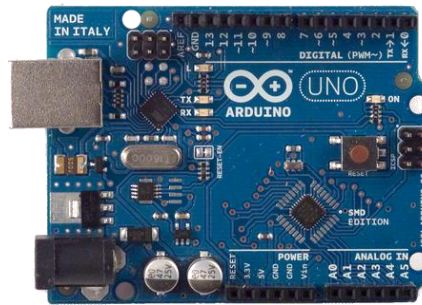


Labview + Arduino

Utilización de Labview para la Visualización y Control de la
Plataforma Open Hardware Arduino



+



Arduino

Ver. 1.0

José Manuel Ruiz Gutiérrez

Serie: Herramientas Gráficas para la programación de
Arduino



INDICE

1. Introducción
2. Cinco razones para utilizar Arduino + LabVIEW
3. ¿Dónde encontrar ayuda y herramientas para usar NI LabVIEW y Arduino?
4. Instalación del Software y el Hardware.
5. Instalación del Firmware de comunicación entre LabVIEW Interface y Arduino Uno?
6. Algunas preguntas sobre la interface LIFA
7. Desarrollo de aplicaciones básicas.
 - 7.1. Lectura de una entrada digital.
 - 7.2. Test1
 - 7.3. Comparador
 - 7.4. Contador de impulsos
 - 7.5. Contador de impulsos con puesta a cero
 - 7.6. Intermitente
 - 7.7. Semáforo Simple
 - 7.8. Semáforo Ajustable
 - 7.9. Gobierno de una Salida Analógica PWM
 - 7.10. Gobierno de una salida digital seleccionada
 - 7.11. Escritura/Lectura de todos los canales
 - 7.12. Control de Servos
 - 7.13. Función AND
 - 7.14. Temperatura 1
 - 7.15. Control Motor de cc. Velocidad y Sentido
 - 7.16. Medida de Temperatura mediante el Bus I2C
 - 7.17. Diálogo con Arduino Ethernet.
 - 7.18. Lectura de una Entrada Analógica.
 - 7.19. Conexión de un Módulo BlinkM.
 - 7.20. Lectura de un valor Analógico Continuamente tomando muestras.
 - 7.21. Adquisición de un número determinado de muestras de un canal analógico.
 - 7.22. Medida de Luz
 - 7.23. Control de un Diodo LED Tricolor (RGB).
 - 7.24. Medida de Temperatura.
 - 7.25. Manipulación de un Mando Joystick.
 - 7.26. Generador de Tonos

ANEXOS

- Material Básico
- Librerías de LIFA

Una primera reflexión.

Permíteme amable lector dedicar un pequeño espacio en este trabajo que ahora tienes en tus manos y que forma parte de un conjunto de trabajos que he venido realizando en los últimos años, para realizar una reflexión que estimo muy importante.

Al escribir este, y otros documentos que escribí sobre el mismo tema, mi objetivo principal ha sido poder contribuir al desarrollo del conocimiento en el ámbito de las **Plataformas Open Hardware** y sus aplicaciones en el Desarrollo de Prototipos y Aprendizaje de las Técnicas de Automatización y Control Programable. Mi idea principal, es “compartir” mi trabajo y mis conocimientos con los demás, porque estoy convencido que en esta “sociedad del conocimiento”, que nos ha tocado vivir, ese es un camino grato y amable. No es “puro romanticismo” esto que te digo, es, más bien, una realidad, una forma de pensar, de trabajar y de vivir. Solo en un contexto de generosidad y colaboración es posible que avance la tecnología, siempre con el objetivo de lograr el bienestar de los pueblos y la justicia en sus formas de gobierno.

Ojala y otros muchos profesores, investigadores, alumnos y entusiastas de la electrónica y la informática, sigan este camino y cada vez sean más los trabajos que se pongan gratuitamente al servicio de la comunidad. Son muchos los que lo han hecho hasta ahora y espero que sean más los que se sumen a esta idea de “compartir el conocimiento”.

José Manuel Ruiz Gutiérrez

j.m.r.gutierrez@gmail.com

24 de Agosto 2012

Agradecimientos y reconocimientos.

La elaboración de este trabajo no hubiese sido posible sin recurrir a las fuentes de conocimiento y a la experiencia de numerosas personas que de manera libre o adscritos a una entidad pública o privada han trabajado en el desarrollo de aplicaciones tanto hardware como Software para la Plataforma Open Hardware Arduino.

Debo reconocer y agradecer en primer lugar la valiosa labor de [NI \(National Instruments\)](#) al atender esta demanda de la comunidad académica en el desarrollo de [LIFA](#) (LabVIEW Interface for Arduino). De ellos he utilizado gran parte de la información que me ha permitido desarrollar este manual.

Quiero también agradecer a todos los innumerables investigadores que desde sus distintos ámbitos han colocado en la Web sus trabajos para poder ser utilizados por la comunidad. Algunos ejemplos que figuran en este manual proceden de algunas páginas que se referencian en ellos.

Recordar especialmente a toda la comunidad de “seguidores de Arduino” con los que he compartido gratas sesiones de intercambio real, en vivo y directo, sus conocimientos, en los Congresos, Barcamps y Jornadas que se viene realizando en España y en el resto de países del mundo.



Este trabajo está bajo licencia [Creative Commons Attribution 3.0 Unported License](#)
Agosto de 2012 Versión de Documento: Versión. (Beta en Revisión)
Blog de referencia: <http://josemanuelruizgutierrez.blogspot.com>



1. INTRODUCCIÓN

La interfaz de LabVIEW para Arduino (LIFA) Toolkit es una herramienta gratuita que se puede descargar desde el servidor de NI (National Instruments) y que permite a los usuarios de Arduino adquirir datos del microcontrolador Arduino y procesarlos en el entorno de programación gráfica de LabVIEW. Para utilizar la aplicación LIFA no es necesario disponer de una versión comercial de LabVIEW basta con adquirir el software LabVIEW Student Edition que se distribuye por destinos medios a un precio muy asequible (Sparkfun Electronics lo distribuye junto con la tarjeta Arduino UNO a un precio de unos 50 \$).

Este trabajo pretende ser una respuesta a la necesidad que existe en el ámbito académico universitario de poder disponer de un entorno de Diseño y Prototipado de Aplicaciones de Medida, Control y Automatización de Procesos y Espacios Físicos. La plataforma Open Hardware Arduino ha demostrado en su corta, pero intensa vida, ser una opción muy interesante para incluir en un Laboratorios de Prototipado. Son muchas sus ventajas, entre las que destacamos su costo, su libre difusión y exención de costos de patentes por desarrollo así como la gran comunidad de usuarios que se ha creado y que esta generando una cantidad de aplicaciones increíble.

La participación de una empresa como NI National Instruments en este proyecto de desarrollo de aplicaciones Software para Arduino pone de manifiesto el gran interés de la herramienta. No solo se ha sumado esta empresa al proyecto sino también Google con sus aplicaciones escritas en lenguaje Android, Telefónica con el desarrollo de una tarjeta para programar desde telefonía móvil, y otras compañías.

En este manual he recogido información dispersa y sobre todo he incluido numerosos ejemplos que pretenden ser una ayuda y un estímulo para que quien lo desee pueda experimentar con esta herramienta Software que por otra parte funciona soportada con una simple Licencia de estudiante de LabVIEW.

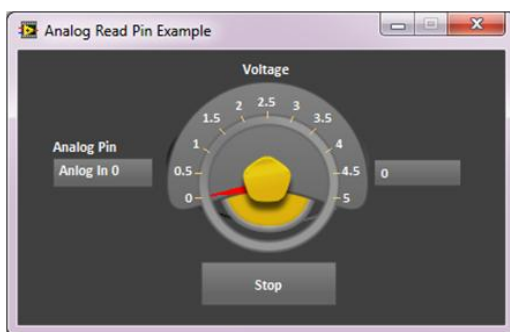
5

2. Cinco razones para utilizar Arduino+Labview

(traducido de la página <http://www.ni.com/white-paper/12879/en>)

El microcontrolador Arduino es una plataforma de bajo costo de electrónica de prototipos. Con la interfaz de LabVIEW para Arduino LIFA se puede aprovechar la potencia del entorno de programación gráfica de LabVIEW para interactuar con Arduino en una nueva dimensión.

1. Interface Gráfica de Usuario (Graphical User Interface GUI)



Visualizar los datos

Mostrar datos de los sensores en el monitor del ordenador mediante los paneles frontales de LabVIEW.

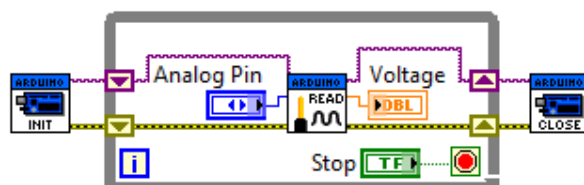
Personalización de la interfaz de usuario

Permite dar al proyecto un toque profesional con los controles del panel frontal de LabVIEW y los indicadores.

2. Programación Grafica

Arrastrar y soltar

En lugar de tratar de recordar un nombre de función, se encuentra en la paleta y colóquelo en su diagrama de bloques.



Documentación simple

Pase el ratón sobre cualquier VI o función con el ratón y ver al instante la documentación con ayuda contextual.



3. ¿Dónde encontrar ayuda y herramientas para usar NI LabVIEW y Arduino?

En el proceso de instalación del software de Arduino para Labview se recomienda recurrir a los siguientes enlaces en donde se podrá descargar el software y leer las recomendaciones en el proceso de instalación esta paginas están el servidor de National Instrument (LabVIEW)

- El paquete LabVIEW Interface for Arduino es totalmente gratuito y se puede descargar en:
<https://decibel.ni.com/content/groups/labview-interface-for-arduino>
- Las instrucciones para la instalación se encuentran aquí:
<https://decibel.ni.com/content/docs/DOC-16204>
- Si se desea adquirir el Kit NI LabVIEW + Arduino se puede adquirir aquí:
<http://www.sparkfun.com/products/10812>
- En el siguiente enlace se pueden encontrar las respuestas a las dudas para utilizar el toolkit de NI LabVIEW para Arduino:
<https://decibel.ni.com/content/docs/DOC-16024>
- La guía de usuario para NI LabVIEW y Arduino se puede encontrar aquí:
<http://nitalk.natinst.com/docs/DOC-39019>
- Para utilizar NI LabVIEW + Mac se puede encontrar ayuda aquí:
<https://decibel.ni.com/content/thread/9782>
- A continuación se muestran los enlaces en donde poder encontrar ejemplos realizados con la el Kit. **NI LabVIEW + Arduino:**
 - Show de luces:
<https://decibel.ni.com/content/docs/DOC-16070>
 - Usando librerías de Arduino con NI LabVIEW:
<https://decibel.ni.com/content/thread/12931>
 - Ejemplo del control de un motor de pasos con Arduino

<https://decibel.ni.com/content/docs/DOC-20084>

- Leer la intensidad de luz en una celda solar con Arduino
<https://decibel.ni.com/content/docs/DOC-16069>
- Manipular los LEDs de la tarjeta arduino
<https://decibel.ni.com/content/docs/DOC-16261>



4. Instalación del Software y el Hardware

A continuación describimos los pasos que se recomiendan para la puesta en marcha de la herramienta **LIFA** (LabVIEW para Arduino):

La configuración de la Interfaz de LabVIEW para Arduino es un proceso de seis pasos que usted sólo tendrá que completar una sola vez. Por favor, siga las siguientes instrucciones para comenzar a crear aplicaciones con la interfaz de LabVIEW para Arduino.

(Para una breve descripción de la interfaz de LabVIEW para Arduino ver post Michaels [aquí](#)).

1. Instalar LabVIEW

Si ha adquirido el paquete de LabVIEW y del Sparkfun.com Arduino puede instalar LabVIEW desde el DVD incluido.

Si usted no posee una copia de LabVIEW, usted puede descargar e instalar la versión de evaluación de 30 días aquí.

2. Instale los controladores VISA NI-

- [Windows Download](#).
- [Linux Download](#).
- [Mac Download](#).

3. Instale JKI VI Package Manager (VIPM) Community Edition (gratis).

Todos los sistemas operativos. [All Operating Systems](#).

4. Instalación de la Interfaz de LabVIEW para Arduino como se describe en KB 5L38JQYG [KB 5L38JQYG](#)

5. Conectar la placa Arduino a su PC como se describe en KB 5INA7UYG [KB 5INA7UYG](#)

6. Carga de la interfaz de LabVIEW para firmware Arduino en su Arduino como se describe en [KB 5LPAQIYG](#)

7. El firmware se puede encontrar en <LabVIEW> \ vi.lib Interface \ LabVIEW para Arduino \ Firmware \ LVIFA_Base. Utilizar el IDE de Arduino para implementar este firmware de la placa Arduino.)

Ahora está listo para usar la interfaz de LabVIEW para Arduino.



5. Instalación del Firmware de comunicación entre LabVIEW Interface y Arduino Uno?

Para poder comunicar Labview con Arduino, previamente, debemos instalar en la tarjeta el firmware correspondiente.

Partimos del supuesto de que ya tenemos instalado en nuestro PC el entorno IDE Arduino.

El fichero que debemos cargar en el IDE de Arduino para luego descargar en la tarjeta se encuentra en la carpeta en donde tengamos instalado Labview

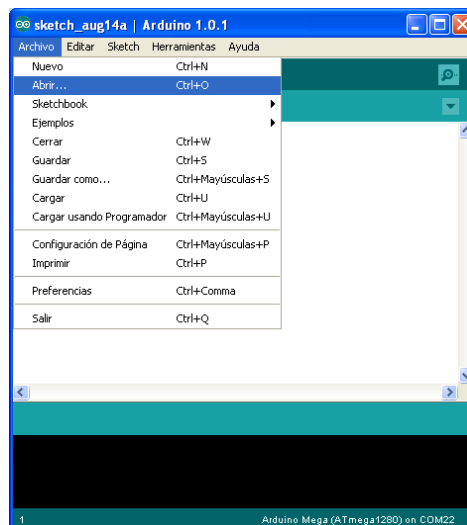
...\\National Instruments\\LabVIEW 20XX\\vi.lib\\LabVIEW Interface for Arduino\\Firmware\\LVIFA_Base

Ejecutamos el IDE Arduino y cargamos el fichero.

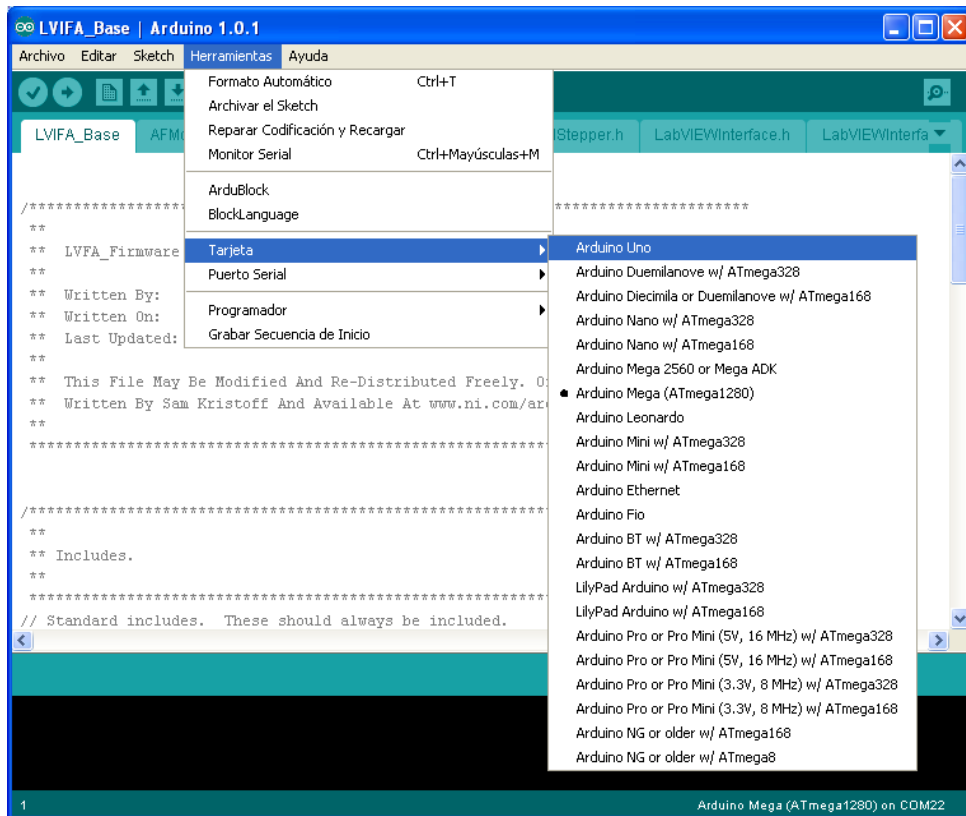
Pasos a seguir:

- Abrir el IDE Arduino . Pulsando sobre arduino.exe

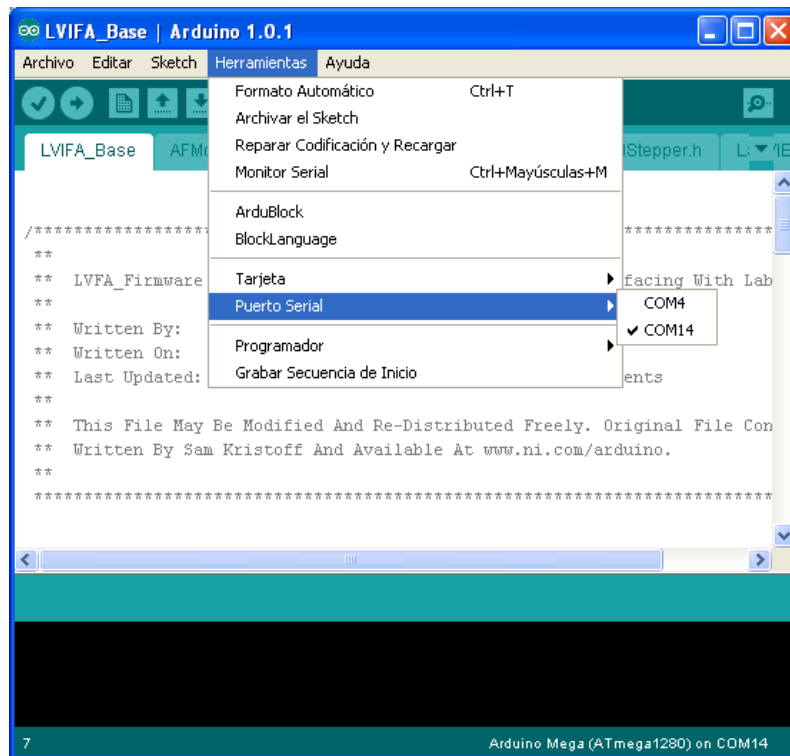
Con la opción **Fichero->Abrir** Buscamos el fichero LVIFA_Base.pde



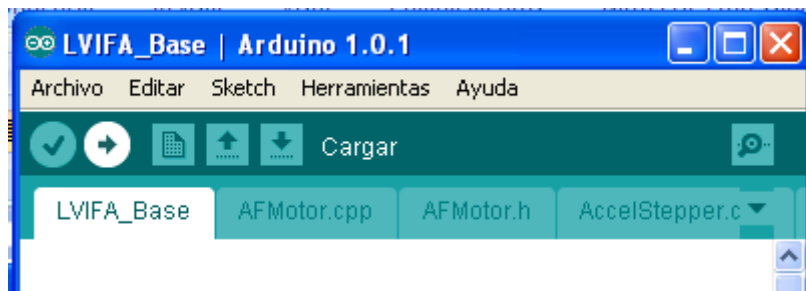
- Seguidamente una vez cargado el fichero en el IDE Arduino seleccionamos la tarjeta con la que trabajaremos.



- Seguidamente seleccionamos el puerto con el realizaremos la descarga del firmware sobre la tarjeta Arduino.



- Una vez realizadas estas operaciones basta con que pulsemos el botón de carga de sketch del IDE para que el fichero se transfiera a la tarjeta y, una vez transferido, ya hemos dejado Arduino listo para comunicarse con LabVIEW





6. Algunas preguntas sobre la interface LIFA

(Traducido de <https://decibel.ni.com/content/docs/DOC-16024>)

1. ¿Qué es la interfaz de LabVIEW para Arduino?

La interfaz de LabVIEW para Arduino (LIFA) Toolkit [Free Toolkit](#) es un conjunto de herramientas gratuitas que permiten a los desarrolladores adquirir datos desde el microcontrolador Arduino y procesarlo en el entorno de programación gráfica de LabVIEW.

2. ¿Qué versiones del entorno LabVIEW permiten la conexión con Arduino?

La interfaz de LabVIEW para Arduino es actualmente compatible con cualquier versión de Windows o Mac OS que soporta LabVIEW 2009 o posterior. El kit de herramientas también funcionan en cualquier versión de Linux que soporta LabVIEW 2009 o posterior, sin embargo actualmente no existe un instalador (JKI VI Package Manager) para Linux. JKI está trabajando actualmente en VIPM 2010 para Linux, que estará disponible aquí cuando haya terminado.

3. ¿Qué versión de LabVIEW Qué necesito para utilizar la interfaz de LabVIEW para Arduino?

2009 o posterior.

4. ¿Qué hardware es necesario para utilizar la interfaz de LabVIEW para Arduino?

Para empezar, el único hardware que se necesita es una tarjeta Arduino, cable USB y un ordenador con LabVIEW y la interfaz de LabVIEW para Arduino. La interfaz de LabVIEW para Arduino fue desarrollado y probado usando Arduino UNO y Arduino MEGA 2560

5. ¿Cómo se instala la interfaz de LabVIEW para Arduino?

Siga los pasos de este documento. [Instalación](#).

6. ¿Dónde se puede obtener soporte para la interfaz de LabVIEW para Arduino?

Soporte para la interfaz de LabVIEW para Arduino se proporciona en los foros de la comunidad [community forums](#). La interfaz de LabVIEW para Arduino no está

soportada por Ingenieros de Aplicaciones a través del teléfono, correo electrónico, o de otra manera.

7. ¿Cómo puedo empezar a utilizar la interfaz de LabVIEW para Arduino?

Después de instalar el kit de herramientas que puede ver [aquí](#) la mejor manera de empezar es mediante los ejemplos que viene en el paquete LIFA de LabVIEW. Se encuentran en ...*National Instruments\LabVIEW 2012\examples\LabVIEW Interface for Arduino*

8. ¿Puedo implementar el Código de LabVIEW a mi Arduino?

No. En este momento no es posible implementar el Código de LabVIEW para la placa Arduino. Solo es posible utilizar la interfaz de LabVIEW para Arduino para comunicarse con la placa Arduino utilizando LabVIEW en modo “esclavo” On-line

9. ¿Tengo que ser tener a un ordenador para utilizar la interfaz de LabVIEW para Arduino?

Sí, sin embargo, puede ser "sin cables " mediante el uso de una o XBee BlueSMiRF

¿Cómo configuro la interfaz de LabVIEW para Arduino para usar una conexión Bluetooth a mi Arduino?

- [XBee](#) or [BlueSMiRF](#)
- [How Do I Setup the LabVIEW Interface for Arduino to use a Bluetooth Connection to my Arduino?](#)

10. ¿Puedo agregar mis propios sensores a la interfaz de LabVIEW para Arduino?

Sí. Tanto el firmware y el VIS son de código abierto pensado para la personalización.

11. ¿Cómo funciona la interfaz de LabVIEW para Arduino (LIFA) de trabajo?

En pocas palabras, la interfaz de LabVIEW para Arduino envía paquetes de datos de LabVIEW para Arduino. Arduino procesa los paquetes y envía paquetes de retorno. Los paquetes de retorno se analizan por LabVIEW para proporcionar información útil para el usuario final. Cada paquete es de 15 bytes por defecto y contiene un encabezado, un byte de comando, los bytes de datos, y una suma de comprobación. La longitud del paquete se puede cambiar para adaptarse a aplicaciones específicas mediante la modificación del firmware y especificar el tamaño del paquete a la VI Init en LabVIEW (La mayoría de los usuarios no tendrán que hacer esto). El firmware LIFA en el Arduino procesa los paquetes, asegurando que los datos no se han dañado durante la transmisión. A continuación, una vez leído el paquete, comprueba el byte de comando y ejecuta las instrucciones con los bytes de datos proporcionados basándose en el byte de comando.



7. Desarrollo de Aplicaciones Básicas.

7.1. Lectura de una entrada Digital

Para empezar con nuestros ejemplos prácticos lo haré con la práctica más sencilla: Leer el valor de una entrada digital de Arduino.

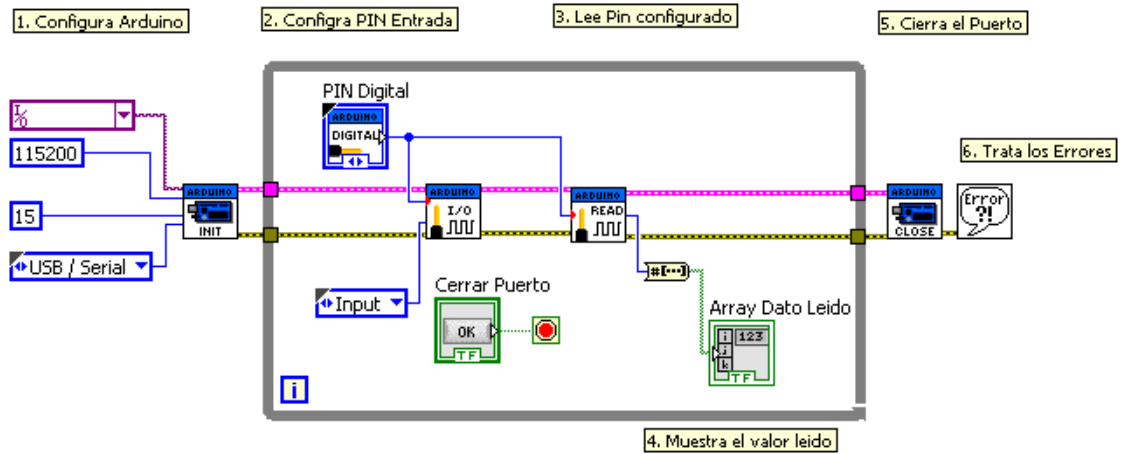
Se trata de realizar un montaje que permita seleccionar el **PIN digital** desde la propia pantalla del Panel y que mediante un Led podamos ver su estado.

En la siguiente figura se muestra el Panel




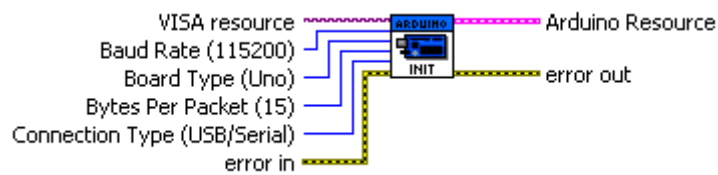
Para empezar debemos configurar Arduino y lo hacemos poniendo el bloque **“Init”** al que le asignamos los parámetros:

- Puerto de comunicación
- Velocidad de transmisión
- Tipo de tarjeta Arduino
- Numero de bits de los paquete de comunicación
- Tipo de puerto de comunicación.



No olvidemos que estos parámetros a excepción del puerto de comunicación los asume por defecto el sistema, es decir que si no los ponemos el sistema los toma con esos valores.

Para asignar las constantes mencionadas basta ponerse sobre el terminal con la herramienta de edición en modo “wire”  y pulsando el botón derecho del ratón podemos seleccionar el control a añadir con la opción “Create” (Constant, Control, Indicador).



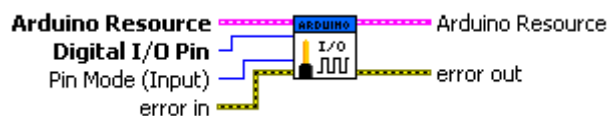
A continuación se coloca una estructura de tipo “While loop”

While Loop



que se ejecutara continuamente hasta que pulsemos el botón “Cerrar puerto”, viene a ser el equivalente al “loop” de un programa escrito para Arduino.

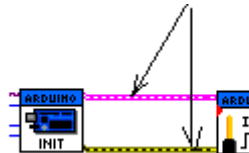
Dentro de esta estructura pondremos el bloque de configuración de E/S y el de lectura de valor de Entrada de la tarjeta Arduino




En la entrada “Pin Mode” debemos seleccionar “INPUT” y la entrada “Digital I/O PIN” deberemos unirla a un bloque “PIN Digital” que creara el control correspondiente en el Panel y que en modo de ejecución permitirá cambiar la entrada a leer.

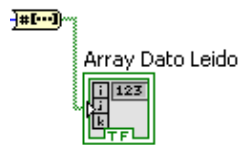
IMPORTANTE: Es muy importante que se sepa que los **PIN 0** y **PIN 1** digitales están ocupados en la comunicación con LabVIEW por lo tanto nunca se deben seleccionar ni para leerlos ni para escribir en ellos.

No debemos olvidarnos de realizar el cableado de los buses de conexión entre módulos:



La salida del dato leído es un dato tipo “Integer de 8 bits” por lo tanto se deberá realizar la conversión a dato tipo booleano **number**  Boolean array


Para después llevarlo a un indicador de tipo array booleano TRUE FALSE.



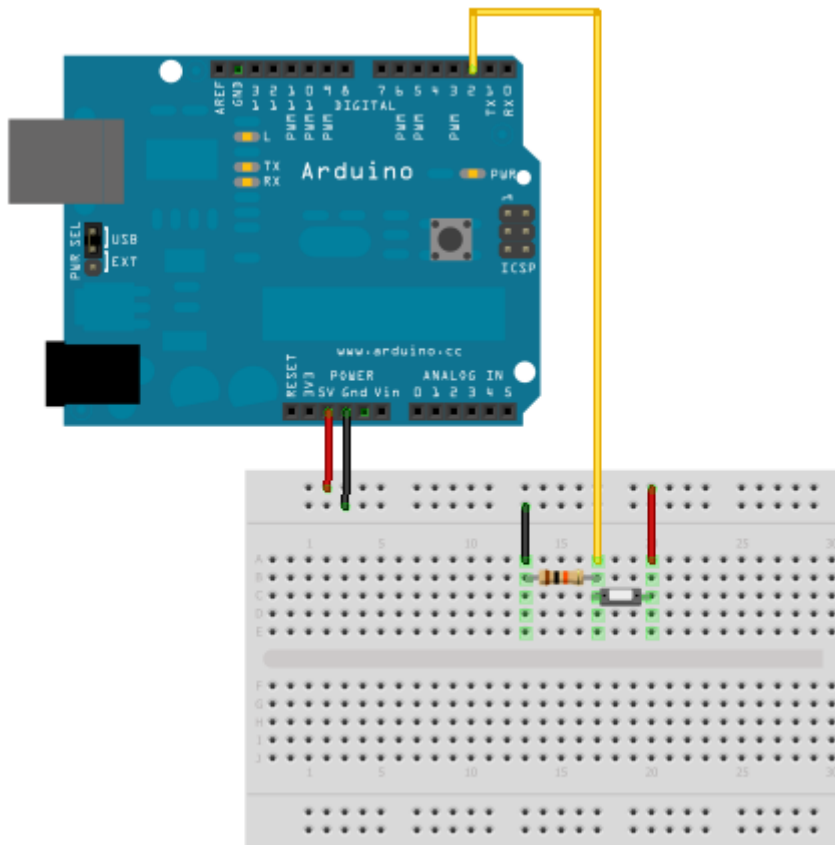
cuyo referente en el Panel es el mostrado



La manera de operar ser ejecutar el ejemplo construido y probar su funcionamiento.

Pulsando  para detener la ejecución siempre se debe hacer pulsando en el botón del Panel “**Cerrar**”

En la figura vemos el montaje de la aplicación en el caso de testear la entrada 2 **PIN 2**

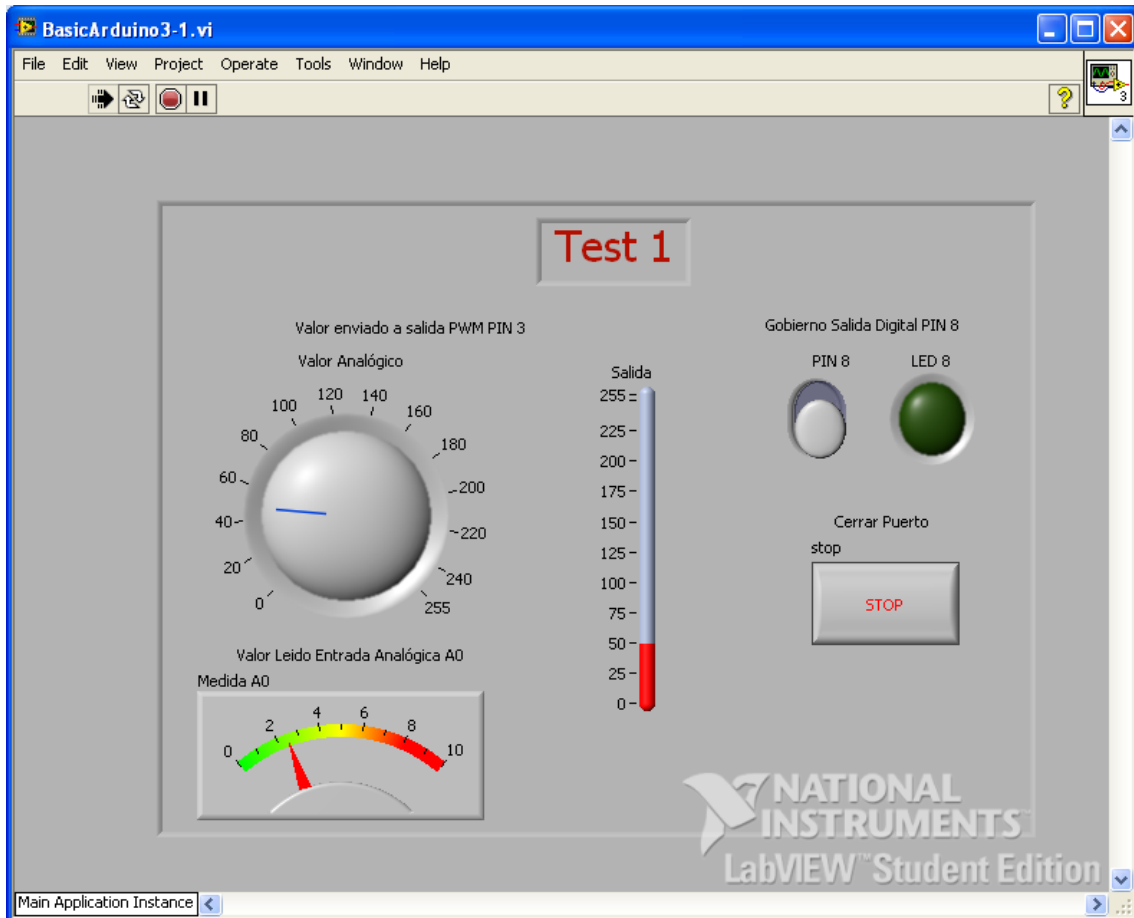


Made with  Fritzing.org

7.2. Test 1

En el siguiente ejemplo se pretende realizar la lectura y escritura de valores en la tarjeta Arduino.

Se enviará un valor analógico a la salida **PWM PIN 3** que obtendremos de un elemento de panel. Se leerá el valor del canal de entrada analógica **A0** y se mostrará en un instrumento de aguja en el panel a la vez que en un instrumento “termómetro”. Finalmente se escribirá un valor digital en el **PIN 8** mediante un interruptor en el panel.



En la figura vemos el aspecto de nuestro Panel.

Tal y como se puede observar en el diagrama de funciones de la siguiente figura procederemos de la siguiente manera.

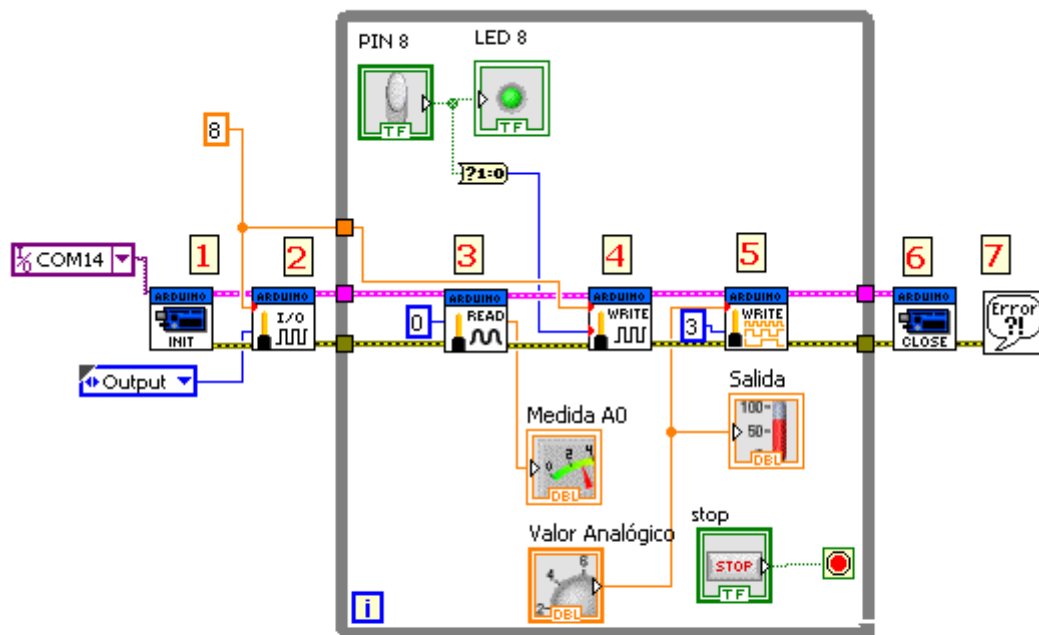
En primer lugar colocamos el bloque de inicialización “**Init**” y le asignamos el parámetro de número de puerto, el resto le dejamos los que toma por defecto. Seguidamente configuramos el **PIN 8** como salida.

Dentro del bucle “**While loop**” procederemos a colocar los siguiente elementos:

Un bloque de lectura de señal analógica “**Analog Read Pin**” tal como se muestra en la figura.

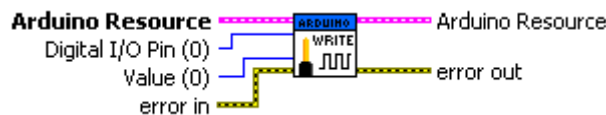


Este bloque necesita que le pongamos el valor del canal de entrada analógica “**Analog Input Pin**” y en su salida nos entrega un valor tipo Double que se corresponde con la lectura realizada. La salida la encaminamos a los instrumentos de medida MedidaA0 que ese corresponde con el medidor de aguja del panel.



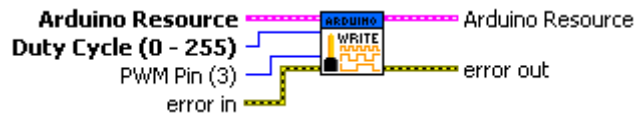
1. Configura la tarjeta Arduino
2. Configura el PIN 8 como salida digital
3. Lee el valor del canal analógico A0
4. Escribe un valor digital en el PIN 8
5. Escribe una señal de salida analógica PWM en el PIN 3
6. Cierra el puerto.
7. Trata los errores

El siguiente bloque que debemos colocar es el correspondiente a la salida digital en el **PIN 8**



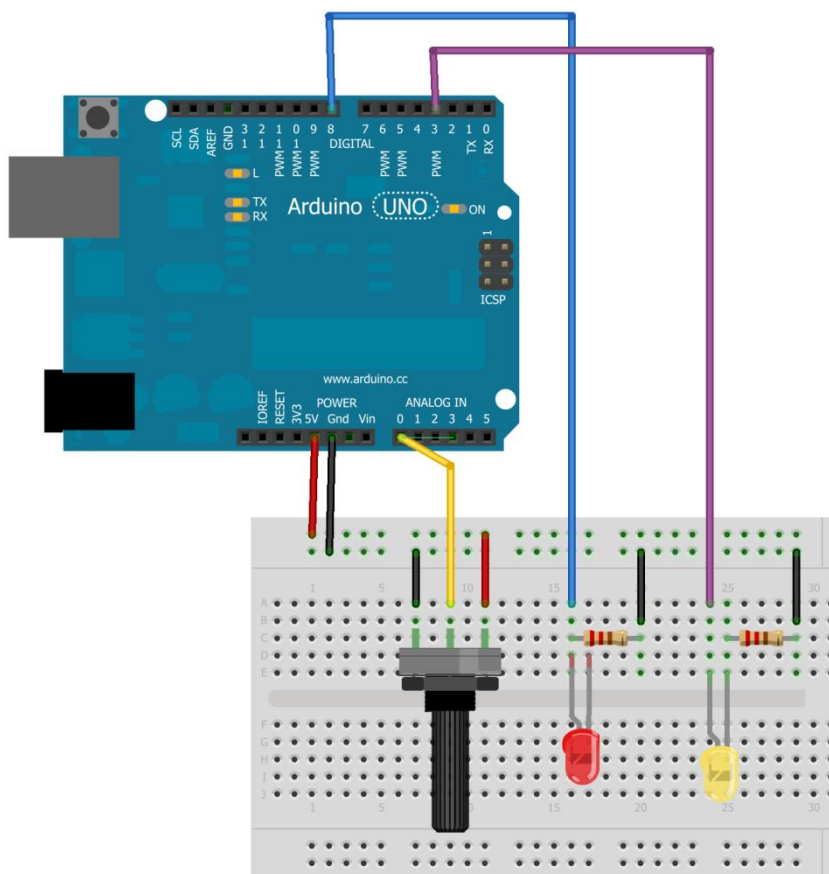
El valor que representa el numero de PIN lo recibe de la correspondiente constante “8” que ya tenemos puesta para la configuración del PIN y el valor “*Value*” que queremos sacar en la salida lo tomamos de un interruptor “**PIN 8**” que a la vez también sacamos a un Led **LED 8**. Ambos en el panel.

El último bloque de función que colocaremos dentro del bucle es el de escritura del valor analógico **PWM** en el **PIN 3**. Lo haremos configurando el numero de PIN “3” y mediante un mando tipo potenciómetro “*Valor analógico*” designaremos el valor de la entrada “*Duty Cycle (0-255)*”, conectando también un indicador tipo termómetro “*Salida*”.



Recordemos que para Arduino UNO las salidas tipo **PWM** son los pines digitales **3,5,6,9,10** y **11**

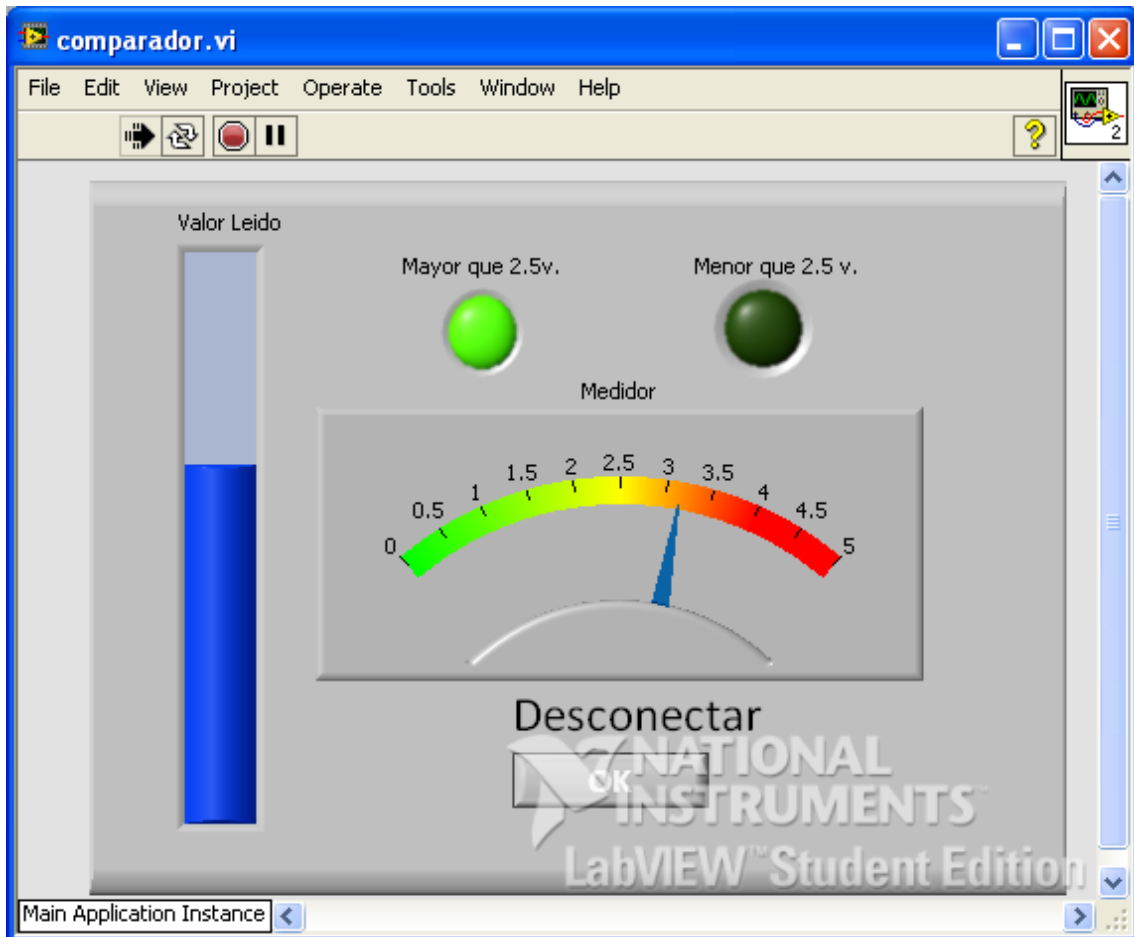
Finalmente, ya fuera del bucle colocamos el bloque de cierre del canal “**Close**” y el bloque de tratamiento de error, que nos permita mostrar en la pantalla una ventana con información sobre los posibles errores que se produzcan.



7.3. Comparador

En este ejercicio vamos a realizar una aplicación que implica un sencillo cálculo matemático: una comparación.

Mediremos un valor analógico tomado del canal **A0** y lo compararemos con una consigna, en este caso 2,5 realizaremos la comparación $>$ y $<$ y recogeremos el resultado de estas dos comparaciones sobre unos Leds indicadores



En la figura anterior se muestra el aspecto del panel.

El procedimiento de elaboración del diagrama funcional es muy sencillo.

Dentro de estamento **“While loop”** colocamos el bloque de lectura de valor analógico **“Analog Read pin”** que ya hemos utilizado en la anterior practica.

La comparación se realiza con dos bloques de función comparación tal como se muestra en la figura. Por un lado ponemos el valor de comparación “2.5” y por otro la señal leída del canal. Las salidas de los bloques se llevan a sendos diodos Leds indicadores: *“Mayor que 2.5”* y *“Menor que 2.5”*

7.4. Contador de Impulsos

Vamos a implementar una aplicación que nos permita contar los impulsos que se reciben a través de un pulsador en la entrada PIN (seleccionado por nosotros) y lo muestre en el panel.

Colocaremos un LED que nos indique que el impulso recibido y un botón para parar

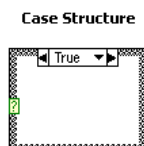


En la figura anterior se muestra el aspecto del Panel.

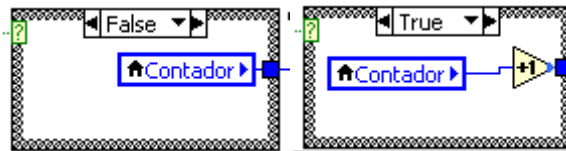
Para realizar el diagrama funcional, como siempre colocaremos el bloque de Inicio.

Dentro del bloque **“While loop”** pondremos dos bloques de la librería de Arduino: Un bloque para configurar el PIN de entrada **“Set Digital Pin Mode”** y otro para leer el valor **“Digital Read Pin”**.

El contador propiamente dicho se implementa con la ayuda de un bloque del tipo **“Case Structure”**.

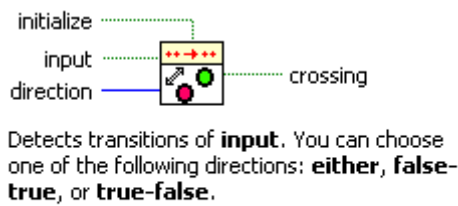


Este bloque tendrá dos estados posibles tal como se muestra en las siguientes figuras:



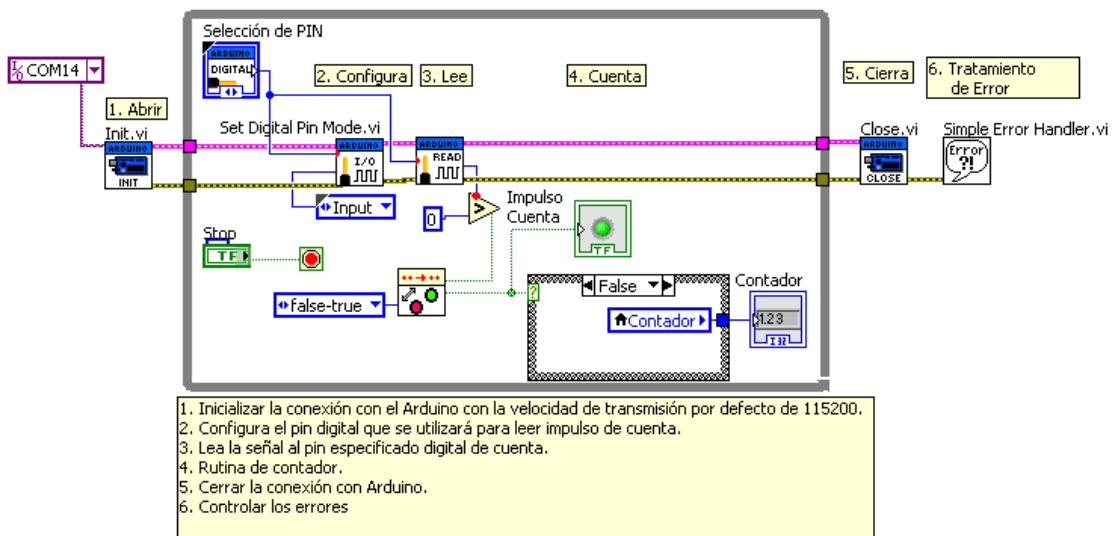
En el caso “False” no hay impulso de cuenta el contador no sufre incremento y su valor se transmite directamente a la salida. En el caso “True”, la entrada al bloque sera TRUE y se activa el contador “*incrementándose en 1*”.

Para obtener la señal de gobierno de esta estructura se ha recurrido a un bloque que detecta transiciones de TRUE a FALSE.



Terminal Data Type

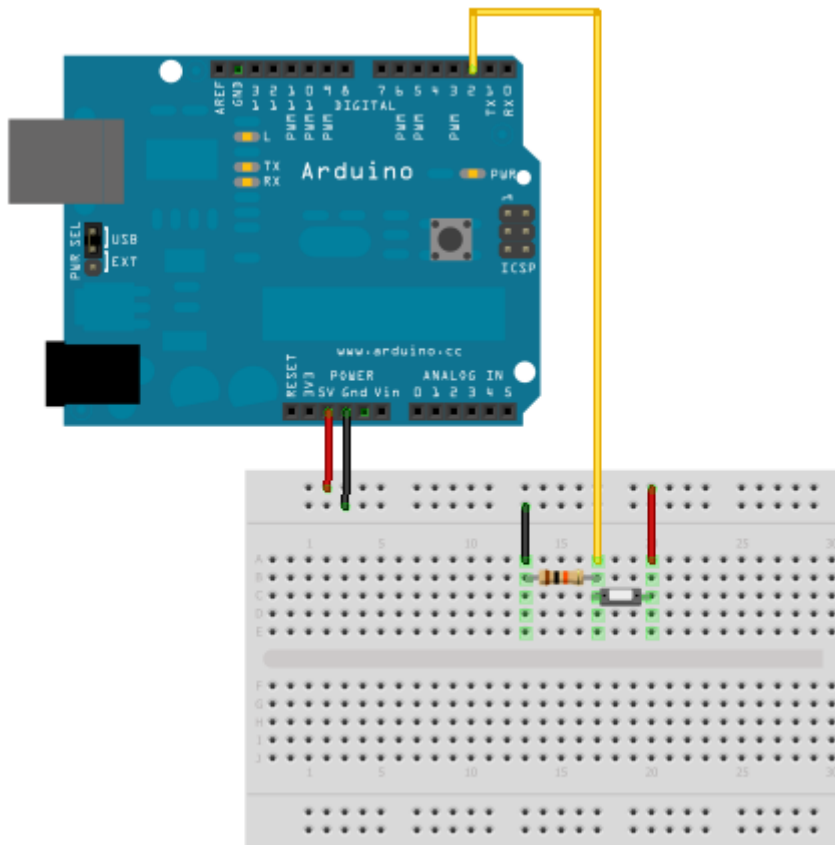
TF crossing (boolean (TRUE or FALSE))



La salida del bucle “Case Structure” se obtiene un valor tipo Integer que se lleva a un indicador.

Finalmente se conectan los bloques de “Cierre” de canal y tratamiento de errores.

En la figura siguiente se muestra u esquema de conexionado para realizar las pruebas físicas de funcionamiento.

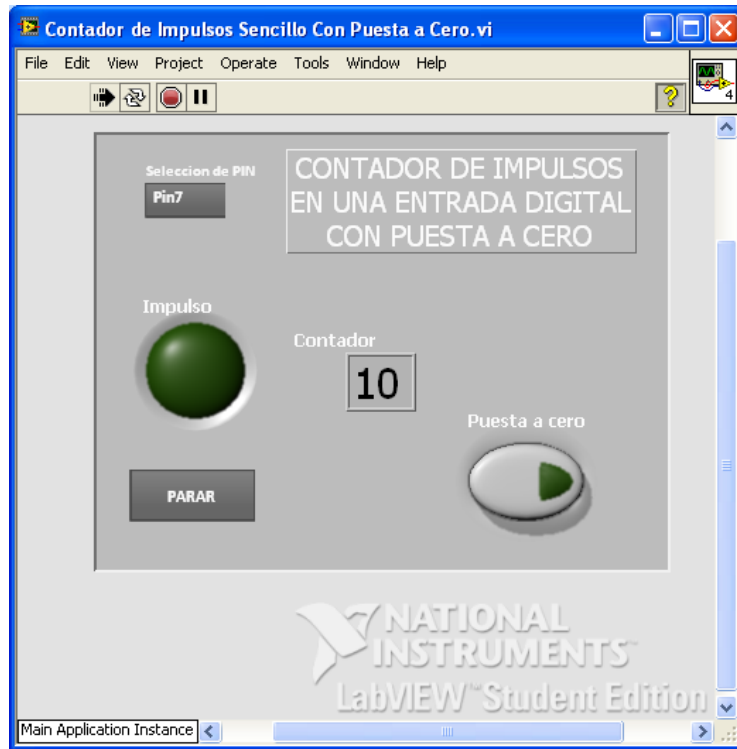


Made with  Fritzing.org

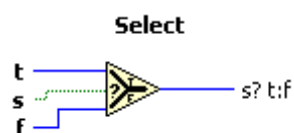
7.5. Contador de impulsos con puesta a cero

En el siguiente ejemplo añadimos al contador explicado en el anterior la posibilidad de poner a cero el contador.

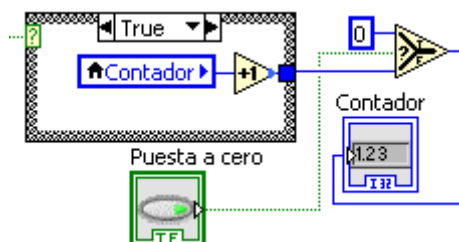
Para ello todo será igual a excepción de que colocaremos un botón de **“Puesta a cero”** que llevara el contador al valor “0”



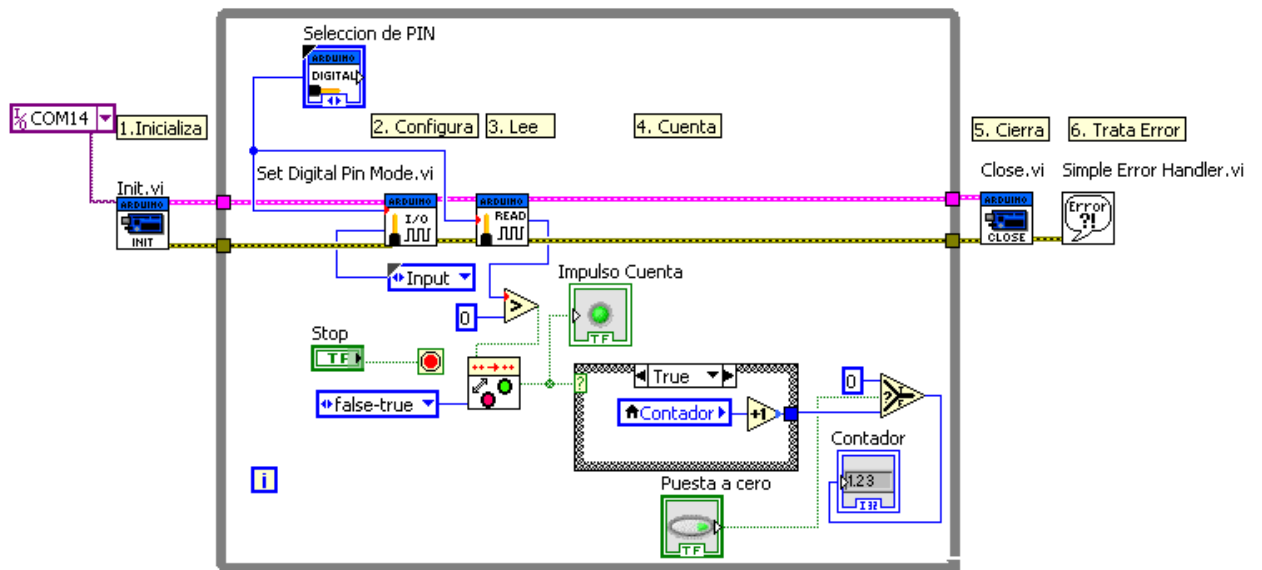
En el esquema de bloques funcionales vemos que el sistema es el mismo a excepción de que hemos añadido una función del tipo **“Select”**.



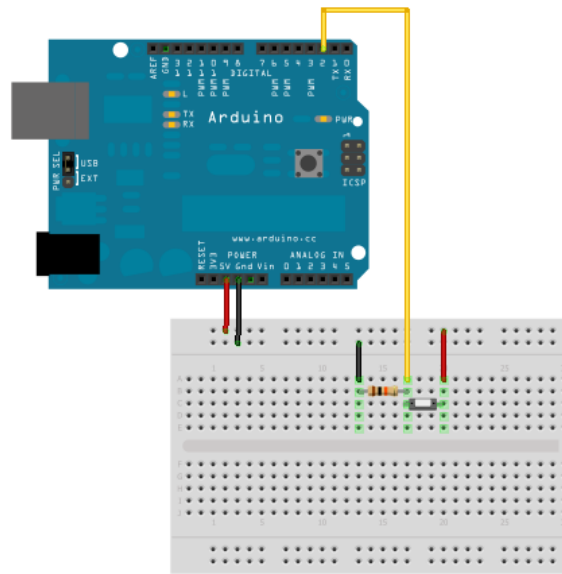
Esta función saca el valor de *“t”* cuando la entrada *“s”* es TRUE y saca *“f”* cuando su entrada *“s”* es FALSE.



A continuación mostramos el esquema completo.



1. Inicializar la conexión con el Arduino con la velocidad de transmisión por defecto de 115200.
2. Configurar el pin digital que se utilizará para leer.
3. Le la señal al pin digital especificado.
4. Rutina de cuenta
5. Cerrar la conexión con Arduino.
6. Controlar los errores



Made with Fritzing.org

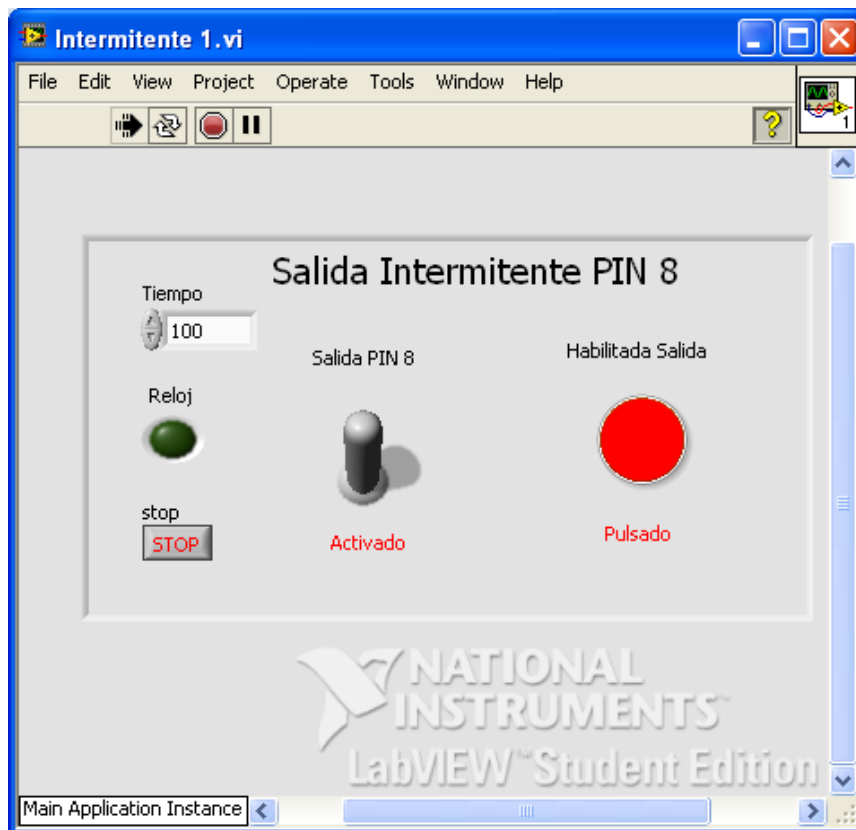
Este sería el esquema de pruebas

7.6. Intermitente

Abordamos en este ejemplo el clásico ejemplo con el que se comienza a estudiar Arduino: una salida intermitente en uno de los Pines digitales.

Vamos a activar la salida digital **PIN 8** de modo intermitente con intervalos de tiempo ajustables desde el Panel de control en tiempo de ejecución.

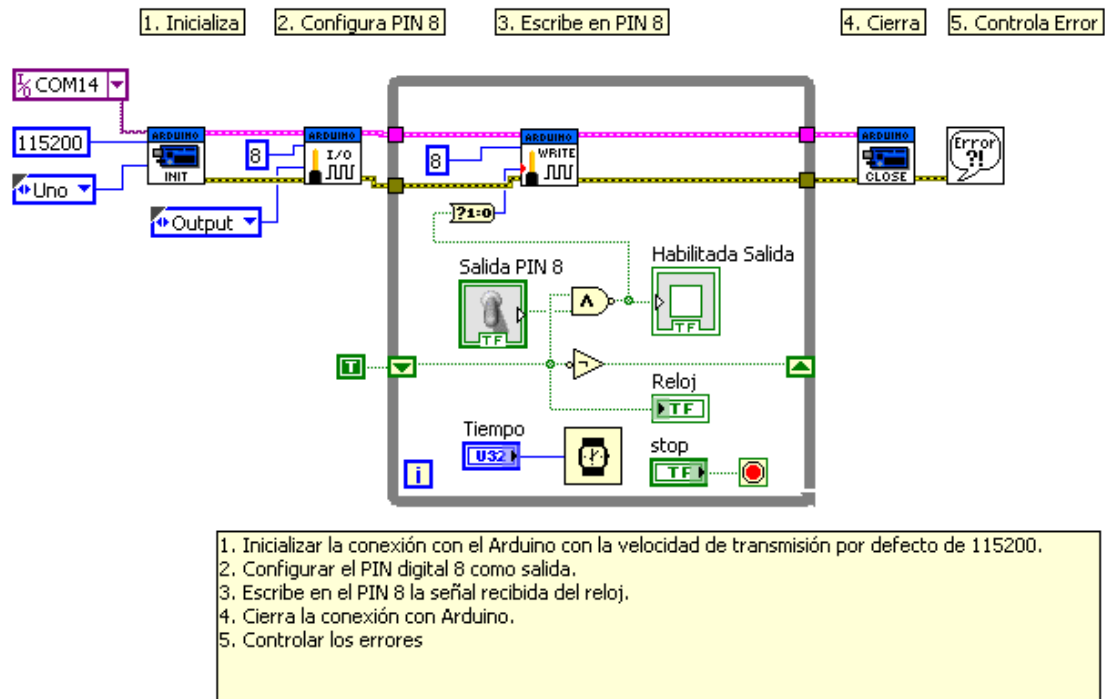
En este caso recurrimos a la ejecución cada cierto tiempo del contenido de nuestro bucle “**While loop**”



En la anterior imagen vemos el aspecto del Panel y en la siguiente vemos el esquema de bloques funcional de la aplicación.

Como siempre inicializamos Arduino y después definimos el **PIN 8** como una salida.

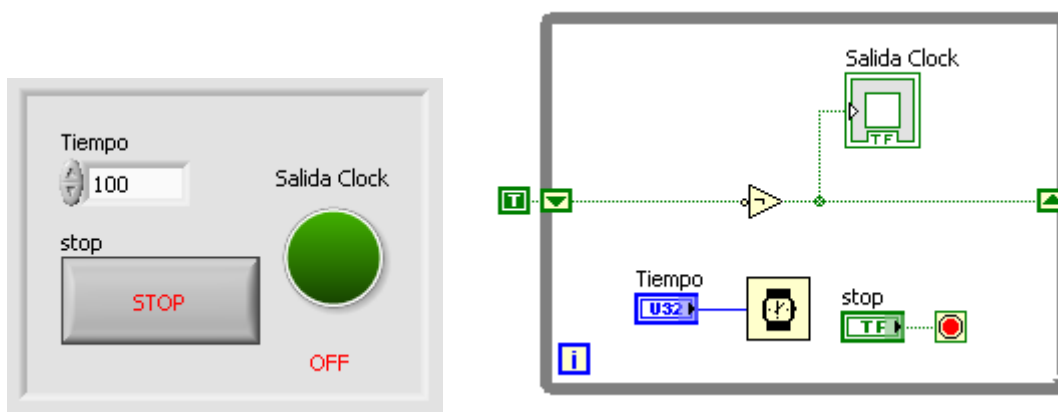
Dentro el bucle colocamos la función de escritura “**Digital Write Pin**”. Este bloque recibe la señal digital de la estructura que constituye el “*oscilador*” de frecuencia variable: Implementación del reloj



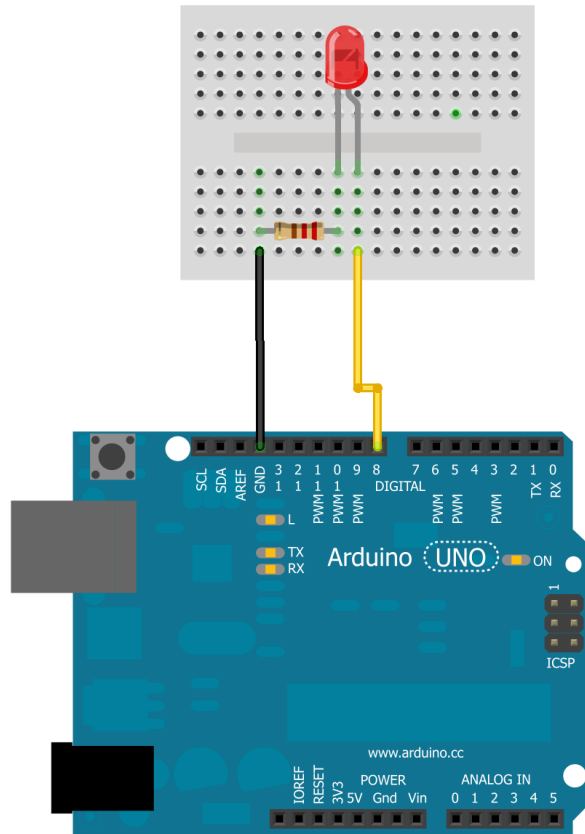
Implementación de un reloj

En el bucle “While Loop” añadimos un “Shift Register” simplemente pulsado en botón derecho del ratón estando exactamente sobre el contorno del bucle, opción “Add Shift Register”. De esta forma conseguimos que se ejecute sistemáticamente cada cierto

tiempo (el indicado en la opción “Tiempo”) la operación que hay dentro del bucle. Se trata de sacar TRUE y FALSE a través de la señal “Reloj”.



Se ha colocado un operador AND para habilitar mediante un interruptor (Salida PIN 8) el paso de la señal de reloj al bloque “Digital Write Pin” de Arduino. Se han colocado también dos indicadores LED para muestra las señales de Reloj “Salida Clock” y la que activa la entrada del bloque Arduino de salida digital “Habilitada Salida”



Este sería el montaje para realizar las pruebas reales de la aplicación.

7.7. Semáforo Simple.

Los circuitos de tiempo son muy utilizados en los automatismos. Uno de los mas clásicos ejemplos de estos circuitos es un semáforo.

En la siguiente practica realizaremos un semáforo simple.

Utilizaremos las siguientes salidas digitales para cada una de las tres lámparas del semáforo:

PIN Digital	Salida	Tiempo
8	Rojo	1000 ms.
9	Ámbar	1000 ms.
10	Verde	700 ms.

El tiempo de activación de cada lámpara en este primer ejemplo será fijo y de 1seg.

En la figura siguiente vemos el aspecto del Panel en modo ejecución.

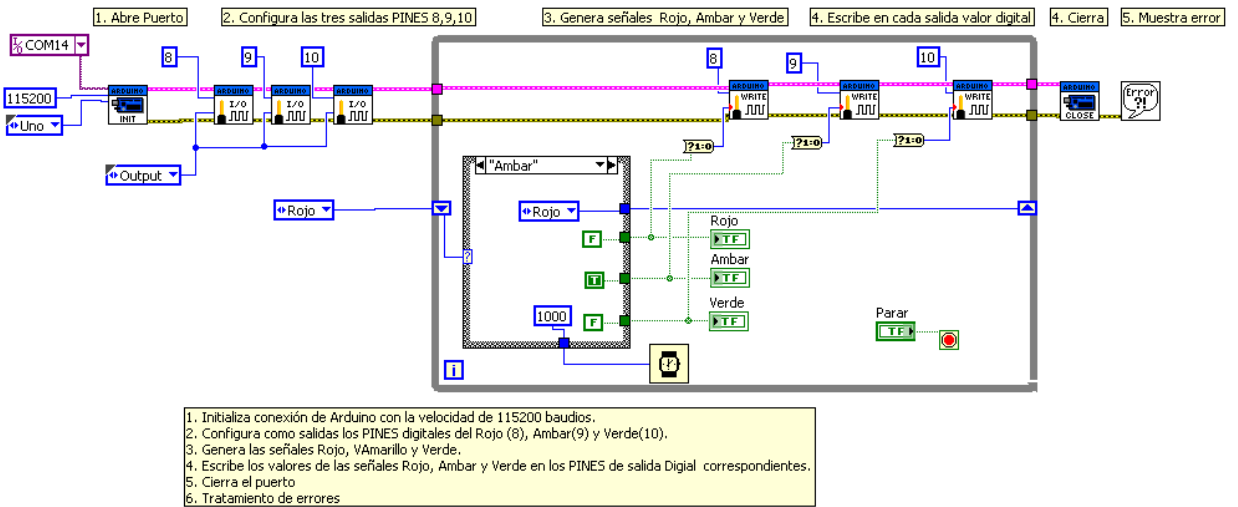


Se muestran las tres lámparas y el botón “Parar”.

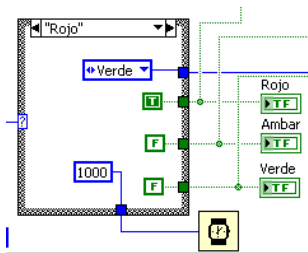
El proceso que seguiremos en el montaje es el siguiente:

1. Inicializamos la conexión de Arduino mediante el bloque “Init”.
2. Configura como salidas de cada una de las tres lámparas: **PIN (8) Roja**, **PIN (9) Ámbar** y **PIN (10) Verde**. Esto lo hacemos mediante los tres bloque de función “Set Digital Pin Mode” de la librería de Arduino

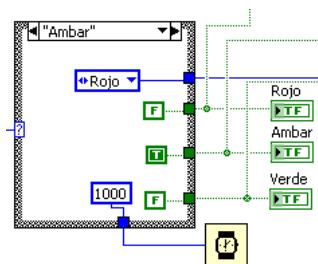
3. Genera las señales Rojo, Amarillo y Verde. Mediante una estructura tipo “Case Structure” que en nuestro caso le añadiremos hasta “tres casos” o estados que se asociaran a cada uno de los estados de nuestro semáforo.



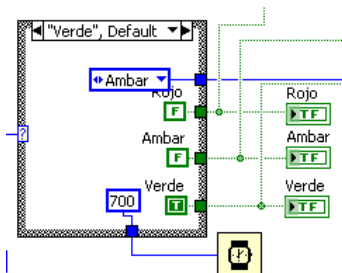
En las figuras siguientes se muestra cada uno e los casos creados. Hemos definido como estado de inicio en la secuencia de ejecución el “Rojo”



Estado Rojo. En el vemos que el estado siguiente debe ser “Verde” y que la secuencia será: Rojo (TRUE), Ámbar (FALSE) y Verde (FALSE). Tiempo 1000 ms.

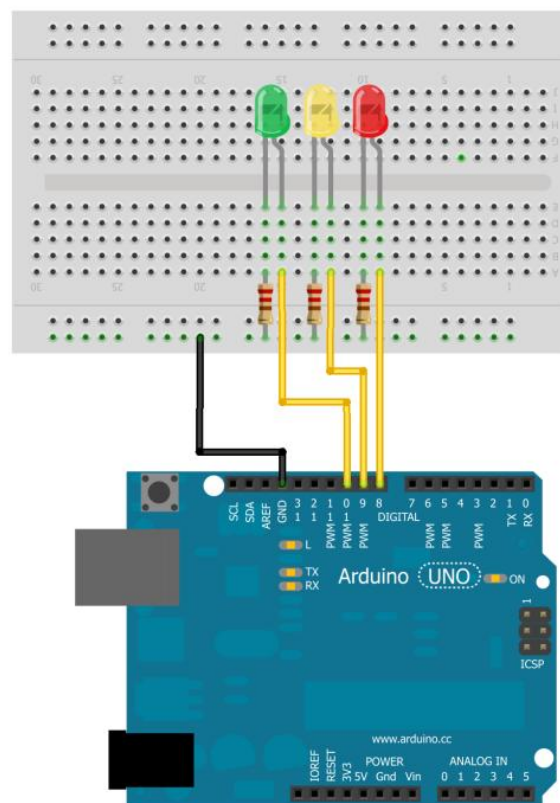


Estado Ámbar. En el vemos que el estado siguiente debe ser “Rojo” y que la secuencia será: Rojo (FALSE), Ámbar (TRUE) y Verde (FALSE). Tiempo 1000 ms.



Estado Verde. En el vemos que el estado siguiente debe ser “Ámbar” y que la secuencia será: Rojo (FALSE), Ámbar (FALSE) y Verde (TRUE). Tiempo 700 ms.

4. Escribe los valores de las señales Rojo, Ámbar y Verde en los PINES de salida Digital correspondientes. Es importante que observemos como las salidas del secuenciador so de tipo "TRUE/FALSE" por eso debemos convertirlas al tipo de señal admisible por los bloques "**Digital Write Pin**" que escriben los valore en las salidas físicas de Arduino.
5. Cierra el puerto mediante el bloque "**Close**"
6. Tratamiento de los errores. Mediante el bloque "**Simple Error**"



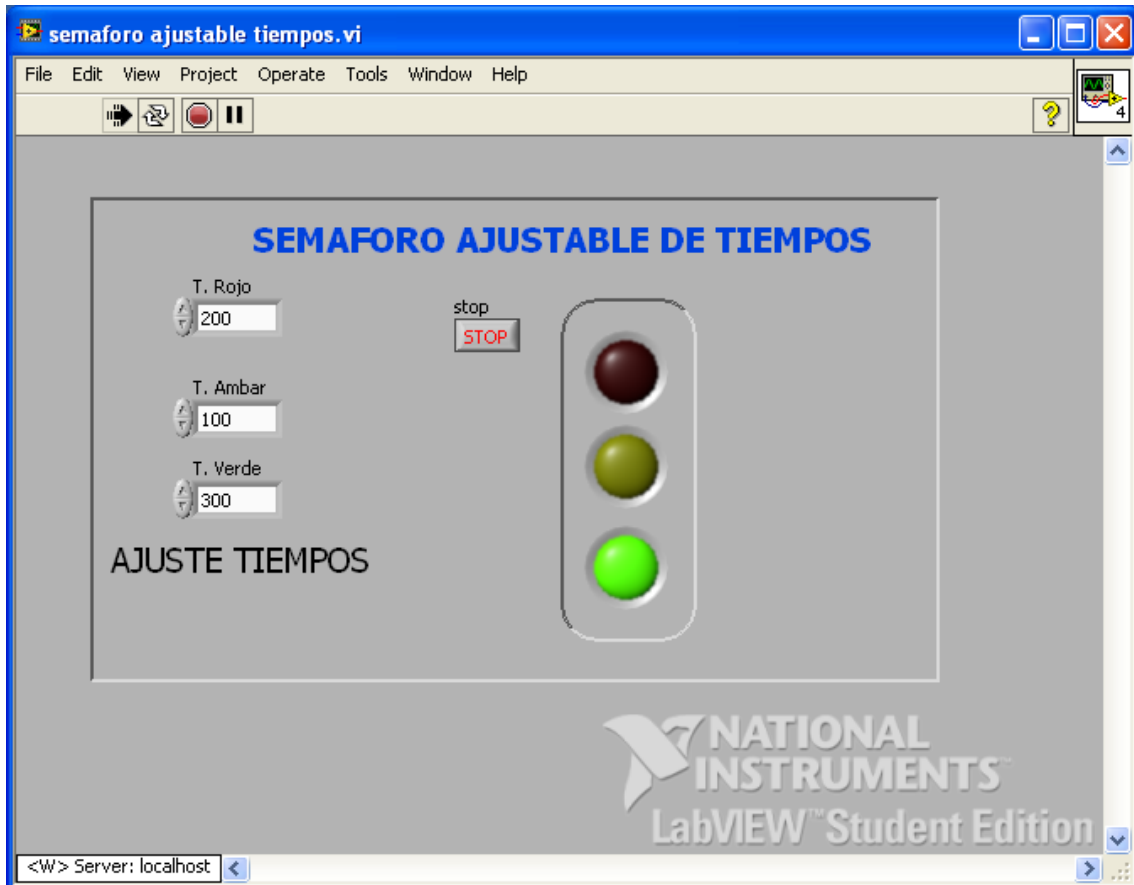
Made with  Fritzing.org

Este es el aspecto del montaje de pruebas

7.8. Semáforo Ajustable

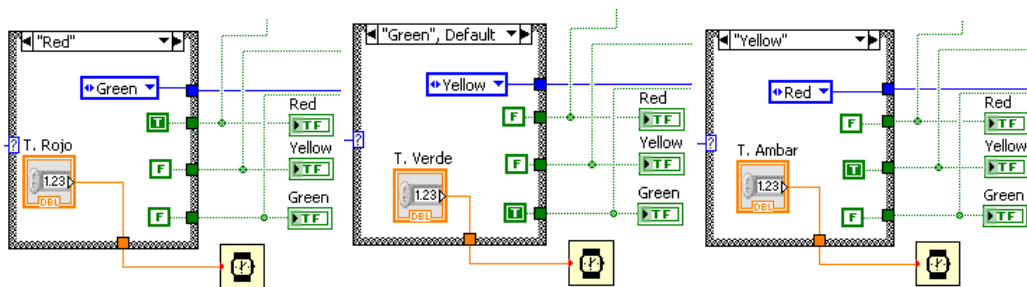
A continuación proponemos un ejercicio de semáforos en el que los tiempos de encendido de cada lámpara sean ajustables por el operador.

Los pines de salida son los mismos y en el Panel se han incluido los objetos de entrada de valor para cada uno de los tiempos: “T. Rojo”, “T. Ambar” y “T. Verde”.

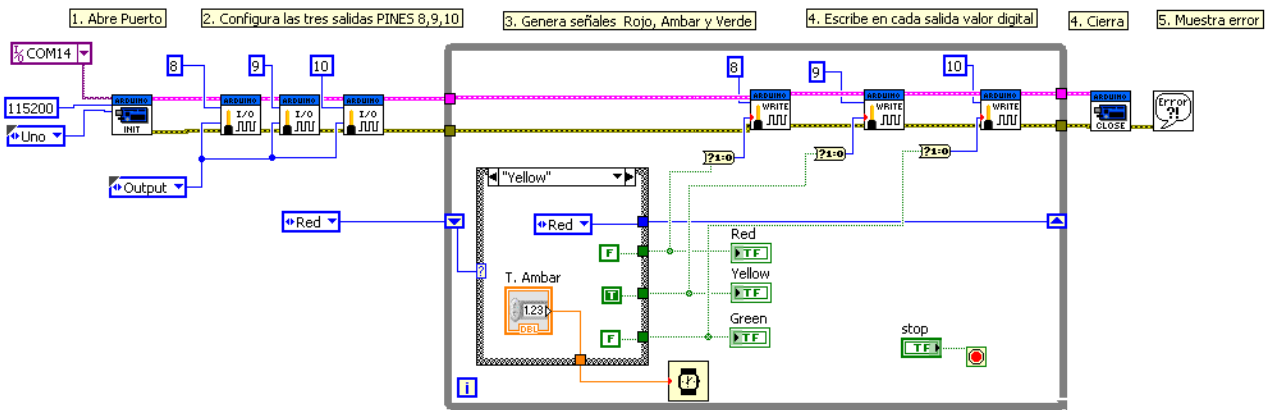


La realización funcional de la aplicación es la misma que la explicada anteriormente con la única diferencia de que en cada uno de los tres estados de la estructura “Case Structure” se ha incluido un elemento distinto para la designación de tiempo.

En las figuras que se muestra a continuación podemos distinguir cada uno de los bloques de captura de valor

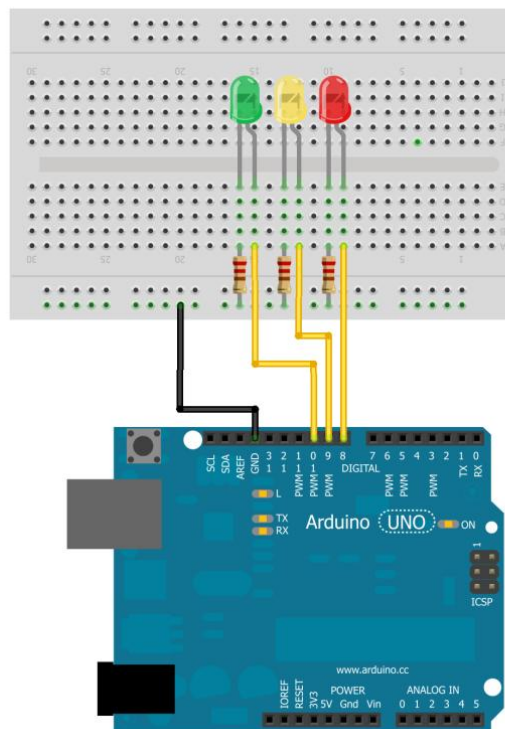


Finalmente se muestra el esquema funcional completo



1. Inicializa conexión de Arduino con la velocidad de 115200 baudios.
2. Configura como salidas los PINES digitales del Rojo (8), Ambar(9) y Verde(10).
3. Genera las señales Rojo, VAmarillo y Verde permitiendo designar los tiempos a cada color de manera independiente y desde el panel.
4. Escribe los valores de las señales Rojo, Ambar y Verde en los PINES de salida Digital correspondientes.
5. Cierra el puerto
6. Tratamiento de errores

Este sería el montaje de prueba

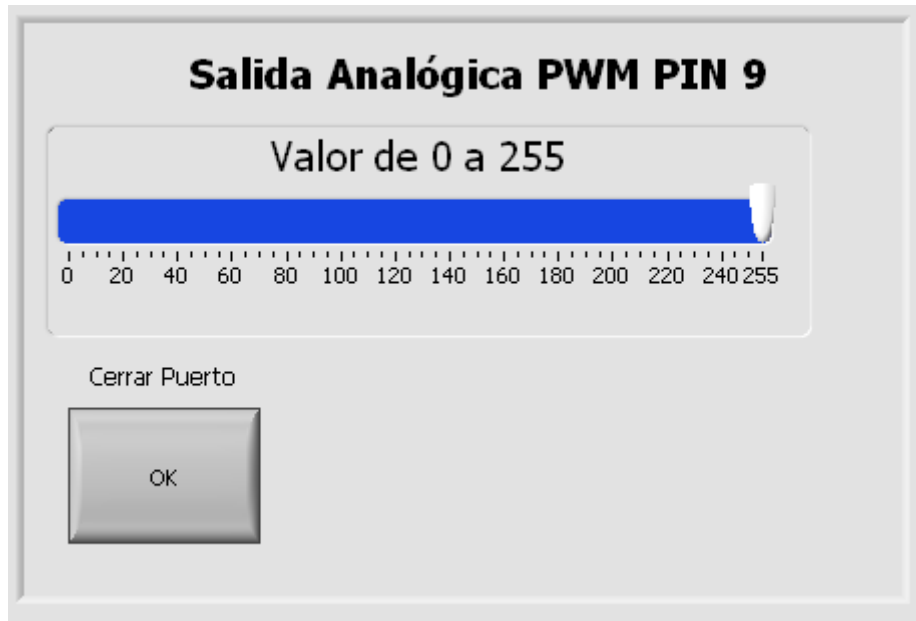


Made with Fritzing.org

7.9. Gobierno de una Salida Analógica PWM

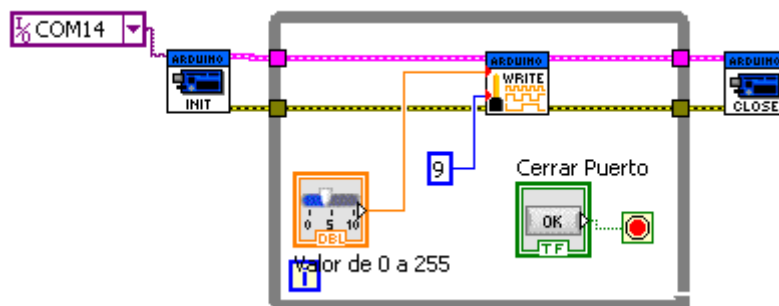
Sabemos que Arduino tiene la posibilidad de programar algunas de sus salidas digitales como salidas del tipo PWM (señal de pulso modulados en amplitud) que viene a ser una “cuasi” salida analógica. Estos pines para Arduino UNO son (Pines digitales 3,5,6,9,10 y11)

En este montaje vamos a enviar a la salida **PWM** del **PIN 9** un valor comprendido entre 0 y 255



El diagrama funcional de esta aplicación muy sencillo.

Por ser una salida digital no es necesario programarla como tal ya que por defecto todas los los Pines digitales están programados como salidas.

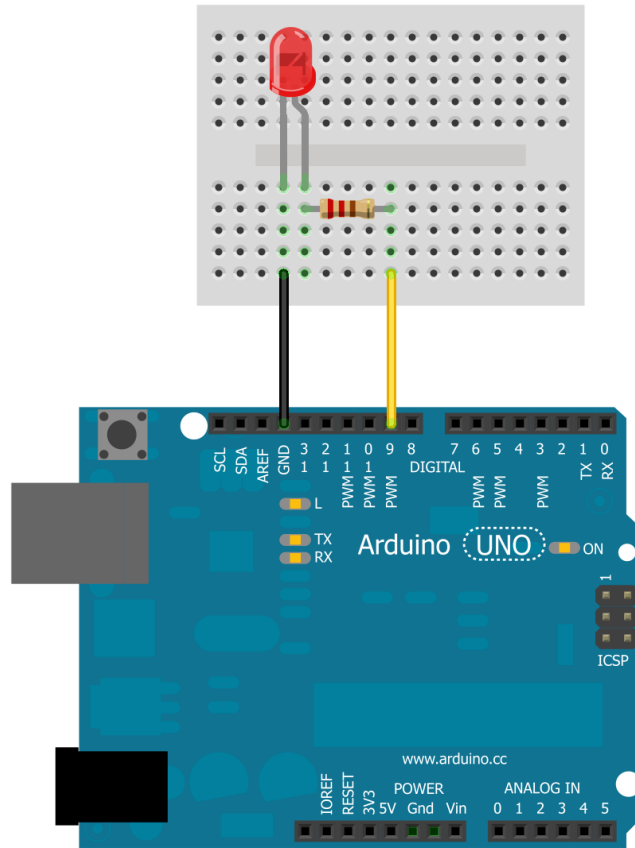


Se comienza con el bloque “**Init**” fuera del bloque “**While Loop**” y dentro se utiliza el bloque “**PWM Write Pin**” en el qe designamos que el PIN será el **9**.

La entrada de señal para este bloque la cableamos de un objeto del tipo “**Slide**” al que configuramos en la escala 0-255.

Hemos obviado el bloque de tratamiento de error dado que no es importante.

Este es el montaje de pruebas



7.10. Gobierno de una Salida Digital Seleccionada

El siguiente ejemplo sirve para manejar el gobierno de salidas desde el Panel de control.


Seleccionando previamente la salida correspondiente podremos activar o desactivar una de las salidas digitales de la tarjeta Arduino.

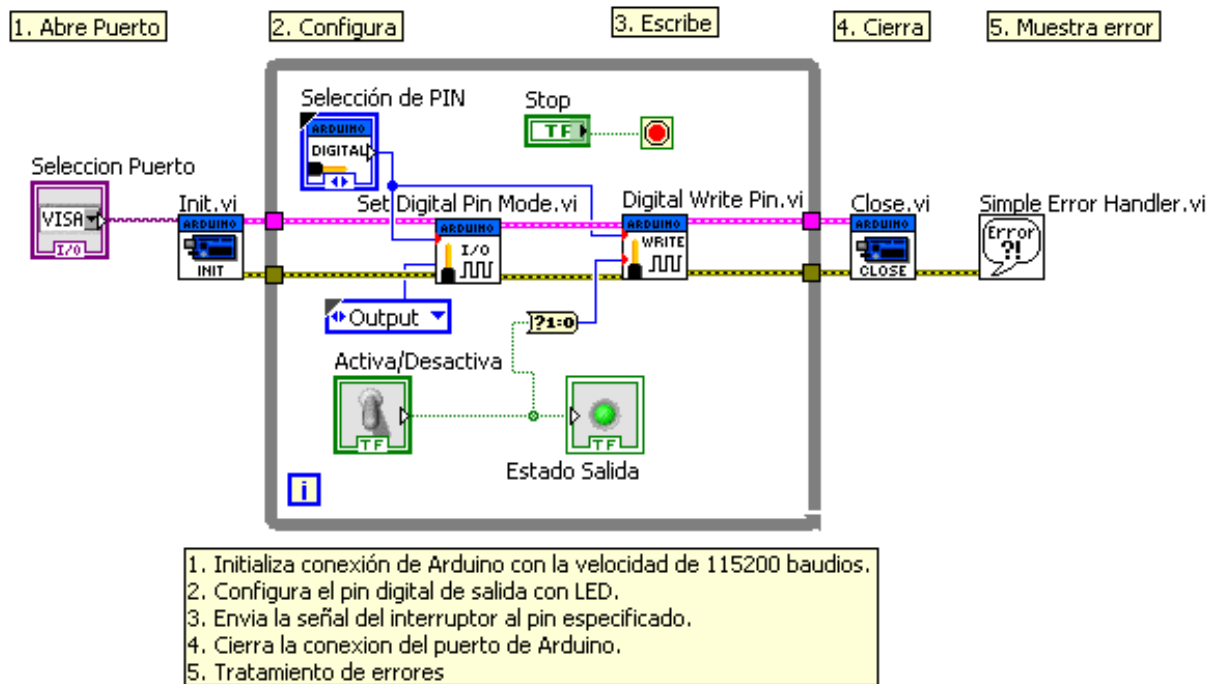
En el Panel se dispone de un selector de PIN y un interruptor para gobernar la salida seleccionada. Se ha colocado esta vez un selector de puerto de comunicación y un LED indicador del estado de la salida.



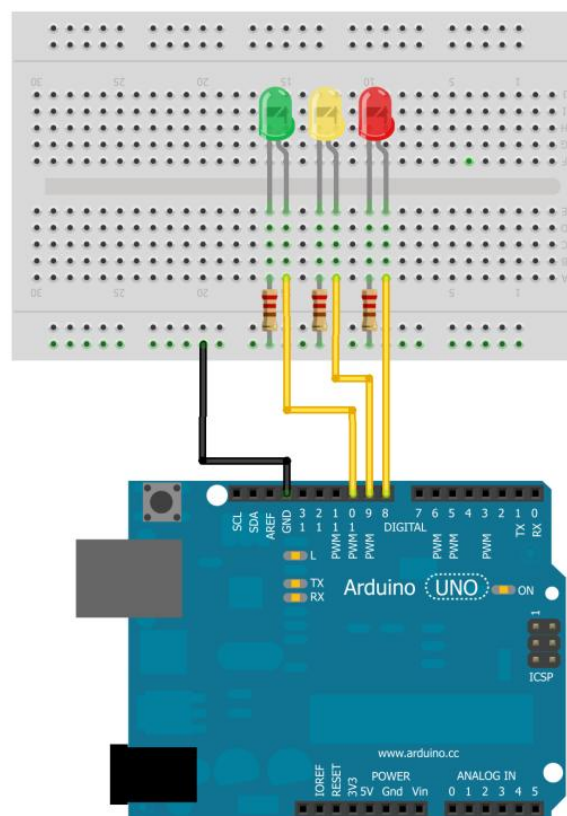
El diagrama funcional es muy sencillo.

Se realiza la inicialización de Arduino y luego, ya dentro el bloque de control de bucle se han colocado un elemento **“Select Digital Pun Mode”** para configurar el modo de trabajo de la salida seleccionada y un bloque **“Digital Write Pin”** que permite el envío de la señal al PIN seleccionado mediante el bloque **“Digital Pin”** que hemos etiquetado como *“Selección de PIN”*

Al bloque **“Digital Write Pin”** se le ha colocado en su entrada un interruptor que hemos etiquetado como *“Activa/Desavctiva”* que entrega un valor de tipo True/False y que convertimos en un valor tipo Integer  se ha colocado también el indicador de estado mediante un LED al que hemos etiquetado como *“Estado salida”*



La siguiente imagen muestra un circuito para probar el funcionamiento del programa. Se han conectado hasta tres diodos leds en las salidas 8, 9 y 10 para probar, pudiéndose cambiar las conexiones.



Made with Fritzing.org

7.11. Escritura/Lectura de todos los Canales

Con este ejemplo vamos a controlar todas las salidas monitorizando a su vez su estado en el panel. Para ello disponemos de dos arrays, uno de interruptores y otros de indicadores leds que nos permitirán las funciones de gobierno de salidas.

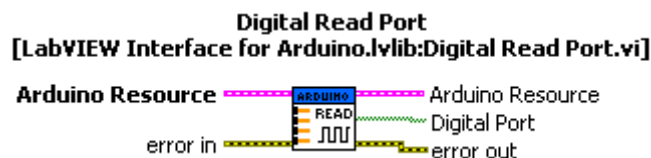
Se ha puesto también la posibilidad de leer el estado de los canales analógicos.



Aquí recurrimos a dos bloques de función de la librería Arduino hasta ahora no utilizados: **“Digital ReadPort”**, **“Analog Read Port”** y **“Digital Write Port”**.

Después de inicializar arduino con el bloque “Init”, dentro del bucle de control colocamos estos tres bloques.

Lectura del puerto digital: En este caso se trata de realizar la lectura del estado de todas las entradas digitales de Arduino. Se realiza con el v;bloque de la figura

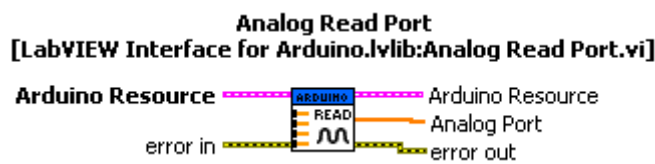


Es muy importante que se sepa que los canales digitales 0 y 1, PIN 0 y PIN 1 están reservados para la comunicación con LabVIEW por lo tanto su lectura obedece al tráfico de datos que se esté realizando y en ningún caso se podrán colocar interruptores de entrada en esos canales.

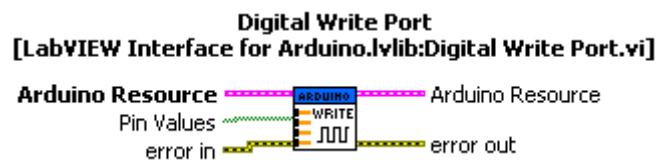
El dato que entrega el bloque es un *Array de dimensión 1D*. para poder visualizar estos datos tenemos que seleccionar un elemento de visualización de tipo Indicador que mostrará todo el array. Bastara, en el Panel de visualización estirar la caja del indicador para mostrar todo el contenido del array. Las salida en este caso se ha llevado a un indicador que hemos etiquetado con el nombre *“Lectura Canales Digitales”*



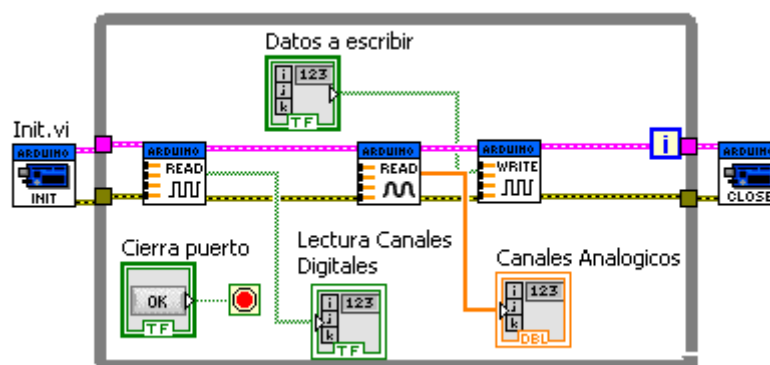
Para leer en un array todo el bloque de entradas analógicas, dese el canal A0 hasta el A5 se recure a este bloque **“Analog Read Port”** que entrega en su salida un array con los valores de los 6 canales analógicos.



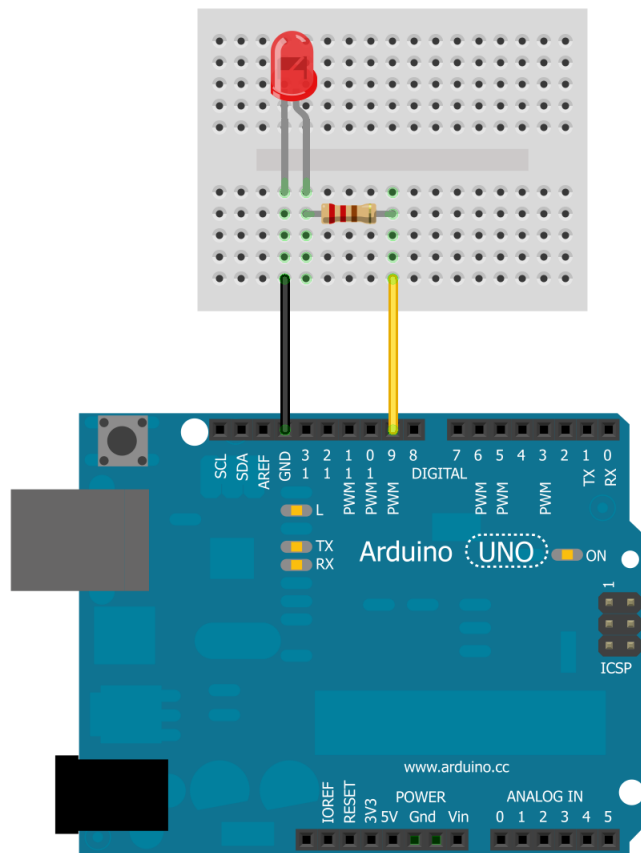
Para escribir datos en el puerto de salida digital se utiliza el bloque **“Digital Write Port”** poniendo en su entrada un valor en forma de Array de datos, mediante el control que hemos etiquetado como *“Datos a escribir”* que saca por su salida.



Finalmente se muestra el esquema funcional completo



Este es el montaje para poder probar el gobierno de salidas. Bastara con cambiar el hilo de la salida para comprobar que funciona.



7.12. Control de Servos

En este ejemplo vamos a realizar el control de dos servos en uno lo haremos modificando la velocidad de giro y en otro el ángulo girado



En la imagen vemos como quedara el panel de control. Se podrá seleccionar el número de servos a controlar, cerrar el canal de comunicación y luego controlar cada uno de los servos.

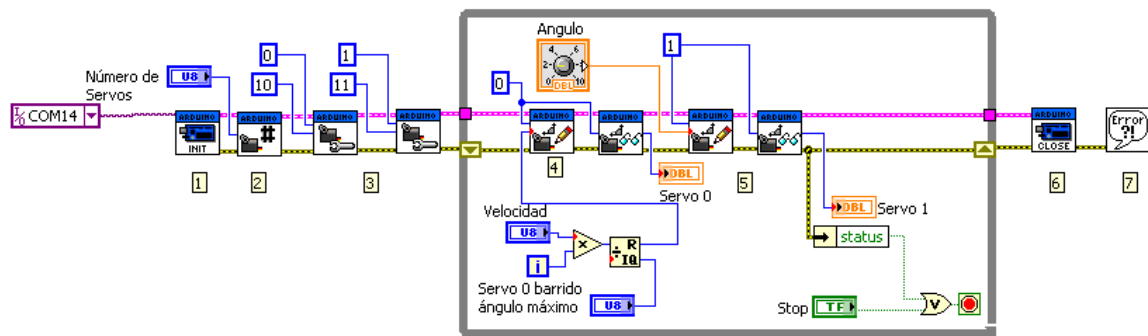
Servo 0: Se controlara la velocidad y el anulo máximo girado

Servo 1: Se controlará el ángulo girado.

Para el desarrollo del diagrama funcional tenemos que contar con cuatro nuevos tipos de bloques de función de la librería Arduino.

1. Inicializar la conexión a la placa Arduino. Si no se conecta a la entrada de recursos VISA del bloque “Init” intentará auto conectarse a la placa Arduino. Para acelerar la conexión o para conectarse de forma inalámbrica utilizar un recurso VISA constante para especificar el puerto COM a utilizar.
2. Establecer el número de servos a utilizar. Esto crea una matriz de cero índice de los servos en el Arduino. Después de especificar el número de servos que desea utilizar se puede hacer referencia a ellos por el número (comenzando con 0).
3. Configuración de los dos servos digitales asignándoles pines I / O (10, 11)

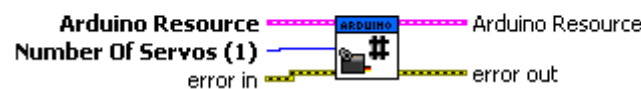
4. Escribir un ángulo de 0 servo basado en la repetición del bucle. Servo 0 barre desde los 0 grados en el ángulo de usuario y la repetición. Este ángulo también se lee desde el servo y se muestra en el panel frontal.
5. Ajuste manualmente el ángulo del servo 2. Este ángulo también se lee desde el servo y se muestra en el panel frontal.
6. Cierra la conexión con el Arduino
7. Controlar los errores.



1. Inicializar la conexión a la placa Arduino. Si no se conecta a la entrada de recursos VISA del Init vi intentará auto conectarse a la placa Arduino. Para acelerar la conexión o para conectarse de forma inalámbrica utilizar un recurso VISA constante para especificar el puerto COM a utilizar.
2. Establecer el número de servos a utilizar. Esto crea una matriz de cero índice de los servos en el Arduino. Después de especificar el número de servos que desea a utilizar se puede hacer referencia a ellos por el número (comenzando con 0).
3. Configuración de los dos servos digitales asignándoles pines I / O (10, 11)
4. Escribir un ángulo de 0 servo basado en la repetición del bucle. Servo 0 barre desde los 0 grados en el ángulo de usuario y la repetición. Este ángulo también se lee desde el servo y se muestra en el panel frontal.
5. Ajuste manualmente el ángulo del servo 2. Este ángulo también se lee desde el servo y se muestra en el panel frontal.
6. Cierra la conexión con el Arduino
7. Controlar los errores.

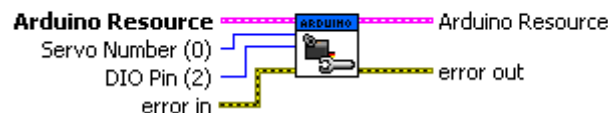
Para empezar, una vez que hemos colocado el bloque de función “Init” se procede a configurar los servos. Primero debemos decir cuántos son los servos que se van a tener en cuenta: **“Set Numbers of servos”** se llevará a la entrada un valor elemento de control que genere un numero de tipo Integer se hace pulsado el botón derecho sobre la entrada y seleccionando *Create-Control* en el menú contextual que aparece. Así creamos la caja de selección **“Numero de servos”**

Set Number of Servos
[LabVIEW Interface for Arduino.lvlib:Set Number of Servos.vi]



A continuación pondremos dos bloques del tipo **“Configure Servo”** uno para cada uno de los dos servos que vamos a gobernar. Este objeto debe parametrizarse con el numero de servo y el pin de salida que utilizaremos para llevar al servo: *Servo Number* y *DIO Pin*

Configure Servo
[LabVIEW Interface for Arduino.lvlib:Configure Servo.vi]

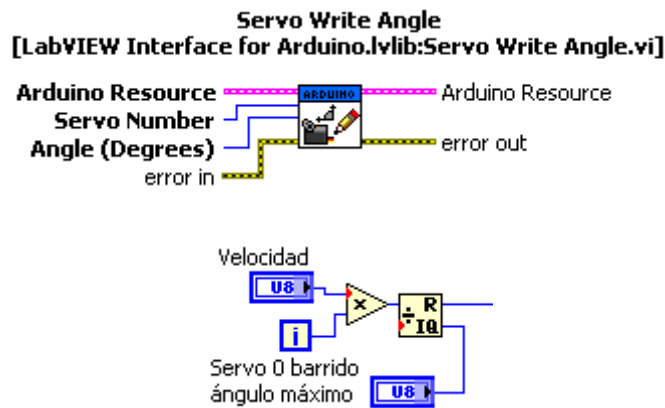


Pondremos:

Servo 0 en el PIN 1
Servo 1 en el PIN 11

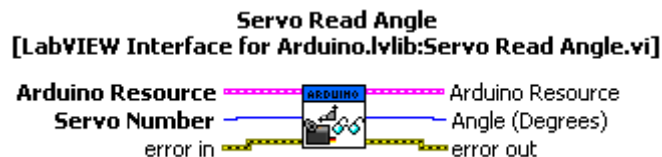
Una vez dentro del bloque de control controlaremos los servos de la manera siguiente:

En primer lugar escribiremos en el Servo 0 mediante el bloque de función “**Servo Write Angle**” mediante un control el “Angulo” a girar en grados



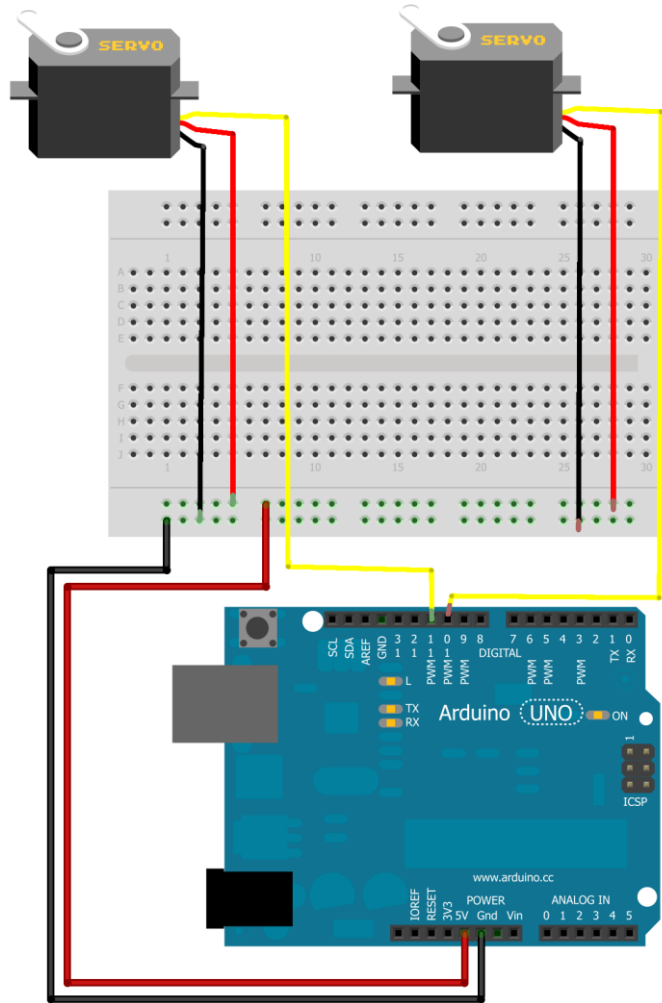
La señal creada se lleva a la entrada “Angle” del bloque

El siguiente bloque “**Servo Read Angle**” sirve para leer la posición del Servo devolviendo el valor en “Angle (Degrees)” que se lleva a un indicador analógico “*Servo 0*”



El Servo 1 se gobernará de la misma forma que el anterior con un bloque de tipo “**Servo Write Angle**” cuya entrada conectamos a un control de tipo analógico al que hemos denominado “**Angulo**”

Finalmente se colocará un bloque “**Servo Read Angle**” que nos indicara la posición del servo y la mostraremos con un instrumento analógico denominado “*Servo 1*”



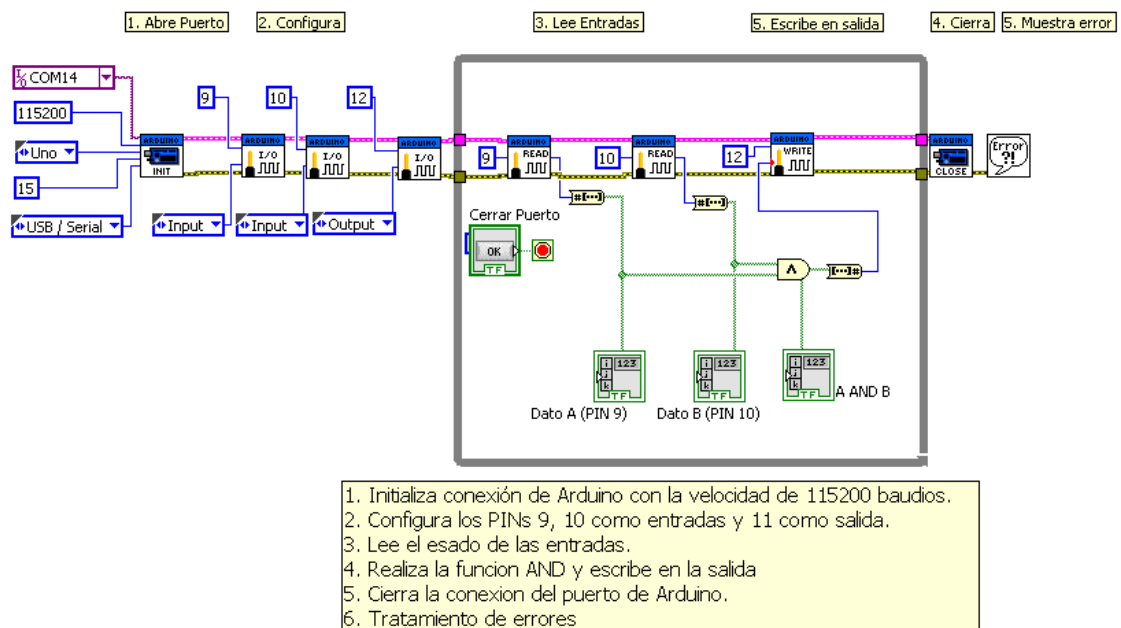
7.13. Función AND

El ejemplo que se muestra a continuación implementa una función AND entre dos entradas **PIN 9** y **PIN 10** cuya salida gobierna el **PIN 12**



En la figura se muestra el aspecto del Panel.

La implementación como siempre comienza por el bloque “**Init**” seguido de los bloques de configuración de PIN “**Set Digital PIN Mode**”



Dentro del bucle de ejecución se deben colocar dos bloques de lectura de PIN “**Digital Read Pin**” y uno de escritura “**Digital Write Pin**”. La señal que gobierna este último bloque se obtiene de realizar la función AND de las salidas obtenidas de las lecturas del **PIN 9** y **PIN 10**.

Los elementos de visualización de estado se han colocado en las salidas de los tres bloques. No olvidemos que las señales deben convertirse de Array 1D a Array booleano mediante el bloque de función:

Para convertir

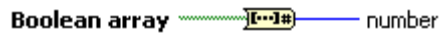
Number To Boolean Array

De Array tipo Número a Array Boolean



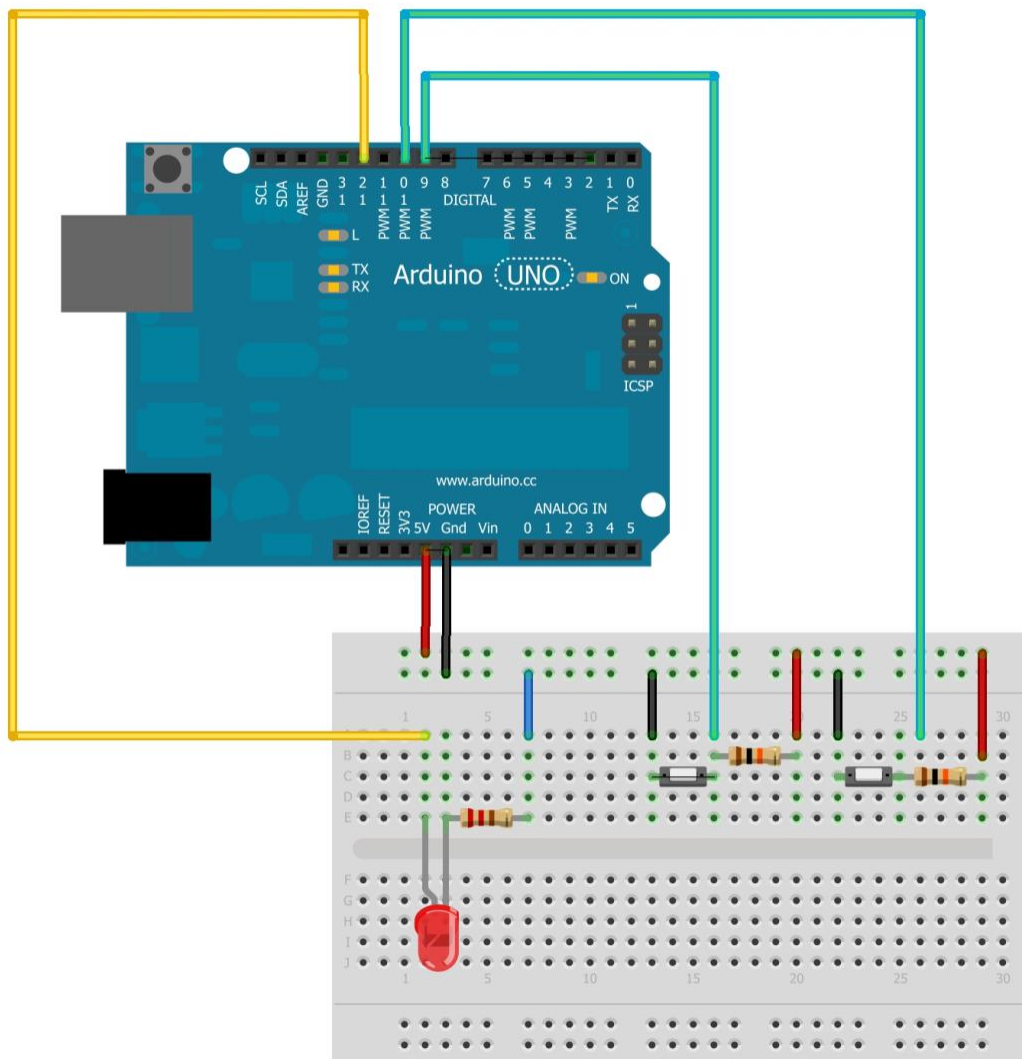
Boolean Array To Number

De Array Boolean a Array Número



Finalmente se deben colocar el bloque de “Close” y el de “Simple Error Handler”.

El esquema de montaje para realizar las pruebas es el siguiente

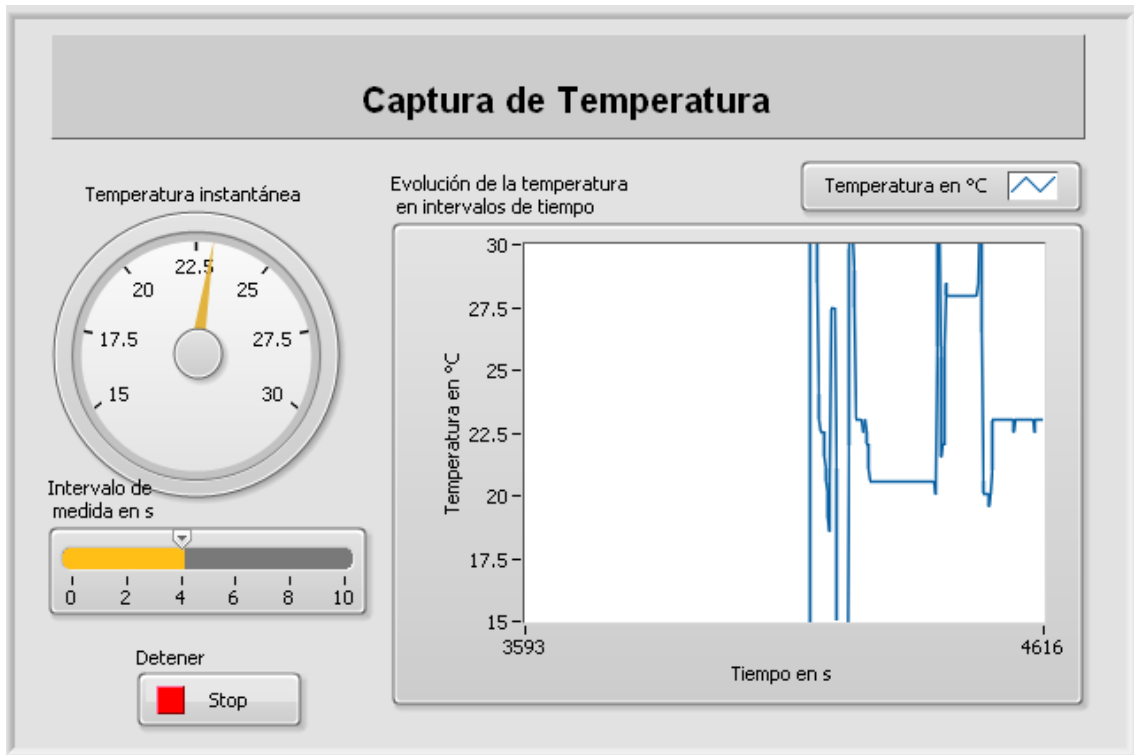


7.14. Temperatura 1

(traducido y adaptado del original “Electronique Innovate”)

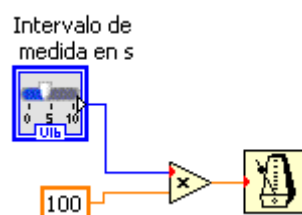
<http://innovelectronique.fr/2012/05/04/arduino-et-lifa-labview-interface-for-arduino/>

Este ejemplo esta sacado de la pagina Web que se indica anteriormente y básicamente sirve para demostrar cómo es posible representar gráficamente el valor de un canal analógico en un grafico. En este caso se fija un tiempo de muestreo de la señal con el fin de realizar la lectura del modo que hacen los sistemas de adquisición de datos reales.



En la figura anterior se muestra el aspecto del Panel en el que se hace uso de un indicador analógico “**Knob**” etiquetado como “*Temperatura instantánea*” de tipo circular y un trazador grafico tipo “**Chart**” etiquetado como “*Evolución de la temperatura en intervalos de tiempos*”. Se puede observar también un slider para ajustar el tiempo de muestreo. Finalmente figura el “**Botón**” de parada etiquetado como “*Detener*”

Para realizar la toma de muestras se ha recurrido a un generador de intervalos que



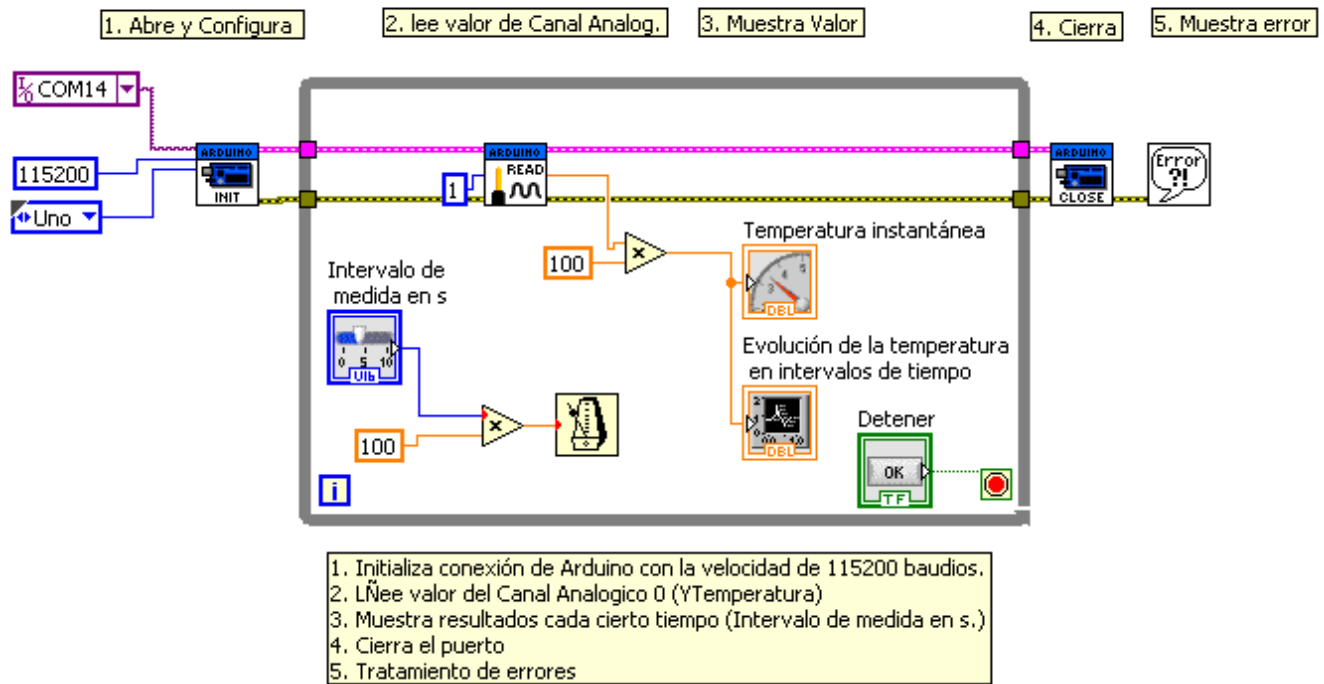
Wait Until Next ms Multiple

millisecond multiple

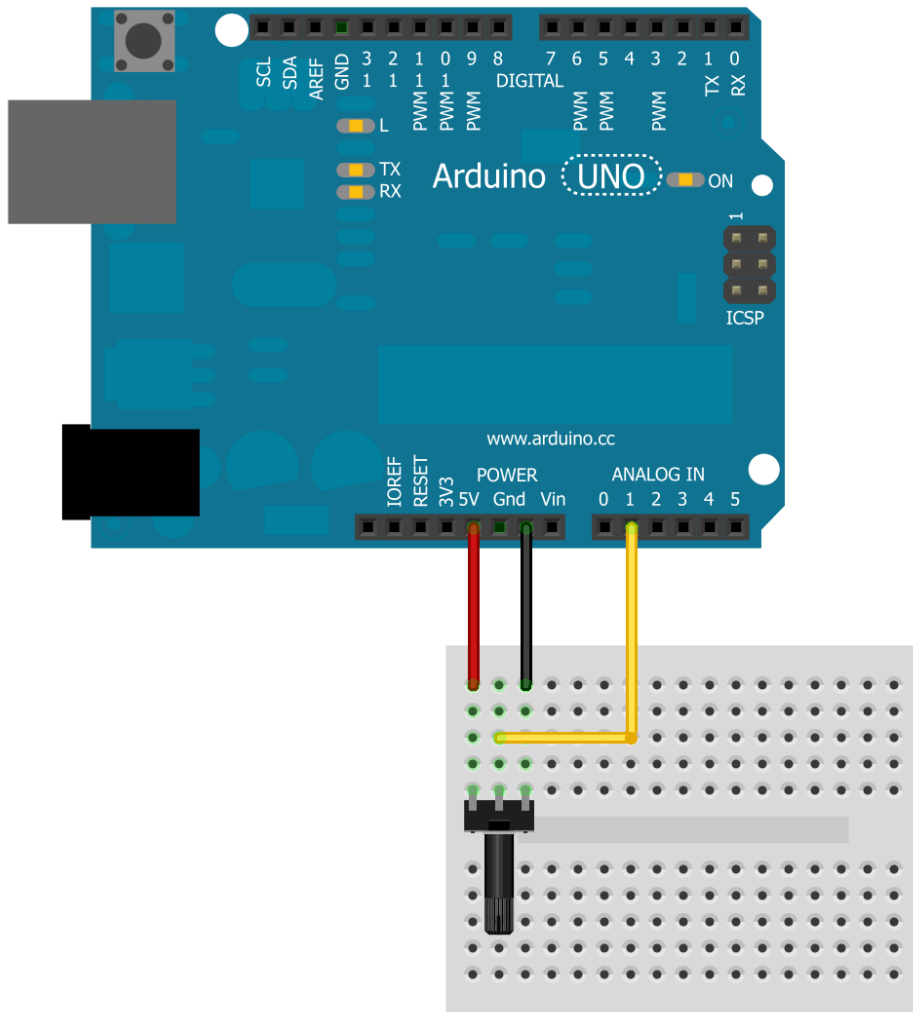


ejecuta cíclicamente el contenido del bucle esta función se alimenta con una entrada de valor proporcionada por un “Slide” etiquetado como “Intervalo de medida en s”.

Los valores leídos del canal de entrada analógico “Temperatura instantánea” se muestran en un medidor de aguja a la vez que en el registrador gráfico



La figura anterior muestra el esquema funcional completo y la siguiente el esquema de montaje para las pruebas.

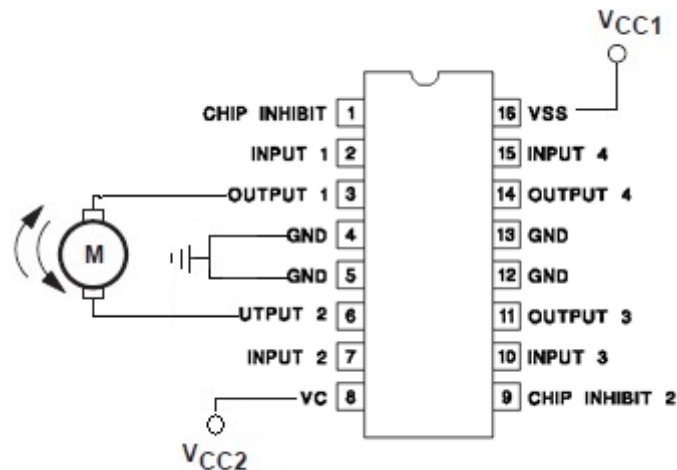


7.15. Control Motor de cc. Velocidad y Sentido

(Traducido y adaptado del original “Electronique Innovate”)

<http://innovelectronique.fr/2012/05/04/arduino-et-lifa-labview-interface-for-arduino/>

El siguiente ejemplo está sacado de la página Web que se anota al comienzo de la página y es un valioso ejemplo de cómo puede controlarse un pequeño motor de cc. Haciendo uso del puente de potencia integrado en el CI L293D que puede controlar dos motores de corriente continua: el L293D (ficha técnica [aquí](#)).



L293D

Tabla de funcionamiento del Motor 1

CHIP INHIBIT 1	INPUT 1	INPUT 2	FUNCION
H	L	H	Gira a la Derecha
H	H	L	Gira a la Izquierda
H	L	L	Para rápida del motor
H	H	H	Para rápida del motor
L	X	X	Para rápida del motor

Este circuito es relativamente fácil de implementar y lo haremos con un solo motor que simplifica aún más el conjunto. Téngase en cuenta que se trata de drivers para el gobierno de dos motores de corriente continua (<http://www.lextronic.fr/P5073-platine-de-commande-de-moteurs-dc.html>).

Descripción de pines para el control del Motor 1:

El Pin 1 (CHIP INHIBIT 1) sirve para activar el Motor 1. Si este pin está conectado a una salida de Arduino del tipo PWM, se puede variar la velocidad del motor haciendo variar el valor de salida de este PIN.

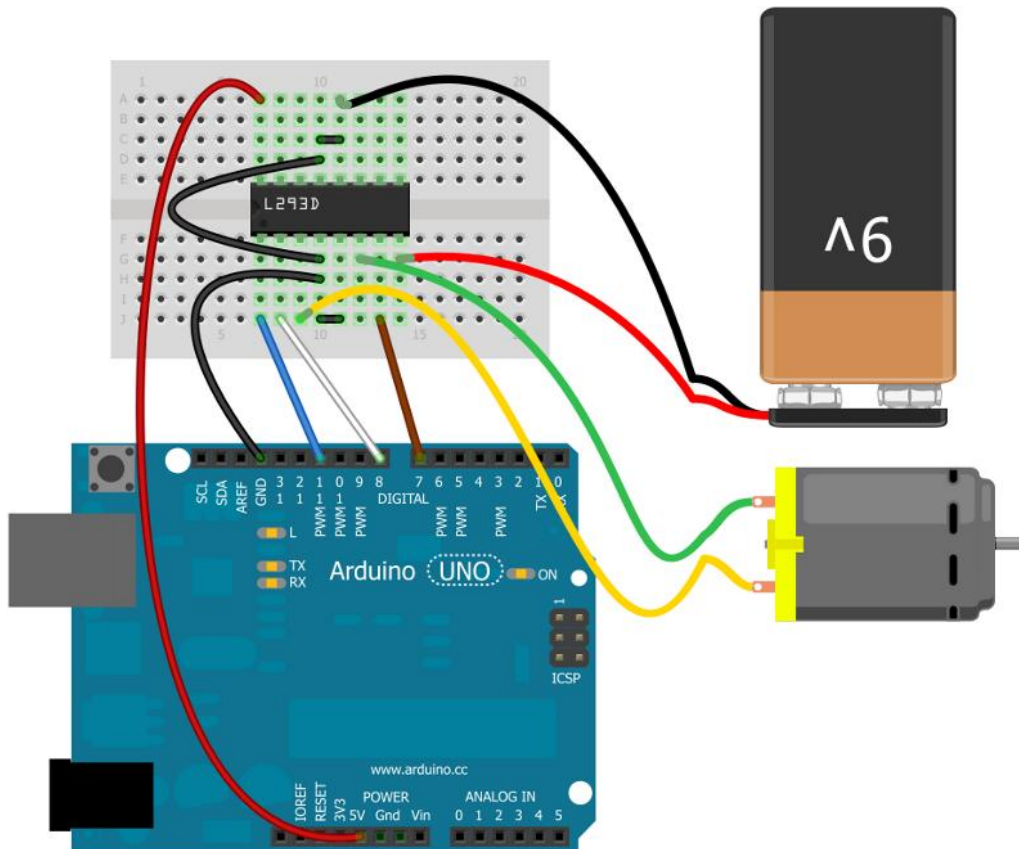
Los Pines 2 (INPUT 1) y 7 (INPUT 2) permiten fijar el sentido de giro del motor o la parada. Los pines 3 (OUT 1) y 6 (OUT 2) son los pines de salida de potencia del motor.

Pin 16 (VSS) recibe una alimentación de 5V de la placa Arduino.

El pin 8 (Vcc2) está conectado a una pila de 9 V para asegurar el suministro de potencia del motor.

Para el MOTOR 2 se utilizan los pines del otro lado del chip (9 al 16)

Finalmente, los pines 4,5,12, 13 están conectados a tierra (GND del Arduino y el polo negativo de la batería). Se obtiene el diagrama de cableado siguiente:

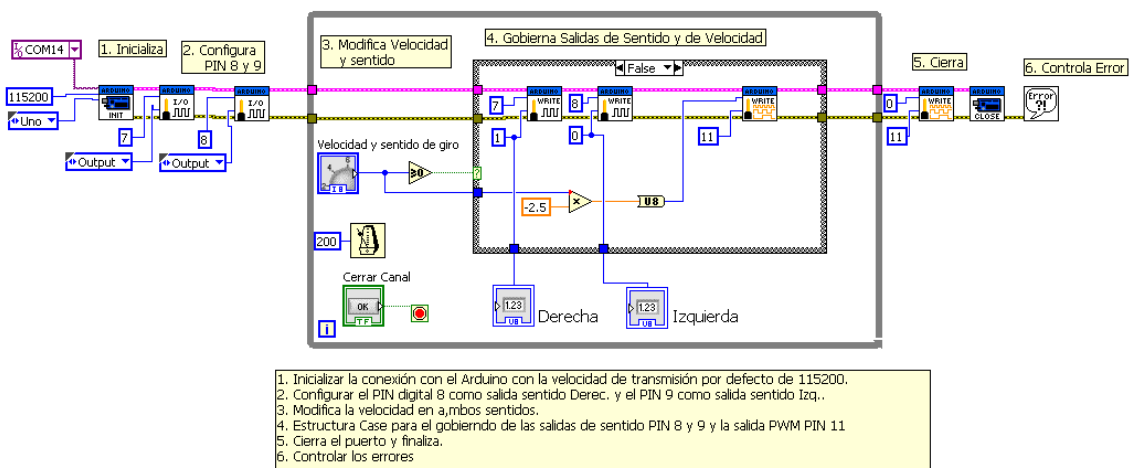


Los pines 7 y 8 de la tarjeta Arduino son salidas digitales para controlar la dirección del motor, el pin 11 el modo PWM. La dirección de rotación sigue la siguiente tabla:

La parte frontal de este ejemplo es muy simple (un mando único que evoluciona desde -100% a 100%, el símbolo utilizado para fijar la dirección de rotación):



El programa de LabVIEW se pueden presentar de la siguiente manera (la única parte que varía de uno a captar el otro es el interior de la prueba de la caja: Verdadero y Falso):



En la figura anterior se muestra el esquema funcional de trabajo.

Para empezar se inicializa la conexión con Arduino. Seguidamente se configura el PIN digital 8 como salida sentido Derecha. y el PIN 9 como salida sentido Izquierda.

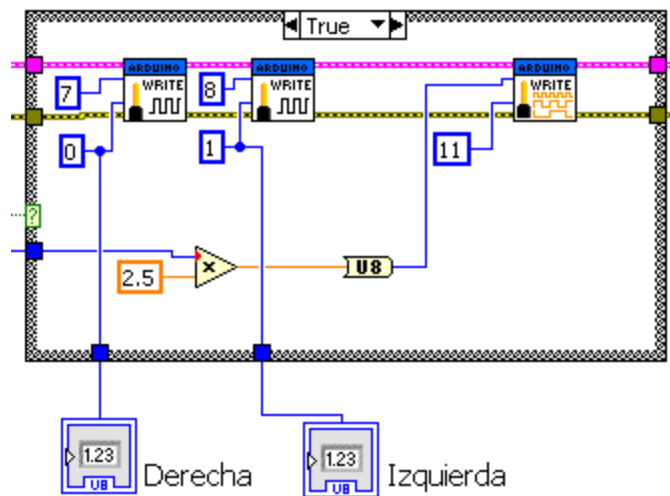
Dentro del bucle se ha colocado una estructura tipo “**Case Estructura**” que se encargara del gobierno del motor. La ejecución del bucle se realiza en intervalos de 200

ms.

Los casos a tener en cuenta en esta estructura son dos, que se corresponden con los dos posibles sentidos de giro del motor.

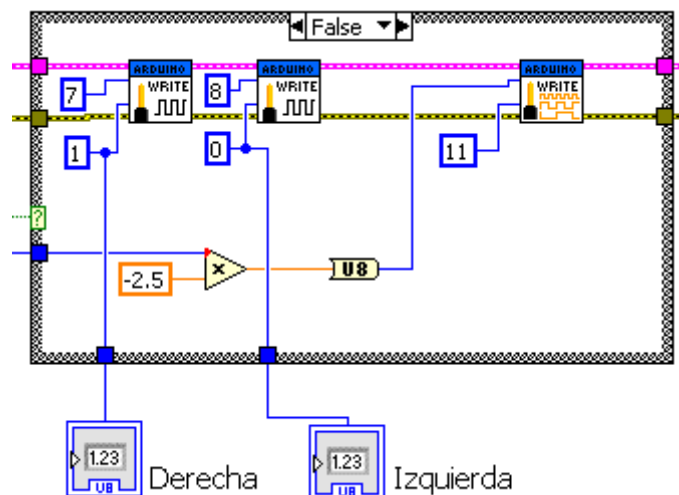
En el primer caso “*True*” el sentido de giro es a la Izquierda por lo que debemos sacar los valores correspondientes en las salidas **PIN 7** y **PIN 8** de Arduino. La velocidad se recoge del control tipo numérico de aspecto circular que hemos etiquetado como “*Velocidad y sentido de giro*”

PIN Arduino	Valor	PIN L293D INPUT 1	PIN L293D INPUT 1	Giro
PIN 7	0	0	1	Izquierda
PIN 8	1			



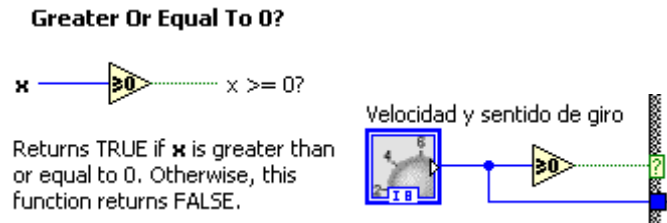
En el segundo caso “*False*” el sentido de giro es a la Derecha por lo que debemos sacar

PIN Arduino	Valor	PIN L293D INPUT 1	PIN L293D INPUT 1	Giro
PIN 7	1	1	0	Derecha
PIN 8	0			



Se han colocado dos indicadores de giro “Derecha” e “Izquierda” para saber en que sentido esta girando el motor.

La conmutación de la estructura “Case Structure” se realiza mediante un operador del tipo “Greater or Equal To 0”



La velocidad se genera en una escala de -100 a 100 por lo que se debe multiplicar por -2.5 para alcanzar los 255 que es el valor máximo que se puede sacar en una salida PWM equivalente a 5v. en el PIN 11

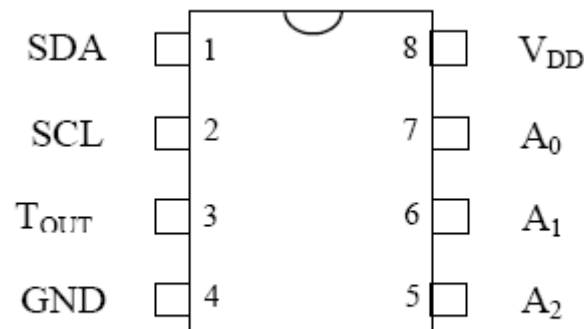
Finalmente en la salida del bucle se detiene el motor enviando un 0 a la salida PWM PIN 11 se cierra el puerto y se tratan los errores.

7.16. Medida de Temperatura mediante el Bus I2C

(traducido del original “Electronique Innovate”)

<http://innovelectronique.fr/2012/05/04/arduino-et-lifa-labview-interface-for-arduino/>

El sensor de temperatura DS1621 es un componente relativamente común ([aquí](#) esta su ficha técnica). Sin detallar demasiado, sólo tendremos que especificar los comandos a enviar al DS1621 para iniciar y hacer una simple lectura de la temperatura. Los pines 1 y dos son las señales SDA y SCL del bus I2C. El Pin 3 (A) no será utilizado en nuestra aplicación. Los pines 7 (A0), 6 (A1) y 5 (A0) se conectan a tierra para establecer la dirección de este sensor I2C. El pin 4 es de tierra y el pin 8 + Vcc (5V aquí).



DS1621 8-PIN DIP (300mil)

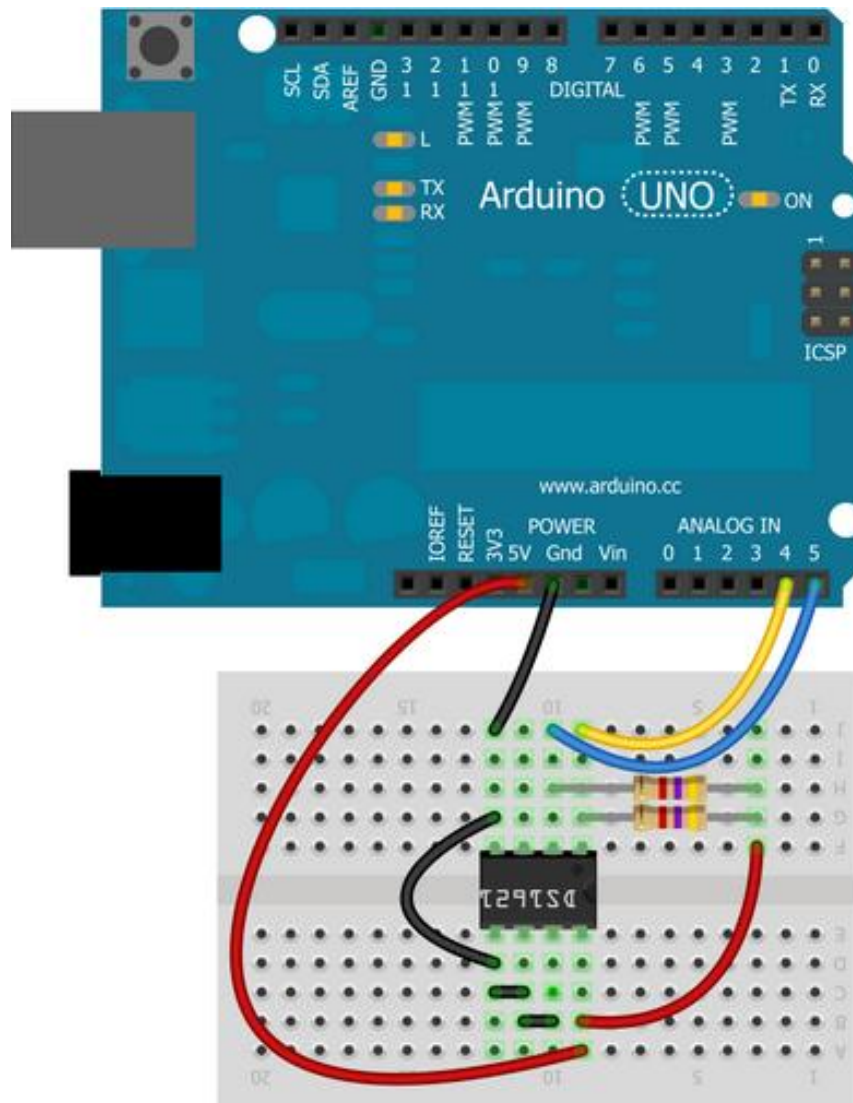
La secuencia de inicialización para integrar este elemento en el bus I2C es la que sigue (contamos con que los pines A0, A1 y A2 se conectan a masa GND):

1. Condiciones de inicio (start)
2. Escriba 0×90 : se selecciona la casilla $A2A1A0 = 000$ escribir
3. Escribir $0xac$: se escribe en el registro de configuración
4. Escriba 0×00 : conversión de la temperatura continua
5. Condición de parada (stop)
6. Esperar 20 ms: escribir eeprom de la configuración anterior
7. Condiciones de inicio (start)
8. Escriba 0×90 : se selecciona la casilla $A2A1A0 = 000$ escribir
9. Escribir $0xEE$ son: lanzamiento de la conversión a la temperatura constante
10. Condición de parada (stop)

La siguiente secuencia se usa para leer la temperatura:

1. Condiciones de inicio (start)
2. Escriba 0×90 : se selecciona la casilla $A2A1A0 = 000$ escribir
3. Escribir $0xAA$: solicitud de lectura de la última temperatura la muestra
4. Condiciones de inicio (reinicio)
5. Escriba 0×91 : se selecciona la casilla $A2A1A0 = 000$ **read-**
6. Leer todos los 8 bits de la temperatura
7. Leer todos los 8 bits de la configuración de la temperatura y un NACK!
8. Condición de parada (stop)

Te das cuenta que no es fácil, pero vamos a tratar de poner en práctica todas las funciones de LabVIEW con Arduino. Dependiendo de la versión de la tarjeta Arduino UNO tiene el SCL y SDA pines no son el mismo lugar (ONU Rev2: A4 = SDA, SCL = A5; UNO Rev3: dos pines en un dedicado SCL y SDA). Al escribir este tutorial tengo una placa Arduino rev2 UNO, el siguiente diagrama de cableado es adecuado para esta plataforma:

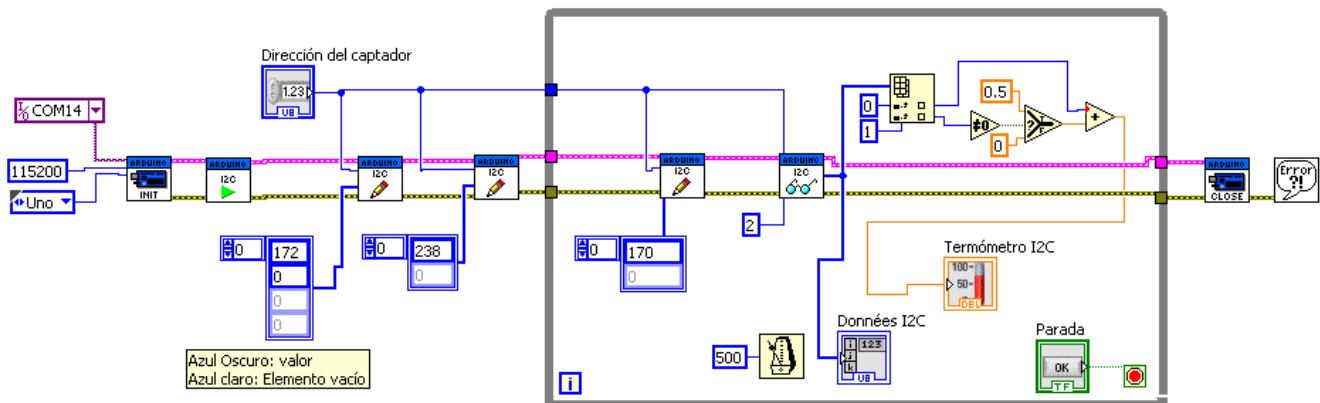


El frontal de LabVIEW se produce como sigue:

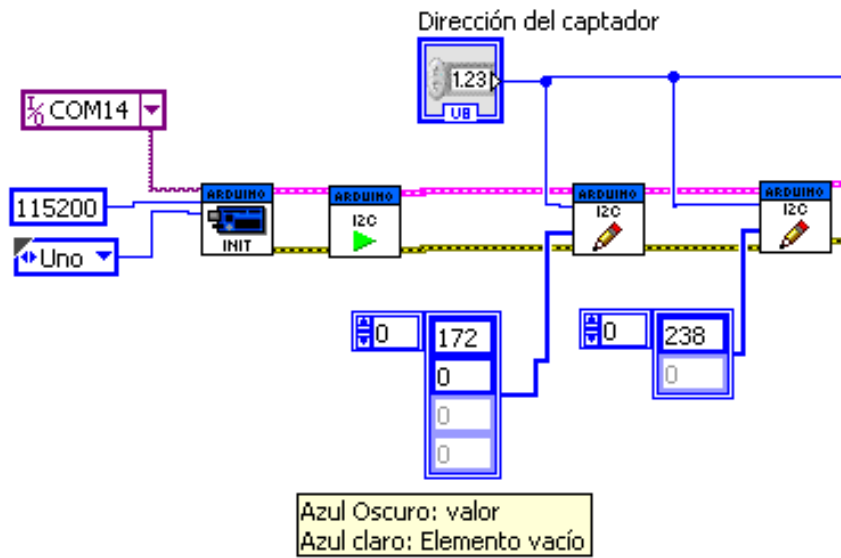


Nota: Un indicador que se utiliza para seleccionar la dirección del sensor DS1621 en el bus I2C. Bajo el título "Datos I2C" es en realidad una forma de tabla de indicadores, se puede ver los datos leídos del bus I2C.

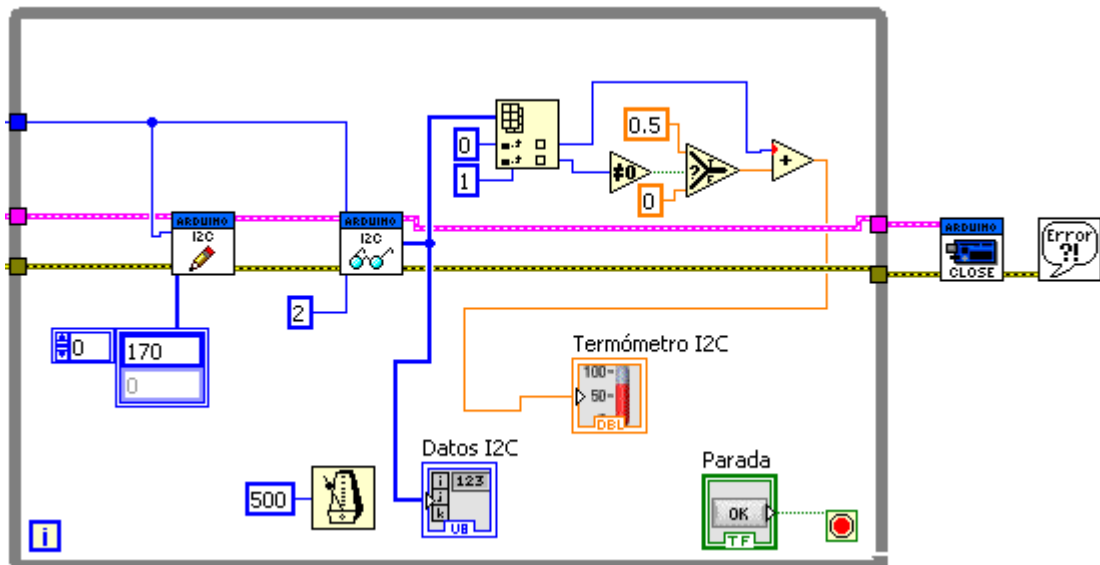
El diagrama se puede presentar como sigue:



En primer lugar para entender cómo trabaja el VI sub I2C miré a las señales procedentes de la Arduino I2C en un osciloscopio y debo decir que me ayudó. La lectura del diagrama de izquierda a derecha:



1. La inicialización del bus I2C (TWI en el arduino terminología)
2. La inicialización del DS1621. Primera dirección del componente. La dirección es de 7 bits (MSB primero), los primeros 4 son fijos (ver la documentación en el 1001), el último 3 están fijados con los pines A2A1A0. En nuestro caso $A2A1A0 = 000$ por lo que la dirección es $0b1001000$ componente en el sistema binario o hexadecimal 0×48 o 72 en decimal. Así que tienes la explicación de los 72 valores en la parte delantera. Entonces usted tiene que escribir los valores $0xac$ (172) y 0×00 (0): el papel de la primera sub VI "Write I2C".
3. Después de la inicialización. Normalmente se tarda alrededor de diez intervalos de tiempo, con la transmisión de 15 bytes hacia LabVIEW para el Arduino lleva su tiempo, podemos considerar que la operación se lleva a cabo. Se escribe el valor de $0xEE$ son (238). En este punto el DS1621 se inicializa correctamente.



4. A continuación, entra en un bucle infinito. La temperatura será solicitada cada 500 ms. Para aplicar la temperatura final, escribir el valor 0xAA (170).
5. A continuación, lea el resultado: aquí el DS1621 tiene que enviarnos dos bytes, donde el valor 2 en la sub VI "Lee I2C". Este sub VI también es responsable de hacer el NACK final (verificado con un osciloscopio). Esto produce una matriz 1D con dos cajas que contienen de 8-bits sin signo. Estos datos se muestran como los datos en bruto en el indicador en el frente "de datos I2C".
6. Debemos tratar a los dos valores proporcionados por el DS1621. Empezamos por la extracción de los datos de la Tabla VI, en el marco del "índice de matriz". El índice de '0': 8 MSB del resultado de la temperatura y el índice de tipo entero firmado '1': 8 LSB de la conversión: en nuestro caso, este byte tomar dos valores: o bien 128 para indicar es necesario añadir 0,5 ° C o 0.
7. A continuación se recogen las señales y la temperatura real se muestra en el termómetro.
8. Esto funciona, pero tenga en cuenta que no se ocupa de temperaturas negativas. Ese es un ejercicio que se queda sin hacer. Inténtelo usted.

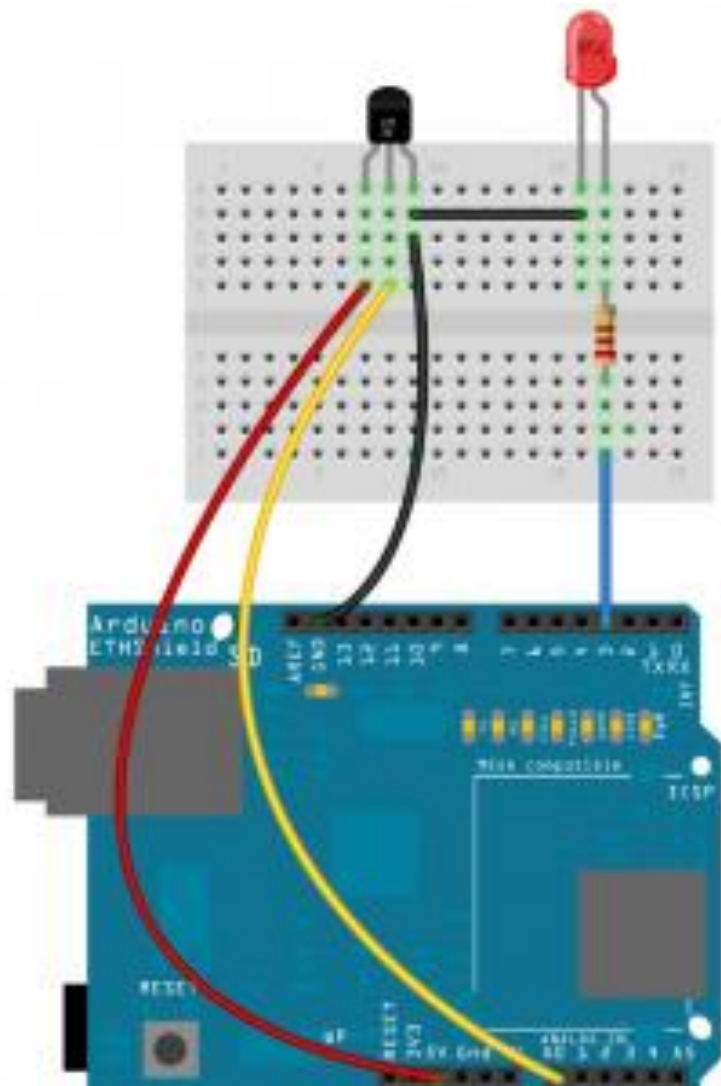
7.17. Diálogo con Arduino Ethernet

(traducido del original “Electronique Innovate”)

<http://innovelectronique.fr/2012/05/23/arduino-et-lifa-episode-2/>

No se trata aquí de trabajar con LIFA sino directamente con el shield Arduino Ethernet. Nos proponemos realizar un ejemplo que permitirá controlar la temperatura del LED y recuperar todo por ethernet (incluso a través de Internet).

Usamos un sensor de temperatura LM35 conectado a la entrada A0 y una Salida de LED conectada al PIN 3. Como LM35 tiene una pendiente de $10 \text{ mV} / ^\circ \text{C}$ y 0 V a $0 ^\circ \text{C}$, la variación de temperatura en un cuarto de grado dará lugar a una variación de unas pocas decenas de mV. Por lo tanto, aumentará la precisión de la medida analógica si utilizamos una referencia de tensión de $1,1 \text{ V}$ el Arduino. El convertidor analógico a digital es de 10 bits, se obtiene así un escalón mínimo de $1,1 \text{ V} / 1.024 = 1 \text{ mV}$ ($1,0742 \text{ mV}$ con precisión). Por lo tanto, se puede medir la temperatura de $0 ^\circ \text{C}$ a $110 ^\circ \text{C}$ que es suficiente. El cableado está presente como sigue:



El código de Arduino se muestra a continuación. Se basa enteramente en el "Servidor Web" ejemplo, es entonces suficiente para simplificar. El código está comentado así que lea con cuidado!

```

/*
  Serveur TCP
  O. DARTOIS, le 23/05/12
  Source directement basée sur l'exemple Web Server
*/

#include <SPI.h>
#include <Ethernet.h>

// Adresses MAC et IP à changer suivant vos besoins
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1, 177);

// Initialisation de la librairie ethernet
// et création d'un objet "Server" qui va accepter
// les connexions extérieures à l'adresse IP défini plus
// haut et sur le port indiqué (ici 8000)
EthernetServer server(8000);

void setup()
{
  Ethernet.begin(mac, ip); // Initialisation de la pile TCP/IP
  server.begin(); // Démarrage du serveur
  pinMode(3,OUTPUT); // Broche 2 en sortie (DEL)
  digitalWrite(3,LOW); // Broche 2 à 0 => DEL éteinte
  analogReference(INTERNAL); // Référence du CAN à 1,1V
}

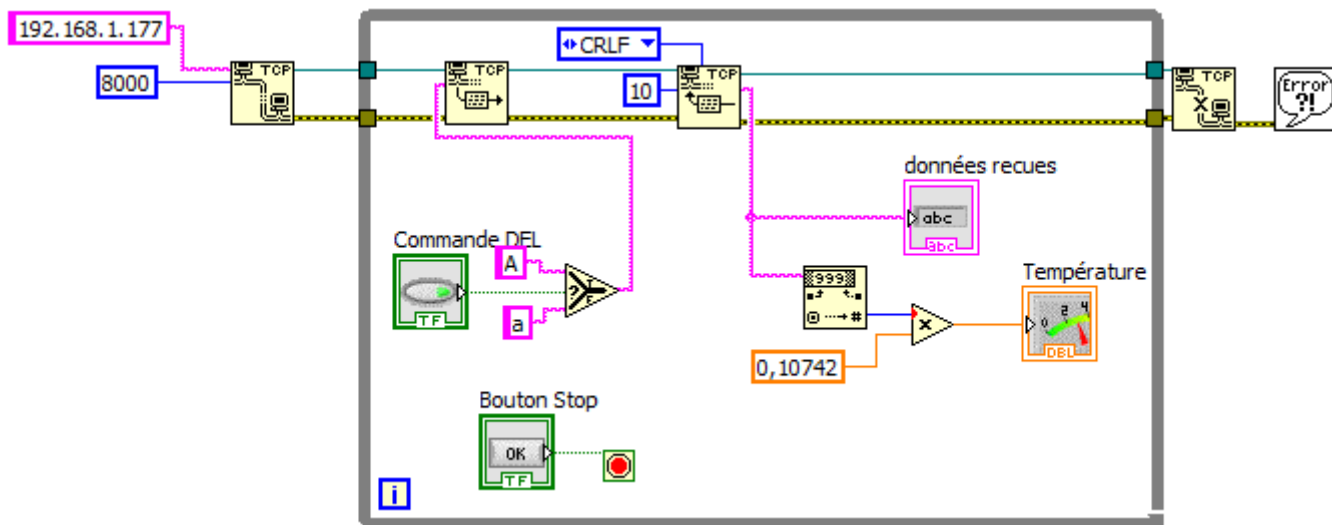
void loop()
{
  // Attente des connexions et création d'un objet client s'il y a lieu
  EthernetClient client = server.available();
  if (client) { // Un client existe
    while (client.connected()) { // il est connecté
      if (client.available()) { // et il a envoyé des caractères
        char c = client.read(); // on lit le caractère reçu
        if (c == 'A') digitalWrite(3,HIGH); // si c'est 'A' on allume la DEL
        if (c == 'a') digitalWrite(3,LOW); // si c'est 'a' on éteint la DEL
      }
      // on renvoie au client une chaîne de caractère qui représente le résultat de
      // la conversion AN (par ex: '236') suivi des caractères CR+LF
      server.println(analogRead(0));
      delay(100); // On ralentit un peu le flot de données
    }
    client.stop(); // Fermeture de la connexion
  }
}

```

Una vez que el código compilado e instalado en el arduino con su ethershield, vamos a hacer un panel frontal de LabVIEW para controlar el conjunto.



El diagrama de bloques de LabVIEW se presenta como sigue:



Como se puede ver, el esquema es muy simple ... es especializado en el LabVIEW VI que lleguemos a ese nivel de simplicidad. Una pequeña explicación de todos modos: el lado del servidor arduino implementa una pila TCP / IP pero es bastante fácil hablar en una red ethernet. Que utiliza las direcciones IP y la versión 4 de IP que se puede hacer de enrutamiento, lo que significa que usted puede poner una caja de montaje detrás de su casa y acceder a ellos desde cualquier lugar en Internet. El protocolo de transporte TCP se utiliza aquí, es un protocolo de transporte de datos en modo conectado y confiable. Esto normalmente significa que sus datos siempre llegarán con seguridad! La pérdida de paquetes será administrado por TCP y no por usted. Como puede verse, se utiliza TCP como el protocolo de transporte y para hacer el intercambio de datos de trabajo con "conexiones" de redes. Todo esto va a ser tapado por el VI disponibles en la paleta "Comunicaciones de datos" y "Protocolos" y "TCP". Para interactuar con el arduino, debe seguir los siguientes pasos:

1. VI puso una "conexión TCP Abrir" y luego configurarlo. En nuestro caso, la dirección IP del arduino es 192.168.1.177 y el puerto de escucha es el 8000. Para ello será necesario que el equipo está en la misma red que el arduino. Por ejemplo, aquí la dirección IP de su computadora portátil podría ser 192.168.1.15.
2. Una vez que la conexión está abierta, puede enviar datos con el VI "TCP escribir". Toma como entrada de caracteres o cadenas. Aquí, según el estado de la tecla "Control del LED", le enviaremos el carácter 'A' (para la luz LED, vea el código arduino) o 'a' del personaje.

3. Para recibir datos, se utiliza el VI "TCP Lea" lo llama un "buffer" bytes para almacenar los datos recibidos (en este caso he puesto 10, incluso si vamos a recibir menos caracteres). En contra de lo que es importante es el "modo" de operación de este VI. Aquí está en "CRLF": es decir, se espera que el búfer está lleno para ajustar su salida de los caracteres recibidos, cuando se recibe "CRLF", lo pone en sus caracteres de salida recibido NO CRLF. Usted puede entender mejor por qué este es el método "println" y no "impresión" que se utilizó (println añade de forma automática después de los caracteres CRLF que ha pasado). La cadena recibida se muestra como "raw" en el indicador de "recibido datos". Desea mostrar la temperatura en un metro, pero debe transformar al resultado de la conversión AD del arduino que hemos recibido un número. Es el papel de la sub VI "número decimal de la cadena" y se multiplica por el coeficiente de encontrar la temperatura (quantum = sensibilidad $1074 \cdot 10^{-3}$ y sensor de $10 \text{ mV} / ^\circ \text{C}$). Finalmente mostrar la temperatura en $^\circ \text{C}$ en el metro.
4. Cuando se detiene el bucle infinito con botón de parada correctamente debe cerrar la conexión TCP con el VI "Cerrar conexión TCP". A continuación, muestra los errores.

Pruebe todo y verá que es funcional. Modifique el diagrama de bloques y el código de Arduino para controlar el LED PWM por ejemplo ...

7.18. Lectura de una entrada analógica.

(Ejemplo traducido y adaptado del original que figura en la librería [LIFA](#) aportada por National Instruments)

En el siguiente ejemplo se trata de leer el valor de una entrada analógica de las 6 que tiene **Arduino UNO** que como sabemos sus valores oscilan entre **0 y 5 v**.

Se montara un Panel como el de la figura en el que se muestra un selector para el número de canal “*Selección de Entrada Analógica*”. Una ventana de dato numérico en el que se muestra el valor leído y un Botón con el que se detiene la aplicación y se cierra el puerto de comunicaciones



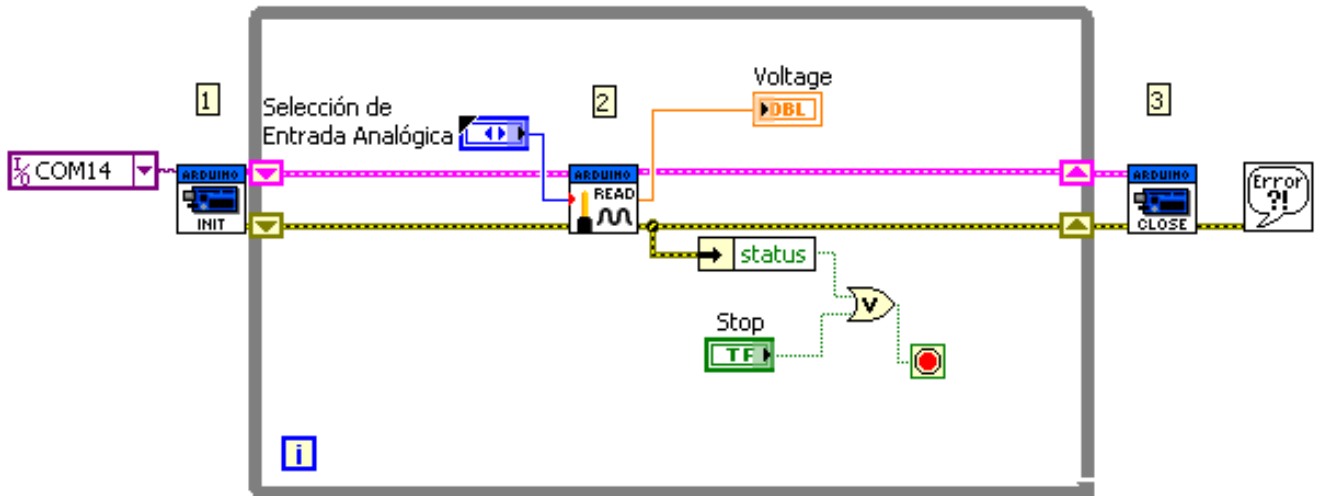
La construcción de Bloques Funcionales mostrada en la figura siguiente es muy sencilla.

Vemos que se colocan los dos bloques **Init** y **Close** que facilitan la configuración de la tarjeta y el cierre del puerto respectivamente.

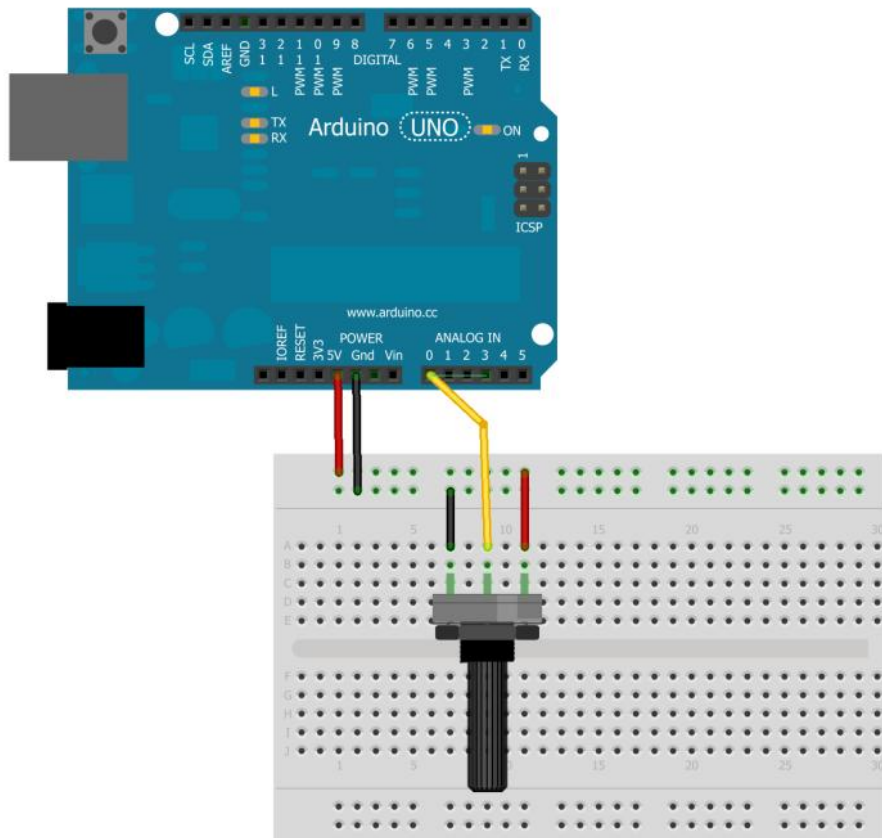
En el bucle del programa (dentro de la recinto “**While Loop**” se colocara un bloque de lectura de canal analógico “**Analog Read Pin**” que recoge el valor de la señal del pin que seleccionemos mediante el bloque de “Selección de Entrada Analógica”

La salida de este bloque (valor leído) se llevara por un lado a la caja de texto que visualiza el valor “**Voltage**” .

Para salir del bucle de ejecución “While Loop” se utiliza un botón “Stop” y a la vez también se puede salir si se produce un error mediante el bloque “Status”



1. Inicializa la conexión con Arduino con una velocidad de 115200 baudios.
2. Lee el canal especificado como entrada.
3. Cierra la conexión con Arduino



7.19. Conexión de un Módulo BlikM

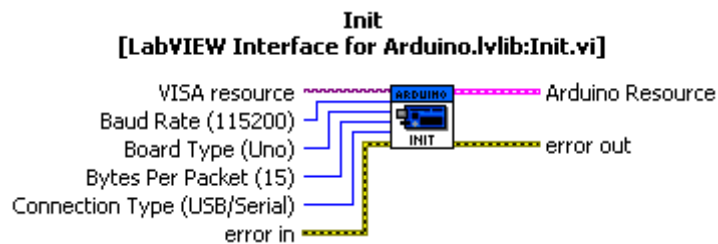
(Ejemplo traducido y adaptado del original que figura en la librería [LIFA](#) aportada por National Instruments)

En esta aplicación se trata de gobernar un módulo tipo BlinKM (tricolor) haciendo uso de la librería que LIFA incorpora para ello “Librería BlinkM” .

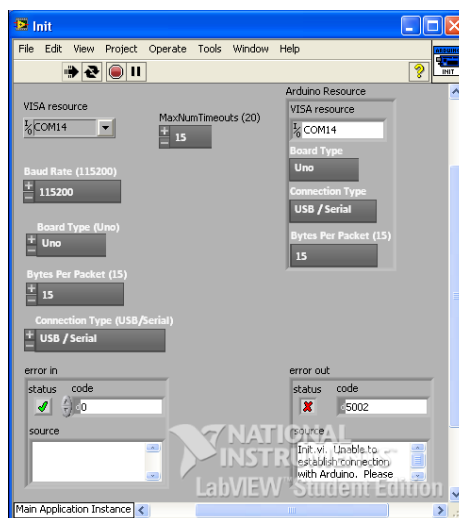


Modulo BlikM

Para empezar configuramos Arduino como siempre con el modulo “Init” en este caso no le hemos puesto los conectores de parametros porque suponemos que ya se han colocado en el SubVI “Init” pulsando dos veces sobre el bloque se muestra la ventana desde donde podemos designar estos valores. En la figura se ve.



Pulsando dos veces sobre el icono del bloque aparecerá la pantalla *vi* del objeto de a librería y vemos que se pueden modificar aquí os parametros que por defecto presenta

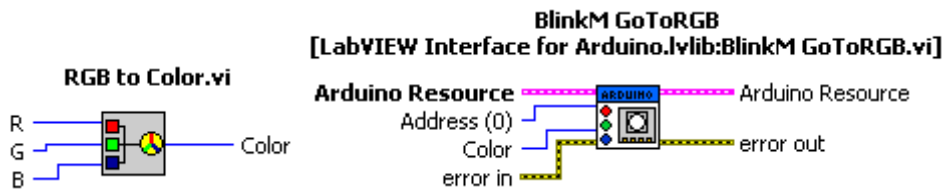


No olvidemos que el sistema de LabVIEW es capaz de reconocer lo que tiene conectado en el puerto es decir “autodetectar” la conexión, lo cual ayuda a la configuración de la aplicación.

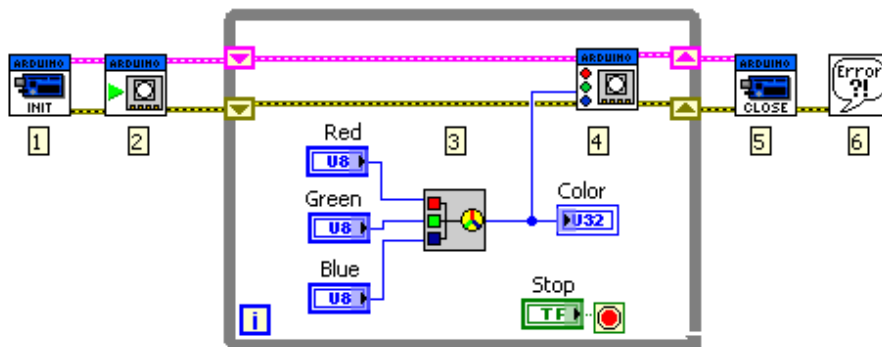
El siguiente bloque a conectar es el que permite hacer constancia de que tenemos un elemento “**BlinkM Init**” que como sabemos se comunicara con Arduino a través del protocolo **I2C** uniéndose al bus como master para controlar el componente **BlinkM**



Una vez dentro del bucle lo que debemos hacer es crear la señal “Color” bloque “**RGB To Color**” con la que alimentaremos el bloque “**BlinkM GoToRGB**”



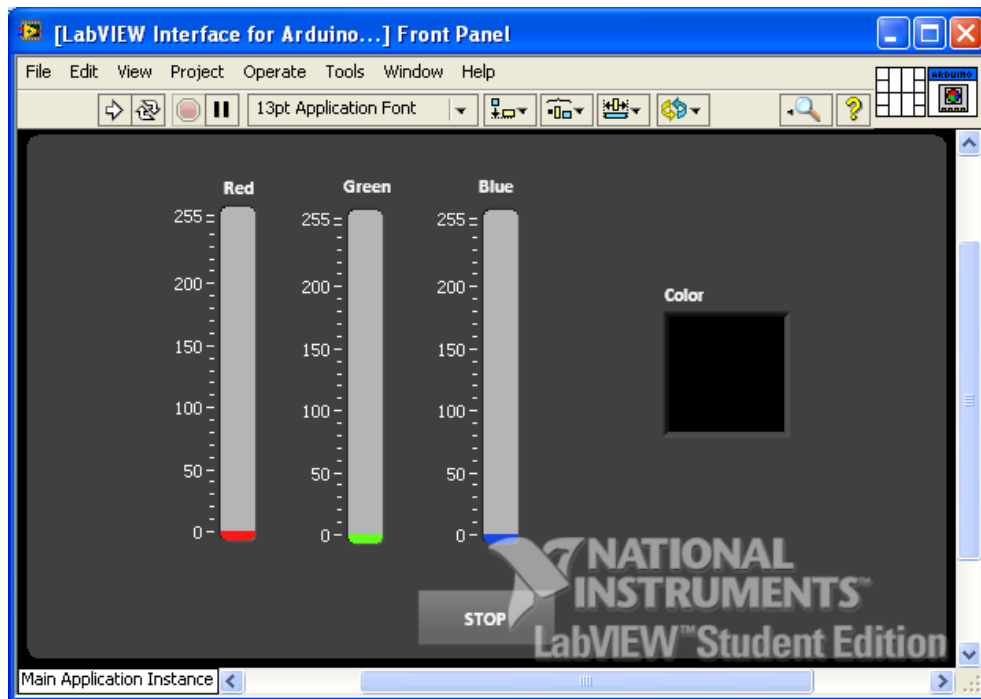
En la siguiente figura vemos el esquema completo que lo cierran, como siempre los bloques “**Close**” y “**Error**”



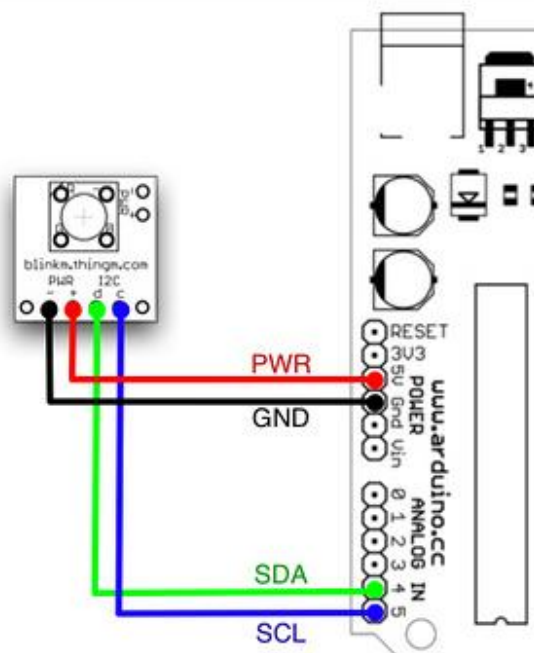
1. Inicializar conexión a la placa Arduino con la velocidad en baudios por defecto de 115200.
2. Inicialice el BlinkM haciendo que el Arduino unirse al bus I2C como maestro.
3. Convertir los valores RGB deslizantes para un color
4. Enviar el color a la BlinkM utilizando el VI GoToRGB
5. Cierra la conexión con Arduino
6. Controlar los errores

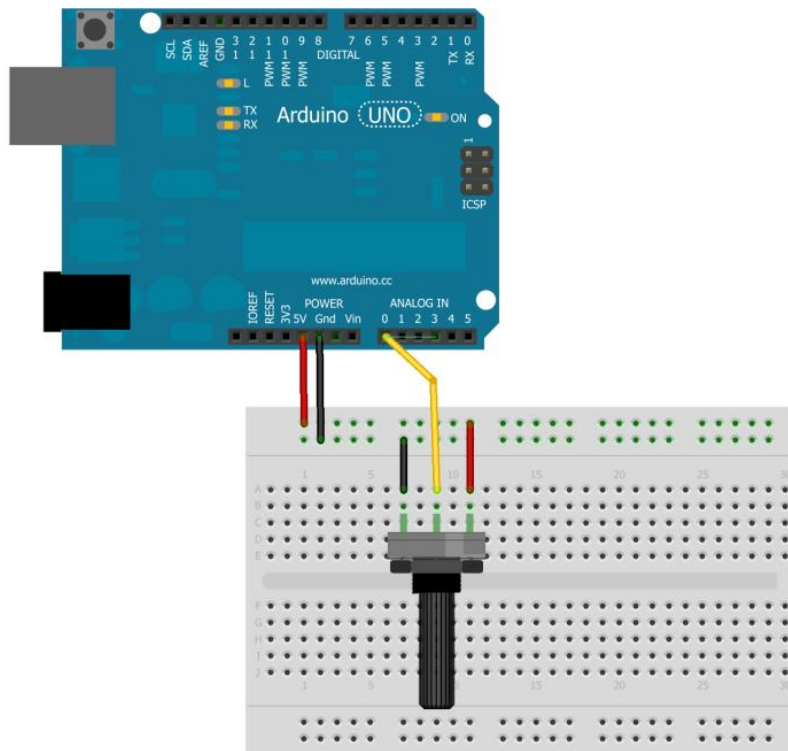
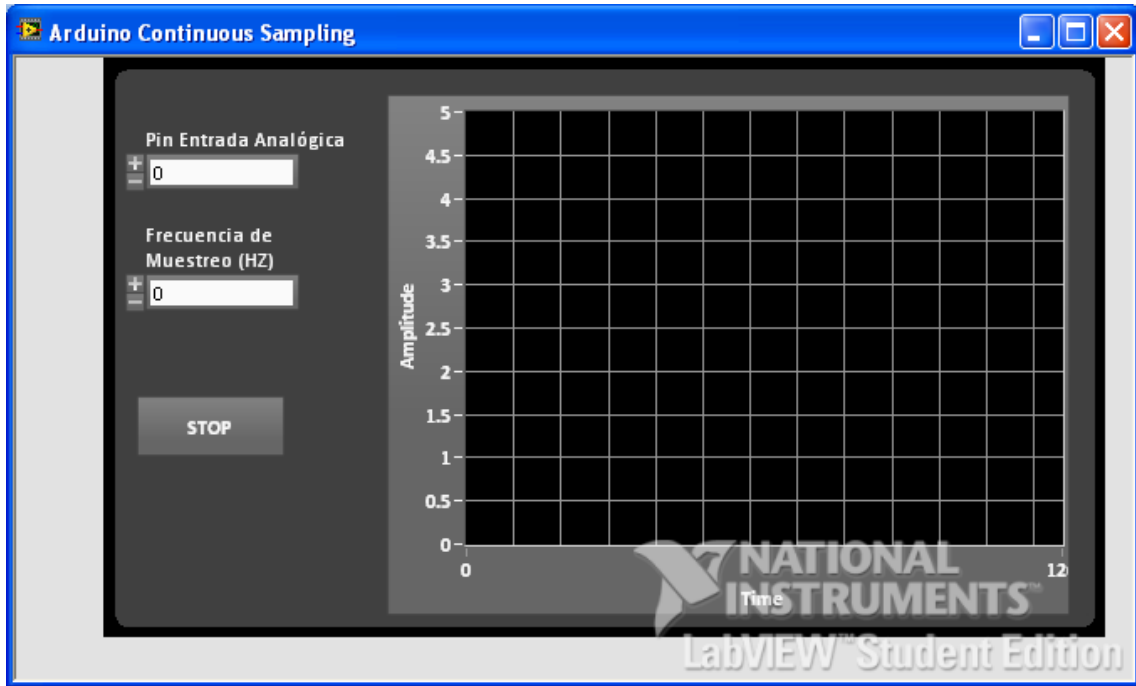
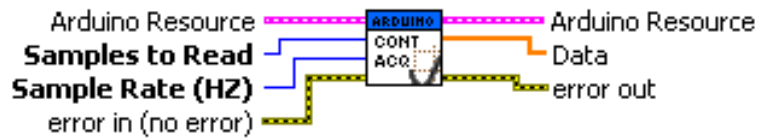
Conexión a Uno	
BlinkM Pin	Arduino Pin
-	GND
+	5V
d	A4
c	A5

E la figura siguiente vemos el aspecto de la pantalla Panel en modo edición.



El montaje es el de la figura.



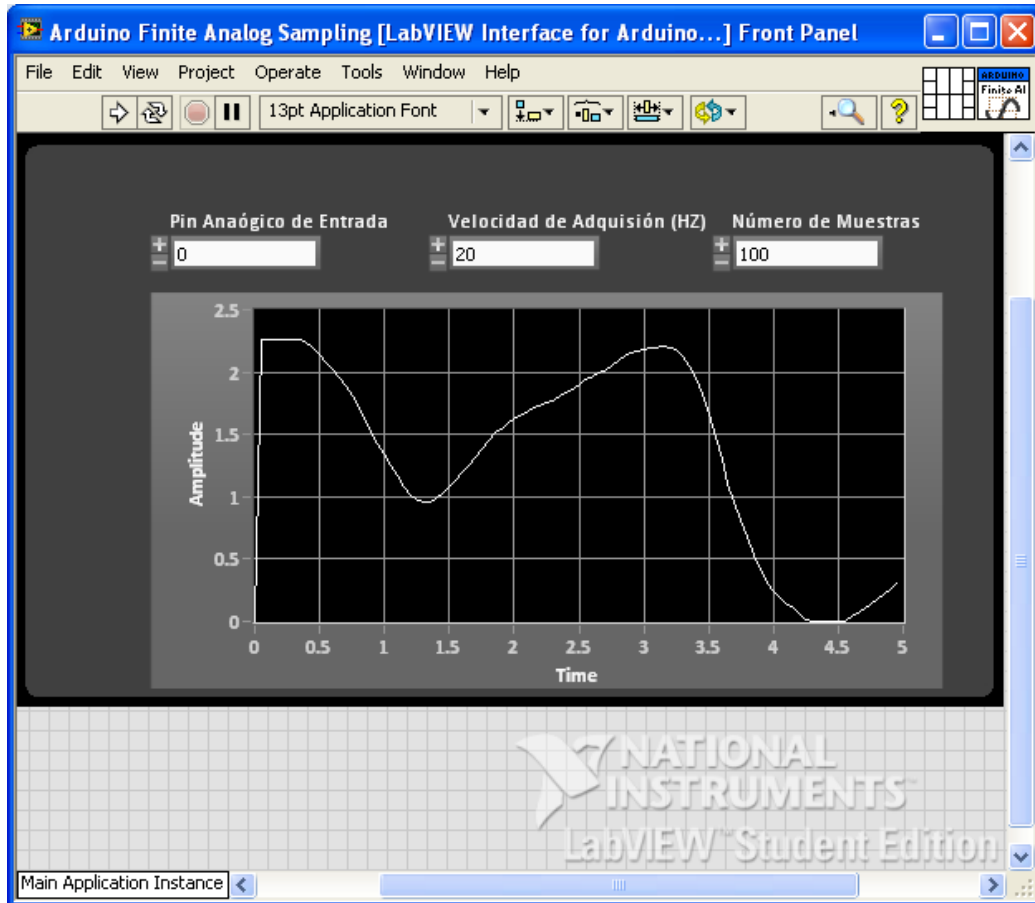


Montaje de pruebas

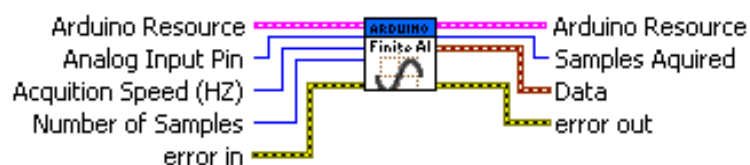
7.21. Adquisición de un número determinado de muestras de un canal analógico.

(Ejemplo traducido y adaptado del original que figura en la librería [LIFA](#) aportada por National Instruments)

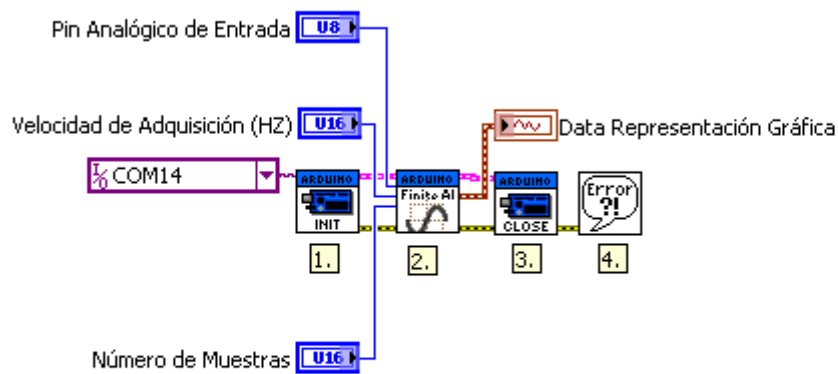
Se trata de leer un número determinado de muestras que se tomarán a una velocidad (frecuencia) dada y posteriormente mostrarlas gráficamente.



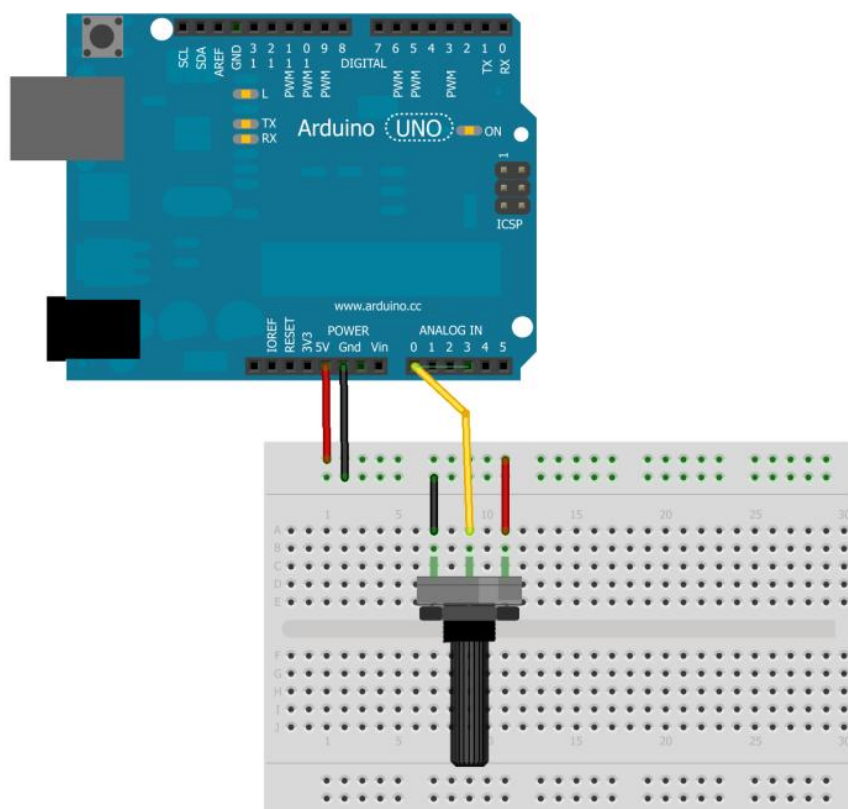
El núcleo de nuestra programa será el bloque “Get Finite Analog Sample” que realiza las funciones p`ropias de adquisición de las muestras y su entrega a un bloque de representación grafica.



Observamos en este ejemplo que no se ha colocado un bloque tipo While Loop” dado que una vez recogidas y mostradas las muestras el programa se debe detener.



1. Inicializar la conexión con el Arduino con la velocidad de transmisión por defecto de 115200.
2. Realizar la adquisición finitos con los parámetros dados
3. Cierra la conexión con Arduino
4. Controlar los errores



Montaje del prototipo

7.22. Medida de luz

(Ejemplo traducido y adaptado del original que figura en la librería [LIFA](#) aportada por National Instruments)

Se trata de realizar una aplicación para medir la cantidad de luz del ambiente.

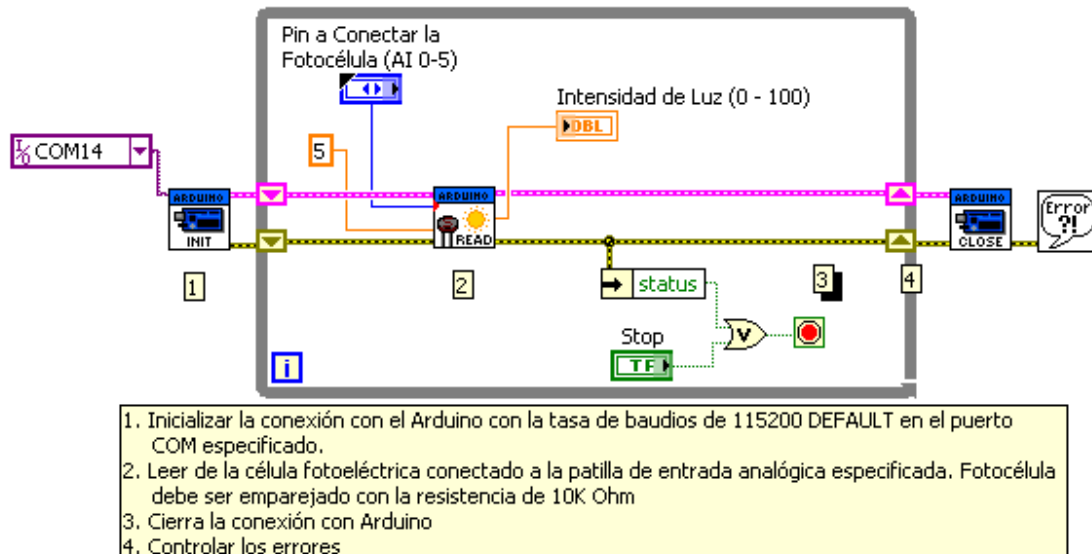
Para ello recurrimos a un bloque de función de la librería LIFA que realiza justo esta tarea. Bloque “**Phocell Read**”

Los Parámetros que hemos de configurar en este bloque se muestra n la figura



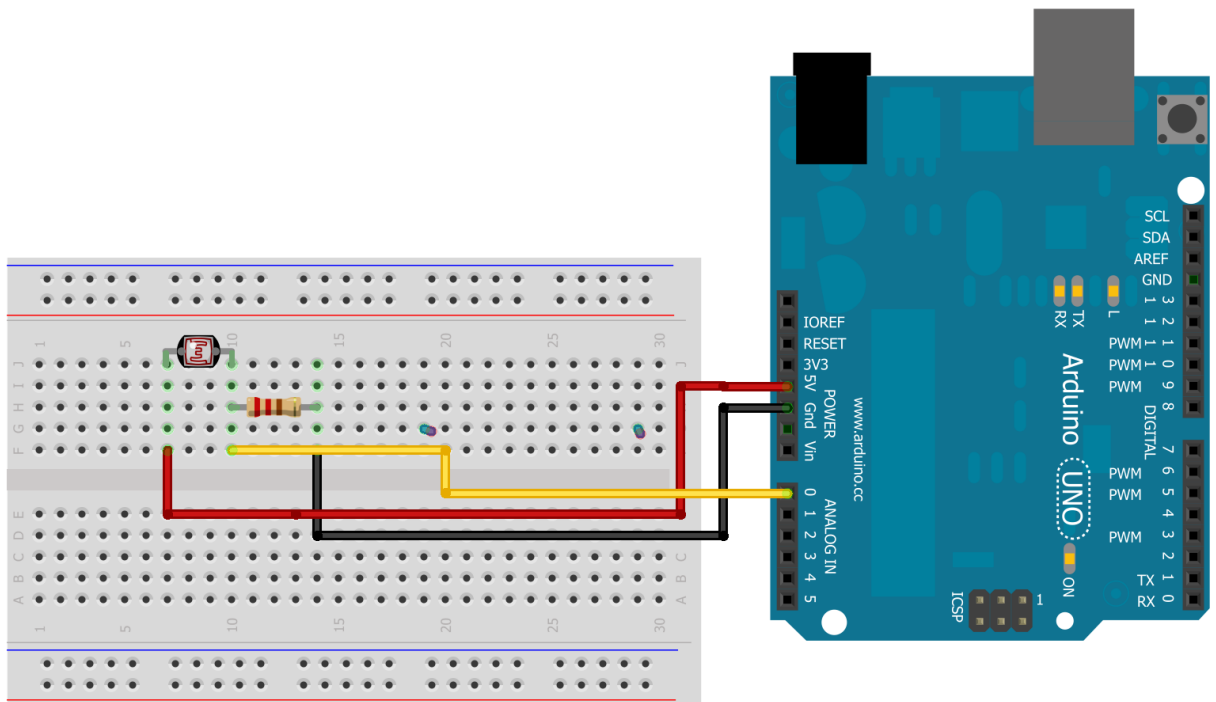
El pin a la que conectaremos la fotocélula “**Photcell Pin**” la tensión de referencia máxima que colocamos en este montaje, normalmente es 5v. sacada de la propia tarjeta Arduino

La salida del bloque es el valor equivalente a la luz medida comprendido entre 0 y 100



En este montaje la salida del bucle se ha colocado mediante un botón “Stop” y también si se produce un evento de “error”

En la figura siguiente se muestra el aspecto de la ventana “Panel” en modo ejecución



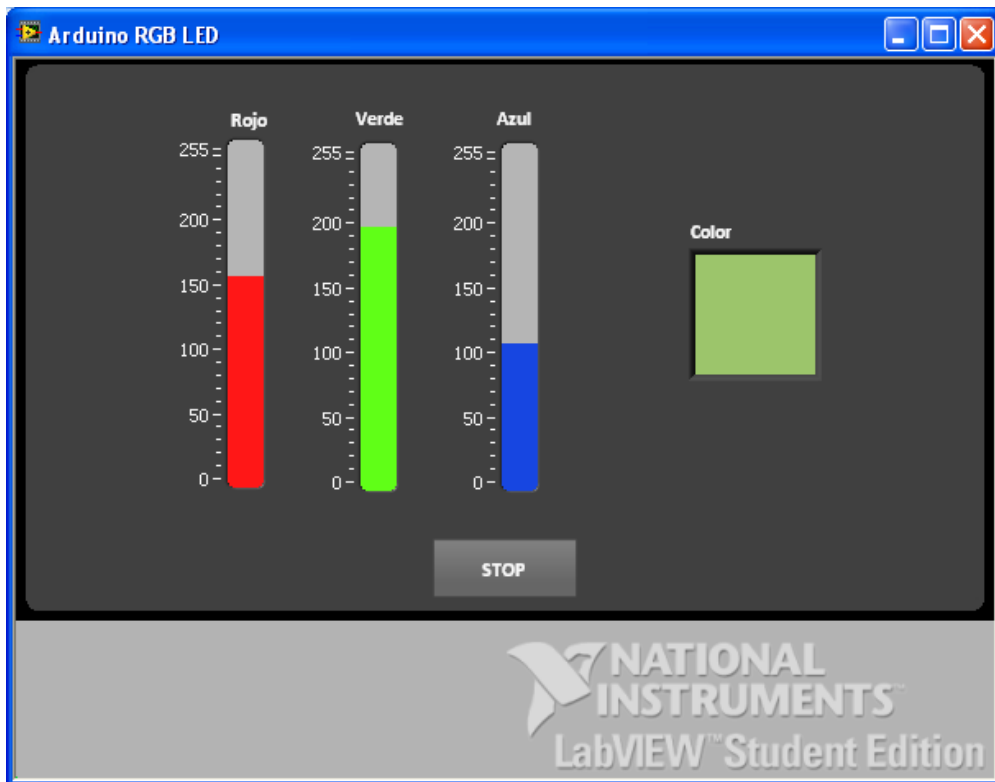
Montaje de pruebas

7.23. Control de un diodo LED Tricolor (RGB)

(Ejemplo traducido y adaptado del original que figura en la librería [LIFA](#) aportada por National Instruments)

Vamos a controlar un diodo del tipo tricolor RGB. Para ello utilizaremos dos bloques de la librería Arduino: “**RGB LED configure**” y “**RGB to Color**” .

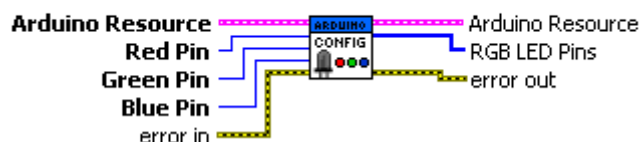
El aspecto del Panel es el de la figura siguiente. Vemos que hay tres Slider de desplazamiento que suministran valores de tipo Integer (0 – 255) correspondientes a cada uno de los tres colores de este tipo de dispositivo luminoso.



El Bloque “RGB LED Configure” permite designar los pines por donde sacaremos el valor analógico tipo PWM para cada uno de los pines RGB del LED.

Red Pin PIN 6
Green Pin PIN 5
Blue Pin PIN 3

RGB LED Configure
[LabVIEW Interface for Arduino.lvlib:RGB LED Configure.vi]

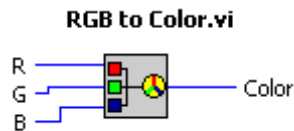


No olvidemos que deben ser pines del tipo PWM (en Arduino UNO son los pines 3,5,6,9,10,11)

Una vez dentro del bucle se colocara un bloque de escritura para sacar los valores por cada una de las salidas: **“RGB LED Write”**. La información de los PIN de las salidas se recoge del bloque de configuración anterior **“RGB LED Pins”**

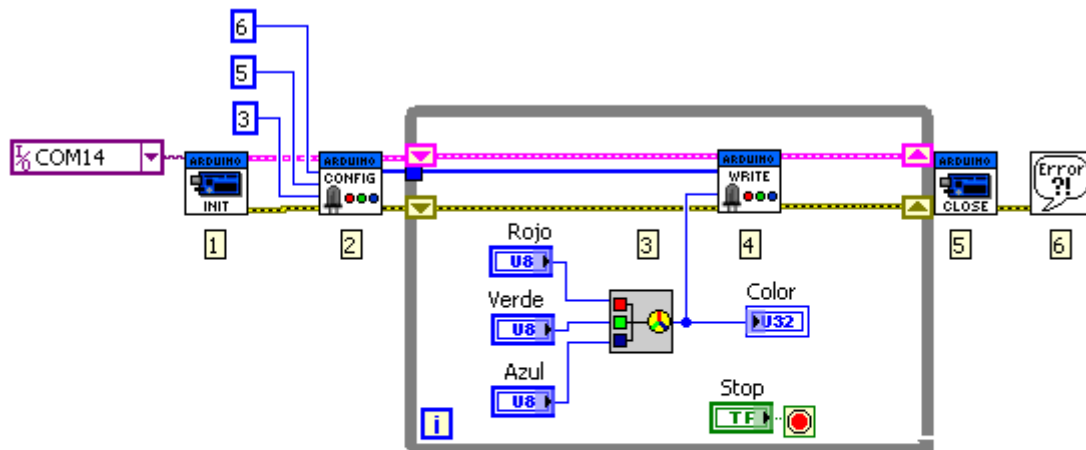


La entrada **“Color”** de este bloque de escritura la recogemos a través del bloque **“RGB to Color”** que a su vez tiene como entradas cada uno de los valores que colocamos en los sliders de entrada de color (*Rojo, Azul y Verde*) salida de este bloque nos permite

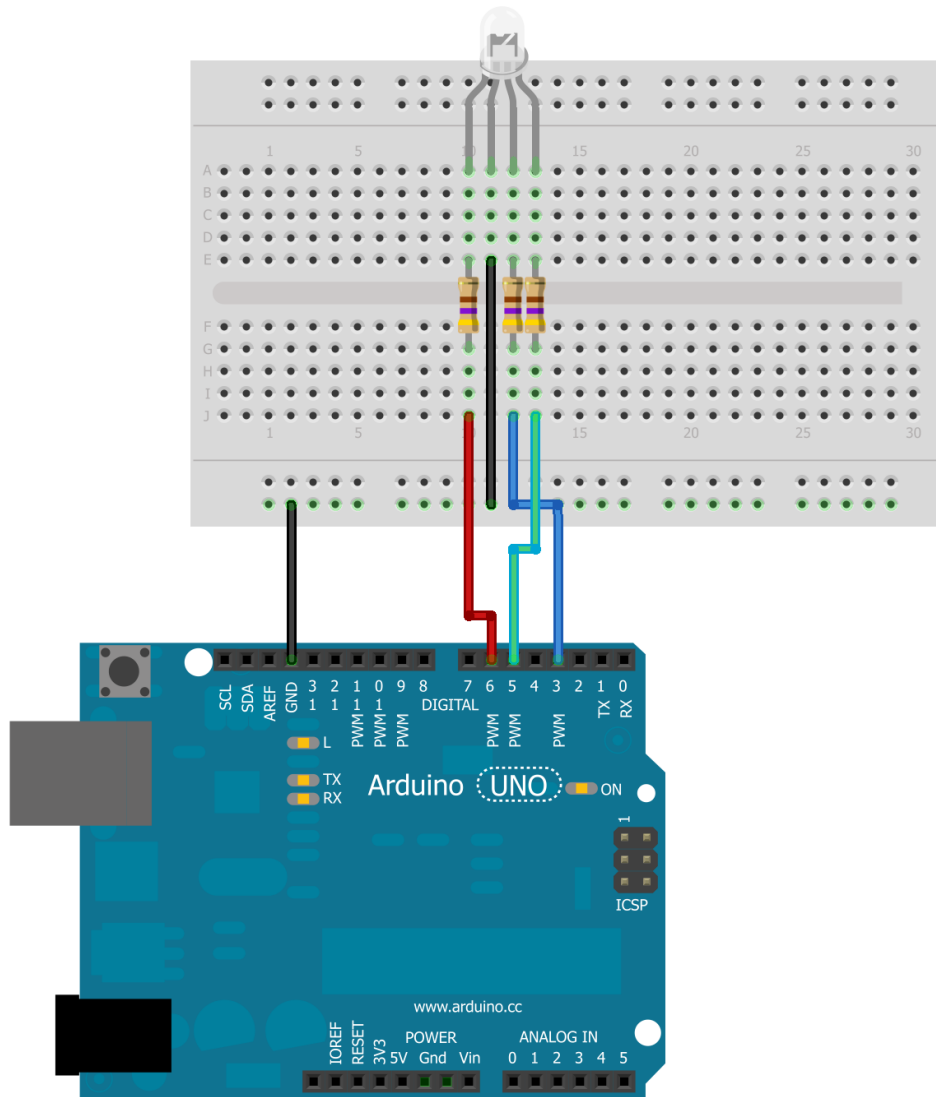


Se ha colocado un indicador de color en la salida de este bloque para mostrarnos una muestra del color que estamos sacando por el **LED RGB**: **“Color”**

A continuación mostramos el esquema funcional completo.



1. Inicializar la conexión con el Arduino con la tasa de baudios de 115200 DEFAULT.
2. Configurar los pines PWM para usar con el LED RGB
3. Convertir los valores RGB deslizando para un color
4. Enviar el color del LED RGB
5. Cierra la conexión con Arduino
6. Controlar los errores



Este sería el montaje del prototipo de pruebas

7.24. Medida de temperatura

(Ejemplo traducido y adaptado del original que figura en la librería [LIFA](#) aportada por National Instruments)

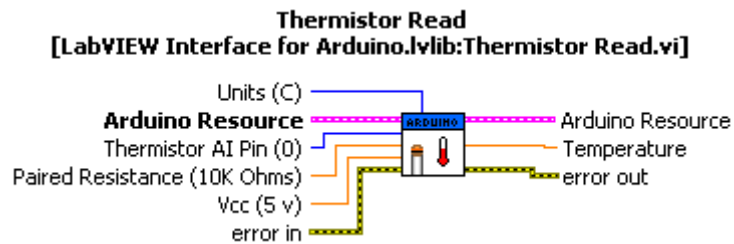
Vamos a realizar un ejemplo en el que utilizaremos un bloque específico de la librería LIFA: “**Thermistor Read**”.

La parametrización de este bloque consiste en definir:

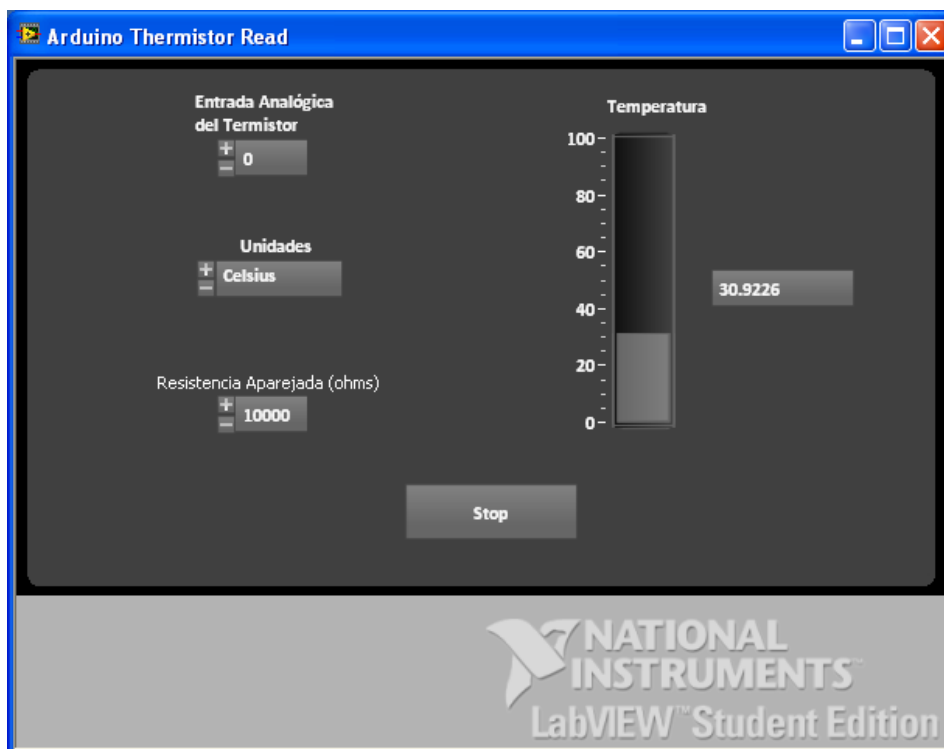
<i>Thermistor AI Pin</i>	Canal de entrada Analógico
<i>Paired Resistance</i>	Resistencia a conectar en serie con el sensor
<i>Vcc</i>	Valor de la tensión de referencia (normalmente 5 v.)
<i>Units</i>	Permite seleccionar las unidades de escala (°C, °F y °K)

Estos valores los pondremos, en esta ocasión, a través del Panel de visualización.

La visualización del valor analógico (temperatura) leído la mostraremos con una “barra de color” *Temperatura* y un cuadro numérico.



A continuación mostramos el aspecto del Panel de visualización en modo ejecución.



7.25. Manipulación de un Mando Joystick.

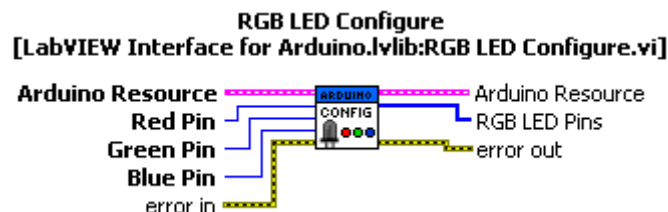
(Ejemplo traducido y adaptado del original que figura en la librería [LIFA](#) aportada por National Instruments)

En numerosas aplicaciones realizadas con Arduino se utiliza un joystick, por eso presentamos este ejemplo en el que se utilizan las librerías de tratamiento de este componente.

Se trata de leer las señales generadas por el mando y redireccionarlas a un bloque de control de LED RGB.

La aplicación lo que hará será controlar el color del LED RGB en función de las posiciones del joystick.

En primer lugar, fuera del bucle de control se pondrá el bloque **“Init”** y los dos bloques de configuración del LED RGB y del joystick: **“RGB LED Configure”**



Se designarán los PIN para los colores Rojo, Verde y Azul

Red Pin: PIN 6

Green Pin: PIN 5

Blue Pin: PIN 3

La salida de este componente se deberá llevar luego al bloque **“RGB LED Write”**

Seguidamente configuramos el joystick para lo cual designamos



Los pines para cada eje y para el botón.

Horizontal Axis AI Pin: 2

Vertical Axis AI Pin: 3

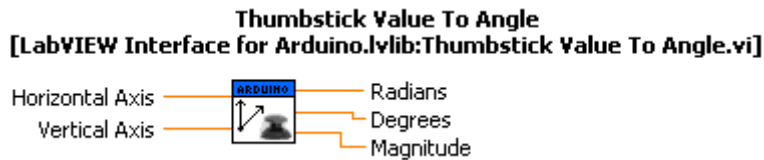
Select DI Pin: 4

En este caso igual que en el anterior este bloque posee una salida que contiene la información de configuración del joystick que deberá trasladarse al bloque **“Thumbdtick Read”**

Dentro del bucle de control colocamos en primer lugar el bloque **“Thumbdtick Read”** que lee la posición y estado del boton del joystick y lo saca en sus salidas *Horizontal Axis*, *Vertical Axis* y *Select*



La información que entrega este control, se lleva a un bloque convertidor **“Thumbstick Value to Angle”** que convierte los valores “x” e “y” en un valor *“angulo”* que entrega a un bloque que convierte este valor en un color **“Angle Mag to RGB”**



De este bloque utilizamos las salidas *“Degrees”* y *“Magnitude”* para atacar el siguiente bloque que las convertirá en un color, salida *“RGB Color”*



Finalmente esta señal se llevará a un bloque de salida RGB Color **“RGB LED Write”**

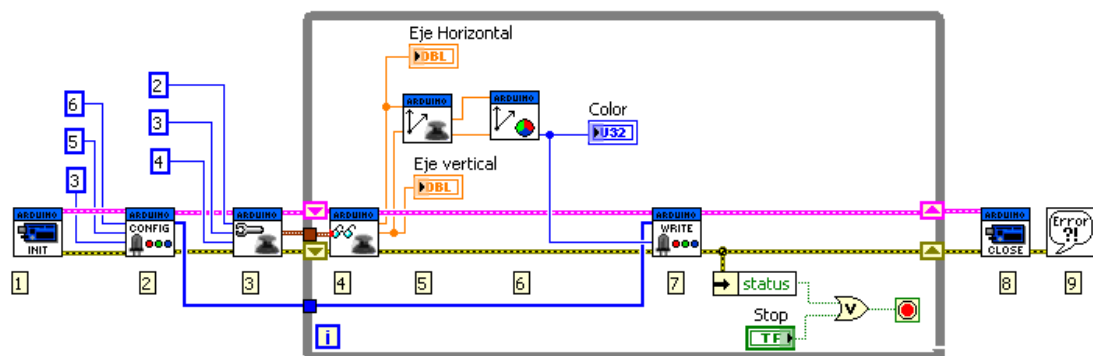


El LED RGB se configurará escribiendo los pines de salida *“RGB Color”* en el pin de entrada *“Color”* del bloque **“RGB LED Write”**

A continuación se muestra el diagrama completo de conexionado de bloques.

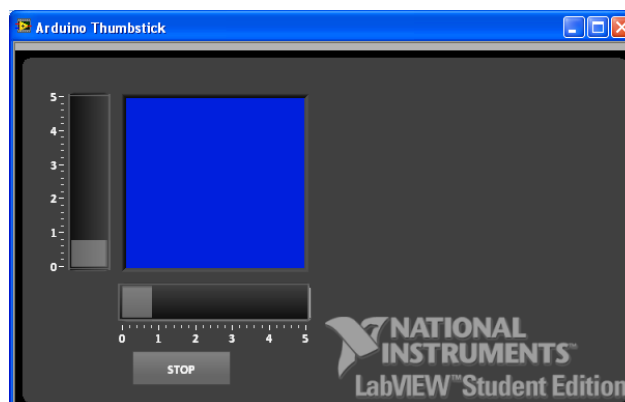
Básicamente los pasos son los anotados en la imagen siguiente:

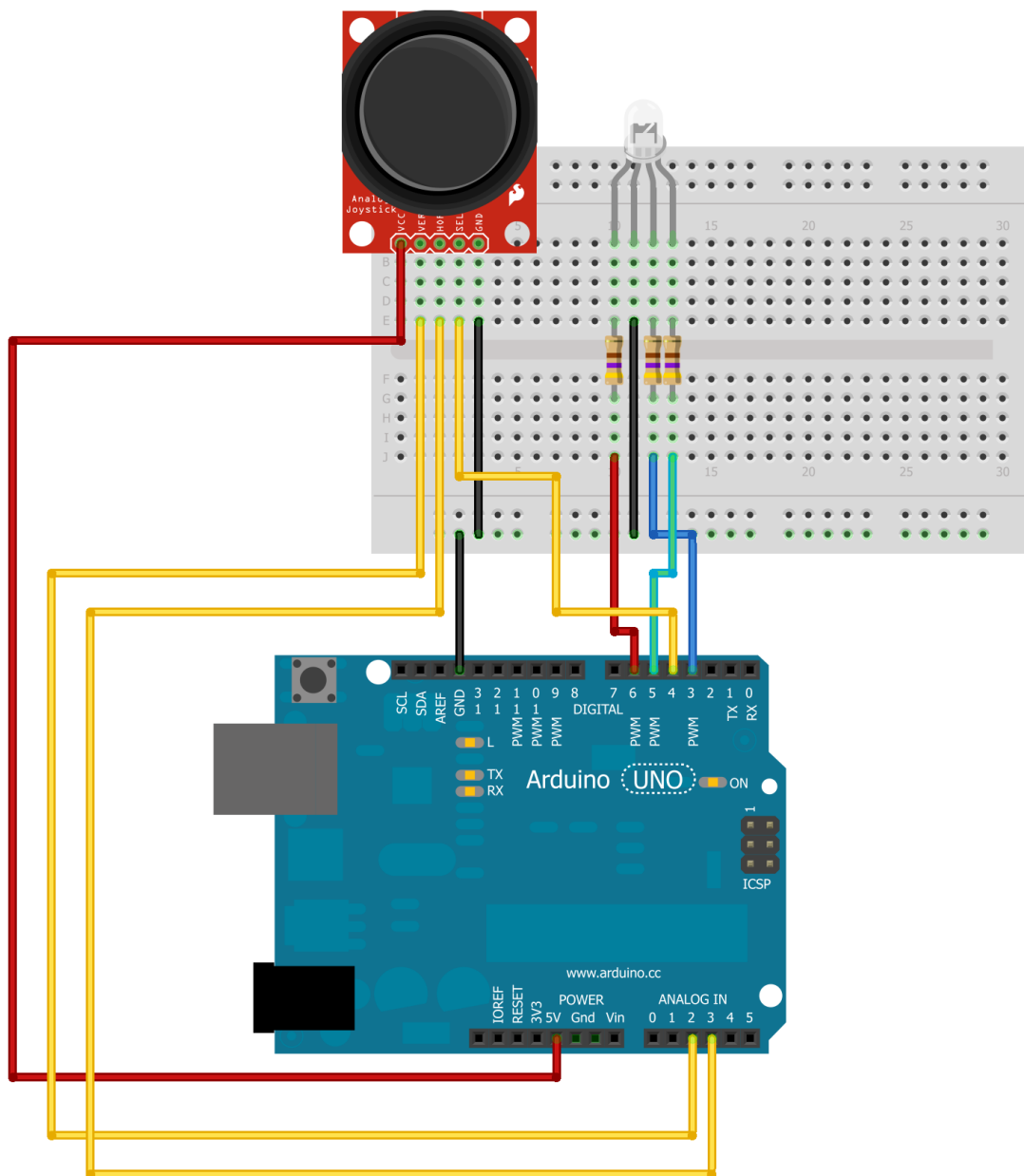
1. Inicializar la conexión con el Arduino con la tasa de baudios de 115200 DEFAULT.
2. Configuración de LED RGB
3. Configuración de pines para el uso con el joystick
4. Leer los valores de joystick (horizontal y vertical del eje y de selección)
5. Convertir los valores de joystick con el ángulo y la magnitud
6. Convertir el ángulo y la magnitud de color y la intensidad
7. Escribe el color de LED RGB
8. Cierra la conexión con Arduino
9. Controlar los errores



1. Inicializar la conexión con el Arduino con la tasa de baudios de 115200 DEFAULT.
2. Configuración de LED RGB
3. Configuración de pines para el uso del joystick
4. Lea los valores de joystick (horizontal y vertical del eje y de selección)
5. Convertir los valores de joystick con el ángulo y la magnitud
6. Convertir el ángulo y la magnitud de color y la intensidad
7. Escribe el color de LED RGB
8. Cierra la conexión con Arduino
9. Controlar los errores

Este sería el aspecto de la pantalla Panel en modo ejecución.



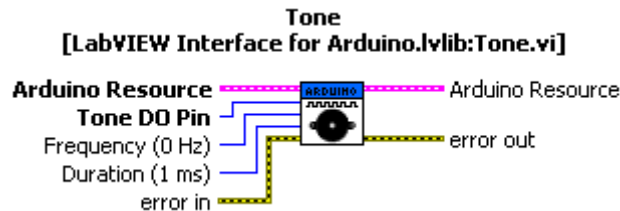


Esquema de conexiones

7.26. Generador de Tonos

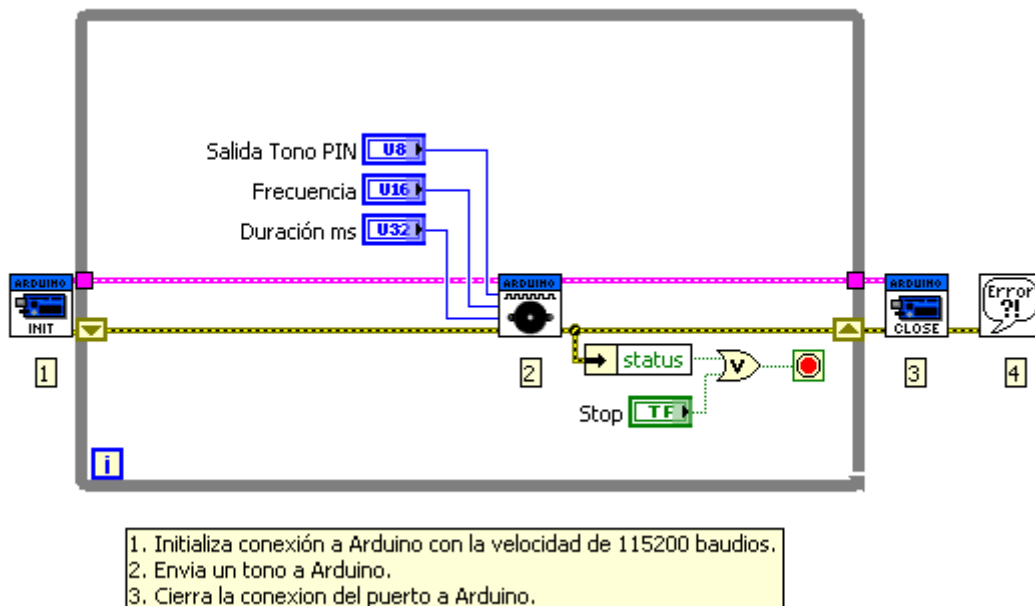
(Ejemplo traducido y adaptado del original que figura en la librería [LIFA](#) aportada por National Instruments)

En este ejemplo vamos a probar la función “**Tone**” de la librería de Arduino



Esta librería se encarga de generar una señal de frecuencia variable en la que es posible también variar la duración. Es decir genera tonos de duración y frecuencia variable.

Probaremos su funcionamiento generando un tono a través de uno de los PIN de salida digital seleccionable con un control numérico desde el Panel en el que además podremos variar frecuencia y duración. Vemos en la siguiente figura el aspecto del diagrama de bloques funcional.

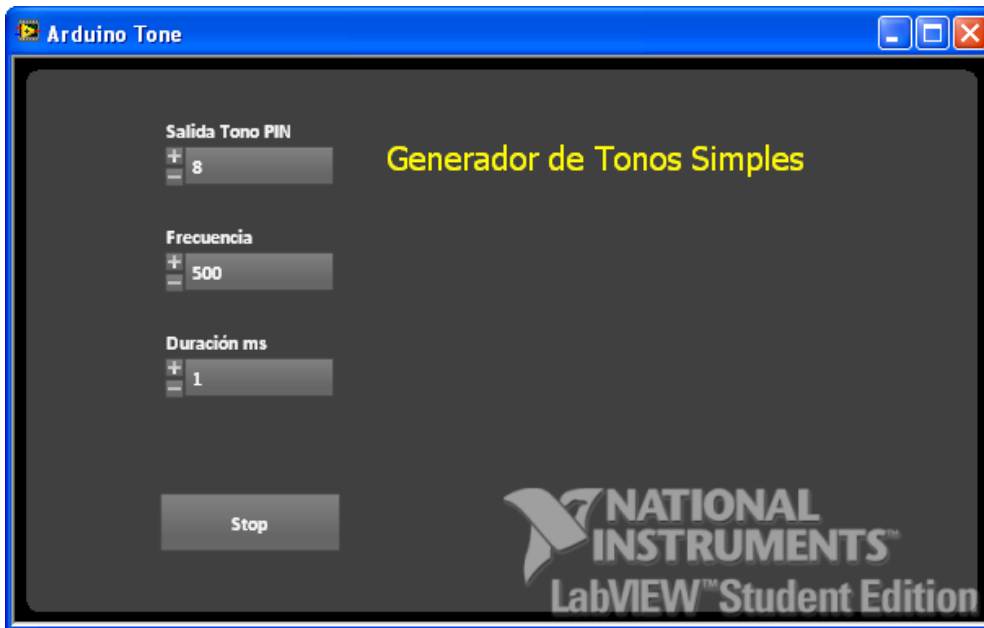


1. Inicializa conexión a Arduino con la velocidad de 115200 baudios.
2. Envía un tono a Arduino.
3. Cierra la conexión del puerto a Arduino.

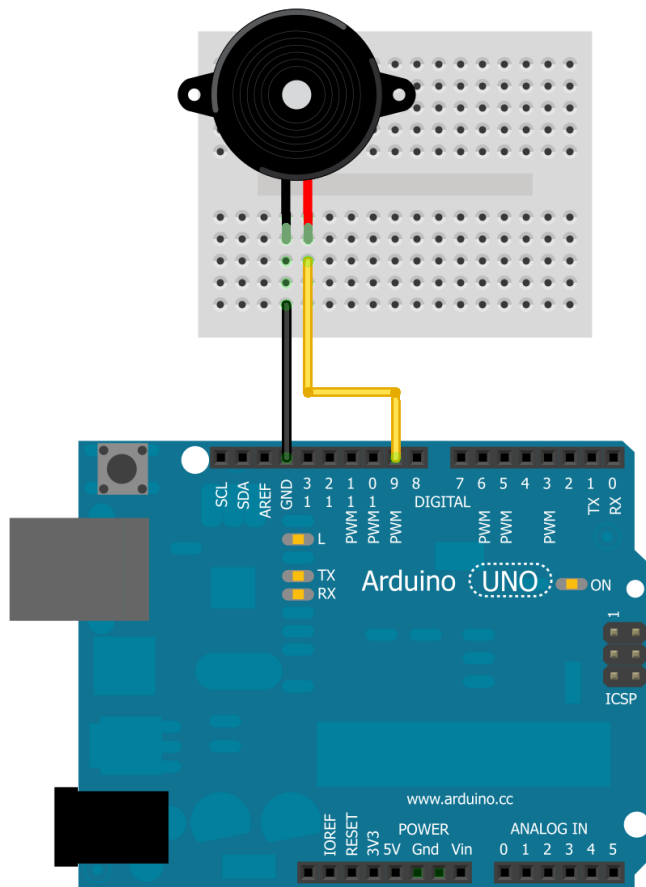
Se ha colocado dentro del bucle de ejecución el bloque “**Tone**” y en esta ocasión se ha cableado la señal de error para que esta pueda también cerrar el bucle y detener el programa.

En la siguiente figura se el Panel.

No olvidemos que en los PIN 0 y 1 no se podrá sacra tono alguno dado que estos están ocupados para la comunicación con LabVIEW.



A continuación mostramos el conexionado para realizar las pruebas





ANEXO:

Material Básico

A modo de sugerencia describimos el Kit que en la actualidad vende Sparkfun Electrónica

<https://www.sparkfun.com/products/11225>



Arduino+LabVIEW Bundle

49.95\$ (agosto 2012)

Características del Kit

Arduino + Paquete de LabVIEW

DEV-11225 RoHS

Descripción: ¿Alguna vez un nuevo sensor de SparkFun y desea probarlo rápidamente? Tal vez usted quiere hacer un mayor procesamiento de los datos que el Arduino puede proporcionar. Imagínese cómo sería tener esos datos a su computadora y tener una GUI (interfaz gráfica de usuario) para su sensor de pocos minutos de abrir la caja roja.

NI LabVIEW es un entorno de programación gráfica utilizado por millones de ingenieros y científicos para desarrollar la medición sofisticado, prueba, y los sistemas de control mediante intuitivos iconos gráficos y cables que se asemejan a un diagrama de flujo. A través de la interfaz de NI LabVIEW para Arduino Kit de herramientas, que ahora pueden aprovechar todos los beneficios de la programación gráfica de LabVIEW de NI para sus proyectos de Arduino.

La LIFA (Interfaz de LabVIEW para Arduino) caja de herramientas es una descarga gratuita que permite a un desarrollador de LabVIEW para obtener fácilmente los datos hacia y desde el microcontrolador Arduino cada vez más popular. La arquitectura básica detrás de esto es que hay un motor de E / S programada para el Arduino que espera órdenes de serie de LabVIEW y responde con los datos solicitados o de acción.

Este kit incluye un R3 Arduino Uno y el LabVIEW Student Edition DVD para Windows y MacOS. Sólo tiene que cargar el firmware de código abierto a la incluida Uno Arduino, conéctelo a su computadora e instalar el software LabVIEW.

Nota: El envío a Canadá y los EE.UU. solamente. Lo sentimos mundo.

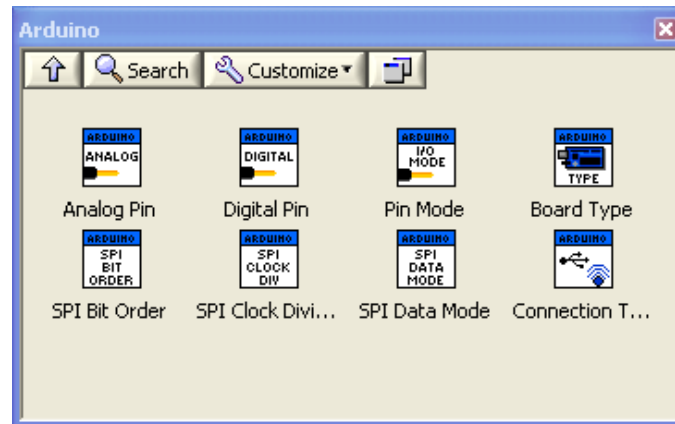
Nota: Aunque las imágenes muestran a la ahora obsoleta LabVIEW 2010, que son el envío de 2011. La foto de Arduino no es también la R3 a pesar de que este paquete incluye ahora la última Arduino Uno R3. Imágenes de los productos se actualizarán para reflejar este cambio.

Características:

- Utiliza USB, XBee, o enlaces Bluetooth para la comunicación.
- Tasa de 150 Hz con conexión de cable (50 Hz Wireless).
- Le da acceso a analógica de Arduino, digitales, PWM, I2C, SPI y la funcionalidad en el equipo.
- Completamente de código abierto del firmware (E / S del motor en el Arduino) y el marco de LabVIEW.

Librerías del Kit LIFA de LabVIEW para Arduino

Descripción de controles del panel frontal de la Librería Arduino



Analog Pin



Digital Pin



Pin Mode



Board Type



SPI Bit Order



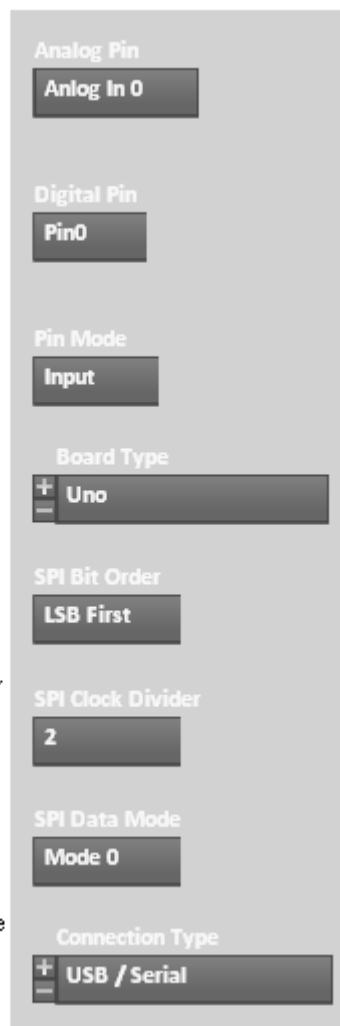
SPI Clock Divider



SPI Data Mode



Connection Type



Permite seleccionar un Pin de Entrada Analógica

Permite seleccionar un Pin de Entrada Salida Digital

Permite designar un Pin como Entrada o como Salida

Selecciona el tipo de Tarjeta

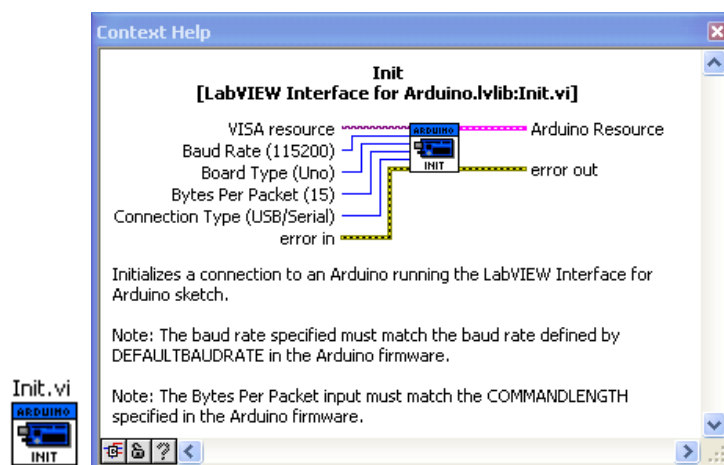
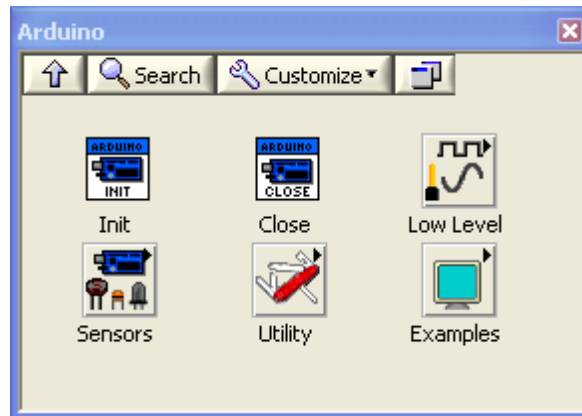
Selecciona tipo de Byte

Factor de División de la frecuencia de Reloj

Modo de dato

Permite seleccionar el puerto con el que nos comunicaremos con Arduino

Descripción e Librería Arduino del módulo Diagrama de Bloques de Labview



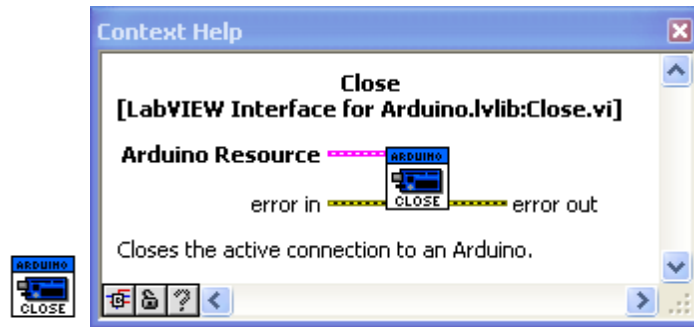
Modulo de Inicio de la Tarjeta Arduino

Este módulo es imprescindible en cualquier configuración que hagamos, permite configurar las características de Arduino. Por defecto viene con los parámetros que se indican en la figura anterior.

Los conectores “**Arduino Resource**” y “**error out**” se deben cablear a todos los elementos que se coloquen en el diagrama con el fin de que los parámetros de

configuración se transfieran a todos los bloques.

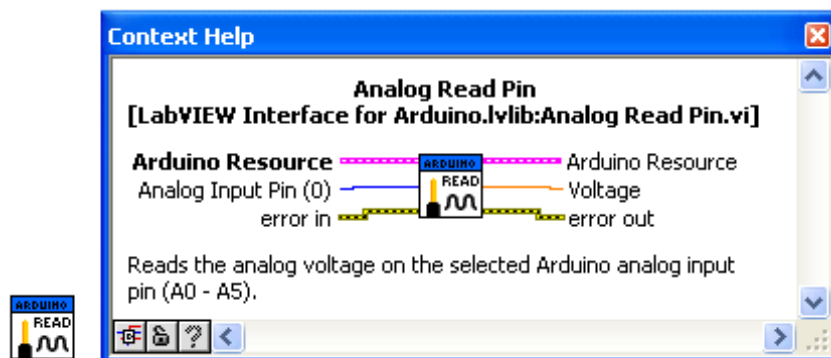
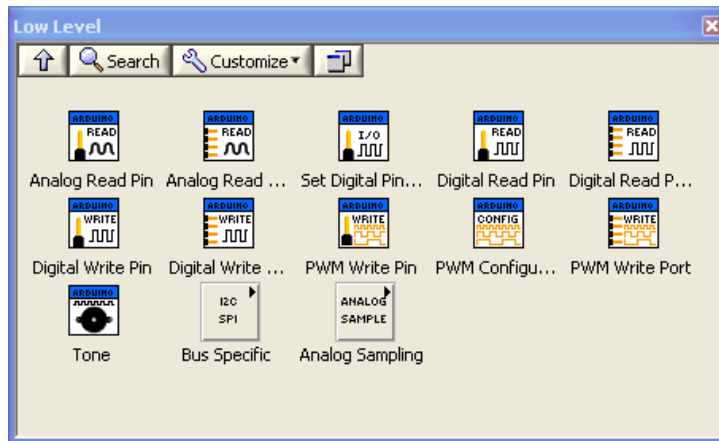




Módulo de cierre de la conexión con el puerto.

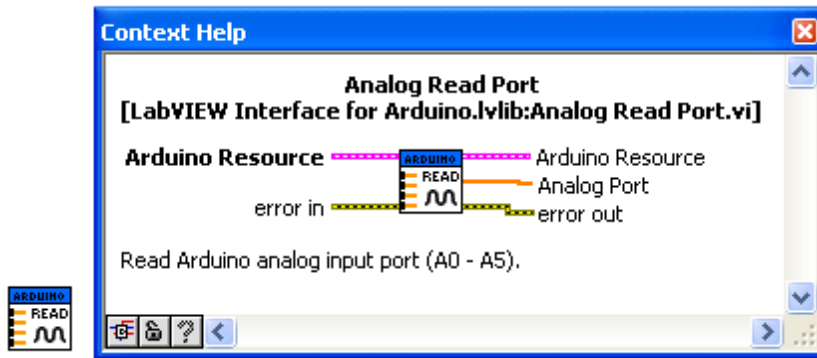


Funciones “low level”



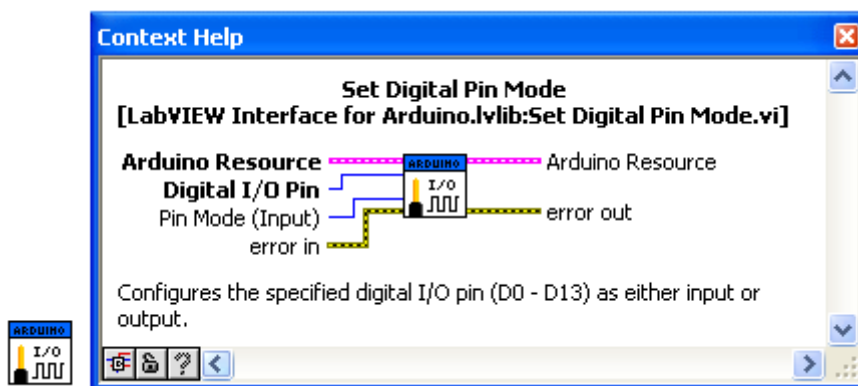
Lectura de una entrada analógica de Arduino.

Recoge de entrada el número de canal en formato INTEGER y devuelve de salida el valor leído en formato DBL.

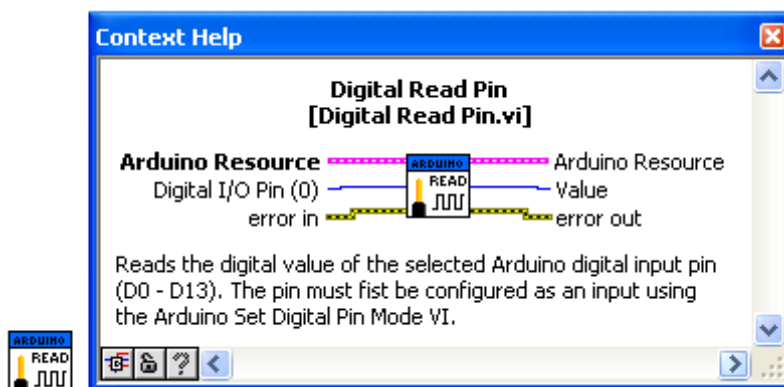


Lectura del puerto analógico.

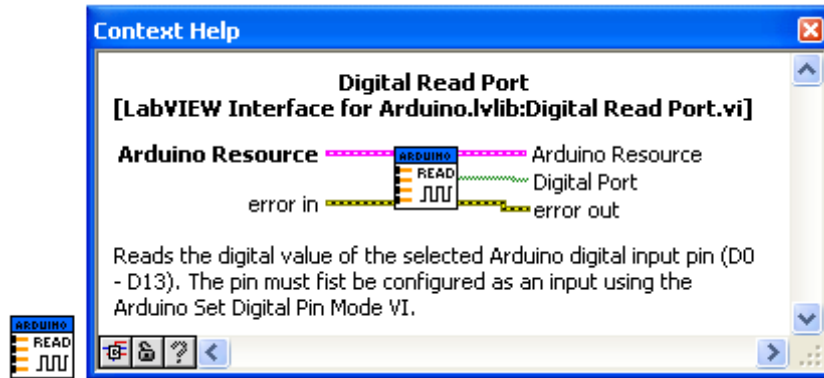
Devuelve el valor leído en el Puerto Analógico



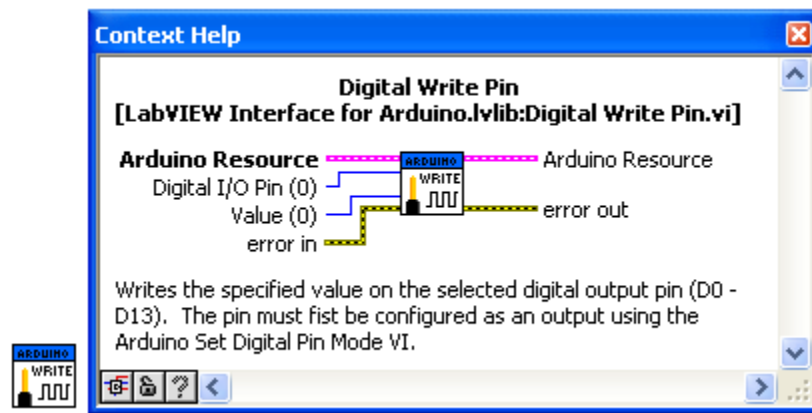
Configura un PIN digital como Entrada o salida. Los parámetros de entrada serán el número de puerto y el modo de trabajo (I/O).



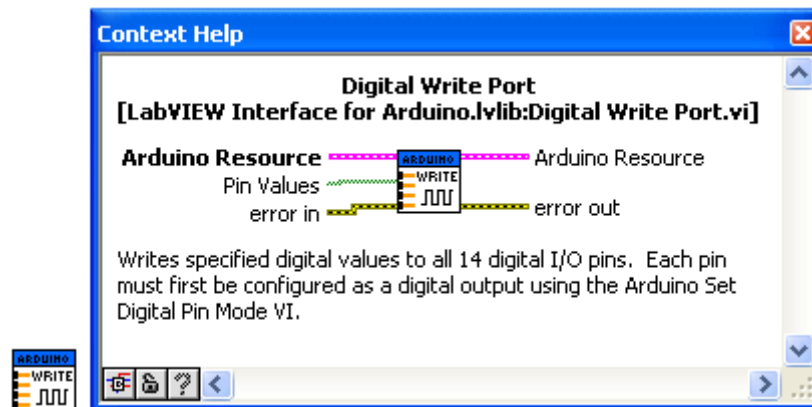
Lectura del valor en un pin digital. Lee el valor de una entrada digital designada mediante el parámetro "Digital I/O" y saca el valor en su salida "Value".



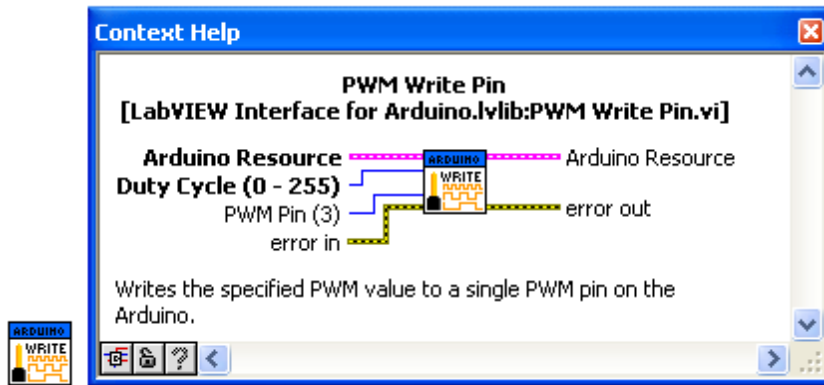
Lectura del puerto digital. Lee el puerto digital completo y lo saca en el terminal “Digital Port” en formato 1D Array



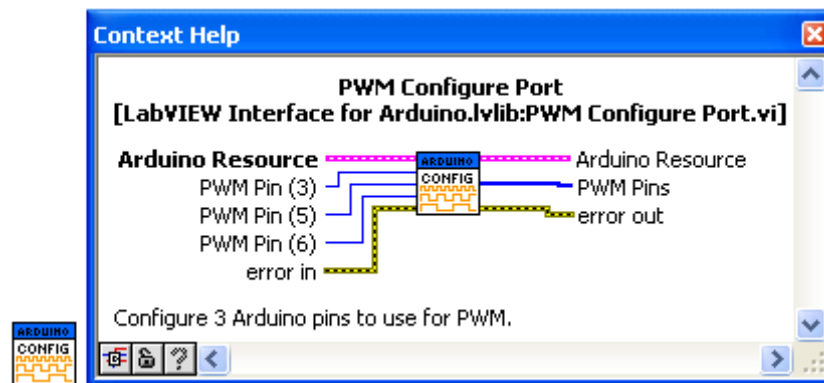
Escritura en PIN Digital. Escribe un valor digital (0 o 1) “Value”, en el Pin indicado en “Digital I/O Pin”



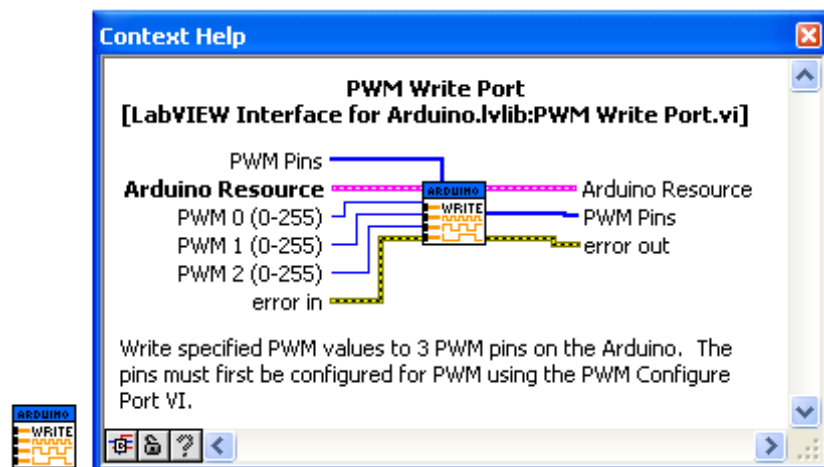
Escritura en puerto digital: Escribe de una sola vez los valores a los 14 PINs de entrada salida digitales de la tarjeta Arduino. Primero debe configurarse como salida digital el conjunto de pines Digitales



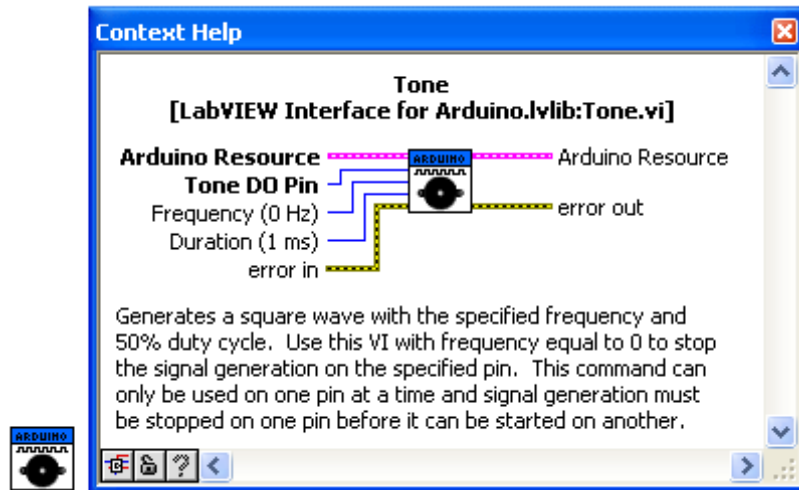
Escritura en salida Analógica PWM: Escribe un valor entre 0-255 en la salida especificada en el Pin



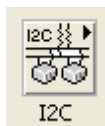
Configura3 PIN como salida PWM: Se indican los PINs y devuelve el valor de los puertos configurados



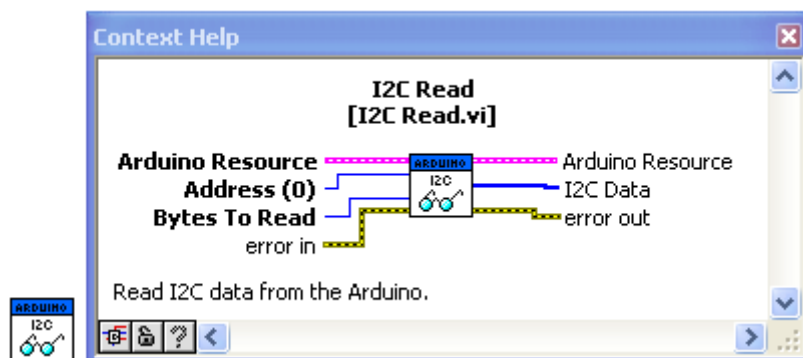
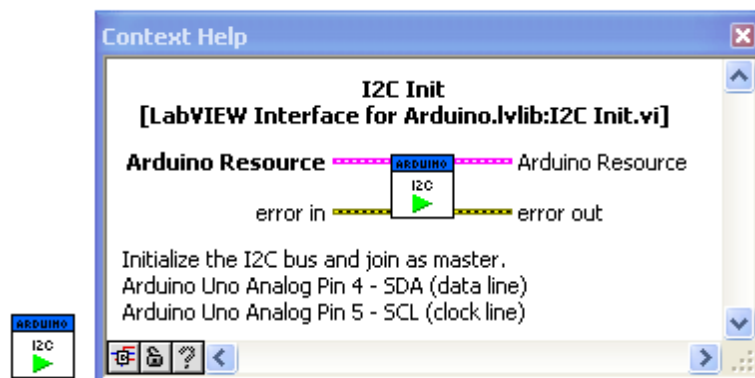
Escribe el puerto PWM: Controla tres salidas PWM cuyo número de PINs los recoge del valor PWM PINs. Los valores de cada señal se suministran en los pines de entrada PW 0, PWM 1 y PWM 2



Genera un Tono: Genera un tono de frecuencia y duración variables en el PIN designado mediante Tone DO Pin

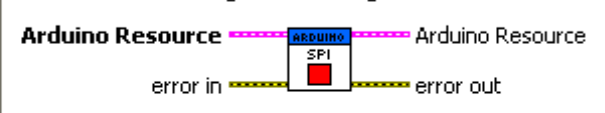


Librerías para implementación Bus I2C



Context Help

SPI Close [SPI Close.vi]




The diagram shows the SPI Close.vi block. It has two 'Arduino Resource' inputs on the top, connected by pink dashed lines. It has an 'error in' input on the left and an 'error out' output on the right, both connected by yellow dashed lines. The block itself is a square with 'ARDUINO' and 'SPI' written inside, and a red square icon.

Disables the SPI bus. Pin modes are unchanged.

ARDUINO SPI

Context Help

SPI Set Bit Order [SPI Set Bit Order.vi]



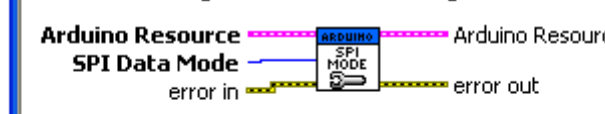
The diagram shows the SPI Set Bit Order.vi block. It has two 'Arduino Resource' inputs on the top, connected by pink dashed lines. It has an 'error in' input on the left and an 'error out' output on the right, both connected by yellow dashed lines. It also has an 'SPI Bit Order' input on the left, connected by a blue solid line. The block itself is a square with 'ARDUINO', 'SPI BIT ORDER', and a circular arrow icon.

Set the bit order of the SPI bus (LSB - Least Significant Bit First or MSB - Most Significant Bit First).

ARDUINO SPI BIT ORDER

Context Help

SPI Set Data Mode [SPI Set Data Mode.vi]



The diagram shows the SPI Set Data Mode.vi block. It has two 'Arduino Resource' inputs on the top, connected by pink dashed lines. It has an 'error in' input on the left and an 'error out' output on the right, both connected by yellow dashed lines. It also has an 'SPI Data Mode' input on the left, connected by a blue solid line. The block itself is a square with 'ARDUINO', 'SPI MODE', and a circular arrow icon.

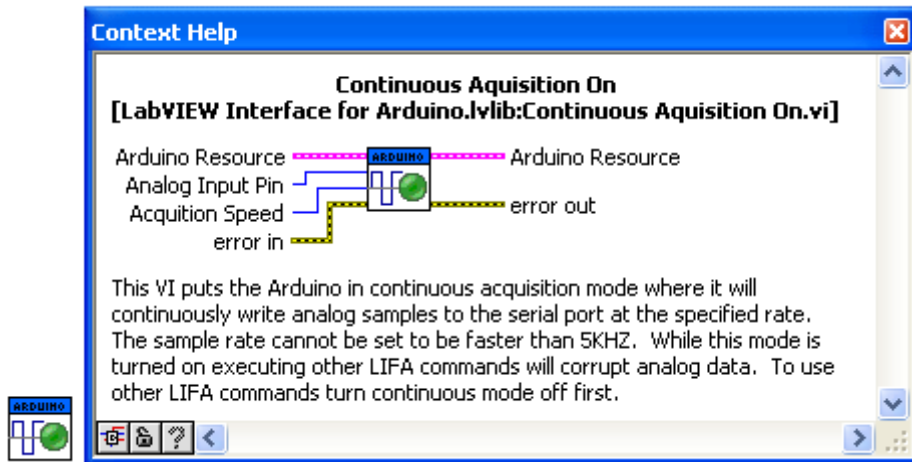
Set the SPI data mode as follows:

Mode 0:	Clock Polarity - 0	Clock Phase - 0
Mode 1:	Clock Polarity - 0	Clock Phase - 1
Mode 2:	Clock Polarity - 1	Clock Phase - 0
Mode 3:	Clock Polarity - 1	Clock Phase - 1

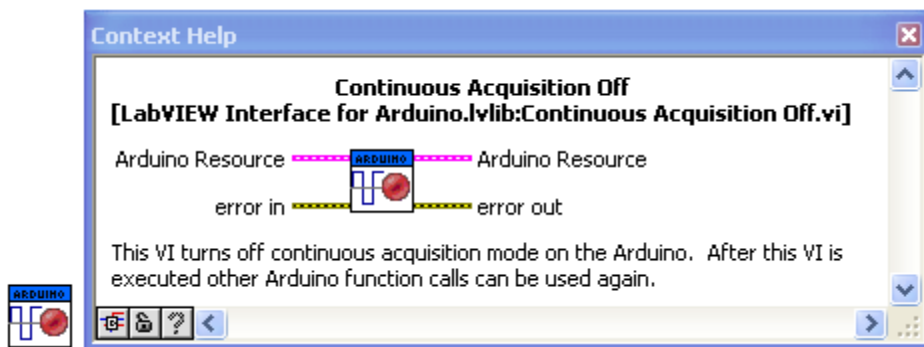
ARDUINO SPI MODE



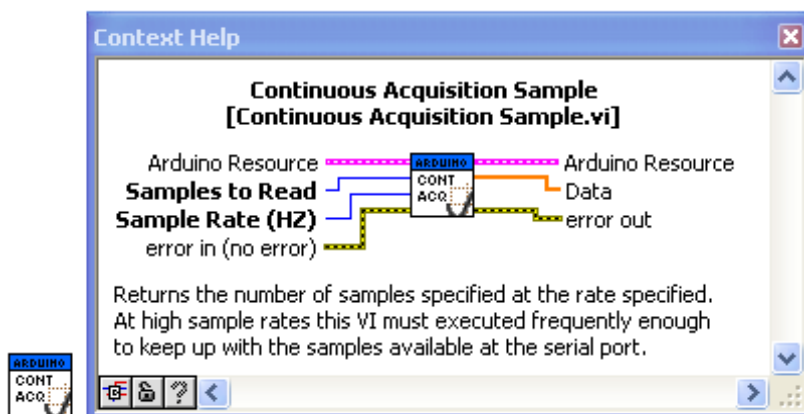
Librería Analog Sampling



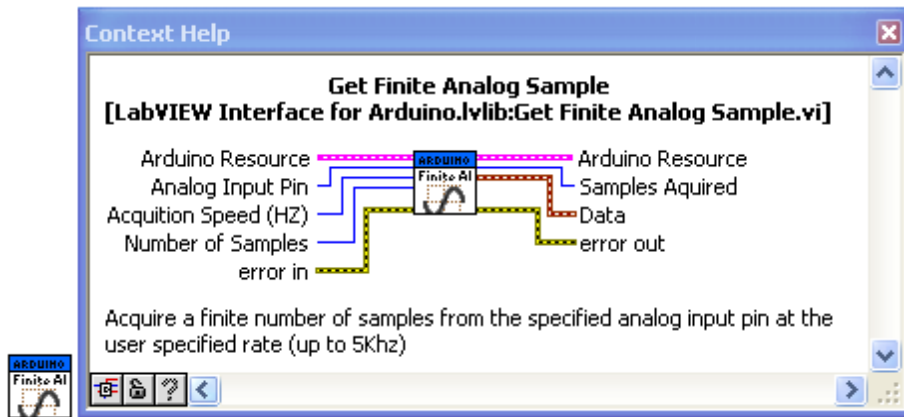
Adquisición continua de datos: Permite activar la lectura de datos en el de una entrada analógica teniendo la posibilidad de asignar el tiempo de muestreo en la lectura



Detiene la adquisición continua de los datos



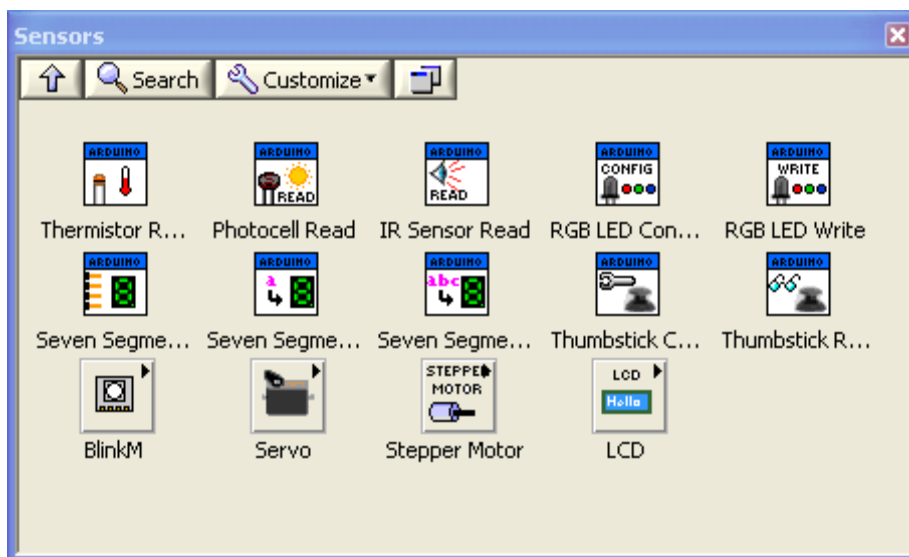
Adquisición de un numero de muestras de valor: Lee un determinado numero de veces (muestras) un canal de entrada analógica con una frecuencia de barrido determinada por el usuario.



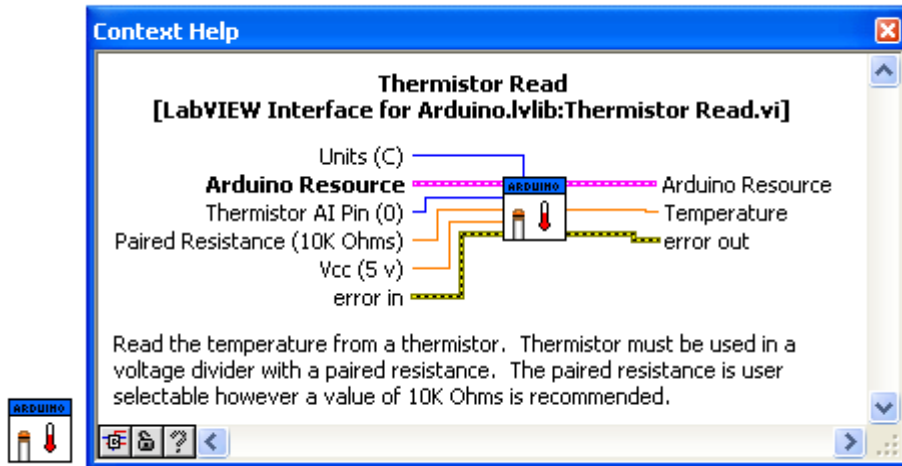
Recoge un numero finito de muestras de datos: Recoge un determinado número de muestras por un Pin analógico devolviendo un array de datos con los valores



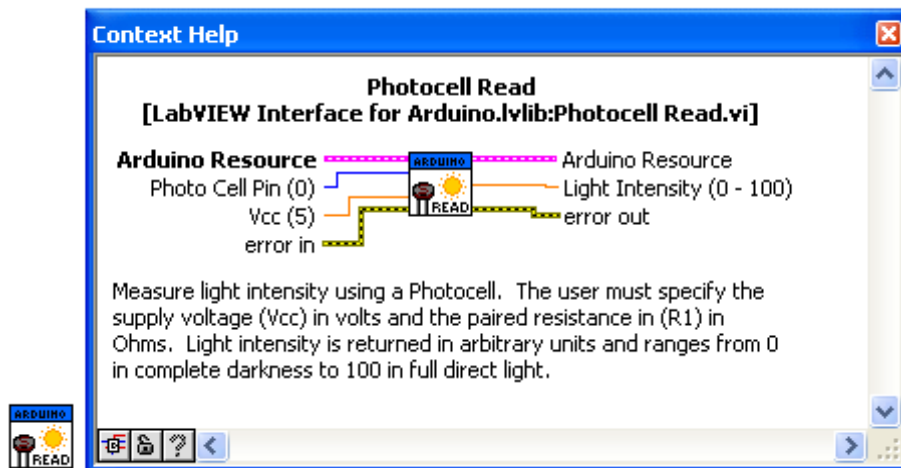
Librería Sensores



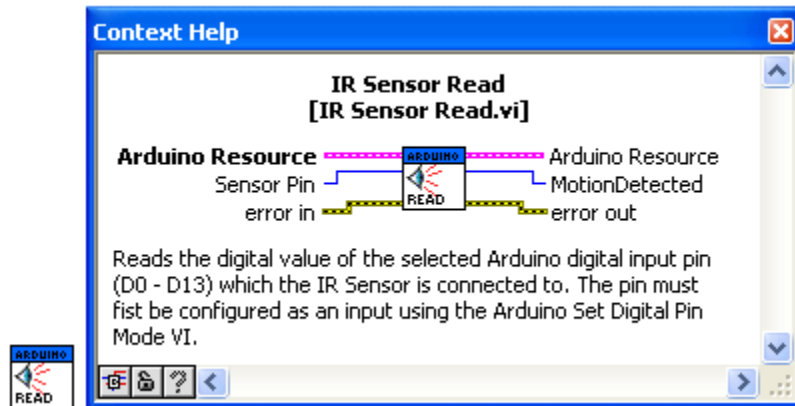
En esta librería se ofrece una amplia colección de bloques de función que permiten la configuración de la conexión de señores a la tarjeta Arduino



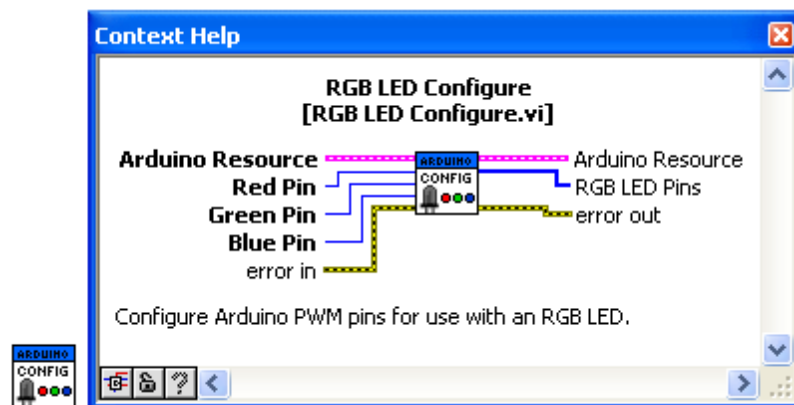
Lee el valor de un sensor de Temperatura: Permite la configuración del PIN en el que se colocara el sensor así como el valor de la resistencia divisora de tensión que conectemos con el fin de realizar el escalado de la medida y también la posibilidad de establecer una tensión de referencia distinta a los 5 vcc que por defecto toma. Devuelve un valor de tipo Double equivalente a la temperatura medida.



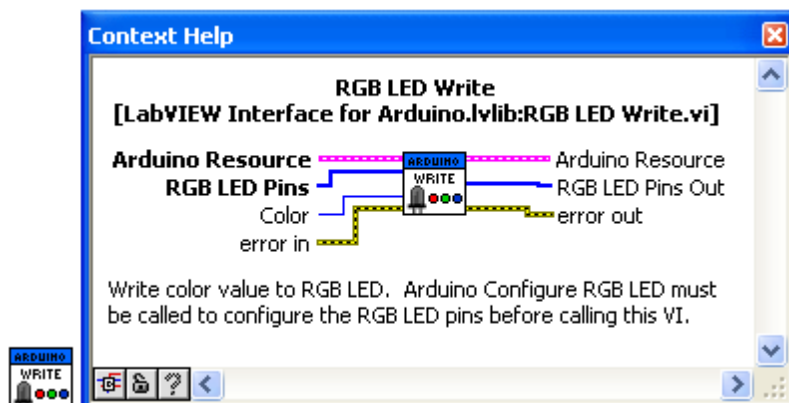
Lee el valor de un sensor de Luz: Permite la configuración del PIN en el que se colocara el sensor así como el y también la posibilidad de establecer una tensión de referencia distinta a los 5 vcc que por defecto toma. Devuelve un valor de tipo Double entre 0-100 equivalente a la cantidad de luz medida.



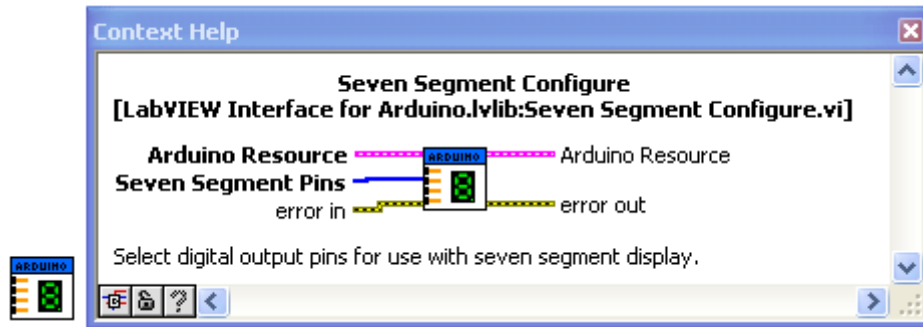
Lee el valor de un sensor de Infrarrojos: Permite la configuración del PIN en el que se colocara el sensor. Devuelve un valor de tipo Integer equivalente a la temperatura medida.



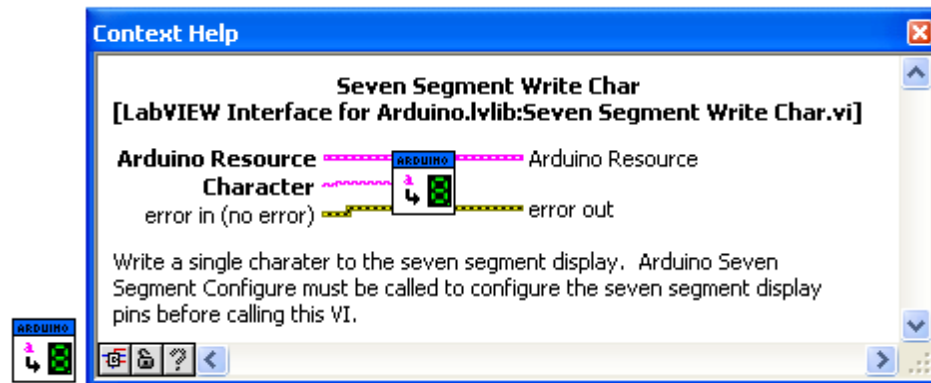
Configura un LED RGB: Se designan los pins para salida de cada color



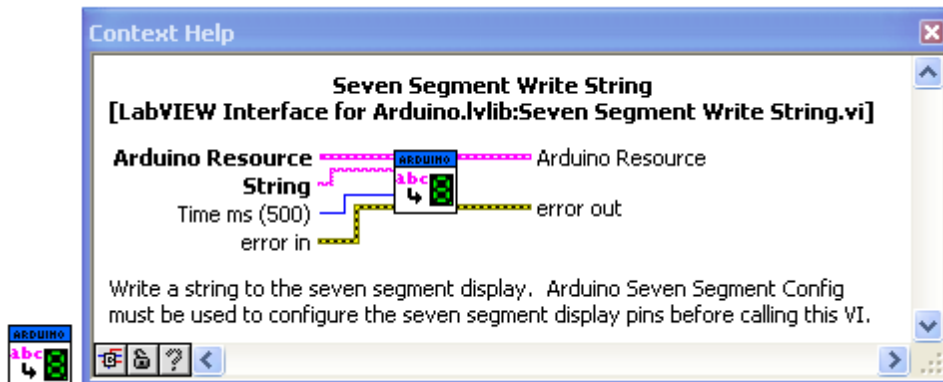
Escribe en un LED RGB: Se escribe el valor Color sobre un led RGB en los pines designados RGB LED Pins



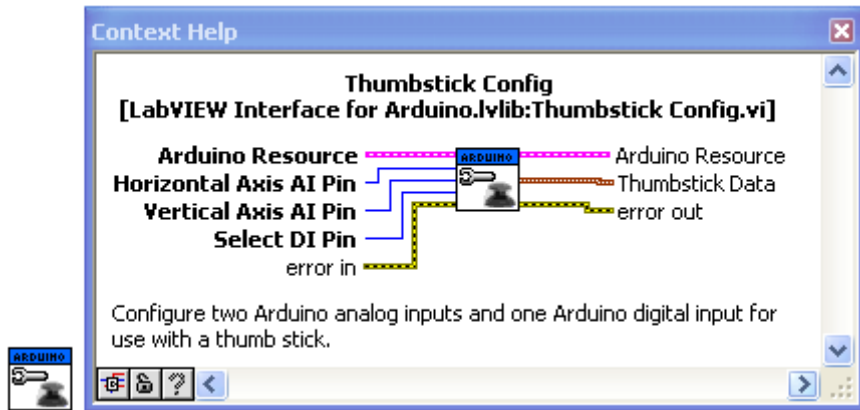
Configura un display de siete segmentos



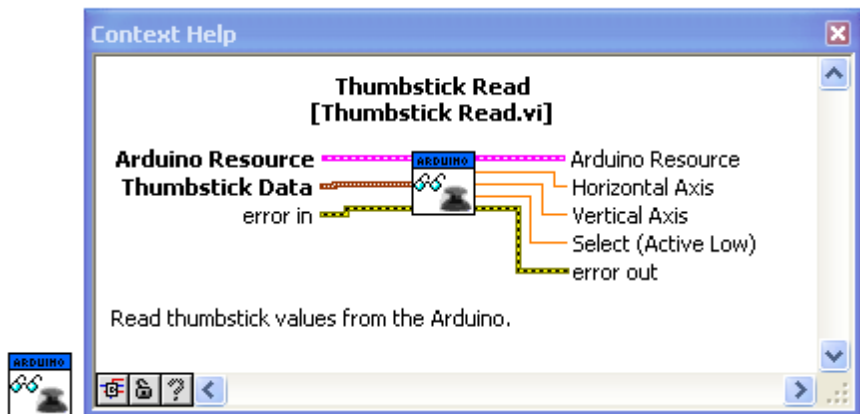
Escribe un carácter en un display de siete segmentos.



Escribe un String en un display de siete segmentos



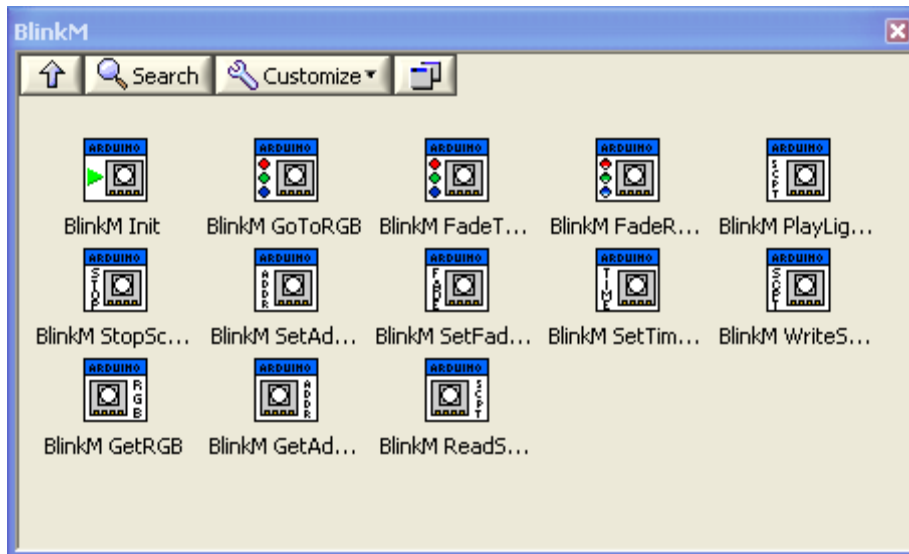
Configura un joystick: Se designan los pines analógicos para el eje X y el eje Y así como la entrada digital para el botón de selección. Genera una señal de salida en forma de cluster de 3 elementos



Lee valores de un joystick: Lee los valores y los presenta en las salidas correspondientes a los ejes X e Y y Selección



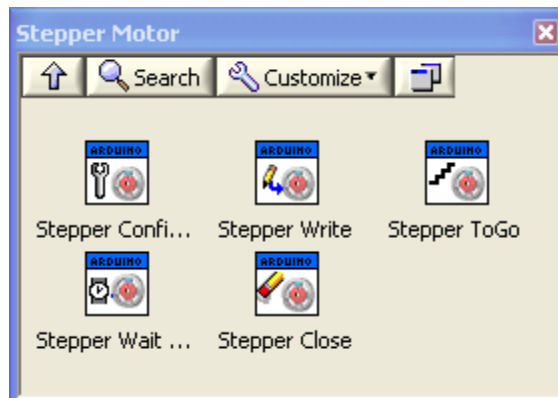
Librería BlinkM



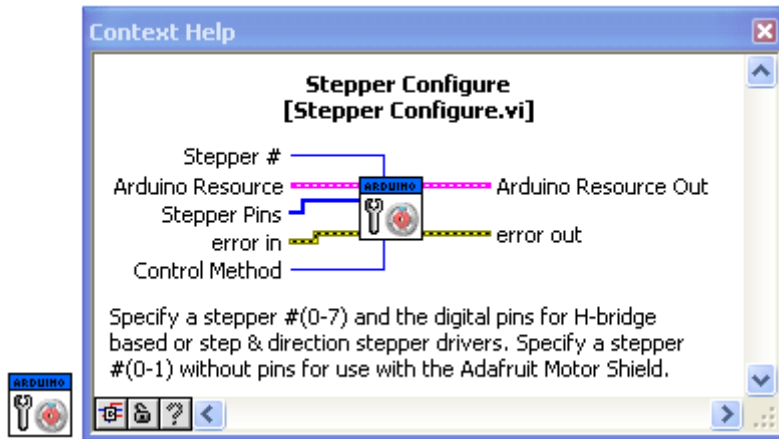
Esta librería sirve para el manejo de elementos luminosos tipo BlinkM



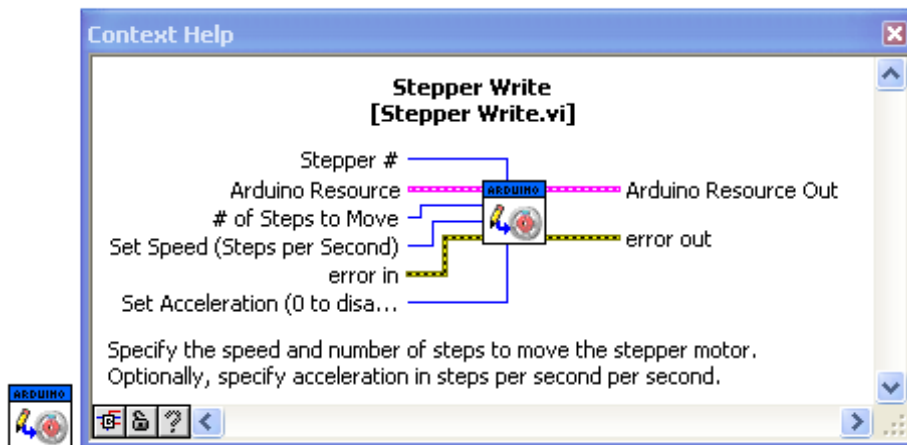
Librería Stepper Motor



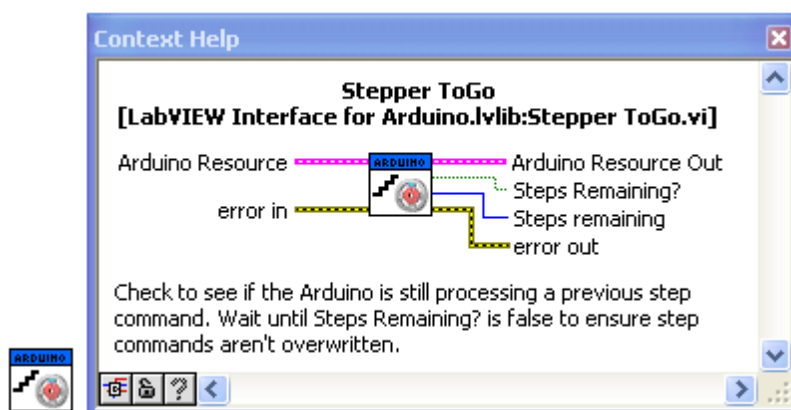
Librería para control de motores paso a paso



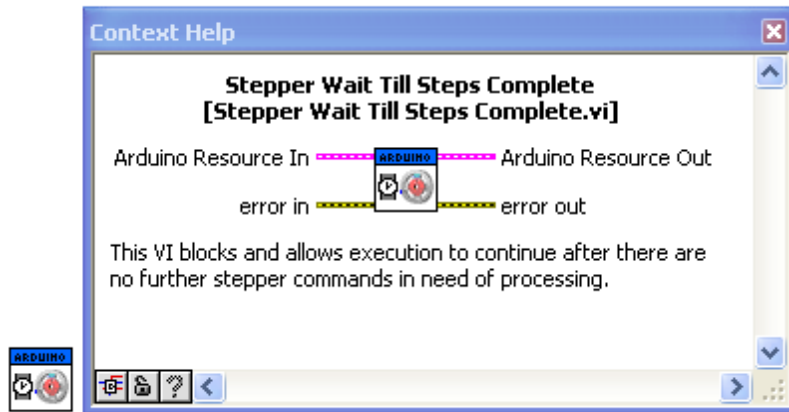
Configura motor paso a paso: Especifica una salida para conectar un motor.



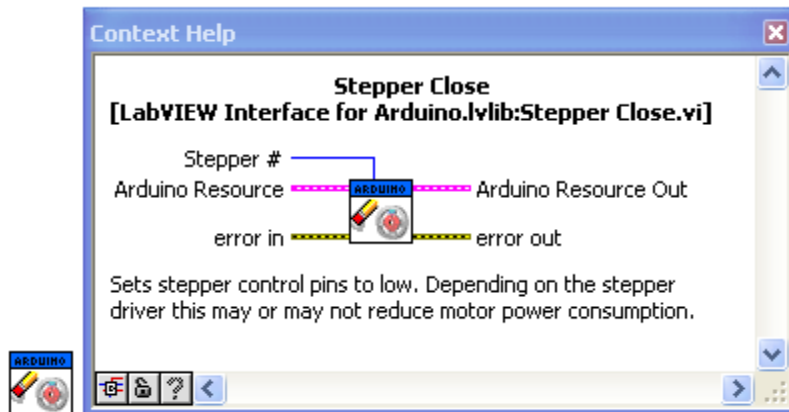
Escribe valores sobre un motor paso a paso: Escribe el número de pasos a girar y la velocidad de generación de pasos, así como el motor a activar



Confirmación de funcionamiento del motor paso a paso: Indica en su salida el número de pasos que faltan para alcanzar la consigna dada.



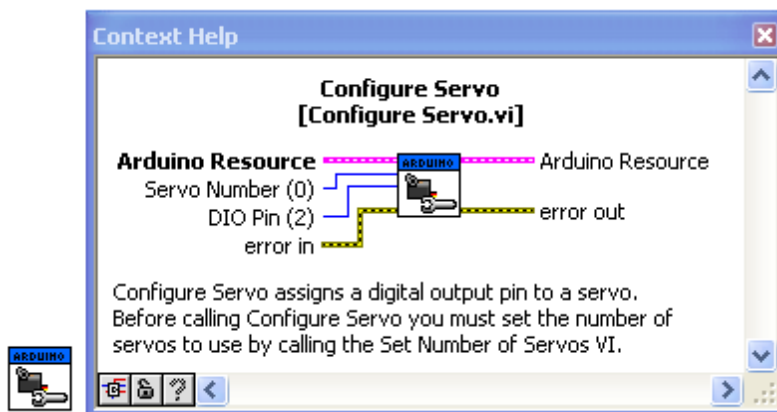
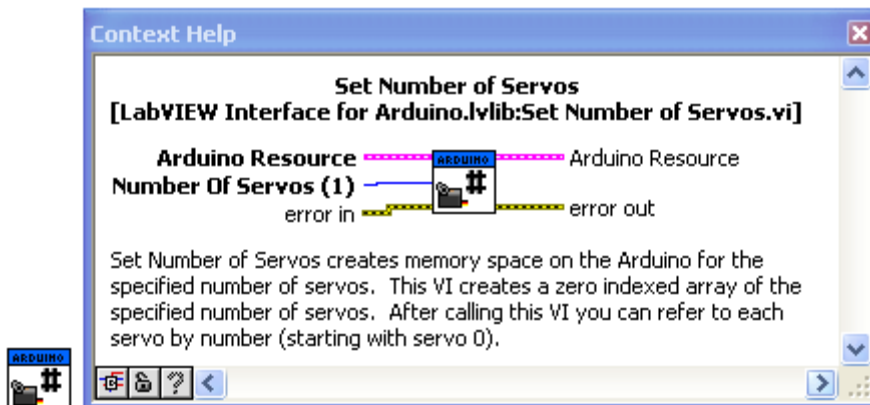
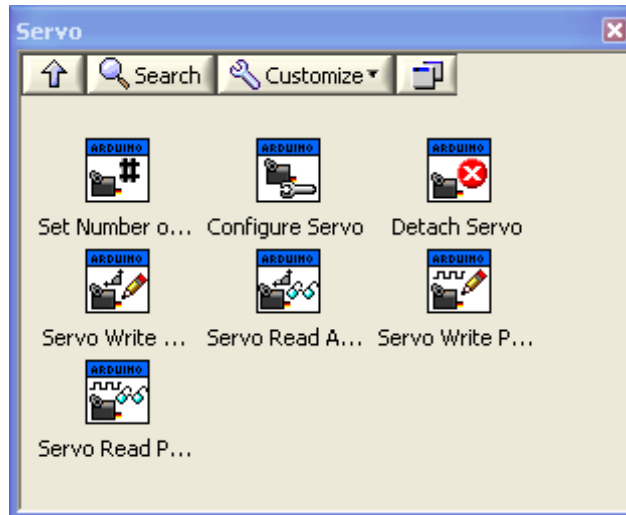
Espera la consecución de todos los pasos consignados para el giro



Cierra el gobierno del motor paso a paso



Librerías de Servos



Context Help

Detach Servo [Detach Servo.vi]

Arduino Resource — Arduino Resource
Servo Number — error in — error out

Detach Servo release the servo from its assigned pin allowing you to use the DIO pin for general purpose digital input and output.

Context Help

Servo Write Angle [Servo Write Angle.vi]

Arduino Resource — Arduino Resource
Servo Number —
Angle (Degrees) — error out
 error in

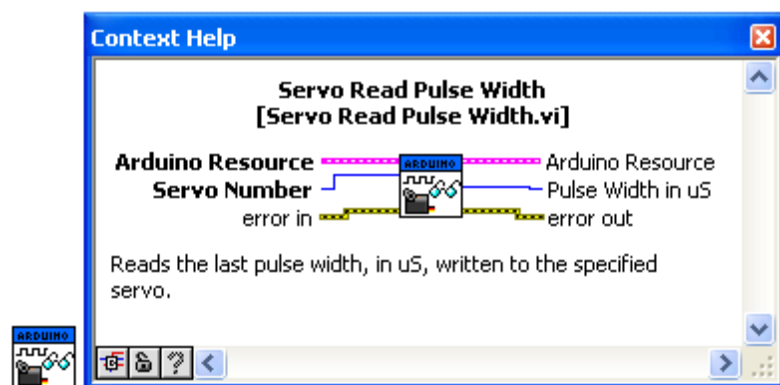
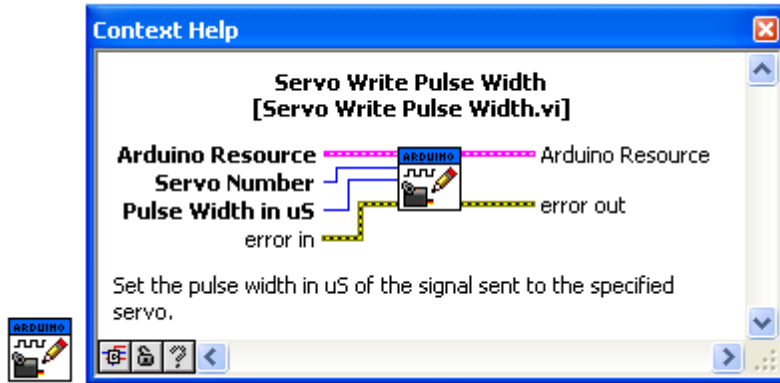
Set the specified servo to the specified angle in degrees.

Context Help

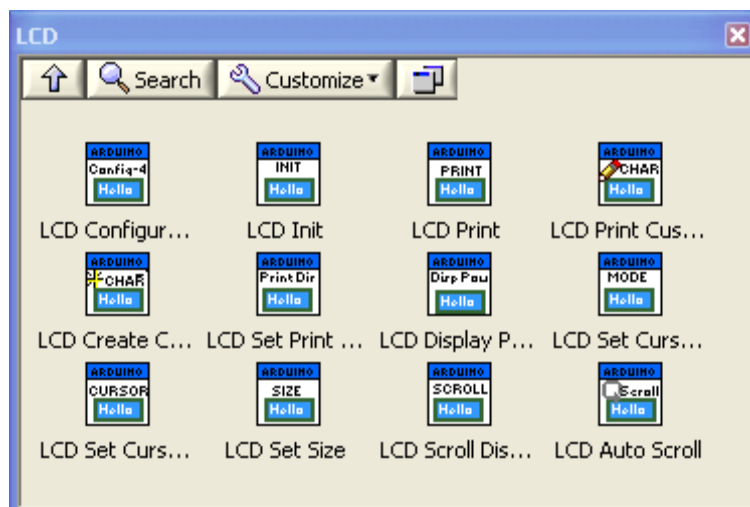
Servo Read Angle [LabVIEW Interface for Arduino.lvlib:Servo Read Angle.vi]

Arduino Resource — Arduino Resource
Servo Number — Angle (Degrees)
 error in — error out

Read the last angle in degrees written to the specified servo.

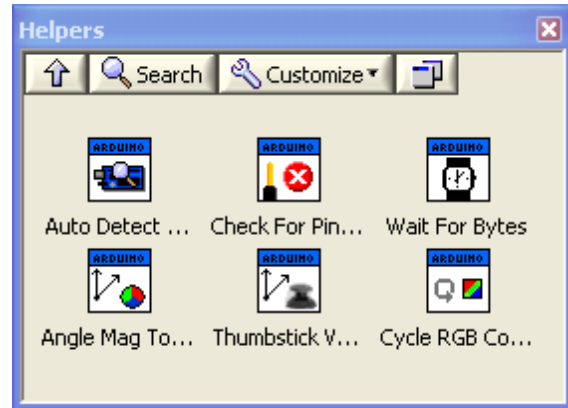
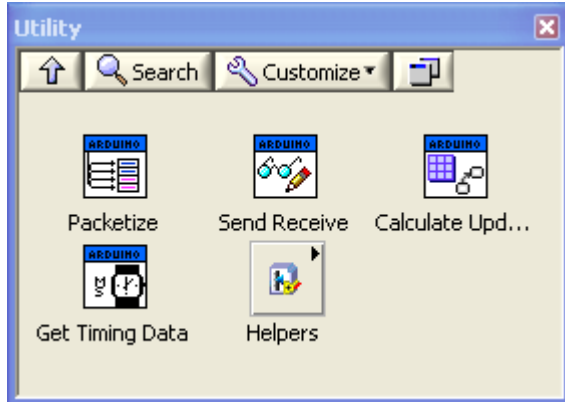


Librería LCD





Librería Utilidades



Ejemplos

