

N (doble) - Tamaño del paso para cada número sucesivo.

S (boolean) - Start / Stop mostrador.

R (boolean) - Restablecer el contador a cero.

I (dominio) - dominio opcional (rango de valores) para limitar el contador.

**Salidas:**

C (doble) - El valor del contador.



## Registro de datos

### Descripción :

El componente de datos de registro almacenará un conjunto de valores en una lista. Si la longitud de lista se establece en " 0 ", entonces el registro almacenará valores indefinidamente ( o hasta que se detenga la grabación o restablecer el registro ) . De lo contrario, la longitud de lista determinará el número de valores para almacenar en el registro . Si la entrada Wrap se establece en True , los valores comenzarán regrabación en los valores de registro anteriores después de que haya llegado al final de la longitud de lista .

### Entradas:

V ( doble ) - Valor numérico para registrar en el registro de datos.

R ( boolean ) - Restablecer el registro de datos.

S ( boolean ) - Comience a grabar los valores en el registro de datos.

W ( boolean ) - Índice Wrap para listar los límites. Si el valor del abrigo se establece en TRUE y el número de valores en el registro de datos excede el límite de datos, el registro de datos empezará valores grabando de nuevo sobre las entradas existentes que comienzan en el número de índice cero.

L ( entero ) - Limite el número de valores registrados . Si el valor límite se establece en cero, el registro de datos registrará valores indefinidamente (o hasta que deje de enviar valor) .

P ( ruta ) - ruta del archivo opcional para transmitir el contenido del registro de datos.

### Salidas:

O ( doble ) - La lista de los valores registrados en el registro de datos.



## One Way Fader

### Descripción:

Fundido de un valor a otro basado en un único intervalo de tiempo (ms). Nota: Este componente requiere el uso del componente GH\_Timer (Parámetros / util).

### Entradas:

- V1 (doble) - El primer extremo numérico del fundido.
- V2 (doble) - El segundo extremo numérico del fundido.
- T (entero) - El tiempo en milisegundos a desaparecer de un extremo (V1) a la otra (V2).
- D (entero) - Retardo en milisegundos antes de reiniciar el fundido.

### Salidas:

- O (doble) - La posición actual del fundido.
- C (entero) - contador de bucle.



## Two Way Fader

### Descripción:

Fundido entre un valor mínimo y máximo basado en el fade in y fade out intervalo de tiempo (ms). Nota: Este componente requiere el uso del componente GH\_Timer (Parámetros / util).

### Entradas:

- V1 (doble) - El primer extremo numérico del fundido.
- V2 (doble) - El segundo extremo numérico del fundido.



T1 (entero) - El tiempo en milisegundos a desaparecer de un extremo (V1) a la otra (V2).

T2 (entero) - El tiempo en milisegundos a desaparecer de un extremo (V2) a la otra (V1).

D1 (entero) - Retardo en milisegundos antes de comenzar la segunda mitad del fundido.

D2 (entero) - Retardo en milisegundos antes de reiniciar el fundido.

**Salidas:**

O (doble) - La posición actual del fundido.

C (entero) - contador de bucle.



## Velocidad de cuadros

**Descripción:**

Devuelve el tiempo en milisegundos desde el último dato ha sido actualizado, así como un sistema de bastidores estimados por segundo intervalo.

**Entradas:**

Ninguno

**Salidas:**

T (doble) - se inició Número de milisegundos desde el último ciclo de la solución.

Hz (entero) - El número aproximado de las soluciones que se calculan por segundo.



## Se pulsa la tecla

### Descripción:

Este componente se registrará si una clave en el teclado sólo se ha pulsado. Sólo tiene que especificar un valor de cadena que le gustaría probar, y conectar un componente Timer a este componente. La salida devolverá True si la clave estaba presionado. Este componente reconoce todas las entradas alfanuméricas, teclas F y las teclas especiales (como arriba, abajo, izquierda, derecha, control de cambios, el espacio, etc.) Nota: Este componente requiere el uso del componente GH\_Timer (Parámetros / util).

### Entradas:

K (tecla) - La clave para la prueba.

### Salidas:

R (boolean) - Resultado.



## Está seleccionado Geometría

### Descripción:

El componente seleccionado es Geometría acepta una lista de números de GUID para la geometría especificada y devuelve un valor booleano verdadero si se ha seleccionado en la actualidad la geometría. Nota: Este componente requiere el uso del componente GH\_Timer (Parámetros / util).

**Entradas:**

ID (GUID) - GUID de objeto que desea probar.

**Salidas:**

R (boolean) - Resultado.

**NI Flip Flop****Descripción:**

El NI flip-flop tiene dos entradas, es decir, un conjunto de entrada (S) y un Reset (R) de entrada. Una simple representación de un flip-flop SR es un par de cross-acoplados puertas NOR, es decir, la salida de una compuerta está vinculado a una de las dos entradas de la otra puerta, y viceversa. Por lo tanto, el establecimiento de la puerta de salida en falso requiere  $R = \text{true}$  y  $S = \text{false}$ , mientras que la creación de la puerta de salida a la verdadera requiere  $S = \text{true}$  y  $R = \text{false}$ .

**Entradas:**

S (boolean) - Set.

R (boolean) - Reset.

**Salidas:**

O (boolean) - Salida.

I (boolean) - Salida de Inverse.



## Reproducción

### Descripción:

El componente de reproducción recuperar valores de un archivo de texto (tipos de archivos aceptables: Txt, csv, y dat.). Y comenzará la devolución de valores individuales a una velocidad determinada (s). Puede introducir varias tasas de refresco para mantener al mismo tiempo un seguimiento de varias salidas a diferentes intervalos.

### Entradas:

P (ruta) - Ruta de archivos para controlar. Aceptables los archivos incluye: Txt, csv, y dat..

S (boolean) - Iniciar el proceso de reproducción.

F (entero) - Lista de números enteros que representan el tiempo aproximado en mseg.

Rupas (boolean) - patrón de repetición de la reproducción.

Rs (boolean) - patrón de reproducción Reset.

### Salidas:

O (string) - Salida de Reproducción.



## Suavizante (media móvil)

### Descripción:

Smooth (o promedio) de un valor de entrada sobre la base de un nivel de muestreo (número de valores).

### Entradas:

V (doble) - valor de entrada para suavizar.

S (entero) - Número de valores para la media (nivel de muestreo).

**Salidas:**

A (doble) - Valor promedio.

**Suavizante (Temporal)****Descripción:**

Este componente converge en un valor numérico alisado basado en una distribución ponderada de las observaciones del pasado y el valor de entrada actual.

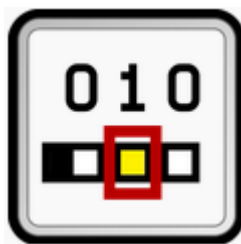
**Entradas:**

V (doble) - valor de entrada para suavizar.

F (doble) - Factor Suavizante dónde;  $0,0 < A < 1,0$ . Nota: Cuanto más alto sea el valor, más tiempo tomará para que el promedio para converger.

**Salidas:**

S (doble) - valor suavizado.

**Detección de Estado****Descripción:**

El componente de detección de estado se utiliza cuando se quiere hacer algo de acción basado en cuántas veces se presiona un botón. Se comprueba el número de veces que el valor actual ha cambiado de ALTO a BAJO (o viceversa) y si o no el módulo de que el valor del contador es igual a cero. Si es así, se voltea la salida de detección de estado.

Por lo tanto, si el valor del módulo es 2, entonces todas las otras veces que se pulsa el botón, el valor de estado cambiará. O si el valor del módulo es de 4, entonces cada cuarto pulsar el botón se activará el valor de estado para cambiar.

**Entradas:**

V (entero) - Valor de entrada para probar.

M (número entero) - valor de módulo para determinar el número de cambios de estado antes de que se hizo un interruptor.

**Salidas:**

S (entero) - Estado actual.

C (entero) - Contador.



## Cronómetro

**Descripción:**

El componente Cronómetro intenta replicar la funcionalidad de un verdadero cronómetro. Cuando ajuste el conmutador de entrada de inicio a la verdadera, el temporizador empieza a contar, y cuando lo enciende de nuevo a falso, entonces se devolverá el tiempo en milisegundos para ese intervalo. Puede repetir este patrón, la adición de más tiempo cada vez que el cronómetro se pone en marcha y se detuvo. Puede restablecer el intervalo Cronómetro activando la entrada de reset.

**Entradas:**

S (boolean) - Comenzar / Finalizar el cronómetro.

R (boolean) - Reajuste el cronómetro.

**Salidas:**

T (entero) - Tiempo en milisegundos desde el cronómetro inició por última vez.