

LA SUITE PROTEUS

Primeros pasos con LA PESTAÑA DISEÑADOR GRÁFICO

labcenter Electronics
www.labcenter.com

Información sobre derechos y copyright

© Labcenter Electronics Ltd 1990-2019. All Rights Reserved.

Los programas de software PROTEUS (Proteus Capture, PROSPICE Simulation, Schematic Capture and PCB Layout y sus archivos de librerías asociadas, archivos de datos y toda la documentación son copyright de ©Labcenter Electronics Ltd. Todos los derechos reservados. Si usted ha comprado una licencia para usar el software en una máquina en un momento dado, usted no es dueño del software. La copia no autorizada, el préstamo y la redistribución del software o de la documentación realizada por cualquier medio, constituye una violación de los derechos de autor. La piratería de software es un delito.

PROSPICE incorpora el código fuente de Berkeley Spice3f5 que es copyright de © Regentes de la Universidad de Berkeley. Los modelos SPICE de los fabricantes incluidos con el software son propiedad de sus respectivos autores.

La herramienta Qt GUI es propiedad de © 2012 Digia Plc y/o sus subsidiarias y se usa bajo licencia LGPL versión 2.1. Algunos iconos son propiedad de 2010 The Eclipse Foundation utilizadas bajo la licencia publica versión 1.0 de Eclipse. Algunos ejecutables son utilidades en formato binario utilizadas bajo la licencia © 2010 The GNU Project, versión GPL 2.

AVISO
PROTEUS también contiene código especial que, mediante la utilización de la clave de licencia de usuario, evitará que se pueda utilizar más de una copia en una red al mismo tiempo. Por lo tanto, usted debe comprar una clave de licencia por cada copia que desea ejecutar de forma simultánea.

RENUNCIA
No se garantiza de ninguna manera el contenido de este paquete de software, ni se garantiza su idoneidad para cualquier propósito en particular. Ni Labcenter Electronics Ltd, ni ninguno de sus

empleados, subcontratistas o distribuidores serán responsables por los errores que puedan existir en el software, librerías de componentes, modelos de simulación o en la documentación de usuario. Ni tampoco por cualquier daño directo, indirecto o consecuente ni de las pérdidas financieras derivadas de la utilización del paquete.

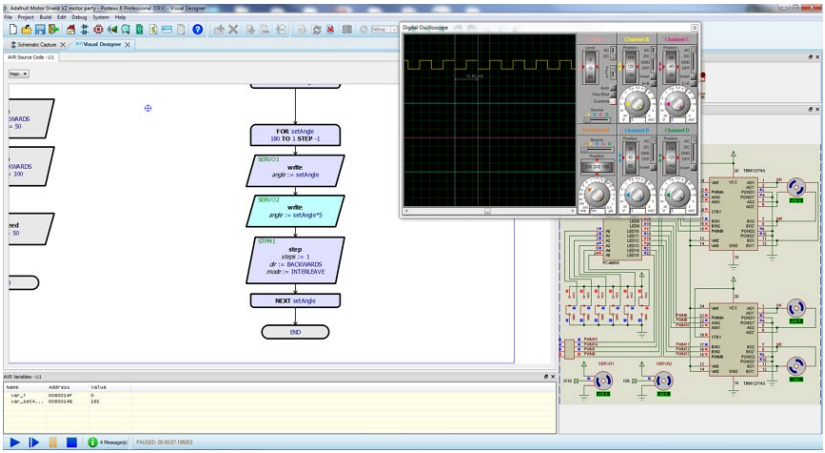
Se recuerda en particular a todos los usuarios que la simulación con éxito de un diseño utilizando el simulador PROSPICE no demuestra de manera concluyente que vaya a trabajar cuando se pasan los datos al proceso de fabricación. Siempre es conveniente hacer un prototipo de pruebas antes de mandar fabricar una tirada de cantidades grandes.

Los modelos SPICE de los fabricantes incluidos con PROSPICE se suministran de acuerdo con el concepto "tal y como son" y tampoco Labcenter ni sus autores ofrecen ningún tipo de garantía en cuanto a su precisión o su funcionalidad.

1.- Introducción.

1.1.-Objetivo.

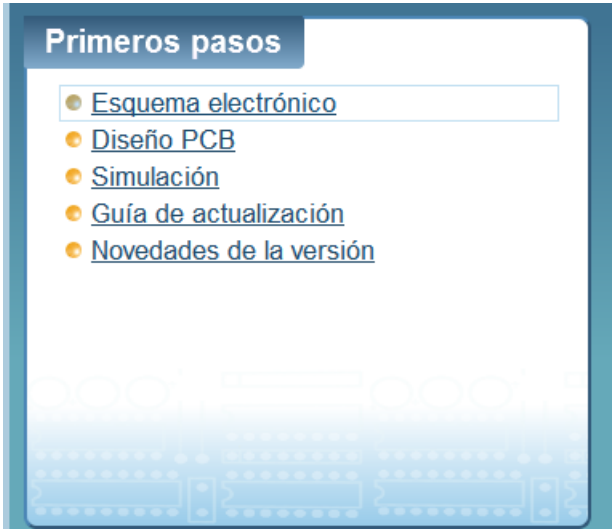
El DISEÑADOR GRÁFICO (VISUAL DESIGNER) es una novedosa herramienta de desarrollo que une la programación utilizando diagramas de flujo con los recortes de los esquemas electrónicos de las tarjetas de expansión (shields[1]) de Arduino para posibilitar la creación de soluciones completas para el mundo real utilizando los sistemas embebidos de Arduino.



El objetivo de esta guía es explicar las funcionalidades del Diseñador Gráfico utilizando varios ejemplos prácticos que sirvan de tutorial. Para seguir las prácticas propuestas en esta guía es necesario tener unos conocimientos previos de Proteus. Proteus se compone de diversos módulos cada uno de los cuales se encarga de una tarea específica: esquema del circuito electrónico, diseño del PCB, visor en 3D, lista de materiales, visor gerber, etc.

[1] N.T.:En inglés las tarjetas de expansión reciben el nombre de 'shields'. Este nombre se ha popularizado tanto en el mundo Arduino que no se ha traducido al español y también se utiliza cuando hablamos de las tarjetas de expansión en nuestro idioma. Sin embargo nosotros hemos preferido referirnos a ellas con el término tarjeta de expansión, que aunque más largo nos parece que tiene más significado y sentido que utilizar el anglicismo.

Para conocer en profundidad las posibilidades que nos brinda Proteus para simular el funcionamiento de nuestro diseño, se puede consultar las ayudas disponibles en la página de inicio en Proteus.



1.2.- ¿Qué es Arduino™ / genuino™?

Arduino es una empresa, un proyecto y una comunidad de usuarios que suministra equipos y software para el control industrial con la filosofía de código abierto. Proporcionan paquetes para la construcción de dispositivos digitales y periféricos interactivos que pueden gobernar y recibir información de objetos del mundo real.
El proyecto se basa en diseños de equipos de control basados en diferentes microcontroladores y que son comercializadas por diversas compañías. Estos sistemas ofrecen conjuntos de pines de entrada y salida (tanto digitales como analógicos), con los que se controlan varias tarjetas de expansión que contienen a su vez otros circuitos especializados.

Se puede encontrar más información sobre Arduino consultado su página web desde el siguiente enlace: https://www.arduino.cc/

El nombre y el logotipo de Arduino son marcas registradas. La herramienta Diseñador Gráfico (VISUAL DESIGNER) de Labcenter no está directamente relacionada ni avalada por Arduino.

1.3.- ¿Qué es Raspberry Pi?

Raspberry Pi es una serie de ordenadores pequeños y de bajo coste construidos utilizando una única tarjeta. Los equipos Raspberry Pi son un completo ordenador Linux pero normalmente se utilizan para llevar a cabo el control de un dispositivo electrónico.

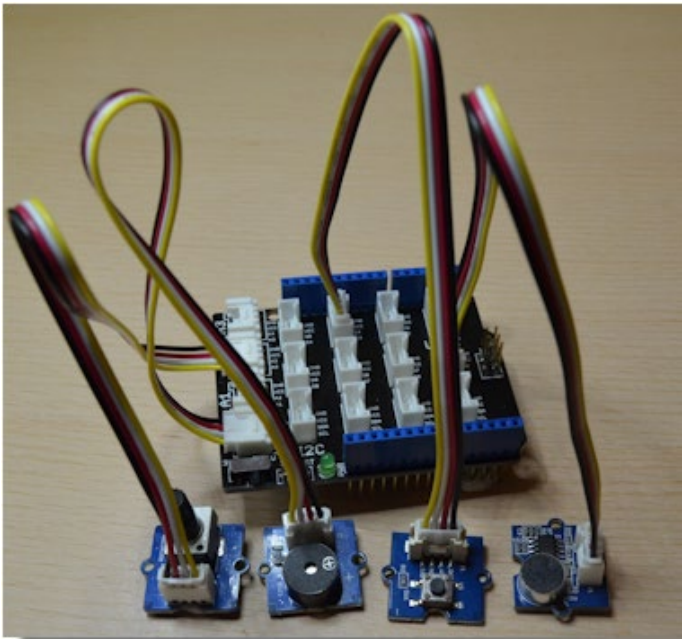
1.4.- ¿Qué es Grove?

Grove es un sistema de módulos electrónicos para su rápida utilización como prototipos dentro de nuestros diseños con Arduino. Cada módulo ejecuta una función diferente. Por ejemplo un sensor de temperatura, un generador de sonidos, un display LCD, etc.



Sólo es necesario conectar el módulo que realice la función que deseamos incorporar a nuestro sistema y ya está todo listo para empezar a probar nuestras ideas.

Los conjuntos ‘starter kit’ de Grove presentes en el mercado, son una fantástica forma de empezar para los estudiantes que se acercan por primera vez al mundo de Arduino.



Podemos conectar hasta 16 módulos Grove con las placas de cualquier microcontroladores Arduino sin ningún problema. El kit incluye una tarjeta de expansión para Arduino específico formado por múltiples conectores a los que resulta muy sencillo enchufar los módulos Grove con nuestra placa Arduino.

Podemos encontrar la información de todos estos kits en el siguiente enlace: http://www.seeedstudio.com/wiki/Grove_System.

1.5.-Limitaciones de la simulación de los equipos Raspberry Pi utilizando Proteus.

El modelo de simulación Raspberry Pi es del tipo modelo de simulación de comportamiento de alto nivel. Puesto que los equipos Raspberry Pi utilizan una copia completa del sistema operativo Linux y pueden ejecutar una enorme cantidad de código arbitrario, es imposible llevar a cabo un modelo de simulación completo al estilo, por ejemplo, del que se utiliza con los microprocesadores PIC.

En concreto la simulación de equipos Raspberry Pi realizada con Proteus soporta las siguientes librerías:

- Smbus
- Pygame
- Wiringpi
- Rpi.GPIO
- spidev.

Además hay un conjunto de librerías construidas utilizando Python que se suministran con Proteus y se encuentran en la carpeta de controladores de la familia Raspberry Pi:

- Adafruit_GPIO
- Adafruit_I2C
- Adafruit_MCP230xx
- Adafruit_MCP3008
- Adafruit_MotorHAT
- Adafruit_PCA9685
- Adafruit_PureIO
- Automationhat
- Explorerhat
- ADS1x15
- cap1xxx
- grove_128_64_oled
- grove_rgb_lcd
- grovepi
- max31855
- pcf8574
- pcf8591
- piglow
- sn3218

Si escribimos el código de nuestro proyecto utilizando el Diseñador Gráfico y los bloques de flujograma disponibles, la simulación del equipo Raspberry Pi trabajará perfectamente en el entorno Proteus y en los equipos reales sin problemas.

Si usamos la opción de convertir nuestro proyecto para poder programarlo utilizando código escrito directamente en el lenguaje Python, en estos casos será posible, por supuesto, que escribamos nuestro código libremente y que utilicemos otras librerías Python

diferentes a las suministradas por Proteus. Pero entonces podría suceder que no pudiéramos llevar a cabo la simulación en el entorno Proteus porque nuestro código puede sustentarse en el uso de librerías no soportadas por el motor de simulación.

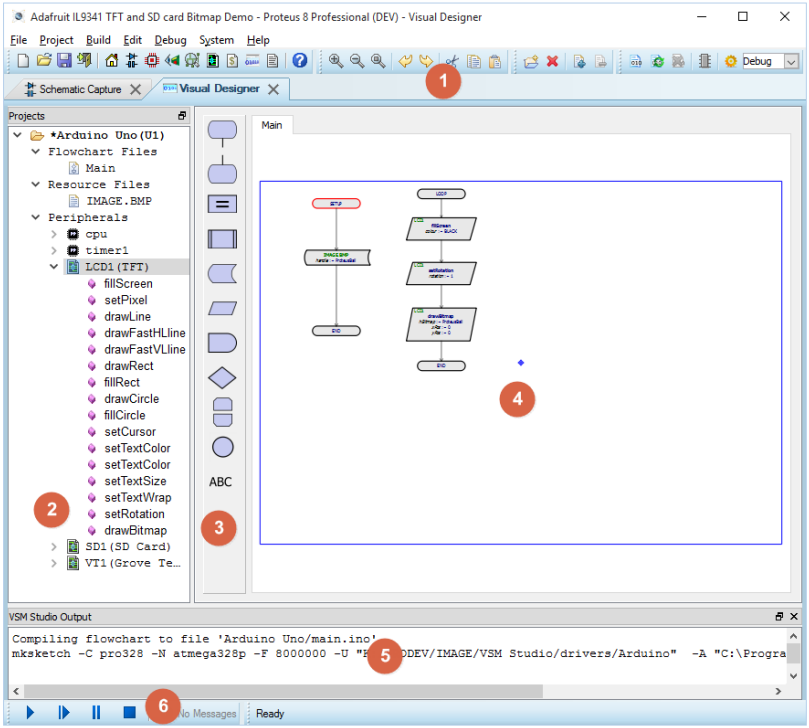
2.-El entorno de trabajo del Diseñador Gráfico.

El Diseñador Gráfico es una herramienta para el diseño de sistemas de control embebido mediante la utilización de una interfaz sencilla basada en un flujograma que nos permite simular y depurar el sistema completo utilizando únicamente herramientas de software. Para que este modelo funcione es necesario dotar al entorno de desarrollo de un lugar donde editar el código y otro donde realizar la depuración. En este apartado vamos a introducir al lector en ambos espacios de trabajo.

2.1.-El área de trabajo de diseño.

Durante el proceso de diseño, vamos a crear nuestro sistema de control embebido, añadiendo las tarjetas de expansión que lo componen y el código encargado de la lógica de control de los mismos. El entorno de trabajo del Diseñador Gráfico se divide en seis áreas.

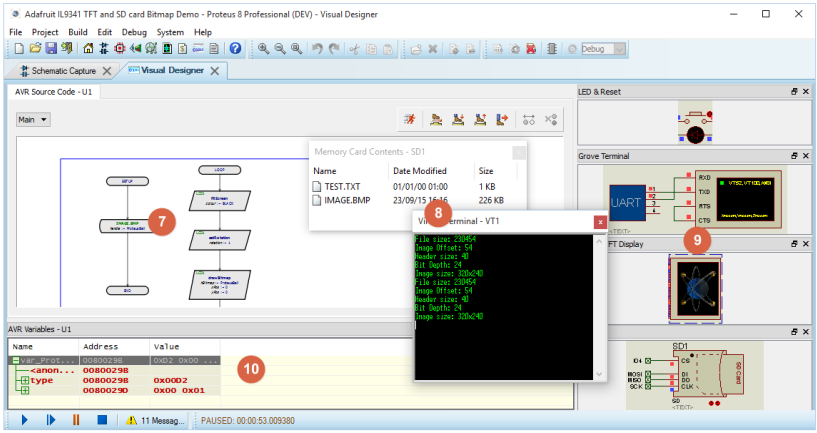
- Las barras de menús, iconos y pestañas habituales en el entorno Proteus (1).
- El árbol de estructura del proyecto (2).
- Los bloques tipo disponibles para utilizar en nuestro diagrama (3).
- La ventana de edición (4).
- La ventana de salida de resultados (5).
- El panel de controles de simulación y barra de estado (6).



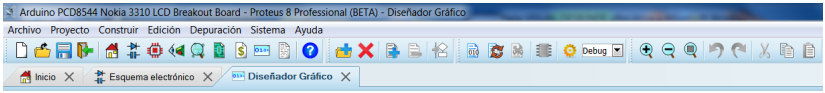
2.2.-El Área de trabajo de depuración.

Cuando arrancamos la simulación de nuestro proyecto el área de trabajo cambia y nos ofrece una serie de herramientas muy importantes y útiles para la depuración.

- Ventana con el código fuente (7).
- Ventanas emergentes activas (8).
- Ventanas emergentes para mostrar detalles de los periféricos (9).
- Ventanas de inspección de variables y posiciones de memoria (10).



2.3.-La barra de menús, la barra de herramientas y las pestañas.



La barra de menús y la barra de herramientas situadas en la zona superior resultarán familiares a todos los usuarios de Windows y especialmente a los usuarios de Proteus, puesto que es común a todos los módulos. A lo largo de esta guía se irán viendo con detalle aquellos que son más relevantes. Además se pueden consultar las

otras guías mencionadas especializadas en otros módulos de Proteus para encontrar más información.

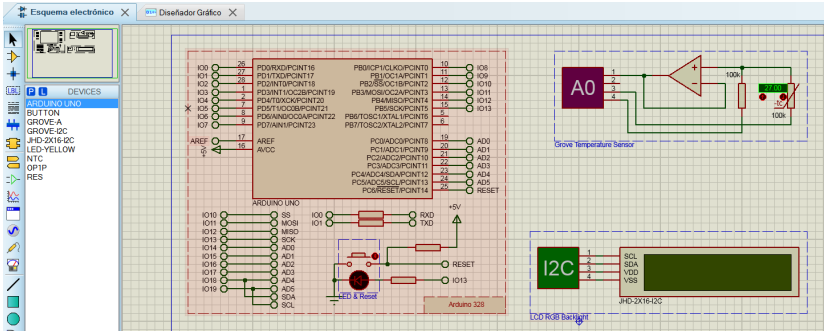
ICON	DESCRIPTION	ICON	DESCRIPTION	ICON	DESCRIPTION
	New Proteus Project		Open VSM Studio / Visual Designer		Create New Firmware Project
	Open Project		Open Project Notes		Delete Existing Firmware Project
	Save Project		Open the Help files		Add Source Files to the Existing Project
	Close Project		Zoom In		Remove Source Files to the Existing Project
	Open Proteus Home Page		Zoom Out		Build / Compile Project
	Open Schematic Capture Program		Zoom All		Rebuild / Compile Project
	Open the PCB Layout Program		Undo		Stop Build
	Open the 3D Viewer		Redo		Attached to IDE / Hardware
	Open the Gerber Editor		Cut		Project Settings
	Open the Design Explorer		Copy	Debug	Type of Build
	Open the Bill of Materials		Paste		

Las pestañas situadas debajo de la barra de herramientas, nos permiten saltar de un módulo a otro del entorno de trabajo Proteus de forma rápida y sencilla. El trabajo que vamos a realizar a lo largo de esta ayuda se desarrollará la mayor parte del tiempo utilizando la pestaña Diseñador Gráfico. Pero, puesto que vamos a construir un sistema de control embebido, eso supone que tenemos que crear nuestro diseño electrónico en la pestaña 'Esquema electrónico'.

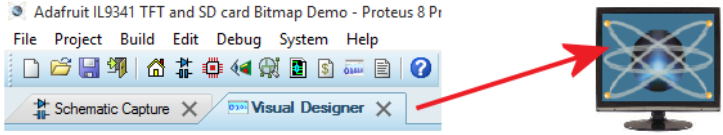
Proteus se encarga de añadir de forma automática los objetos gráficos necesarios (los componentes electrónicos que forman nuestro equipo Arduino y sus tarjetas de expansión) en el esquema electrónico cuando añadamos las tarjetas de expansión y/o periféricos desde la pestaña 'Diseñador Gráfico'.

El funcionamiento es similar al que se produce cuando añadimos un componente electrónico y automáticamente se crea una entrada en la pestaña 'lista de materiales' o una representación 3D en la pestaña 'vista 3D'.

En cualquier momento podemos ir a la pestaña ‘Esquema electrónico’ y observar los ‘equipos virtuales’ que se han ido creando al completar los diferentes pasos en la construcción del código de nuestro proyecto en la pestaña ‘Diseñador gráfico’.

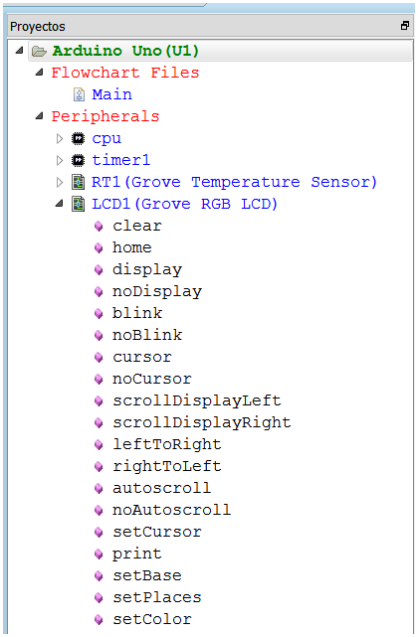


Proteus nos permite utilizar más de un monitor al mismo tiempo para poder mostrar más de una pestaña distinta de forma simultánea en varios terminales.



La posibilidad de disponer de ‘equipos virtuales’ dentro de nuestro diseño electrónico muestra la enorme potencia de los sistemas de simulación del tipo Proteus, que nos permiten colocar sondas de medida en cualquier lugar. Sólo tenemos que colocar y conectar cualquiera de los múltiples dispositivos virtuales disponibles (osciloscopio, analizadores lógicos, voltímetros, etc.) para disponer de un completo laboratorio de electrónica. A lo largo de esta guía se mostrarán múltiples ejemplos.

2.4.-El árbol de estructura del proyecto.



El árbol de estructura del proyecto nos permite llevar a cabo tres operaciones fundamentales:

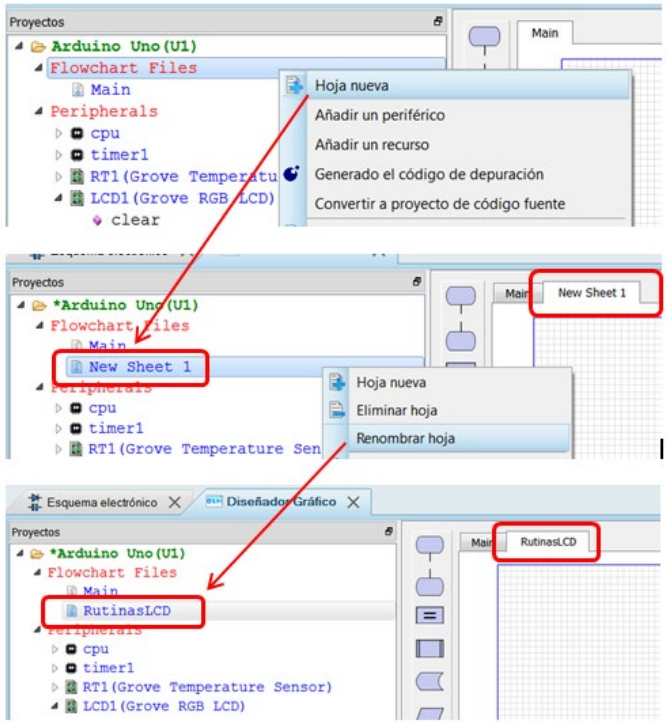
- Controlar las distintas hojas que contienen el flujograma.
- Controlar los diferentes recursos utilizados en nuestro sistema de control embebido.
- Controlar los periféricos utilizados en nuestro proyecto que conectados con nuestra placa Arduino.

2.4.1-.Control de las hojas del flujograma.

Cuando empezamos a trabajar con un nuevo proyecto por defecto sólo tenemos una hoja con nuestro flujograma en la ventana de edición llamada ‘Main’. Podemos añadir tantas hojas más como deseemos. Por ejemplo, para situar en ella las rutinas que usemos

en nuestro proyecto cuando éstas crecen y hacen que el código se extienda en tamaño de forma considerable.

Cada hoja del flujograma se muestra en una pestaña diferente.

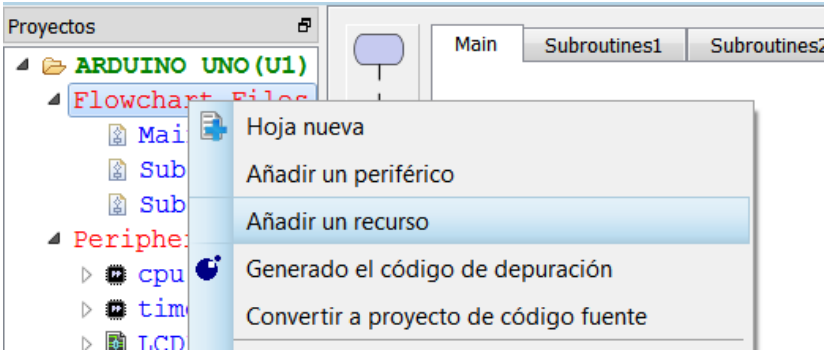


Cuando nuestro proyecto contiene más de una hoja podemos movernos entre ellas de haciendo doble click en el nombre de la hoja a la que queremos desplazarnos situada en el árbol de las estructura de proyecto. De todas formas, la forma más sencilla para desplazarnos entre hojas es utilizar las pestañas para ir de una hoja a otra.

2.4.2.-Control de recursos.

El control de recursos es una técnica muy potente para incluir ficheros con imágenes o audios dentro de nuestro proyecto.

Podemos añadir o eliminar un recurso utilizando el menú contextual del árbol de estructura de proyecto.

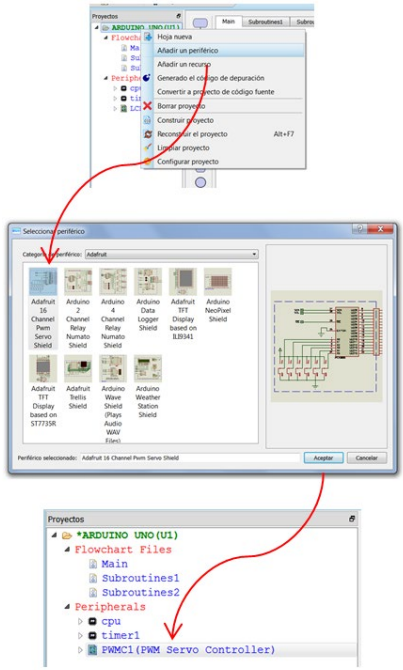


Desde el momento en que dispongamos de un recurso seremos capaces de 'cogerlo y arrastrarlo' dentro de la función setup() de nuestro código para asignarle un gestor y poder luego utilizarlo en cualquier parte de nuestro programa.

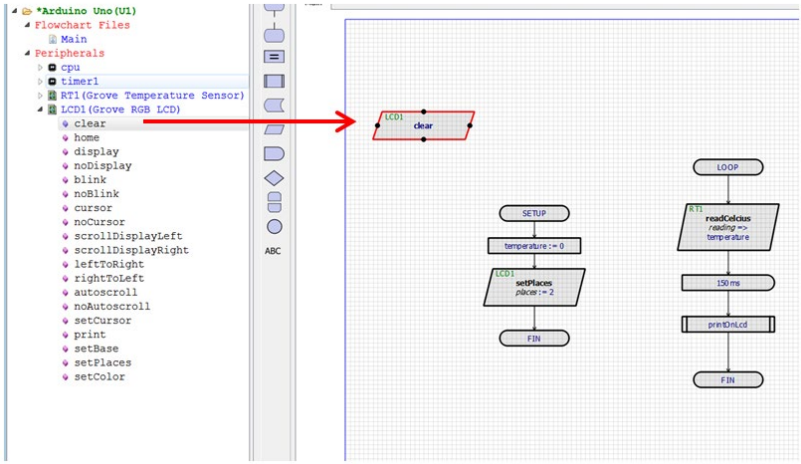
⚠ Conviene ser cuidados con los nombres de los ficheros que utilicemos. La gestión de las tarjetas SD en Arduino sólo soporta nombres en formato 8.3 de acuerdo con el estándar FAT16. Esto significa que los nombres no deben superar los 8 caracteres y que las extensiones deben ser de 3 o menos. Por ejemplo una imagen llamada HuborLogo.jpg no podrá funcionará correctamente y deberá ser cambiado a un nombre del tipo logo.jpg o hubor.jpg para que funcione adecuadamente.

2.4.3.-Control de los periféricos.

El diseñador gráfico es un entorno de desarrollo para un sistema de control embebido completo. Por este motivo, incluye la gestión de bloques de periféricos situados en la misma tarjeta principal de Arduino y bloques de periféricos externos (tipo las tarjetas de expansión o los sensores Grove). Cuando comenzamos con un nuevo proyecto sólo encontraremos dos o tres periféricos. Normalmente, la CPU y un TIMER1 asociados ambos a la placa base de Arduino. A partir de ahí, podemos ir añadiendo periféricos utilizando el menú contextual desde el árbol de estructura del proyecto o utilizando las opciones del menú.

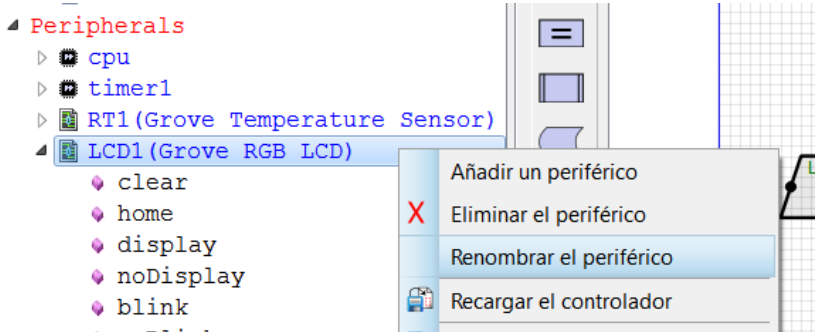


Una vez que hayamos añadido un periférico nuevo, podremos ver todos los métodos disponibles para gestionarlo. Un método es la técnica pone a nuestra disposición Proteus en el Diseñador Gráfico para manejar el hardware. Sólo tenemos que coger y arrastrar el método deseado dentro de nuestro flujograma en el lugar donde queramos situarlo para que se cree un bloque del tipo ‘operación con periférico’.



Es posible renombrar un periférico después de que lo hayamos añadido a nuestro proyecto. Esta opción puede resultar especialmente útil cuando, por ejemplo, tenemos varios módulos de pulsador o de leds y queramos utilizar un nombre que describa su función para facilitarnos su identificación posterior.

Sólo tenemos que usar el botón derecho del ratón sobre el periférico y seleccionar la opción del menú contextuar para renombrarlo.

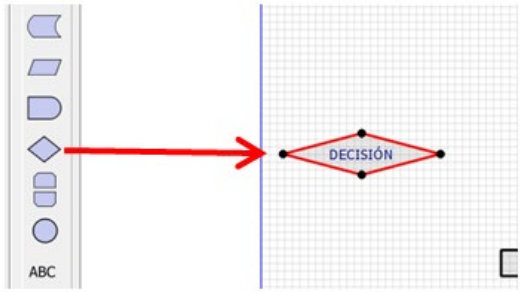


⚠ Al asignar nombre a los periféricos no debemos utilizar espacios ni caracteres específicos de un idioma diferente al inglés (ñ, acentos, etc.).

2.5.-Los bloques del flujograma.

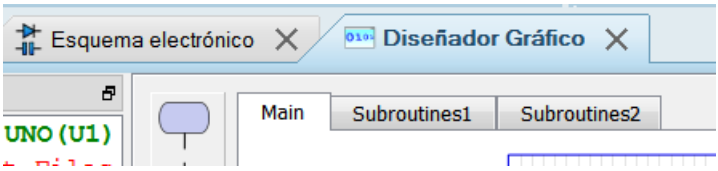
Los bloques de flujograma son los ladrillos que utilizamos para construir el código de nuestro programa. En muchos casos, nuestro trabajo consistirá en coger y soltar los bloques desde el árbol de estructura del proyecto hasta la ventana de edición.

Sin embargo, algunos tipos de objeto, como los bloques de aplazamiento temporal, los bloques de bucle y los bloques de llamada a, sólo se pueden utilizar desde las herramientas situadas en la columna de los bloques de flujograma.



2.6.-La ventana de edición.

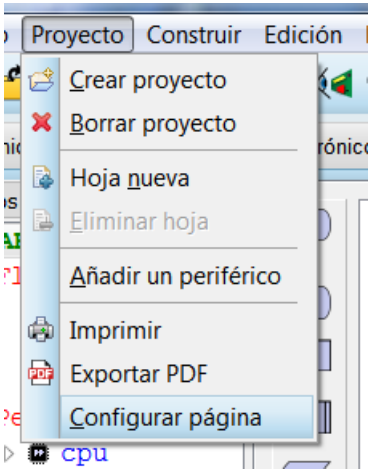
La ventana de edición es el lugar donde colocamos nuestro flujograma para crear el código de nuestro proyecto. Si el proyecto contiene múltiples hojas de flujograma, la ventana de edición mostrará la que esté actualmente activa. Con las pestañas situadas en la zona superior de la ventana de edición podemos pasar de una hoja a otra con facilidad.

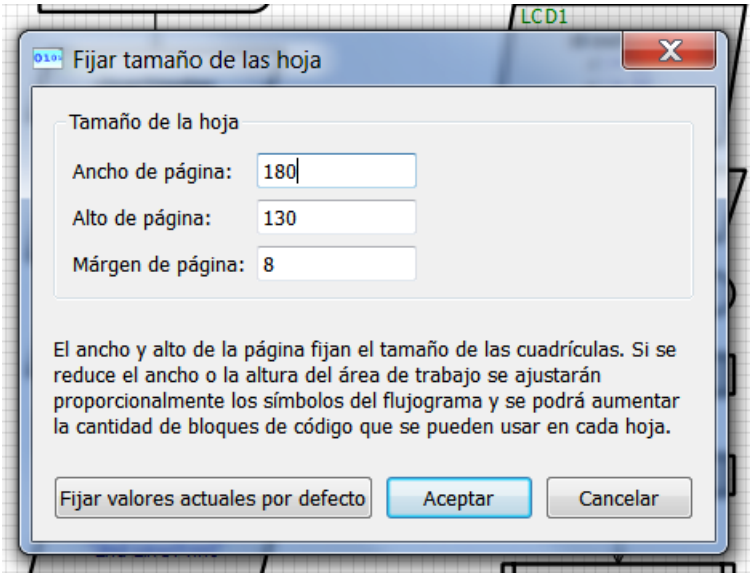


El tamaño de la hoja de trabajo la definimos indicando el número de celdas que componen la rejilla sobre la que se ubican los diferentes bloques del flujograma. Cada bloque que colocamos en nuestro

flujograma ocupa siempre un número fijo prefijado de celdas. Por lo tanto, cuanto mayor sea el número de celdas que componen la rejilla, podremos ubicar más bloques en una misma hoja, pero tendrán un tamaño más pequeño. Y cuanto menor sea el número de celdas que componen la rejilla, podremos colocar menos bloques un una misma hoja, pero tendrán un tamaño más grande.

Desde la opción 'Configurar página' del menú 'Proyecto' se puede fijar el número de celdas que componen la rejilla. De esta manera, podemos configurar a nuestro gusto el tamaño de los bloques. Si deseamos utilizar bloques más grandes porque nuestro ordenador utiliza una pantalla pequeña, podemos reducir el número de celdas de la hoja de trabajo.





Cuando ordenamos la salida por la impresora de nuestro flujograma, los gráficos se ajustarán automáticamente al tamaño de página definida desde la configuración de la impresora.

Es posible hacer un zoom para ampliar o disminuir un área de la ventana de edición usando los atajos de teclado F6 y F7 o la rueda del ratón. El atajo de teclado F8 fija el tamaño para encajar todo el contenido den nuestra hoja de trabajo en la pantalla.

2.7.-La ventana de salida de resultados.

La ventana de salida de resultados nos permite volcar la información del resultado de la operación de compilado de nuestro código y consultar el listado de errores, si éstos se producen.

> Si reducimos la ventana de salida de resultados, la ventana de edición será un poco más grande y podremos ver más cómodamente nuestro flujograma. Incluso podemos cerrarla del todo con el botón que contiene una 'x' situado en la zona superior derecha. La ventana volverá a abrirse de forma automática cuando sea necesario porque se generen nuevos

mensajes, informativo o de error, al poner en marcha el proceso de compilación.

2.8.-El panel de controles de simulación.

La simulación de nuestro proyecto se gobierna con el panel de controles, que funciona muy similar al de cualquier mando de control a distancia. El panel de control está situado en la zona inferior de la ventana de la aplicación.



Se compone de cuatro botones que son utilizados para controlar la ejecución de la simulación.

El botón de PLAY arranca la simulación.

El botón STEP avanza la simulación un paso. Si mantenemos el botón pulsado, la simulación avanza hasta que dejemos de presionarlo. El tamaño del tiempo de duración de un salto se puede configurar desde la ventana de configuración de la animación.

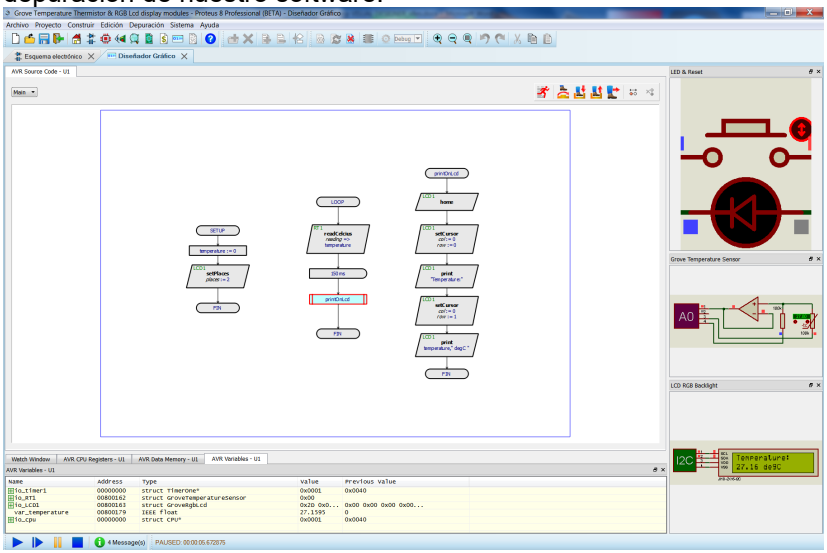
Configurar el tamaño del salto es útil para poder realizar simulaciones de circuitos en modo 'cámara lenta' y poder observar con más detenimiento que está sucediendo en nuestro diseño en cada momento.

El botón PAUSE suspende la simulación. Una nueva presión sobre el botón la pone en marcha de nuevo. Cuando la simulación está en pausa, puede avanzarse un paso utilizando la tecla STEP. La simulación también se coloca en estado de pausa cuando se encuentre con un punto de ruptura durante la ejecución del código.

El botón STOP detiene la simulación. Toda la información de la simulación se descarga de la memoria y todos los indicadores son reiniciados a las condiciones iniciales de funcionamiento, salvo los actuadores (pulsadores, interruptores, etc.) que se mantienen en el último estado utilizado.

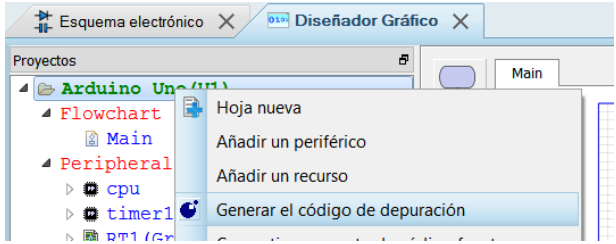
2.9.- La ventana del código fuente.

La ventana del código fuente es la herramienta principal que tenemos para depurar el código de los programas de nuestros proyectos. Cuando se está ejecutando la simulación, con sólo pulsar sobre el botón STEP entramos en este avanzado entorno de depuración de nuestro software.



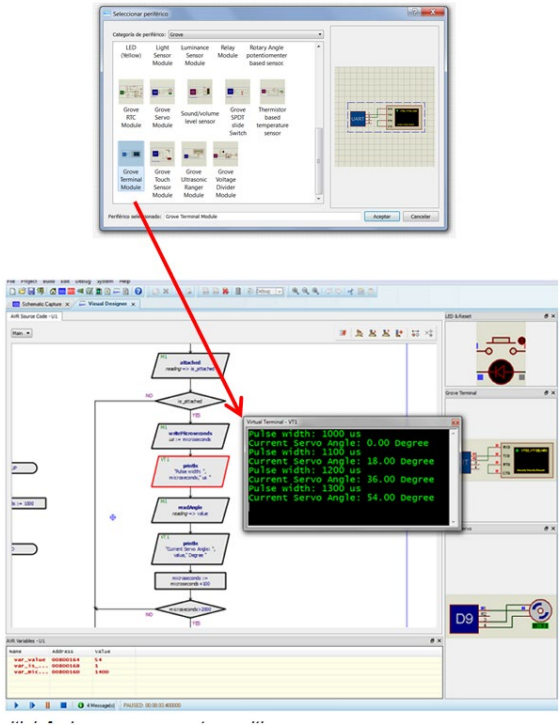
Cuando ponemos la simulación en pausa, la ventana del código fuente mostrará el flujograma con el bloque actualmente en ejecución destacado de color rojo.

También es posible seleccionar que deseamos utilizar para la depuración el código fuente en lugar del flujograma. Para hacerlo debemos seleccionar previamente la opción 'Generar el código de depuración' desde el menú contextual del árbol de estructura del proyecto.



2.10.-Las ventanas emergentes de periféricos.

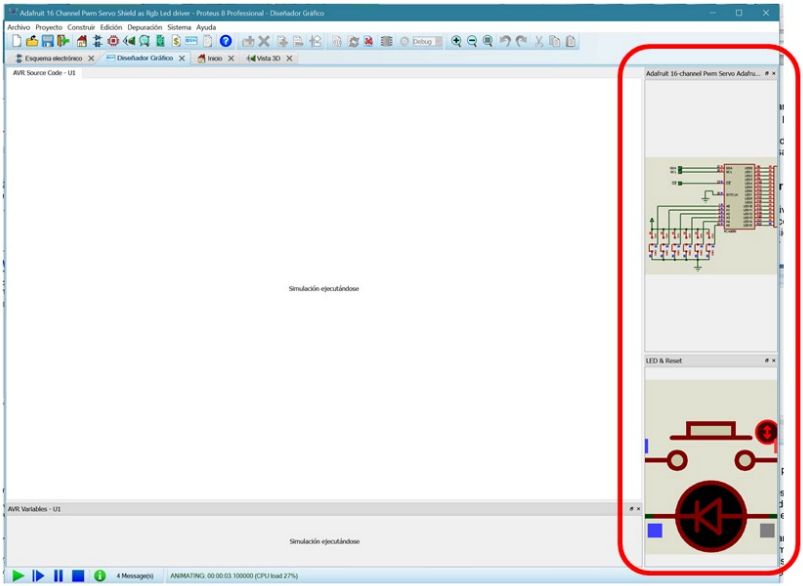
Algunos de los periféricos que podemos utilizar en nuestros proyectos tienen la opción de presentar una ventana emergente durante la simulación. Por ejemplo, si añadimos un módulo del tipo Terminal Grove podemos utilizarlo como consola y los mensajes que intercambiemos con ella se visualizarán en la ventana emergente que se abre durante la simulación.



Lo habitual es que las ventanas emergentes de periféricos estén visibles durante la simulación para que el usuario pueda interactuar con ellas. Si nos conviene, pueden ser cerradas en cualquier momento y reabiertas más tarde desde el menú Depuración siempre que la simulación esté en pausa.

2.11.-Las ventanas emergentes activas.

Las ventanas emergentes activas son áreas que muestran recortes de nuestro esquema electrónico que son especialmente interesantes durante el proceso de simulación. Aparecen en mosaicos en la zona derecha del módulo Diseñador Gráfico.



Se utilizan para dos funciones principales:

1. Mostrar elementos de especial relevancia dentro de nuestro diseño durante la fase de depuración del software. Por ejemplo un display LCD donde se muestran textos.
2. Permitir interactuar durante la fase de depuración del sistema con elementos de especial relevancia. Por

ejemplo, un pulsador o ajustar el valor de un determinado sensor.

La mayor ventaja que nos proporciona su utilización es que nos evitan tener que estar constantemente pasando de la pestaña donde se muestra el diseño electrónico y la pestaña donde se lleva a cabo la depuración del código. De esta manera podemos tener en una sola pantalla toda la información relevante para la depuración.

> Siempre es posible, si el usuario lo prefiere, trabajar con las dos pestañas y pasar de una a otra cuando sea necesario. Incluso es posible mostrar cada pestaña en una pantalla diferente si nuestra estación de trabajo cuenta con varios monitores.

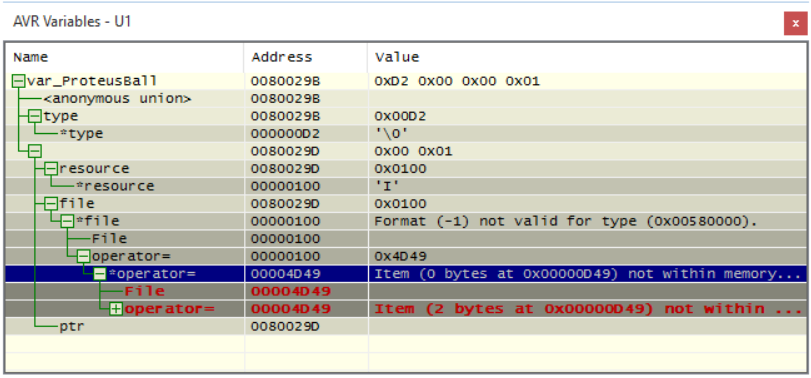
2.12.-Las ventanas de inspección.

Las ventanas de inspección son una herramienta de depuración que nos permite mostrar el valor de las variables utilizadas en nuestro código durante la sesión de depuración. Incluye potentes características que nos ayudan en la tarea de depurar de errores nuestro código.

2.12.1.-Expansión de elementos compuestos.

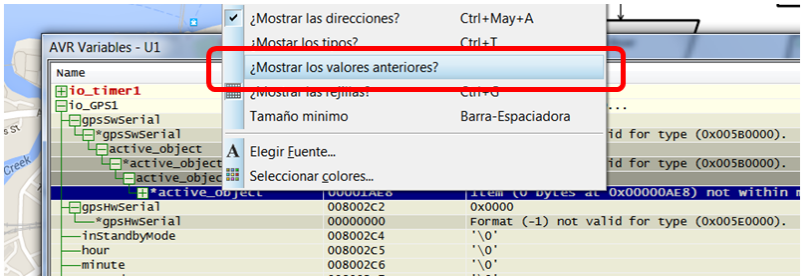
La ventana de inspección nos permite un manejo potente y sencillo de los elementos de programación compuestos: estructuras, matrices y enumeraciones.

También nos facilita el trabajo con punteros en nuestro código ya que es capaz de mostrar en forma de árbol los contenidos (valores) actuales de cada elemento y la dirección de memoria dónde se ubican.



2.12.2.-Mostrar los valores cambiantes.

Cualquier variable que sufra una modificación queda resaltada en la ventana de inspección de un color diferente cuando la simulación se coloca en pausa. Además nos permite ver el valor anterior que se almacenaba en ella, seleccionando la opción '¿Mostrar los valores anteriores?' desde el menú contextual. Para que aparezca el menú contextual, como siempre, hay que utilizar el botón derecho del ratón sobre la superficie de la ventana de inspección.

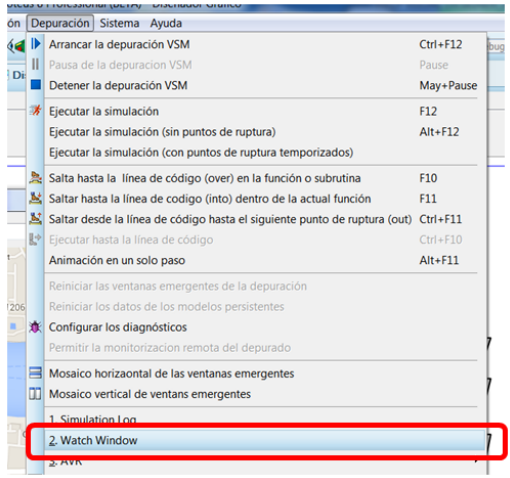


Si usamos un elemento compuesto en nuestro código (una estructura, matriz o enumeración) y cambia sólo uno de los elementos que la forman, sólo se resaltará dicho elemento en la ventana de inspección. Esto significa que, al menos que dicho elemento esté expansionado en el árbol de variables, no seremos capaces de ver resaltados los cambios que se hayan efectuado en dicho elemento.

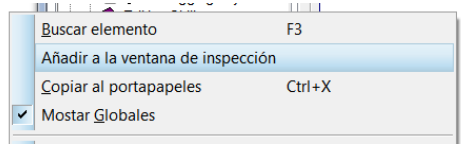
2.12.3.-Las ventanas de inspección emergentes.

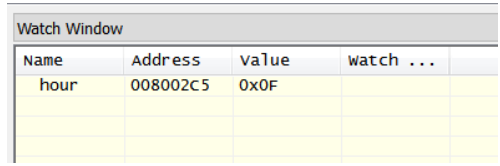
La ventana de inspección dónde se muestra el valor de las variables no es visible durante la ejecución de la simulación, salvo que la pongamos en pausa.

Si queremos visualizar el valor de determinadas variables durante la ejecución de la simulación podemos utilizar las ventanas de inspección emergentes. Se puede mostrar la ventana de inspección emergente desde el menú depuración.



Y elegir las variables que se monitorizarán en ella con ayuda del menú contextual de cada variable en la propia ventana.



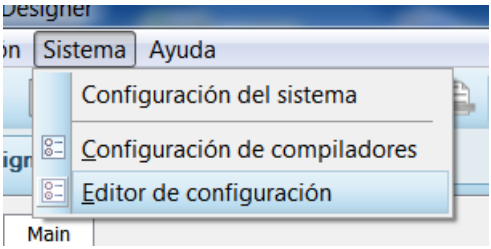


⚠ Si pegamos una variable en la ventana de inspección, volvemos a compilar el código y ponemos de nuevo la simulación en marcha, no podemos estar seguros de que el valor mostrado para esa variable en la ventana de inspección sea el correcto. El resultado dependerá de que el compilador haya asignado o no la misma posición de memoria a esa variable. Hay que tener en cuenta que la ventana de inspección siempre monitoriza direcciones de memoria y no las variables propiamente dichas. Hay que ser cuidadoso al utilizar esta herramienta para no obtener datos erróneos.

ℹ Además de visualizar los valores de las variables y de utilizar la ventana de inspección, Proteus pone a nuestra disposición otras potentes herramientas de depuración que nos facilitan mucha información detalla del estado del microcontrolador. Estas herramientas adicionales se activan todas desde el menú 'Depuración' siempre que la simulación se encuentre en pausa. Su uso está perfectamente documentado en los ficheros de ayuda de Proteus.

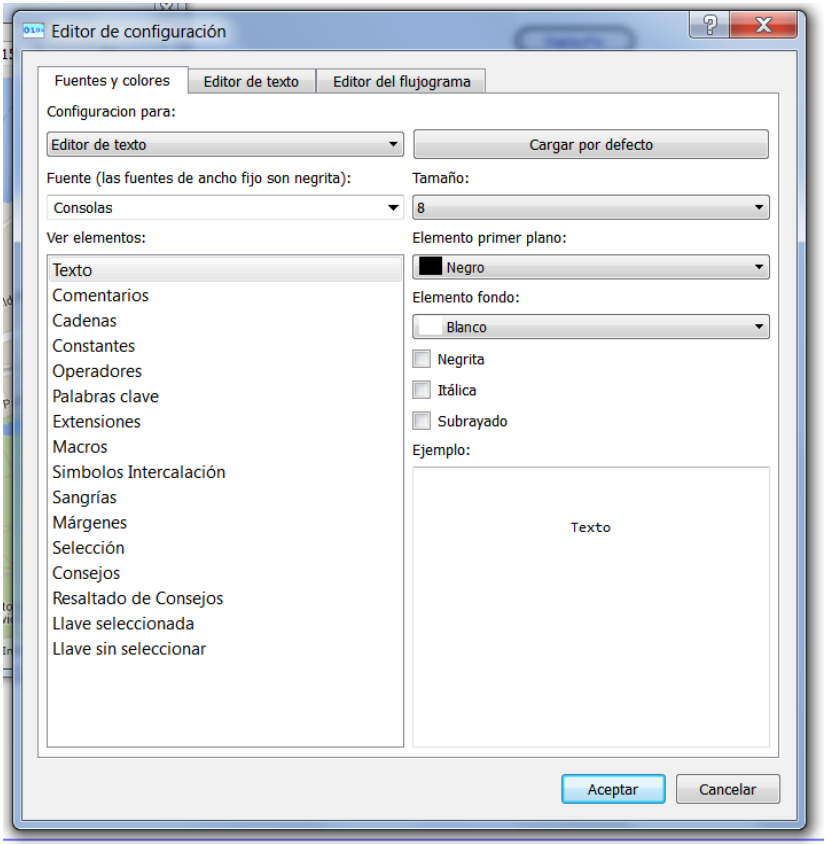
2.13.-Configuración del aspecto del diseñador gráfico.

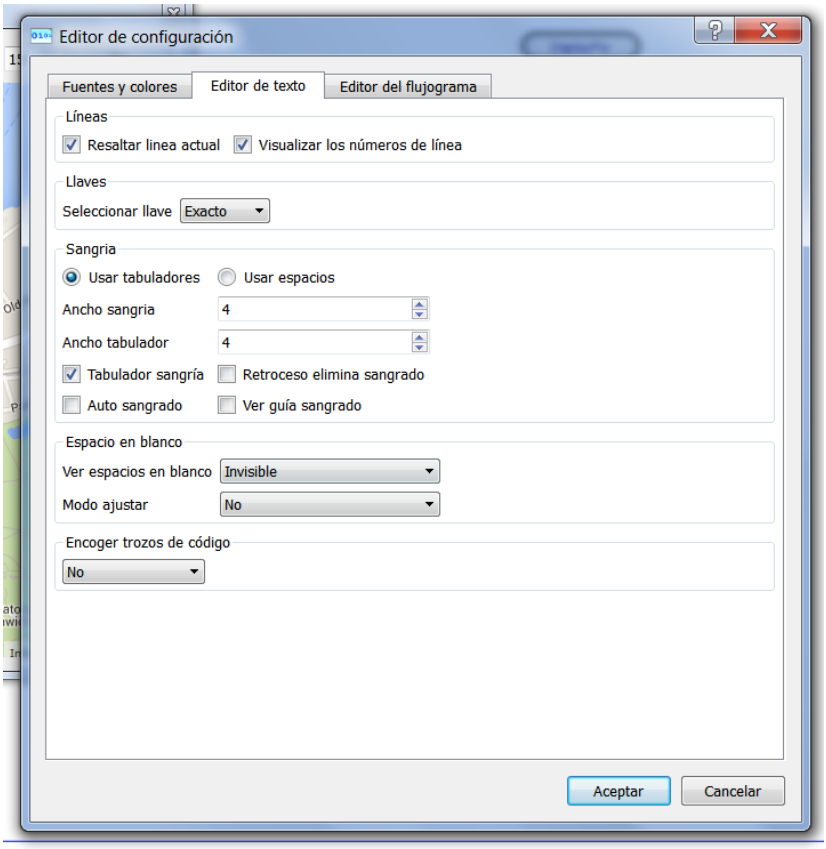
Podemos configurar el aspecto del diseñador gráfico utilizando la opción Editor de configuración del menú Sistema.

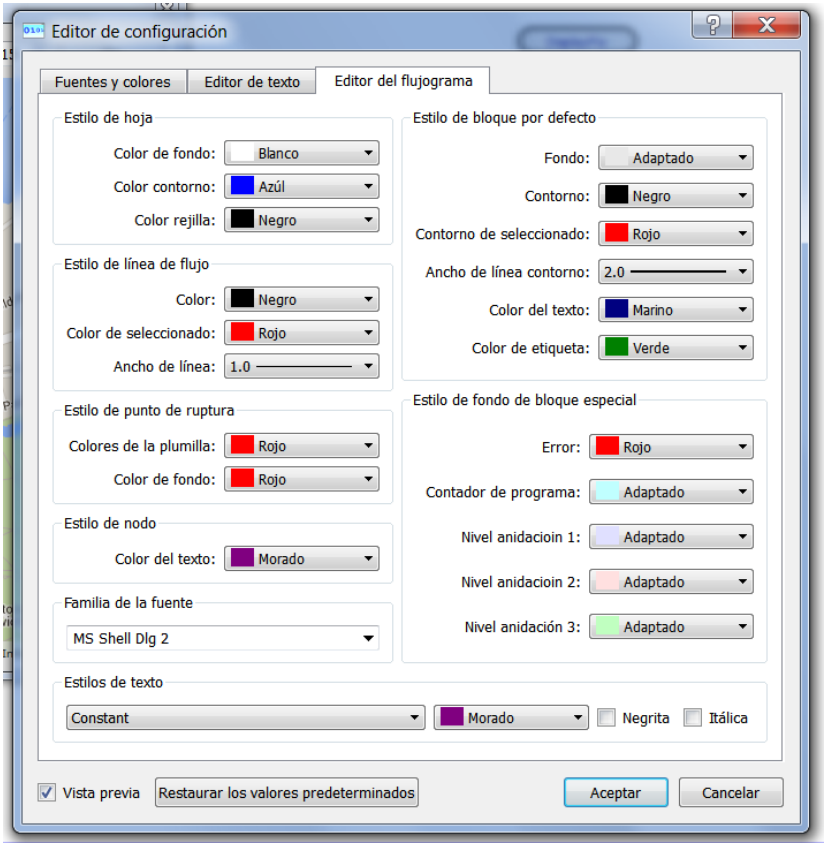


En la ventana de diálogo que se abre, disponemos de tres pestañas desde las que podemos configurar a nuestro gusto las fuentes y

colores, el comportamiento del editor de texto y los colores que se usarán para el flujograma.




Editor de configuración
Fuentes y colores
Editor de texto
Editor del flujograma
Líneas
Resaltar linea actual
Visualizar los números de línea
Llaves
Seleccionar llave
Exacto
Sangria
Usar tabuladores
Usar espacios
Ancho sangria
4
Ancho tabulador
4
Tabulador sangría
Retroceso elimina sangrado
Auto sangrado
Ver guía sangrado
Espacio en blanco
Ver espacios en blanco
Invisible
Modo ajustar
No
Encoger trozos de código
No
Aceptar
Cancelar


Editor de configuración
Fuentes y colores
Editor de texto
Editor del flujograma
Estilo de hoja
Color de fondo:
Blanco
Color contorno:
Azúl
Color rejilla:
Negro
Estilo de línea de flujo
Color:
Negro
Color de seleccionado:
Rojo
Ancho de línea:
1.0
Estilo de punto de ruptura
Colores de la plumilla:
Rojo
Color de fondo:
Rojo
Estilo de nodo
Color del texto:
Morado
Familia de la fuente
MS Shell Dlg 2
Estilo de bloque por defecto
Fondo:
Adaptado
Contorno:
Negro
Contorno de seleccionado:
Rojo
Ancho de línea contorno:
2.0
Color del texto:
Marino
Color de etiqueta:
Verde
Estilo de fondo de bloque especial
Error:
Rojo
Contador de programa:
Adaptado
Nivel anidacioin 1:
Adaptado
Nivel anidacioin 2:
Adaptado
Nivel anidación 3:
Adaptado
Estilos de texto
Constant
Morado
Negrita
Itálica
Vista previa
Restaurar los valores predeterminados
Aceptar
Cancelar

3.-Las técnicas de edición de código.

En este apartado vamos a aprender a utilizar el editor de diagramas de flujo. Nos centraremos en aprender las técnicas necesarias para construir nuestro código y crear el programa que se ejecutará en nuestro microprocesador.

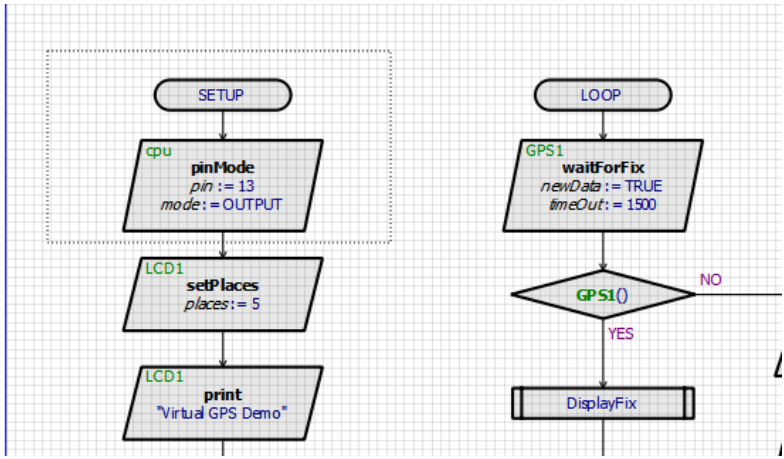
La simulación y depuración del código se explican en otros apartados de esta misma ayuda. Forzosamente tendremos que ir intercalando algunos de los conceptos de aquellas técnicas en el desarrollo de este capítulo. En estos casos, será interesar consultar los apartados correspondientes si el lector necesita una explicación más detallada de los mismos.

3.1.-Navegación.

3.1.1.-Utilización del zoom en la ventana de edición.

Para hacer zoom en la ventana de edición podemos utilizar:

- La rueda del ratón hacia adelante o hacia atrás. El origen del zoom se situará en la actual posición del cursor.
- Los iconos de la barra de herramientas. El origen del zoom se situara en el centro de la ventana de edición.
- Los atajos de teclado F6 y F7. El origen del zoom se situará en la actual posición del cursor. El atajo de teclado F8 mostrará la hoja en su totalidad.
- Dibujar una caja en la ventana de edición mientras se mantiene pulsada la tecla MAYÚSCULAS. El zoom se hará para mostrar el área de la ventana de edición situada dentro de la caja.



3.1.2.-Utilización de la panorámica en la ventana de edición.

Igual que con el zoom, también existen diferentes formas de variar la panorámica cuando estamos utilizando la ventana de edición:

- Pulsar sobre el botón central (o en la rueda) del ratón para entrar en el modo 'vista panorámica'. Esto sitúa al diseñador gráfico en una situación donde la hoja entera del diseño es capturada y puede ser movida desplazando el ratón. Un cursor en forma de cruz nos indica que nos encontramos en este modo. Pulsando el botón central (o en la rueda) del ratón abandonaremos el modo panorámica.
- Mantener pulsada la tecla Mayúsculas y llevar el puntero del ratón contra uno de los cuatro límites de la ventana de edición (izquierda, derecha, superior o inferior) cuando el flujograma se extiende más allá de los límites del área de trabajo.

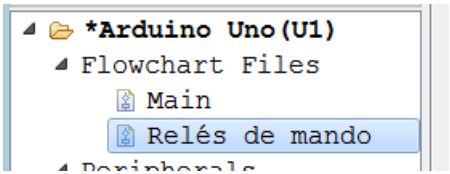
Mientras se está cambiando la panorámica también se puede al mismo tiempo modificar el zoom utilizando la rueda del ratón. Por

ejemplo se puede pulsar en el botón del medio del ratón para desplazarse por el dibujo y al mismo tiempo con la rueda aumentar o disminuir el zoom. Se debe pulsar sobre el botón central (o la rueda) del ratón para abandonar el modo 'vista panorámica'. Es una buena práctica tomarse un tiempo para familiarizarse con las posibilidades de navegación que nos brinda Proteus. Modificar la panorámica, aumentar o disminuir el zoom es de las tareas más frecuentes que se realizan durante el diseño de un proyecto. Es especialmente útil practicar el uso del botón del medio y rueda del ratón.

3.1.3.-Desplazamientos entre las diferentes hojas del flujograma.

Podemos desplazarnos de una hoja del flujograma a otra de las siguientes maneras:

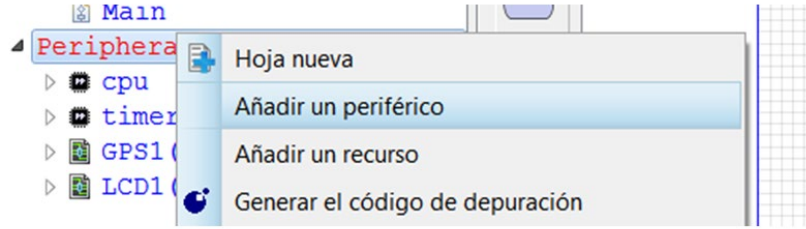
- Pulsando sobre la pestaña de la hoja correspondiente.
- Haciendo doble pulsación en la rama del árbol de estructura del proyecto de la hoja correspondiente.

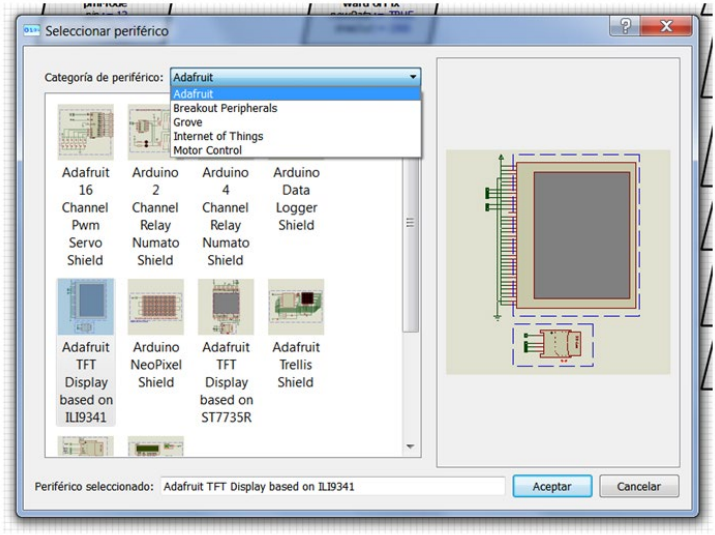


3.2.-Colocación, selección y eliminación de bloques.

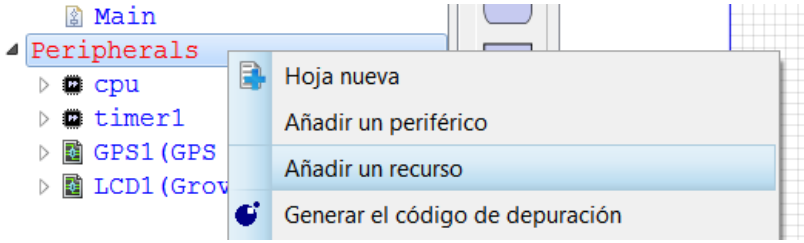
3.2.1.-Colocación de bloques.

Para colocar un periférico a nuestro proyecto debemos utilizar el menú contextual sobre el árbol de estructura del proyecto para que aparezca la ventana de selección de periféricos disponible.

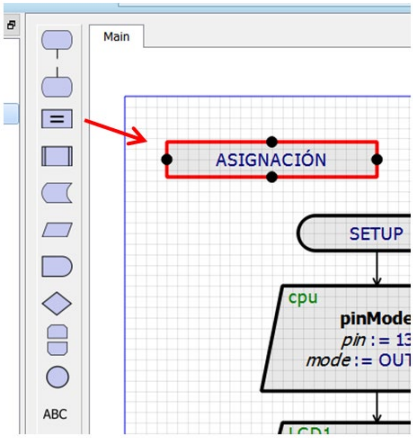




Para añadir un recurso debemos utilizar el menú contextual sobre el árbol de estructura del proyecto para que aparezca la ventana de selección del fichero correspondiente.



Y para colocar cualquier otro tipo de bloque, solo es necesario arrastrarlo a la ventana de edición de código desde la barra de herramientas lateral izquierda.

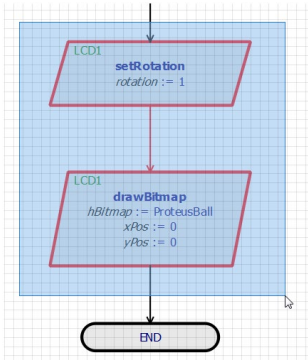


3.2.2.-Selección de bloques.

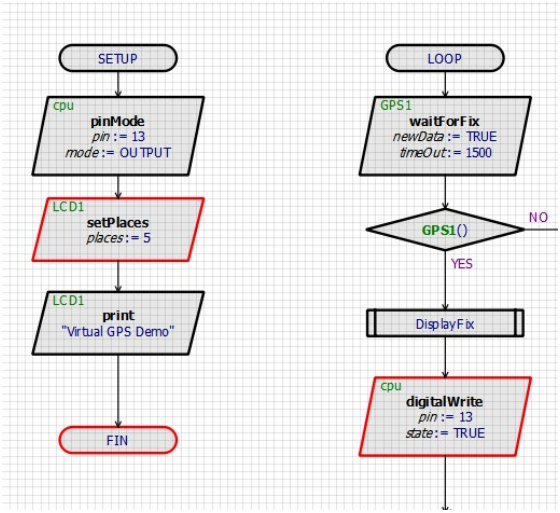
Para seleccionar cualquier bloque situado en la ventana de edición de código sólo hay que pulsar sobre él con el botón izquierdo del ratón.

Para seleccionar un conjunto de bloques podemos utilizar uno de los dos procedimientos siguientes:

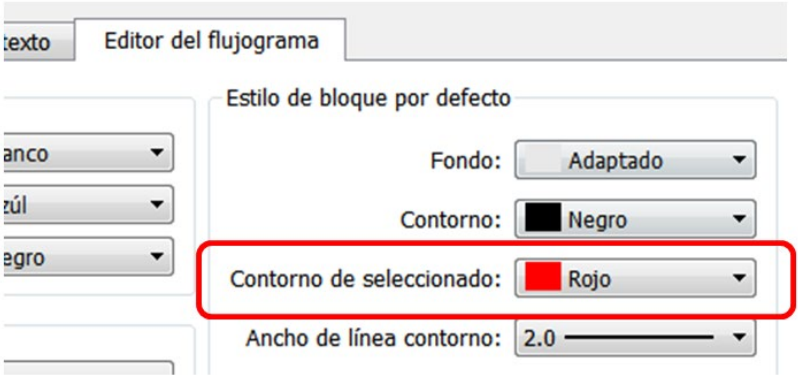
- Dibujar una caja con el puntero del ratón que incluya los bloques que deseamos seleccionar.



- Mantener pulsada la tecla MAYÚSCULAS e ir seleccionando uno a uno los bloques que deseamos seleccionar con el botón izquierdo del ratón. De esta manera se pueden seleccionar bloques que no estén juntos.



Los bloques seleccionados se muestran de color rojo siempre que no hayamos modificado la configuración por defecto de colores del editor del flujograma.



Hay que tener en cuenta que muchas de las acciones que podemos llevar a cabo sobre un objeto del flujograma sólo se pueden realizar sobre un único bloque (por ejemplo, editar). Así que si tenemos seleccionado un grupo de bloques las operaciones disponibles pueden ser menores. Por ejemplo, podemos mover un grupo de bloques, pero si tratamos de utilizar el menú contextual, automáticamente el grupo quedará disuelto y el foco se colocará sobre un único objeto.

3.2.3.-Eliminación de bloques.

Para borrar un bloque o un conjunto de bloques, tenemos que seleccionar el bloque o el conjunto de bloques en cuestión. A continuación podemos utilizar el menú contextual y usar la opción eliminar. También es posible borrar el bloque o conjunto de bloques seleccionados pulsando la tecla SUPRIMIR del teclado.

3.3.-Conexión, desplazamiento y desconexión de bloques.

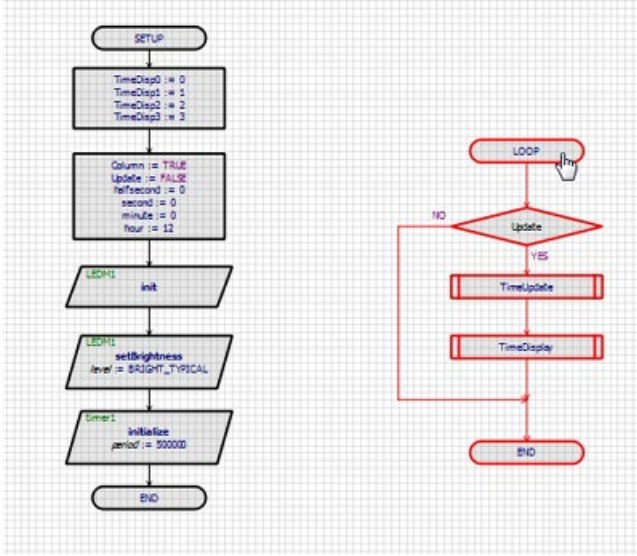
Para seleccionar un bloque sólo hay que posicionar el cursor sobre él y pulsar el botón izquierdo del ratón.

3.3.1.-Desplazamiento de bloques.

Para desplazar un bloque tenemos que seleccionarlo y arrástralo con el ratón hasta la posición dónde queramos situarlo.

3.3.2.-Desplazamientos de bloques conectados entre sí.

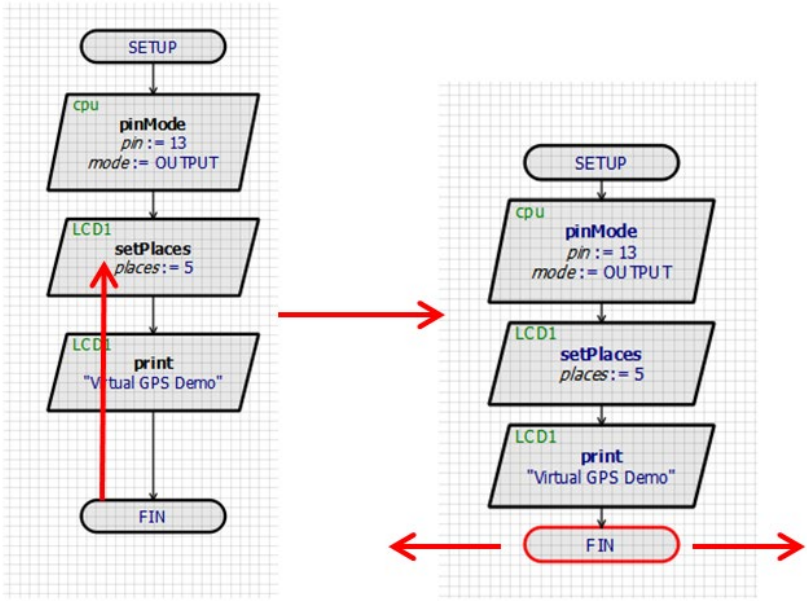
Para mover un conjunto de bloques enlazados unos con otros, utilizaremos la técnica coger y soltar. Primero seleccionamos el conjunto de bloques, nos desplazamos hasta dónde queremos reubicarlos y, por último, los soltamos como si fuera un único bloque.



Si intentamos mover un solo bloque que esté enlazado a otros, se desplazarán también todos los bloques unidos a él en la misma dirección del movimiento, obteniendo como resultando el llamado ‘efecto acordeón’.

Conviene practicar esta importante técnica de desplazamiento de bloques, porque la tendremos que utilizar cada vez que queremos hacer sitio para poder insertar un nuevo bloque dentro de nuestro flujograma.

> Esta técnica puede resultar muy útil para reorganizar nuestro diagrama de flujo verticalmente. Si cogemos el primer bloque y lo desplazamos verticalmente hacia abajo, todos el grupo de bloques que componen la rutina se agruparán manteniendo la distancia de separación entre bloques constante. Igualmente si cogemos el último bloque y lo desplazamos verticalmente hacia arriba. Si cogemos un bloque y lo desplazamos horizontalmente, todo el grupo de bloques se desplazaran a izquierda o derecha.



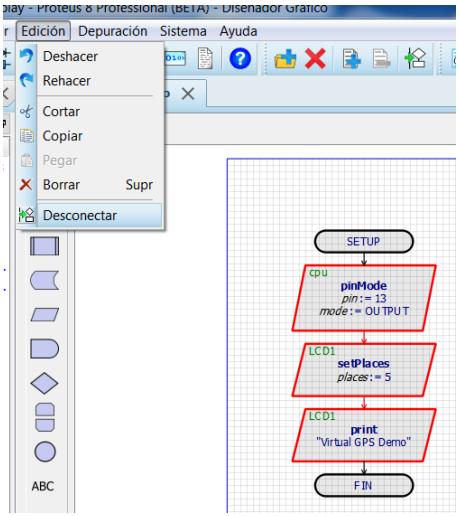
Si deseamos mover un conjunto de bloques de una hoja del flujograma a otra, tendremos que utilizar los comandos copiar y pegar de la barra de herramientas, copiando de una hoja y pegando en la otra.



Para mover un conjunto de bloques que forman parte de una subrutina pero no mover la subrutina completa, tenemos que seguir el siguiente procedimiento:

- Separar la serie de bloques de la rutina.
- Desplazarlos hasta donde deseemos reubicarlos.

Para separar los bloques, los seleccionaremos mientras mantenemos pulsada la tecla CTRL y los moveremos de sitio. O utilizaremos la opción 'Desconectar' del menú 'Edición' después de tener seleccionado un conjunto de bloques.

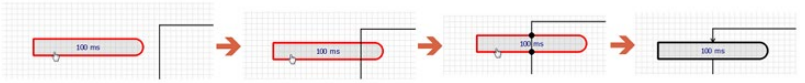


Observe que si no separamos los bloques utilizando la técnica mencionada no seremos capaces de desplazar el conjunto de bloques, porque el resto de bloques de la subrutina se moverán solidariamente con ellos.

3.3.3.-Conexión de bloques al flujograma.

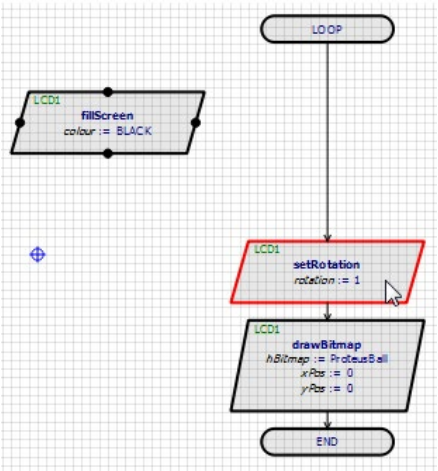
Para insertar un bloque usaremos la técnica coger y soltar. Una vez que hemos cogido un bloque y comenzamos a desplazarlo, si lo

acercamos al flujograma y tenemos el bloque posicionado correctamente sobre alguno de los cables que conectan unos bloques con otros, aparecen los puntos de anclaje. Si soltamos el bloque en ese punto, el bloque quedará correctamente conectado dentro del flujograma.



Podemos llevar a cabo esta operación cuando utilizamos un bloque nuevo por primera vez o seleccionando un bloque ya ubicado en la ventana de trabajo pero que aún no hubiera sido enlazado dentro del flujograma.

Algunos bloques sólo tienen puntos de anclaje en los lados superior e inferior (por ejemplo, los bloques de aplazamiento temporal) mientras que otros tienen puntos de anclaje en los cuatro lados (por ejemplo los bloques de asignación).



En ocasiones puede ser necesario tener que hacer espacio en el flujograma antes de que seamos capaces de insertar un nuevo bloque.

Se puede obtener espacio libre para insertar un nuevo bloque de forma sencilla cogiendo uno de los bloques ya situados en el flujograma y arrastrarlo hacia arriba o hacia abajo para hacer sitio.

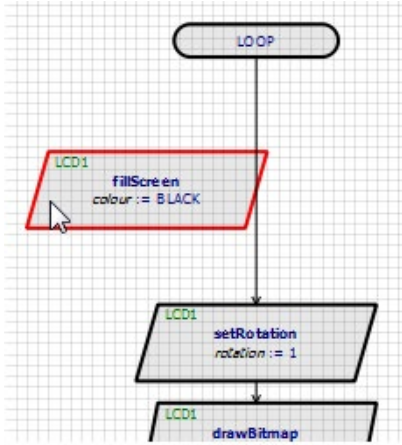
3.3.4.-Desconexión de bloques del flujograma.

Podemos desconectar un bloque del flujograma separándolo del bloque siguiendo uno de los dos procedimientos siguientes:

- Utilizando el menú contextual del bloque en cuestión y seleccionar la opción 'Desconectar'. De esta manera, separaremos el bloque seleccionado del flujograma y quedará libre para reubicarlo donde nos interese.



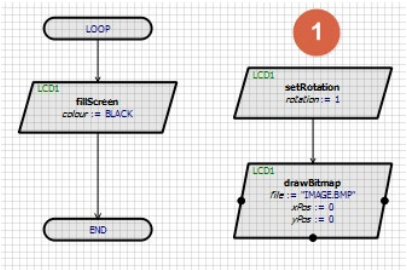
- Seleccionar el bloque en cuestión y mientras mantenemos pulsada la tecla CTRL mover el bloque. De esta manera el bloque seleccionado se separará del flujograma y quedará libre para reubicarlo dónde nos interese.



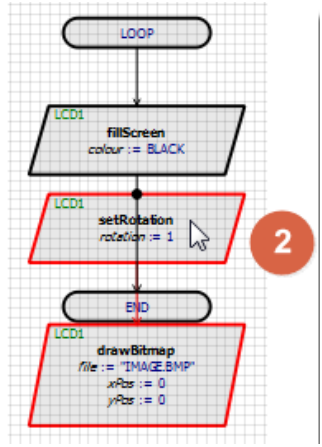
3.3.5.-Conexión de conjuntos de bloques al flujograma.

Para conectar un conjunto de bloques al flujograma debemos seguir el siguiente procedimiento.

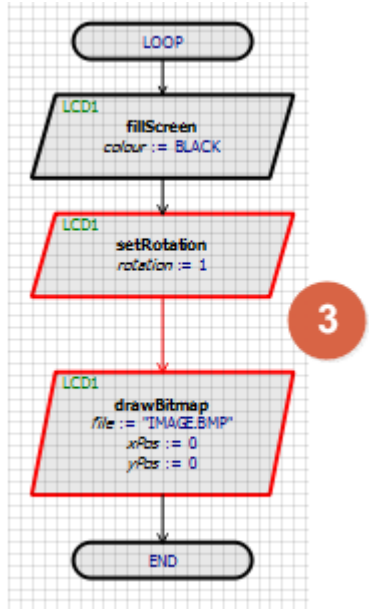
- Hacer espacio para poder colocar los nuevos bloques dentro del flujograma donde deseemos insertarlos.



- Seleccionar la serie de bloques que queremos insertar.



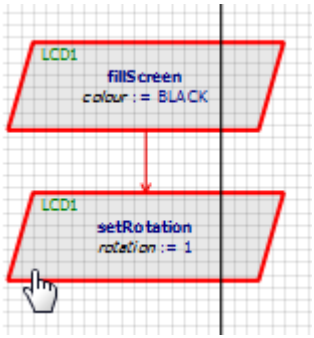
- Desplazarlos hasta el lugar del flujograma dónde deben ir ubicados hasta que observemos que aparecen el punto de anclaje en el bloque que ocupa la posición más alta del conjunto a conectar. Cuando aparezca, comprobar que el punto de anclaje de la zona inferior también es visible.



- Soltar el botón del ratón para que quede el conjunto de bloques insertado.

3.3.6.-Desconexión de conjuntos de bloques del flujograma.

Para desconectar un conjunto de bloques del flujograma tenemos que seleccionar el conjunto de bloques que queremos desconectar y, mientras mantenemos pulsada la tecla CTRL desplazarlos. También podemos seleccionar el conjunto y usar la opción 'desconectar' del menú Edición.

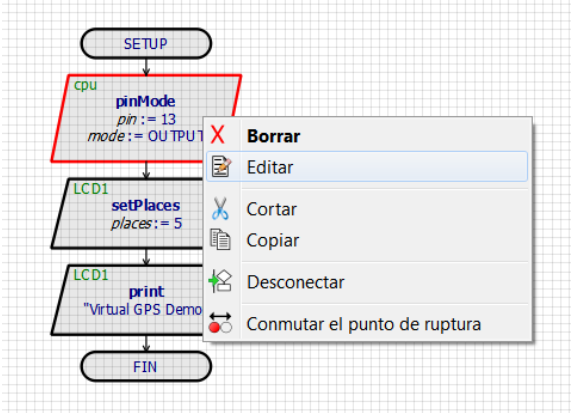


Las órdenes que se ejecutan utilizando los menús contextuales sólo se aplican a un único bloque mientras que las órdenes que se ejecutan utilizando las opciones del menú Edición se pueden aplicar a conjuntos de bloques.

Este es un punto primordial que hay que tener muy en cuenta durante la ejecución de algunos de los comandos (por ejemplo la edición de un bloque que veremos en el apartado siguiente) que no tiene sentido su uso para conjuntos de bloques.

3.4.-Edición de bloques.

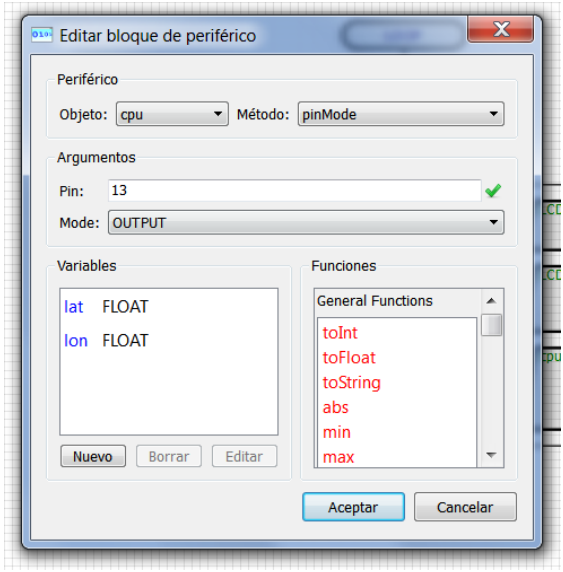
Para editar un bloque podemos utilizar el menú contextual y seleccionar la opción 'Editar' o hace una doble pulsación con el ratón sobre el bloque en cuestión.



Independientemente del tipo de bloque que deseemos editar, siempre tenemos que introducir algún tipo de información para proveerle de sentido. En algunos casos la información necesaria para que un bloque funcione correctamente puede ser extremadamente sencilla (el caso de una llamada a una subrutina). En otros bloques, puede ser un poco más compleja. Afortunadamente, el editor gráfico nos ofrece una serie de herramientas que nos ayudan a llevar a cabo este trabajo y que detallamos a continuación.

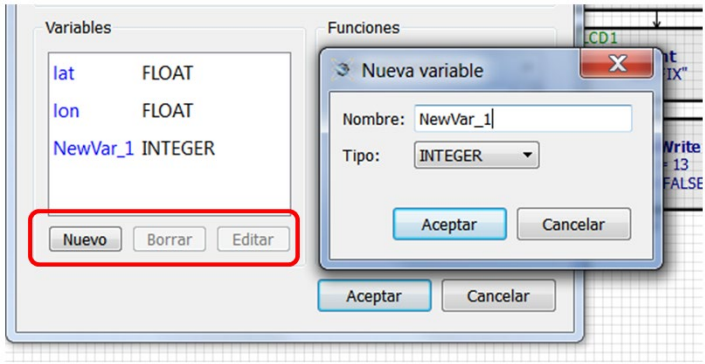
3.4.1.-Lista de variables y funciones.

En los tipos de bloque en los que sea necesario, todas las variables implicadas se mostrarán en la zona inferior izquierda de la ventana de diálogo y las funciones de la librería en la zona inferior derecha.



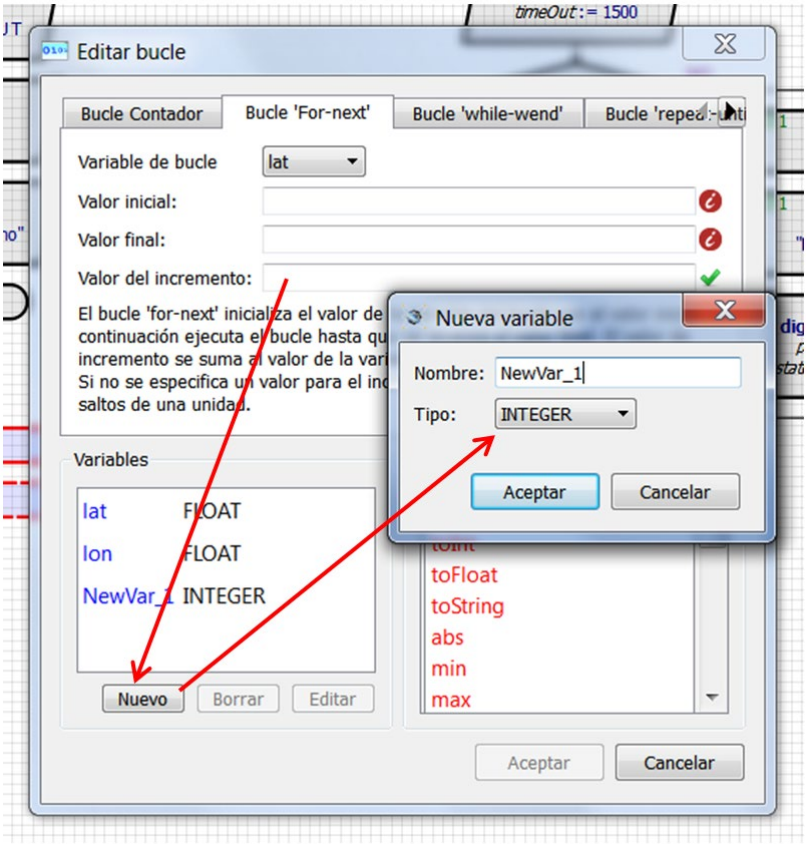
Una doble pulsación del ratón sobre cualquier variable o función, la añadirá automáticamente a la expresión actual que estamos construyendo.

Resulta muy sencillo añadir una nueva variable, eliminarla o editarla, utilizando los tres botones situados justo debajo de la ventana de variables.



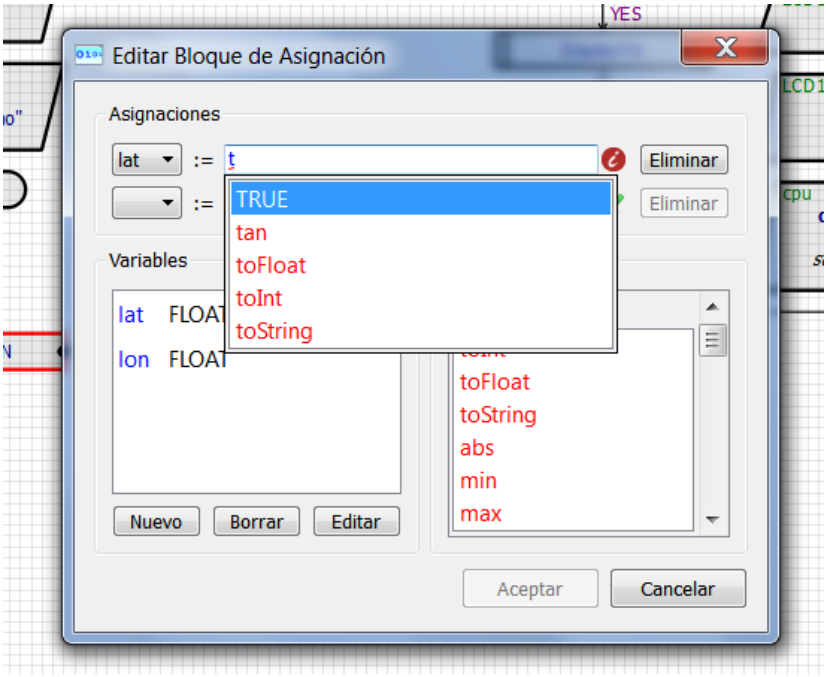
Los cuadros de diálogo nos servirán de ayuda para restringir la selección de variables a aquellas cuyo tipo pueda ser utilizado en cada momento.

Así, por ejemplo, si estamos situados en el campo ‘valor del incremento’ en la pestaña ‘bucle for-next’ de un bloque de bucle y pulsamos el botón ‘nuevo’ para crear una variable para contar los ciclos de ejecución, la variable a utilizar debe ser siempre del tipo ENTERO y ésta es la opción que se nos ofrece por defecto.



3.4.2.-Función auto-completar.

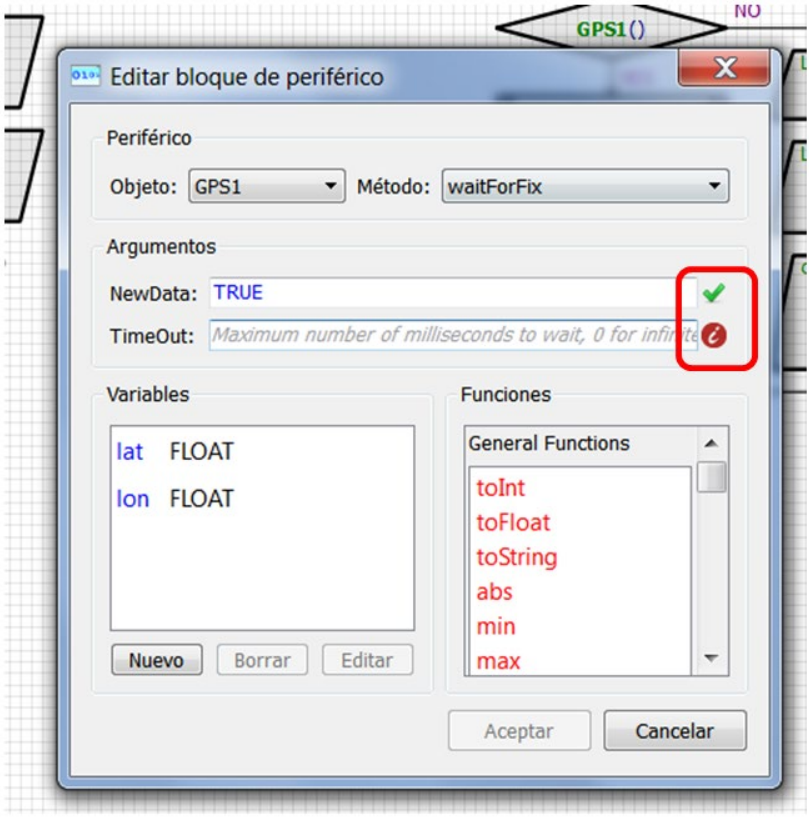
El sistema auto-completar nos ayudará tanto a completar el nombre de una función a medida que vamos tecleando su nombre como a completar el nombre de las variables.



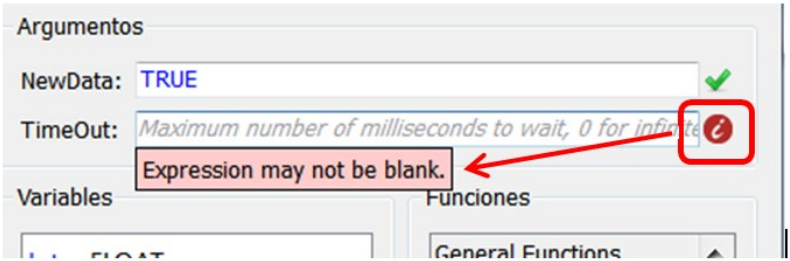
Pulsando la tecla TABULADOR o ENTER, validaremos la propuesta del asistente y el nombre de la variable o función se completará.

3.4.3.-Comprobación de tipos y sintaxis.

El sistema de comprobación de sintaxis y tipos trabaja cuando validamos una expresión con la tecla INTRO. El estado de la validación se muestra a la derecha del cuadro con un signo de con una i roja o un signo de verificación positiva verde.



Si hacemos una doble pulsación del ratón sobre el icono con la i roja, se mostrará un rótulo con el detalle del error que se haya sido detectado por el sistema.

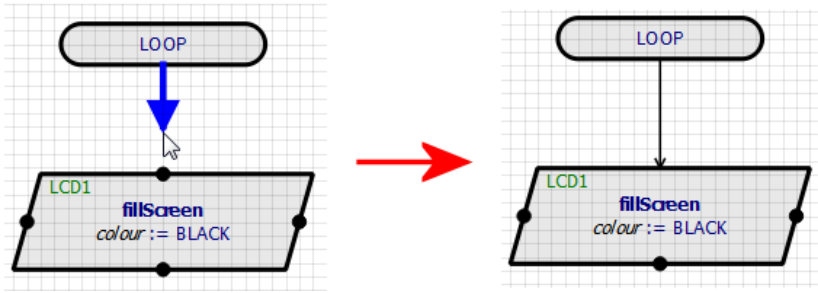


Escribiendo algo sobre el campo, el mensaje informativo desaparecerá.

En los apartados dedicados a cada tipo de bloque se puede encontrar más información detallada sobre los mensajes informativos de error.

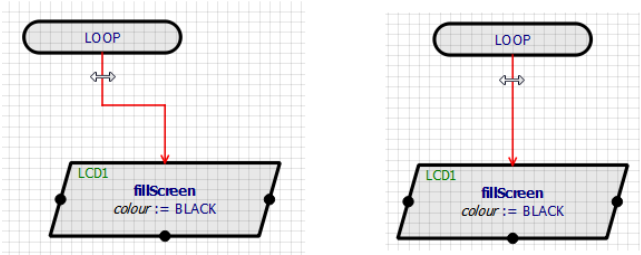
3.5.-Conexión de un bloque con otro.

Un bloque se conecta con otro mediante las líneas de flujo. El editor gráfico nos facilita un enrutador punto a punto para ayudarnos en la tarea de conectar bloques. La regla de oro que tenemos que tener siempre presente es que hay que conectar un punto de anclaje de salida de un bloque con un punto de anclaje de entrada de otro bloque.

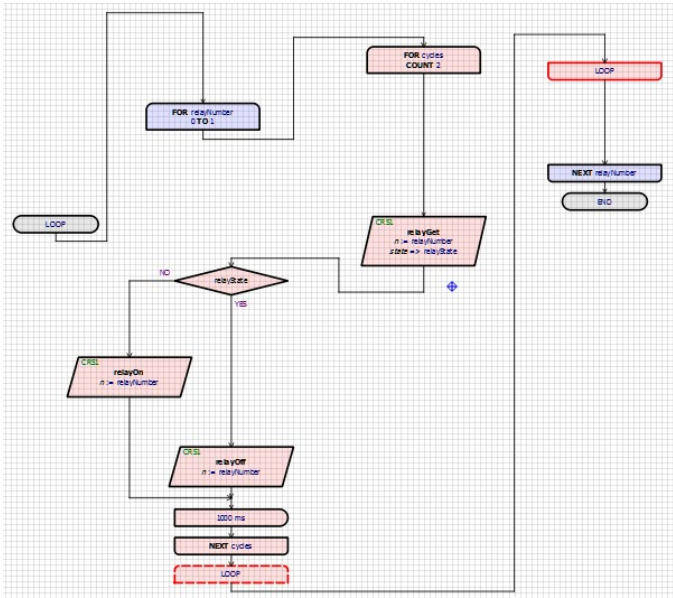


Para unir dos bloques ejecutaremos tres pasos:

- Una pulsación sobre el botón izquierdo del ratón en el punto de anclaje de salida del primer bloque.
- Desplazamiento del puntero del ratón hasta el punto de anclaje de entrada del segundo bloque.
- Pulsación sobre el botón izquierdo del ratón para confirmar el trazado de la línea de flujo.

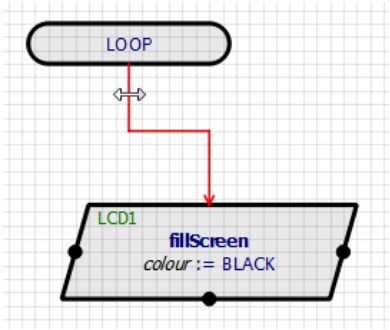


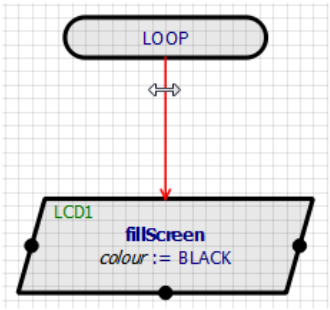
Es una buena práctica de programación para facilitar la lectura la lógica de los programas cuando se usan flujogramas utilizar siempre líneas de flujo que parten desde un bloque que se encuentra más arriba hasta alcanzar un bloque que se encuentre más abajo. Aunque es posible, evite conexiones de bloques de abajo hacia arriba que hacen muy farragosa la interpretación del flujograma.



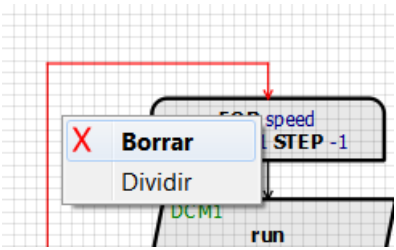
Para desplazar o ajustar una línea de flujo existente ejecutaremos los tres pasos siguientes:

- Pulsación del ratón sobre la línea de flujo para seleccionarla (el trazo aparecerá de color rojo para indicar que la hemos 'agarrado').
- Desplazamiento del puntero sin dejar de pulsar el ratón hasta el punto de donde deseamos dejar situada la línea de flujo.
- Liberación del ratón para 'soltar' la línea de flujo.



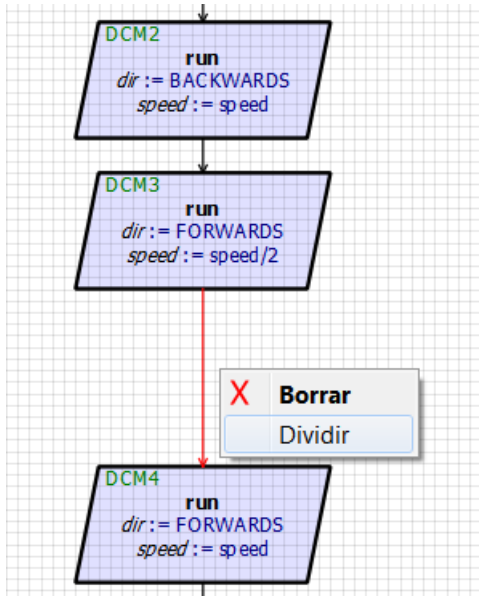


Para eliminar una línea de flujo existente, tenemos que seleccionar la línea en cuestión y pulsar la tecla SUPRIMIR para borrarla. También podemos seleccionar la línea y desde su menú contextual elegir la opción 'Borrar'.

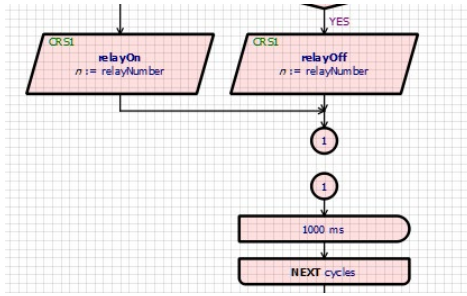


En ocasiones el flujograma es muy largo y al dibujarlo alcanzaremos la zona inferior del área de trabajo, por lo que será necesario dividirlo en varios trozos. Para dividir varios bloques conectados de un flujograma en dos conjuntos de bloques conectados en diferentes columnas debemos seguir los cuatro pasos siguientes.

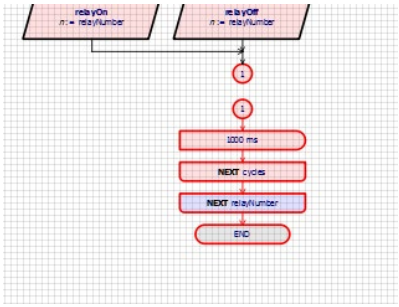
- Seleccionamos la línea de flujo donde deseamos hacer la división y desde el menú contextual seleccionamos la opción 'dividir'.



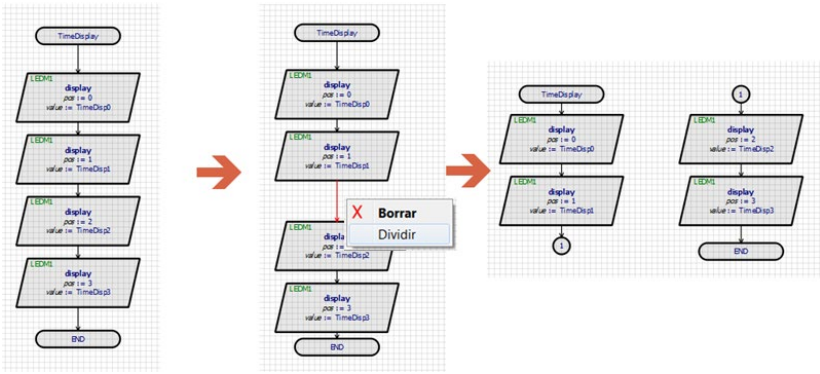
- El flujograma se divide en dos partes. En la zona inferior del conjunto superior aparecerá un bloque conector con un número. En la zona superior del conjunto inferior aparecerá otro bloque conector con el mismo número.



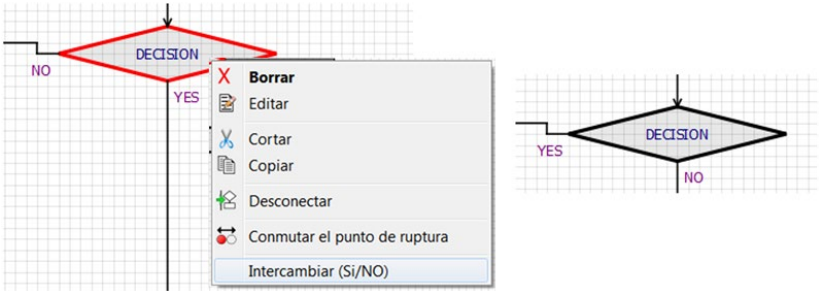
- Seleccionamos el conjunto inferior resultado de la división.



- Desplazamos el conjunto hasta dónde queramos reubicarlo.



Para cambiar el flujo del programa según el resultado de una condición nos ayudamos de los bloques de decisión. Si deseamos modificar la salida que se utiliza para la evaluación positiva (YES) y negativa (NO), podemos utilizar la opción 'Intercambiar (SI/NO)' del menú contextual de este tipo de bloques.



3.6.-Utilización del portapapeles.

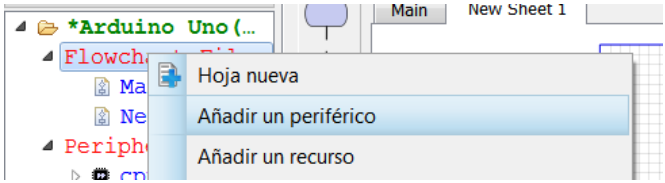
Los comandos de utilización del portapapeles en el diseñador gráfico son idénticos a los del resto de herramientas de Proteus y de la mayoría de los programas del entorno Windows. El atajo de teclado CTRL-X para cortar, el atajo de teclado CTRL-C para copiar y el atajo de teclado CTRL-V para pegar. Las tres opciones también están disponibles desde el menú 'Edición'. Y los iconos de la barra de herramientas utilizados para estas tres operaciones son:

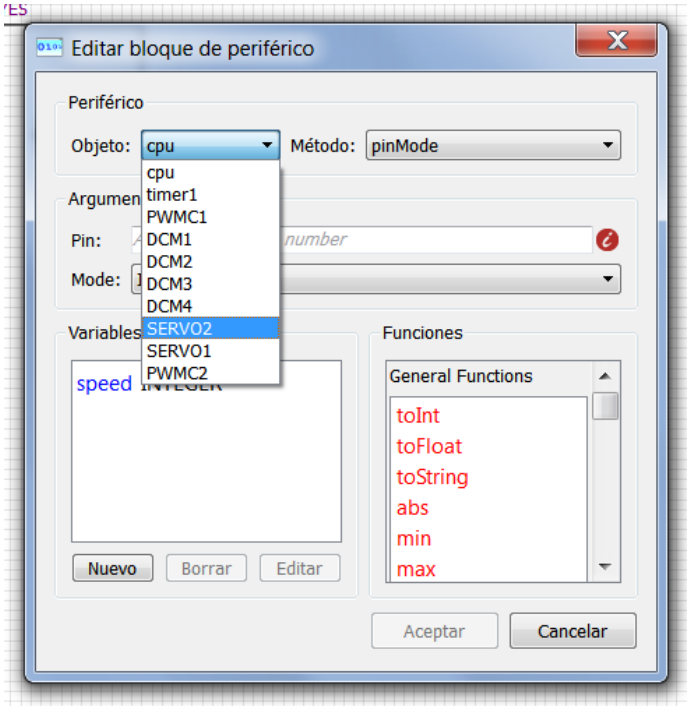


Utilizaremos el portapapeles habitualmente desde el diseñador gráfico para reorganizar las rutinas de bloques del flujograma en una misma hoja o para desplazar rutinas de una hoja a otra de nuestro diseño.

Hay que tener siempre presente que cuando copiamos y pegamos, podemos hacerlo incluyendo bloques que están directamente relacionados con periféricos (por ejemplo, tarjetas de expansión). En estos casos, el pegado duplicará los bloques en el flujograma y las asignaciones directamente implicadas. Pero no se crearán nuevos elementos de hardware en nuestro diseño electrónico.

Si lo que deseamos hacer es duplicar el hardware, tendremos que añadir el periférico, editar los bloques pegados y asignar los métodos al nuevo hardware creado de manera correcta.



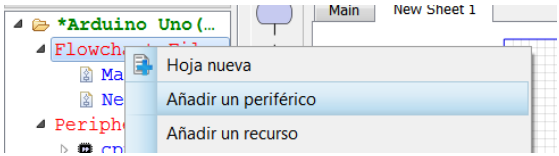


3.7.-Colocación, utilización y eliminación de periféricos.

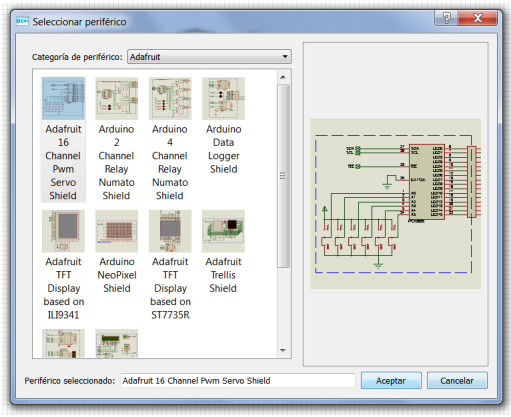
3.7.1.-Añadir un periférico.

Para añadir un periférico nuevo en nuestro proyecto, seguiremos los siguientes pasos:

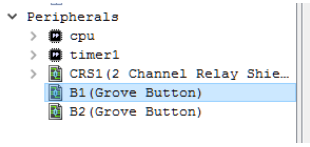
- Desde el menú contextual del árbol de estructura del proyecto, seleccionamos la opción añadir un periférico nuevo.

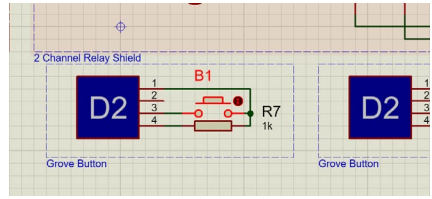


- Seleccionamos la tarjeta de expansión o el periférico Grove desde la ventana de selección de periféricos que aparece.

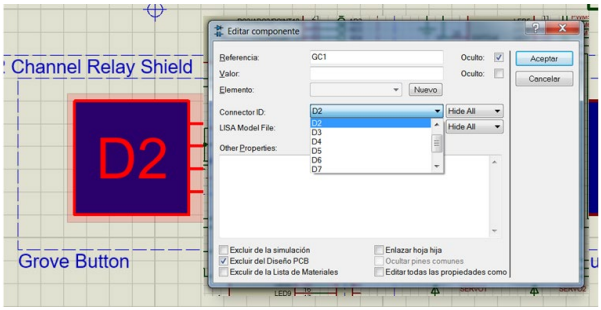


- Si estamos utilizando un periférico Grove es muy posible que tengamos que modificar su ID para que no entre en conflicto con uno utilizado previamente. Para lograrlo, haremos una doble pulsación sobre la etiqueta del nuevo periférico creado en el árbol de estructura del proyecto. Al hacerlo nos desplazaremos a la pestaña del esquema electrónico y Proteus nos situará de forma automática en el lugar dónde se encuentra dicho periférico.





Y una vez allí, tenemos que hacer una doble pulsación sobre el conector para cambiar su identificador (ID).



> Mientras utilizamos Arduino es realmente fácil que cuando añadamos una tarjeta de expansión, ésta pueda resultar incompatible con otra ya existente en nuestro proyecto.

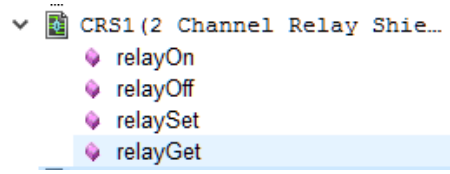
Por ejemplo si dos tarjetas de expansión utilizan el mismo temporizador (timer) de la CPU, el programa no funcionará correctamente.

Es muy importante que comprobemos bien la compatibilidad de las diferentes tarjetas de expansión que utilicemos en nuestro proyecto puesto que el diseñador gráfico no interviene en lo relativo a los drivers de Arduino.

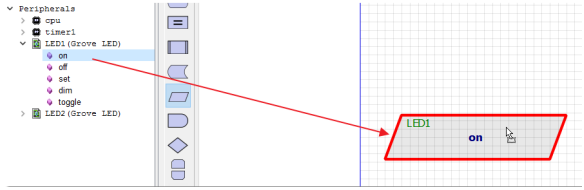
3.7.2.-Utilizar un periférico.

Para utilizar un periférico ya creado en nuestro proyecto tenemos que seguir dos pasos:

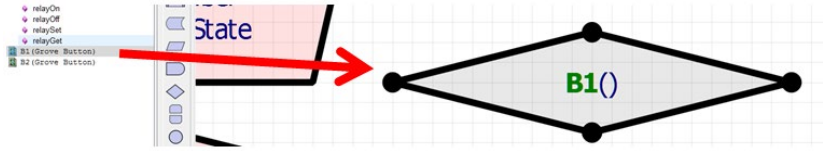
- Expandir en el árbol de estructura de proyectos, la rama correspondiente al periférico que deseemos utilizar para que se visualicen los métodos de gestión de dicho periférico.



- Copiar y pegar el método dentro de nuestro flujograma.

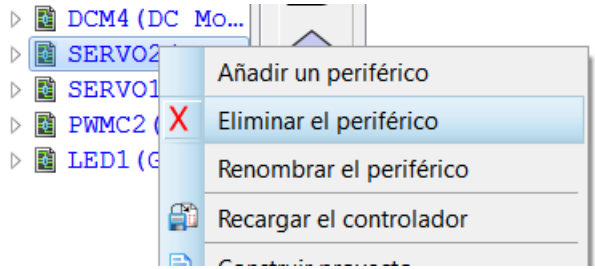


Algunos periféricos sólo tienen un método asociado. Por ejemplo el módulo de pulsador GROVE. En estos casos no es necesario ni posible expandir en el árbol de estructura de nuestro proyecto ninguna rama. Así que sólo tenemos que coger y pegar el propio dispositivo dentro de nuestro flujograma y el único método asociado se colocará en él.



3.7.3.-Eliminar un periférico.

Para eliminar un periférico que hubiéramos puesto con anterioridad en nuestro proyecto, tenemos que seleccionarlo en el árbol de estructura de nuestro proyecto y seleccionar la opción ‘Eliminar el periférico’ del menú contextual .



Si eliminamos un periférico, no se eliminarán de forma automática todos los bloques colocados en el flujograma que guarden relación con él. Necesitaremos modificar o eliminar cada uno de estos bloques de forma manual.

3.8.-Creacion, utilización y eliminación de variables.

Las variables se crean y manipulan desde dentro de las diferentes ventanas de diálogo asociadas con los objetos de nuestro flujograma. Todas las variables son siempre del tipo global. Las variables globales son aquella que su vida se extiende a lo largo de todo el proyecto. En contraposición, las variables locales son las que su vida sólo se alarga durante el tiempo que se ejecuta la función dentro de la cual han sido creadas.

Se soportan los siguientes tipos de datos para definir las variables:

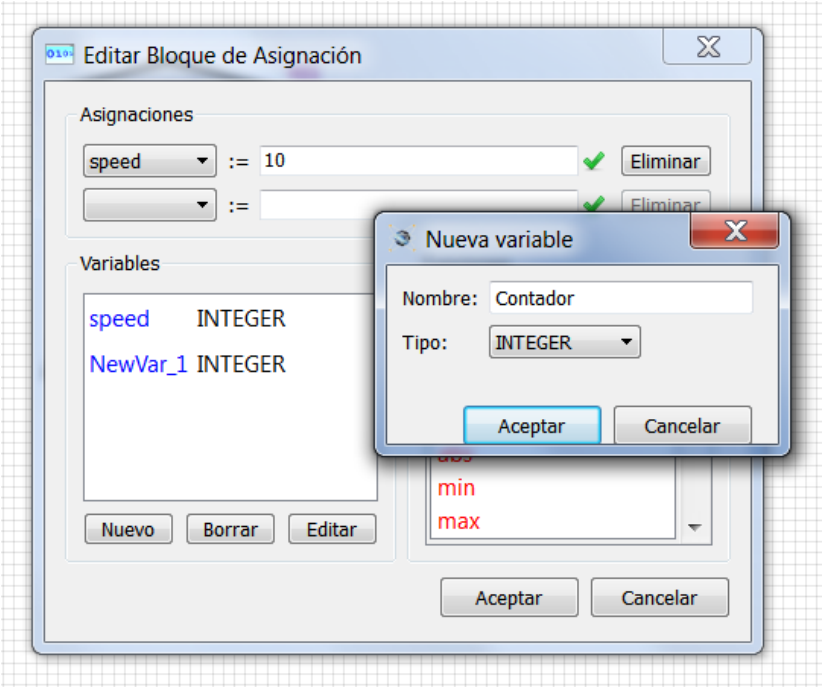
- Boolean (booleana de valores verdadero o falso).
- Integer (número entero).
- Float (número de coma flotante).
- String (cadena de caracteres).
- Handle (gestor que sirve para identificar a un determinado recurso).

El editor de expresiones realiza la comprobación del tipo de variable asignado de forma automática (ver el epígrafe 3.4.3.).

3.8.1.-Creación de una variable.

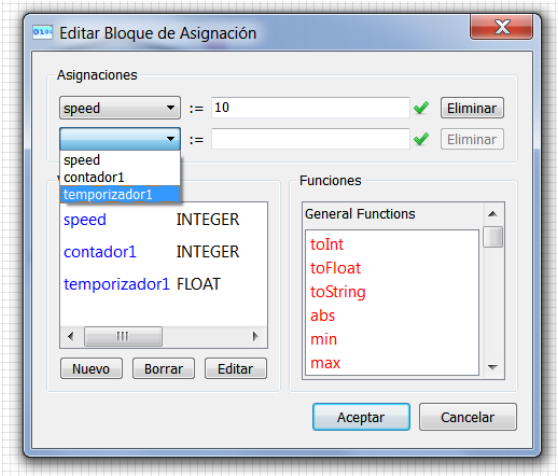
Para crear una variable seguiremos los siguientes pasos:

- Editar el bloque con el que deseamos trabajar. Se abrirá la ventana de diálogo correspondiente.
- En la zona inferior derecha está el botón que utilizaremos para crear una nueva variable (rotulado 'Nuevo').
- Introducimos el nombre de la nueva variable, le asignamos el tipo correspondiente y pulsamos sobre el botón 'Aceptar'.
- Cerramos la ventana de diálogo pulsando sobre el botón 'Aceptar'.

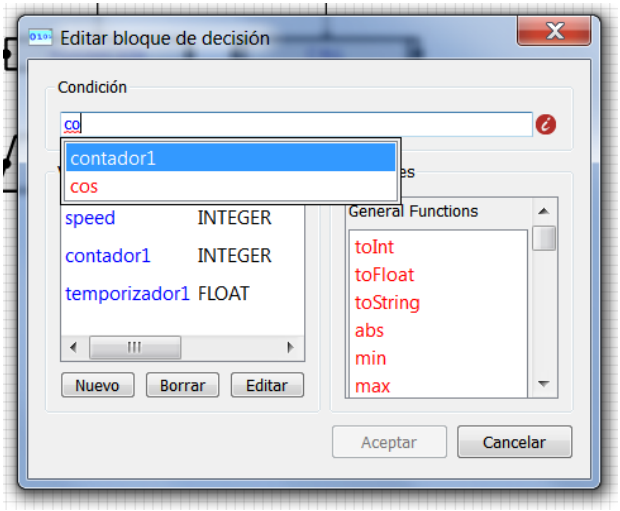


3.8.2.-Utilización de una variable.

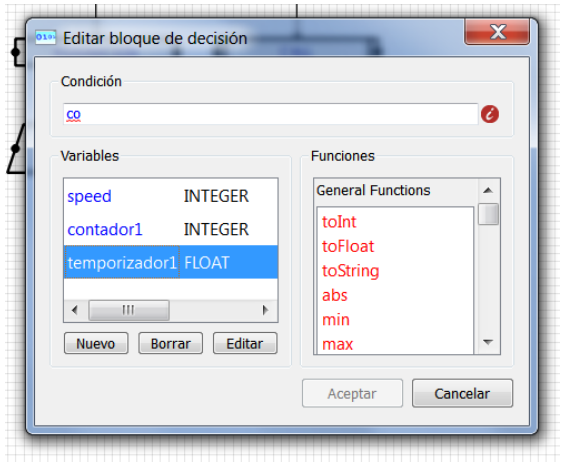
Para usar una variable tenemos que editar el bloque con el que está esa variable relacionado. Según el tipo de bloque podemos trabajar con una variable seleccionándola desde la lista de variables.



O podemos escribir el nombre de la variable en el cuadro de texto disponible. Al ir escribiendo, la función auto-completar (ver el epígrafe 3.4.2) nos ayudará a seleccionar la variable que deseamos utilizar de entre las disponibles.

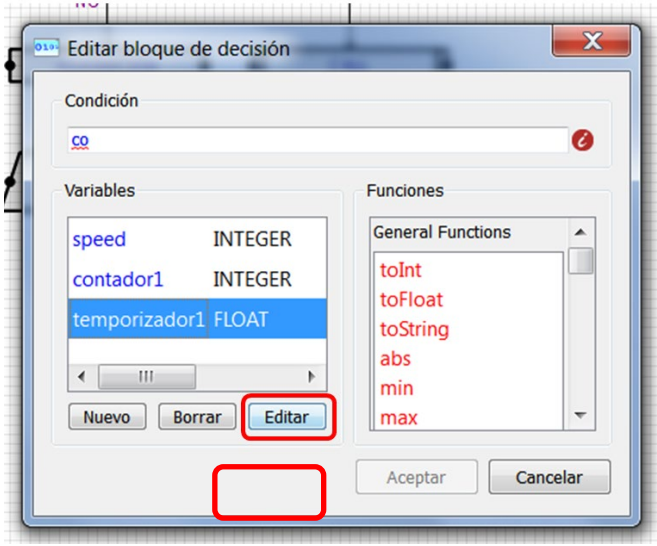


O hacer una doble pulsación sobre el nombre de la variable para insertar en el cuadro de texto activo.



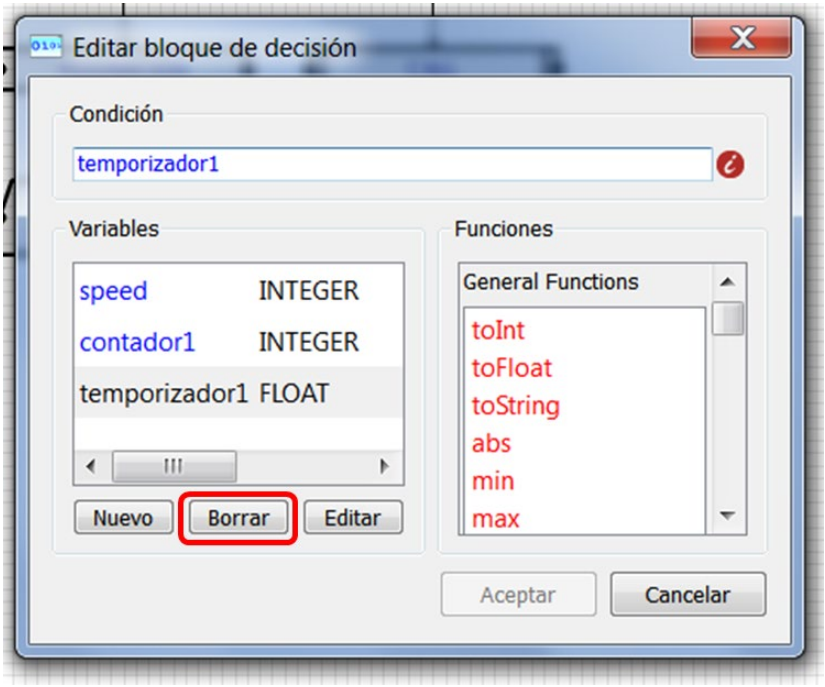
3.8.3.-Edición de una variable.

Para editar una variable tenemos que editar el bloque en el que se utiliza dicha variable. A continuación, seleccionaremos la variable que deseamos editar desde la lista de variables ubicada en la zona inferior izquierda. Por último, pulsaremos sobre el botón 'editar' para cambiar el nombre o del tipo de una variable.



3.8.4.-Eliminación de una variable.

Para eliminar una variable tenemos que editar el bloque en el que se utiliza dicha variable. A continuación, seleccionaremos la variable que deseamos editar desde la lista de variables ubicada en la zona inferior izquierda. Por último, pulsaremos sobre el botón 'borrar' para borrar la variable.



Si eliminamos o editamos una variable, necesitaremos comprobar si otros bloques utilizan esta variable y modificarlos si fuera necesario.

4.-Los bloques de flujograma.

El diseñador gráfico utiliza un amplio conjunto de bloques diferentes que podemos utilizar en nuestro flujograma para construir el código que se ejecutará en nuestro Arduino. Los usuarios familiarizados con los diagramas de flujo les resultará sencillo reconocerlos.

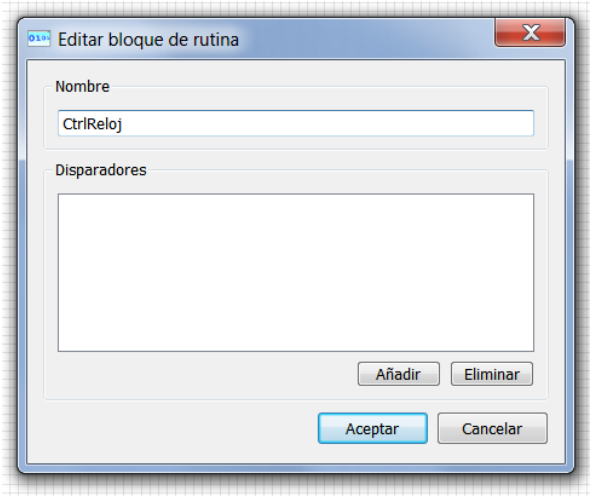


- Bloque de arranque de evento o rutina
- Bloque de fin de evento o rutina
- Bloque de asignación
- Bloque de llamada a subrutina
- Bloque de almacenamiento de datos
- Bloque de operación con periférico
- Bloque de aplazamiento temporal
- Bloque de decisión
- Bloque de bucle
- Bloque de conector
- Bloque de comentario

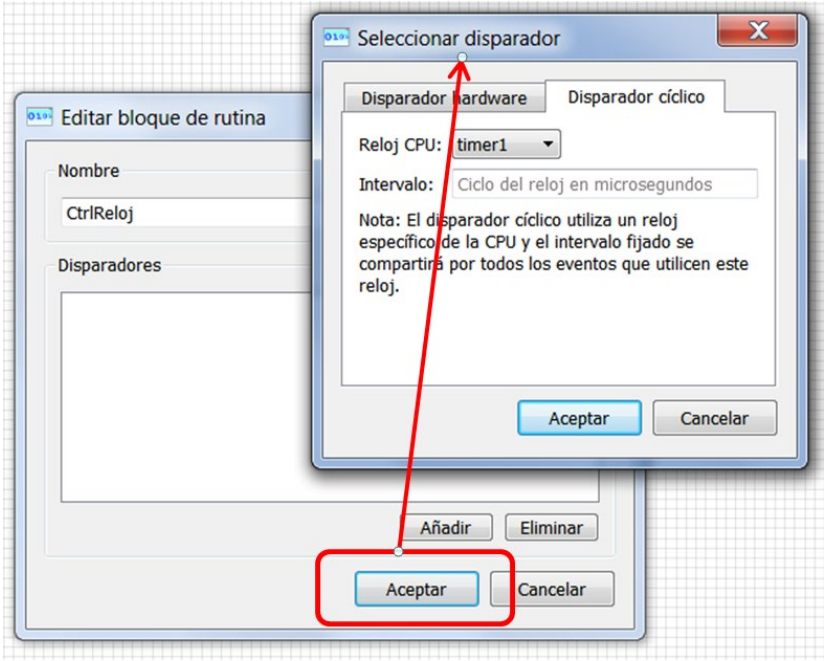
4.1.-Bloque de arranque de evento o subrutina.

El bloque de arranque de evento o subrutina se utilizar, en unión del bloque de fin de evento o subrutina, para definir el comienzo y el final de una subrutina (por ejemplo, la tarea de escribir sobre un display) o del gestor encargado de gestionar la respuesta a un evento (por ejemplo, gestionar una interrupción de reloj).

Para crear una nueva subrutina todo sólo es necesario asignarle un nombre a este bloque. Este nombre será el mismo que se le asigne a la subrutina y el que se utilizará más tarde cuando desde otro bloque del tipo llamada a subrutina se quiera ejecutar una llamada a dicha subrutina.



Para crear un gestor para gestionar la respuesta a un evento del tipo interrupción necesitamos, además de darle un nombre, indicar el evento al que responde este gestor (el disparador del evento).

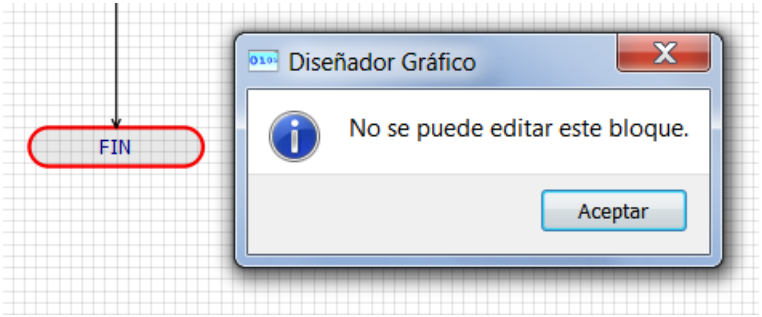


El disparador asociado a un reloj es una buena forma de generar llamadas a rutinas que se ejecutarán a intervalos regulares de tiempo. Si se utiliza uno de los relojes de la CPU para este fin hay que ser cuidadoso y asegurarse de que ninguna otra subrutina del proyecto utiliza este mismo TIMER generando un conflicto. Especialmente hay que estudiar bien las rutinas de gestión de periféricos (asociadas a las tarjetas de expansión o módulos Grove) que se incorporan a nuestro código de forma automática al usar dichos módulos dentro de nuestro proyecto.

4.2.-bloque de fin de evento o subrutina.

El bloque de fin de evento o subrutina se utiliza para indicar el final de una subrutina o gestor de eventos. Siempre trabaja en conjunto con el bloque de arranque de evento o subrutina. No tiene ninguna necesidad de ser configurado ni parametrizado. Sólo tenemos que

colocarlo donde termine el conjunto de bloques que forman la subrutina.

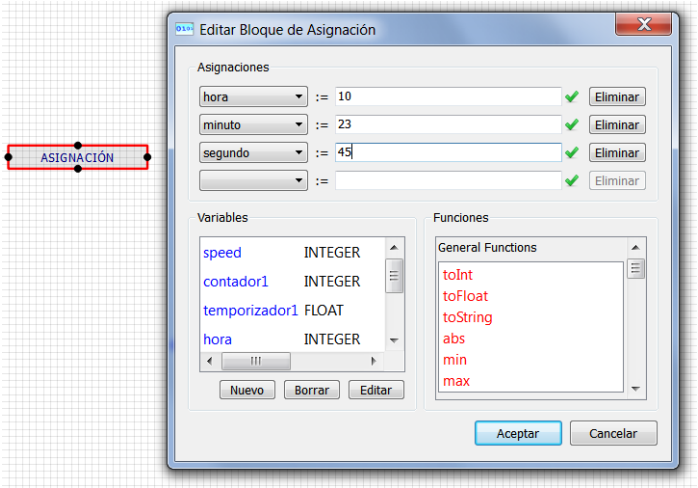


Un proyecto tipo de Arduino siempre empieza utilizando dos pares de bloques de inicio-fin de evento o subrutina. Una de estas parejas contiene el código que ejecuta la función setup() con las instrucciones de configuración e inicialización.

La otra pareja contiene el código que ejecuta la función loop() con las instrucciones que se repiten en cada ciclo de ejecución de la rutina principal.

4.3.-Bloque de asignación.

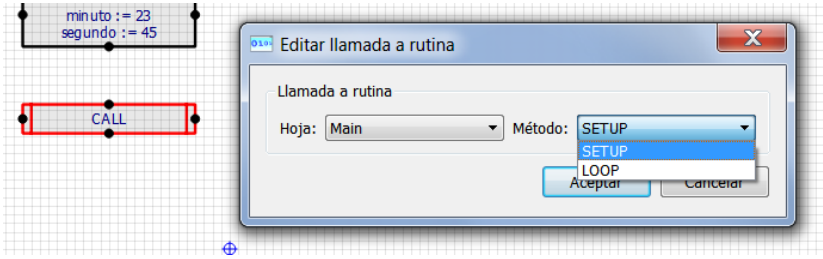
Este tipo de bloques se utilizan para asignar valores a las variables. Es el lugar dónde podemos crear, editar o eliminar una o más variables que tengamos que utilizar en nuestro código.



4.4.-Bloque de llamada a subrutina.

Un bloque de llamada a subrutina nos permite ejecutar una determinada función en el lugar de nuestro flujograma que deseemos.

Cuando utilizamos un bloque de este tipo hay que indicar la hoja en la que se encuentra la subrutina que deseamos utilizar y el nombre de la subrutina a la que queremos llamar.

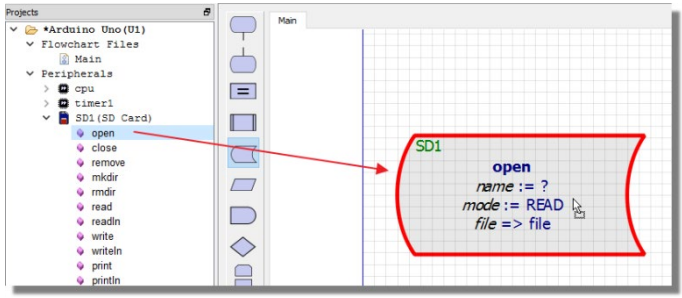


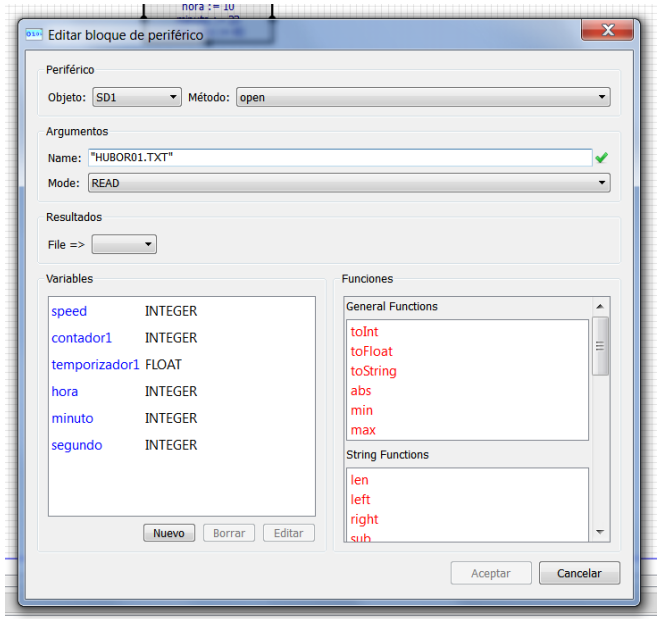
Antes de poder utilizar un bloque de llamada a una subrutina es necesario crear la subrutina utilizando los bloques de arranque y fin de evento o subrutina. La lista desplegable

sólo muestra los nombres de las subrutinas correctamente creadas en nuestro proyecto.

4.5.-Bloque de almacenamiento de datos.

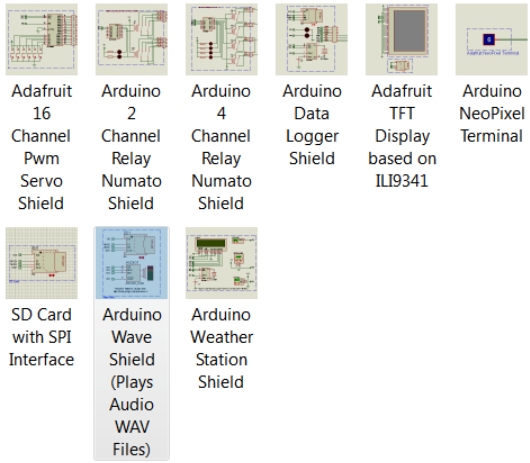
Un bloque de almacenamiento de datos se utiliza para llevar a cabo una acción o un método asociado a un objeto donde podemos guardar datos (v.g. una tarjeta de memoria SD). Los métodos habitualmente utilizados son aquellos que se usan para manipular los ficheros almacenados (v.g. abrir, escribir, leer, cerrar un fichero de texto) en el dispositivo de memoria.

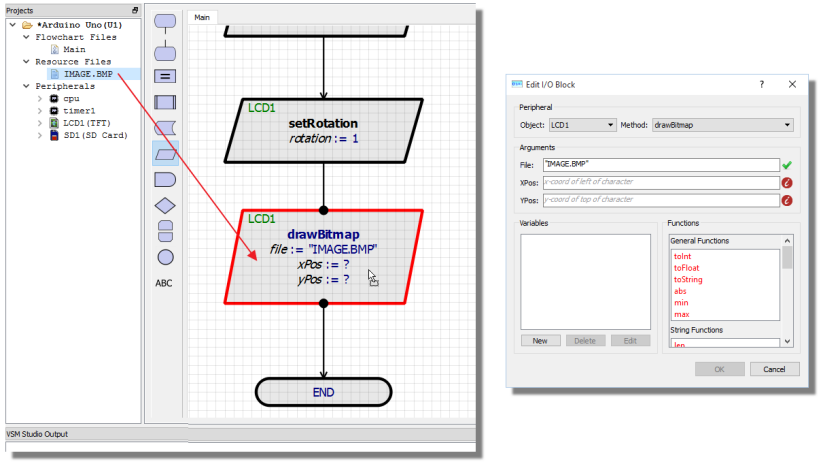




Es importante tener en cuenta que cuando añadimos a nuestro proyecto una tarjeta de expansión que incluye una tarjeta de memoria SD, casi siempre dispondremos de uno o más métodos que nos permitan usar los ficheros contenidos en dicha tarjeta.

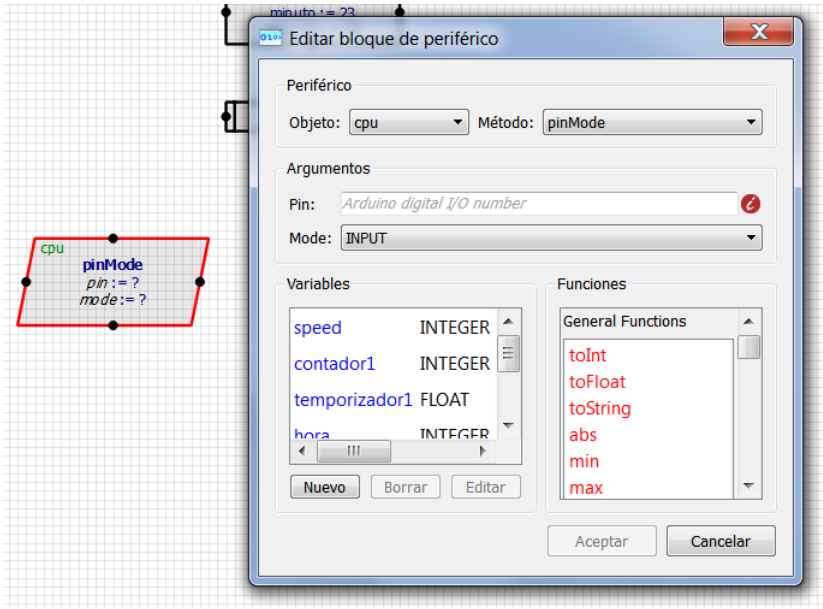
Si, por ejemplo, utilizamos una tarjeta de expansión del tipo Adafruit TFT Display, podemos comprobar que incluye el método DrawBitmap(). Y si seleccionamos una tarjeta de expansión del tipo Arduino Wave, tendremos disponible el método Play(). En estos dos casos, no deberíamos de utilizar los bloques de almacenamiento de datos y solamente deberíamos coger y arrastrar el recurso que necesitemos dentro de nuestro flujograma.





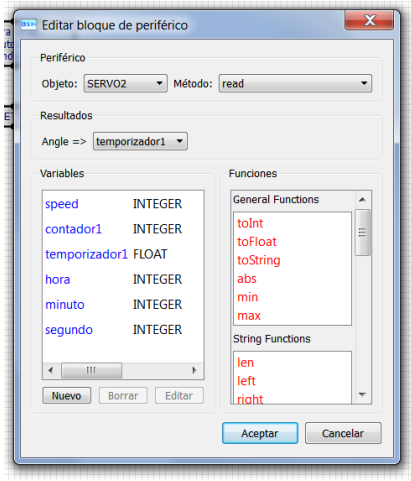
4.6.-Bloque de operación con periférico.

Un bloque de operación con periférico nos permite llevar a cabo acciones relacionadas con el hardware que estamos utilizando. Cuando creamos un nuevo proyecto el hardware sólo consiste en el procesador Arduino y las acciones disponibles relacionadas con él son muy limitadas.



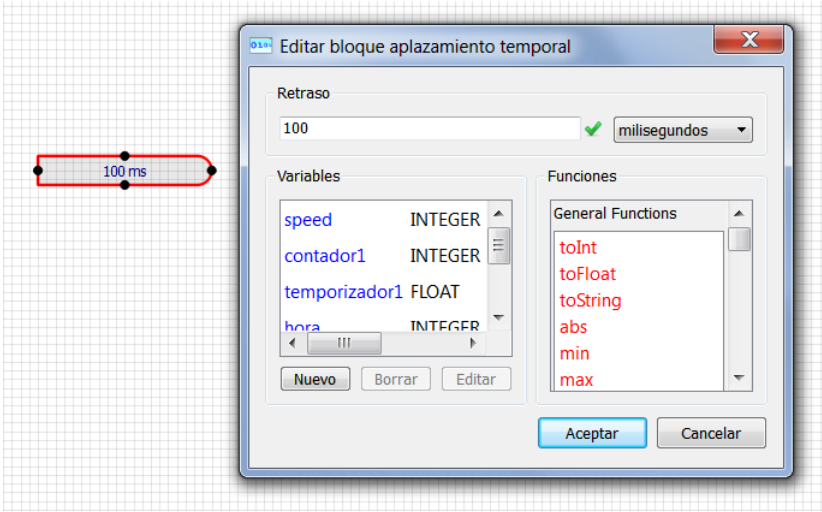
En cuanto añadimos alguna tarjeta de expansión con cualquier tipo de periférico a nuestro proyecto, ya estaremos en condiciones de interactuar y controlar sus entradas y salidas utilizando estos tipos de bloques.

En la imagen siguiente se muestra el ejemplo de un servomotor.



4.7.-Bloque de aplazamiento temporal.

Este tipo de bloque se utiliza para poder introducir dentro de nuestro código una función delay(). En la imagen siguiente se muestra un ejemplo de uso de uno de estos bloques para implementar un delay() de 100mseg.

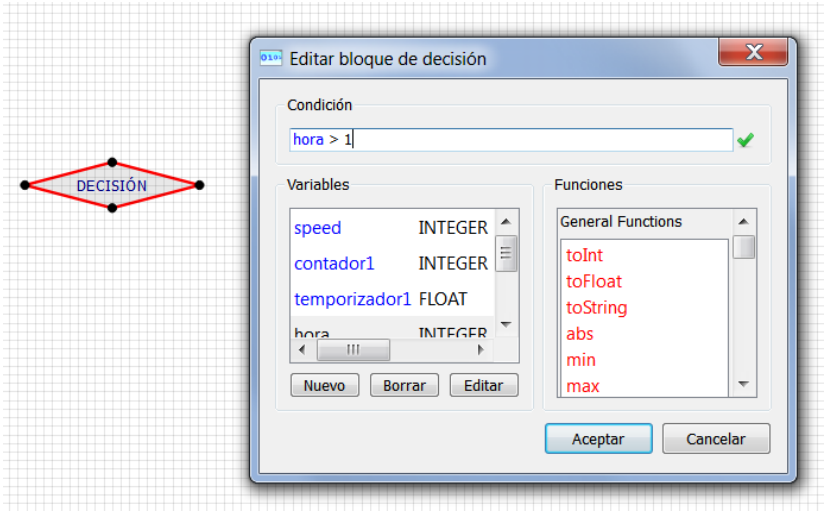


⚠ Los equipos Arduino no permiten leer sensores, realizar cálculos matemáticos o manipular pines de entrada y salida mientras se esté ejecutándose una función delay(). Por eso, utilizar una función delay() es lo más parecido a mantener 'colgado' nuestro microprocesador durante ese periodo de tiempo. Las interrupciones y otras pocas funciones específicas si seguirán activas y trabajarán si fuera necesario. Consultar la documentación sobre la función delay() en la web de Arduino para más detalles.

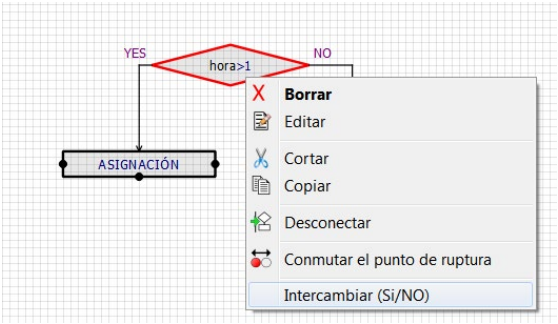
4.8.-Bloque de decisión.

Un bloque de decisión implementa una pregunta básica que sólo admite como respuesta un sí o un no (la estructura de control if/else estándar de Arduino). En función de la respuesta evaluada en cada

momento nos permite determinar por qué ramal del flujograma seguirá la ejecución de nuestro código.



Cuando colocamos un bloque de decisión dentro de nuestro flujograma, automáticamente dos etiquetas con los rótulos YES y NO son colocadas en sus posiciones por defecto. Podemos intercambiarlas utilizando el menú contextual si ello resulta más cómodo para nuestro flujograma.



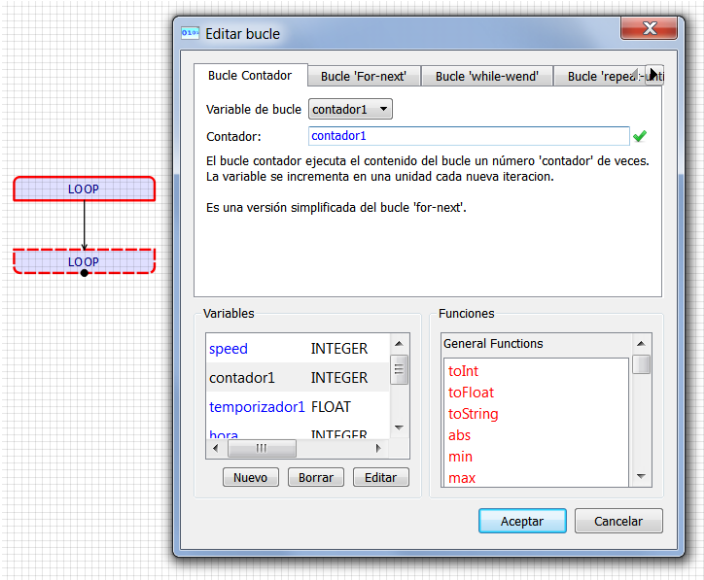
4.9.-Bloque de bucle.

El bloque de bucle es una forma sencilla de implementar dentro de nuestro flujograma las estructuras de control más utilizadas en el

lenguaje C (for-next, while y do-while). Se usa siempre por parejas (un inicio y un fin de la estructura de control) y dentro de ellas debemos colocar los bloques lógicos que deseamos ejecutar dentro del bucle en función de la evaluación de la estructura de control utilizada. El bloque de bucle nos permite utilizar cuatro diferentes tipos de estructuras de control.

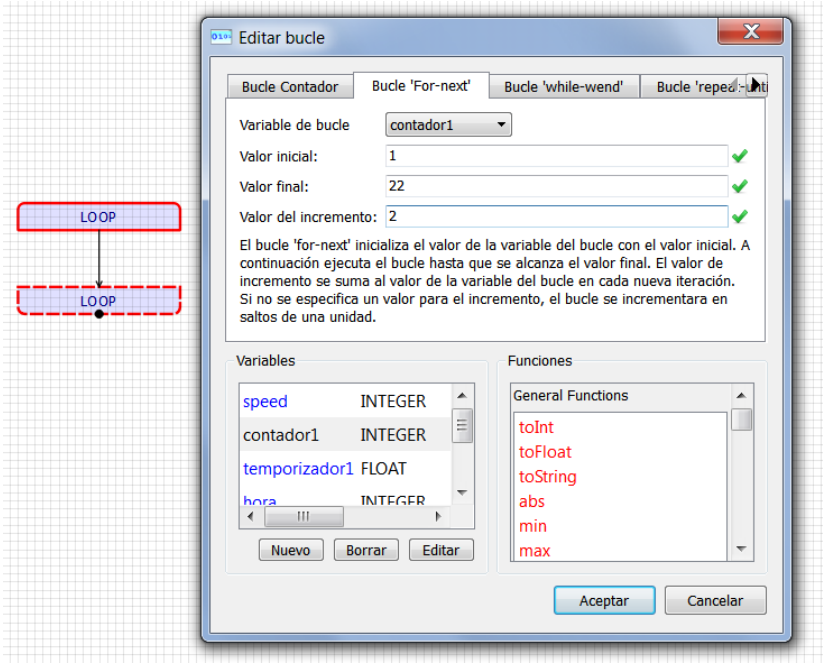
4.9.1.-Bucle tipo Contador.

Una estructura del tipo contador es la forma más sencilla de ejecutar los bloques lógicos incluidos dentro del bucle un número determinado de veces.



4.9.2.-Bucle tipo For – next.

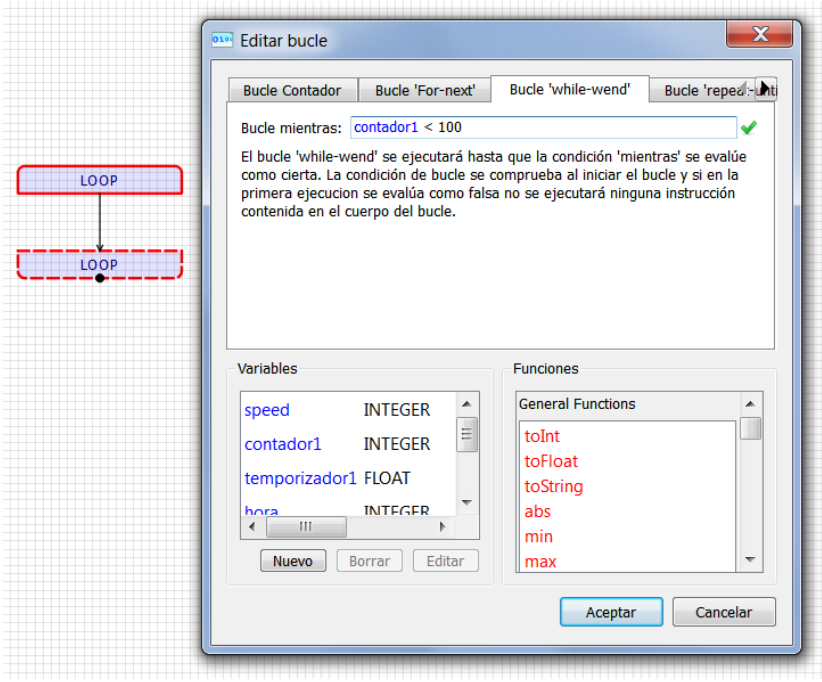
Una estructura del tipo 'bucle for-next' es una variación más compleja de la estructura contador. Nos permite especificar el valor inicial, el valor final y el tamaño del incremento del contador. De esta forma tenemos un mayor control de las veces que se ejecutará nuestro bucle y del rango de valores que asumirá la variable utilizada como contador.



4.9.3.-Bucle tipo While – wend.

Una estructura del tipo ‘while-wend’ ejecuta los bloques lógicos incluidos dentro del bucle mientras la condición de evaluación es verdad.

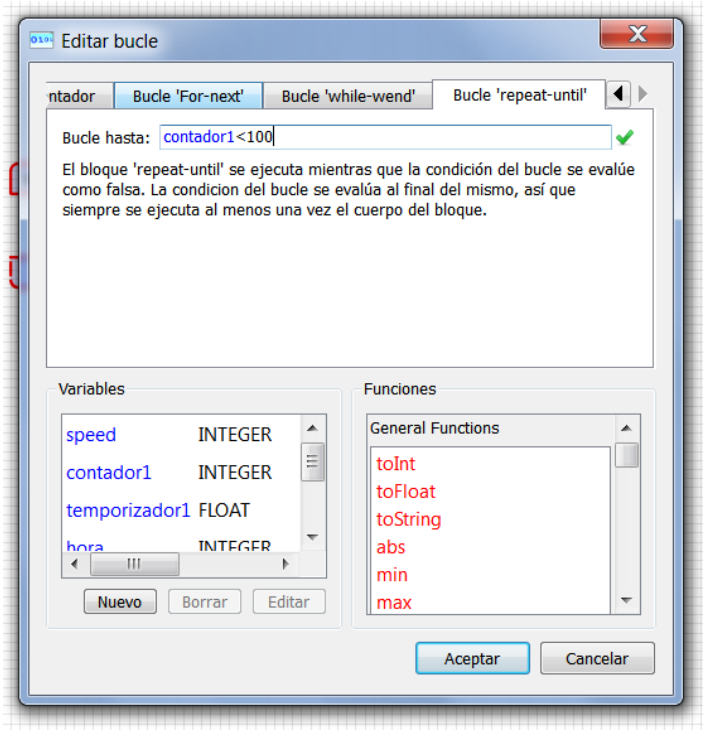
> Puesto que la evaluación se realiza al comienzo del bucle, si la primera evaluación de la condición da como resultado un ‘FALSO’ ninguno de los bloques lógicos incluidos en el bucle se ejecutarán.



4.9.4.-Bucle tipo Repeat – until.

Una estructura del tipo ‘REPEAT-UNTIL’ es muy similar a la del tipo ‘while-wend’ salvo que ejecuta los bloques lógicos incluidos dentro del bucle hasta que la condición de evaluación es falsa.

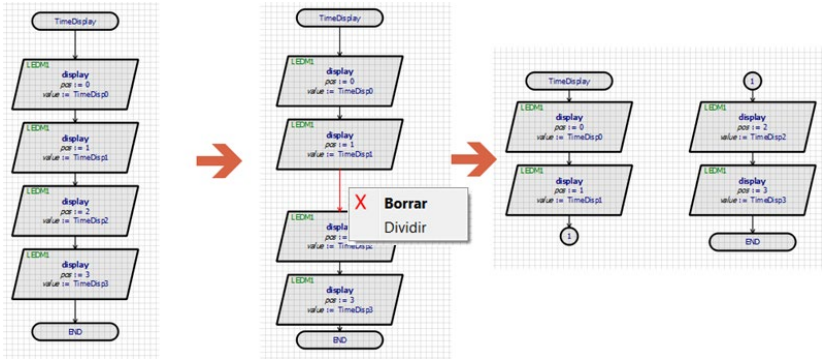
> Puesto que la evaluación se realiza al final del bucle, si la primera evaluación de la condición da como resultado un FALSO los bloques lógicos incluidos en el bucle se ejecutarán al menos una vez.



4.10.-Bloque conector.

Los bloques conector son conexiones virtuales que sirven para enlazar dos puntos del flujograma. Siempre deben ser utilizados en parejas. Cuando en un flujograma encontramos dos bloques conectores con el mismo número podemos imaginárnoslos cómo si un cable invisible los mantuviera unidos. La función de los bloques conector es permitirnos dividir el flujograma en diversas columnas sin perder la conexión entre ellas.

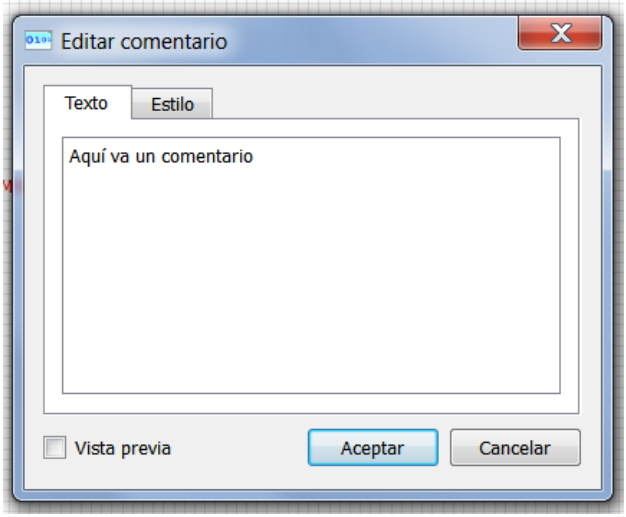
Para usarlos podemos utilizar dos métodos. Coger y arrastrar hasta el flujograma dos bloques conector y numerarlos con el mismo número. O colocar el cursor en una de las líneas que conectan dos bloques, pulsar el botón derecho del ratón y seleccionar la opción del menú contextual ‘split’.



Para suprimir dos bloques conector, juntarlos y dejar el flujograma cómo antes de la división, sólo hay que coger el primero y arrastrarlo hasta sobreponerlo al segundo.

4.11.-Bloque comentario.

El bloque comentario permite introducir un texto en formato libre dentro del flujograma para poder incluir comentarios.



5.-El diseño de los equipos virtuales.

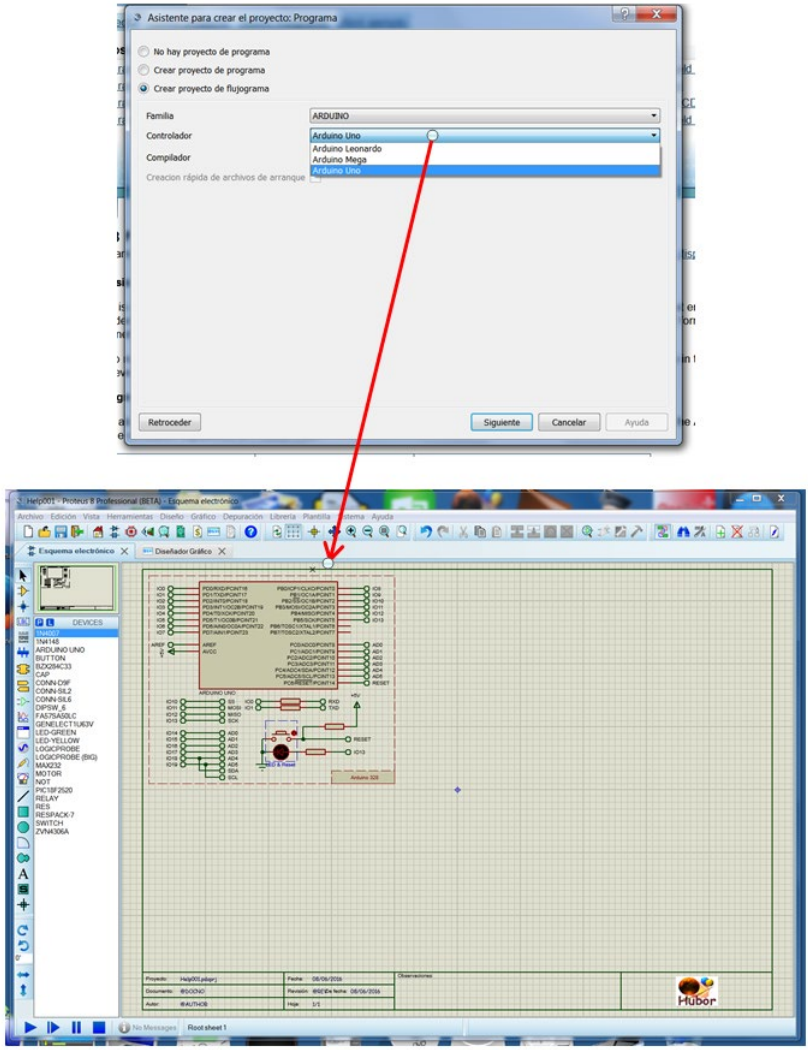
Diseñar los equipos que vamos a utilizar en nuestros proyectos de forma 'virtual' es una tarea tremendamente sencilla utilizando el Diseñador Gráfico. Las diferentes opciones que Proteus pone a nuestra disposición son descritas con todo detalle en este apartado.

5.1.-La placa base Arduino.

Desde el punto de vista del hardware, Arduino es un sistema que se construye alrededor de una placa base bastante sencilla. La placa se compone de un microcontrolador AVR, una pequeña cantidad de componentes electrónicos y unos pocos conectores del tipo 'tira de pines' macho.

Además el sistema se complementa con una gran cantidad de tarjetas de expansión (shields), que se pueden conectar a la placa base utilizando los conectores. De esta forma el sistema es muy flexible para permitir configuraciones muy diferentes en cada uno de nuestros proyectos.

Cuando trabajamos con el Diseñador Gráfico para Arduino, el momento en el que tenemos que tomar la decisión de la placa base de Arduino (Arduino Uno, Arduino Mega o Arduino Leonardo) que vamos a utilizar es, justo, al crear un proyecto. Al hacerlo, la placa base se colocará de forma automática en nuestro esquema electrónico.



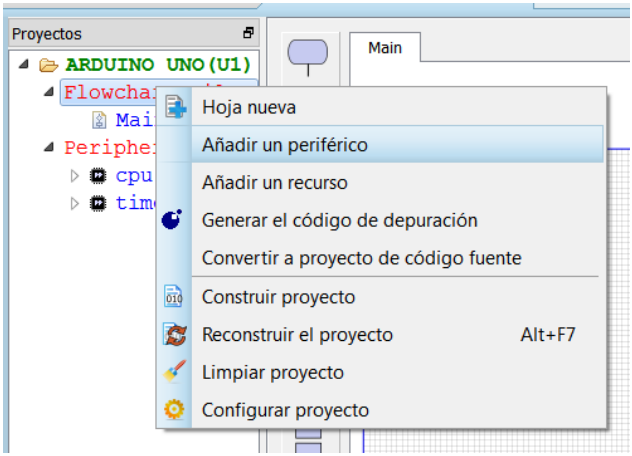
5.2.-Las tarjetas de expansión para Arduino.

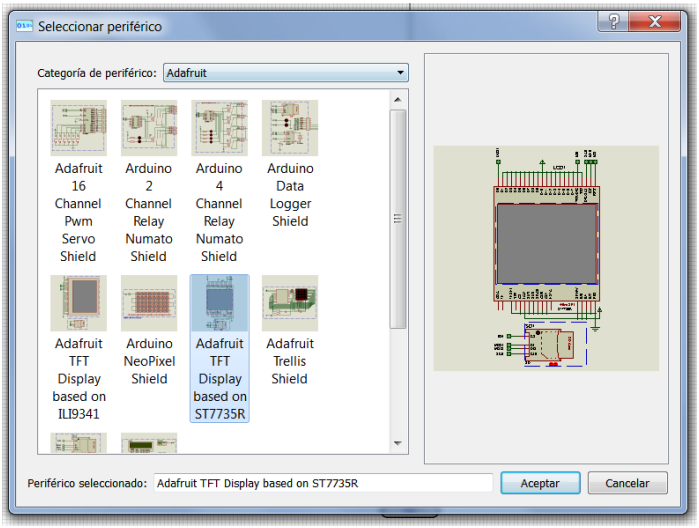
Las tarjetas de expansión son placas de circuitos electrónicos modulares que pueden ser enchufadas sobre la placa base Arduino para dotarlas con funcionalidades de muy diferente tipo.

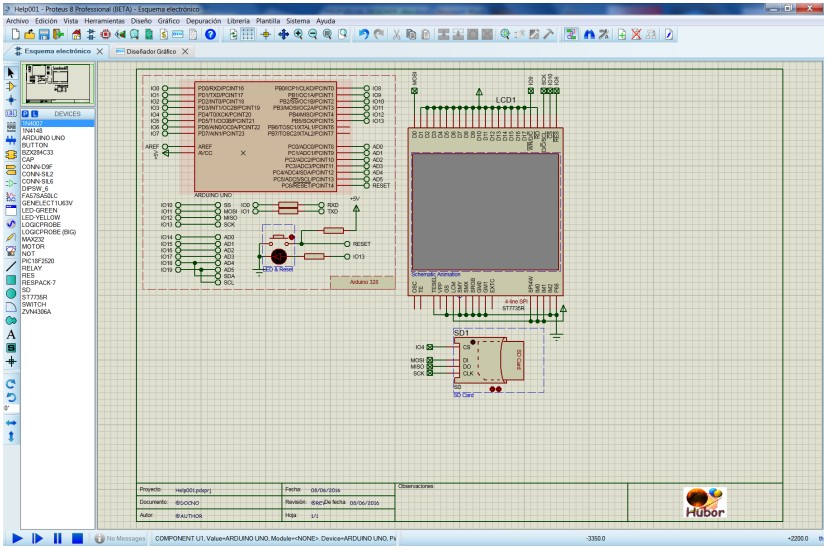
El uso de las placas de expansión es la técnica más sencilla para añadir elementos de hardware a nuestro proyecto, porque los circuitos electrónicos que las componen y el inter-conexionado de todos los elementos con la placa base se crearan en la pestaña de diseño electrónico de forma totalmente automática.

5.2.1.-Añadir un tarjeta de expansión.

La manera de añadir una tarjeta de expansión utilizando el Diseñador Gráfico es mediante una pulsación del botón derecho del ratón sobre el árbol de proyectos para que aparezca el menú contextual y seleccionar la opción 'añadir periférico'.



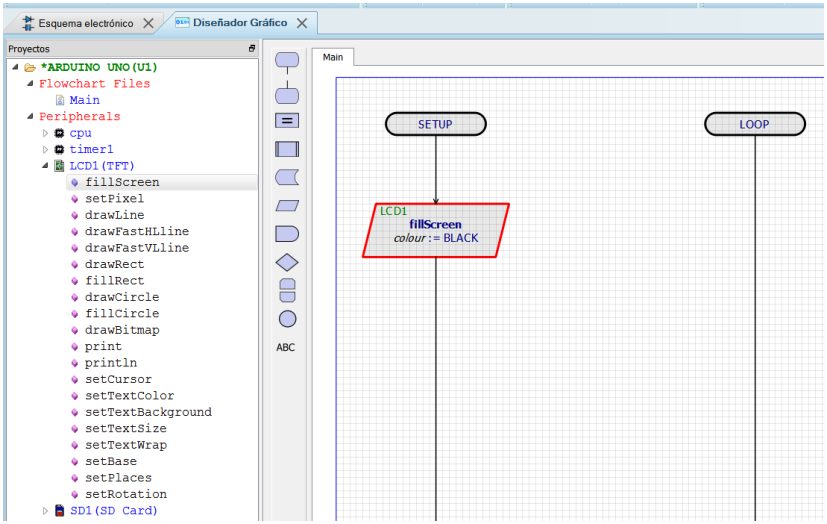




5.2.2.-Escribir el código para controlar una tarjeta de expansión.

Las tarjetas de expansión aparecen en el árbol de estructura de proyectos una vez que las hemos seleccionado desde la galería de selección de periféricos. Incluyen, de forma automática, una serie de

métodos de alto nivel que nos facilitan de forma muy sencilla el control de todos los componentes electrónicos que incluyen mediante la utilización de bloques de flujograma.



5.2.3.-Posibles dificultades al usar tarjetas de expansión.

Trabajar con tarjetas de expansión virtuales es el método más sencillo para crear de forma rápida proyectos utilizando el Diseñador Gráfico. De todas formas es necesario tener una serie de precauciones cuando seleccionamos las tarjetas de expansión que deseamos utilizar en nuestro proyecto:

- Debemos asegurarnos que estamos utilizando una tarjeta de expansión compatible con el resto del hardware que compone nuestro proyecto. Por ejemplo, algunas tarjetas de expansión son incompatibles entre sí porque utilizan un mismo recurso de la CPU (el TIMER1 por ejemplo). Otras tarjetas pueden mostrar incompatibilidad porque utilizan los mismos pines.
- Debemos asegurarnos que la placa base de Arduino que estamos utilizando en nuestro proyecto tiene suficiente memoria para controlar las tarjetas de expansión que estamos incorporando al proyecto. En concreto, cuando utilizamos la placa base Arduino Uno podemos quedarnos sin memoria libre si utilizamos tarjetas de

expansión que incorporan métodos de software que usan ‘pilas’ de almacenamiento grandes y complejas.

5.3.-Los sensores y módulos Grove para Arduino.

Grove es una plataforma modular de componentes electrónicos que permite la construcción rápida de prototipos. Cada módulo realiza una única función. Por ejemplo, disponemos de módulos que contienen un sensor de contacto, un generador de sonidos, un pulsador, un sensor de proximidad, un relé, etc. Sólo tenemos que enchufar los módulos que necesitemos a la tarjeta de expansión ‘base de enchufes’ del sistema Grove.

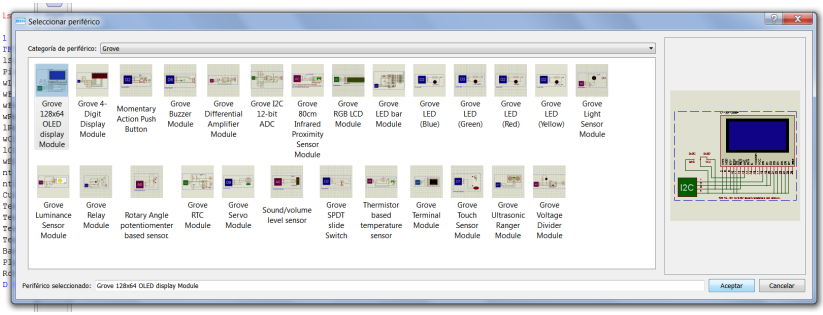
El sistema ‘Grove Starter Kit Plus’ es un magnífico camino de arranque para los estudiantes que comienzan a utilizar el sistema Arduino. La tarjeta ‘base de enchufes’ Grove, se enchufa a la placa base Arduino y nos proporciona bases estándar para enchufar en ella hasta 16 módulos Grove sin ningún problema.

Además, los conectores de la placa base Arduino todavía están disponibles para su utilización, porque la tarjeta base de enchufes Grove los reproduce, haciendo posible que podamos utilizar cualquier otra tarjeta de expansión diseñada para Arduino.



5.3.1.-Añadir un módulo Grove.

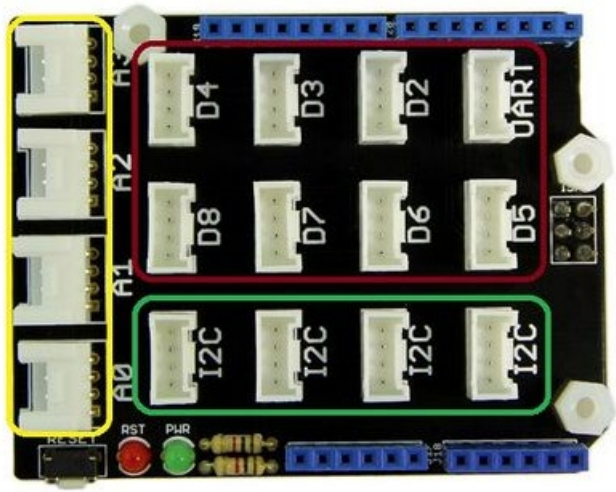
Podemos añadir a nuestro proyecto cualquier tipo de módulo Grove de la misma manera que añadíamos cualquier otra tarjeta de expansión. Sólo tenemos que seleccionar la categoría 'Grove' en la galería de selección de periféricos.



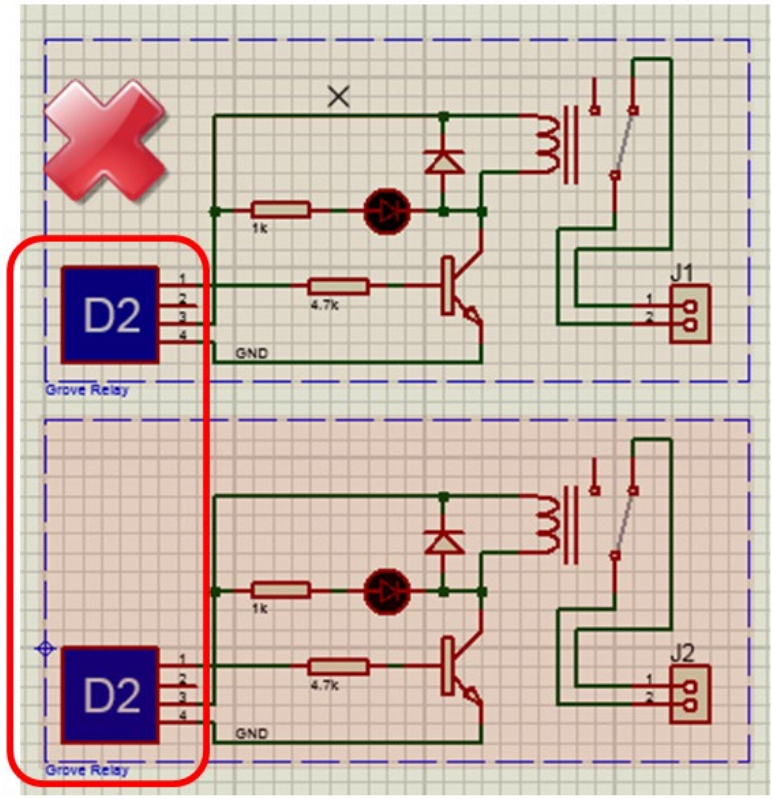
Podemos controlar el funcionamiento de los periféricos Grove con los métodos de alto nivel asociados con cada módulo Grove y que se ponen a nuestra disposición en el árbol de estructura del proyecto. De esta manera podemos añadir de forma muy sencilla múltiples sensores y actuadores de funcionalidades diferentes a nuestro proyecto. El uso de los módulos Grove proporciona un nivel extra de flexibilidad con respecto a la utilización de las placas de expansión para Arduino tradicionales.

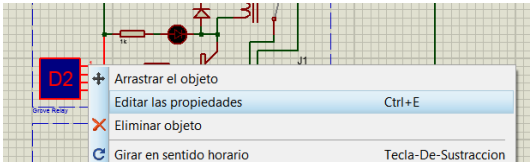
5.3.2.-Posibles dificultades al usar módulos Grove.

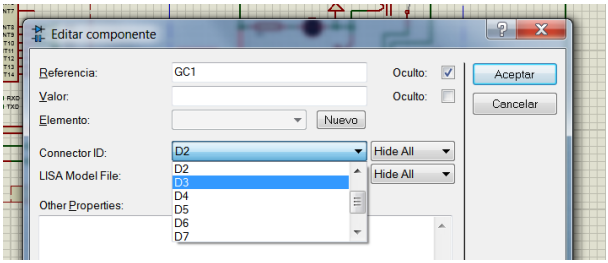
Cuando se trabaja con módulos Grove en el Diseñador Gráfico el método de trabajo es idéntico al utilizado con las tarjetas de expansión Arduino. De todas formas, puesto que el módulo base de enchufes Grove va, así mismo, enchufado en la placa base Arduino, necesitamos tener la precaución de no usar los mismos conectores del módulo base de enchufes Grove en dos módulos. El módulo base de enchufes Grove se muestra en la imagen siguiente:

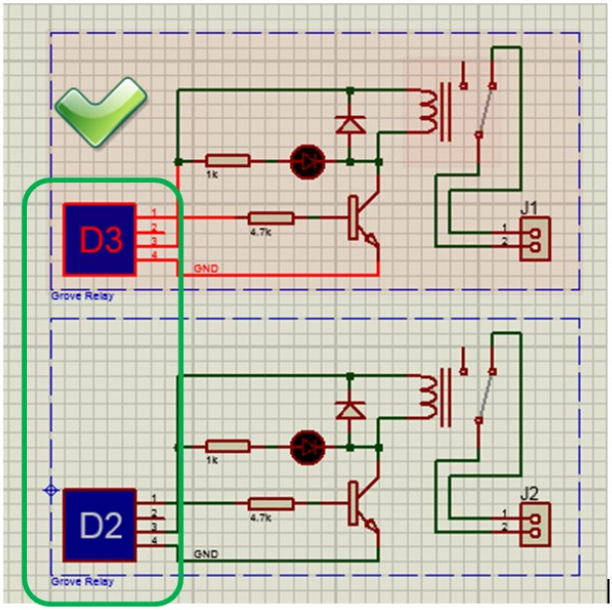


Cuando añadimos un módulo Grove utilizando el Diseñador Gráfico, debemos asignarle un Identificador (ID) que se corresponde con el enchufe del módulo base real. Si utilizamos un identificador repetido, será necesario editarlo y cambiarle el nombre por el de otro enchufe que permanezca libre.


D2
1k
4.7k
GND
J1
Grove Relay
D2
1k
4.7k
GND
J2
Grove Relay


D2
Arrastrar el objeto
Editar las propiedades Ctrl+E
Eliminar objeto
Girar en sentido horario Tecla-De-Sustraccion

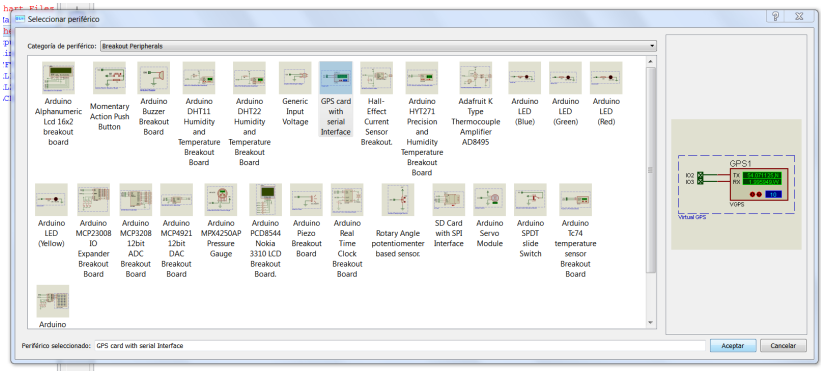

Editar componente
Referencia:
GC1
Oculto:
Aceptar
Valor:
Oculto:
Cancelar
Elemento:
Nuevo
Connector ID:
D2
Hide All
LISA Model File:
Hide All
Other Properties:
D2
D3
D4
D5
D6
D7

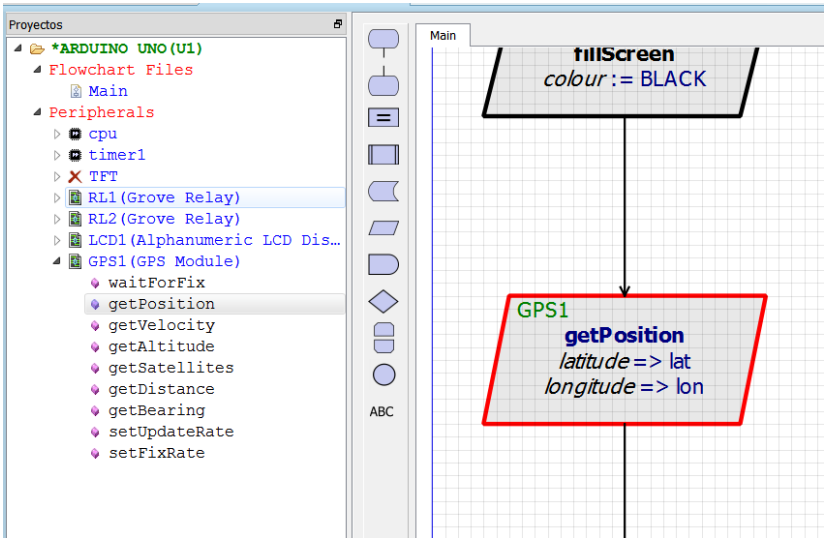


⚠ Es necesario ser muy cuidadoso y asegurarnos que utilizamos módulos Grove que sean compatibles entre sí a nivel de hardware. Puesto que varios enchufes del módulo base de enchufes comparten un mismo pin del procesador Arduino es fácil cometer un error y utilizar un mismo pin para más de un módulo al mismo tiempo. Un buen ejemplo de cómo controlar estas situaciones se explica detalladamente en el tutorial de ejemplo 2 un poco más adelante en esta misma ayuda.

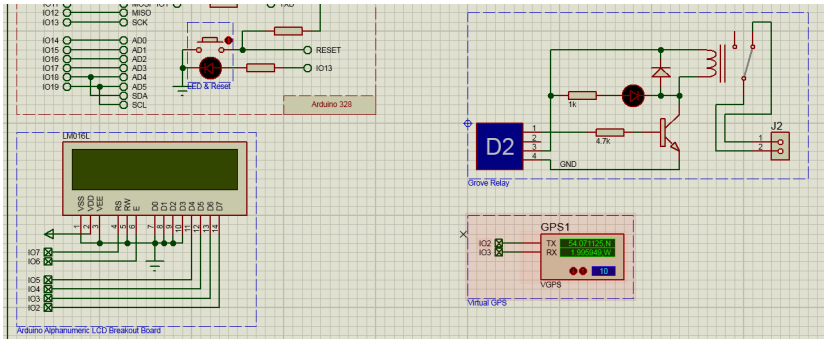
5.4.-Las tarjetas de periféricos para Arduino.

La tercera categoría que puede ser elegida en el la galería de selección de periféricos son las tarjetas de periféricos. En este apartado se engloban una serie de elementos electrónicos (muchas veces compuesto por un único elemento simple) para los que el Diseñador Gráfico facilita una serie de métodos de control. De esta manera, cuando añadimos un elemento de este tipo a nuestro árbol de estructura del proyecto podemos utilizar los métodos para gestionarlos dentro de nuestro flujograma.





Las tarjetas de periféricos proporcionan una gran flexibilidad en la elaboración del esquema electrónico. De hecho, en muchas ocasiones se pueden conectar con otros componentes de nuestro esquema que deseamos utilizar y cablear por nosotros mismos.



5.4.1.-Posibles dificultades en el uso de tarjetas de periféricos.

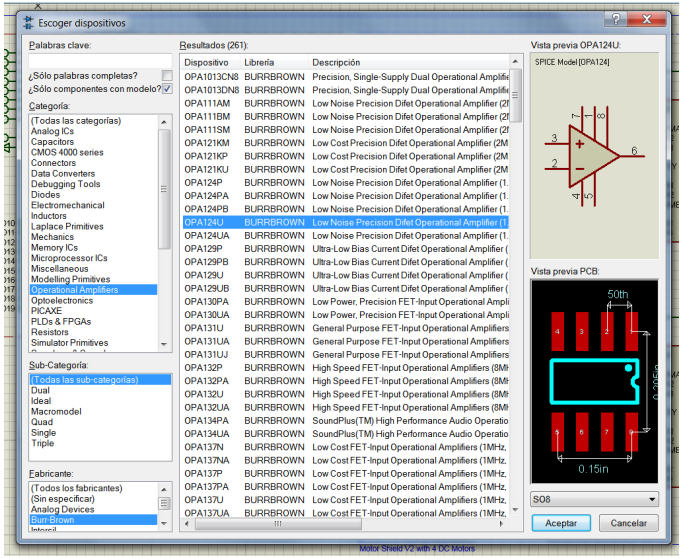
Como en los casos anteriores debemos ser cuidadosos cuando utilizamos tarjetas de periféricos de comprobar los posibles conflictos que se pueden producir si varios componentes comparten un mismo pin del procesador.

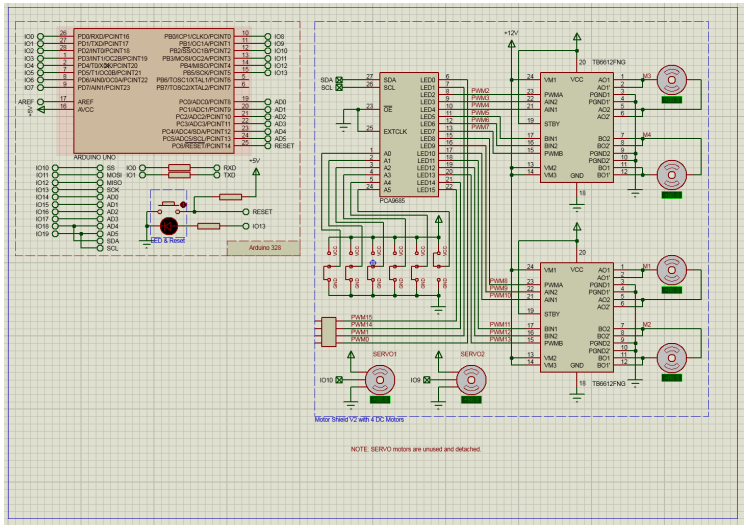
También es necesario vigilar que los componentes utilizados no desborden las capacidades de la placa base Arduino y del microprocesador que lleva instalado que estamos utilizando.

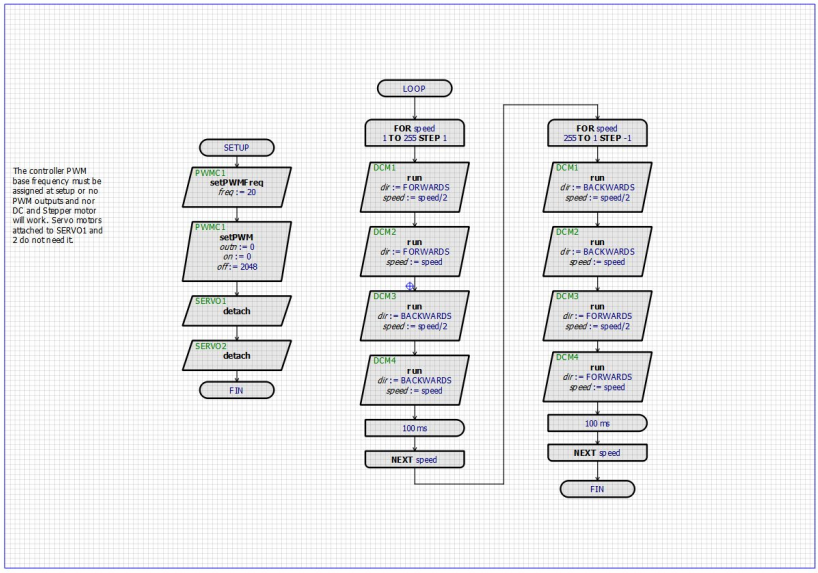
5.5.-El diseño del esquema electrónico para los equipos Arduino.

Utilizar el Diseñador Gráfico nos permite aprovecharnos de toda la potencia que Proteus pone a nuestra disposición para la completa simulación de nuestros proyectos. Entre las herramientas que componen Proteus se encuentra el editor del 'esquema electrónico' que nos permite dibujar el diagrama electrónico de nuestro proyecto. El editor pone a nuestra disposición las librerías de componentes que nos proporcionan más de diez mil dispositivos que podemos situar en nuestro esquema, cablear entre ellos y simular su funcionamiento.

Esta es la forma de diseñar proyectos de construcción de equipos electrónicos más flexible y potente con la curva de aprendizaje más rápida de todos los sistemas disponibles en el mercado.


Escoger dispositivos
Palabras clave:
¿Sólo palabras completas?
¿Sólo componentes con modelo?
Categoría:
(Todas las categorías)
Analog ICs
Capacitors
CMOS 4000 series
Connectors
Data Converters
Debugging Tools
Diodes
Electromechanical
Inductors
Laplace Primitives
Mechanics
Memory ICs
Microprocessor ICs
Miscellaneous
Modelling Primitives
Operational Amplifiers
Optoelectronics
PICAXE
PLDs & FPGAs
Resistors
Simulator Primitives
Sub-Categoría:
(Todas las sub-categorías)
Dual
Ideal
Macromodel
Quad
Single
Triple
Fabricante:
(Todos los fabricantes)
(Sin especificar)
Analog Devices
Burr-Brown
Resultados (261):
Dispositivo
Librería
Descripción
OPA1013CN8 BURRBROWN Precision, Single-Supply Dual Operational Amplifie
OPA1013DN8 BURRBROWN Precision, Single-Supply Dual Operational Amplifie
OPA111AM BURRBROWN Low Noise Precision Difet Operational Amplifier (2f
OPA111BM BURRBROWN Low Noise Precision Difet Operational Amplifier (2f
OPA111SM BURRBROWN Low Noise Precision Difet Operational Amplifier (2f
OPA121KM BURRBROWN Low Cost Precision Difet Operational Amplifier (2M
OPA121KP BURRBROWN Low Cost Precision Difet Operational Amplifier (2M
OPA121KU BURRBROWN Low Cost Precision Difet Operational Amplifier (2M
OPA124P BURRBROWN Low Noise Precision Difet Operational Amplifier (1.
OPA124PA BURRBROWN Low Noise Precision Difet Operational Amplifier (1.
OPA124PB BURRBROWN Low Noise Precision Difet Operational Amplifier (1.
OPA124U BURRBROWN Low Noise Precision Difet Operational Amplifier (1
OPA124UA BURRBROWN Low Noise Precision Difet Operational Amplifier (1.
OPA129P BURRBROWN Ultra-Low Bias Current Difet Operational Amplifier (
OPA129PB BURRBROWN Ultra-Low Bias Current Difet Operational Amplifier (
OPA129U BURRBROWN Ultra-Low Bias Current Difet Operational Amplifier (
OPA129UB BURRBROWN Ultra-Low Bias Current Difet Operational Amplifier (
OPA130PA BURRBROWN Low Power, Precision FET-Input Operational Ampli
OPA130UA BURRBROWN Low Power, Precision FET-Input Operational Ampli
OPA131U BURRBROWN General Purpose FET-Input Operational Amplifiers
OPA131UA BURRBROWN General Purpose FET-Input Operational Amplifiers
OPA131UJ BURRBROWN General Purpose FET-Input Operational Amplifiers
OPA132P BURRBROWN High Speed FET-Input Operational Amplifiers (8M
OPA132PA BURRBROWN High Speed FET-Input Operational Amplifiers (8M
OPA132U BURRBROWN High Speed FET-Input Operational Amplifiers (8M
OPA132UA BURRBROWN High Speed FET-Input Operational Amplifiers (8M
OPA134PA BURRBROWN SoundPlus(TM) High Performance Audio Operatio
OPA134UA BURRBROWN SoundPlus(TM) High Performance Audio Operatio
OPA137N BURRBROWN Low Cost FET-Input Operational Amplifiers (1MHz,
OPA137NA BURRBROWN Low Cost FET-Input Operational Amplifiers (1MHz,
OPA137P BURRBROWN Low Cost FET-Input Operational Amplifiers (1MHz,
OPA137PA BURRBROWN Low Cost FET-Input Operational Amplifiers (1MHz,
OPA137U BURRBROWN Low Cost FET-Input Operational Amplifiers (1MHz,
OPA137UA BURRBROWN Low Cost FET-Input Operational Amplifiers (1MHz,
Vista previa OPA124U:
SPICE Model [OPA124]
Vista previa PCB:
50th
0.15in
SO8
Aceptar
Cancelar
Motor Shield V2 with 4 DC Motors


ARDUINO UNO
Arduino 328
LED & Reset
PCA9685
Motor Shield V2 with 4 DC Motors
NOTE: SERVO motors are unused and detached.



5.5.1.-Posibles dificultades en el diseño del esquema electrónico.

El diseño de esquemas electrónicos para simulación es una técnicamente relativamente compleja. El aprendizaje va mucho más allá de los procedimientos básicos de colocación e interconexión de los componentes electrónicos.

Hay cuestiones, por ejemplo, relativas a la utilización de ciertos protocolos de comunicación que exigen el conocimiento de técnicas más complejas que hay que conocer apropiadamente para poder definir adecuadamente la manera en que intercambian datos los diversos equipos electrónicos dotados de capacidad lógica.

En este punto es necesario mencionar que la transición para pasar de utilizar el Diseñador Gráfico a la utilización del sistema VSM (taller de modelado virtual de microcontroladores) se debe entender como un salto cualitativo dónde los conocimientos de programación en lenguaje C son imprescindibles para la construcción de sistemas embebidos de control realmente efectivos.

5.6.-La placa base Raspberry Pi.

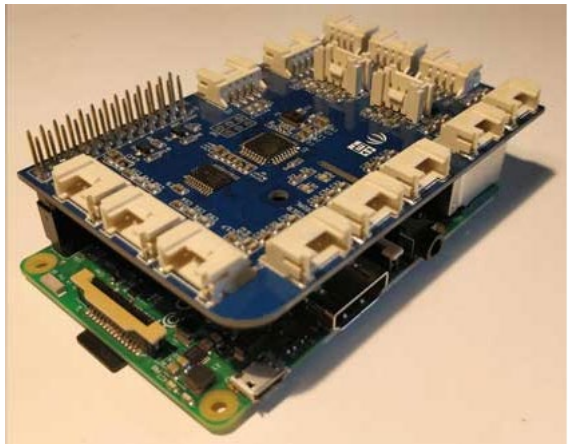
Desde el punto de vista del hardware, todos los proyectos basados en Raspberry Pi arrancan con la tarjeta base y, según los casos, puede completarse utilizando electrónica adicional en forma de placas enchufables (header blocks). Existen dos formas básicas de ampliar nuestra placa base Raspberry Pi:

Las tarjetas de expansión son bloques que llevan a cabo una funcionalidad completa como, por ejemplo, una pantalla LCD, un bloque de relés, etc. En este caso en el mundo Pi se les conoce con el nombre de 'hats'.

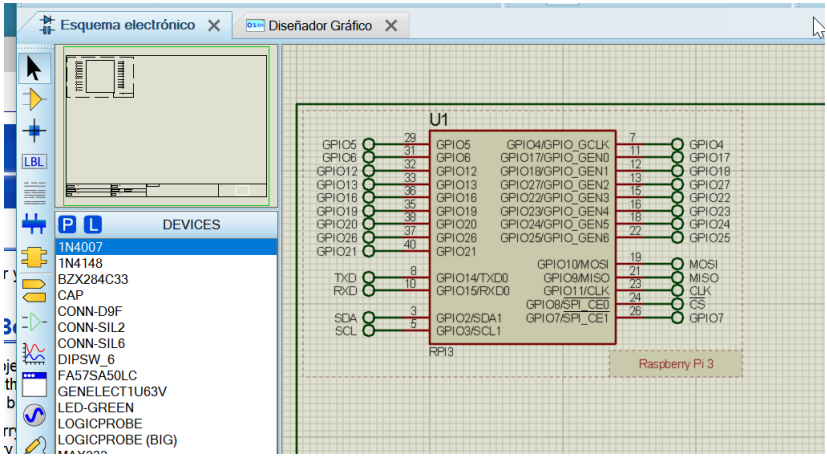
Los módulos son bloques que contienen componentes electrónicos individuales como un led, un relé o un pulsador.

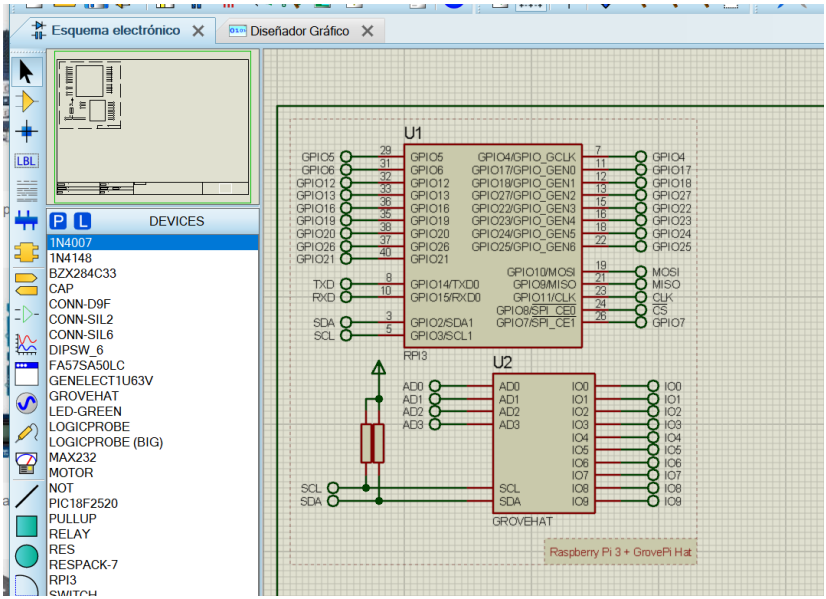
La decisión sobre la tarjeta base que deseamos utilizar se debe tomar al principio, en el momento de creación de nuestro proyecto para Raspberry Pi usando el Diseñador Gráfico. Es posible seleccionar entre una placa Raspberry Pi 3 o Raspberry Pi 3 con Grove. Esta segunda opción equivale a utilizar en la vida real una placa base Raspberry Pi con la tarjeta de expansión Grove conectada a ella.





Nuestra elección se reflejará automáticamente en la pestaña esquema electrónico donde aparecerá la placa base seleccionada lista para su uso en nuestro proyecto.





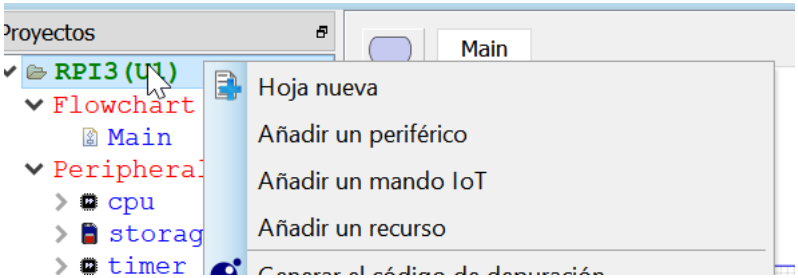
> En principio, la simulación de la placa Raspberry Pi Zero W debe funcionar perfectamente ya que el mapa de pines es idéntico más el chip para controlar las comunicaciones vía WiFi. Si deseamos trabaja con esta placa, debemos seleccione Pi 3 en el momento de configurar nuestro proyecto. Sin embargo, la simulación de la placa Raspberry Pi 2 no funcionará con Visual Designer porque el mapeado de pines es completamente diferente.

5.7.-Las tarjetas de expansión (hats) para Raspberry Pi.

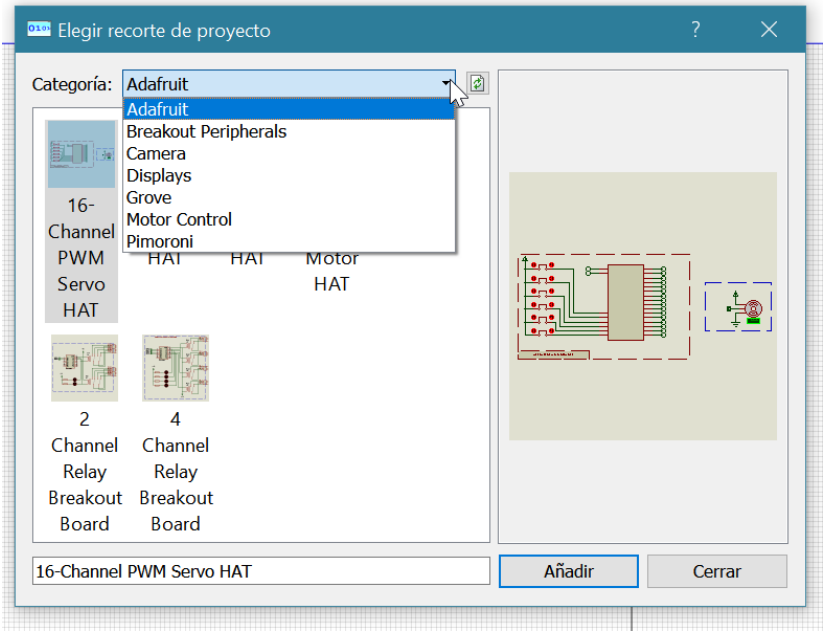
Las tarjetas de expansión Raspberry Pi son placas modulares con diseños electrónicos que complementan con nuevas funcionalidades a las placas base del tipo Raspberry Pi 3. El uso de estas tarjetas es la forma más sencilla de añadir nuevas funcionalidades de hardware a nuestros proyectos, porque tanto los nuevos circuitos como sus conexiones se enlazan de forma automática y aparecen correctamente en nuestro proyecto.

5.7.1.-Añadir una tarjeta de expansión.

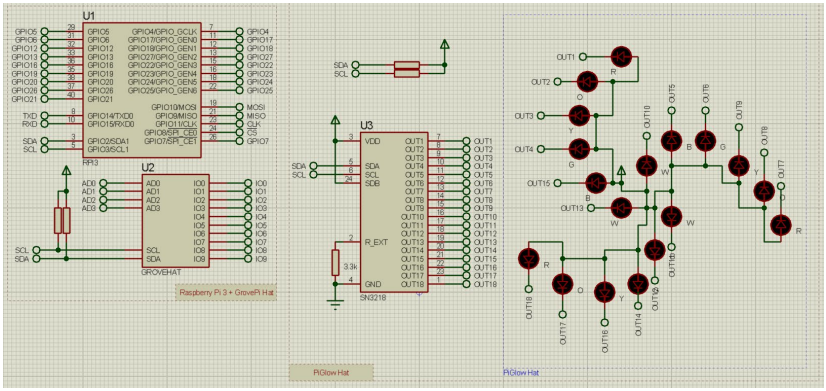
La manera de añadir una tarjeta de expansión utilizando el Diseñador Gráfico es mediante una pulsación del botón derecho del ratón sobre el árbol de proyectos para que aparezca el menú contextual y seleccionar la opción ‘añadir periférico’.



En la ventana de diálogo que aparece podemos elegirla categoría que deseemos (Adafruti, Pimoroni, etc.) y seleccionar luego la tarjeta de expansión que deseemos añadir a nuestro proyecto.



En cuanto la seleccionemos y utilicemos el botón ‘Añadir’ la tarjeta de expansión se añadirá de forma automática en la pestaña esquema electrónico y se generarán todas las conexiones necesarias con la placa base Raspberry Pi. En la imagen siguiente se muestra un ejemplo de utilización de la tarjeta de expansión “Pi Grow Hat” que se encuentra situada en la categoría ‘Pimoroni’.

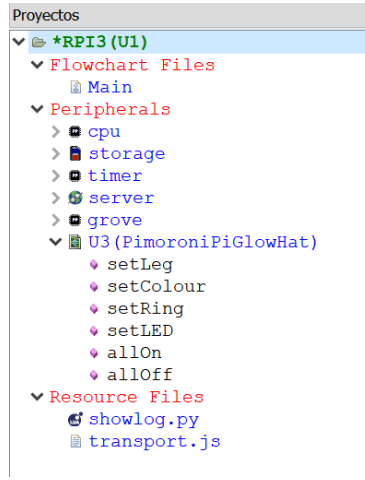


Las conexiones entre las tarjetas de expansión y la placa base Raspberry se realizan utilizando terminales del mismo nombre. En Proteus dos terminales que tengan el mismo nombre funcionan, independientemente del lugar del esquema donde los coloquemos, como si estuvieran unidos por un cable invisible y los dos puntos se encuentran eléctricamente conectados.

5.7.2.-Escribir el código para controlar una tarjeta de expansión.

La tarjeta de expansión Raspberry Pi aparecerá automáticamente en el árbol de estructura del proyecto de la pestaña Diseñador Gráfico una vez que la hayamos seleccionado desde la galería de periféricos disponible. Ya tendremos todos los métodos de alto nivel disponibles para manejar la tarjeta de expansión mediante el uso de los bloques de flujograma.

En nuestro ejemplo, la tarjeta de expansión se muestra en la rama U3 (PimoroniPiGlowHat) y muestra los seis métodos disponibles para su manejo.



5.7.3.-Posibles dificultades al utilizar tarjetas de expansión.

Trabajar con tarjetas de expansión es el método más sencillo para crear de forma rápida proyectos utilizando el Diseñador Gráfico. De todas formas es necesario tener una serie de precauciones cuando seleccionamos las tarjetas de expansión que deseamos utilizar en nuestro proyecto:

- Debemos asegurarnos que estamos utilizando una tarjeta de expansión compatible con el resto del hardware que compone nuestro proyecto. Por ejemplo, algunas tarjetas de expansión son incompatibles entre sí porque utilizan un mismo recurso de la CPU (el TIMER1, por ejemplo). Otras tarjetas pueden mostrar incompatibilidad porque utilizan los mismos pines.

5.8.-Los sensores y módulos Grove para Raspberry Pi.

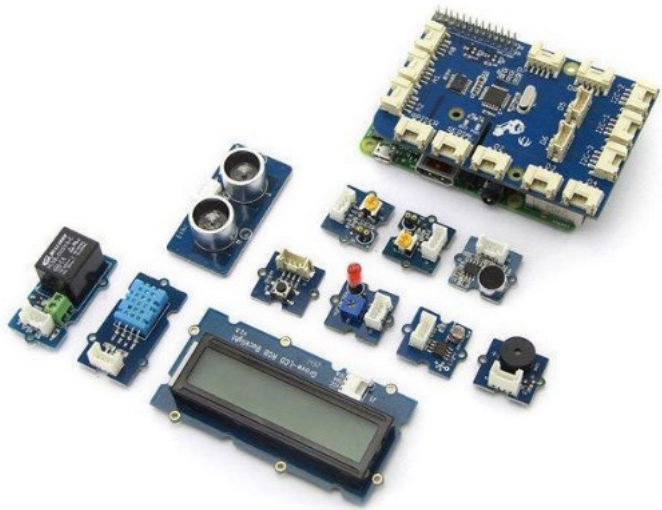
Grove es una plataforma modular de componentes electrónicos que permite la construcción rápida de prototipos. Cada módulo realiza una única función. Por ejemplo, disponemos de módulos que contienen un sensor de contacto, un generador de sonidos, un pulsador, un sensor de proximidad, un relé, etc. Sólo tenemos que

enchufar los módulos que necesitemos a la tarjeta de expansión ‘Grove Pi+’ del sistema Grove.

La tarjeta de expansión ‘Grove Pi+’ se puede enchufar sobre la placa base Raspberry Pi 3 y proporciona 15 enchufes de cuatro pines para conectar los diferentes módulos de sensores Grove directamente a nuestra placa base.

> Recuerde que si quiere utilizar en el Diseñador Gráfico la placa base Raspberry Pi 3 con la tarjeta de expansión Grove Pi+, es necesario seleccionar durante el proceso de creación de un nuevo proyecto, la tarjeta base ‘Raspberry Pi 3 with Grove’ que incluye de forma automática en nuestro proyecto las dos cosas: la placa base y la tarjeta de expansión Grove Pi+.

El sistema ‘Grove Pi+’ es un magnífico camino de arranque para los estudiantes que comienzan a utilizar el sistema Raspberry Pi. La tarjeta ‘base de enchufes’ Grove, se enchufa a la placa base Raspberry y nos proporciona bases estándar para enchufar en ella hasta 15 módulos Grove sin ningún problema.

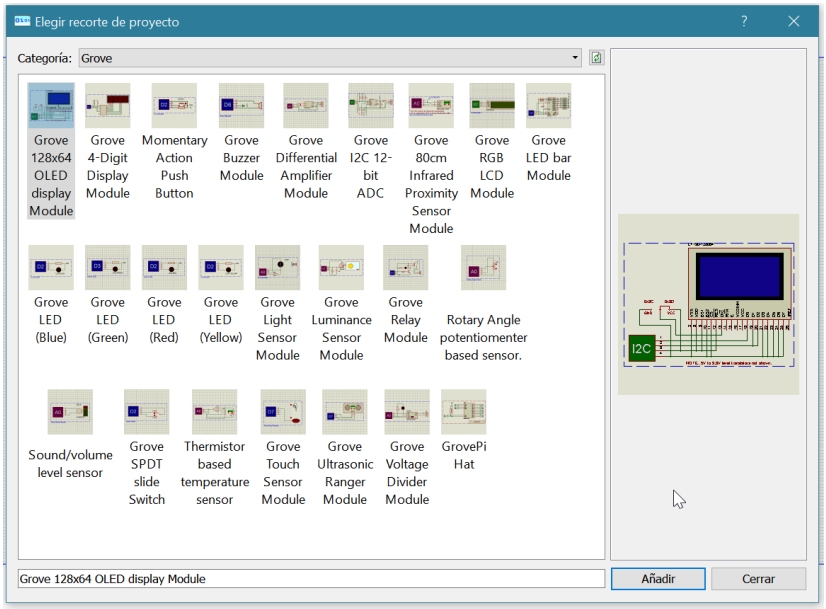


Además, los conectores de la placa base Raspberry todavía están disponibles para su utilización, porque la tarjeta base de enchufes

Grove los reproduce, haciendo posible que podamos utilizar cualquier otra tarjeta de expansión diseñada para Raspberry.

5.8.1.-Añadir un módulo Grove.

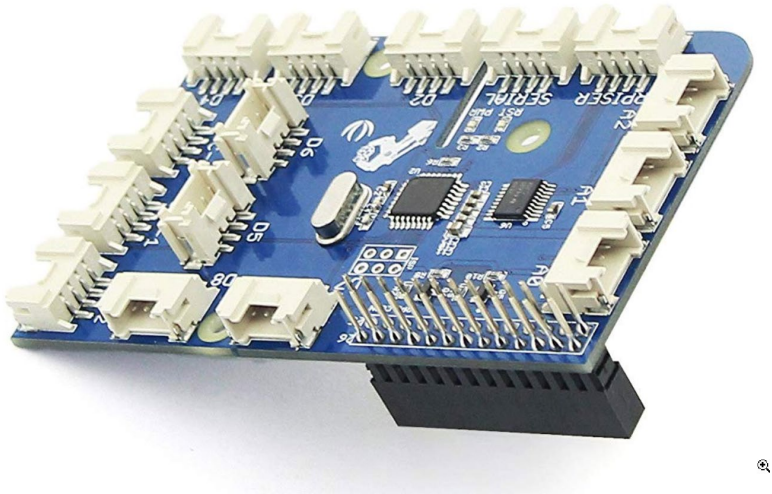
Podemos añadir a nuestro proyecto cualquier tipo de módulo Grove de la misma manera que añadíamos cualquier otra tarjeta de expansión. Sólo tenemos que seleccionar la categoría 'Grove' en la galería de selección de periféricos.



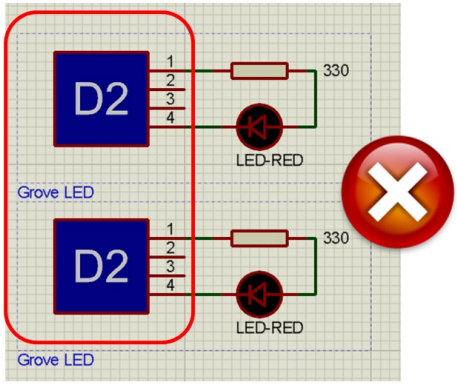
Podemos controlar el funcionamiento de los periféricos Grove con los métodos de alto nivel asociados con cada módulo Grove que se ponen a nuestra disposición en el árbol de estructura del proyecto. De esta manera podemos añadir de forma muy sencilla múltiples sensores y actuadores de funcionalidades diferentes a nuestro proyecto. El uso de los módulos Grove proporciona un nivel extra de flexibilidad con respecto a la utilización de las placas de expansión para Raspberry Pi tradicionales.

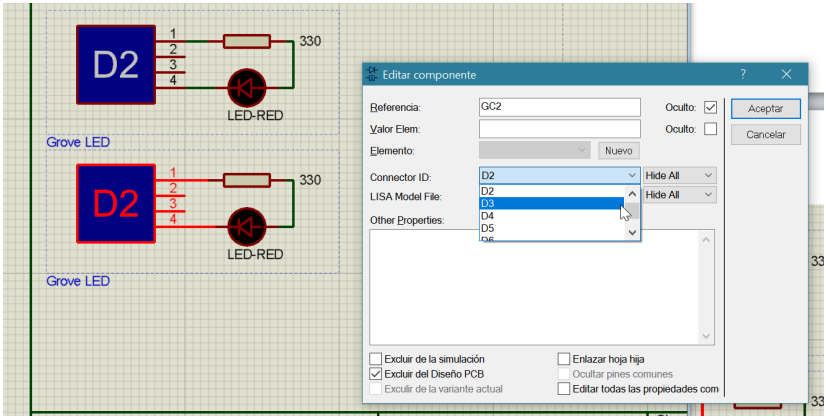
5.8.2.-Posibles dificultades al usar módulos Grove.

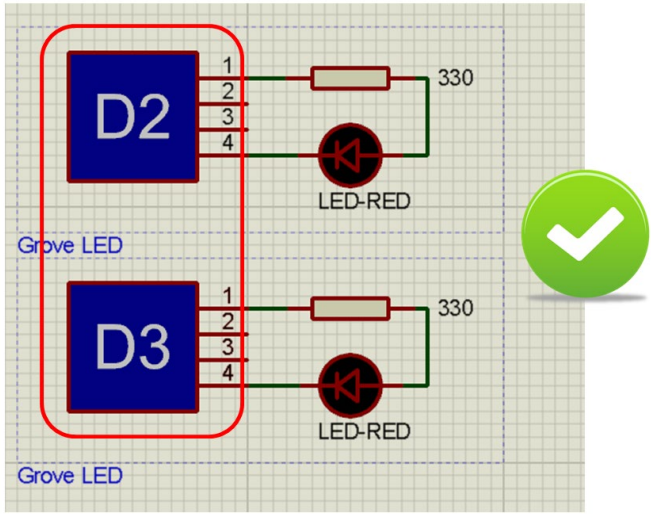
Cuando se trabaja con módulos Grove en el Diseñador Gráfico el método de trabajo es idéntico al utilizado con las tarjetas de expansiónn. De todas formas, puesto que el módulo base de enchufes Grove va, así mismo, enchufado en la placa base Raspberry Pi, necesitamos tener la precaución de no usar los mismos conectores del módulo base de enchufes Grove en dos módulos. El módulo base de enchufes Grove se muestra en la imagen siguiente:



Cuando añadimos un módulo Grove utilizando el Diseñador Gráfico, debemos asignarle un Identificador (ID) que se corresponde con el enchufe del módulo base real. Si utilizamos un identificador repetido, será necesario editarlo y cambiarle el nombre por el de otro enchufe que permanezca libre.


D2
1
2
3
4
330
LED-RED
Grove LED
D2
1
2
3
4
330
LED-RED
Grove LED

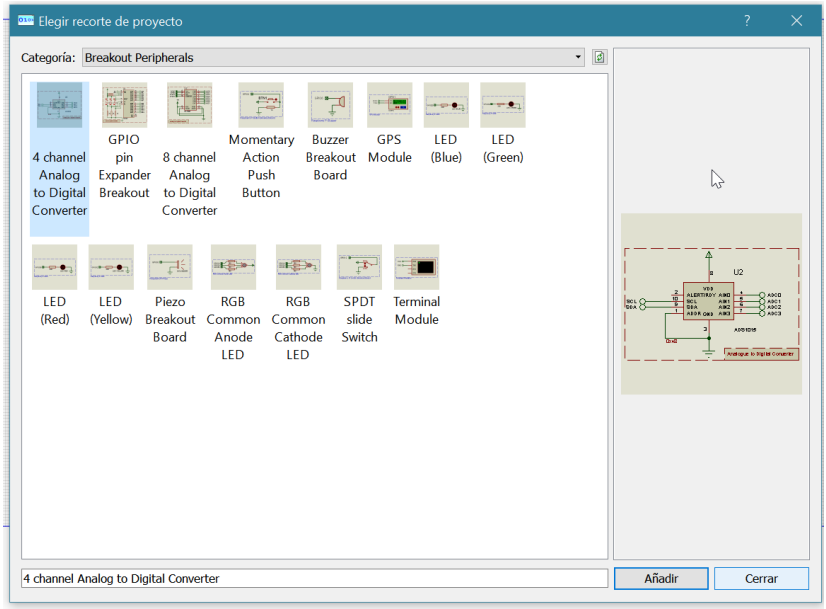

D2
1
2
3
4
330
LED-RED
Grove LED
D2
1
2
3
4
330
LED-RED
Grove LED
Editar componente
Referencia:
GC2
Oculto:
Aceptar
Valor Elem:
Oculto:
Cancelar
Elemento:
Nuevo
Connector ID:
D2
Hide All
LISA Model File:
Hide All
Other Properties:
D2
D3
D4
D5
Excluir de la simulación
Excluir del Diseño PCB
Excluir de la variante actual
Enlazar hoja hija
Ocultar pines comunes
Editar todas las propiedades com

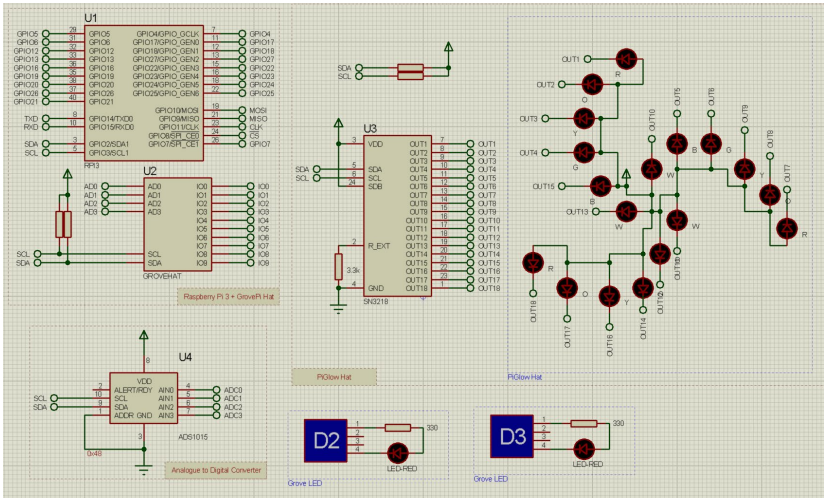


⚠ Es necesario ser muy cuidadoso y asegurarnos que utilizamos módulos Grove que sean compatibles entre sí a nivel de hardware. Puesto que varios enchufes del módulo base de enchufes comparten un mismo pin del procesador Arduino es fácil cometer un error y utilizar un mismo pin para más de un módulo al mismo tiempo.

5.9.-Las tarjetas de periféricos para Raspberry Pi.

La tercera categoría que puede ser elegida en el la galería de selección de periféricos son las tarjetas de periféricos. En este apartado se engloban una serie de elementos electrónicos (muchas veces compuesto por un único elemento simple) para los que el Diseñador Gráfico facilita una serie de métodos de control. De esta manera, cuando añadimos un elemento de este tipo a nuestro árbol de estructura del proyecto podemos utilizar los métodos para gestionarlos dentro de nuestro flujograma.





Las tarjetas de periféricos proporcionan una gran flexibilidad en la elaboración del esquema electrónico. De hecho, en muchas ocasiones se pueden conectar con otros componentes de nuestro esquema que deseamos utilizar y cablear por nosotros mismos.

5.9.1.-Posibles dificultades en el uso de tarjetas de periféricos.

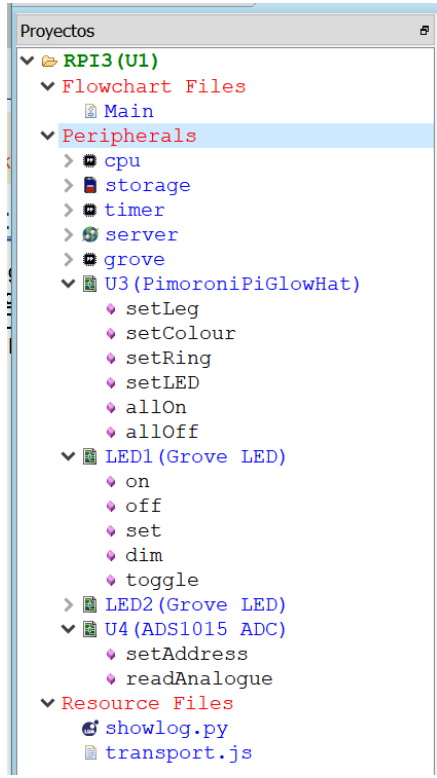
Como en los casos anteriores debemos ser cuidadosos cuando utilizamos tarjetas de periféricos de comprobar los posibles conflictos que se pueden producir si varios componentes comparten un mismo pin del procesador.

También es necesario vigilar que los componentes utilizados no desborden las capacidades de la placa base Raspberry Pi.

5.10.-El diseño del esquema electrónico para los equipos Raspberry Pi.

Utilizar el Diseñador Gráfico nos permite aprovecharnos de toda la potencia que Proteus pone a nuestra disposición para la completa simulación de nuestros proyectos. Entre las herramientas que componen Proteus se encuentra el editor del 'esquema electrónico' que nos permite dibujar el diagrama electrónico de nuestro proyecto. El editor pone a nuestra disposición las librerías de componentes que nos proporcionan más de diez mil dispositivos que podemos situar en nuestro esquema, cablear entre ellos y simular su funcionamiento.

El diseñador gráfico es el método más flexible de todos los existentes que posibilita las curvas de aprendizaje más cortas. La programación se lleva a cabo utilizando los métodos de alto nivel disponibles para el control tanto de la CPU como de los diferentes periféricos que hayamos añadido a nuestro proyecto.



5.10.1.-Posibles dificultades en el diseño del esquema electrónico.

El diseño de esquemas electrónicos para simulación es una técnicamente relativamente compleja. El aprendizaje va mucho más allá de los procedimientos básicos de colocación e interconexión de los componentes electrónicos.

Hay cuestiones, por ejemplo, relativas a la utilización de ciertos protocolos de comunicación que exigen el conocimiento de técnicas más complejas que hay que conocer apropiadamente para poder definir adecuadamente la manera en que intercambian datos los diversos equipos electrónicos dotados de capacidad lógica.

Además es muy importante tener en cuenta que el modelo de simulación de equipos Raspberry Pi utilizado por Proteus es del tipo

modelo de simulación de comportamiento de alto nivel. Puesto que los equipos Raspberry Pi utilizan una copia completa del sistema operativo Linux y pueden ejecutar una enorme cantidad de código arbitrario, es imposible llevar a cabo un modelo de simulación completo al estilo, por ejemplo, del que se utiliza con los microprocesadores PIC. En el apartado 1.3 de esta misma ayuda encontrará la lista completa de librerías compatibles.

6.-Las técnicas de simulación y depuración.

Cuando se desarrollan los sistemas de control embebido es muy habitual querer probar el funcionamiento de nuestro código a medida que lo vamos escribiendo. Y, sin duda, siempre se desea poder hacerlo una vez que el código está totalmente terminado antes de pasarlo a producción.

Es rarísima la ocasión en la que todo funciona perfectamente a la primera en un proyecto. Por este motivo, surge la necesidad de disponer de potentes herramientas para depurar nuestro código, encontrar los errores y buscarles solución.

El Diseñador Gráfico permite que la simulación y la depuración sea un proceso extremadamente simple porque todo ocurre dentro de la misma herramienta de software sin necesidad de ningún hardware. Este apartado recoge las técnicas esenciales para poder simular, depurar y medir valores dentro de nuestros proyectos utilizando El Diseñador Gráfico.

A lo largo de todo este apartado asumimos que el lector está familiarizado con todas las técnicas de edición de esquemas electrónicos y de código disponibles en Proteus y que ya disponemos de un proyecto listo para proceder a probarlo.

6.1.-Las técnicas de simulación.

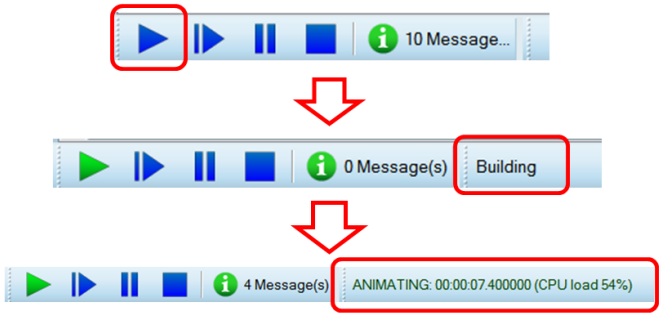
Cuando ya hemos añadido los periféricos situados alrededor de nuestro controlador Arduino y enlazado dentro de nuestro flujograma los diferentes bloques de código necesarios para su gobierno, hemos terminado de construir nuestro sistema de control embebido. En términos generales, ya podemos decir que disponemos de nuestra placa controladora y de los periféricos que se sitúan a su alrededor.

La capacidad de ejecutar la simulación de todo el sistema que acabamos de construir, tanto del microprocesador como de todos los diferentes componentes digitales o analógicos situados a su

alrededor, es lo que realmente hace de Proteus una herramienta increíblemente potente. Proteus nos permite probar el funcionamiento y depurar nuestro proyecto sin necesidad de otras herramientas complementarias.

6.1.1.-Arrancar la simulación.

Podemos poner en marcha ya nuestra simulación utilizando el botón 'Play' de los controles de simulación de la zona inferior de la pantalla o utilizando el atajo de teclado F12. El código será compilado y la simulación arrancará. En la barra de estado podemos comprobar la información sobre el progreso de la operación.



6.1.2.-Detener la simulación.

Para detener la simulación tenemos que utilizar el botón 'Stop' de los controles de simulación de la zona inferior de la pantalla o utilizando el atajo de teclado ESCAPE.



La simulación se detendrá y El Diseñador Gráfico mostrará de nuevo la pestaña desde la que lanzamos la simulación.

6.1.3.-Poner la simulación en pausa.

Poner la simulación en pausa es un concepto importante del sistema de simulación de Proteus que conviene entender bien, porque la situación es muy diferente a cuándo utilizamos las herramientas de depuración sobre una placa de desarrollo en el mundo real.

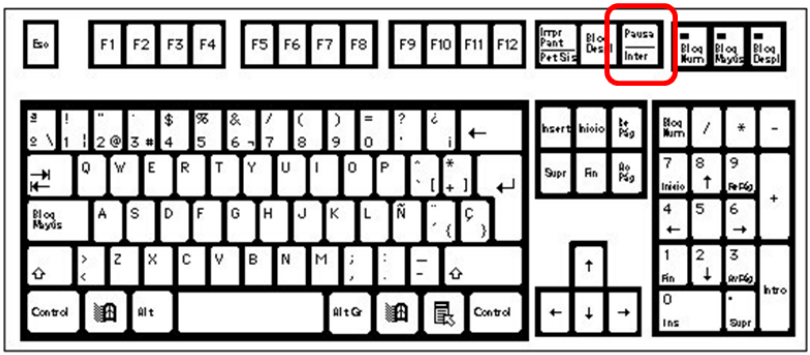
En Proteus, literalmente, es cómo si estuviéramos deteniendo el tiempo. Por ejemplo, los condensadores no se descargan durante la pausa y los motores mantienen sin variación su posición angular o su par motor (torque).

De esta forma se posibilita la inspección tanto del código como de los componentes de hardware en un momento dado con la situación 'congelada' en el tiempo. Y si ordenamos depurar el proyecto utilizando la opción 'Saltar hasta la línea de código (over) en la función o subrutina' todo el sistema avanzará en el tiempo de forma sincronizada, tanto el software como el hardware.

Es muy importante tener en cuenta que nuestro sistema de control embebido trabaja de acuerdo con el reloj que se muestra en la barra de estado y no de acuerdo con los relojes del mundo real. La enorme potencia de la simulación es permitirnos controlar el reloj a nuestro antojo. En las siguientes páginas de esta guía encontraremos muchos ejemplos para aprovecharnos de esta facilidad que nos brinda Proteus. Y en el apartado de tutoriales encontraremos varias aplicaciones prácticas de esta técnica.

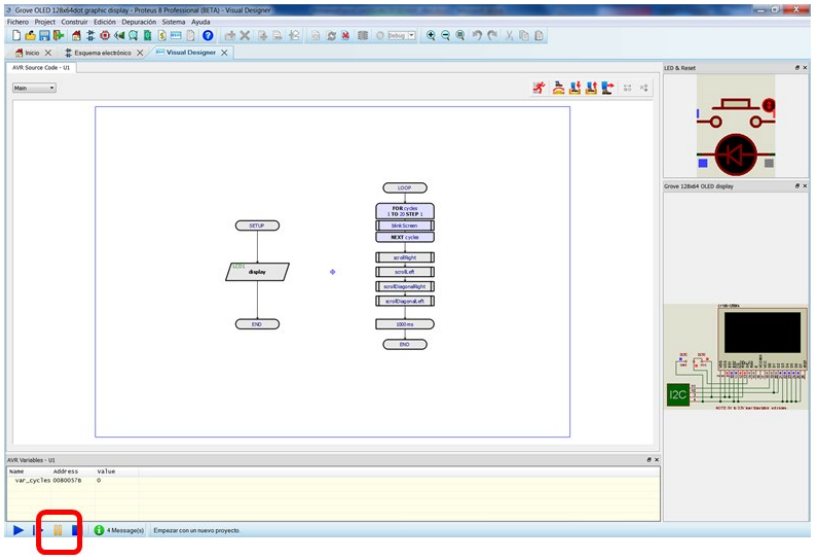
Para poner en pausa la simulación tenemos que utilizar el botón 'Pausa' de los controles de simulación de la zona inferior de la pantalla o utilizando el atajo de teclado 'PAUSA/INTERR.'





La simulación también se pondrá en pausa de forma automática si se alcanza algún punto de ruptura que hayamos situado en nuestro código. Los puntos de ruptura serán discutidos en profundidad en el apartado de esta ayuda dedicado a las técnicas de depuración. Y en el apartado de tutoriales encontraremos también varias aplicaciones prácticas.

En cuanto la simulación entra en pausa, El Diseñador Gráfico se coloca en modo depuración y nos mostrará en la pestaña tanto el flujograma como las distintas ventanas emergentes activas disponibles que estarán todas ellas listas para que podamos realizar las operaciones de inspección o, si lo deseamos, interactuar con ellas.

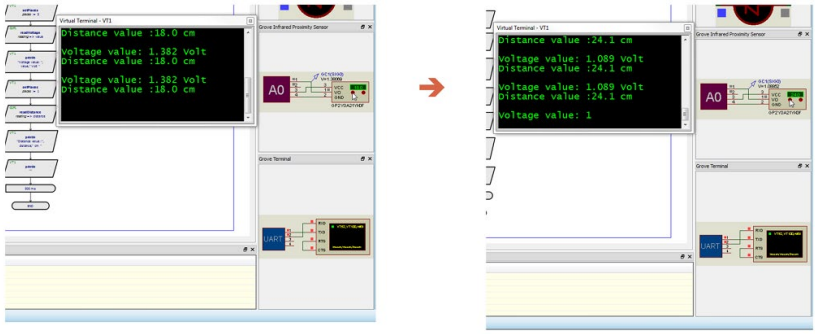


El lector debe tener en cuenta que durante la ejecución de la simulación no puede ver el flujograma salvo que la sitúe en modo pausa, porque el código se está ejecutando tan rápido que refrescar la situación actual en cada momento para mostrar el bloque que está activo es imposible.

6.1.4.-Interactuar con el sistema durante la simulación.

Cuando la simulación está en marcha, podemos ver que el Diseñador Gráfico se coloca de forma automática en el modo depuración. En él todos los paneles que muestran los diferentes elementos interactivos están situados en el lado derecho de la pestaña. Casi seguro que estos elementos interactivos sean una mezcla de indicadores (leds, displays LCD, etc.) y actuadores (pulsadores, interruptores, teclados, etc.) Los primeros nos proporcionan información en tiempo real del estado del proyecto y los segundos no permiten interactuar con el equipo para observar su reacción.

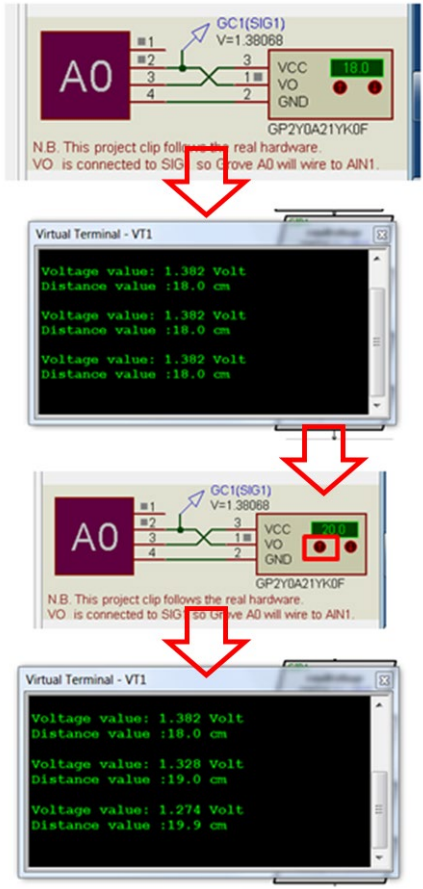
Veamos un ejemplo real en el que utilizamos un sensor de proximidad que usa la tecnología de infrarrojos.



Primero, arrancaremos la simulación.

Una vez en marcha podemos utilizar el ratón para actuar sobre los dos pulsadores incluidos en el modelo del sensor de proximidad.

Podemos observar el resultado provocado sobre el diseño al cambiar los valores de lectura del sensor.



También es posible utilizar la pestaña dónde se muestra el esquema electrónico e interactuar sobre ella si lo preferimos. Aunque, muy posiblemente, nos resultará mucho más útil para la depuración de nuestro código poder hacerlo desde la pestaña del Diseñador Gráfico donde tenemos todo junto, el flujograma y las ventanas emergentes activas con las zonas más significativas de nuestro diseño.

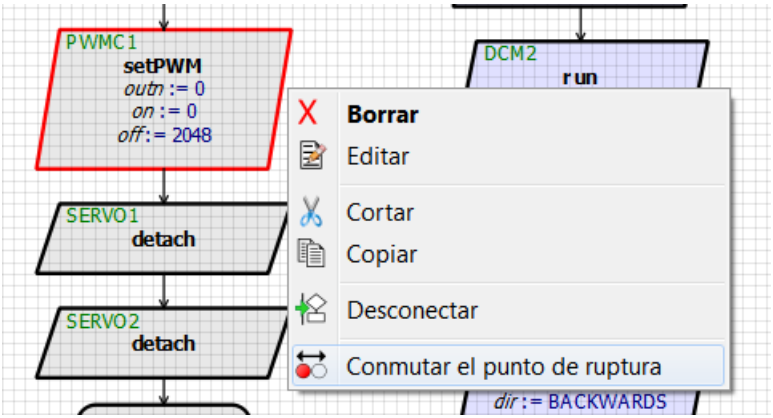
> La utilización del terminal virtual es sensiblemente diferente a la técnica que acabamos de mostrar de las ventanas emergentes activas y añade nuevas posibilidades a la capacidad de simular proyectos en Proteus.

Usándolo durante la simulación podemos enviar comando con instrucciones específicas a nuestro código lo cual tiene muchas e interesantes utilidades.

6.2.-Las técnicas de depuración.

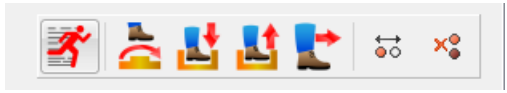
6.2.1.-Insertar un punto de ruptura en el flujograma.

Hasta ahora hemos visto que podemos poner en pausa en cualquier momento nuestra simulación y aprovechar para inspeccionar el estado de nuestro proyecto en ese instante. En ocasiones puede ser interesante que la simulación se ponga en pausa de forma automática cuando se ejecute un determinado bloque de nuestro flujograma. Para insertar un punto de ruptura (y que la ejecución se ponga en pausa en ese momento) en algún lugar de nuestro flujograma sólo tenemos que pulsar el botón derecho del ratón y seleccionar desde el menú contextual la opción 'conmutar el punto de ruptura'.



Esa operación la podemos realizar antes de arrancar la simulación o cuando la simulación está en pausa.

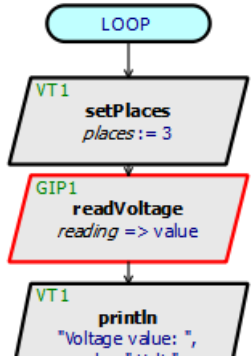
Una vez puesta la simulación en pausa es posible ejecutar el código por saltos simples de una sola línea de código utilizando los comandos que se encuentran situados en la zona superior de la ventana de código fuente, las opciones del menú 'Depuración' o los atajos de teclado "F10" y "F11"



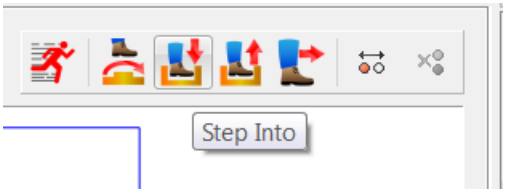
Para visualizar correctamente el flujo del programa resulta muy útil ejecutar la simulación pulsando el botón 'PAUSA' desde los controles de la simulación en lugar de usar el botón 'PLAY'. Al hacerlo así, la simulación arranca y se pone inmediatamente en pausa. A partir de ahí podemos ir ejecutando el programa paso a paso desde la opción 'Salta hasta la línea de código (over) en la función o subrutina' del menú Depuración, el comando de la barra de herramientas o el atajo de teclado F10. De esta manera el programa se ejecuta suficientemente despacio para que podamos ir viendo cómo nos vamos desplazando por los diferentes bloques que conforman nuestro flujograma, cómo se comporta nuestro sistema y encontrar en que parte del código se produce un determinado problema.

6.2.2.-Ejecutar hasta un determinado bloque del flujograma.

Una vez que la simulación se encuentra en pausa, podemos ordenar que nuestro código se ejecute hasta un determinado bloque del flujograma. Para hacerlo, debemos señalar el bloque en cuestión dentro de nuestro flujograma.



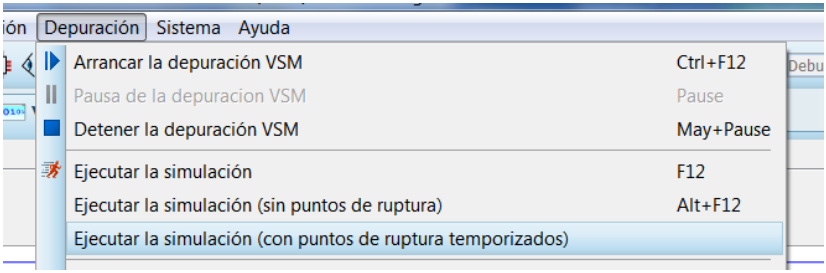
Y una vez marcado utilizar la opción de menú ‘Saltar hasta la línea de código (into) dentro de la actual función’, el comando ‘Step Into’ de la barra de herramientas o el atajo de teclado ‘F11’



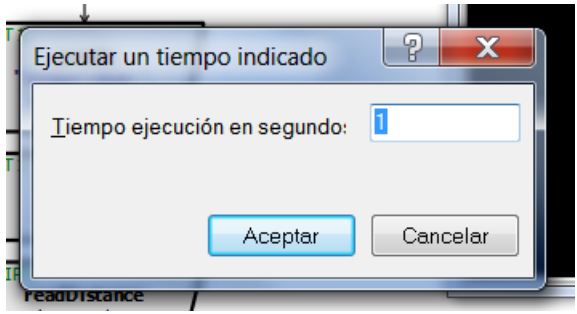
6.2.3.-Insertar un punto de ruptura temporizado.

En ocasiones, también puede resultar muy útil ordenar que se genere una pausa en la simulación después de transcurrido una determinada cantidad de tiempo en lugar de hacerlo en base al flujograma.

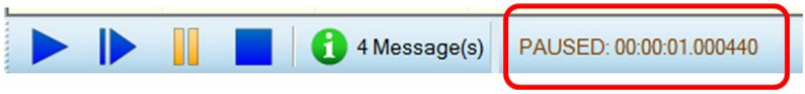
Para hacerlo usaremos la opción ‘Ejecutar la simulación (con puntos de ruptura temporizados)’ del menú Depuración



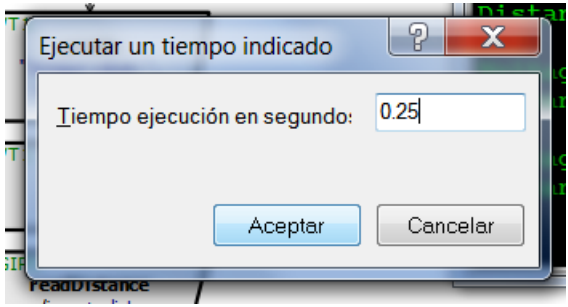
Se nos mostrará una ventana donde podemos seleccionar los segundos que deseamos que se ejecute nuestro código antes de generarse la siguiente pausa:



En la barra de estado, podemos comprobar como el reloj de la simulación ha avanzado la cantidad especificada.



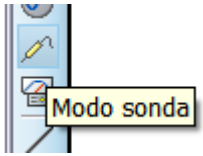
Podemos introducir valores de tiempo menores de un segundo utilizando como separador decimal el punto.



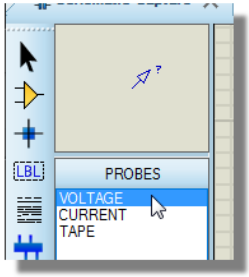
6.2.4.-Insertar un punto de ruptura por hardware.

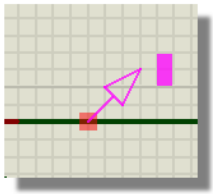
Hasta ahora, hemos visto como podíamos interrumpir la ejecución de la simulación fijando puntos de ruptura basados en condiciones que se producían durante la ejecución del software. PROTEUS VSM también nos permite configurar puntos de ruptura basados en el hardware con objeto de comprobar en un momento dado la situación del código cuando ocurre una condición determinada en alguno de los dispositivos que componen nuestro diseño electrónico.

Activemos la pestaña ‘Esquema electrónico’. A continuación usaremos el icono ‘modo sonda’ situado en la barra de herramientas lateral.



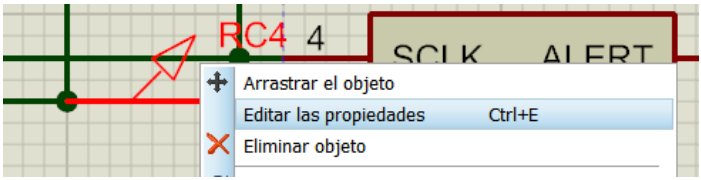
Pondremos el cursor en el área de trabajo y pulsaremos con el botón izquierdo del ratón para comenzar su posicionamiento dentro de nuestro esquema electrónico.



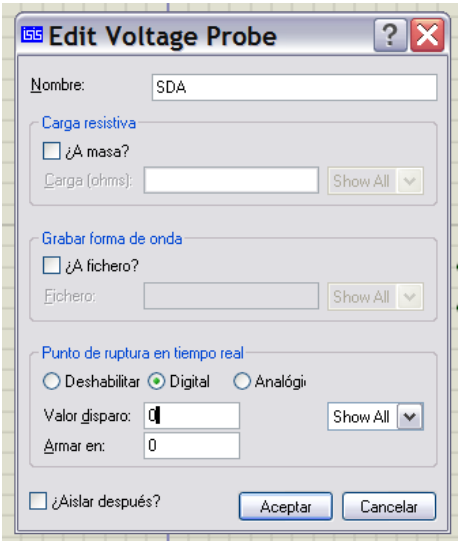


Los terminales que tienen el mismo nombre son considerados por Proteus cómo elementos conectados por un ‘cableado invisible’. Esta es una técnica muy sencilla de mantener tarjetas de expansión o módulos Grove perfectamente conectados entre sí o con la placa base sin tener que llenar la pantalla de confusos cableados que se entrecruzan y hacen la lectura del diagrama electrónico realmente complicada.

Pondremos el cursor del ratón sobre la sonda que acabamos de instalar (si es necesario utilizaremos el zoom para facilitar esta tarea) y pulsaremos el botón derecho. Seleccionaremos la opción ‘Editar las propiedades’ en el menú contextual que aparece.



En la ventana de diálogo elegiremos un punto de ruptura digital cuando el valor de tensión sea el que deseemos que produzca la pausa en la simulación.



Pulsaremos sobre el botón ‘Aceptar’ para cerrar la ventana de diálogo, utilizaremos el atajo de teclado “F8” para hacer un zoom que contenga todo nuestro diseño electrónico y pondremos en marcha la simulación utilizando el botón ‘PLAY’ desde los controles de la simulación.

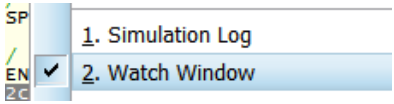
Si todo ha ido bien, la simulación se detendrá justo cuando en ese punto del circuito se alcance el valor de tensión que hayamos indicado.

6.2.5.-Utilización de la ventana de inspección.

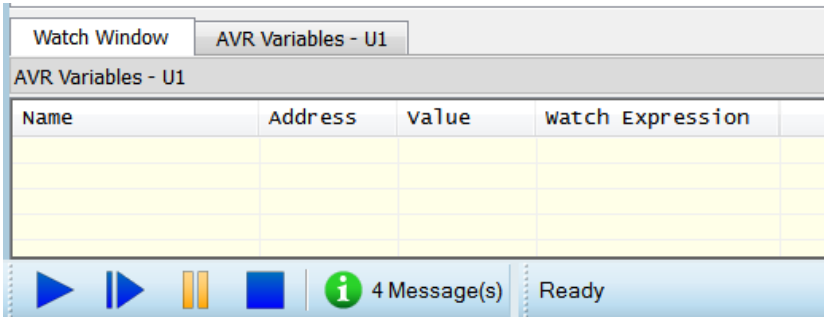
La ventana de inspección (watch window) es la única que permanece visible mientras la simulación se está ejecutando y nos facilita otra forma de trabajar diferente a la de los puntos de ruptura.

Pondremos en marcha la simulación utilizando el botón PLAY de los controles de simulación.

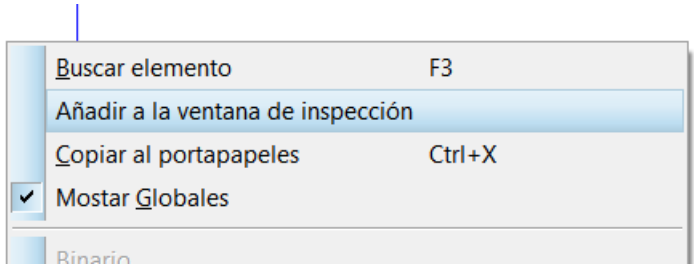
A continuación abriremos la ventana de inspección utilizando la opción '2. Watch Window' del menú 'Depuración'.



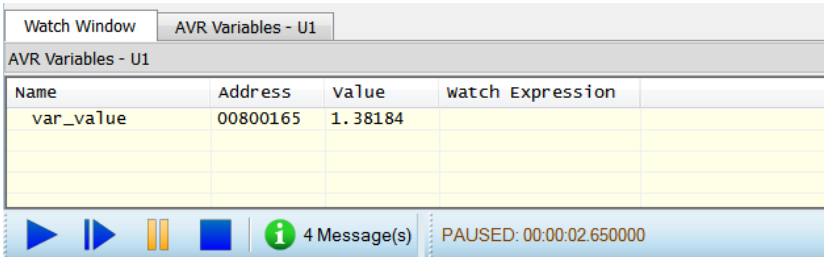
La ventana se abre una pestaña nueva en la ventana de depuración situada en la zona inferior.



Nos colocaremos en la pestaña 'AVR Variables – U1', nos situaremos sobre la variable que deseemos visualizar (por ejemplo 'var_value') y pulsaremos el botón derecho del ratón. Seleccionaremos la opción 'Añadir a la ventana de inspección' del menú contextual que nos aparece.



Volveremos a la pestaña 'Watch Window' y pondremos en marcha la simulación. Podremos ver los diferentes valores que toma la variable en cuestión.

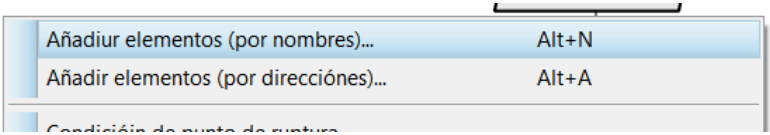


Consulte la ayuda sobre la simulación de Proteus para encontrar una información mucho más detallada de la utilización de la ventana de inspección y sus aplicaciones.

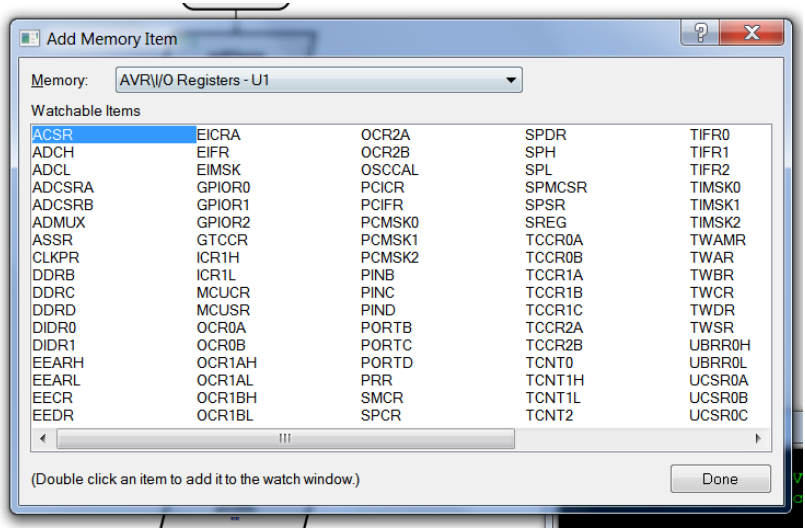
⚠ Las variables son identificadas por la ventana de inspección según su dirección. Si llevamos a cabo una modificación en nuestro código y volvemos a poner en marcha la simulación puede ocurrir que el compilador asigne una dirección de memoria diferente a esta variable y consecuentemente que dejemos de ver monitorizados sus valores en una nueva simulación. Esto puede resultar extremadamente confuso para el usuario, por lo que Labcenter recomienda siempre eliminar todas las variables monitorizadas en la ventana de inspección y volver a añadirlas después de cualquier cambio efectuado en el código.

Podemos utilizar la ventana de inspección para monitorizar cualquier registro de nuestro procesador o cualquier otra posición de memoria. Por ejemplo podemos añadir la monitorización de un registro

utilizando la opción del menú contextual ‘añadir elementos por nombre’.

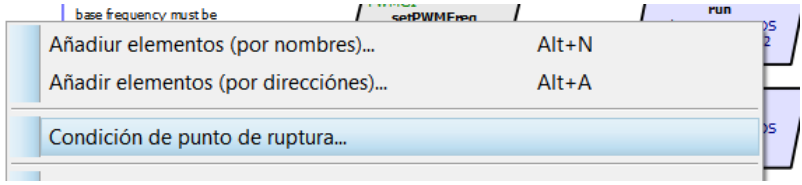


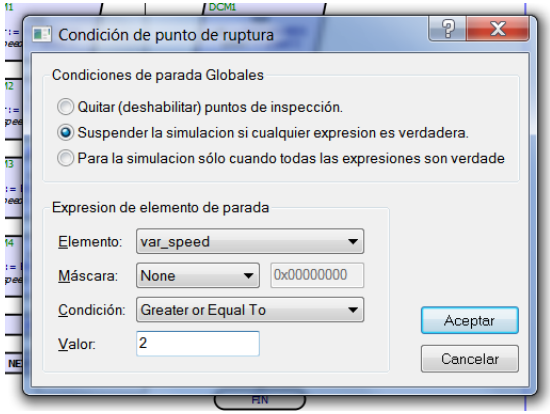
Y seleccionar desde la ventana que aparece el registro que deseamos monitorizar.



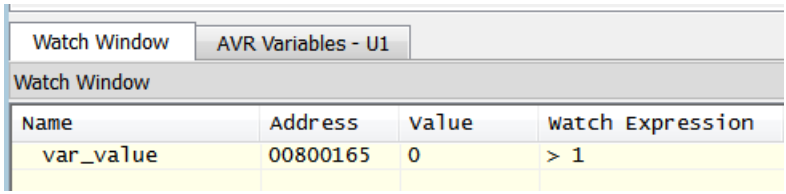
6.2.6.-Utilización de los puntos de ruptura condicional.

Podemos fijar un punto de ruptura condicional sobre los elementos que estamos inspeccionando en la ventana de inspección para detener la ejecución del programa cuando se alcance un determinado valor. Para llevarlo a cabo, pulsaremos con el botón derecho del ratón cuando el cursor esté sobre uno de los elementos visualizados en la ventana de inspección para abrir el menú contextual y seleccionaremos la opción ‘Condición de punto de ruptura…’.

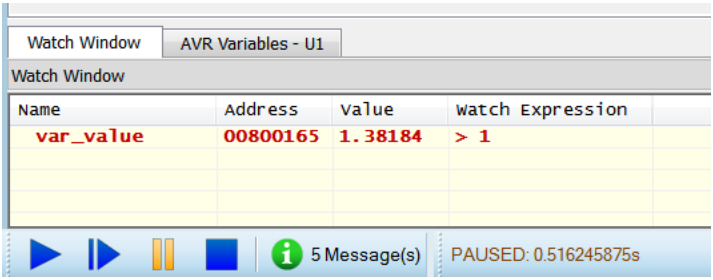




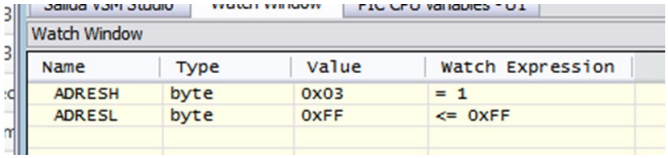
En la ventana que aparece podemos seleccionar la opción 'Suspender la simulación si cualquier expresión es verdad' y en la zona inferior seleccionar la condición que provocará la pausa en la simulación. Por ejemplo si seleccionamos que se genere el punto de ruptura cuando el valor de la variable sea mayor que uno, en la ventana de inspección veremos que queda reflejado de la siguiente manera.



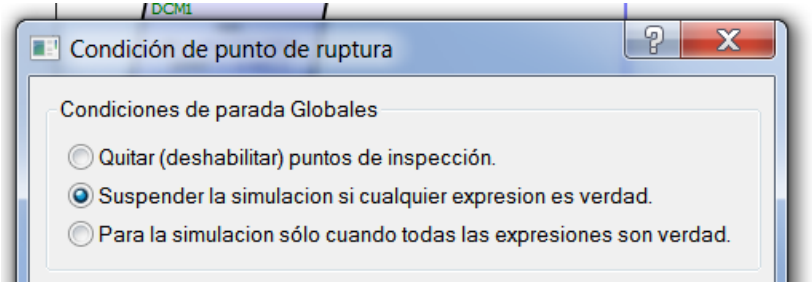
Si ejecutamos la simulación y se cumple la condición, la simulación se detendrá cuando la variable supere el valor 1 y en la ventana de inspección veremos la variable destacada de color rojo.



Podemos construir condiciones complejas basadas en más de una variable. En ese caso tenemos que utilizar la opción del menú contextual 'Parar la simulación cuando todas las condiciones son verdad'.



Para eliminar todos los puntos de ruptura condicional, usaremos la opción del menú contextual 'Quitar (deshabilitar) puntos de inspección.'

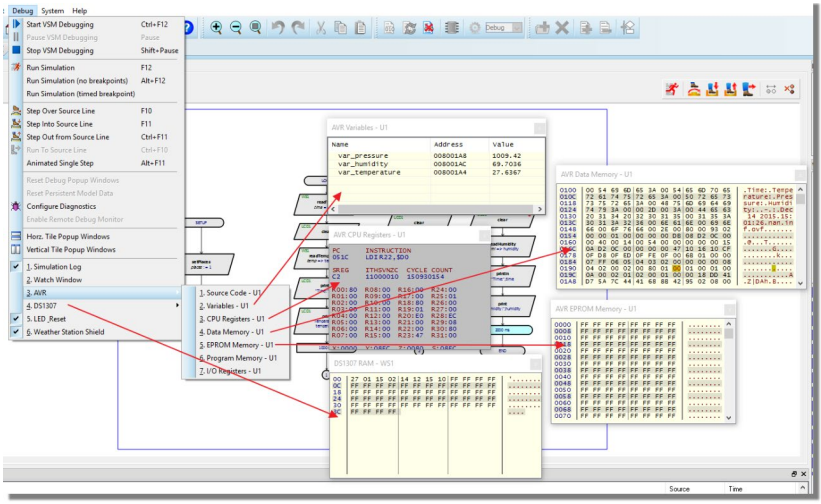


6.2.7.-Utilización de la ventana de depuración.

PROTEUS pone a nuestra disposición otras muchas ventanas de depuración diferentes. Todas ellas pueden ser activadas desde el menú 'Depuración' siempre que la simulación esté en pausa. Es posible abrir o cerrar cada una de las ventanas de depuración en cualquier momento y volver a activarlas de nuevo utilizando los comandos del menú 'Depuración'

Aunque no podemos entrar en el detalle de su utilización en esta ayuda, no queremos dejar de llamar la atención del lector sobre su existencia y posibilidades. Cuando uno se acostumbra a utilizar la potencia de las ventanas de depuración que nos ofrece Proteus, se convierten en una herramienta imprescindible en la depuración de problemas.

Incluso disponemos de ventanas de depuración específicas para muchas de las tarjetas de expansión disponibles.



> Si en un momento dado nos encontramos con una pantalla demasiado poblada de diferentes ventanas de depuración podemos cerrar algunas de ellas y reabrirlas cuando sea necesario desde el menú 'Depuración'.

Además, excepto la ventana de inspección, todas ellas desaparecerán mientras la depuración esté en marcha. Sólo se visualizan al poner la depuración en pausa.

6.2.8.-Los recursos.

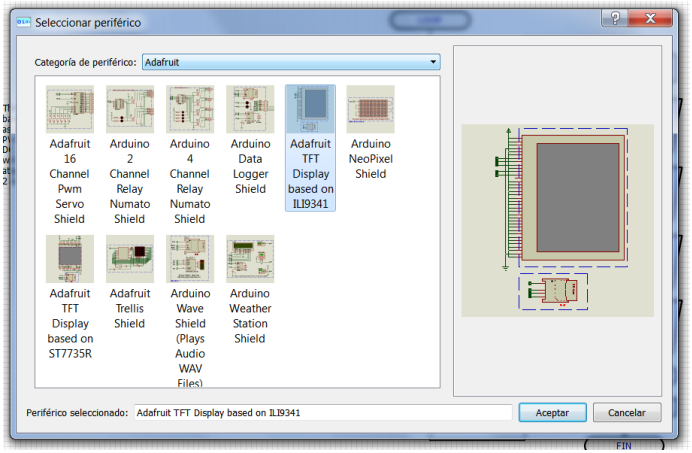
El Diseñador Gráfico incorpora un modelo de almacenamiento de datos realmente potente y, a la vez, muy sencillo. El funcionamiento se basa en la utilización de ficheros (los recursos) que se pueden incorporar a nuestro proyecto directamente desde el menú

contextual del árbol de estructura del proyecto. Un recurso puede ser cualquier fichero de texto, una imagen o un archivo de audio.

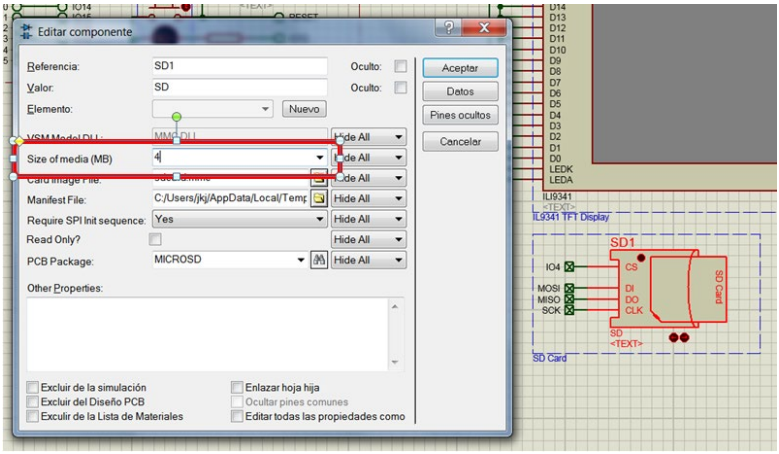
Es muy habitual que alguna de las tarjetas de expansión que incorporemos a nuestro proyecto tenga una tarjeta SD además de la circuitería electrónica asociada para gestionarla. Podemos citar los ejemplos de la tarjeta de expansión de la pantalla TFT que es capaz de mostrar las imágenes almacenadas en la tarjeta SD y de la tarjeta de expansión reproductora de sonidos que es capaz de reproducir un fichero WAV almacenado en la tarjeta SD.

Con una forma de funcionar muy similar a la que nos tiene ya acostumbrados el Diseñador Gráfico, el procedimiento para trabajar con este tipo de tarjetas de expansión es el siguiente:

Añadir a nuestro proyecto la tarjeta de expansión que gestionará el recurso. Por ejemplo, cualquiera de la dos mencionadas con anterioridad para reproducir sonidos o manejar una pantalla TFT.

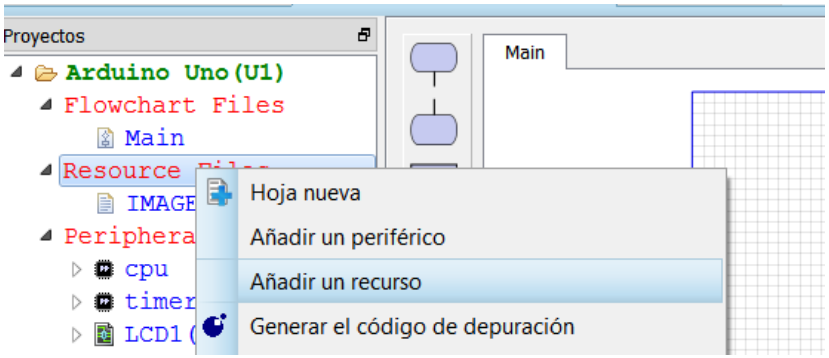


Comprobar que el tamaño del fichero que simula el espacio de almacenamiento de la tarjeta SD es suficientemente grande para contener la imagen que deseamos visualizar. Normalmente, los tamaños de las tarjetas SD que necesitamos para almacenar ficheros WAV son mayores que los necesarios para guardar imágenes.



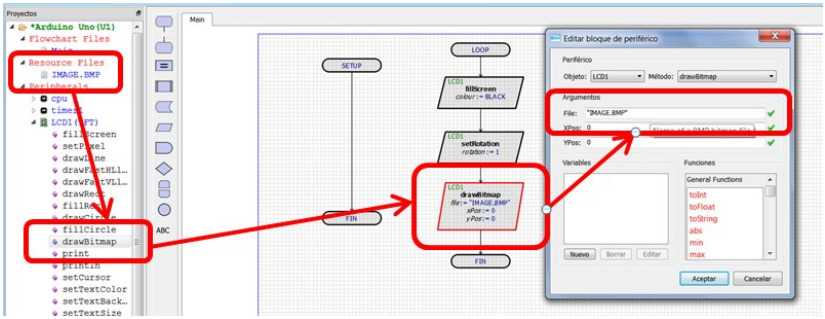
> Proteus aprovecha la potencia del software de simulación y hace un poco de magia, creando para nosotros una imagen de la estructura FAT de tamaño adecuado. Si estuviéramos programando hardware real necesitaríamos estar seguros si la tarjeta SD que estamos usando introducida en nuestra tarjeta de expansión tiene una estructura FAT16 o FAT32 de tamaño adecuado antes de que pudiéramos utilizar el botón de programación.

Añadir el recurso con los datos desde el árbol de estructura del proyecto utilizando el menú contextual.



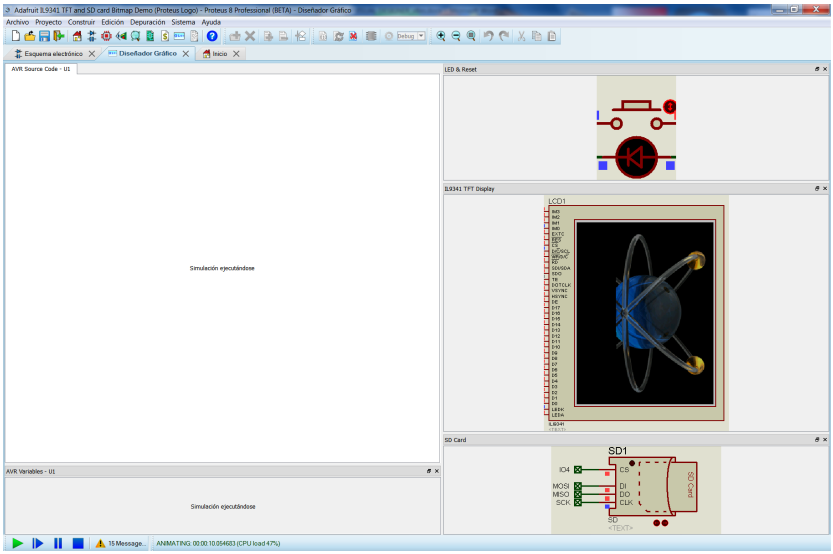
⚠ Debemos ser cuidadosos con el uso de los nombres de los recursos de fichero por una peculiaridad propia de la gestión de las tarjetas de memoria SD que hace Arduino. Sólo soporta nombres de archivo según el estándar 8.3 que es el utilizado por las tarjetas formateadas según el estándar FAT16. Esto significa, que el nombre del archivo debe tener ocho caracteres o menos y la extensión tres o menos. Por ejemplo, si intentamos utilizar un fichero de mapa de bits llamado 'LogoProteus.bmp' nos generará un error. Si, por el contrario, utilizamos el nombre 'Logo.bmp' o 'imagen1.bmp' funcionará correctamente.

Coger y arrastrar el recurso dentro de nuestro código. El Diseñador Gráfico automáticamente detectará la rutina de gestión adecuada para el recurso. Por ejemplo, el método DrawBitmap() o Play(). Y establecerá como la fuente para este método del recurso que acabamos de seleccionar.



⚠ Debemos tener cuidado al añadir un recurso que en nuestro Proyecto estemos utilizando una tarjeta de expansión o una tarjeta de periférico que contenga una tarjeta SD. Si no, no tendremos ningún hardware donde almacenar el recurso.

La siguiente imagen recoge un momento de la simulación de mostrar el logotipo de Proteus almacenado en la tarjeta SD como un recurso, sobre la pantalla TFT de nuestra tarjeta de simulación.


Adafruit ILI9341 TFT and SD card Bitmap Demo (Proteus Logo) - Proteus 8 Professional (BETA) - Diseñador Gráfico
Archivo Proyecto Construir Edición Depuración Sistema Ayuda
Esquema electrónico
Diseñador Gráfico
Inicio
AVR Source Code - U1
Simulación ejecutándose
AVR Variables - U1
Simulación ejecutándose
LED & Reset
ILI9341 TFT Display
LCD1
SD Card
SD1
IOH
MOSI
MISO
SCK
CS
DI
DO
CLK
SD Card
SD
<TEXT>
15 Message...
ANIMATING: 00:00:10.054603 (CPU load 47%)

7.-Ejecución del proyecto en un equipo Arduino real.

En esta sección vamos a mostrar los pasos a seguir para cargar nuestro programa desarrollado usando el Diseñador Gráfico dentro de un equipo Arduino real. Con este último paso, Proteus se convierte en un entorno de desarrollo para equipos Arduino que nos permite llevar a cabo todas las fases de nuestro proyecto.

Para encontrar más información relativa a la programación de los equipos Arduino se pueden consultar los siguientes enlaces a las preguntas frecuentes y al foro de desarrolladores Arduino.

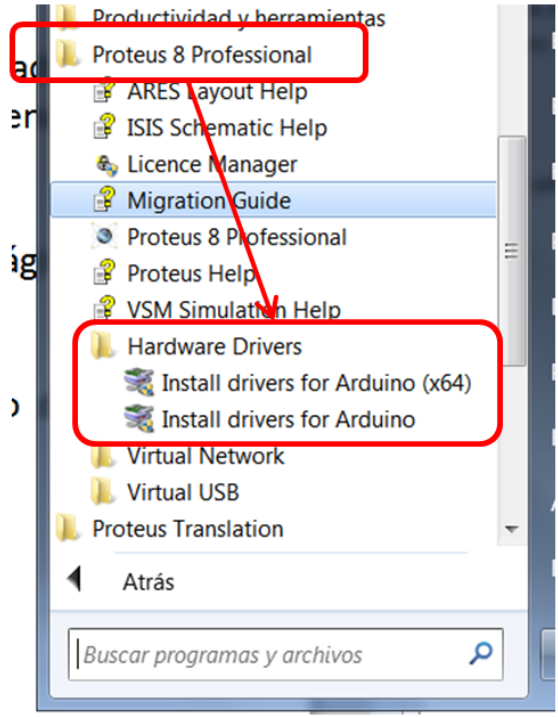
https://www.arduino.cc/en/Main/FAQ

http://forum.arduino.cc/

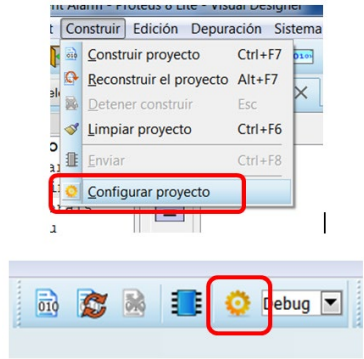
7.1.-Carga del código en el equipo Arduino.

Para facilitar nuestro trabajo, Proteus incluye un interface que se comunica con el programador del Arduino AVR. Para asegurarnos de que seremos capaces de cargar el programa en nuestro equipo Arduino desde el propio Proteus necesitamos realizar las siguientes comprobaciones.

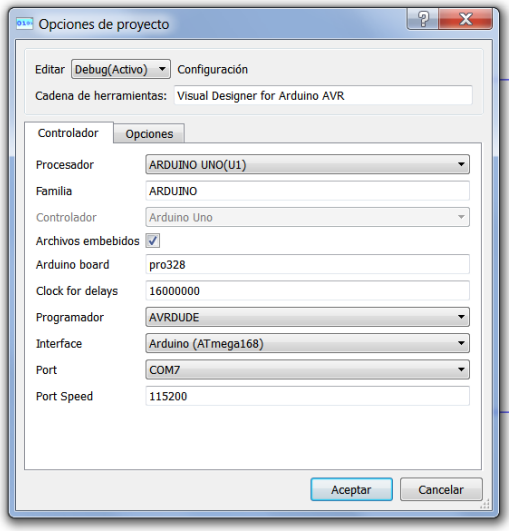
1.- Asegurarnos que tenemos los drivers para Arduino instalados en nuestro ordenador. Si no los tenemos, se pueden instalar siguiendo utilizando la opción del menú de inicio de Windows situada en el grupo de aplicaciones de Proteus.



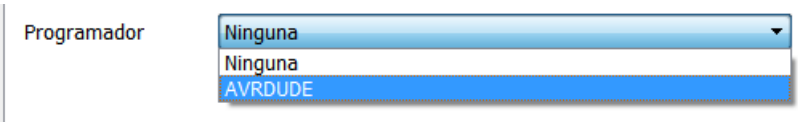
2.-Configurar correctamente el Proyecto desde la opción ‘configurar proyecto’ del menú ‘Construir’ o desde la barra de herramientas.



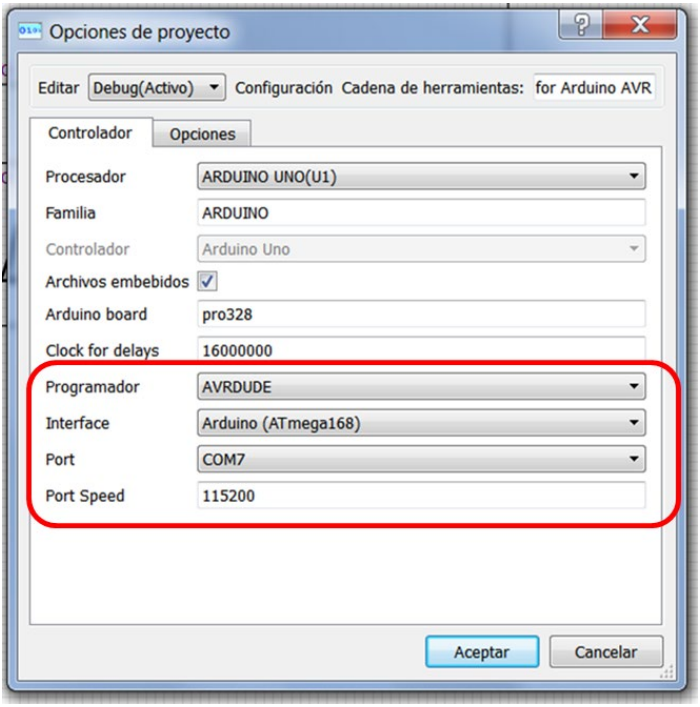
Se abre la ventana de diálogo ‘opciones de proyecto’.



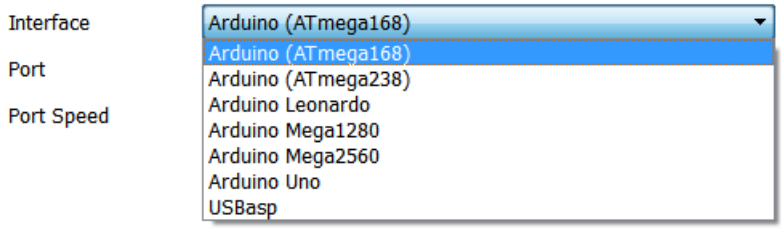
Debemos seleccionar como programador, la opción AVRDUDE.



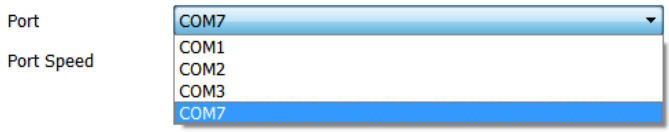
Al hacerlo, nos aparecen nuevas opciones para seleccionar: escoger el equipo Arduino que deseamos utilizar, el puerto USB con el que nos conectaremos y la velocidad de transmisión.



Debemos seleccionar como interface el que se corresponda con el procesador que estamos utilizando en nuestro proyecto.

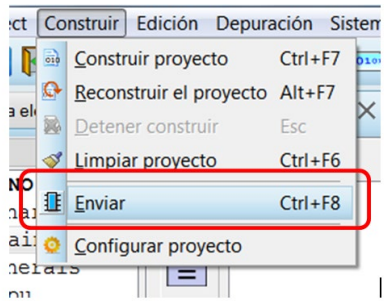


Elegir el puerto serie de nuestro ordenador personal al que tenemos enchufado el equipo Arduino. Los puertos serie que aparecen en el desplegable varían de un equipo a otro.

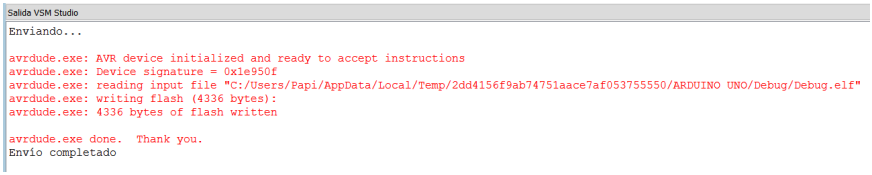


Y, por último, seleccionar la velocidad del puerto serie. En la imagen aparece que hemos seleccionado en nuestro caso la velocidad de 115200 baudios. No sabemos bien la razón por la que me he encontrado placas Arduino Uno que se conectan a diferentes velocidades por el puerto de programación, aunque sospechamos que posiblemente la diferencia está relacionada con el fabricante. Cómo es sabido, las placas Arduino tienen un esquema abierto y hay muchos fabricantes diferentes que ofrecen placas Arduino totalmente compatibles. Si la velocidad seleccionada no coincide con la de nuestra placa Arduino, el programa no se cargará y recibiremos un mensaje de error después de un par de minutos. En este caso, conviene probar con varias velocidades diferentes hasta que logremos dar con la que está usando nuestra placa Arduino. Las velocidades estándar para probar son: 9600, 14400, 19200, 38400, 57600, 115200, 128000.

3.-Cargar el programa en nuestro equipo Arduino, utilizando la opción 'Enviar' del menú Construir, el atajo de teclado Ctrl-F8 o el botón enviar de la barra de herramientas superior.



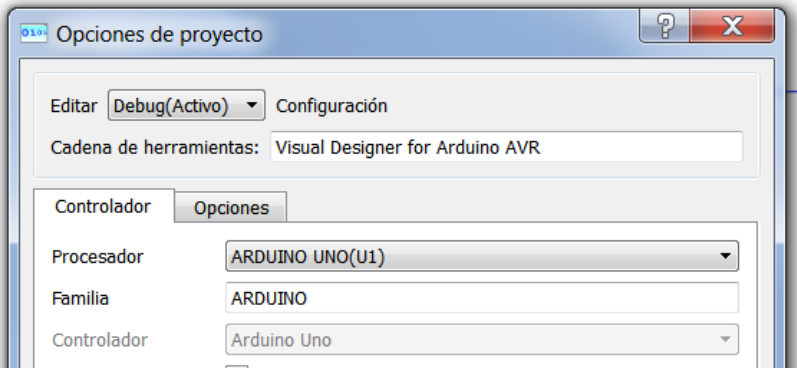
En la ventana de estado, podemos comprobar que todo se ha producido conforme a lo esperado y que el programa se ha cargado correctamente en nuestro equipo Arduino.



7.2.-Configuración del proyecto.

En 'opciones de proyecto' además de los campos con la configuración más básica que hemos visto en el punto anterior, hay otros muchos aspectos que podemos seleccionar. Los más importantes son los siguientes.

Los campos procesador, familia y controlador se fijan cuando creamos un proyecto de software para Arduino en Proteus utilizando el asistente para la creación de un nuevo proyecto. Por eso nos aparecen rellenos por defecto en esta ventana de diálogo.

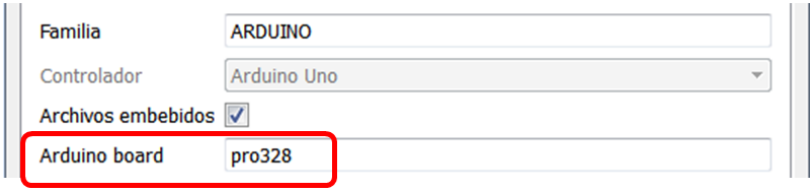


La opción 'Archivos embebidos' mantiene todos los ficheros fuente de nuestro proyecto y la información de depuración dentro del fichero de proyecto (extensión .pdsprj). De esta manera se evitan los posibles conflictos que pueden surgir por utilizar rutas a ficheros o librerías utilizados en el proyecto y posibilita que sea muy fácil migrar los proyectos realizados con el diseñador gráfico de unos equipos a

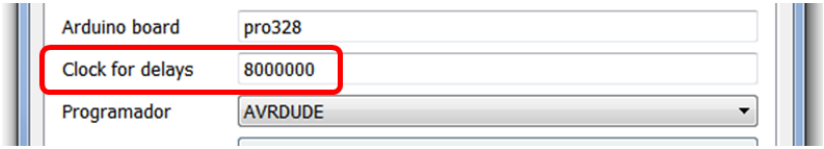
otros. Se recomienda dejar esta opción seleccionada si no hay alguna buena razón para cambiarla.



La opción 'Arduino board' contiene una cadena que se pasa al compilador y enlazador Arduino para que sea capaz de identificar que procesador tiene instalado la placa controlador de Arduino. Al menos que esté muy seguro de lo que está haciendo, no debería modificarse tampoco esta opción.

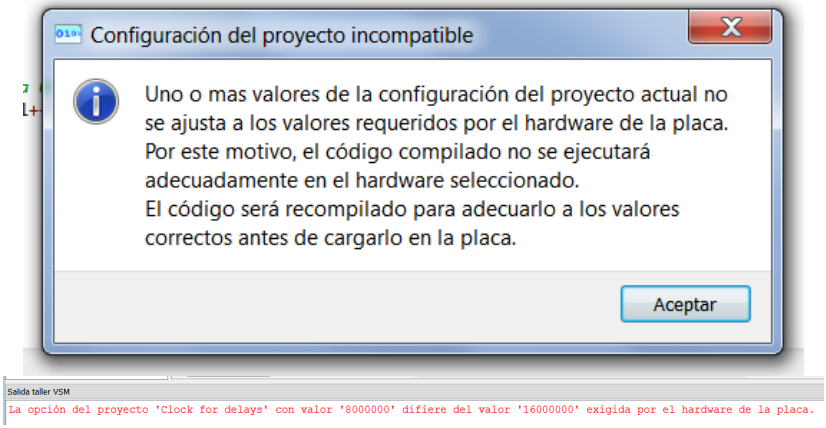


La opción 'clock for delays' básicamente nos sirve para fijar la frecuencia del reloj que se está utilizando. Aquí hay que hacer una salvedad importante. A efectos de realizar la simulación de nuestro código desde Proteus es una buena opción seleccionar 8Mhz (8000000) en este parámetro porque con ello nos aseguramos que incluso un ordenador personal de prestaciones modestas pueda llevar a cabo la tarea.



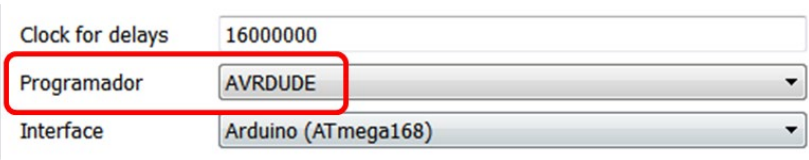
En el momento en que queramos cargar nuestro código en un equipo Arduino de la vida real, debemos seleccionar el valor de 16MHz (16000000). Pero, y esto es importante, tenemos dos maneras de hacerlo y el lector escogerá la que prefiera según el ordenador personal que esté utilizando para ejecutar Proteus.

Podemos dejar este campo con el valor de 8MHz. En este caso, cada vez que utilicemos la opción 'enviar' recibiremos un mensaje de advertencia de que el proyecto se recompilará para que nuestro equipo Arduino de la vida real se programe con la velocidad de reloj correcta de 16Mhz. De esta manera disfrutamos lo mejor de los dos mundos, simulamos con una frecuencia de reloj de 8MHz y programamos nuestro Arduino de la vida real con una frecuencia de reloj de 16Mhz.

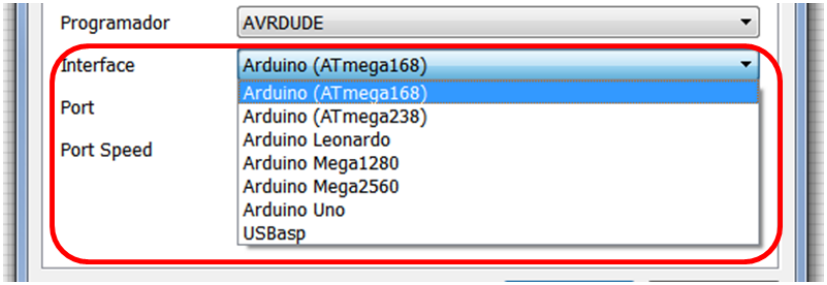


También podemos cambiar este valor en la ventana de configuración que estamos viendo. De esta manera no será necesario recompilar el proyecto cada vez que lo enviemos a nuestro equipo Arduino de la vida real, pero es posible que si nuestro ordenador personal no es lo suficientemente potente, la simulación se vea afectada y se ejecute 'a cámara lenta'

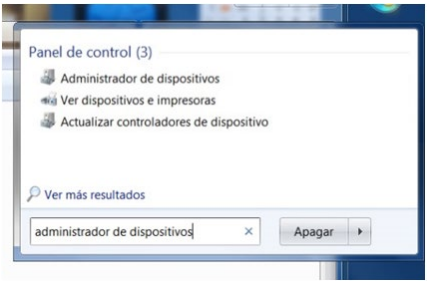
En el campo programador debemos seleccionar siempre la opción AVRDUDE. Si no estuviera disponible, debemos instalar el IDE de Arduino disponible desde la página web oficial.



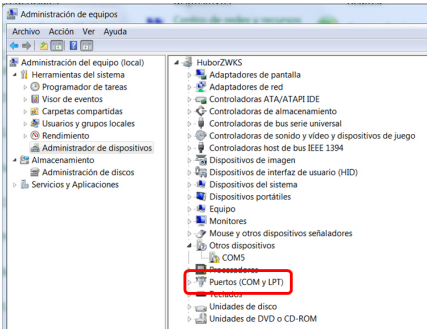
El campo Interface, deberá ser rellenado de acuerdo con el hardware que estemos utilizando.



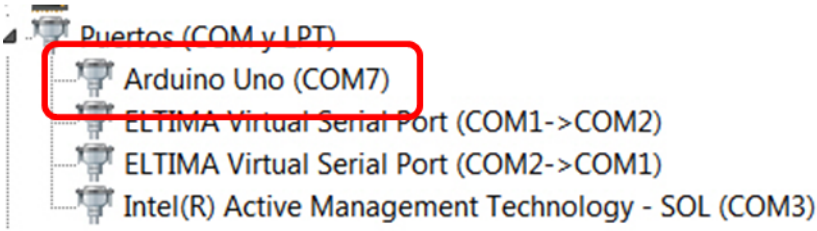
El campo ‘port’ es el que es menos intuitivo de rellenar. Los drivers instalados han tenido que asignar un puerto serie (COM<n>) al que se ha conectado nuestro equipo Arduino. Nuestra primera labor consiste en averiguar qué puerto se le ha asignado. Para ello consultaremos el administrador de equipos. Como dependiendo de la configuración de nuestro ordenador, el administrador de equipos puede estar visible en un menú u en otro, lo mejor es dejar que Windows lo busque por nosotros. Para eso abrimos el menú inicio y en el cuadro de búsqueda de la zona inferior escribimos 'Administrador de dispositivos'. Windows se encargará de buscarlo y nos lo ofrecerá en los resultados.



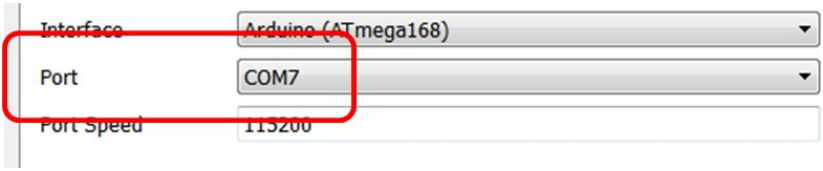
Pulsando sobre el resultado obtenido, se abrirá el administrador de equipos de Windows. En la parte izquierda seleccionaremos, en el árbol de opciones, 'Administrador de dispositivos' y en la zona derecha la opción Puertos (COM y LPT):



En la rama de esta opción tiene que aparecer nuestro equipo y el puerto que se le ha asignado. En nuestro caso, como hemos utilizado una placa Arduino Uno, podemos observar que el puerto serie ha sido el COM7.



Una vez que ya sabemos el puerto, lo podemos seleccionar utilizando el campo ‘port’.



Para el campo ‘port speed’ que contiene la velocidad del puerto serie que utilizaremos para cargar el código en nuestro equipo Arduino de la vida real, nos remitimos al apartado anterior dónde ya lo hemos visto. No sabemos bien la razón por la que me he encontrado placas Arduino Uno que se conectan a diferentes velocidades por el puerto de programación, aunque sospechamos que posiblemente la diferencia está relacionada con el fabricante. Cómo es sabido, las placas Arduino tienen un esquema abierto y hay muchos fabricantes

diferentes que ofrecen placas Arduino totalmente compatibles. Si la velocidad seleccionada no coincide con la de nuestra placa Arduino, el programa no se cargará y recibiremos un mensaje de error después de un par de minutos. En este caso, conviene probar con varias velocidades diferentes hasta que logremos dar con la que está usando nuestra placa Arduino. Las velocidades estándar para probar son: 9600, 14400, 19200, 38400, 57600, 115200, 128000.

7.3.-Solución de los problemas reportados con más frecuencia.

Si al enviar el código a nuestro equipo Arduino nos encontramos problemas, conviene repasar los problemas que los usuarios han reportado a Labcenter con más frecuencia, para ver si nuestro caso está entre ellos.

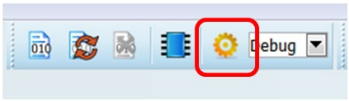
7.3.1.-El botón enviar está deshabilitado.



Debemos asegurarnos que nuestro cable USB está correctamente enchufado tanto en nuestra tarjeta Arduino como en nuestro ordenador personal.

Comprobar que el led de nuestra tarjeta Arduino que indica que está correctamente alimentada luce.

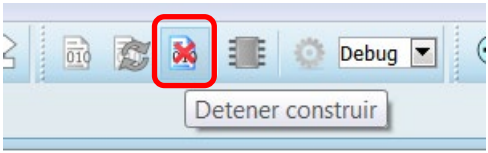
Proteus no detecta automáticamente si hemos enchufado una tarjeta Arduino al puerto usb de nuestro ordenador personal. Si lo hemos hecho y el icono 'enviar' sigue inactivo, utilizar la opción 'configurar proyecto' del menú Construir, validar los datos y comprobar si ya está el icono 'enviar' activo.



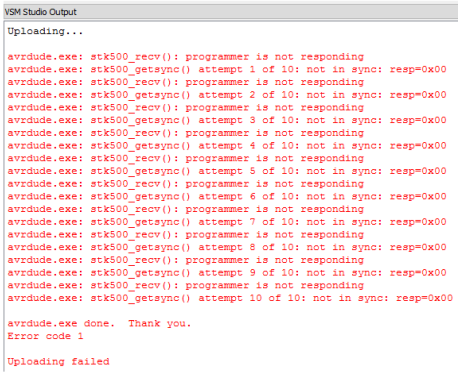
Comprobar desde la opción ‘configurar proyecto’ del menú Construir que tenemos seleccionado AVRDUDE como programador. Si no podemos seleccionarlo necesitamos instalar el IDE de Arduino desde la web oficial de Arduino.

7.3.2.-Al enviar el código tarda mucho tiempo y se generan un gran número de errores.

En primer lugar, si es preciso, se puede ordenar detener el proceso de enviar el código a nuestro utilizando el botón ‘Detener construir’ de la barra de herramientas.



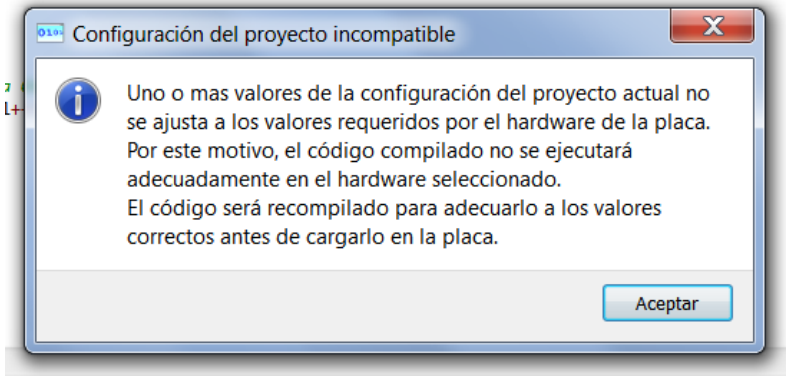
Las causas más frecuentes que provocan un tiempo de envío muy elevado son dos. La primera, que hayamos seleccionado un puerto que no es dónde realmente está conectado nuestro equipo Arduino.

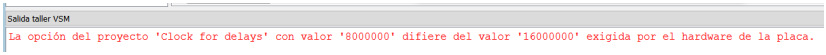


La otra es que hayamos seleccionado una velocidad del puerto equivocada. En este caso, consultar el apartado 7.2 la zona dedicada a la configuración de la velocidad del puerto serie.

7.3.4.-El error ‘clock for delays’.

Cuando enviamos el código a nuestro equipo Arduino real podemos encontrar el siguiente mensaje de error.

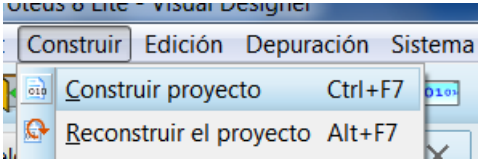




Este mensaje nos aparece porque en la ventana de configuración de proyecto tenemos seleccionada una velocidad de reloj diferente a la que está configurada en la placa Arduino. Ver el apartado 7.2 donde se trata de este asunto.

7.3.5.-No existe el fichero elf.

Recibimos un mensaje de este tipo cuando intentamos enviar un código a nuestro Arduino real cuando no hemos compilado nuestro código. Simplemente, tenemos que compilarlo utilizando la opción construir proyecto del menú construir.



8.-Ejecución del Proyecto en un equipo Raspberry Pi real.

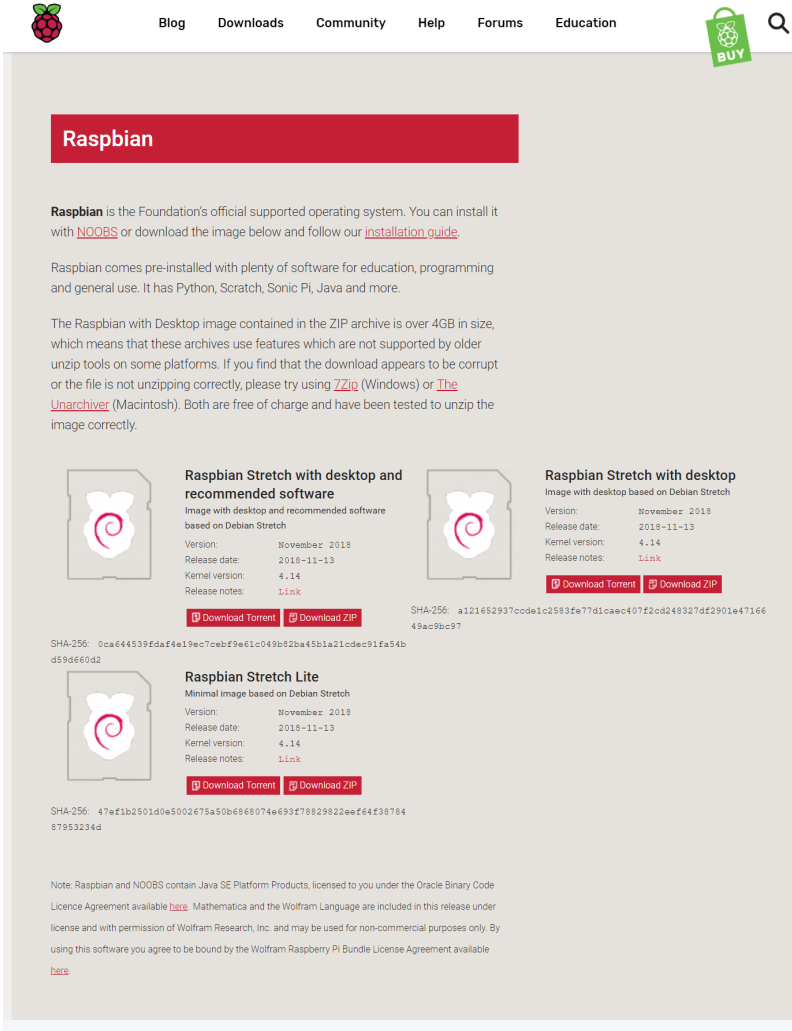
A diferencia de los equipos Arduino, los equipos Raspberry Pi deben ser configurados previamente para poder programarlos utilizando Proteus. Normalmente esta operación sólo tendremos que llevarla a cabo una sola vez, pero puede ser bastante complicada. El proceso se compone de tres pasos:

1. Preparar la tarjeta SD.
2. Localizar el equipo Raspberry Pi.
3. Configurar el equipo Raspberry Pi para utilizarlo desde el entrono Proteus.

8.1.-Preparación de la tarjeta SD.

Antes de empezar a trabajar con nuestro equipo Raspberry Pi es necesario instalar una copia de Raspbian en nuestra tarjeta SD. Si no sabe cómo llevar a cabo esta tarea puede utilizar el siguiente enlace que le guiará en la forma de hacerlo.

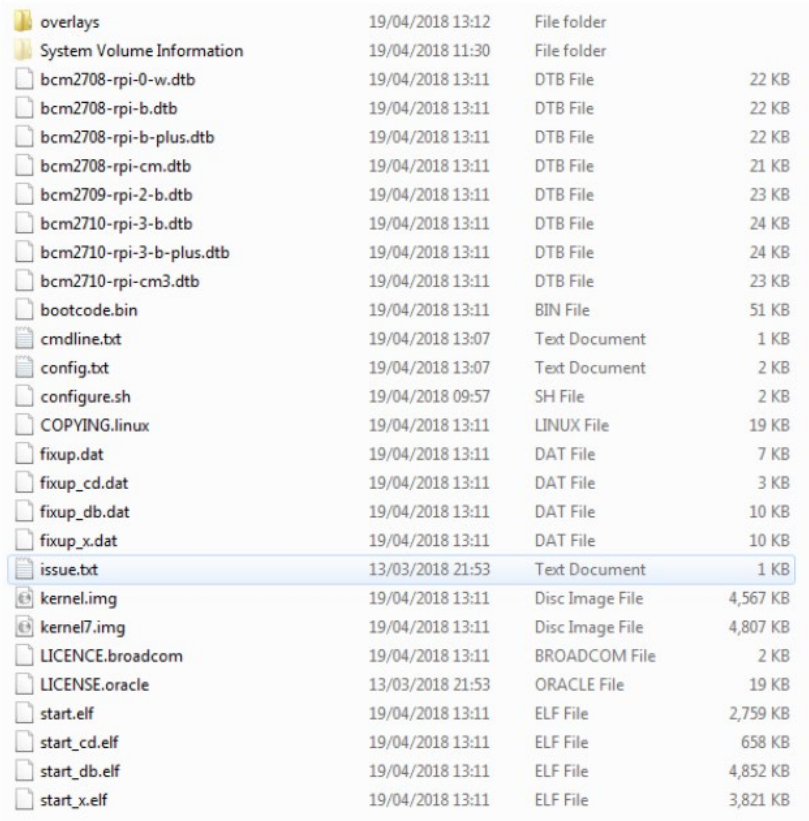
https://www.raspberrypi.org/downloads/raspbian/



Proteus puede trabajar tanto con Raspbian como con Raspbian Lite según prefiramos. La versión Lite arranca mucho más rápido.

8.2.-Copia de los ficheros a la tarjeta SD.

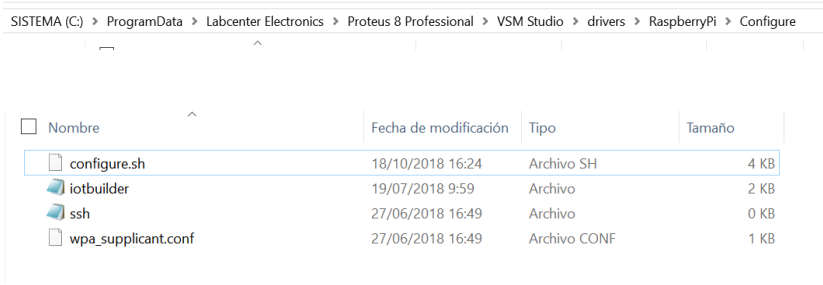
Una vez que tenemos instalado Raspbian en nuestra tarjeta SD tenemos que acceder a ella desde nuestro ordenador personal. No es necesario que enchufemos la tarjeta SD a nuestro equipo Raspberry Pi todavía. Lo que tenemos que hacer es sacarla y volver a introducirla en nuestro ordenador personal. Una vez que accedemos a la tarjeta SD desde nuestro ordenador personal tendremos que ver una lista de ficheros similar a la que se muestra en la imagen siguiente.



overlays	19/04/2018 13:12	File folder	
System Volume Information	19/04/2018 11:30	File folder	
bcm2708-rpi-0-w.dtb	19/04/2018 13:11	DTB File	22 KB
bcm2708-rpi-b.dtb	19/04/2018 13:11	DTB File	22 KB
bcm2708-rpi-b-plus.dtb	19/04/2018 13:11	DTB File	22 KB
bcm2708-rpi-cm.dtb	19/04/2018 13:11	DTB File	21 KB
bcm2709-rpi-2-b.dtb	19/04/2018 13:11	DTB File	23 KB
bcm2710-rpi-3-b.dtb	19/04/2018 13:11	DTB File	24 KB
bcm2710-rpi-3-b-plus.dtb	19/04/2018 13:11	DTB File	24 KB
bcm2710-rpi-cm3.dtb	19/04/2018 13:11	DTB File	23 KB
bootcode.bin	19/04/2018 13:11	BIN File	51 KB
cmdline.txt	19/04/2018 13:07	Text Document	1 KB
config.txt	19/04/2018 13:07	Text Document	2 KB
configure.sh	19/04/2018 09:57	SH File	2 KB
COPYING.linux	19/04/2018 13:11	LINUX File	19 KB
fixup.dat	19/04/2018 13:11	DAT File	7 KB
fixup_cd.dat	19/04/2018 13:11	DAT File	3 KB
fixup_db.dat	19/04/2018 13:11	DAT File	10 KB
fixup_x.dat	19/04/2018 13:11	DAT File	10 KB
issue.txt	13/03/2018 21:53	Text Document	1 KB
kernel.img	19/04/2018 13:11	Disc Image File	4,567 KB
kernel7.img	19/04/2018 13:11	Disc Image File	4,807 KB
LICENCE.broadcom	19/04/2018 13:11	BROADCOM File	2 KB
LICENSE.oracle	13/03/2018 21:53	ORACLE File	19 KB
start.elf	19/04/2018 13:11	ELF File	2,759 KB
start_cd.elf	19/04/2018 13:11	ELF File	658 KB
start_db.elf	19/04/2018 13:11	ELF File	4,852 KB
start_x.elf	19/04/2018 13:11	ELF File	3,821 KB

Ahora debemos copiar todos los ficheros del directorio drivers de Proteus. Si hemos instalado Proteus en la forma estándar se deben encontrar en el directorio:

C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\VSM S Studio\drivers\RaspberryPi\Configure



Y debemos colocarlos dentro de la tarjeta SD. Si no queremos usar una red Ethernet cableada, tenemos que editar el fichero wpa_supplicant.conf para permitir la conexión a la red wifi durante el arranque del equipo. Solo tenemos que indicar el nombre y la contraseña de la red y la Raspberry Pi se conectará en el arranque.

wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=«your_ISO-3166-1_two-letter_country_code»

network={
    ssid="«your_SSID»"
    psk="«your_PSK»"
    key_mgmt=WPA-PSK
}
```

Y tendremos que editar el fichero configure.sh para indicar un nombre de host adecuado para nuestro equipo Raspberry Pi que no entre en conflicto con nuestra red.

wpa_supplicant.conf | configure.sh

```
#!/bin/sh
# Raspbian Configuration Script

hostname=raspberrypi
```

Ya podemos expulsar de manera segura nuestra tarjeta SD de nuestro ordenador personal.

8.3.-Localizar el equipo Raspberry Pi.

El siguiente paso es localizar la dirección IP que está utilizando nuestro equipo Raspberry Pi. Lo podemos llevar a cabo por diferentes caminos. En esta ayuda, vamos a seguir uno de ellos.

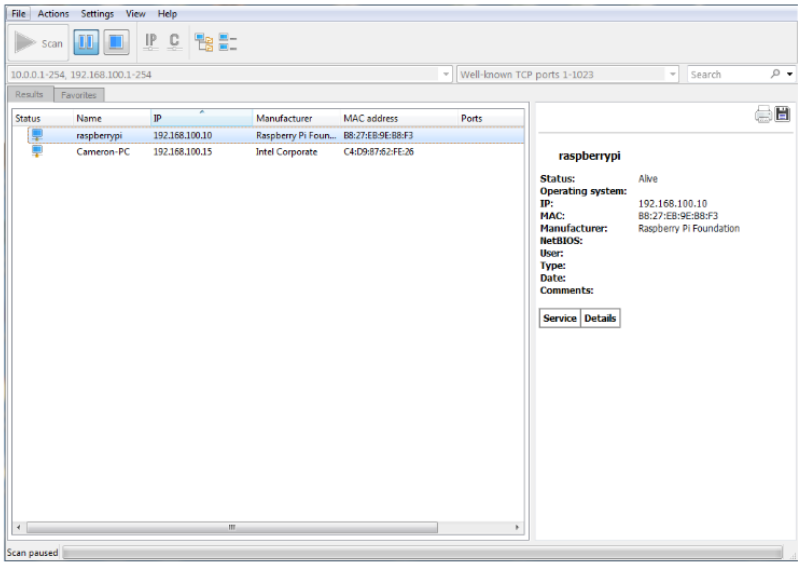
Primero, instalaremos un escáner de puertos gratuito. Con ello ahorraremos tiempo y no será necesario conectar a nuestro equipo Raspberry Pi una pantalla y un teclado para llevar a cabo esta operación. Utilizaremos la escáner de puertos gratuito que podemos encontrar en la siguiente página web.

https://www.advanced-port-scanner.com/



Seguiremos las instrucciones indicadas para instalarlo en nuestro ordenador personal. El botón para descargarlo se encuentra en la página de inicio y se trata de una herramienta gratuita.

Al terminar la instalación encenderemos nuestro equipo Raspberry Pi. Si utilizamos Raspian Lite el arranque será más rápido que con la versión normal de Raspian puesto que no tiene que cargar el interface gráfico de usuario (GUI). Cuando nuestro Raspberry Pi haya acabo su proceso de arranque podemos utilizar la herramienta Advanced Port Scanner para localizar nuestro equipo Raspberry Pi.



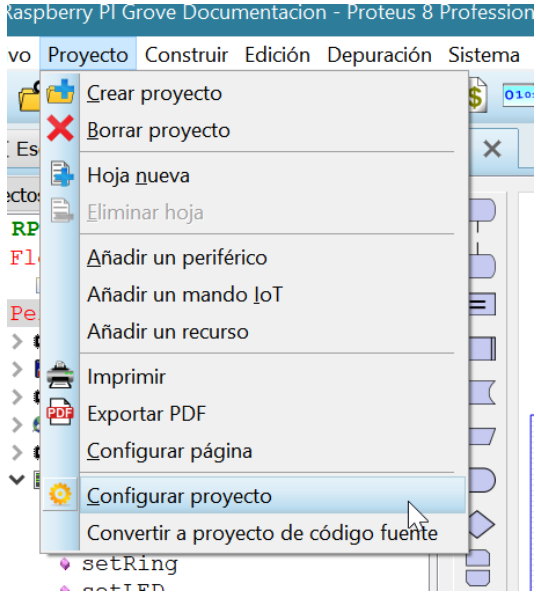
Evidentemente en nuestra red pueden estar conectados cientos de equipos. En la imagen de ejemplo sólo hay dos equipos conectados: nuestro ordenador personal y nuestro equipo Rapsberry Pi. En cada caso tenemos que localizar el equipo Raspberry Pi y lo haremos porque será el equipo que tenga como nombre 'raspberrypi' y como frabricante 'Raspberry Pi Foundation'. Una vez localizado, anotaremos su dirección IP. En la imagen de ejemplo la dirección IP es 192.168.100.10.

Seguiremos adelante sin apagar nuestro equipo Raspberry Pi.

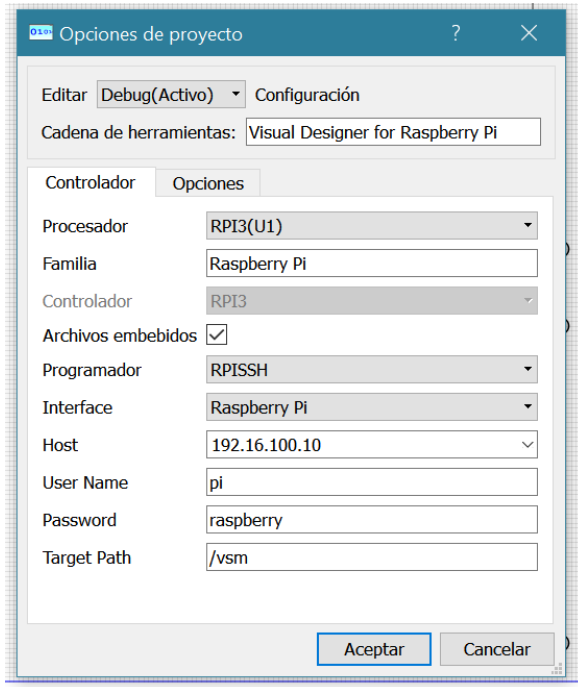
8.4.- Configurar el equipo Raspberry Pi para utilizarlo desde el entrono Proteus.

8.4.1.-Primer método: desde Proteus.

Crearemos un proyecto nuevo del tipo Raspberry Pi en Proteus. Desde la pestaña 'Diseñador Gráfico' seleccionamos la opción 'configurar proyecto' del menú 'Proyecto'



En la ventana de diálogo que aparece, seleccionamos la pestaña 'Controlador'. En la opción 'Programador' seleccionamos la opción 'RPISSH', en la opción 'Interface' seleccionamos 'Raspberry Pi' y en la opción 'Host' introducimos la dirección IP de nuestro equipo Raspberry Pi que localizamos en el apartado anterior.



A continuación, cargamos el proyecto en blanco en nuestro equipo Raspberry Pi. La ventana emergente que aparece nos pedirá nuestra confirmación para completar el proceso de configuración de nuestro equipo Raspberry Pi. Aceptaremos con el botón 'Aceptar'. Este proceso llevará un tiempo relativamente largo hasta que se carguen todas las librerías necesarias y se configure todo de forma adecuada.

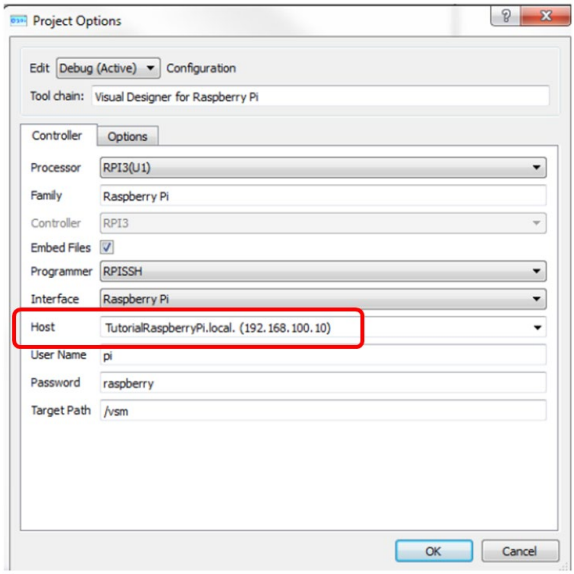
Al completarse todo el trabajo, el equipo Raspberry Pi se reiniciará y perderemos la conexión entre Proteus y el equipo recibiendo el código de error -1.



Debemos darle un tiempo hasta que el equipo Raspberry Pi pueda completar todo el proceso de arranque.

A partir de esta primera operación, el equipo Raspberry Pi será detectado por Proteus cada vez que lo conecte y ya no será necesario seguir utilizando el programa Advanced Port Scanner.

Si volvemos a entrar en la configuración del proyecto, el equipo Raspberry Pi aparecerá en el apartado Host con el nombre que le hayamos proporcionado cuando modificamos el archivo configure.sh que cargamos en la tarjeta SD.



8.4.2.-Segundo método: usando PuTTY.

También es posible utilizar la herramienta PuTTY para comunicarnos con nuestro equipo Raspberry Pi sin que le tengamos que conectar un teclado, ratón y pantalla.

En el siguiente enlace podemos descargar la utilidad PuTTY desde su página web:

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html



En el siguiente enlace encontraremos la guía completa con los pasos necesarios a seguir:

https://www.raspberrypi.org/documentation/remote-access/ssh/windows.md

Conectaremos PuTTY a nuestro equipo Raspberry Pi y escribiremos el comando:

sudo /boot/configure.sh

Llevará un tiempo hasta que todo se instala y configura correctamente. Una vez que el proceso se haya completado será necesario reiniciar el equipo Raspberry Pi.

8.4.3.-Tercer método: Utilizando teclado y ratón.

Si disponemos de teclado y pantalla conectados directamente a nuestro equipo Raspberry Pi podemos utilizar el sistema de introducción de comandos de sistema que, en el sistema raspbian recibe el nombre de 'terminal'. Una vez abierto podemos escribir el comando:

sudo /boot/configure.sh

Llevará un tiempo hasta que todo se instala y configura correctamente. Una vez que el proceso se haya completado será necesario reiniciar el equipo Raspberry Pi.

9.-Tutoriales de ejemplo de equipos Arduino.

En este apartado queremos mostrar una serie de tutoriales con ejemplos sencillos que muestran el proceso completo de construcción de proyectos Arduino realizados utilizando el Diseñador Gráfico para que el lector pueda aprender las diferentes técnicas que nos ofrece esta herramienta de forma práctica.

9.1.-Intermitencia sobre un led.

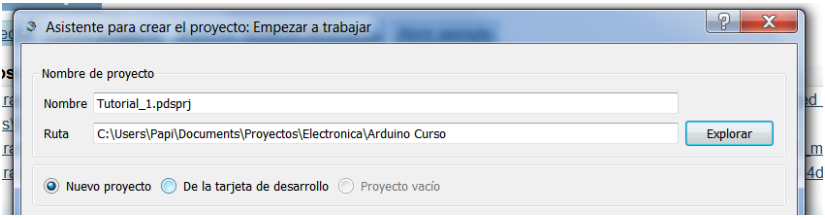
Este primer tutorial pretende abarcar el ciclo completo en la construcción de un proyecto Arduino. Arranca cuando creamos el proyecto en Proteus y finaliza cuando somos capaces de simular la intermitencia sobre el led. Nuestro objetivo es mostrar las técnicas más simples referentes a la construcción de proyectos.

9.1.1.-Utilizar el asistente para la creación de un nuevo proyecto.

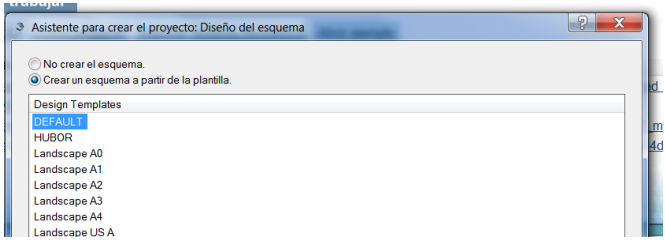
El Diseñador Gráfico está totalmente integrado dentro de la suite Proteus por lo que para crear un nuevo proyecto lo tendremos que hacer desde la pantalla de inicio de Proteus. La forma mejor y más sencilla es utilizar el asistente para crear proyectos que se ejecuta con el botón 'Nuevo Proyecto' que se encuentra en el panel 'Empezar a trabajar' de la pantalla de inicio.



Desde la ventana que se abre podemos seleccionar el nombre y el directorio (ruta) dónde se guardará nuestro proyecto.

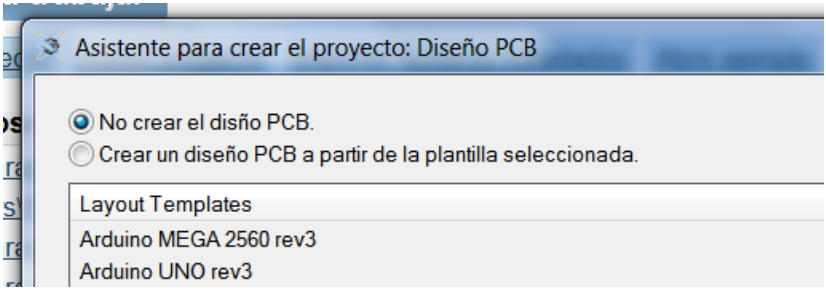


En la segunda ventana, después de darle al botón 'Siguiente', podemos seleccionar el tamaño de la hoja que deseamos utilizar para dibujar nuestro diseño electrónico. La opción por defecto será apropiada para la mayoría de los proyectos y, por supuesto, para nuestro primer proyecto de generación de una intermitencia sobre un led.

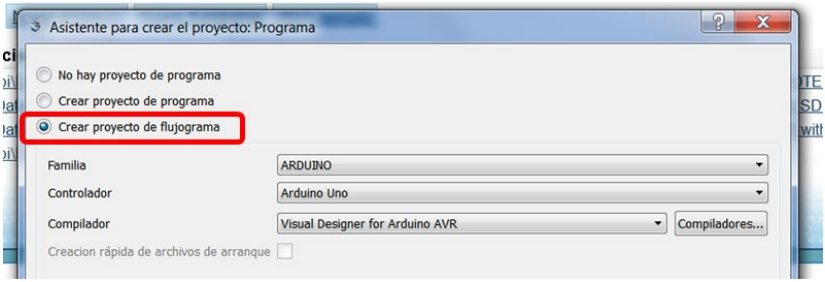


Si nuestra licencia de Proteus nos permite el diseño de placas de circuito impreso la siguiente pantalla del asistente nos invita a seleccionar las opciones disponibles para crear el PCB de nuestro

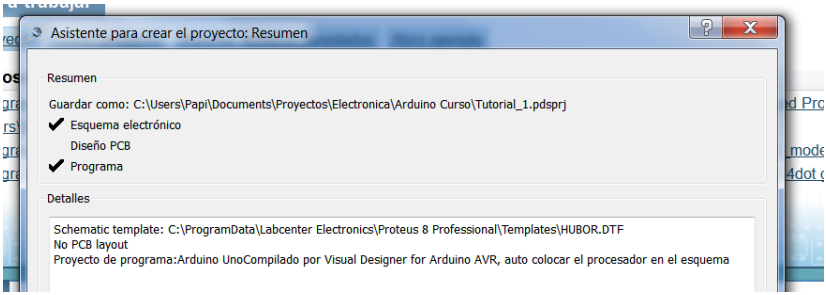
proyecto. Si nuestra licencia de Proteus no incluye el diseño de placas de circuito impreso esta ventana no se mostrará. En esta ayuda no vamos a entrar en la construcción de las placas de circuito impreso de nuestros proyectos, así que seleccionaremos la opción 'No crear el diseño PCB'.



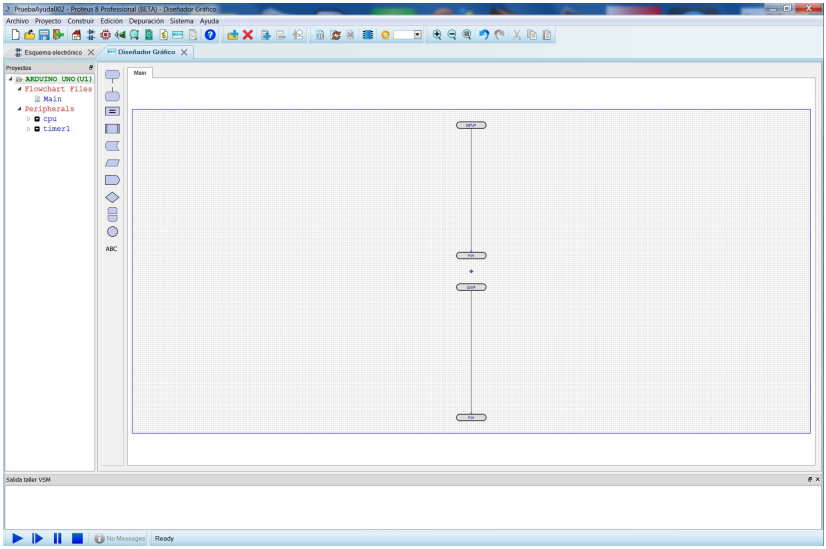
La siguiente pantalla nos permite configurar la parte del software de nuestro proyecto y en ella definiremos que queremos construir un proyecto utilizando El Diseñador Gráfico. La programación de equipos Arduino utilizando código escrito en C de forma tradicional también está disponible en Proteus pero no está contemplada en esta ayuda. Primero seleccionaremos en los botones de opciones de la parte superior la opción 'Crear proyecto de flujograma'. Luego seleccionaremos la familia 'Arduino', el controlador 'Arduino Uno' para nuestro proyecto desde los cuadros combinados y 'Visual Designer for Arduino AVR' como compilador.

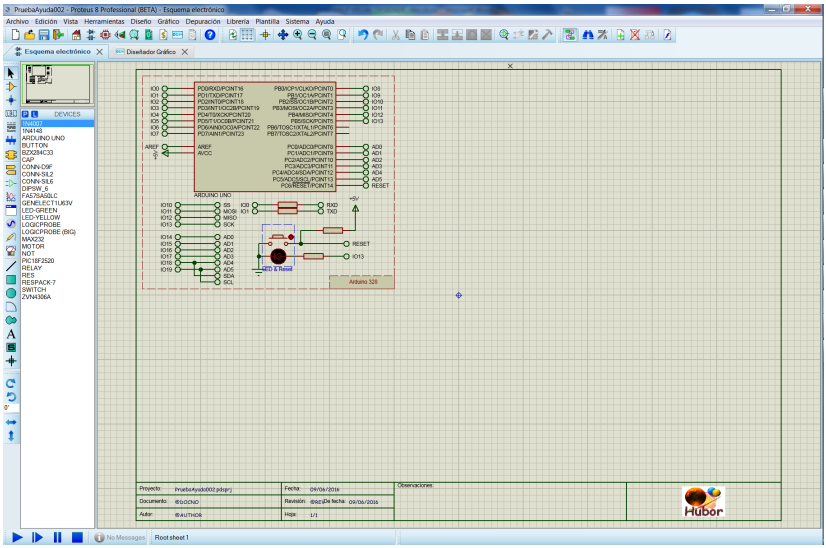


Finalmente se nos presenta una ventana con el resumen de todas las opciones que hemos seleccionado.

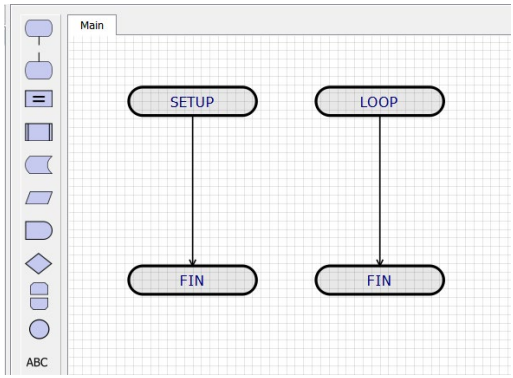


Si estamos de acuerdo con ellas, pulsaremos el botón finalizar y el asistente creará el proyecto con dos pestañas para la estructura básica del software (Diseñador Gráfico) y del hardware (Esquema electrónico).





El estándar Arduino incluye dos funciones llamadas setup() y loop(). La función setup() se utiliza para las labores de configuración e inicialización que se ejecutan una sola vez durante el arranque. La función loop() es el lugar donde colocamos toda la lógica de control de nuestro equipo y se ejecuta de forma cíclica. El Diseñador Gráfico utiliza el mismo paradigma de programación por lo que debemos tener al menos estos dos bloques en nuestro flujograma para que la compilación se pueda llevar a cabo. Si en un momento, borramos alguna de estas dos funciones, será necesario que las volvamos a crear y nombrar adecuadamente.

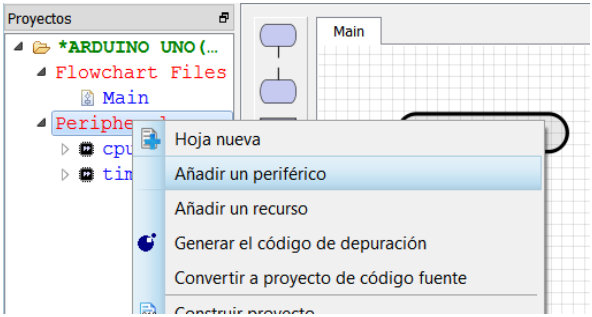


Ya estamos listos para empezar con el diseño de nuestro proyecto.

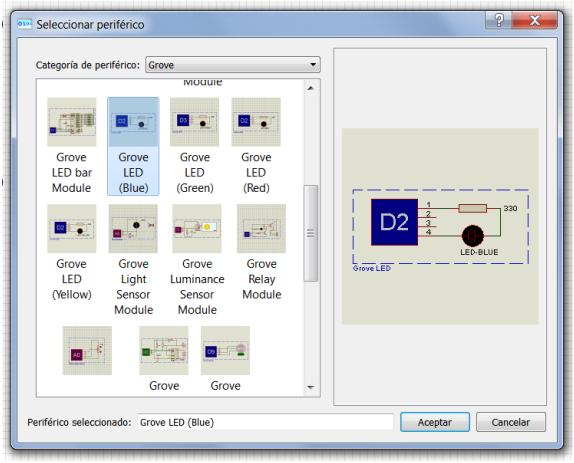
9.1.2.-Añadir los periféricos.

El primer paso lógico a dar en la construcción de nuestro proyecto es añadir los periféricos que queramos utilizar. Podemos utilizar como periféricos tarjetas de expansión Arduino o módulos Grove. El sistema Grove está diseñado específicamente para la experimentación con los equipos Arduino así que vamos a utilizar un módulo Grove LED para nuestro primer proyecto.

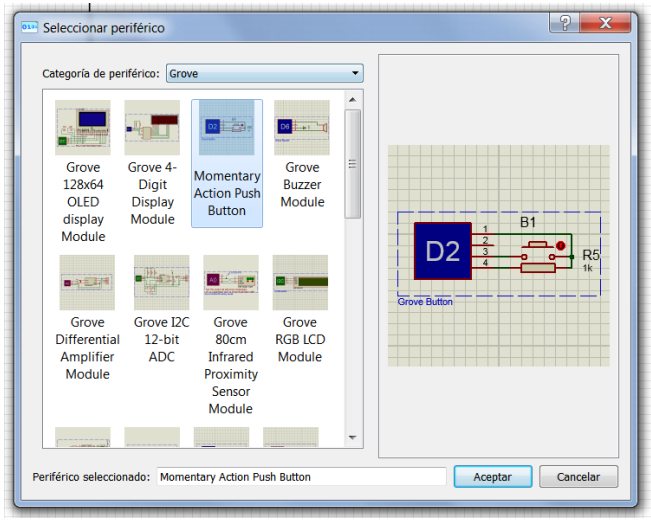
Para añadir un módulo Grove Led pulsaremos el botón derecho del ratón sobre el árbol de estructura del proyecto y seleccionaremos la opción 'Añadir periférico'.



A continuación, en la nueva ventana de diálogo que aparece, seleccionaremos en el cuadro de diálogo desplegable la opción Grove y el módulo 'Grove LED (blue)'.

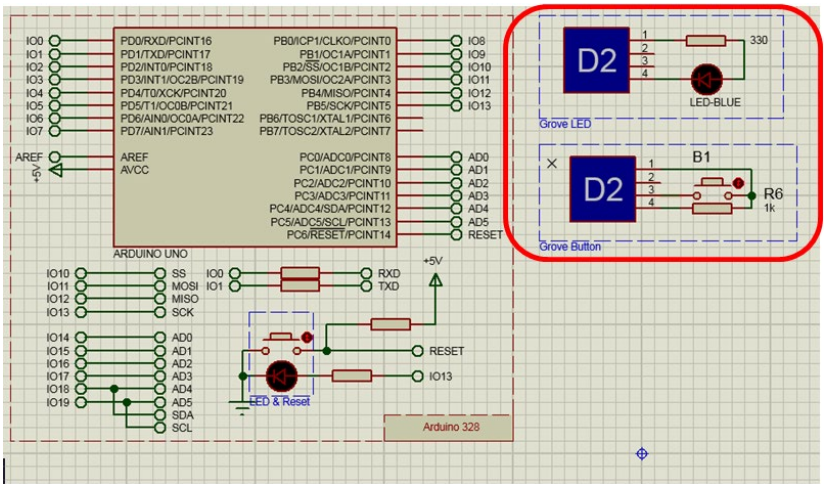


Repetiremos el proceso para añadir un módulo ‘Grove Momentary Action Push Button’ y disponer de un pulsador en nuestro proyecto.

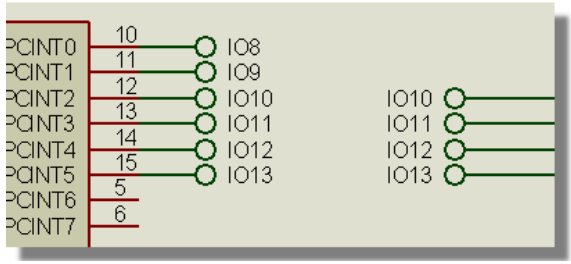


Una vez que hemos añadido nuestros dos periféricos, tenemos que ser capaz de verlos, tanto en la pestaña del Diseñador Gráfico (en el árbol de estructura de proyectos) como en la pestaña del esquema electrónico.

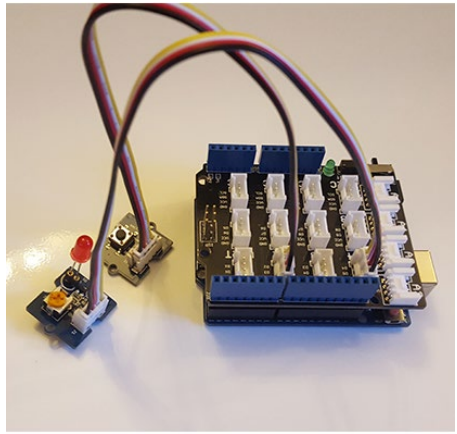




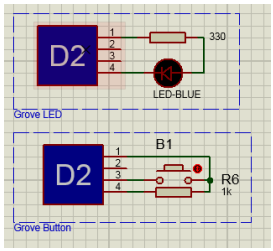
Para evitar confusiones en el esquema electrónico, las conexiones se han realizado utilizando terminales del mismo nombre. Todos los terminales utilizados en el proyecto que tienen el mismo nombre están internamente enlazados como si existiera un cableado invisible.



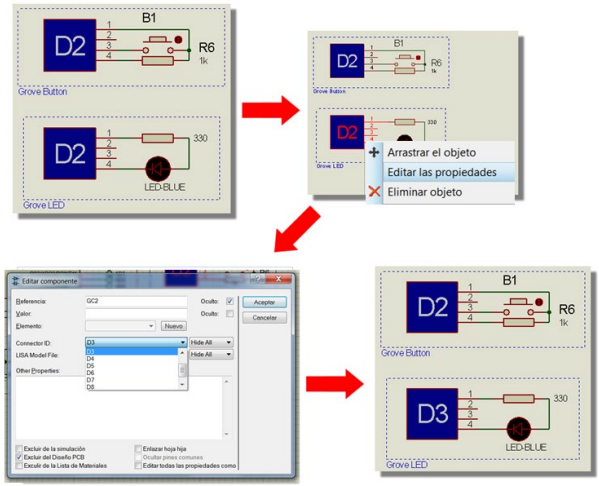
Lo que hemos hecho tiene su equivalencia en el mundo real cuando conectamos un módulo Grove LED y un módulo Grove Button a la tarjeta de expansión Grove que está conectada a su vez a la placa Arduino Uno.



Pero en la imagen podemos fijarnos que cada módulo Grove se conecta en un enchufe de la tarjeta de expansión Grove diferente. En la vida real es imposible confundirse y enchufar dos módulos en el mismo conector. Pero en la simulación esta confusión si puede producirse. Al crear un nuevo periférico, Proteus le puede asignar un conector por defecto que ya está siendo utilizado. Debemos asegurarnos de que no tenemos dos periféricos con el mismo ID de conector en nuestro esquema electrónico. En nuestro ejemplo los dos módulos utilizan el mismo conector 'D2'.

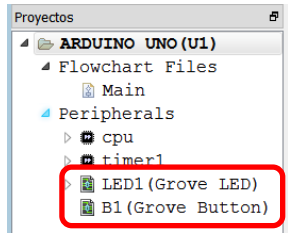


Así que tenemos que cambiar el ID del conector de uno de ellos.



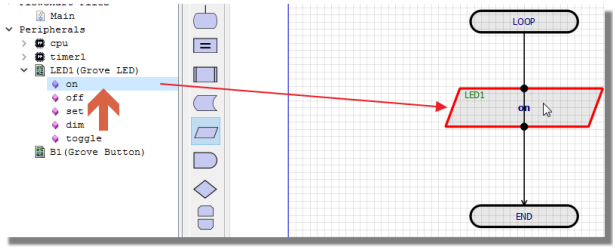
9.1.3.-Construir el flujograma.

Volvamos a la pestaña Del Diseñador Gráfico. A la izquierda podemos ver el árbol de estructura del proyecto y en él podemos encontrar dos ramales nuevos debajo de la sección periféricos que se corresponden con los dos módulos Grove que hemos creado.

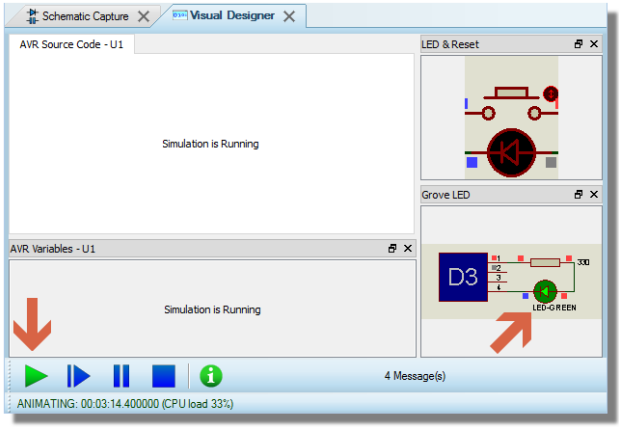


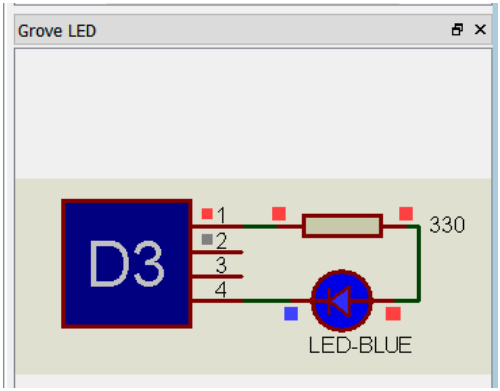
Una de los aspectos más potentes del Diseñador Gráfico es que nos permite expandir la rama dedicada a cada periférico y encontrar la lista completa de los métodos disponibles para gestionarlo. Los métodos son la herramienta básica que nos posibilita interactuar con el hardware.

Veamos un ejemplo. Cojamos y arrastremos el método 'On' asociado al módulo LED1 y lo colocaremos dentro de la rutina loop().



Con esta sencilla operación hemos ordenado que el led se encienda. Ejecutemos la simulación utilizando el botón PLAY del cuadro de controles de la simulación. El programa se compilará y podremos observar en la ventana emergente activa de la zona derecha cómo el led se enciende.

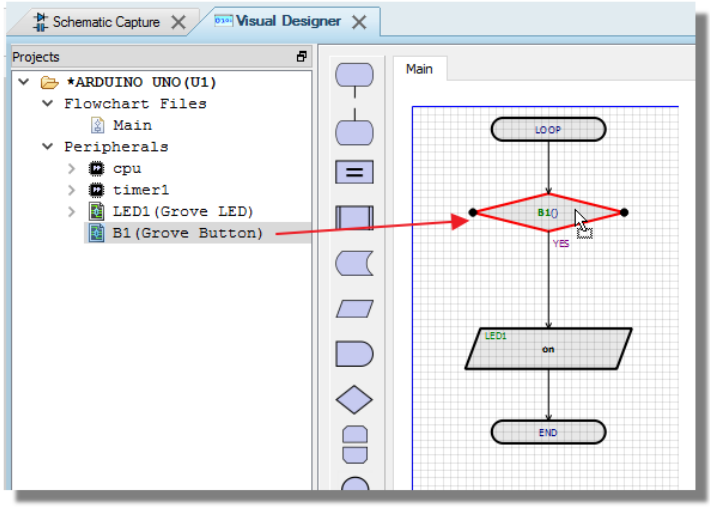




Presionando el botón STOP del cuadro de controles de la simulación la detendremos. Vamos ahora a ampliar nuestro código para que el led se encienda o apague cuando presionemos o liberemos el pulsador.

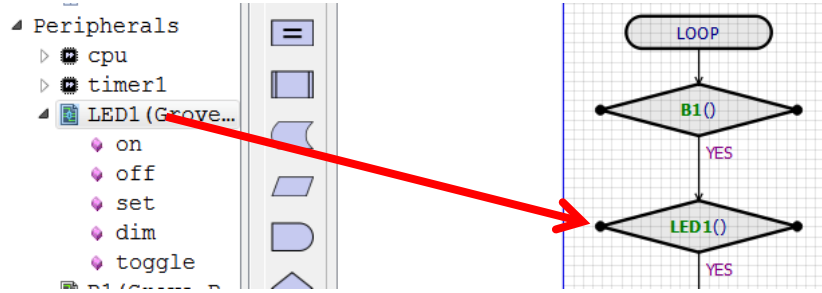


Lo primero que tenemos que destacar es que el módulo Grove 'Button' no tiene ningún método asociado en el árbol de estructura del proyecto. La única información sensible necesaria que nos proporciona un pulsador es saber cuándo está siendo actuado. Esta información nos la proporciona la función que llamaremos 'función sensor' que es la que se ejecuta siempre por defecto. Por este motivo, en el caso de un módulo botón, puesto que no es necesario llamar a ninguna otra función asociada con este módulo sólo se dispone de la función por defecto. El módulo 'Grove Button' nos devolverá un valor 'TRUE' o 'FALSE' dependiendo de su estado. Así que cogeremos y arrastraremos hasta nuestro flujograma el bloque 'Grove Button' y automáticamente se colocará la 'función sensor' asociada.

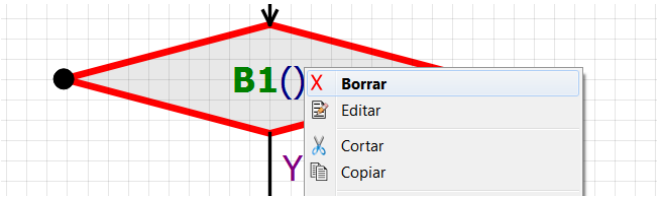


Podemos comprobar que el bloque que se ha colocado sobre el flujograma es uno del tipo bloque de decisión. Cuando la evaluación del pulsador devuelva un TRUE (el pulsador está siendo presionado) el flujograma seguirá por el ramal dónde está situada la palabra YES.

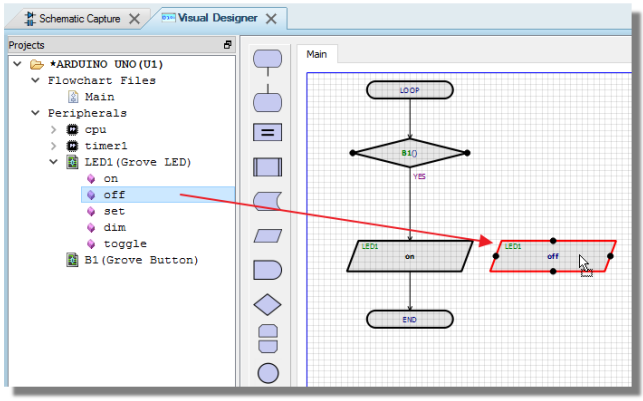
De forma similar si cogiéramos el módulo LED y lo arrastráramos hasta nuestro flujograma también se colocaría un tipo de bloque de decisión que devolverá el valor TRUE cuando el led esté luciendo y un valor FALSE cuando el led esté apagado (la 'función sensor' por defecto asociada a ese elemento).



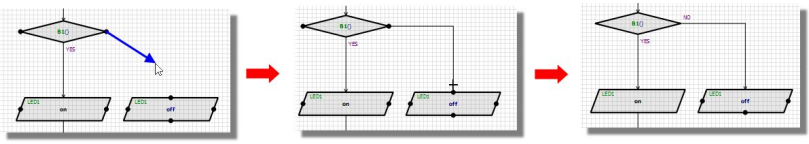
Eliminemos el bloque de la función sensor del led que acabamos de crear, situándonos sobre él, y utilizando la opción ‘Borrar’ del menú contextual y continuemos con nuestro proyecto.



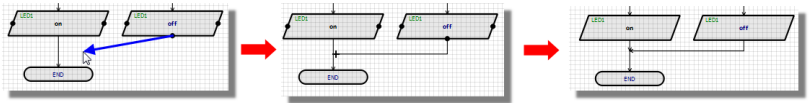
Hasta ahora estamos comprobando constantemente el estado del pulsador y cuándo está presionado ejecutamos la orden de que el led se encienda. Nos toca ahora decirle a nuestro microprocesador que tiene que hacer cuando el pulsador esté liberado. Deseamos que el led se apague al dejar de presionar el pulsador. Así que cojamos y arrastremos el método ‘off’ asociado al led y coloquémoslo al lado del bloque ‘on’ situado ya en nuestro flujograma.



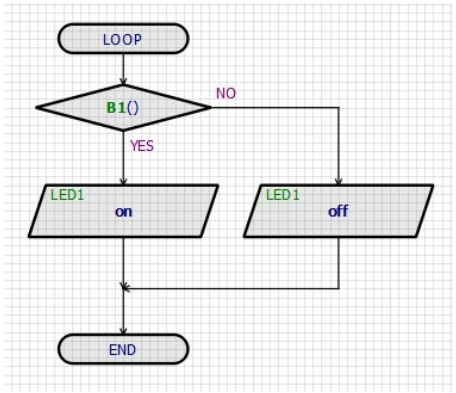
Tenemos que unir el bloque de decisión que comprueba el estado del pulsador con la parte superior del bloque operación que ordena apagar el led. Para hacerlo, debemos colocar el ratón sobre el punto de anclaje derecho del bloque de decisión, pulsar el botón derecho del ratón, llevar el cursor hasta el punto de anclaje de la parte superior del bloque OFF y volver a pulsar el botón derecho del ratón.



De igual manera tenemos que conectar el anclaje inferior del bloque OFF con la línea que une el bloque ON y el bloque END.

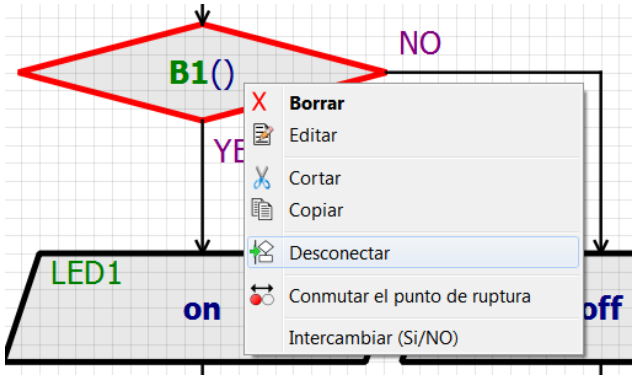


Nuestro flujograma finalizado tiene que tener el aspecto que se muestra en la imagen siguiente.

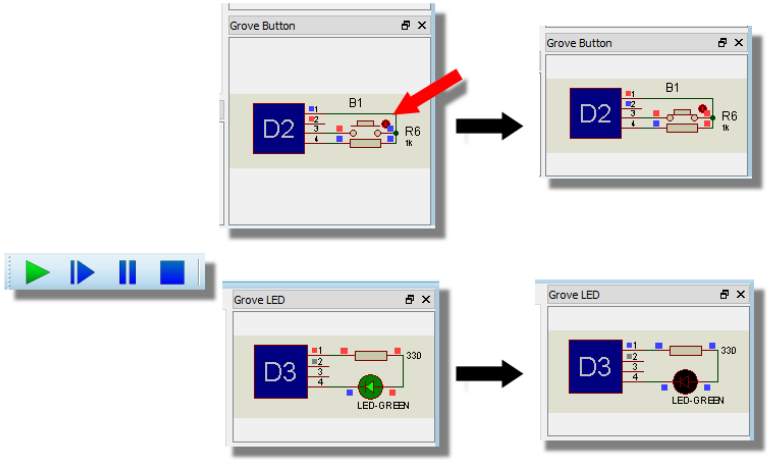


El lector debe fijarse que el bloque de decisión tiene ahora dos etiquetas (YES y NO) que indican que ruta seguirá el programa cuando el resultado de la evaluación sea verdadera (YES) y cuál cuando sea falsa (NO).

Si nos viniera mejor que las etiquetas estuvieran intercambiadas y el NO quedara en la zona inferior y el YES en la zona derecha, podemos intercambiar sus posiciones utilizando la opción 'Intercambiar (SI/NO)' del menú contextual asociado al bloque.



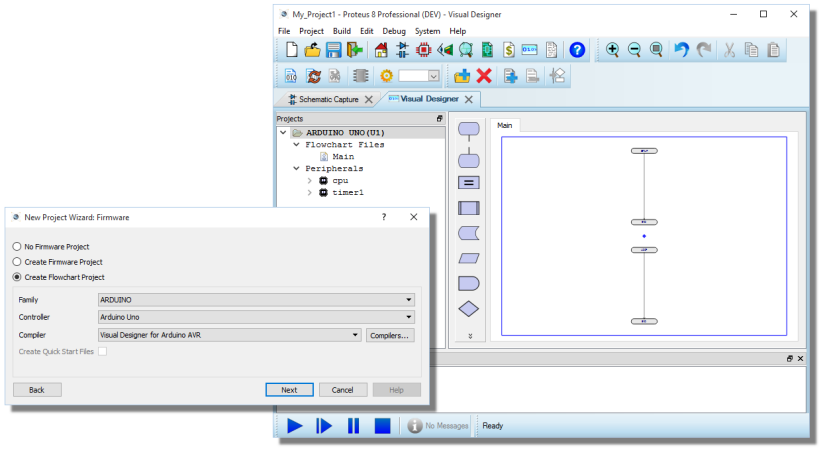
Una vez hechos todos los cambios ya podemos volver a compilar y arrancar la simulación para comprobar el funcionamiento de nuestro proyecto. Arrancaremos la simulación con el botón PLAY del cuadro de control de la simulación y podremos utilizar las ventanas emergentes activas para actuar sobre el pulsador y ver el efecto obtenido en el led.



9.2.-Lámpara de iluminación nocturna.

En este segundo tutorial vamos a controlar una lámpara de iluminación nocturna utilizando un par de módulos de sensor y un módulo de led Grove. La utilización del asistente para la creación de un proyecto nuevo se explicó en el primer tutorial (apartado 8.1.1),

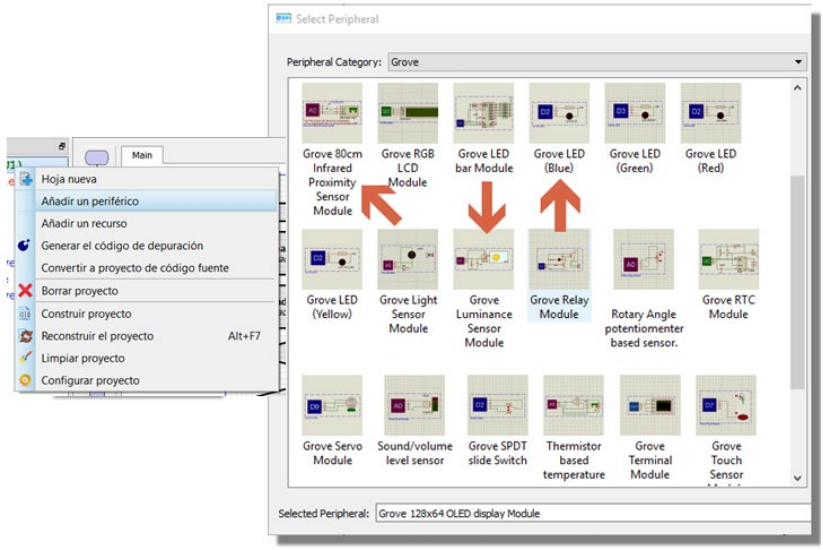
así que en este caso asumiremos que el lector ya ha creado un nuevo proyecto basado en la placa Arduino UNO, seleccionando como compilador el Diseñador Gráfico (Visual Designer) y que la estructura por defecto ya está lista para ser utilizada.



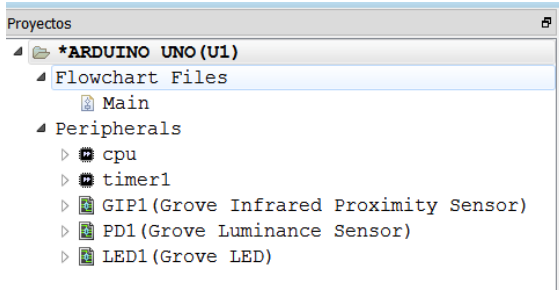
9.2.1.-Añadir los periféricos.

Vamos a utilizar un módulo Grove con un sensor de proximidad por infrarrojos, un módulo Grove con un sensor de luz y un módulo Grove con un led.

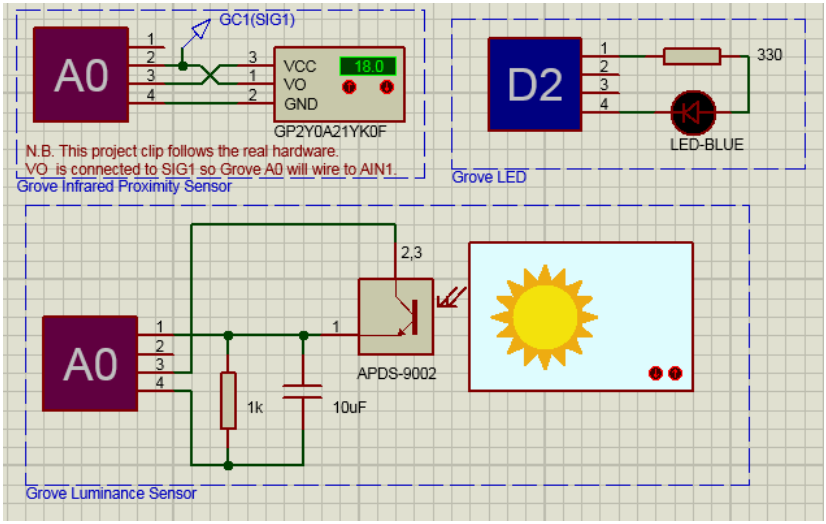
Todos ellos, los podemos incluir en nuestro proyecto utilizando el comando ‘Añadir un periférico’ desde el menú contextual del árbol de estructura del proyecto.



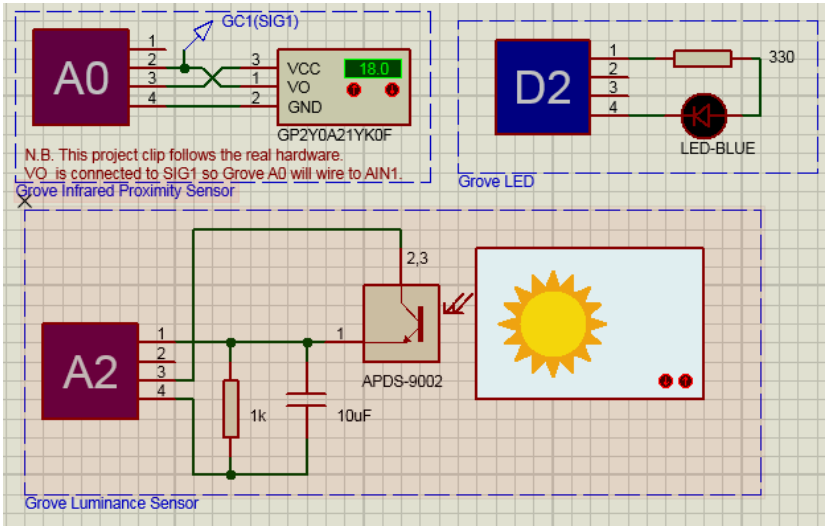
Al terminar de añadirlos a nuestro proyecto, tenemos que ser capaces de visualizar en el árbol de estructura de proyecto los métodos disponibles para su control.



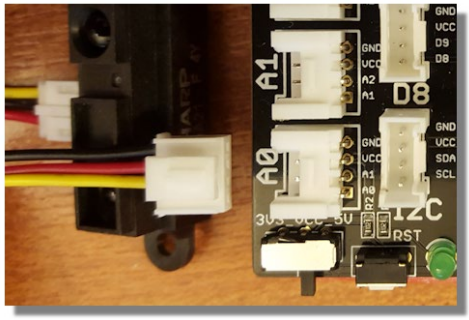
Y en la pestaña Esquema electrónico todo el hardware virtual de los periféricos recién creados.

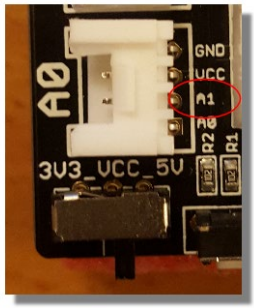


El lector puede comprobar que tanto el módulo del sensor de infrarrojos como el módulo del sensor de luminosidad ambos tienen como id el valor A0. Debemos cambiarlos manualmente para que el primero tenga el valor A0 y el segundo el valor A2 siguiendo el proceso ya visto en el apartado 9.1.2.



En los proyectos desarrollados con el Diseñador Gráfico es de vital importancia configurar los conectores de los módulos Grove adecuadamente. Esta operación no siempre es tan sencilla como parece. Veamos un ejemplo. El sensor de infrarrojos requiere sólo tres cables para su conexión (VCC, GND y A0), quedando el cuarto como NC (no conectado). Sin embargo, si nos fijamos bien, cuando enchufamos nuestro módulo Grove en el enchufe rotulado A0 el sensor se conecta, no sólo al pin A0, sino también al pin A1 de nuestro Arduino.

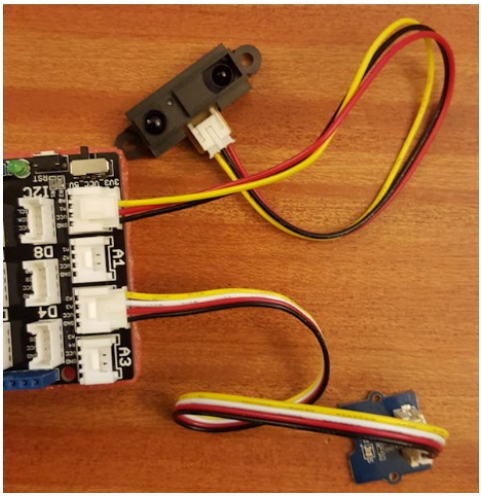




Hasta aquí todo correcto y sin problemas. Pero si enchufamos ahora nuestro sensor de luminosidad en el enchufe A1, utilizará también el

pin A1 de nuevo (porque el pin A1 de nuestro Arduino se utiliza en ambos, el enchufe A0 y el enchufe A1) y se generará un conflicto.

Por este motivo tenemos que enchufar nuestro sensor de luminosidad en el enchufe A2 y dejar libre el A1.



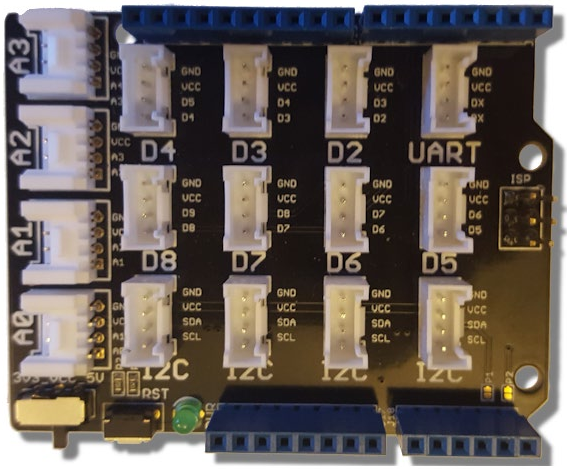


Debemos tener siempre muy en cuenta esta regla de oro cuando utilicemos módulos Grove: hay que fijarse detalladamente en las rotulaciones de la tarjeta de expansión Grove para Arduino, comprobar bien los pines utilizados por cada enchufe y prevenir los conflictos entre los diferentes módulos que conectemos a la tarjeta de expansión.

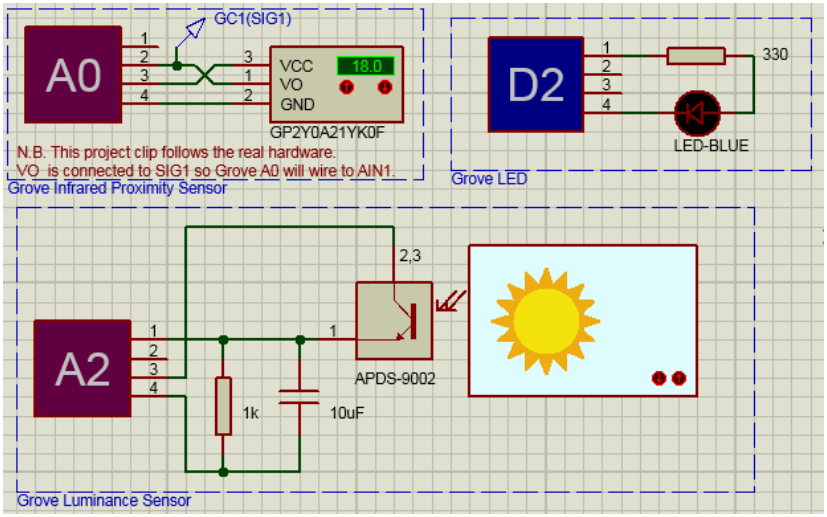
Cada enchufe se conecta a dos pines de entrada/salida de nuestro microprocesador Arduino y existe un solapamiento entre los diferentes enchufes:

- El enchufe A0 utiliza los pines A0 y A1.
- El enchufe A1 utiliza los pines A1 y A2.
- El enchufe A2 utiliza los pines A2 y A3.
- El enchufe A3 utiliza los pines A3 y A4.

Con los enchufes que se conectan a los pines digitales sucede exactamente lo mismo.



Por este motivo hemos configurado nuestro proyecto utilizando el sensor de proximidad conectado al enchufe A0 y el sensor de luminosidad al enchufe A2. El led lo podemos conectar en cualquiera de los enchufes destinados a módulos digitales (así que no tenemos problemas en conectarlo al enchufe D2 que Proteus ha asignado automáticamente).

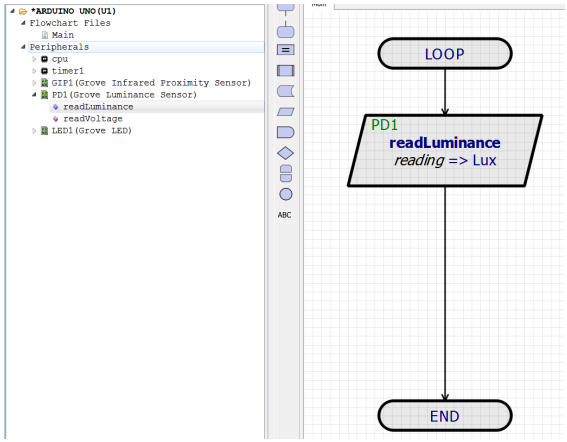


9.2.2.-Construir el flujograma.

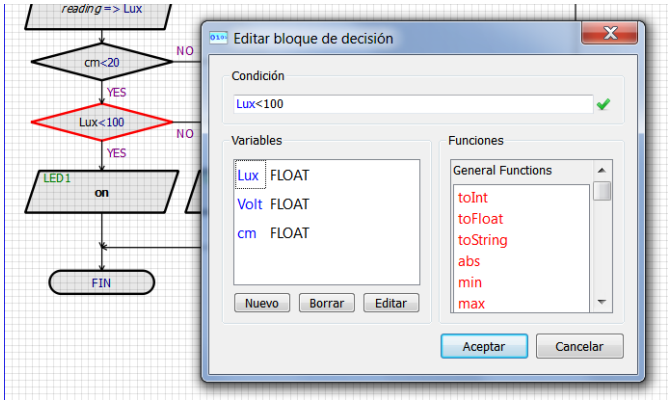
La lógica de funcionamiento de nuestra lámpara de iluminación nocturna es muy sencilla: nuestro led se iluminará cuando esté oscuro (sensor de luminosidad) y alguien se encuentre cerca (sensor de proximidad por infrarrojos).

Así que tenemos que incluir estas dos condiciones en nuestro flujograma.

Empecemos cogiendo y arrastrando el método readLuminance() del sensor de luminosidad hasta el interior de la subrutina loop().

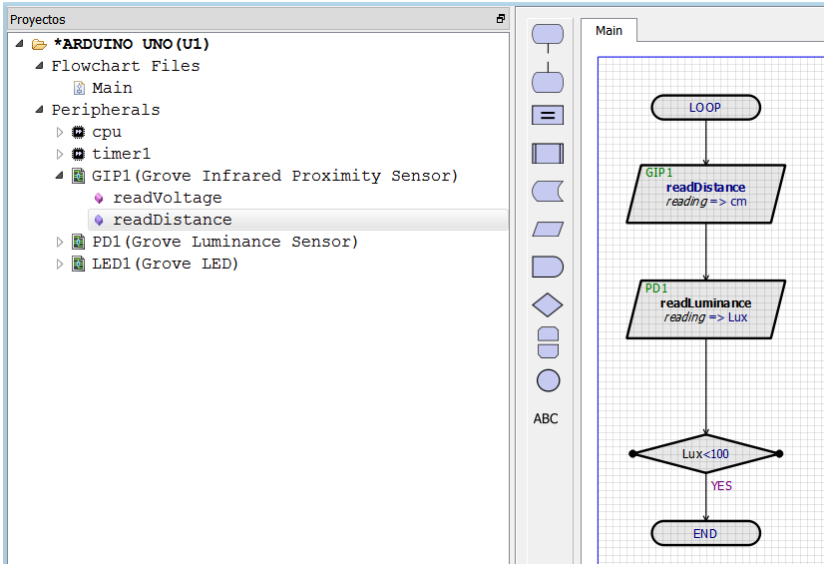


Este sensor devuelve el valor en lux medido en un rango comprendido entre 0 y 1.000, siendo 0 la oscuridad total y 1.000 la iluminación máxima. Así que podemos utilizar el resultado obtenido dentro de un bloque de decisión probando con un primer valor de 100.



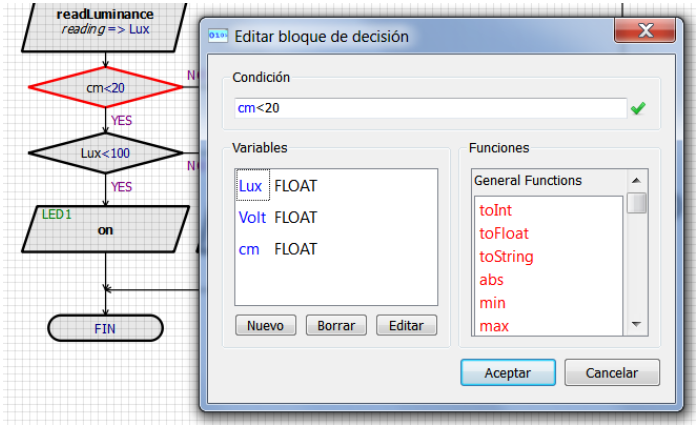
La hoja de datos del módulo del sensor de luz Grove APDS-9002 nos suministra toda la información sobre los valores medidos en lux. Consulte la información completa en el enlace: http://www.seeedstudio.com/wiki/Grove_-_Luminance_Sensor

Para el sensor de proximidad por infrarrojos necesitamos recibir la lectura de la distancia a la que se encuentra el objeto que rompe el haz de infrarrojos. Así que utilizaremos el método readDistance() y lo colocaremos encima del bloque que realizaba la lectura de la luminosidad.

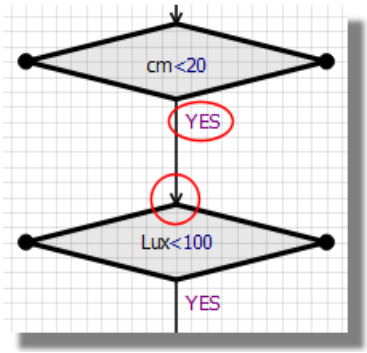


El sensor de proximidad por infrarrojos de Grove puede leer objetos hasta una distancia máxima aproximada de 80 centímetros.

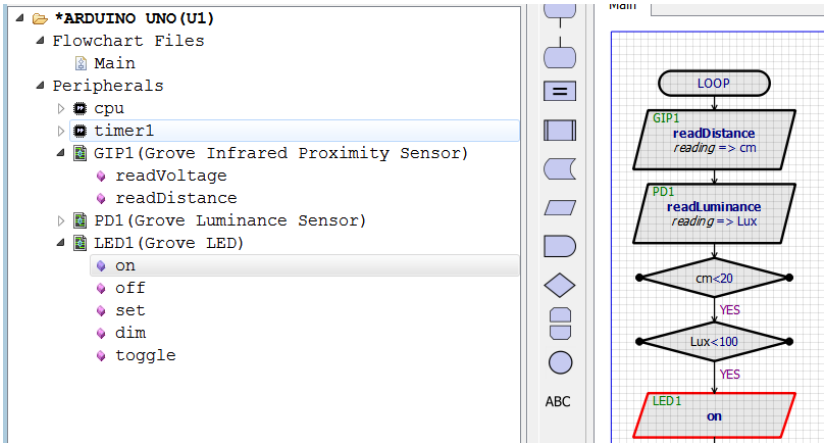
Así que fijaremos un valor inicial para nuestras pruebas de 20 cm utilizando un bloque de condición colocándolo encima del anterior bloque de condición.



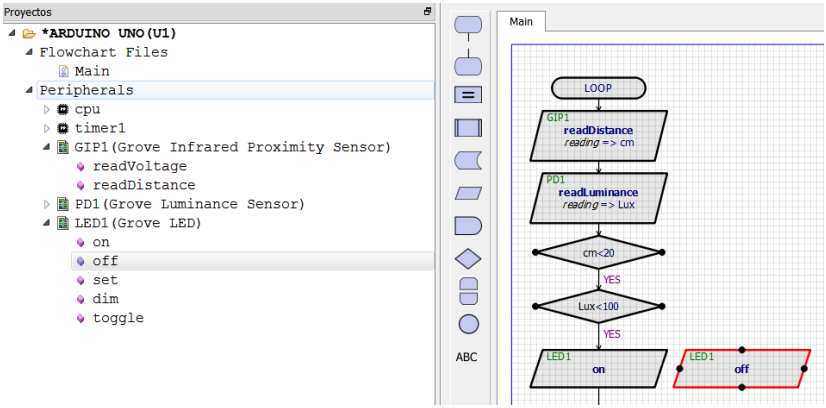
Puesto que necesitamos que se cumplan las dos condiciones para que encender el led, debemos asegurarnos que el segundo bloque de decisión está en la rama de la respuesta afirmativa del primer bloque de decisión.



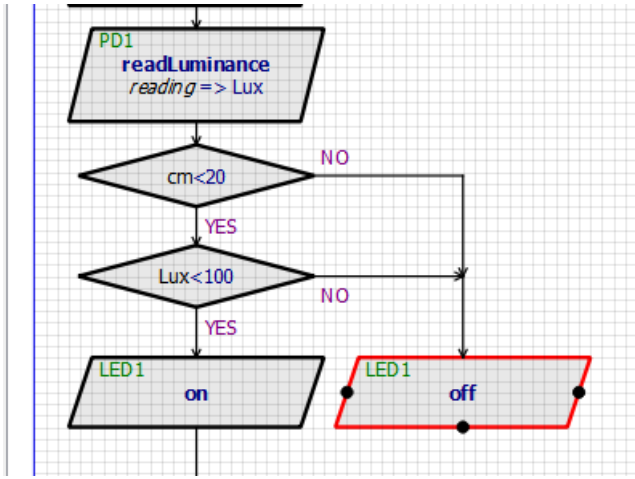
El siguiente paso será colocar un bloque con el método ‘on’ del led situado inmediatamente debajo del segundo bloque de decisión y en la rama de la respuesta afirmativa.



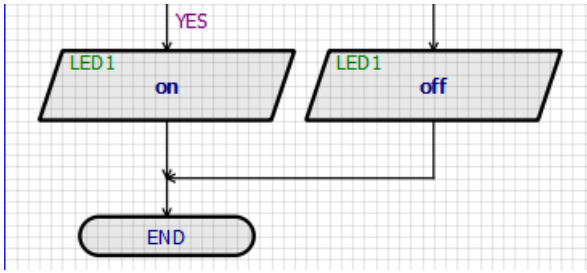
Sólo nos resta colocar un bloque con el método ‘off’ del led y situarlo a la derecha del bloque con el método ‘on’.



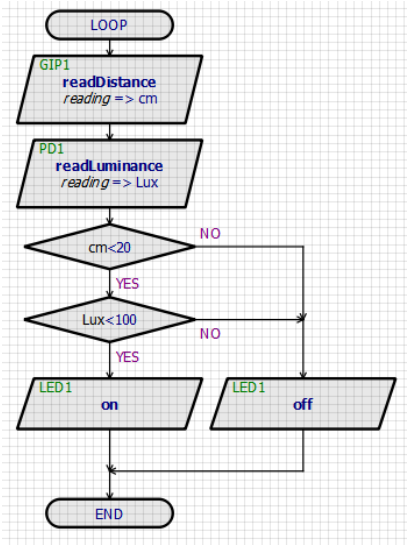
Tenemos que conectar el punto de anclaje superior del módulo off a la salidas FALSE de los dos bloques de condición existentes.



Y terminaremos todo nuestro flujograma conectando el punto de anclaje inferior del módulo off con el hilo que conecta el módulo on y el end.



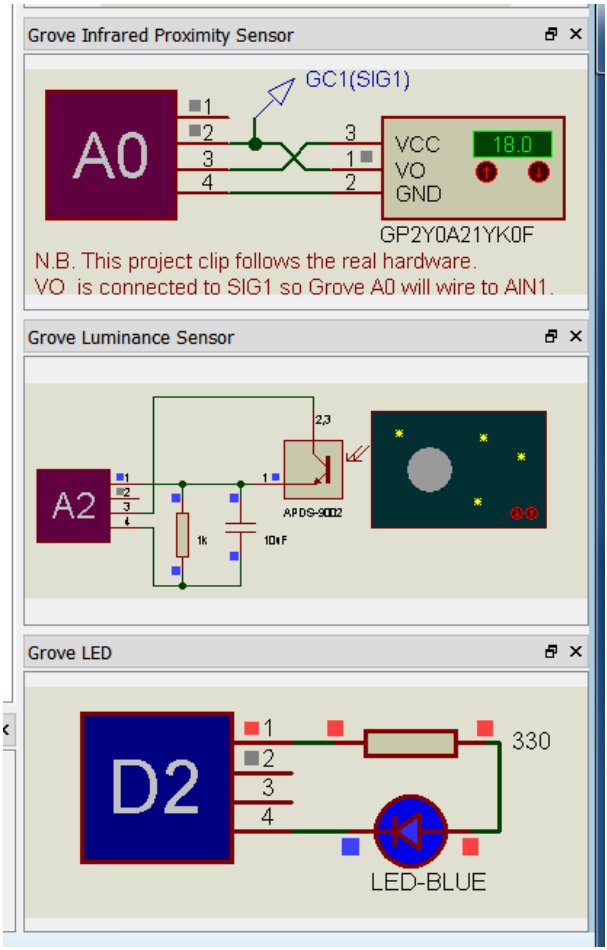
El flujograma completo debe mostrar el aspecto que se recoge en la siguiente imagen.

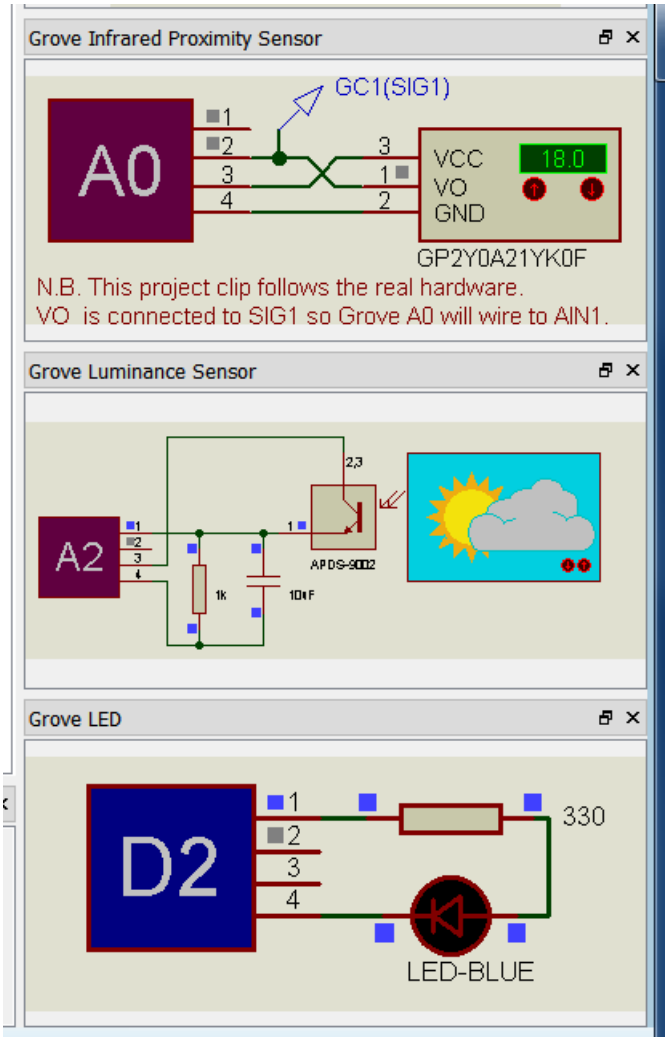


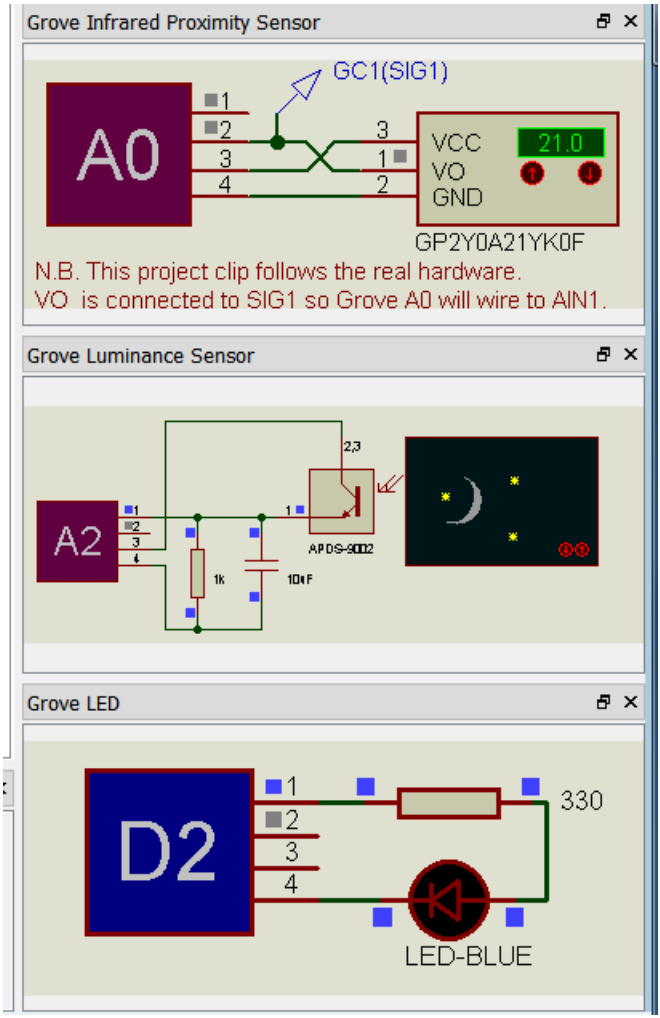
9.2.3.-Simular el proyecto.

Para comprobar el funcionamiento de nuestro proyecto sólo necesitamos utilizar el botón PLAY del cuadro de control de la simulación. Una vez arrancada la simulación podemos utilizar las ventanas emergentes activas para modificar los valores de lectura de los sensores y observar el efecto en el led.

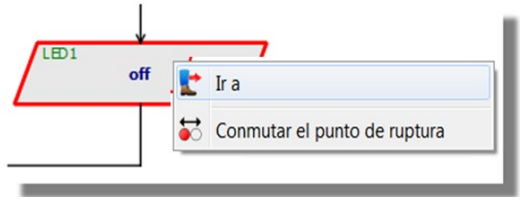
En las siguientes imágenes se muestran varias situaciones tipo de pruebas.


Grove Infrared Proximity Sensor
GC1(SIG1)
A0
1
2
3
4
3
1
2
VCC
VO
GND
18.0
GP2Y0A21YK0F
N.B. This project clip follows the real hardware.
VO is connected to SIG1 so Grove A0 will wire to AIN1.
Grove Luminance Sensor
A2
2,3
1
APDS-9002
1k
10nF
Grove LED
D2
1
2
3
4
330
LED-BLUE

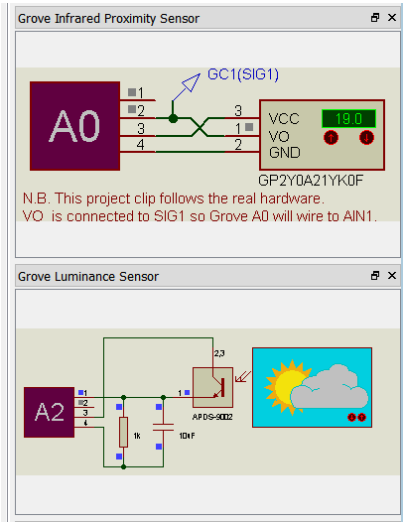

Grove Infrared Proximity Sensor
GC1(SIG1)
A0
1
2
3
4
3
1
2
VCC
VO
GND
18.0
GP2Y0A21YK0F
N.B. This project clip follows the real hardware.
VO is connected to SIG1 so Grove A0 will wire to AIN1.
Grove Luminance Sensor
A2
2,3
1
APDS-9002
1k
10nF
Grove LED
D2
1
2
3
4
330
LED-BLUE


Grove Infrared Proximity Sensor
GC1(SIG1)
A0
1
2
3
4
3
1
2
VCC
VO
GND
21.0
GP2Y0A21YK0F
N.B. This project clip follows the real hardware.
VO is connected to SIG1 so Grove A0 will wire to AIN1.
Grove Luminance Sensor
A2
2,3
1
APDS-9002
1k
10nF
Grove LED
D2
1
2
3
4
330
LED-BLUE

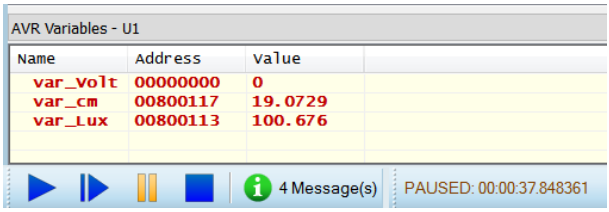
Dejemos las condiciones en las que el led permanece encendido y pongamos la simulación en pausa y asociemos un punto de ruptura con el método OFF del módulo LED.



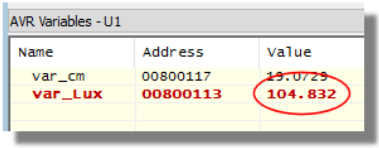
Reanudemos la simulación y modifiquemos la distancia medida por el sensor de infrarrojos o la luminosidad hasta que alcancemos uno de los puntos de ruptura y el led se apague.

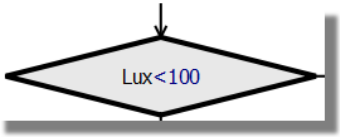


La simulación se pondrá en pausa de forma automática en la ventana de inspección podemos comprobar los valores leídos de las variables justo en el momento de producirse el punto de ruptura.

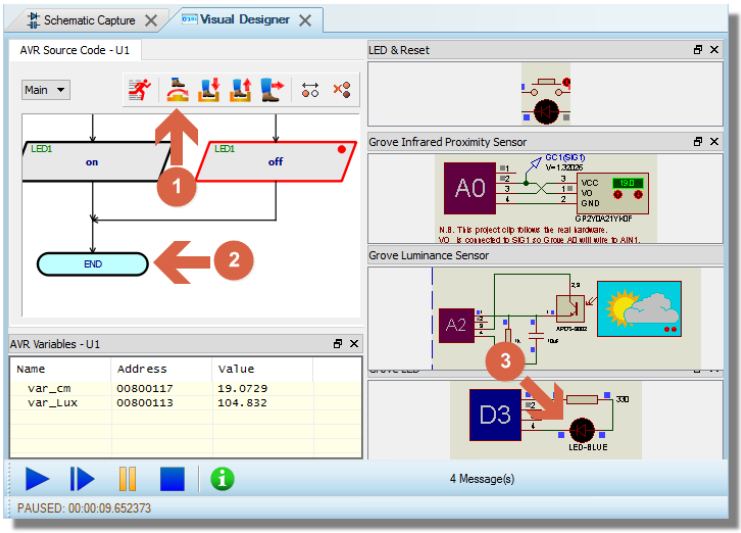


Observando los valores es muy fácil saber cuál de las dos variables ha generado la condición para provocar el apagado del led.





Con un avance paso a paso de la simulación que podemos llevar a cabo utilizando los iconos de la barra de herramientas (o utilizando el menú 'Depuración' o con el atajo de teclado F10), podemos comprobar que el próximo resultado de ejecutar nuestro código es, efectivamente, el apagado del led.

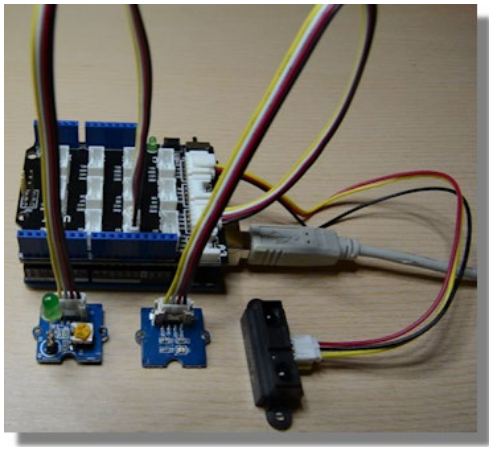


9.2.4.-Cargar el proyecto en un equipo Arduino real.

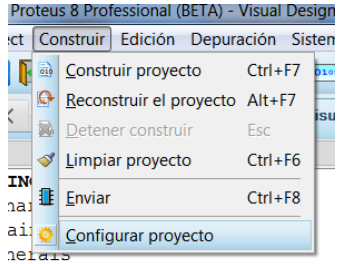
Una vez que hemos comprobado que nuestro proyecto funciona correctamente según lo previsto, ya estamos en condiciones de enchufar nuestro equipo Arduino y cargar el programa sobre el hardware real.

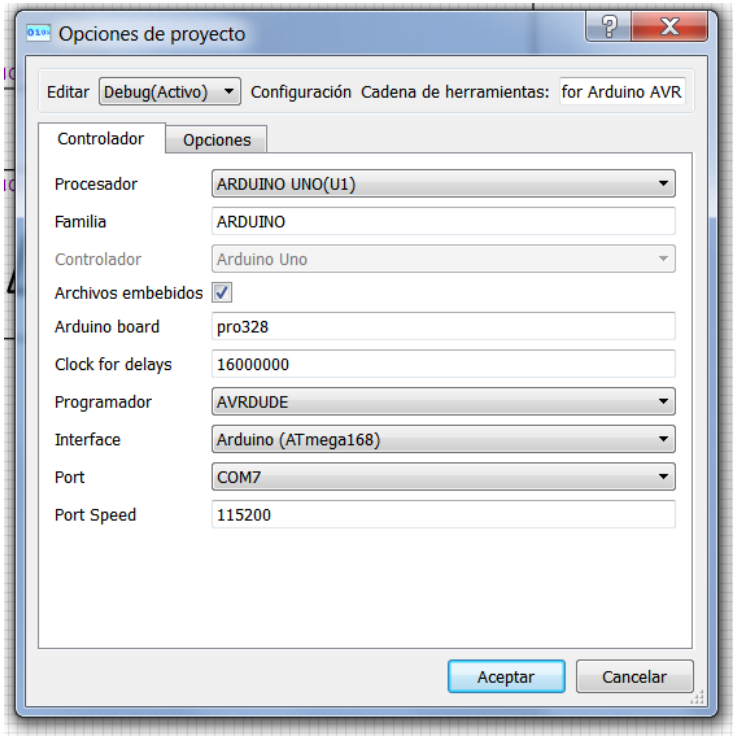
Conectaremos cada uno de los sensores Grove a los enchufes correspondientes de acuerdo con nuestro esquema electrónico de la tarjeta de expansión Grove. Tener siempre presente la regla de oro que mencionamos en el punto 8.2.1.

Conectar la tarjeta de expansión Grove a la placa base del Arduino y éste al puerto USB de nuestro ordenador personal dónde tenemos instalado Proteus.

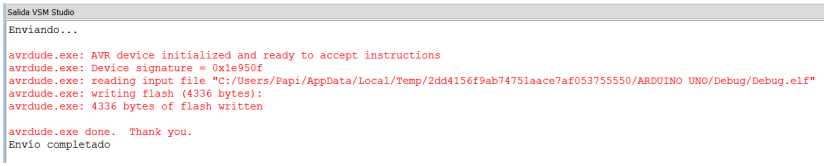


Utilizar la opción de menú ‘Configurar proyecto’ del menú ‘Construir’ para configurar el puerto dónde hemos conectado nuestro equipo Arduino.





En la ventana de estado, podemos comprobar que todo se ha producido conforme a lo esperado y que el programa se ha cargado correctamente en nuestro equipo Arduino.



Si el proceso falla conviene comprobar que hemos seleccionado correctamente el puerto y la velocidad para la que esté configurado nuestro Arduino. Seguir las instrucciones del punto 7.2 en lo referente a la selección de la velocidad del puerto serie.

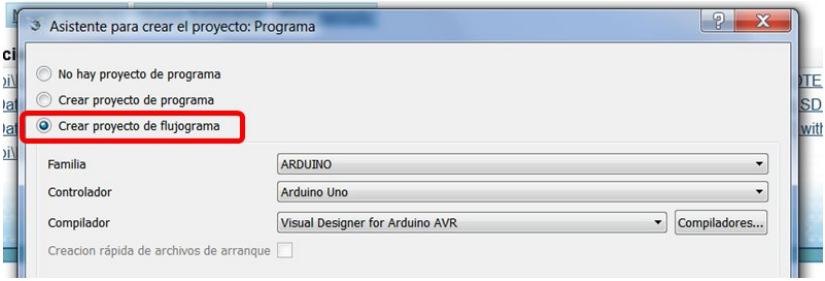
9.3.-Almacenamiento de datos usando una tarjeta SD.

En este tercer tutorial nos introducirá en la gestión del modelo de almacenamiento de datos. Vamos a crear un proyecto que visualiza en un display TFT un mapa de bits que se encuentra almacenado en una tarjeta de memoria SD. Los mismos principios son fácilmente transportables para hacer lo mismo con ficheros de audio (.wav) utilizando como salida una tarjeta de expansión del tipo Wave.

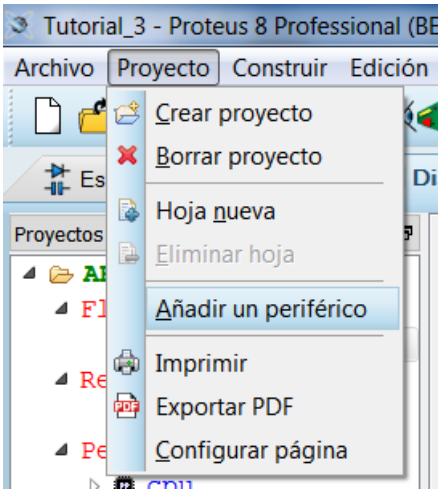
En esta ocasión asumiremos que el lector ya ha trabajado los dos primeros tutoriales y que está suficientemente familiarizado con las técnicas generales de colocación y conexión de bloques dentro de un flujograma.

9.3.1.-Añadir los periféricos.

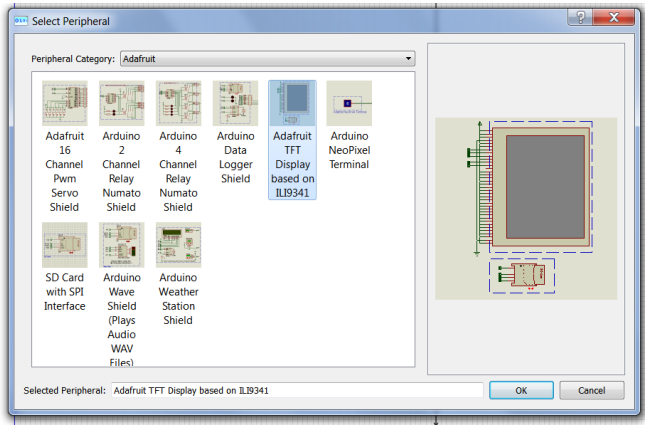
La utilización del asistente para la creación de un proyecto nuevo se explicó en el primer tutorial (apartado 8.1.1), así que en este caso asumiremos que el lector ya ha creado un nuevo proyecto basado en la placa Arduino UNO, seleccionando como compilador el Diseñador Gráfico y que la estructura por defecto ya está lista para ser utilizada.



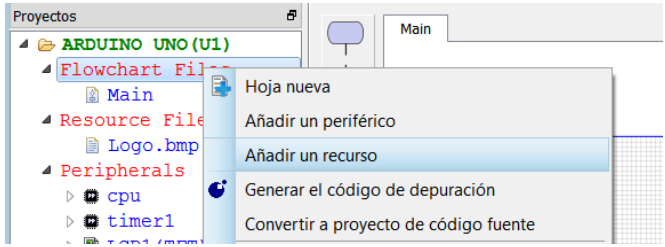
A continuación añadiremos la tarjeta de expansión que hará las veces de periférico. Para hacerlo, nos situaremos sobre el árbol de estructura del proyecto y utilizaremos la opción 'Añadir un periférico' del menú 'Proyecto'. Como en el proyecto anterior hicimos esta misma operación desde el menú contextual del árbol de estructura del proyecto, el lector practica otro de los caminos posibles.



En este Proyecto, a diferencia de los anteriores, no vamos a utilizar una tarjeta de expansión Grove ni ninguno de los módulos de sensores disponibles para ella. Esta vez vamos a seleccionar una tarjeta de expansión que contiene dentro tanto el display TFT como el habitáculo para alojar dentro de él la tarjeta de memoria SD. En concreto el módulo 'Adafruit TFT Display'.



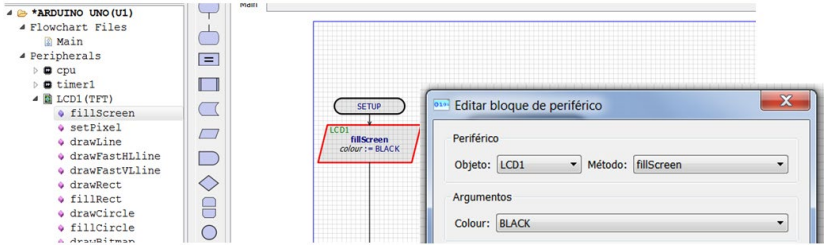
Para este proyecto es necesario también añadir un recurso. En nuestro caso se trata del fichero dónde almacenamos la información de la imagen que deseamos almacenar en nuestra tarjeta de memoria SD y mostrar en el display TFT. Para hacerlo, nos situaremos sobre el árbol de estructura del proyecto y utilizaremos la opción del menú contextual 'Añadir un recurso'.



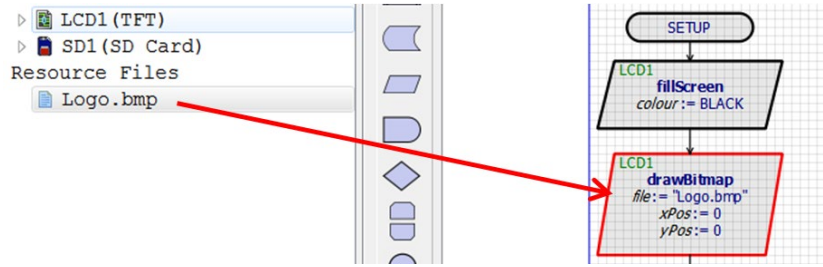
⚠ Debemos ser cuidadosos con el uso de los nombres de los recursos de fichero por una peculiaridad propia de la gestión de las tarjetas de memoria SD que hace Arduino. Sólo soporta nombres de archivo según el estándar 8.3 que es el utilizado por las tarjetas formateadas según el estándar FAT16. Esto significa, que el nombre del archivo debe tener ocho caracteres o menos y la extensión tres o menos. Por ejemplo, si intentamos utilizar un fichero de mapa de bits llamado 'LogoProteus.bmp' nos generará un error. Si, por el contrario, utilizamos el nombre 'Logo.bmp' o 'imagen1.bmp' funcionará correctamente.

9.3.2.-Construir el flujograma.

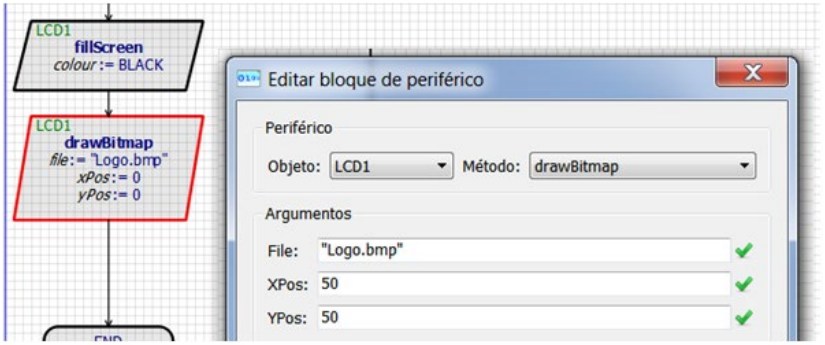
Ahora que todo el hardware ya está correctamente definido, nuestra siguiente tarea es crear el código de nuestro proyecto. Esto no podría resultar más sencillo utilizando el Diseñador Gráfico. Primero cogeremos el método fillScreen() del módulo LCD1 y lo arrastraremos a la parte superior de la subrutina loop() y configuraremos el argumento color con el valor 'Black'.



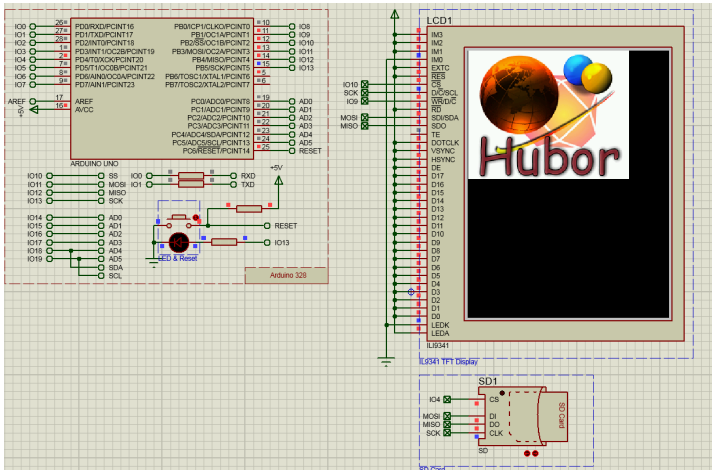
A continuación, cogeremos el recurso que contiene el fichero de imagen de bits y lo arrastraremos dentro de la subrutina loop(). Visual Designar detecta que el único destino posible para este tipo de recurso es el display TFT y sabe que el método necesario para manejar este recurso es drawBitmap() por lo que se encargará de configurar el flujograma de forma automática por nosotros.



Ahora debemos configurar las coordenadas de la pantalla dónde se mostrará la imagen seleccionando como valores de xPos e yPos el valor 0. Proteus ha definido automáticamente por nosotros las coordenadas x e y ambas con el valor 0. Pero añadimos este punto aquí, por si el lector desea utilizar otras coordenadas de origen diferentes. En el ejemplo hemos introducido los valores 50 para ambas coordenadas.

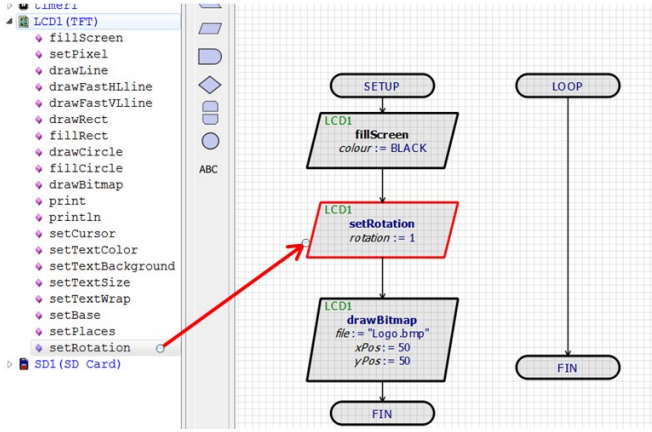


Es el momento de utilizar el botón PLAY del cuadro de control de la simulación para ponerla en marcha y ver el resultado de nuestro proyecto.

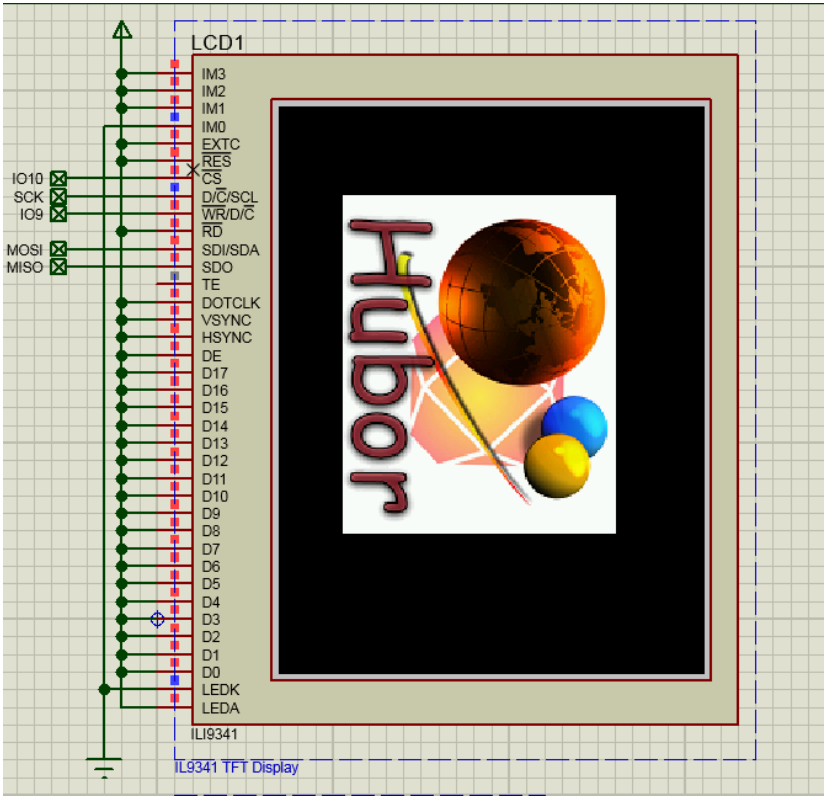


Un resultado, francamente espectacular utilizando sólo dos bloques en nuestro flujograma.

Si lo deseamos podemos también utilizar el método ‘SetRotation’ para girar la imagen.



El resultado obtenido se muestra en la imagen siguiente:

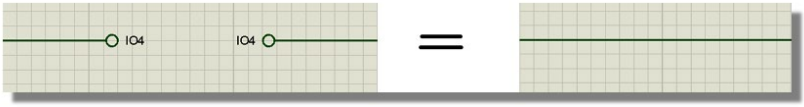


9.3.3.-Utilizar un analizador de bus de datos virtual.

Una de las características más importantes del Diseñador Gráfico es que funciona en perfecta armonía con las capacidades de Proteus para gestionar los esquemas electrónicos, incluidos todos los instrumentos virtuales que pone a nuestra disposición. Esto significa, que si lo deseamos, podemos ver con todo detalle lo que está ocurriendo durante la ejecución de nuestro proyecto. Esto es especialmente útil cuando tenemos que depurar nuestro diseño o nuestro código en busca de errores.

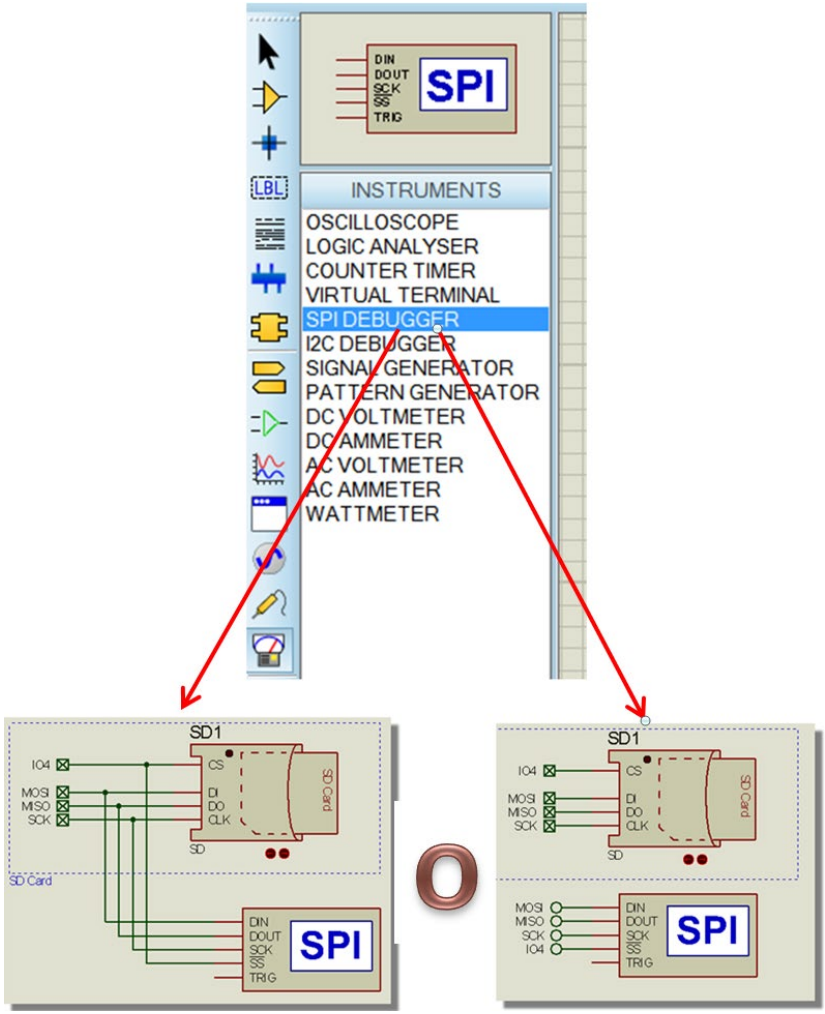
Por ejemplo, en este proyecto la tarjeta de memoria SD se conecta con nuestro procesador Arduino utilizando el bus SPI. Por lo tanto, podemos conectar el analizador de bus de datos al bus serie y examinar con detalle los paquetes que navegan por el bus.

Recordemos aquí, que en Proteus todo tipo de señales y ondas viajan a través de los cables (o pistas) y que todos los terminales que tienen el mismo nombre están unidos por un cable invisible entre ellos.

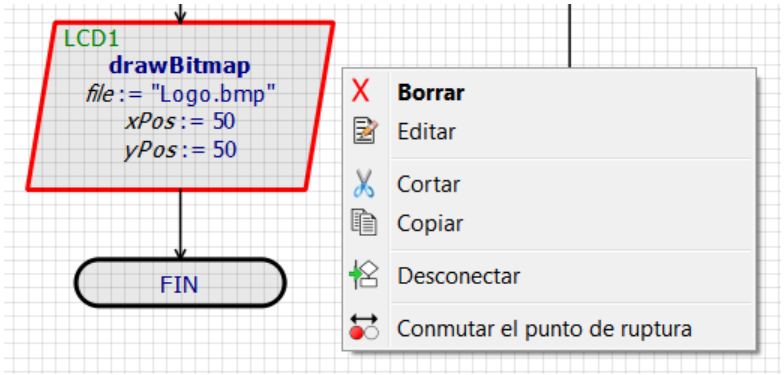


Por lo tanto, y aprovechando estas ventajas que nos ofrece Proteus, la forma más fácil de conectar nuestro analizador del bus SPI es colocarlo en nuestro esquema electrónico y conectarlo a terminales que tengan el mismo nombre que las líneas del bus SPI.

Alternativamente, se puede también cablearlo directamente a las líneas del bus SPI. El lector puede escoger el método que más le convenga.



Para que el uso del analizador nos resulte más cómodo, vamos a fijar un punto de ruptura dentro del bloque del comando drawBitmap(), que es el lugar dónde se van a ejecutar la mayor parte de las transmisiones a través del bus SPI.



Si ejecutamos de nuevo la simulación veremos que se ejecuta toda la carga del fichero con la imagen utilizando el bus SPI hasta que se produce el punto de ruptura. Si pulsamos el icono de la barra de herramientas para generar un salto en nuestro código podremos ver cómo una gran cantidad de datos se mueve desde la tarjeta de memoria SD hasta el procesador. Este proceso puede llevar un tiempo largo en función del tamaño del fichero de imagen que hayamos utilizado y la capacidad de nuestro ordenador personal para llevar a cabo la simulación. Hay que tener en cuenta que esta simulación es bastante compleja y consume muchos recursos. Podemos utilizar el analizador del bus de datos para ver con el detalle que deseemos las transmisiones que se han llevado a cabo sobre el bus.

Podemos también utilizar cualquier otro de los instrumentos virtuales que nos ofrece Proteus: osciloscopio, analizador del bus i2c, analizador lógico, voltímetro, etc.

Utilizar instrumentos virtuales puede resultar esencial para analizar nuestro proyecto, pero provocará una carga inevitable de trabajo para el microprocesador de nuestro ordenador personal y la velocidad de simulación puede resentirse. Es muy importante eliminar los instrumentos virtuales de nuestro diseño que no vayan a ser utilizados durante la sesión de depuración.

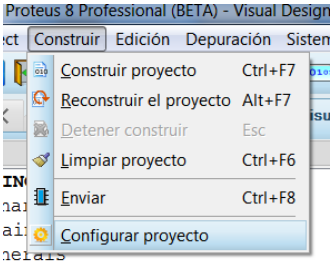
9.3.4.-Cargar el proyecto en un equipo arduino real.

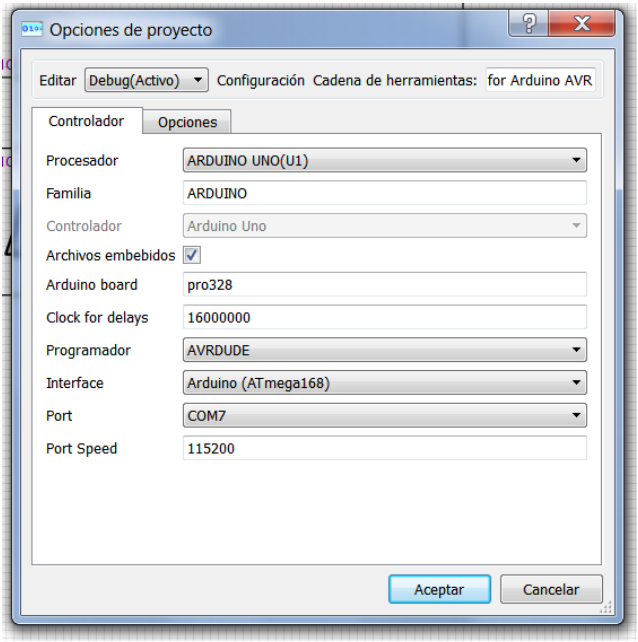
Una vez que hemos comprobado que nuestro proyecto funciona correctamente según lo previsto, ya estamos en condiciones de enchufar nuestro equipo Arduino y cargar el programa sobre el hardware real.

Conectaremos la tarjeta de expansión Adafruit TFT a la placa base del Arduino. Nos aseguraremos de que hemos introducido una tarjeta de memoria SD compatible. Debe ser formateada como FAT16 o FAT32 y ser de un tamaño suficiente para alojar nuestro fichero de imagen.

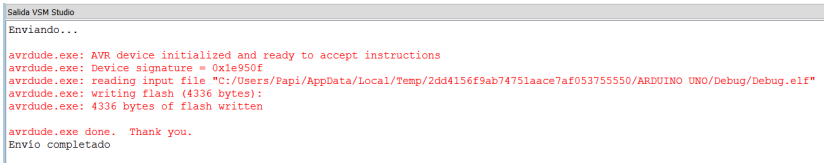
Conectaremos la placa Arduino al puerto USB de nuestro ordenador personal dónde tenemos instalado Proteus.

Utilizaremos la opción de menú 'Configurar proyecto' del menú 'Construir' para configurar el puerto dónde hemos conectado nuestro equipo Arduino.





En la ventana de estado, podemos comprobar que todo se ha producido conforme a lo esperado y que el programa se ha cargado correctamente en nuestro equipo Arduino.

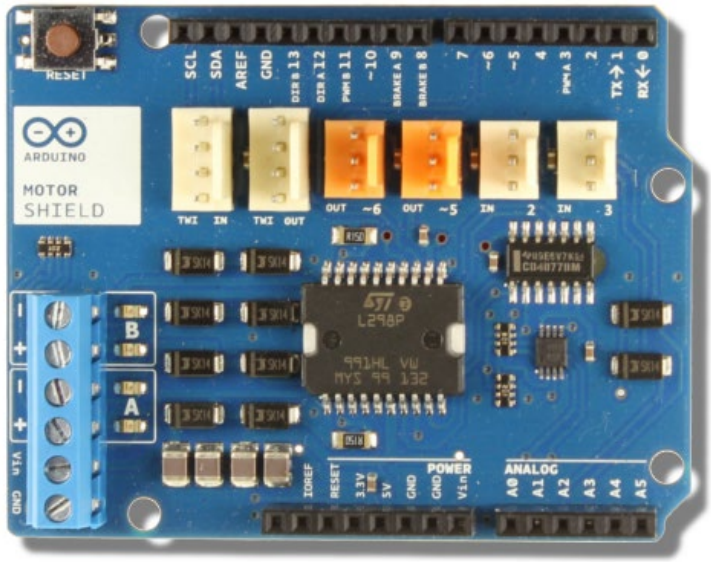


Si el proceso falla conviene comprobar que hemos seleccionado correctamente el puerto y la velocidad para la que esté configurado nuestro Arduino. Seguir las instrucciones del punto 7.2 en lo referente a la selección de la velocidad del puerto serie.

9.4.-Control de un motor.

En este cuarto tutorial vamos a ver un proyecto en el que controlaremos, en primer lugar, un motor DC y, en segundo lugar,

un motor paso a paso utilizando el Diseñador Gráfico. Como hardware vamos a utilizar la tarjeta de expansión para el control de un motor de Arduino V1 R3 (https://www.arduino.cc/en/Main/ArduinoMotorShieldR3). Este proyecto también puede llevarse a cabo utilizando ul módulo Grove Servo o utilizando el modelo de Proteus MOTOR-BLDCM directamente en el diagrama electrónico, con sólo realizar unas sencillas modificaciones.

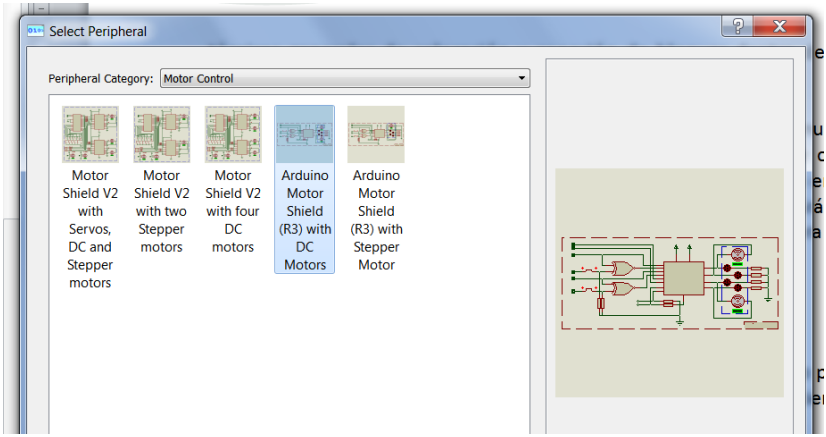


En esta ocasión asumiremos que el lector ya ha trabajado los primeros tutoriales y que está suficientemente familiarizado con las técnicas generales de colocación y conexión de bloques dentro de un flujograma.

La utilización del asistente para la creación de un proyecto nuevo se explicó en el primer tutorial (apartado 8.1.1), así que en este caso asumiremos que el lector ya ha creado un nuevo proyecto basado en la placa Arduino UNO, seleccionando como compilador el Diseñador Gráfico y que la estructura por defecto ya está lista para ser utilizada.

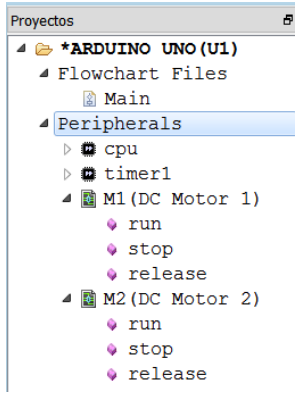
9.4.1.-Control de un motor dc.

Empezaremos el proyecto añadiendo la tarjeta de expansión de Arduino para controlar motores DC de la misma manera que lo hemos venido haciendo en los tutoriales anteriores.



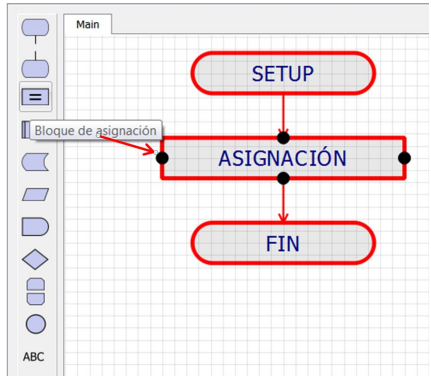
Existen dos tarjetas de expansión disponibles para controlar motores de la versión R3. Los dos son muy similares pero con una diferencia muy importante. Uno está preparado para controlar/simular dos motores DC y el segundo sólo controla/simula un motor paso a paso. En el mundo real, es necesario que nosotros realicemos las conexiones de acuerdo con el proyecto que estamos desarrollando.

Al elegir la tarjeta de expansión mencionada, tenemos que ver en el árbol de estructura del proyecto dos motores con todos los métodos asociados disponibles.

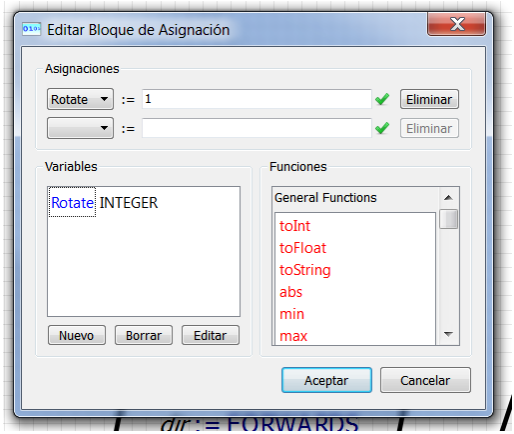


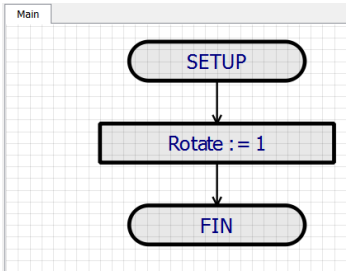
A continuación vamos a escribir un sencillo programa que ponga en movimiento los dos motores en sentidos opuestos. Lo primero que tenemos que hacer es crear una variable que contendrá el valor de la velocidad con que deseamos que giren los motores.

Así que vamos a coger y arrastrar un bloque de asignación dentro de la subrutina Setup(). Sólo es necesario que lo hagamos una vez porque utilizaremos la misma variable para los dos motores.

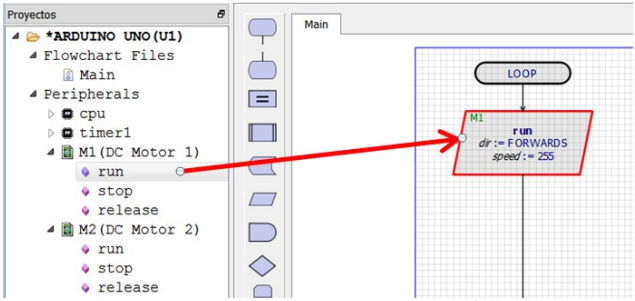


Editaremos el bloque, crearemos la variable de nombre ‘Rotate’, del tipo ‘integer’ y le asignamos el valor 1.



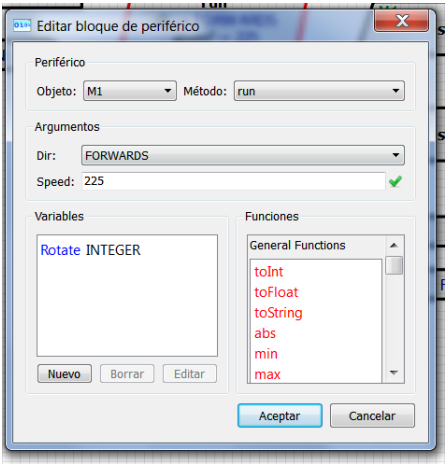


Ahora vamos a introducir el código para controlar el motor que girará en sentido de las agujas del reloj (forwards). Cogeremos y arrastremos el método 'run' del primer motor dentro de la subrutina loop().

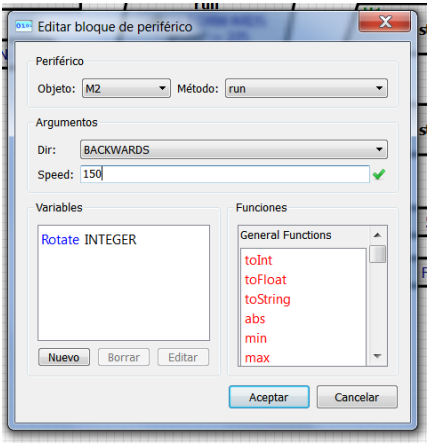


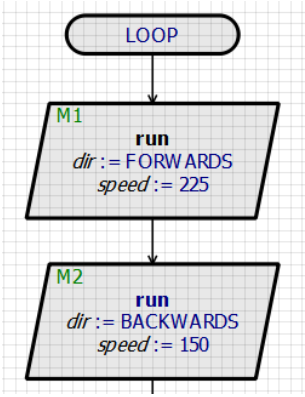
Editamos el bloque recién pegado y seleccionamos la dirección de rotación (forwards) y la velocidad de giro (255). Por defecto, Proteus

ha creado el bloque con estos valores, pero lo mencionamos para que el lector comprenda cómo se puede configurar el bloque.

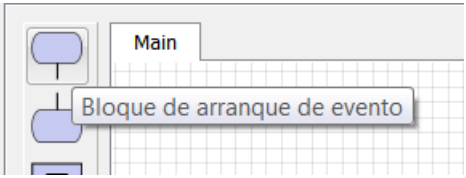


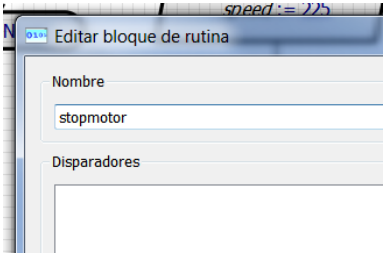
Repetimos el proceso con el segundo motor, pero seleccionando la dirección de giro contraria (backwards) y una velocidad de giro diferente (150).

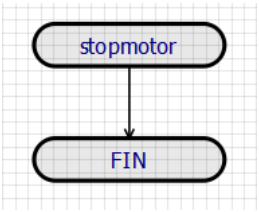




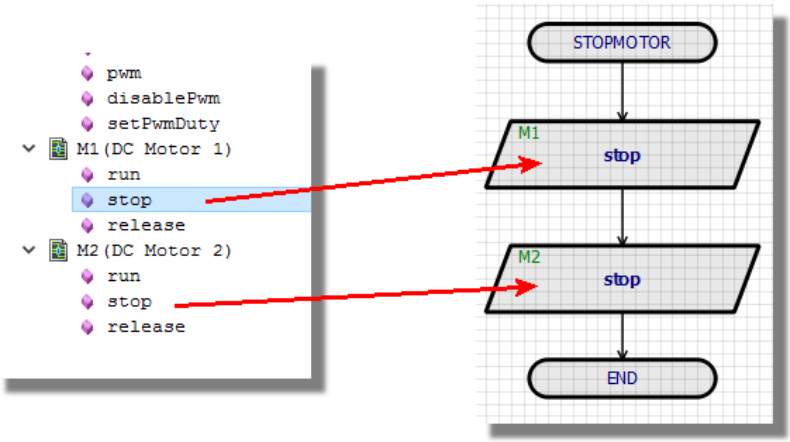
Ahora, vamos a añadir a nuestro código una subrutina nueva que vamos a utilizar para detener los dos motores. Cogemos un bloque de arranque de evento o subrutina y lo arrastramos en cualquier área libre de nuestra ventana de trabajo. Una vez colocado, debemos editarlo y asignarle el nombre 'stopmotor'.



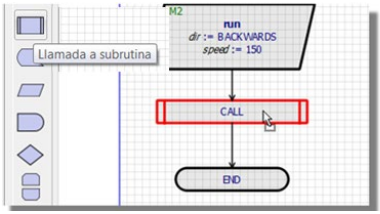


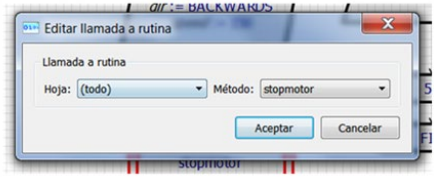


Una vez creada la subrutina, cogemos para cada uno de los dos motores el método ‘stop’ y lo arrastramos dentro de la subrutina.



Y, para acabar, cogemos un bloque de llamada a subrutina y lo arrastramos dentro de la subrutina loop(). Lo editamos y le asignamos como nombre de la subrutina llamada ‘stopmotor’.

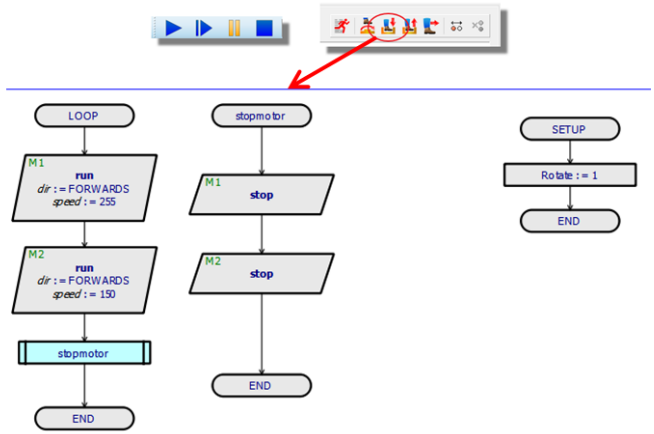


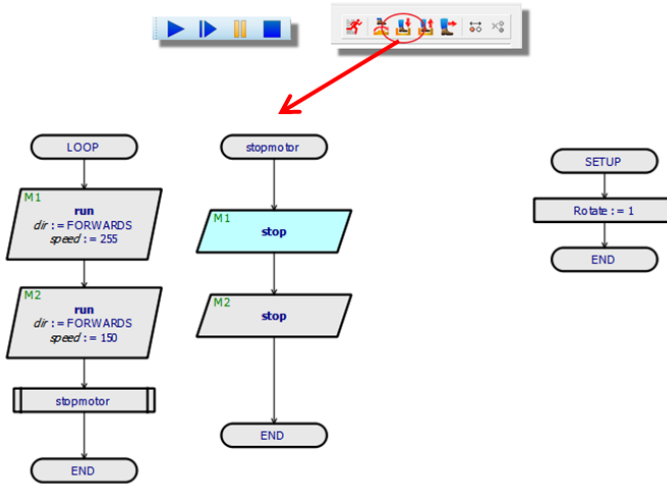


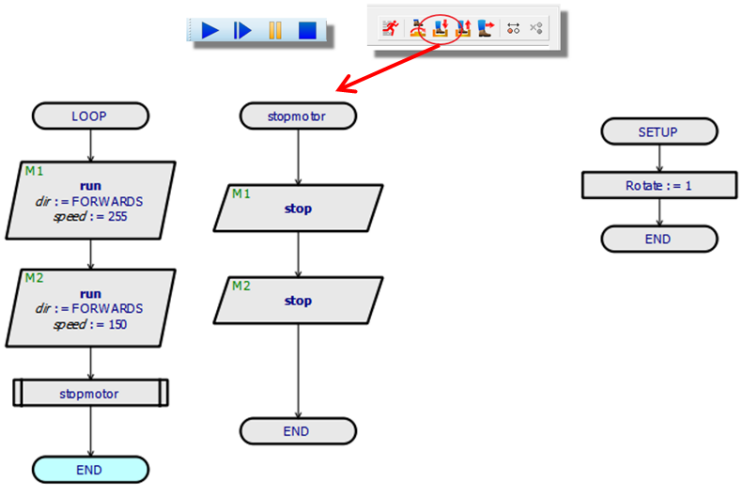
Para comprobar el funcionamiento del código vamos a poner la simulación en marcha utilizando el botón de PAUSA del cuadro de control de la simulación. No confundirse y no usar el botón PLAY. El proyecto se compilará y la simulación se pondrá en marcha.

Si utilizamos el botón ‘STEP INTO’ de la barra de herramientas de depuración de forma repetida, podemos comprobar cómo se va realizando paso a paso la ejecución del código. Los bloques van cambiando de color a medida que se ejecuta la simulación.

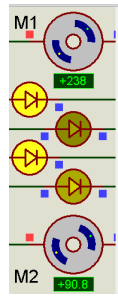
Ejecutaremos sucesivos pasos hasta ver cómo en un momento dado se ejecuta la subrutina ‘stopmotor’ y cómo más tarde se vuelve a la rutina principal loop().


LOOP
M1
run
dir : = FORWARDS
speed : = 255
M2
run
dir : = FORWARDS
speed : = 150
stopmotor
END
stopmotor
M1
stop
M2
stop
END
SETUP
Rotate : = 1
END


LOOP
M1
run
dir : = FORWARDS
speed : = 255
M2
run
dir : = FORWARDS
speed : = 150
stopmotor
END
stopmotor
M1
stop
M2
stop
END
SETUP
Rotate : = 1
END

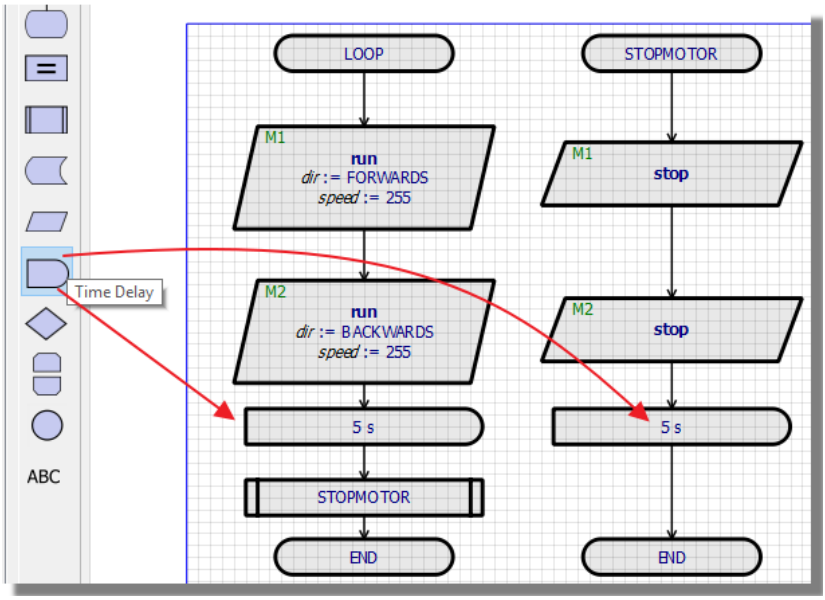


Si ejecutamos la simulación normal, utilizando el botón PLAY, podemos comprobar que nuestros motores se ponen en marcha con sentidos de rotación opuestos.

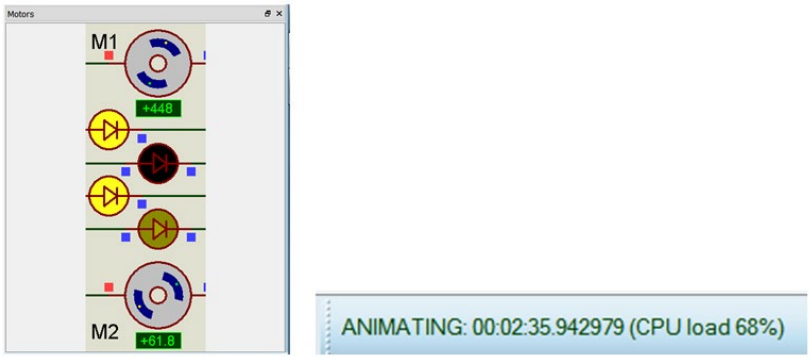


Todas las piezas básicas de nuestro proyecto están en su sitio, pero tal y como hemos escrito el código el arranque y la parada de los motores se produce muy rápido y, por esta razón, no les va a permitir coger el impulso necesario.

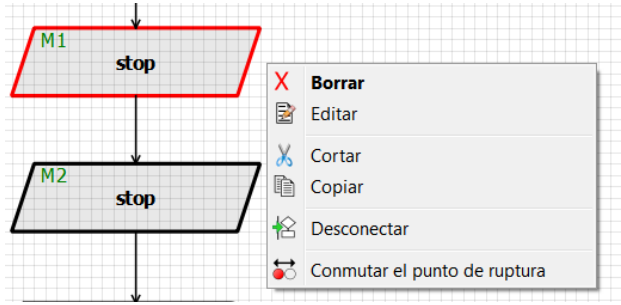
La mejor solución es añadir un bloque de aplazamiento temporal tanto en el momento en que generamos la marcha como en el momento en que ordenamos parar. Así que cojamos bloques del tipo aplazamiento temporal y arrastrémoslos dentro de nuestro flujograma. El valor del aplazamiento temporal debe ser de 5 segundos.



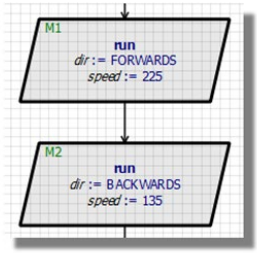
Volvamos a ejecutar la simulación y comprobemos el resultado viendo girar los motores y controlando el tiempo de ejecución en la barra de estado.

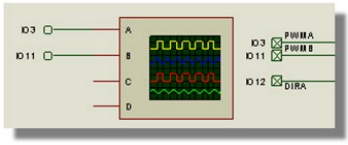


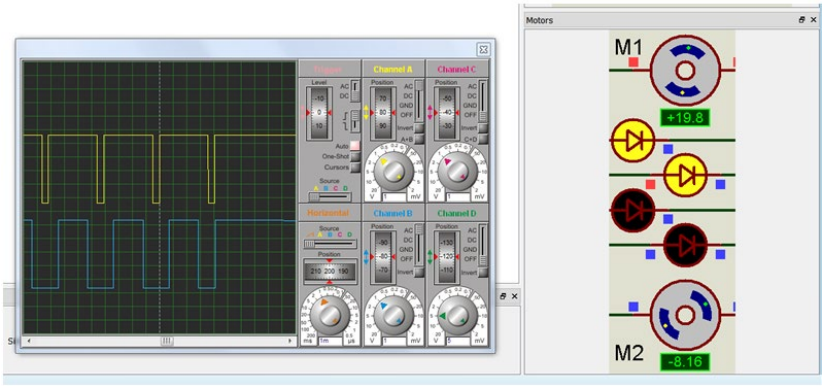
Podemos probar a poner la simulación en pausa y fijar un punto de ruptura en uno de los bloques que ordena la parada de un motor. A diferencia del mundo real, en la simulación los motores no pierden su par motor cuando se alcanza el punto de ruptura lo que significa una importante ventaja a la hora de realizar la depuración del software.



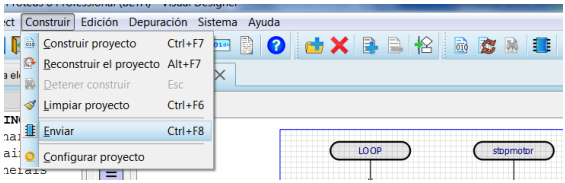
Para lograr un análisis todavía más detallado del funcionamiento de nuestro proyecto podemos colocar y cablear el osciloscopio virtual en nuestro esquema electrónico y visualizar la señal de ondas tipo PWM que controlan nuestro motor. La captura de pantalla se corresponde con valores introducidos de velocidades de 225 y 135 para cada uno de los motores.



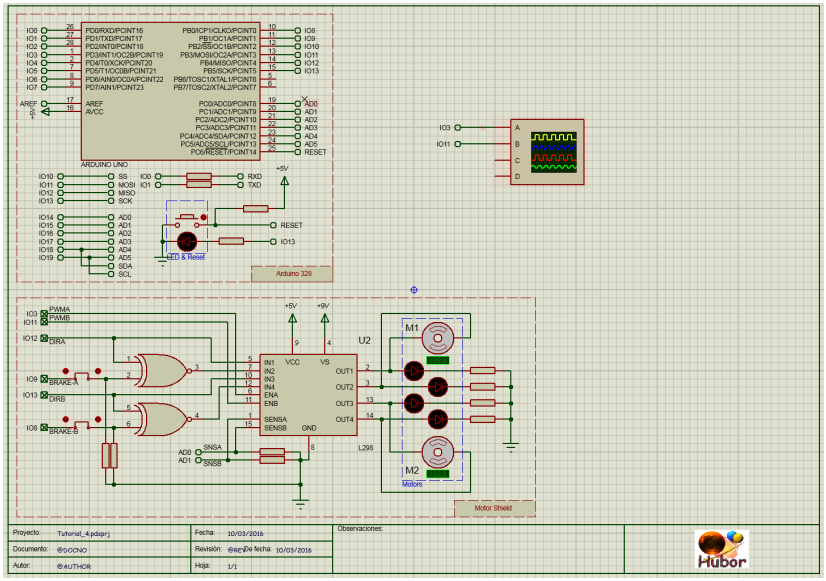


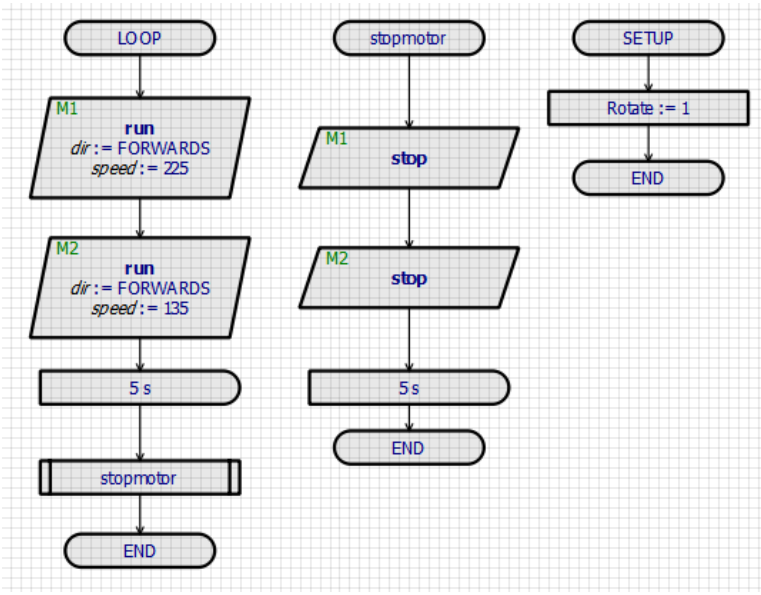


Como vimos en los proyectos anteriores, una vez que todo funciona según nuestros deseos podemos descargar el código en un equipo Arduino del mundo real utilizando el menú Construir, el botón correspondiente de la barra de herramientas o el atajo de teclado CTRL-F8.



El diagrama electrónico y el flujograma completos se muestran a continuación.

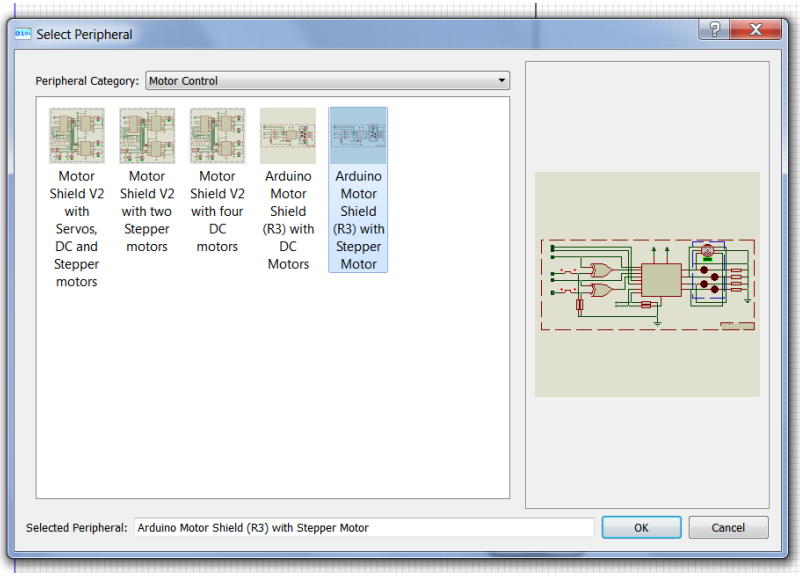




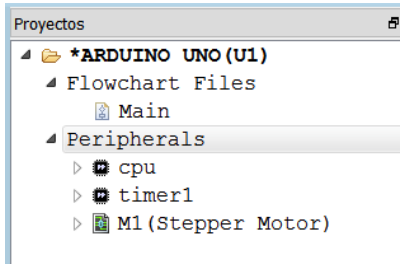
9.4.2.-Control de un motor paso a paso

Si en lugar de un motor DC deseamos controlar un motor paso a paso la tarjeta de expansión que debemos utilizar como hardware es el mismo (Arduino Motor Shield V1 R3) pero configurado para controlar un único motor paso a paso unipolar.

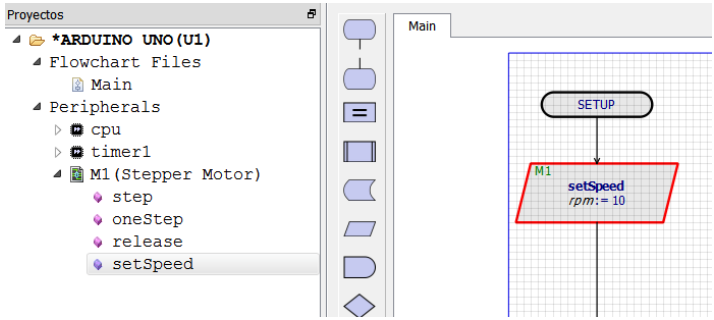
Creamos un proyecto siguiendo el mismo procedimiento que acabamos de seguir al principio del apartado 8.4.1., pero seleccionando como periférico el módulo de 'Arduino Motor Shield (R3) with Stepper Motor' dentro del apartado 'Motor Control'.



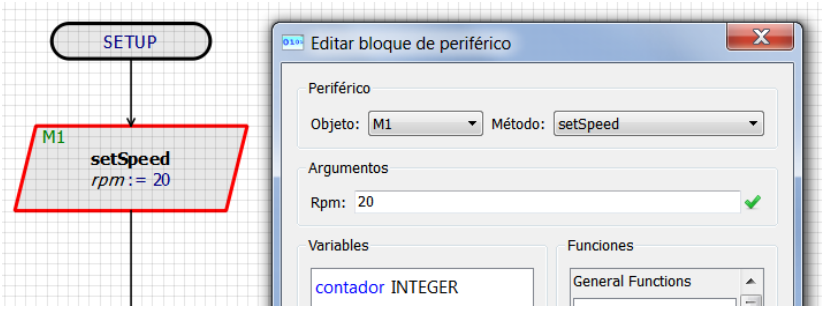
Observe que en este caso en el árbol de estructura del proyecto sólo se crea un motor y no dos como en el caso anterior.



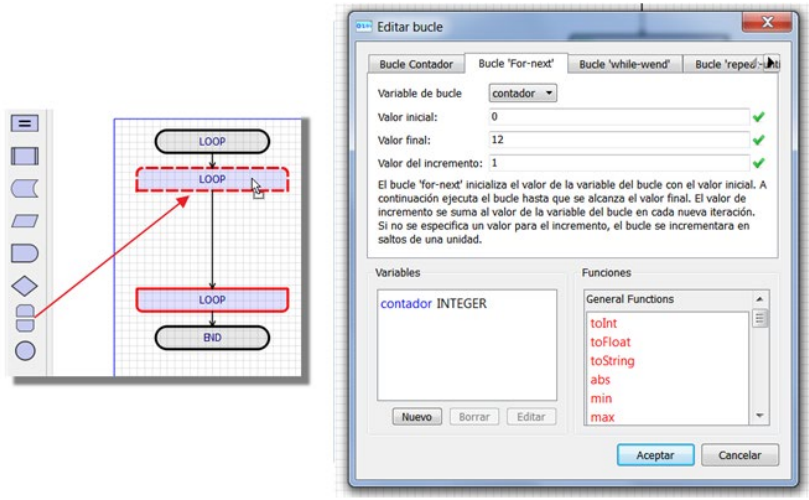
Crearemos un código con un pequeño programa que moverá el motor primero en un sentido y luego en el contrario. Nuestra primera tarea es definir la velocidad con la que rotará el motor. Para hacerlo cogeremos el método setSpeed() del periférico motor y lo arrastraremos dentro de la subrutina setup().



Editaremos el bloque recién colocado y configuraremos la velocidad de rotación utilizando el argumento 'Rpm'. Le asignaremos el valor de 20.

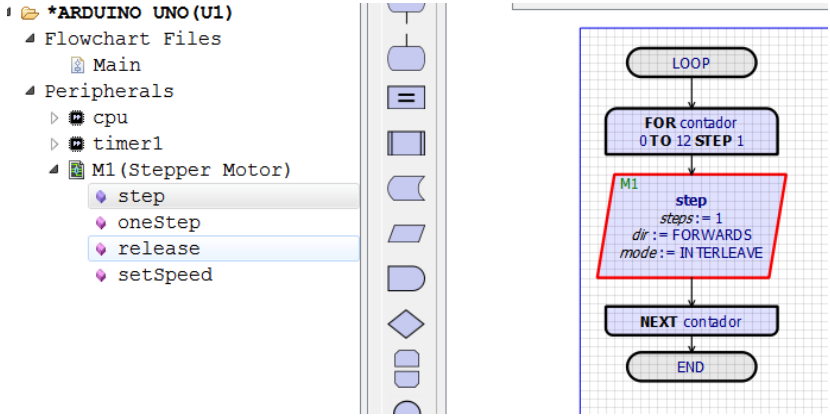


Para generar el número de pasos que queremos que se mueva nuestro motor, utilizaremos un bloque de bucle. Cogemos un bloque de este tipo y lo arrastramos hasta dentro de la rutina principal loop(). Una vez colocado lo editamos.

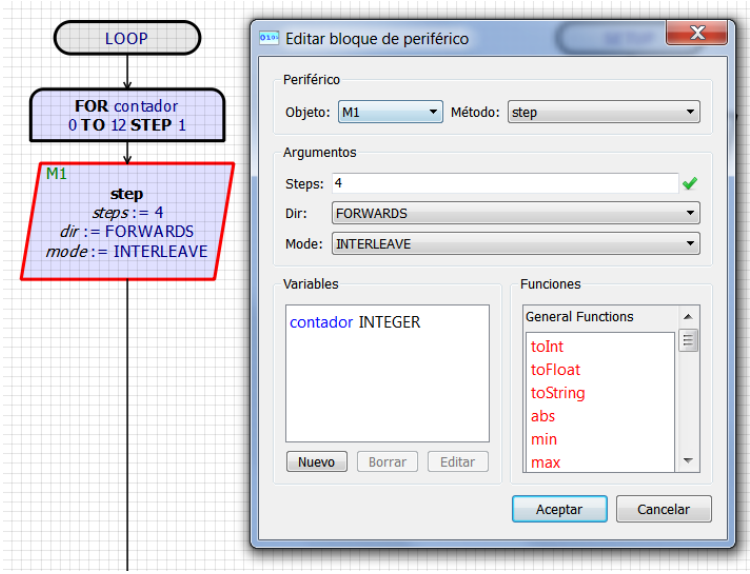


Seleccionamos la pestaña del bucle tipo Bucle 'for-next'. Añadimos una variable del tipo 'integer' a la que le daremos el nombre de 'contador'. Definimos 0 como valor inicial, 12 como valor final y 1 como valor de incremento.

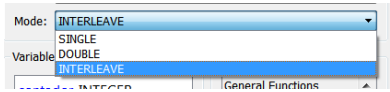
A continuación, cogemos el método step() del periférico motor y lo arrastramos dentro de la estructura de bucle que acabamos de crear. Si todo ha ido bien tenemos que ver que los tres bloques que forman ahora el bucle están de color azul. Es la manera que usa Proteus para indicarnos que el bloque que acabamos de posicionar forma parte del bucle.



Editaremos el bloque para asignar el valor 4 al parámetro ‘Steps’, ‘forwards’ al parámetro ‘dir’ y ‘interleave’ al parámetro ‘mode’.



El parámetro ‘steps’ controla el número de pasos. El parámetro ‘dir’ indica el sentido de giro. El parámetro ‘mode’ puede recibir tres valores.

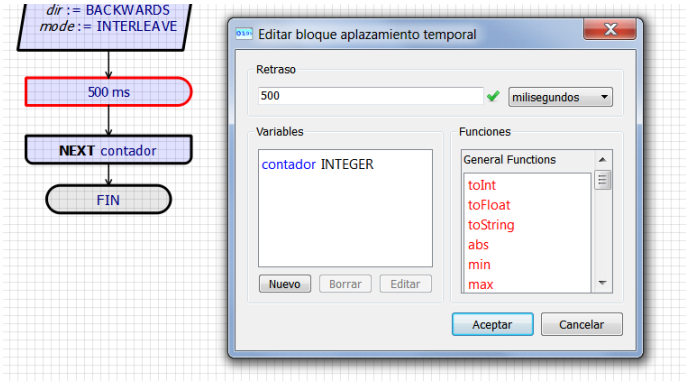


Paso simple (Single) indica que el motor se activa mediante una única bobina. Paso doble (Double) que se activa utilizando dos bobinas con lo que se consigue aumentar el par motor. Y medio paso (interleave) significa que se usa alternativamente tanto el modo simple como el doble para mejorar la precisión del movimiento al doble, aunque reduciendo la velocidad a la mitad.

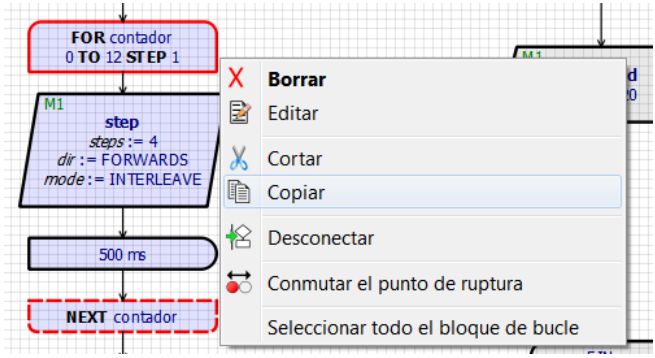
En Internet existe multitud de información sobre la utilización de los motores paso a paso. Por ejemplo, es muy didáctica y con muy buenos gráficos explicativos la descripción del siguiente documento:

http://server-die.alc.upv.es/asignaturas/lsed/2002-3/MotoresPasoaPaso/ftomotpap.htm

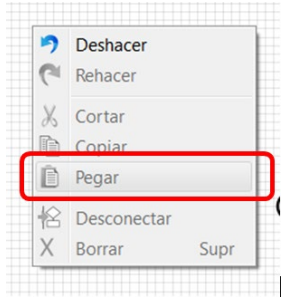
Añadiremos un bloque de aplazamiento temporal después del método step() que acabamos de colocar en nuestro flujograma para permitir el ajuste completo de fase antes de reiniciar un nuevo ciclo del bucle.



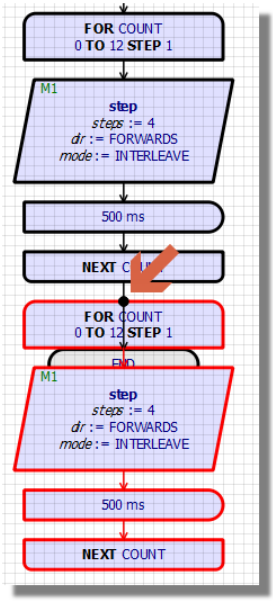
Hasta aquí hemos escrito el código necesario para que el motor se mueva en un sentido de giro. Para conseguir que el motor gire en sentido opuesto, el proceso es idéntico pero cambiando la dirección en el método step(). Podemos volver a coger y arrastrar unos bloques idénticos a los que acabamos de usar para crear un segundo bucle o, mejor aún, podemos utilizar la técnica copiar y pegar. Seleccionamos todos los bloques que deseamos copiar y utilizamos la opción copiar del menú contextual.



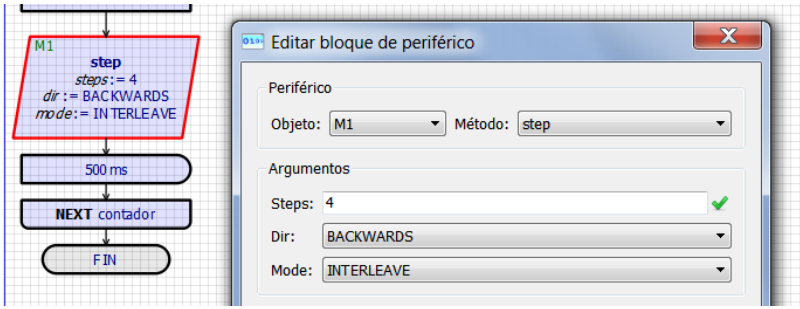
A continuación, usamos el botón derecho del ratón para mostrar el menú contextual y seleccionamos la opción pegar (o el atajo de teclado CTRL-V).



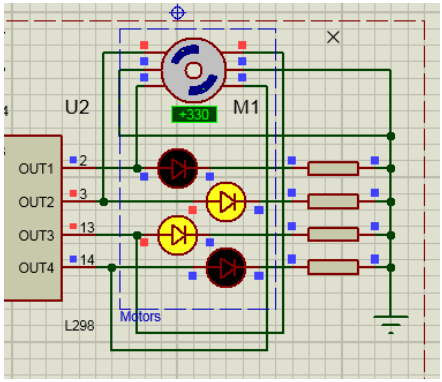
Y arrastramos todo el nuevo bloque al lugar que deseemos de nuestro flujograma.

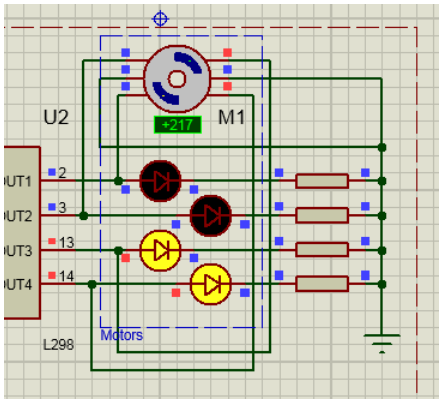


No debemos olvidar cambiar el método step() del segundo bucle para seleccionar el sentido de giro opuesto al del primer bluce.

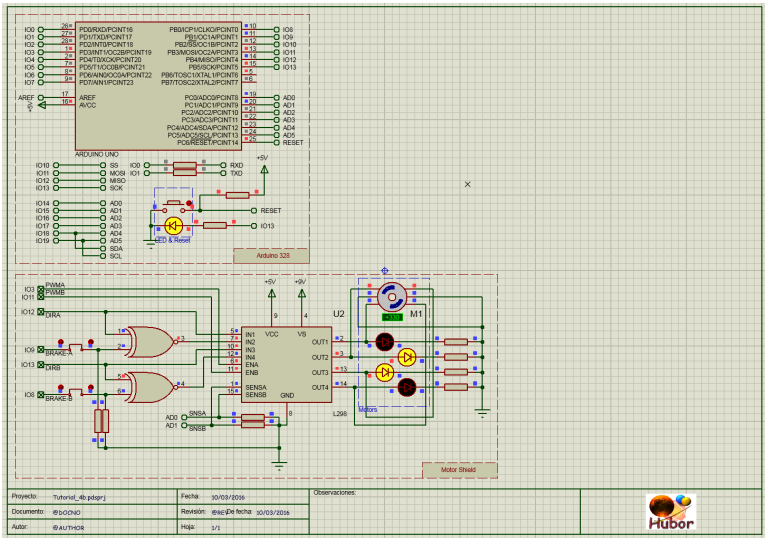


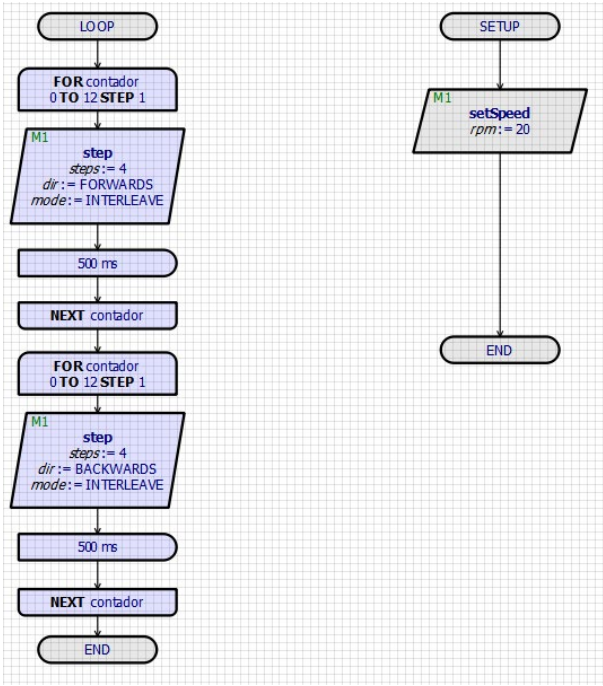
Ya estamos en condiciones de arrancar la simulación utilizando la tecla PLAY del cuadro de control de la simulación y disfrutar del movimiento de nuestro motor paso a paso en ambos sentidos de giro.





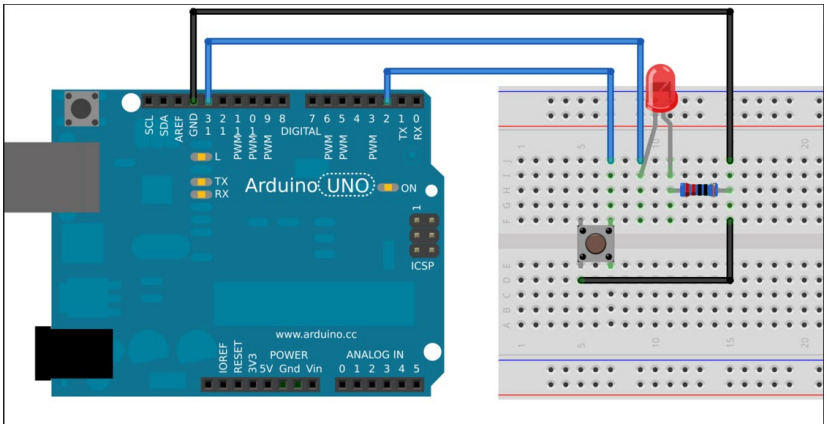
El diagrama electrónico y el flujograma completos se muestran a continuación.





9.5.-Simulación de una placa de pruebas (protoboard).

En los tutoriales de ejemplo que hemos visto hasta ahora siempre hemos utilizado alguna tarjeta de expansión existente en el mercado para Arduino o módulos del tipo Grove como hardware. Pero nada nos impide utilizar diseños dónde nosotros construyamos nuestra propia tarjeta de expansión dibujando directamente el esquema electrónico que deseemos. Es lo que hemos llamado simular el uso de una placa de pruebas (protoboard).



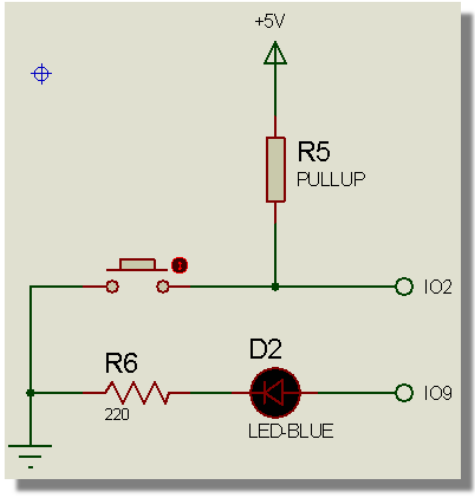
En este tutorial queremos mostrar sólo un ejemplo muy básico de esta forma de trabajar utilizando un pulsador y un led. El lector que siga este tutorial comprenderá inmediatamente que usando esta técnica podemos construir multitud de diferentes tarjetas de expansión con cualquiera de los miles de componentes diferentes que pueden ser simulados utilizando las librerías suministradas por Proteus.

La utilización del asistente para la creación de un proyecto nuevo se explicó en el primer tutorial (apartado 8.1.1), así que en este caso asumiremos que el lector ya ha creado un nuevo proyecto basado en la placa Arduino UNO, seleccionando como compilador el Diseñador Gráfico y que la estructura por defecto ya está lista para ser utilizada.

9.5.1.-Diseñar el esquema electrónico.

Proteus facilita mucha documentación sobre las herramientas disponibles para dibujar el esquema electrónico de nuestro proyecto. En el apartado 1.1 de esta misma guía las mencionamos. Por ese motivo, en este punto aceptaremos que el lector conoce perfectamente todas estas técnicas y sólo describiremos muy brevemente los pasos que debemos seguir para llevar a cabo nuestro proyecto.

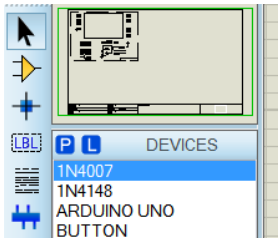
Nuestra tarea en este momento es dibujar el siguiente esquema electrónico.



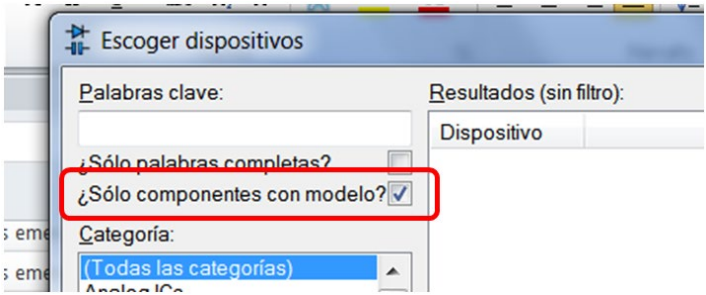
Los tres pasos necesarios son seleccionar el componente, colocarlo en el esquema y cablearlo.

9.5.1.1.-Seleccionar el componente.

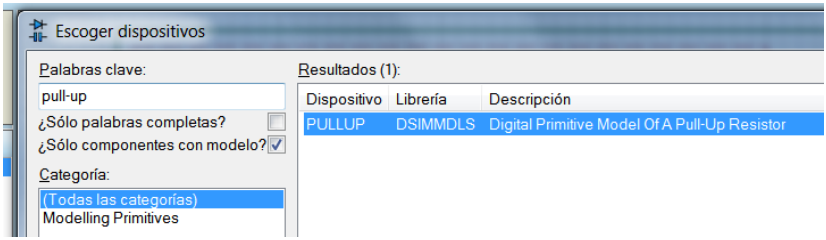
Para seleccionar un componente desde las librerías que nos ofrece Proteus, debemos utilizar el botón con una P situado en la zona superior izquierda de la ventana del selector de componentes, utilizando el icono situado en la caja de herramientas o haciendo uso del atajo de teclado (en este caso pulsar la tecla P).



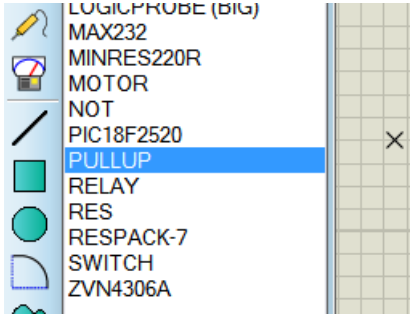
Se mostrará la ventana del explorador de librerías. Cómo sólo estamos interesados en los componentes que tienen un modelo para poder simulados, marcaremos la casilla correspondiente para filtrar los resultados.



Para este proyecto necesitamos un led, una resistencia de 220 ohms, una resistencia de 'pullup' y un pulsador.



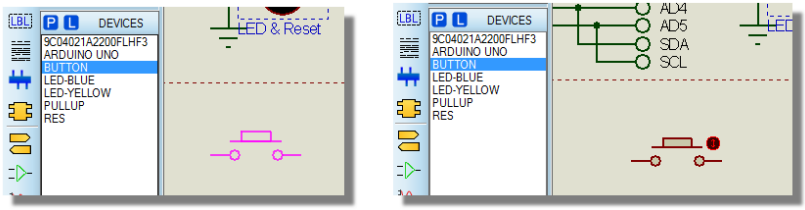
Para tener listo el componente para ser utilizado en nuestro diseño, sólo tenemos que poner el ratón sobre él y hacer una doble pulsación. Podremos ver que aparece en el selector de componentes.



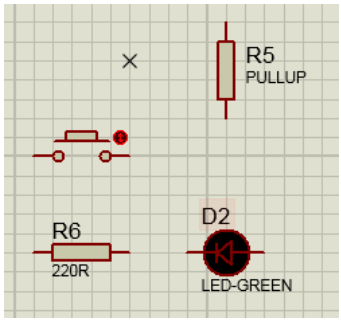
9.5.1.2.-Colocar el componente.

Para ubicar un componente en nuestro esquema electrónico, lo seleccionaremos desde el selector de componentes. Luego colocaremos el cursor del ratón en el lugar de nuestro esquema

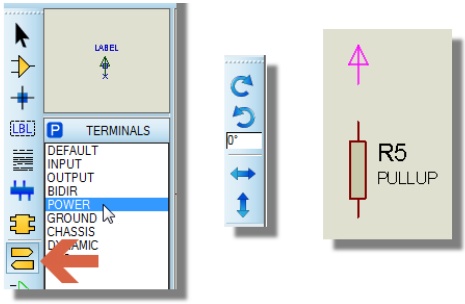
donde deseamos ubicarlo y pulsaremos el botón izquierdo para finalizar el posicionamiento. Si fuera necesario rotarlo, lo podemos hacer utilizando las teclas '+' y '-'.



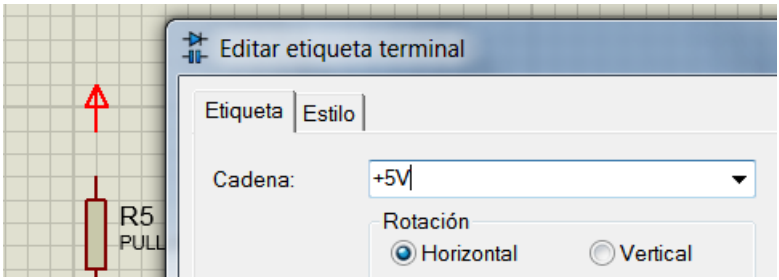
Coloquemos todos los componentes que necesitamos siguiendo el modelo propuesto.



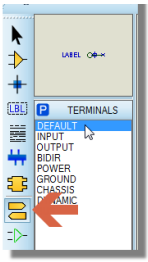
Para alimentar nuestro esquema utilizaremos los terminales del tipo 'power' y 'ground'. También podemos girarlos si fuera necesario.

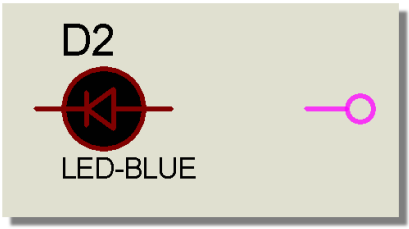


Editaremos el terminal para indicar el valor de tensión que le queremos asignar.



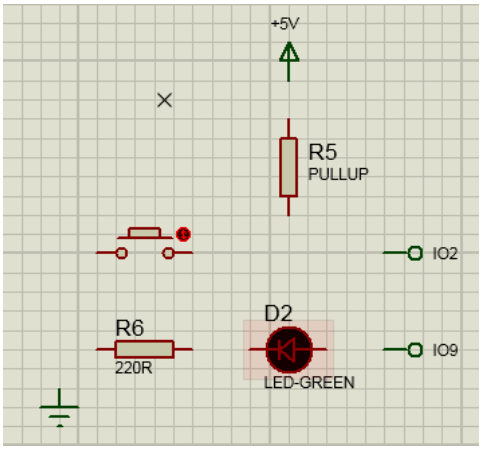
También necesitamos utilizar un terminal del tipo 'default' para conectar el pulsador y el led con los pines correspondientes de nuestro Arduino.





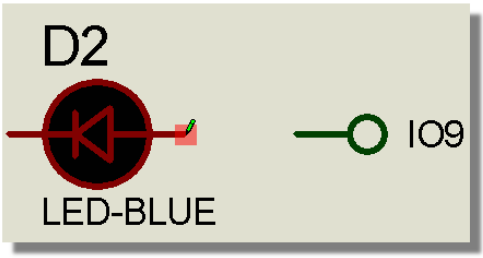
En este momento el hardware que estamos utilizando puede ayudarnos en la selección de los pines del controlador de la placa base Arduino donde conectaremos nuestros componentes. La mejor manera de comprobar si un pulsador está siendo presionado es usar uno de los pines que tiene asociada una interrupción que se activa cuando se detecta un cambio en dicho pin. Por lo tanto, lo mejor será conectar el pulsador a un pin de del Arduino de este tipo.
En el caso de las placas Arduino UNO podemos utilizar el pin IO2 para este trabajo y cualquier otro de los pines de I/O para conectar en él el led.

Al finalizar, nuestro diagrama tiene que tener un aspecto similar al que se muestra en la siguiente imagen.

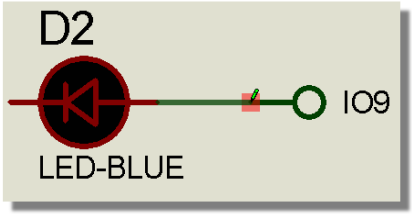


9.5.1.3.-Cablear el componente.

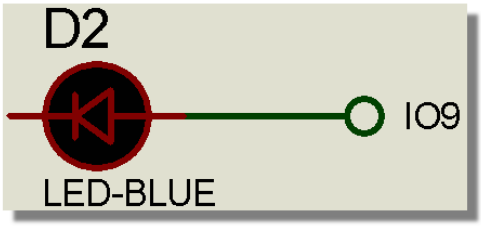
Es el momento de cablear nuestros componentes. Colocaremos el cursor del ratón sobre uno de los pines de un componente hasta que el cursor cambia (se muestra como un lápiz verde) y en ese momento pulsaremos el botón izquierdo.



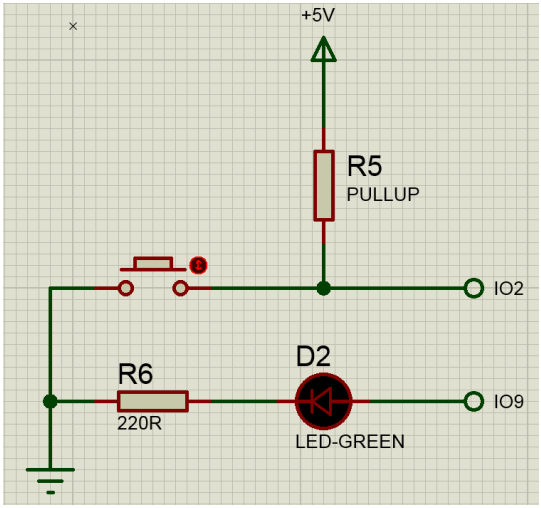
A continuación desplazamos el cursor hasta el pin del componente destino. Veremos que el cursor vuelve a tomar el aspecto de un lápiz verde.



Volvemos a pulsar el botón izquierdo y el cable quedará totalmente colocado conectando los dos componentes.



Después de conectar todo, el aspecto de nuestro diseño tiene que ser como el que se muestra en la imagen siguiente.

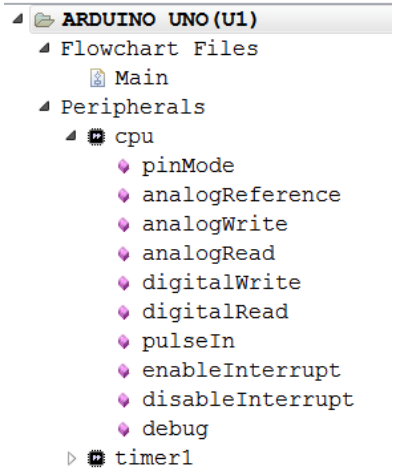


9.5.2.-Construir el flujograma.

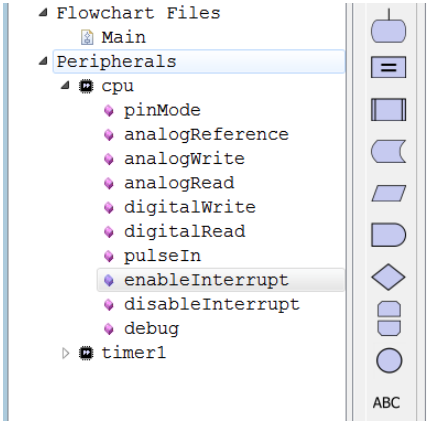
Hemos finalizado el diseño de nuestro hardware. Ahora es el momento de escribir nuestro código. Esta parte se realiza en la pestaña del Diseñador Gráfico.

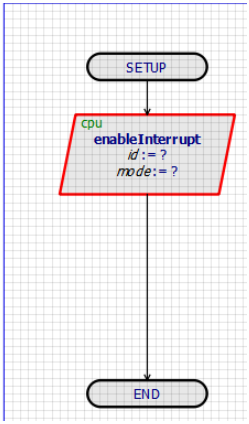
A diferencia de todos los tutoriales anteriores, en este caso no tenemos ningún método asociado a un periférico en el árbol de estructura del proyecto. Así que para gobernar los componentes

electrónicos que acabamos de crear tenemos que hacerlo utilizando métodos asociados al microprocesador (CPU).

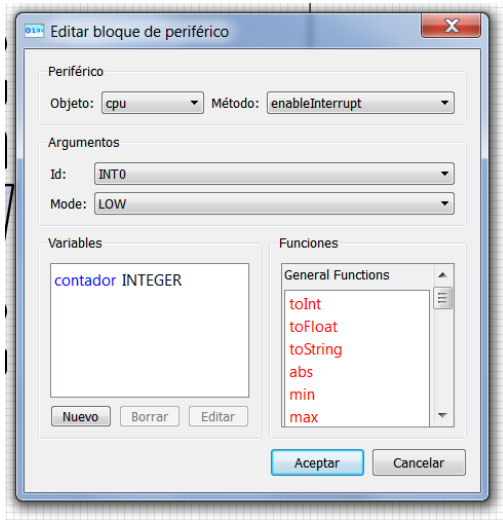


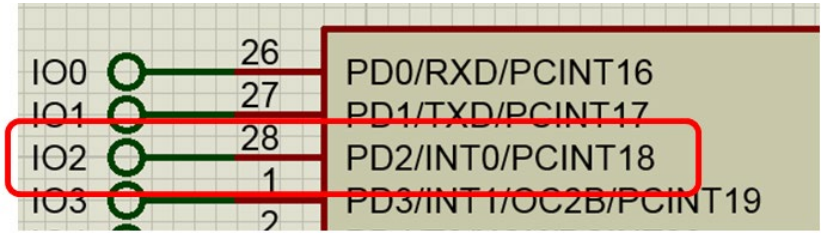
Empezaremos definiendo la interrupción asociada al pin IO2, utilizando el método 'enable interrupt' y arrastrándolo dentro de la subrutina 'setup' del flujograma.



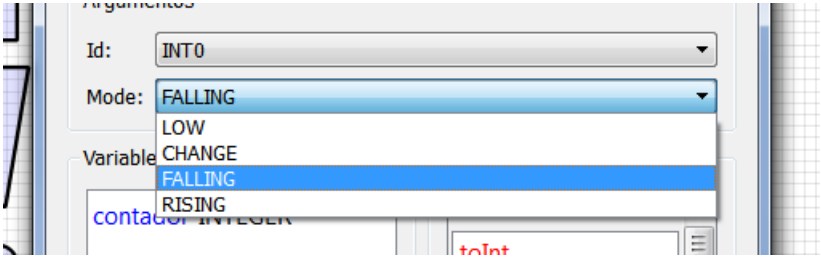


Puesto que tenemos dos interrupciones asociadas con pines en la placa Arduino UNO, necesitamos editar el bloque que acabamos de colocar y seleccionar la interrupción INT0 que es la asociada al pin IO2 dónde hemos cableado nuestro pulsador.



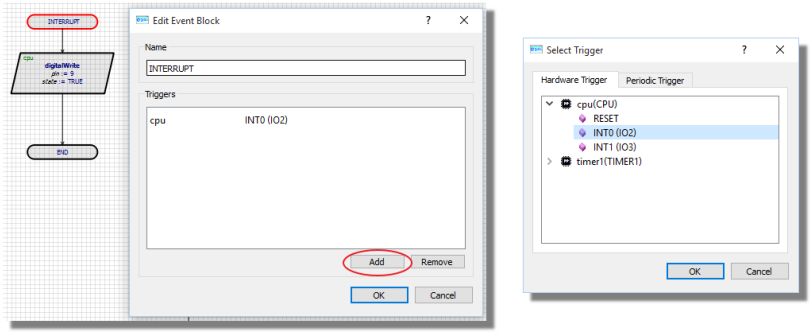


También debemos seleccionar 'falling' en la opción 'mode' porque queremos que nuestro código se active con el flanco de subida de la señal recibida desde el pulsador.

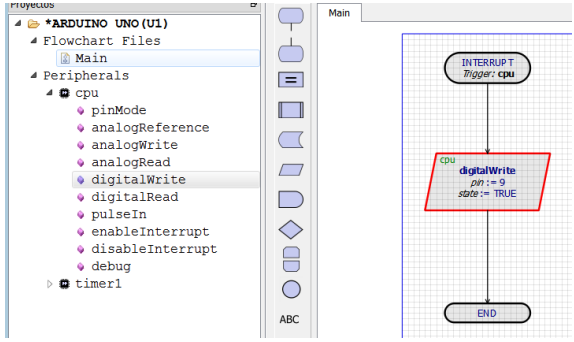


Para terminar la gestión de la interrupción, es necesario crear un evento que gestione lo que se debe hacer cuando se produzca la interrupción. Tenemos que usar un bloque del tipo arranque de

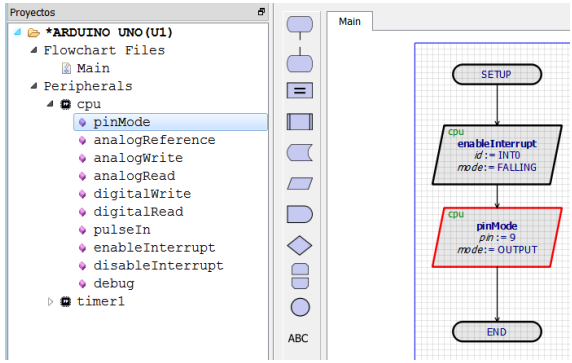
evento o subrutina y editarlo. Le tenemos que asignar un nombre que nos sirva para identificarlo y asociarlo con la interrupción que queremos gobernar (la INT0).



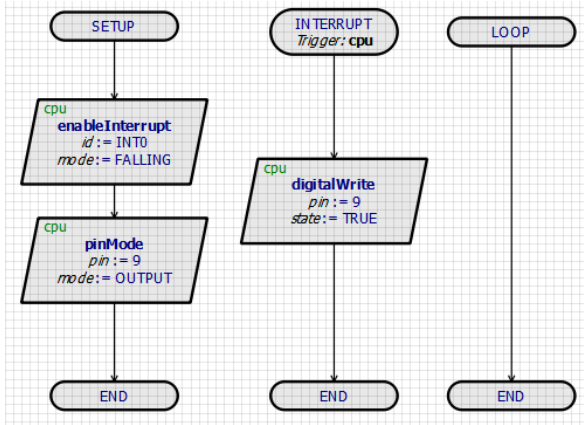
Ahora vamos a programar la orden de encendido de led dentro de la subrutina de gestión de la interrupción. Para hacerlo, necesitamos colocar a nivel alto el pin de E/S al que está conectado. Así que usaremos un bloque con el método digitalWrite() de la CPU, editándolo para seleccionar el pin 9 y seleccionando 'state' como 'TRUE' y lo arrastraremos dentro de la subrutina de gestión de la interrupción.



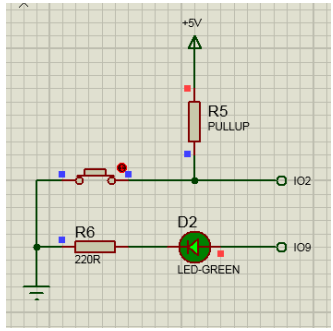
Sólo nos resta indicar a nuestro Arduino que el pin IO9 va a ser utilizado como un pin de salida. Para hacerlo, usaremos un bloque con el método pinMode() de la CPU, editándolo para seleccionar el pin 9 y el modo 'output' y lo arrastraremos dentro de la subrutina de setup().



El aspecto final de nuestro flujograma se muestra a continuación.



Ya podemos arrancar la simulación utilizando la tecla PLAY del cuadro de control de simulación y comprobar que cuando presionamos el pulsador, se activa el led.



10.-Tutoriales de ejemplo de equipos Raspberry Pi.

10.1-Intermitencia sobre un led usando un equipo Raspberry Pi.

Este primer tutorial pretende abarcar el ciclo completo en la construcción de un proyecto Raspberry Pi usando la programación basada en flujogramas del Diseñador Gráfico. Arranca cuando creamos el proyecto en Proteus, continuaremos con la simulación completa de la intermitencia sobre el led usando Proteus y terminaremos cargando nuestro código en un equipo real Raspberry Pi.

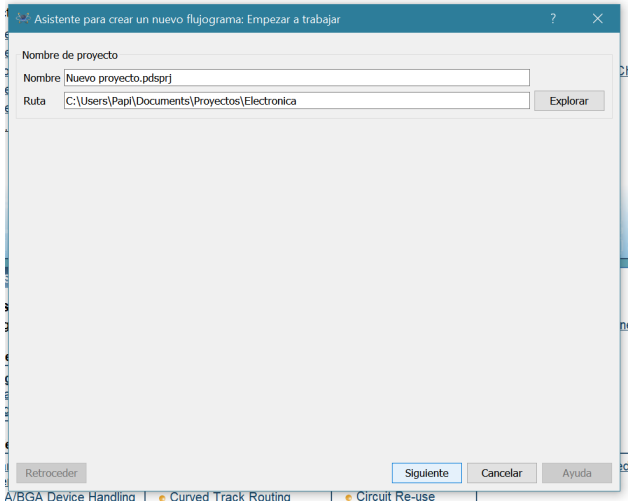
Nuestro objetivo es mantener el código tan simple como sea posible para mostrar las técnicas más importantes referentes a la construcción de proyectos desde el principio al fin. El trabajo con periféricos más complejos será cubierto en otros tutoriales de este mismo documento de ayuda.

10.1.1.-El asistente de creación de un nuevo proyecto.

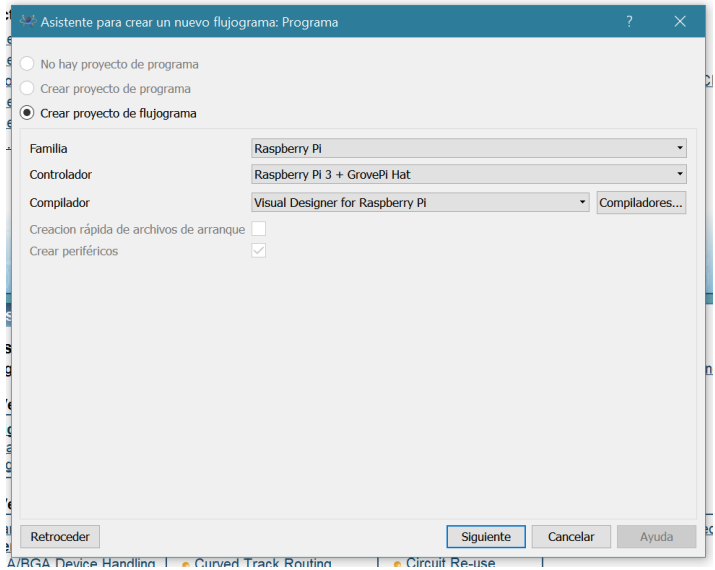
El Diseñador Gráfico es una pestaña de Proteus, así que para crear un nuevo proyecto del Diseñador Gráfico debemos partir de la pestaña Inicio. La forma más sencilla y cómoda de crear un proyecto del Diseñador Gráfico si sorpresas es utilizar el Asistente de creación de un nuevo proyecto de flujograma. Para abrir el asistente utilizaremos el botón Nuevo Flujograma del panel 'Empezar a trabajar'.



La primera ventana del asistente nos permite seleccionar la carpeta donde guardaremos nuestro proyecto y asignarle un nombre.

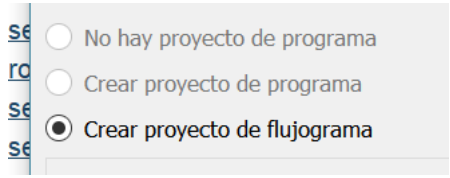


En la segunda ventana definimos el software de nuestro proyecto y es el lugar dónde realmente definimos la parte del proyecto donde utilizaremos el Diseñador Gráfico.

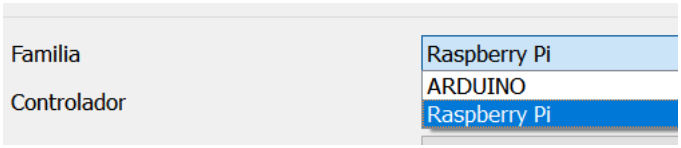


En la zona superior hay tres casillas de selección donde elegimos el entorno que usaremos para programar nuestro proyecto. Puesto que

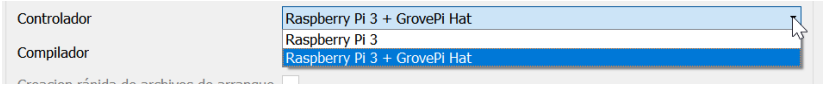
hemos usado el asistente para crear un proyecto de tipo flujograma, sólo está habilitada esta opción.



En la opción familia podemos elegir entre Arduino o Raspberry Pi, así que para este tutorial seleccionaremos la segunda.

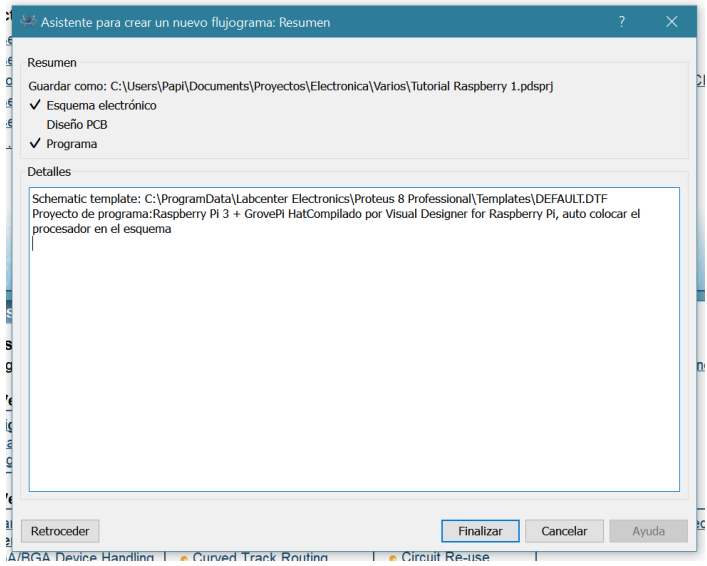


En la opción controlador, podemos elegir entre una placa Raspberry Pi simple o una con la tarjeta de expansión de conectoes Groove. Seleccionaremos para este tutorial la segunda opción.

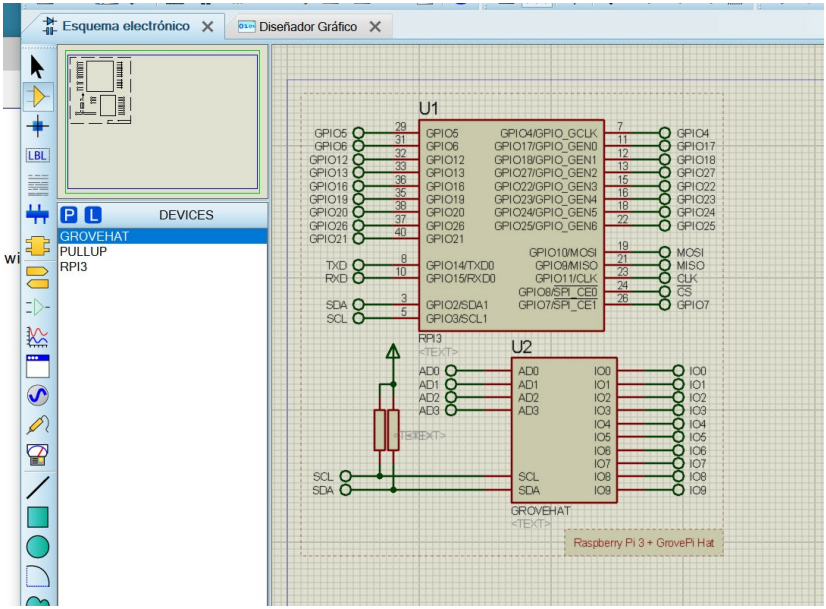


En la opción compilador sólo podemos elegir ‘Visual Designer’ porque hemos usado el asistente para crear un tipo de proyecto flujograma.

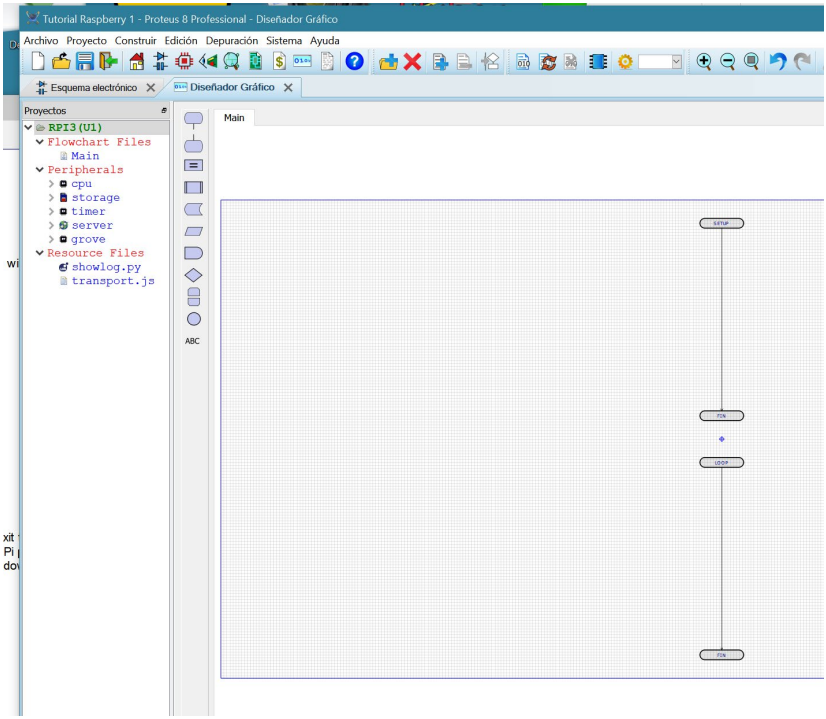
Ya podemos pulsar el botón ‘Siguiente’ para avanzar a la última pantalla. En esta pantalla Proteus nos muestra un resumen del proyecto que vamos a crear y nos permite retroceder para cambiar las opciones, cancelar y abandonar el asistente o finalizar y crear el proyecto. Elegiremos esta última opción.



Al finalizar el asistente tendremos el proyecto creado y listo para empezar a trabajar en él. En la pestaña 'Esquema electrónico' encontraremos el hardware creado.



Y en la pestaña ‘Diseñador Gráfico’ un esqueleto básico del programa que se ejecutará en el procesador Raspberry Pi compuesto por las dos rutinas básicas: la de configuración (setup) y la que se encarga de ejecutar el bucle principal del programa (loop).

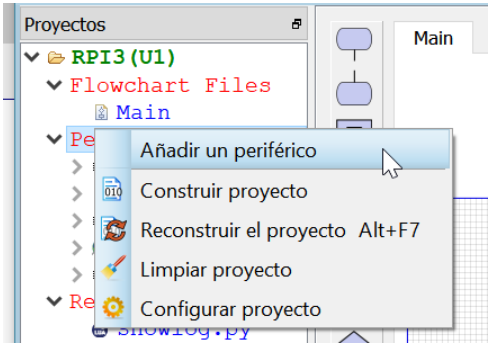


Ya tenemos todo lo necesario para empezar a diseñar nuestro primer proyecto.

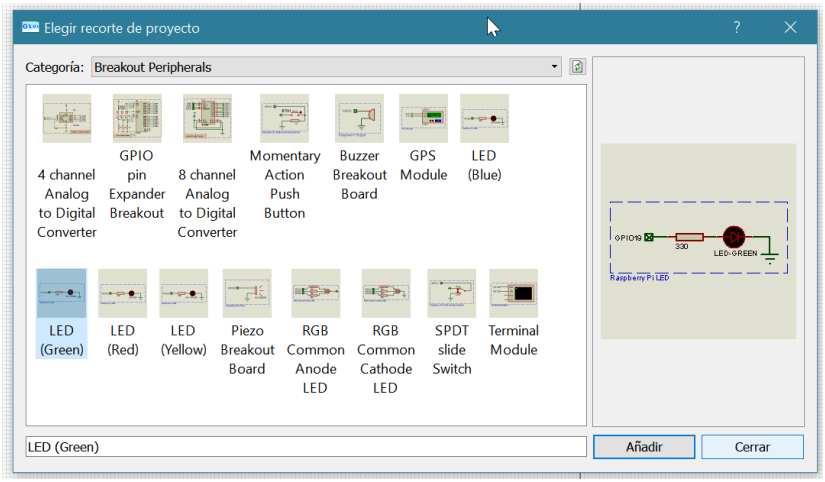
10.1.2.-Añadir los periféricos.

La primera tarea que tenemos que llevar a cabo es añadir tantos periféricos como deseemos a nuestro proyecto. Pueden ser tarjetas de expansión Raspberry PI, módulos Grove o tarjetas de periféricos. Los equipos Raspberry PI no necesitan utilizar tarjetas de expansión y es posible conectar módulos Grove o tarjetas de periféricos que se comuniquen directamente con el microprocesador utilizando conexiones directas a los diferentes pines GPIO disponibles.

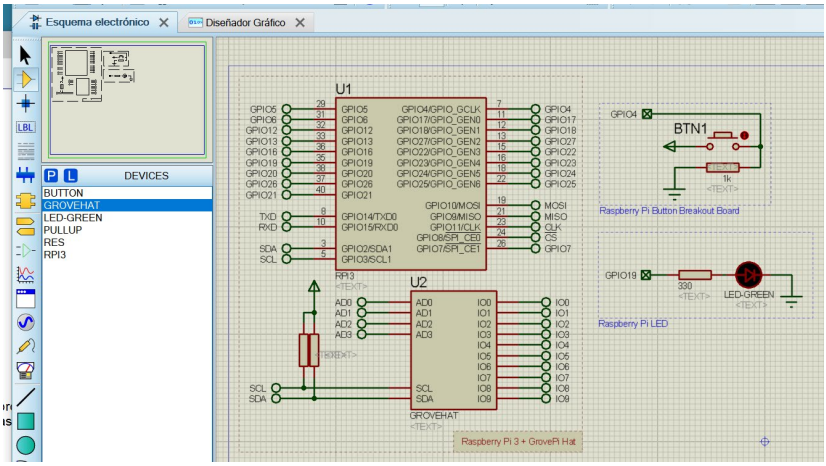
En este tutorial vamos a añadir una tarjeta de periférico compuesta por un circuito de led (resistencia y led). Desde la pestaña 'Diseñador Gráfico' nos colocamos en el árbol de estructura del proyecto y con el botón derecho del ratón abrimos el menú contextual. Seleccionamos la opción 'añadir un periférico'.



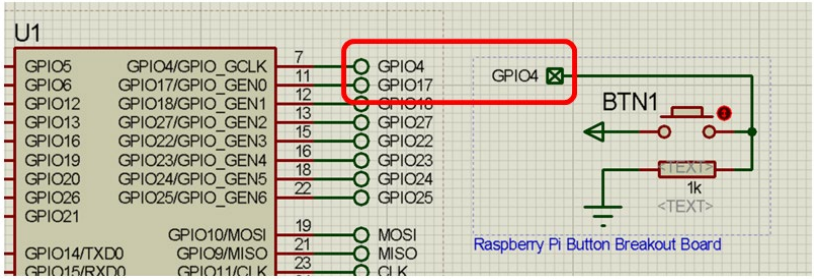
En la ventana de diálogo que aparece seleccionamos la categoría 'Breakout Peripherals' y seleccionamos alguno de los módulos de led que aparecen y damos al botón 'Añadir'.



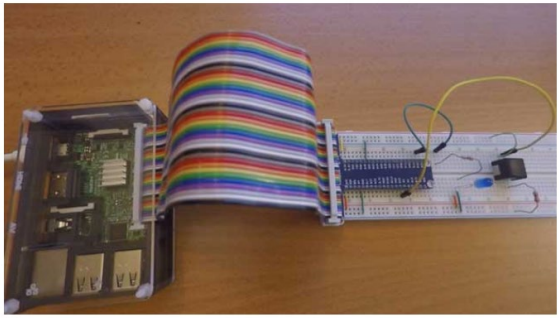
Hacemos lo mismo para seleccionar un 'Momentary Action Push Button'. Si cerramos la ventana emergente y volvemos a la pestaña 'Esquema electrónico' podemos comprobar que los dos periféricos han sido correctamente añadidos a nuestro esquema electrónico.



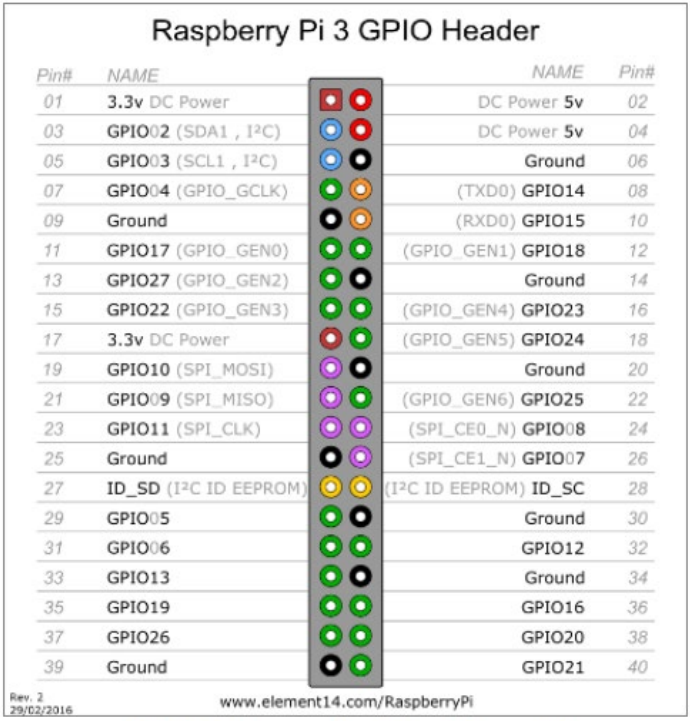
Para evitar llenar nuestro esquema electrónico de conexiones enrevesadas y confusas, Proteus ha utilizado terminales del mismo nombre para conectar los pines de los periféricos con la placa base de nuestro Raspberry Pi. Todos los terminales utilizados en el proyecto que tienen el mismo nombre están internamente enlazados como si existiera un cableado invisible.



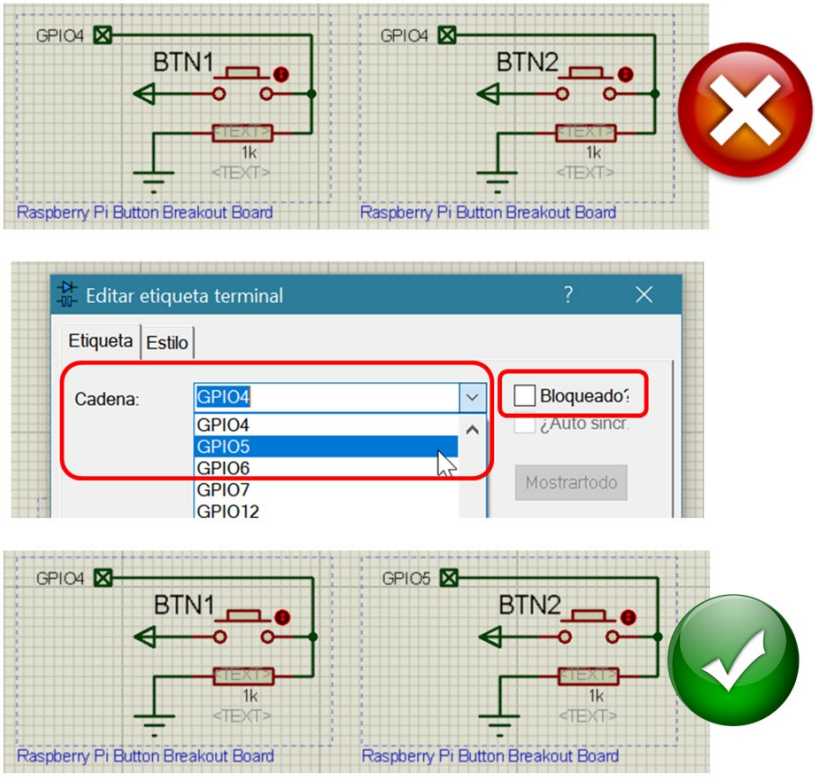
En el mundo real, el uso de estos dos terminales llamados GPI04 equivale a decir que el pin GPIO7 de nuestro Raspberry Pi está cableado al conector de nuestra tarjeta de periférico de pulsador.



Es importante que nos aseguremos que hemos conectado en los equipos reales los periféricos a los mismos pines GPIO de nuestra placa Raspberry Pi conforme a lo que definimos en nuestro diseño realizado con Proteus. Puede ser necesario consultar la función de cada pin GPIO para comprobar que lo estamos conectando todo bien.

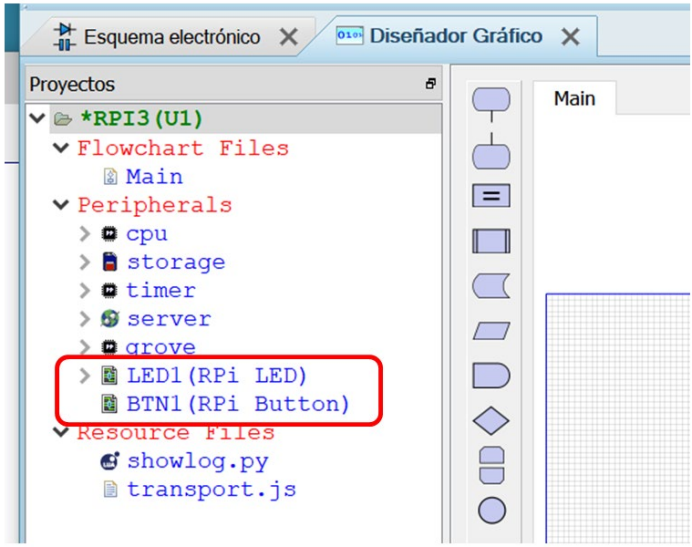


También es importante que nos aseguremos que no tenemos dos periféricos conectados a un mismo terminal, es decir, a dos terminales con el mismo nombre. Si al añadir un periférico sucediera que se conectan al mismo terminal, tendremos que editar la etiqueta del nombre del terminal desde el menú contextual de uno de ellos y cambiarlo a otro de los pines GPIO disponibles. Para editar la etiqueta es posible que tengamos que quitar la marca 'bloqueado'.

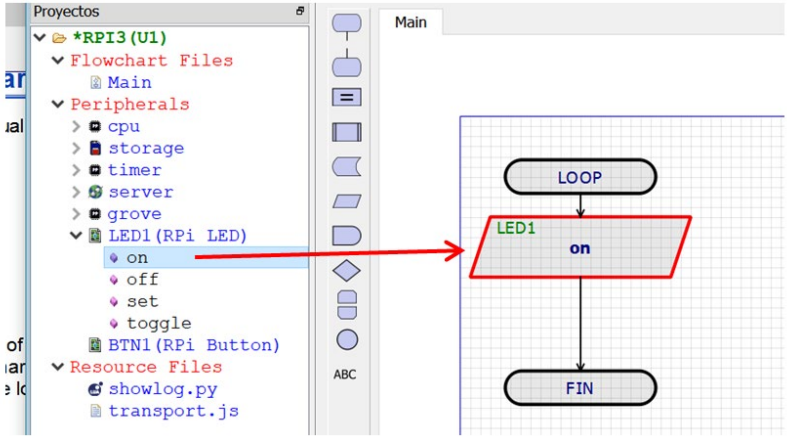


10.1.3.-Construir el flujograma.

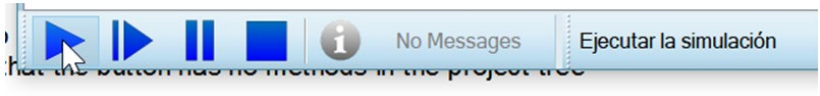
Coloquémonos de nuevo en la pestaña 'Diseñador Gráfico' y podemos comprobar que se han añadido en el árbol de estructura del proyecto nuevas ramas para los periféricos que acabamos de incorporar a nuestro diseño.

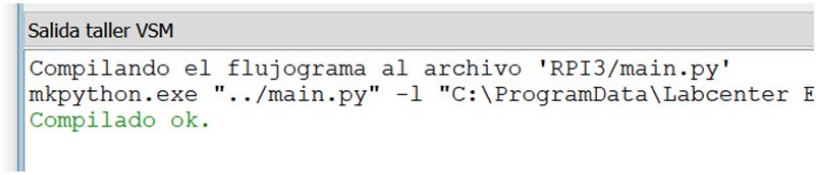


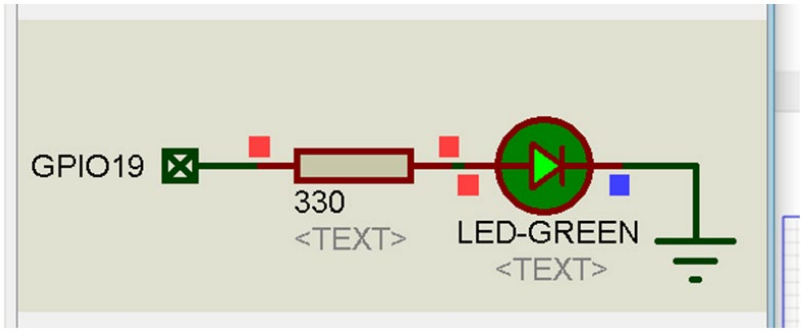
La potencia del Diseñador Grafico se muestra cuando expandimos la rama de un periférico y encontramos la lista de todos los métodos disponibles para su gestión. Los métodos son la técnica que usaremos con más frecuencia para gestionar desde el software el comportamiento de nuestro hardware. Por ejemplo, vamos a coger y arrastar el método 'On' asociado al 'Led1' y vamos a colocarlo dentro del bucle 'loop'.



Para comprobar que funciona nuestro código, sólo tenemos que utilizar el botón de puesta en marcha de la simulación. Al hacerlo, se compilar nuestro código, arrancará la simulación y veremos que nuestro led luce. Podemos comprobarlo desde la pestaña Esquema electrónico o utilizando las ventanas emergentes activas que aparecen en la pestaña 'Diseñador Gráfico'





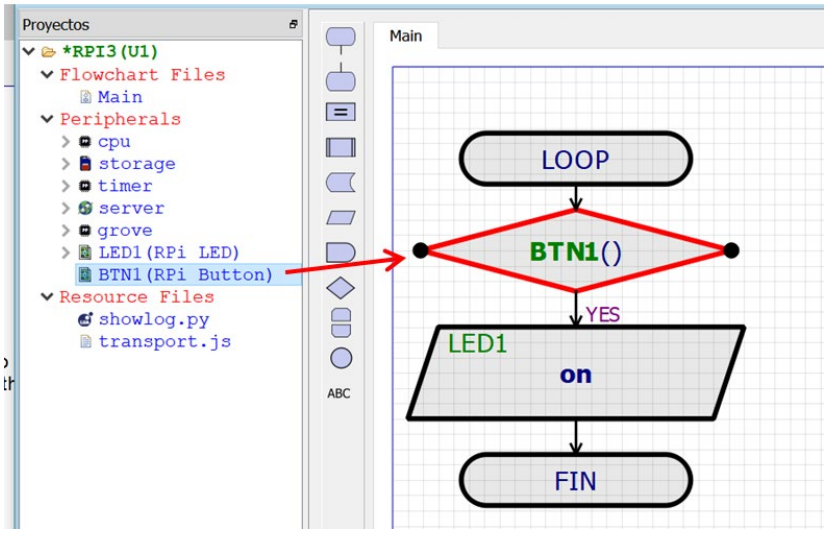


Podemos detener la simulación con el botón `parar` del cuadro de mandos de simulación y cambiaremos nuestro código para que lograr que el led se encienda o apague siguiendo las órdenes del usuario utilizando el pulsador.



La única información sensible que necesitamos para gestionar un pulsador es conocer si ha sido pulsado o no. Para ello usaremos la 'función de periférico' asociada con él. La función de periférico devuelve verdadero o falso dependiendo del estado del periférico en

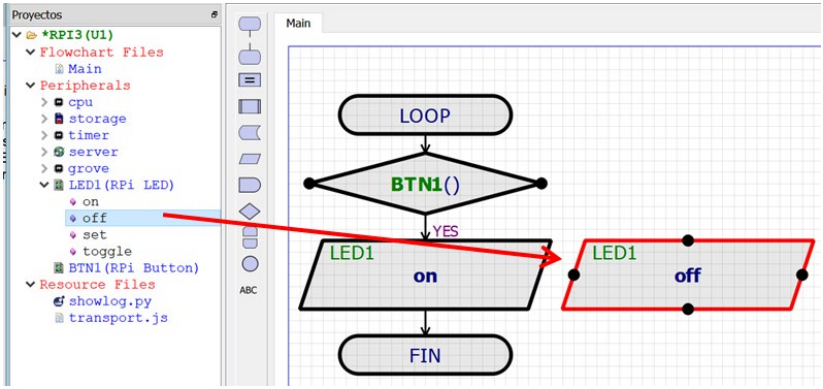
cuestión. Por ejemplo, la función de periférico asociada con un dispositivo tipo pulsador devuelve verdadero cuando el pulsador está presionado y devuelve falso cuando el pulsador está liberado. De forma similar, la función de periférico asociada con un dispositivo tipo led devuelve verdadero cuando el led está luciendo y devuelve falso cuando está apagado. Para utilizar una función de periférico en nuestro código sólo tenemos que coger y arrastrar el dispositivo en cuestión y colocarlo en la parte de nuestro flujograma deseada. En este caso dentro del bucle 'loop'.



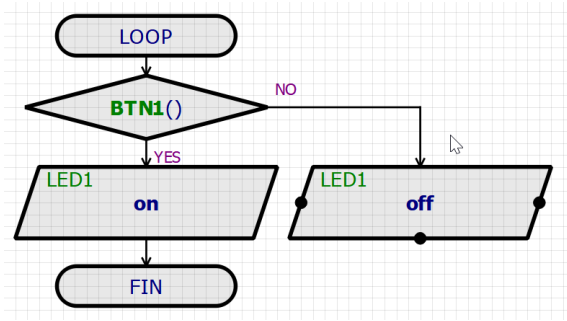
La función de periférico aparece como un bloque de decisión. Puesto que las funciones de periférico siempre devuelven 'verdadero' o 'falso', un bloque de decisión que nos permite dirigir el flujo de nuestro código en función de que recibamos un valor verdadero o falso es la mejor opción posible. La lectura del bucle 'loop' en este momento sería: 'vigila el estado del pulsador y si está activo enciende el led'.

Nuestro controlador estará constantemente vigilando el estado del pulsador y cuando detecte que han presionado el pulsador, encenderá el led. Por lo tanto, nos falta escribir el código necesario para indicar qué debe suceder cuando el pulsador deje de estar actuado. Como deseamos que el led se apague cuando el pulsador quede liberado, tenemos que 'coger y arrastar' el método 'off'

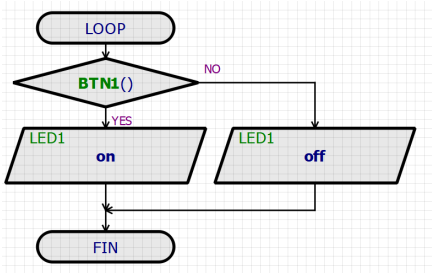
asociado al led y colocarlo en el espacio libre junto al bloque que encendía el led.



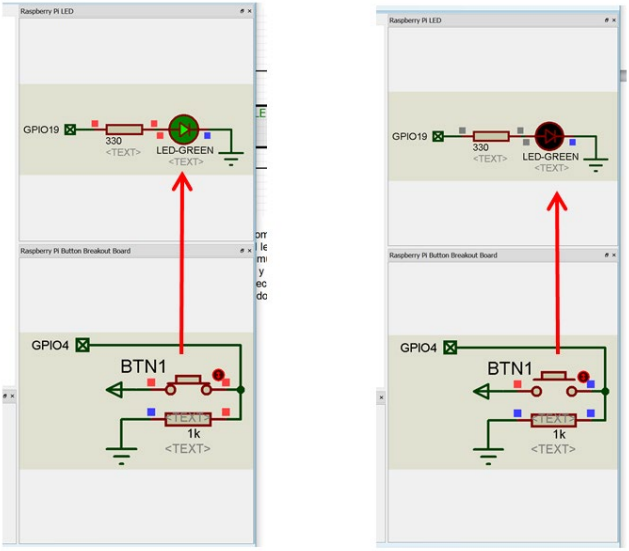
Y, a continuación, enlazar la salida NO del bloque de decisión asociado al pulsador con la entrada superior del bloque de asignación que apaga el led.



Y terminaremos enlazando la salida inferior del bloque de asignación que apaga el led antes del final del bucle ‘loop’.



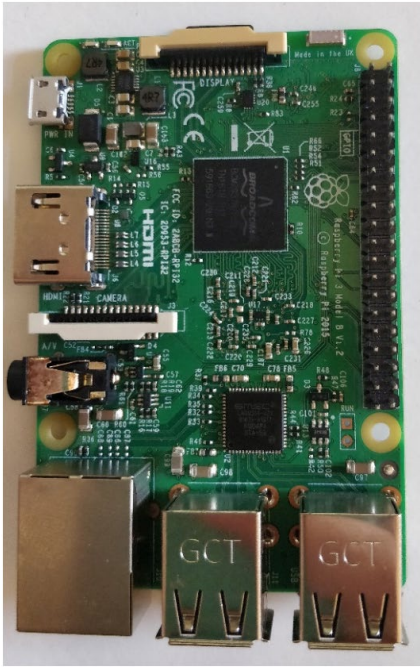
La lectura del bucle 'loop' en este momento sería: 'vigila el estado del pulsador, si está activo enciende el led y si está liberado apaga el led'. Podemos poner en marcha la simulación con el botón 'PLAY' para que nuestro código se compile y arranque la simulación. Ya podemos comprobar que nuestro proyecto funciona como deseamos actuando sobre el pulsador y observando el comportamiento del led.

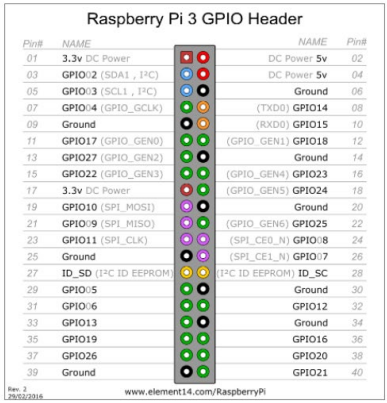


10.1.4.-Comprobando la equivalencia del hardware en un equipo Raspberry Pi real.

En este apartado asumimos que hemos configurado correctamente nuestro equipo Raspberry Pi para trabajar con Proteus. El proceso

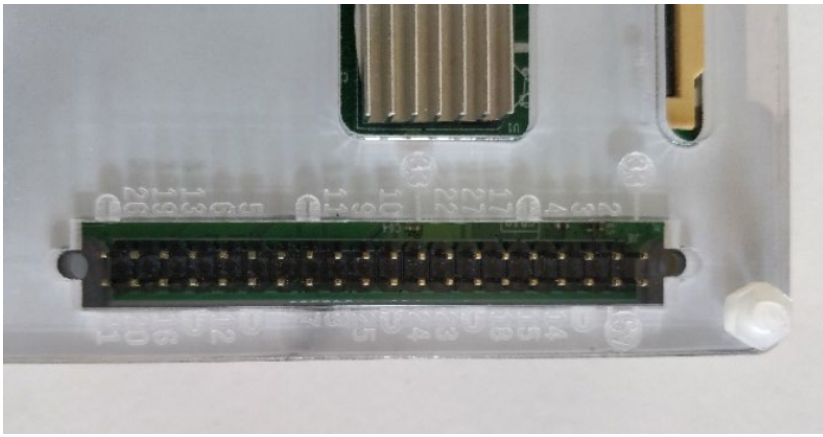
de configuración se detalla en el apartado 8.3 de este mismo documento de ayuda.





En la imagen de la izquierda tenemos una placa base Raspberry Pi 3 y en la de la derecha el mapeado de sus pines La primera cosa importante a tener en cuenta es la orientación adecuada de ambas imágenes para identificar perfectamente la función de cada pin del conector y no cometer errores al conectar los periféricos. En nuestro caso hemos hecho que coincidan en las dos la posición de los pines del conector. Los dos pines superiores tal y como se muestran son en los dos casos los de alimentación (3,3V en el de la izquierda y 5V en el de la derecha). En la página web oficial de Raspberry se puede comprobar que esta es la definición de pines.

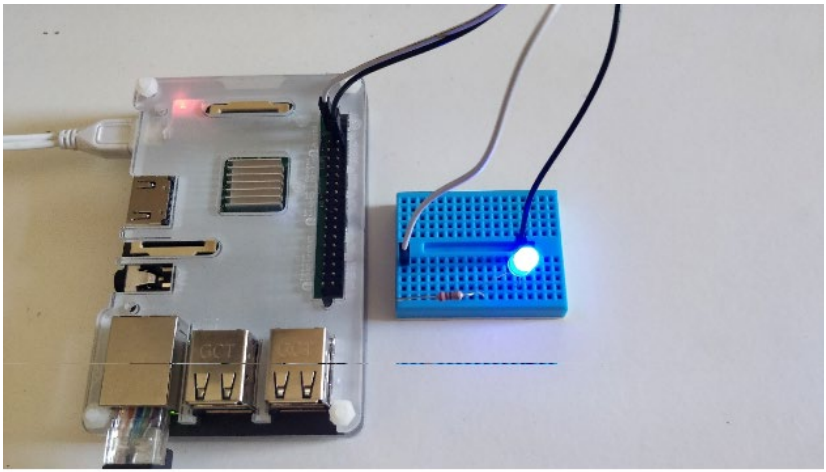
En algunas envolventes disponibles en el mercado para contener las placas principales Raspberry Pi figuran marcados cada pin con su funcionalidad para facilitar su identificación.



En los casos en que no usemos envolvente o que la envolvente que utilicemos no tenga marcados los pines, resulta muy conveniente que utilicemos una plantilla que podemos realizar de forma muy sencilla con una cartulina.



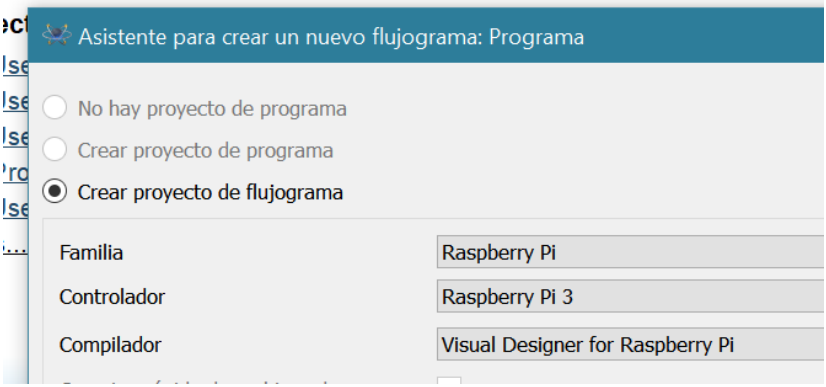
En este primer tutorial, en nuestro diseño electrónico henos conectado nuestro led al pin GPIO19. La primera cuestión importante a tener en cuenta es que no hay que confundir el pin 19 del conector que podemos ver en el mapeado que se corresponde con el pin GPIO10 del Raspberry Pi. Los pines del conector no coinciden con los pines GPIO del Raspberry Pi, por lo que tenemos que ser cuidadosos y comprobar bien este aspecto. En nuestro caso el pin GPIO19 se corresponde con el pin 35 del conector.



Si conectamos correctamente nuestro led y nuestro pulsador en los pines correspondientes, el equipo tendrá que responder cómo esperamos y encenderse el led mientras mantenemos actuado el pulsador y apagarse el led cuando liberemos el pulsador.

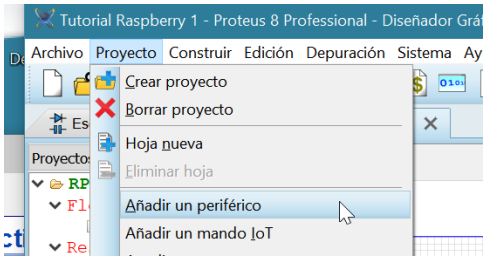
10.2.-Lámpara de iluminación nocturna usando un equipo Raspberry Pi con Grove.

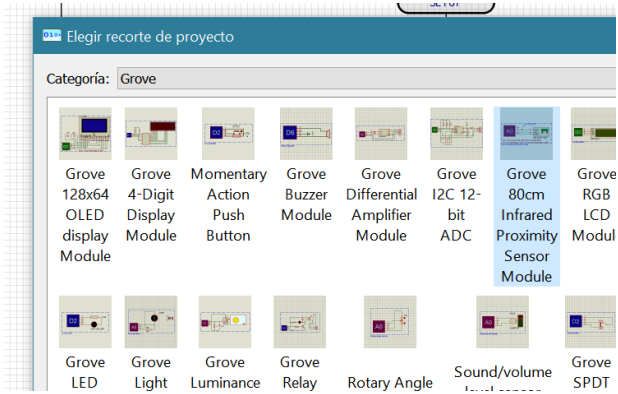
Para este tutorial vamos a utilizar un par de módulos Grove con los sensores que necesitamos y uno de led. Con todo ello diseñaremos una lámpara de iluminación nocturna muy simple. La creación y configuración inicial de un proyecto del Diseñador Gráfico utilizando el asistente para crear un flujograma, se cubrió en el tutorial anterior.

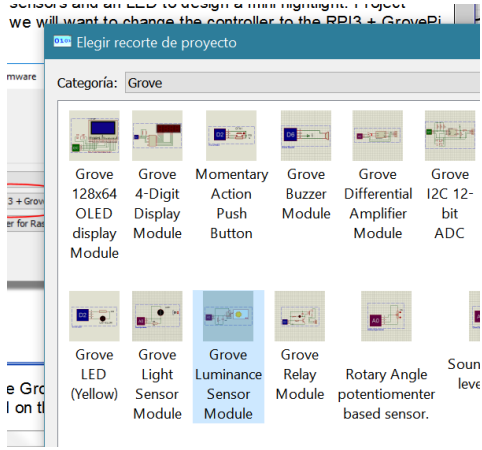


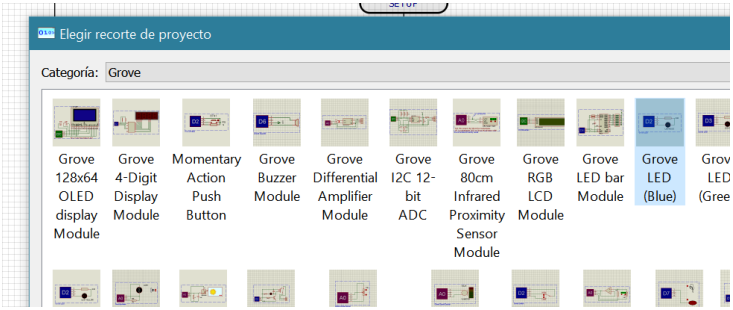
10.2.1.-Añadir los periféricos.

Vamos a añadir el módulo Grove de sensor de proximidad mediante infrarrojos, el módulo Grove de sensor de luz y el módulo Grove de led. Todos ellos, los podemos incluir en nuestro proyecto desde la pestaña Diseñador Gráfico utilizando el comando añadir un periférico del menú Proyecto.



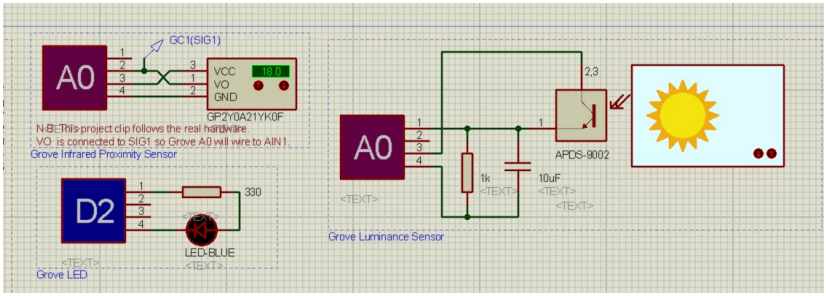


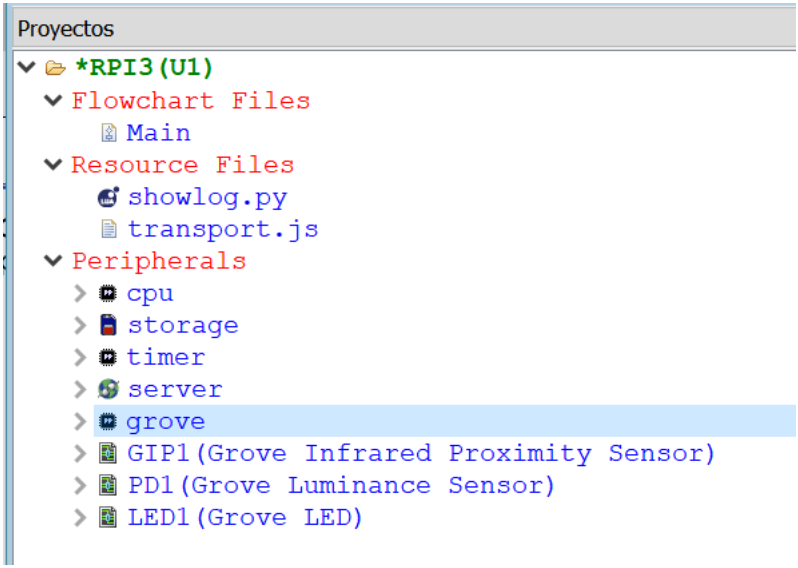




Si hemos hecho todo el proceso correctamente y hemos añadido los tres módulos Grove a nuestro proyecto, tendremos que verlos en la

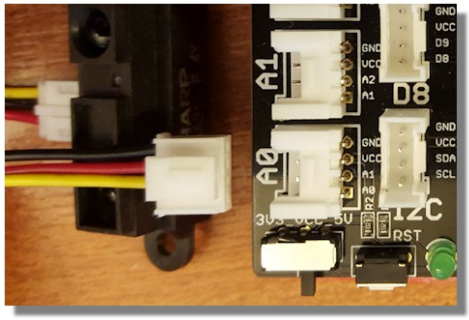
pestaña Esquema Electrónico y en el árbol de estructura del proyecto de la pestaña Diseñador Gráfico.

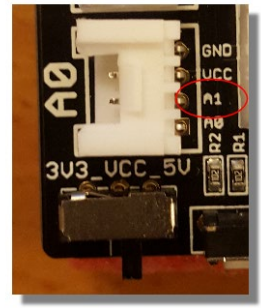




En los proyectos desarrollados con el Diseñador Gráfico es de vital importancia configurar los conectores de los módulos Grove adecuadamente. Esta operación no siempre es tan sencilla como parece. Veamos un ejemplo.

El sensor de infrarrojos requiere sólo tres cables para su conexión (VCC, GND y A0), quedando el cuarto como NC (no conectado). Sin embargo, si nos fijamos bien, cuando enchufamos nuestro módulo Grove en el enchufe rotulado A0 el sensor se conecta, no sólo al pin A0, sino también al pin A1.



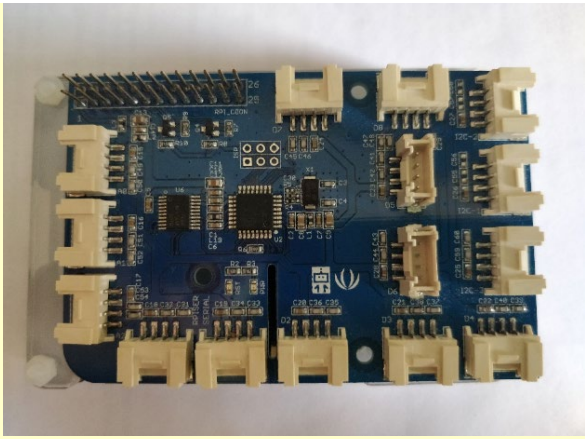


Hasta aquí todo correcto y sin problemas. Pero si enchufamos ahora nuestro sensor de luminosidad en el enchufe A1, utilizará también el pin A1 de nuevo (porque el pin A1 se utiliza en ambos, el enchufe A0 y el enchufe A1) y se generará un conflicto.

Por este motivo tenemos que enchufar nuestro sensor de luminosidad en el enchufe A2 y dejar libre el A1.



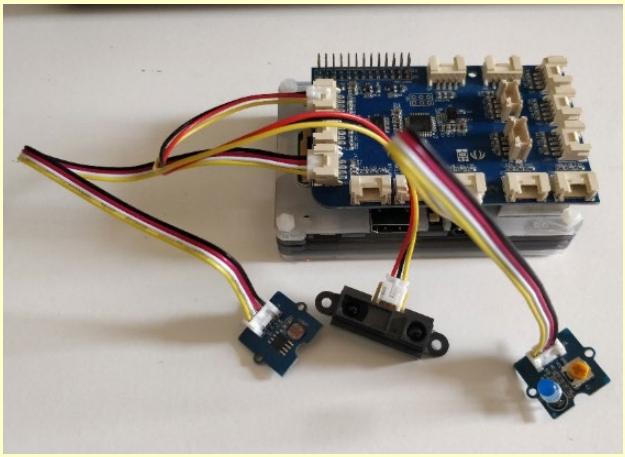
Debemos tener siempre muy en cuenta esta regla de oro cuando utilicemos módulos Grove: hay que fijarse detalladamente en los enchufes de la tarjeta de expansión Grove, comprobar bien los pines utilizados por cada enchufe y prevenir los conflictos entre los diferentes módulos que conectemos a la tarjeta de expansión. A diferencia de las placas de expansión Grove para Arduino, los conflictos no se muestran en el marcado de pines de cada enchufe A<n>.



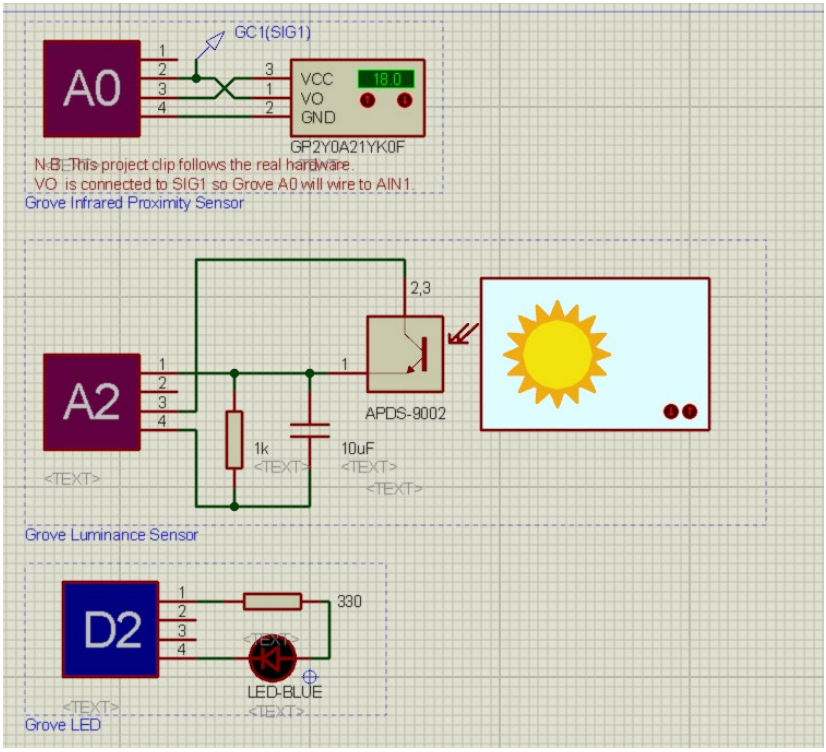
Cada enchufe se conecta a dos pines de entrada/salida de nuestro microprocesador Raspberry Pi y existe un solapamiento entre los diferentes enchufes:

- El enchufe A0 utiliza los pines A0 y A1.
- El enchufe A1 utiliza los pines A1 y A2.
- El enchufe A2 utiliza los pines A2 y A3.
- El enchufe A3 utiliza los pines A3 y A4.

Con los enchufes que se conectan a los pines digitales sucede exactamente lo mismo.



Por este motivo hemos configurado nuestro proyecto utilizando el sensor de proximidad conectado al enchufe A0 y el sensor de luminosidad al enchufe A2. El led lo podemos conectar en cualquiera de los enchufes destinados a módulos digitales (así que no tenemos problemas en conectarlo al enchufe D2 que Proteus ha asignado automáticamente).



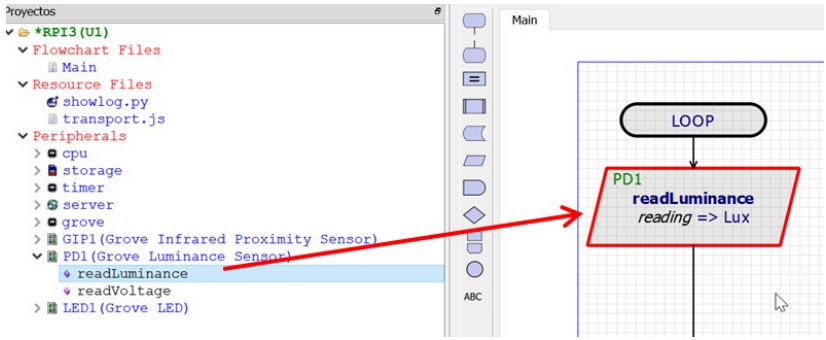
10.2.2.-Construir el flujograma.

En nuestra sencilla lámpara de iluminación nocturna nos fijamos como objetivo que el led se encienda cuando se cumplan dos condiciones:

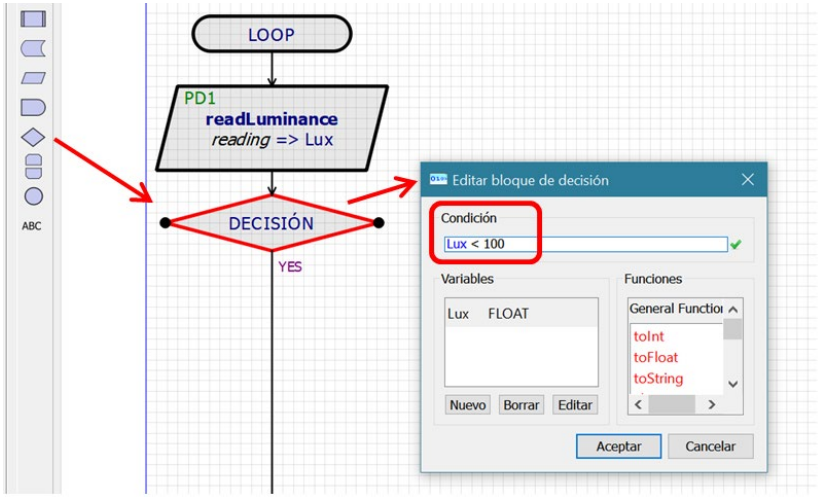
- Que esté oscuro
- Que alguien se encuentre cerca de la lámpara

Las dos condiciones debemos controlarlas desde nuestro flujograma y tomar las decisiones pertinentes. Empezaremos 'cogiendo y

arrastrando' el método readLuminance() asociado al sensor de luminosidad dentro del bucle de control 'loop'. Observe que el valor de lectura se guarda de forma automática en la variable de nombre 'Lux'.



El sensor de luminosidad devuelve un valor entre 0 y 1.000, correspondiéndose 0 con la máxima oscuridad y 1.000 con la mayor luminosidad. Así que podemos utilizar un bloque de decisión para detectar cuando la luminosidad leída se encuentra por encima del valor 100 (un valor que nos permitirá llevar a cabo una primera comprobación del funcionamiento del sensor y aprender a qué nivel de luminosidad reacciona).

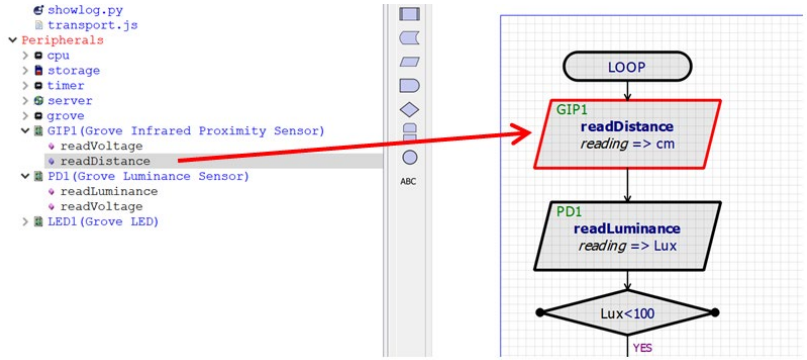


> La hoja de datos del sensor de luminosidad APDS-9002 usado en el módulo Grove proporciona l ainformación necesaria para saber que los valores leídos se corresponden lux leídos. Podemos encontrar la información completa en el siguiente enlace: http://wiki.seeedstudio.com/Grove-Luminance_Sensor/

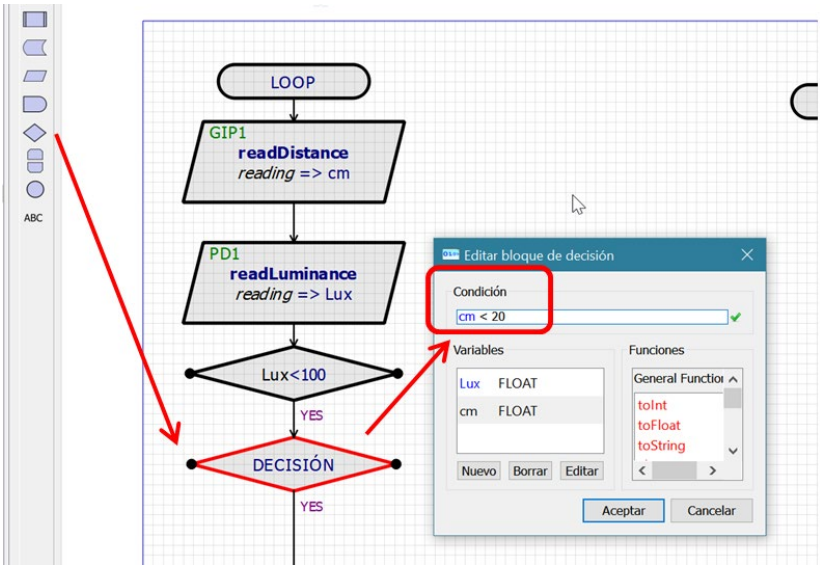
El sensor de proximidad nos proporciona la lectura de la distancia a la que se encuentran el objeto detectado, así que volveremos a coger y arrastrar el método encargado de recuperar este valor (readDistance) y colocarlo en el bucle loop encima de la lectura de la luminosidad. Observe que el valor de lectura se guarda en este segundo caso de forma automática en la variable de nombre 'cm'.

Podemos encontrar más información sobre el funcionamiento de este sensor en el siguiente enlace:

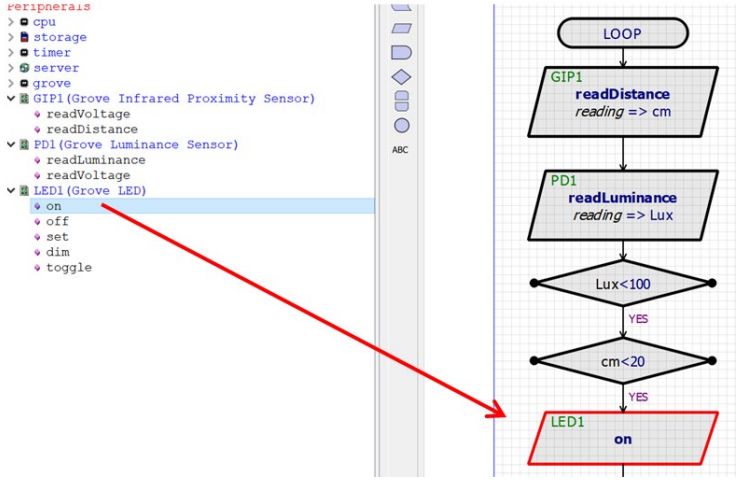
http://wiki.seeedstudio.com/Grove-80cm_Infrared_Proximity_Sensor/



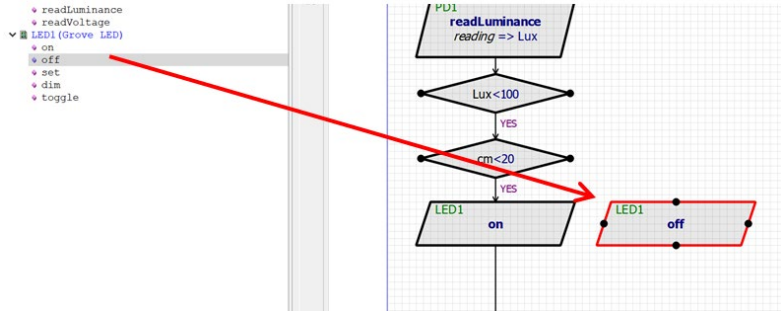
El sensor de proximidad devuelve valores entre 0 y 80cm. Así que podemos utilizar un segundo bloque de decisión para detectar cuando tenemos un objeto a una distancia inferior a 20cm (un valor que nos permitirá llevar a cabo una primera comprobación del funcionamiento del sensor).



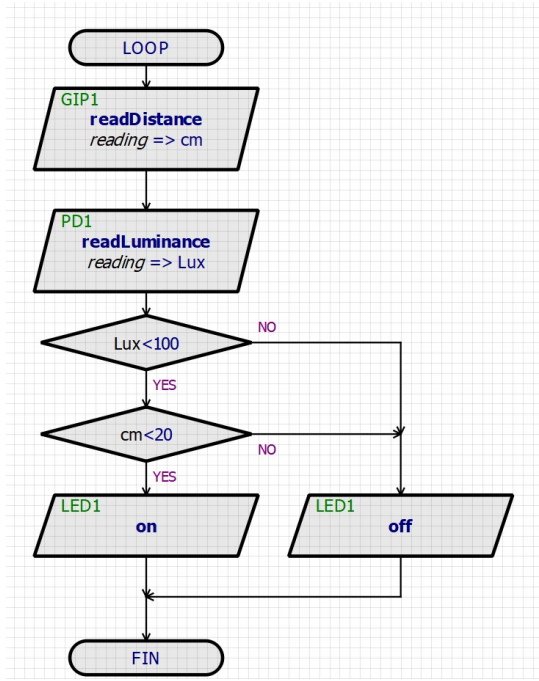
Tal y como hemos diseñado nuestro flujograma, al poner el segundo bloque de decisión a la salida del valor YES del primero, cuando las dos condiciones se cumplen (que la luminosidad es mayor de 100lux y que tenemos un objeto a menos de 20cm) nos encontraremos en la salida del valor YES del segundo. Es ahí donde podemos situar el método asociado al led encargado de hacer que se encienda.



A continuación colocaremos el método asociado al led encargado de hacer que se apague situándolo a la derecha del encargado que se encienda.

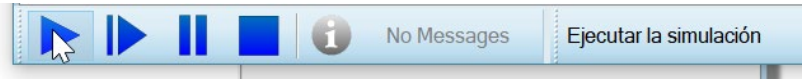


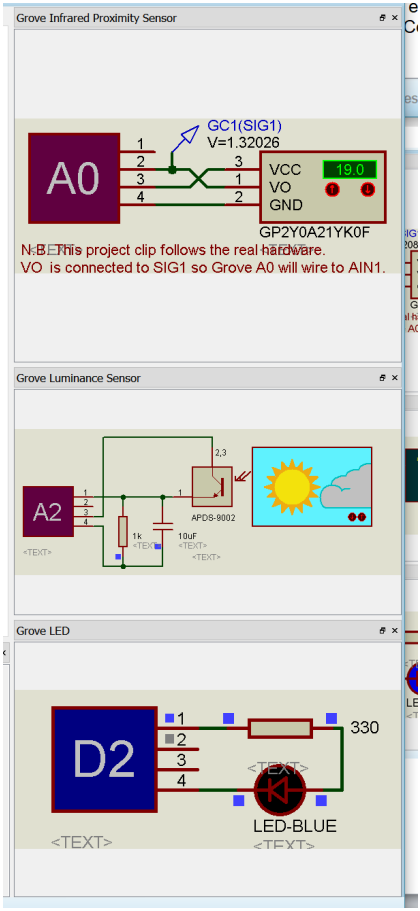
Y sólo nos queda conectar las salidas NO de los dos bloques de decisión a la entrada del bloque encargado de apagar el led y la salida de este último con la línea principal del bucle loop. El resultado final se muestra en la siguiente imagen.



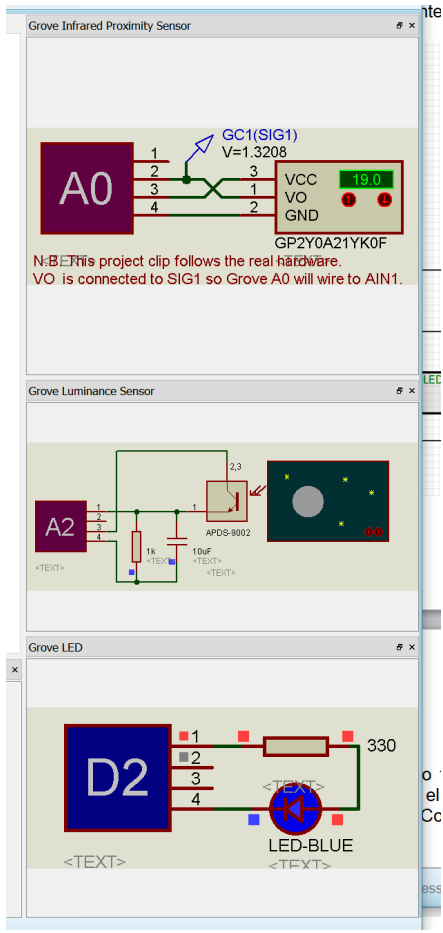
10.2.3.-Simular el proyecto.

Para comprobar que nuestro proyecto funciona correctamente todo lo que necesitamos hacer es pulsar el botón PLAY del cuadro de mando de control de la simulación. Con ello se compilará nuestro proyecto y se arrancará la simulación.

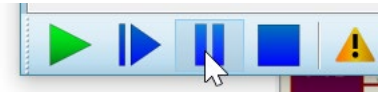




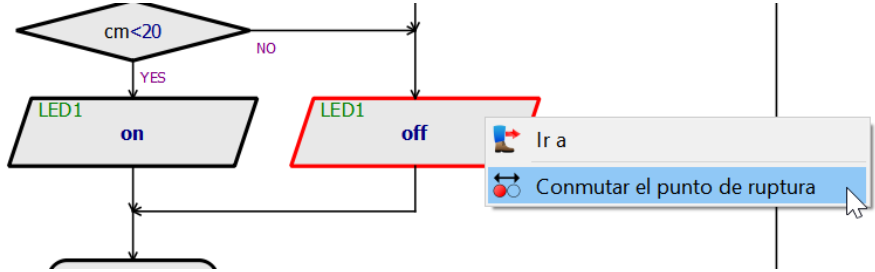
Si manipulamos los controles de los dos sensores para que la distancia del objeto detectado sea menor de 20 y la luminosidad menor de 100 veremos que el led se enciende.



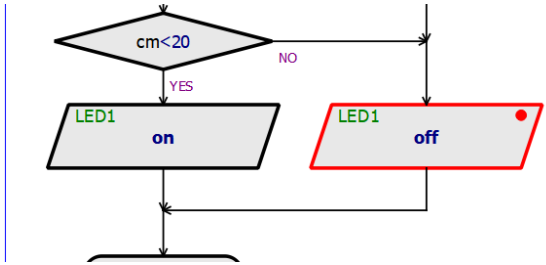
Ahora podemos poner en pausa la simulación para seguir descubriendo las posibilidades que nos permite Proteus para depurar nuestros proyectos.



Situaremos el cursor del ratón sobre el bloque que apaga el led y abriremos el menú contextual con el botón derecho. Seleccionamos la opción 'Conmutar el unto de ruptura'.

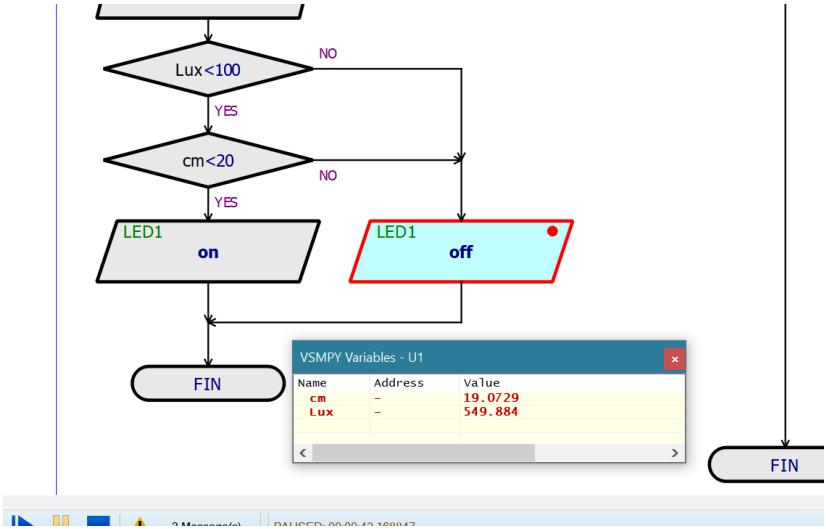


Al hacerlo aparece un punto rojo en el bloque con el punto de ruptura.

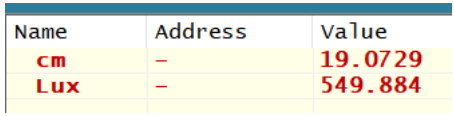


Pondremos en marcha de nuevo la simulación con el botón PLAY y modificaremos los valores de distancia y luminosidad hasta provocar la activación del punto de ruptura (al darse las condiciones para apagar el led).

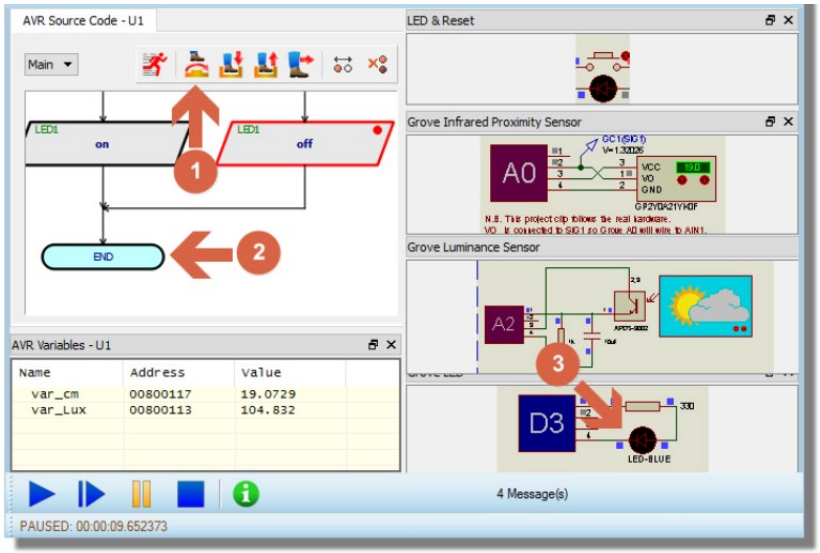
Cuando lo alcancemos, el bloque con el punto de ruptura se iluminará de color azul y en la ventana de inspección de variables, podemos comprobar los valores de las dos variables que manejamos en nuestro proyecto.



Podemos comparar los valores con las condiciones que hemos programado y comprobar si todo funciona como deseamos. Programamos que la distancia menor de 20 y la luminosidad menor de 100 encienden el led. En nuestro caso se ha apagado el led porque el valor de luminosidad está por encima del valor fijado.



También es posible ejecutar la simulación paso a paso de Proteus utilizando los iconos correspondientes de la barra de herramientas para comprobar cómo se lleva a cabo el apagado del led.



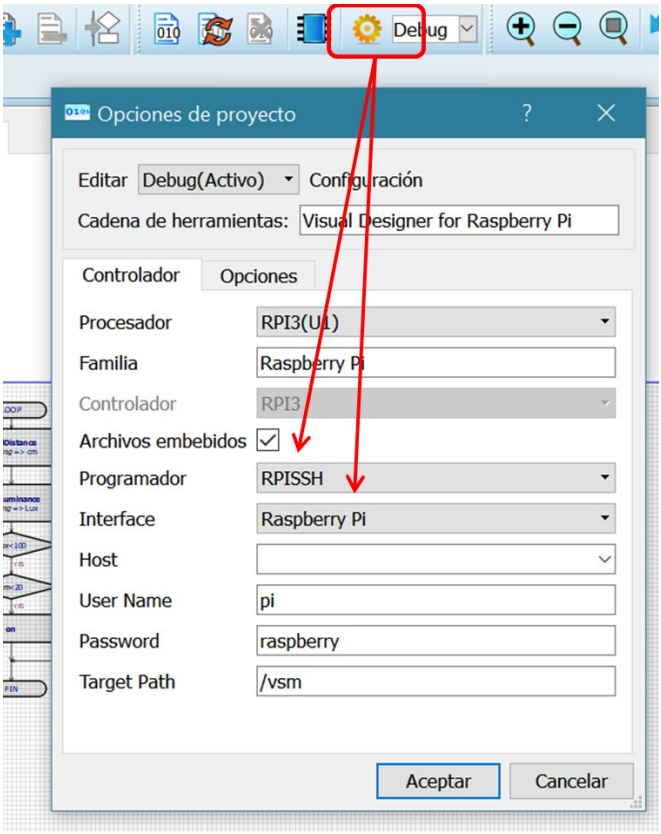
10.2.4.-Cargando el programa en un equipo Raspberry Pi real.

> En este apartado asumimos que hemos configurado correctamente nuestro equipo Raspberry Pi para trabajar con Proteus. El proceso de configuración se detalla en el apartado 8.3 de este mismo documento de ayuda.

Cuando ya hemos comprobado que nuestro proyecto se comporta como deseamos en la simulación, el siguiente paso es cargar el código en nuestro equipo Raspberry PI real. El proceso se lleva a cabo en tres pasos.

1. Conectaremos los módulos Grove a la placa de expansión de conectores Grove utilizando los mismos conectores que hemos definido en nuestro esquema electrónico. Asegurarnos de que la regla de oro vista en el apartado 10.2.1 se cumple.
2. Conectar la placa de expansión Grove a la placa base Raspberry Pi y conectar nuestro equipo Raspberry a la misma wifi a la que está conectado nuestro ordenador

personal en el que estamos ejecutando Proteus. En la pestaña Diseñador Gráfico, en la ventana de diálogo de 'configuración de proyecto' seleccionaremos como programador 'RPISSH' y como interface 'Raspberry Pi'



3. Utilizamos el botón 'enviar' de la barra de herramientas para cargar el código en nuestro equipo Raspberry Pi.



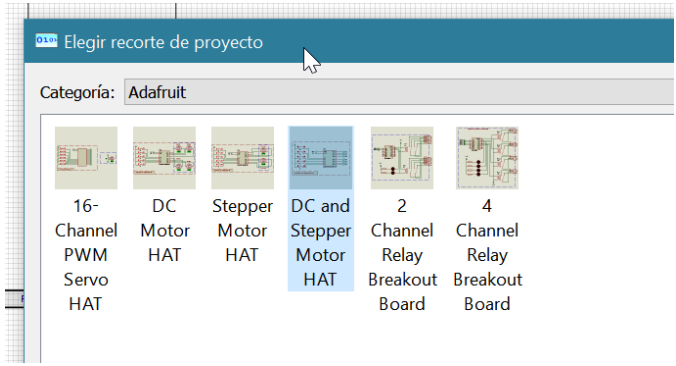
Y ya podemos ejecutar nuestro proyecto en nuestro equipo Raspberry Pi.

10.3.-Control de motores utilizando un equipo Raspberry Pi

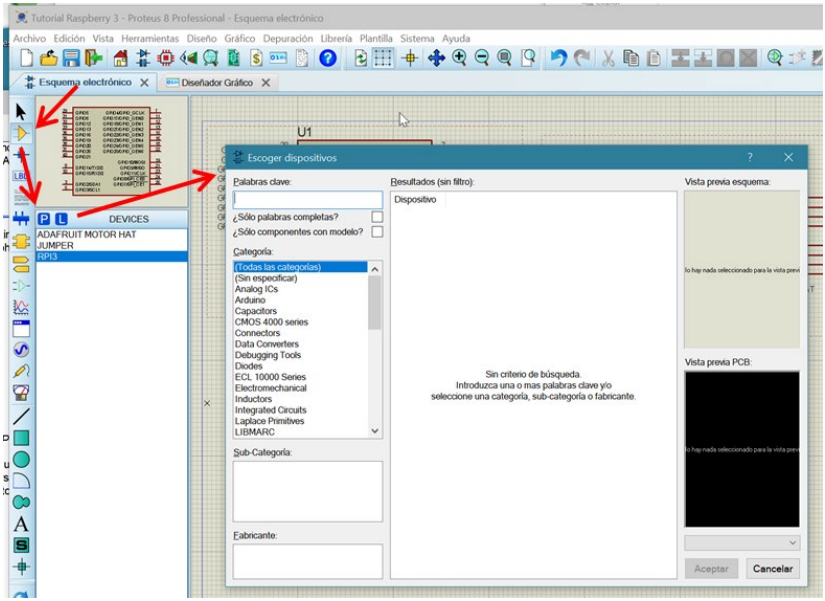
En este tutorial vamos a mostrar cómo podemos controlar de forma muy sencilla un motor de corriente continua y un servomotor utilizando del Diseñador Gráfico. Vamos a utilizar una tarjeta de expansión de control de motores de corriente continua y servomotores de la firma Adafruit para Raspberry Pi.

10.3.1.-Configuración.

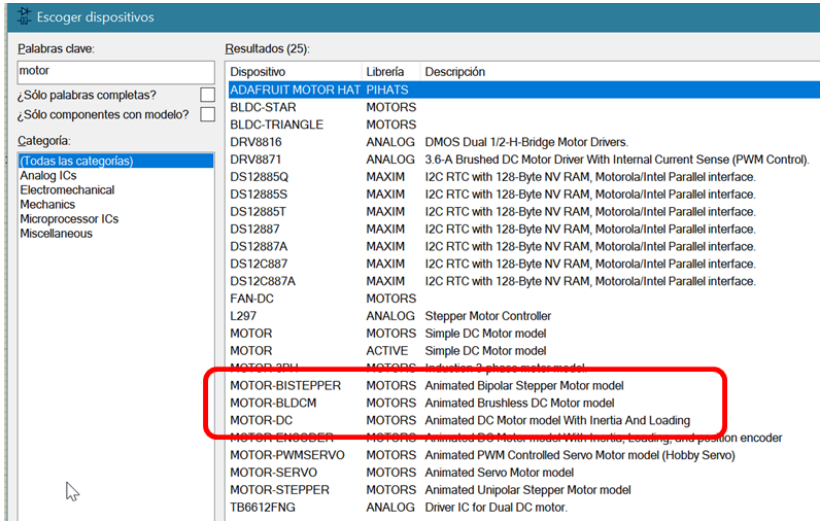
Como en los tutoriales anteriores empezaremos creando un proyecto para flujograma utilizando el asistente. A continuación añadiremos una tarjeta de expansión llamada 'DC and Stepper Motor Hat' que se encuentra en el apartado Adafruit.

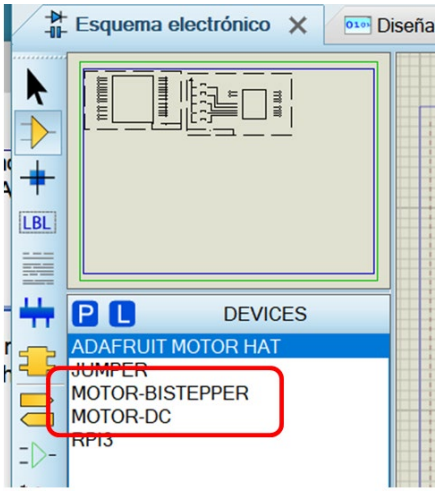


Ha diferencia de las tarjetas de expansión de control de motores de Arduino, en las tarjetas de expansión de control de motores de Raspberry Pi necesitamos conectar los motores por nuestra cuenta. Esto significa que tenemos que seleccionar desde la pestaña Esquema electrónico la herramienta 'modo componentes' y usar el botón 'P' para que se abra la ventana de diálogo de selección de dispositivos.

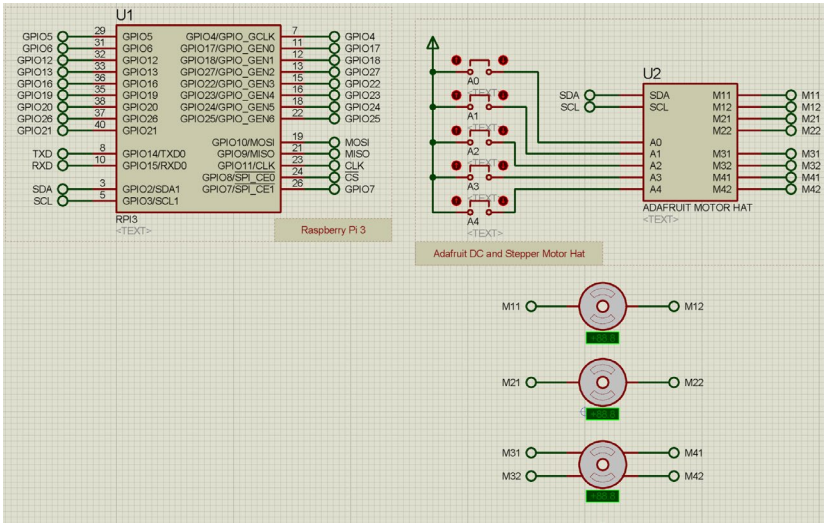


Podemos escribir ‘motor’ en el cuadro de búsqueda ‘Palabras clave’ para reducir la lista de dispositivos a elegir. Elegiremos un MOTOR-DC y un MOTOR-BISTEPPER para incluirlos en nuestro proyecto.



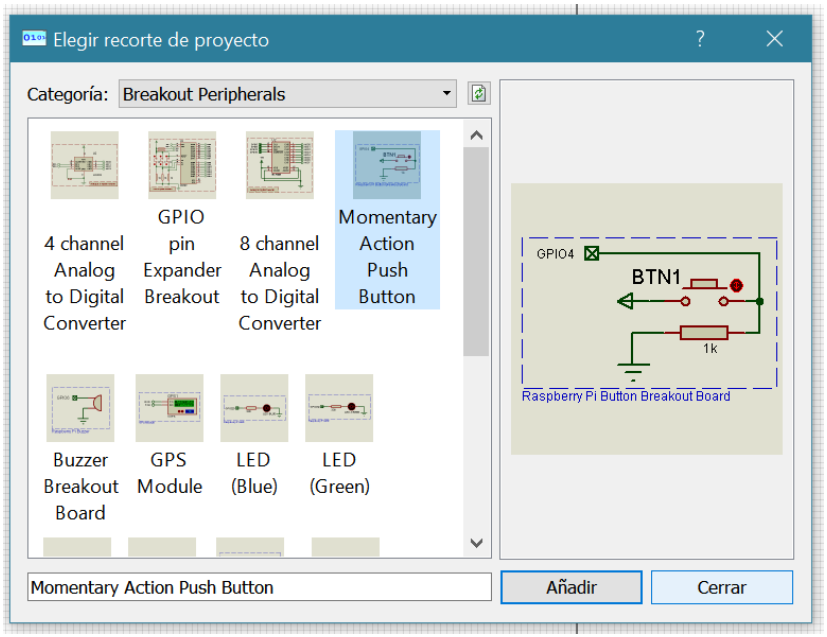


Colocaremos en nuestro esquema dos motores de corriente continua y un servomotor. Los conectaremos a nuestra tarjeta de expansión Adafruit utilizando terminales de los nombres adecuados. El resultado final tiene que ser similar al mostrado en la imagen siguiente.

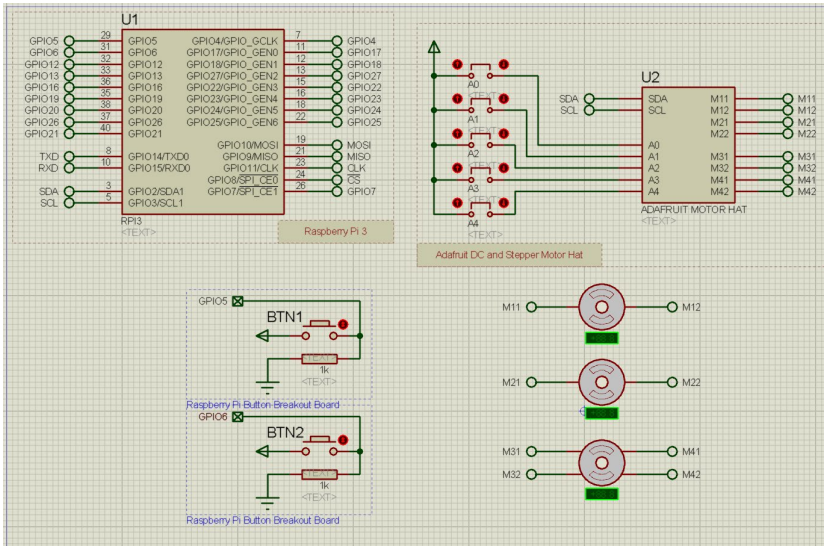


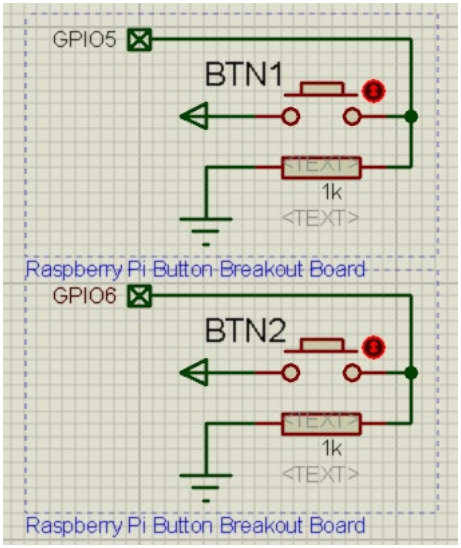
10.3.2.-Construir el flujograma.

Para la primera fase de este proyecto necesitamos dos pulsadores y un motor de corriente continua. Con un pulsador arrancaremos el motor y con otro ordenaremos que pare. Así que vamos añadir dos tarjetas de periféricos de pulsador siguiendo el método habitual que hemos seguido en los tutoriales anteriores.

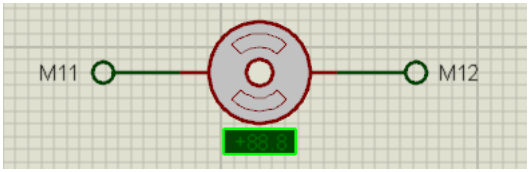


Nos aseguraremos que los pulsadores se conectan a los pines GPIO5 Y GPIO6.

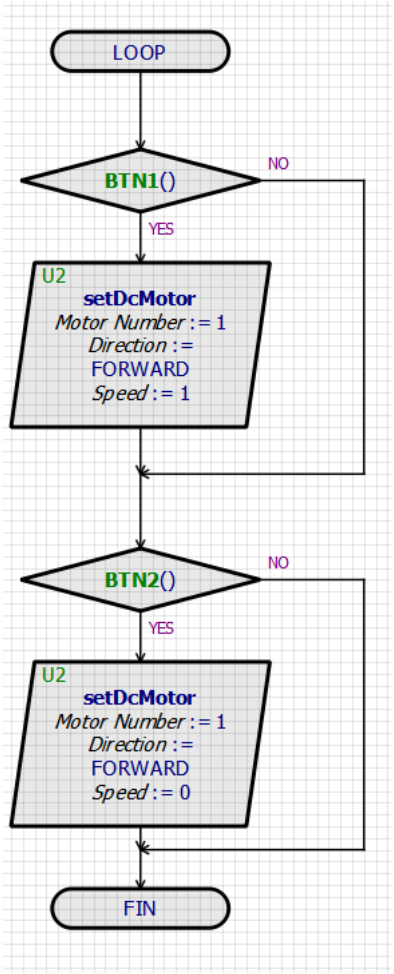




De los tres motores, vamos a usar el que se conecta a los terminales M11 y M12.



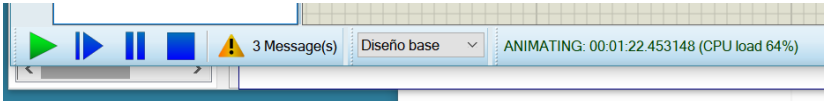
Ahora podemos añadir los bloques necesarios a nuestro flujograma para gestionar el control del motor con los dos pulsadores. Nuestro objetivo es arrancar el motor con el pulsador BTN1 y detenerlo con el pulsador BTN2. Utilizando el Diseñador Gráfico no será necesario que nos preocupemos por escribir código de ningún tipo para configurar nuestro proyecto. El flujograma final debe tener este aspecto.



10.3.3.-Simulación y depuración.

Ya tenemos nuestro proyecto listo. Así que ha llegado el momento de simular su funcionamiento. Recordemos que el objetivo de este tutorial es únicamente lograr un primer contacto con los comandos de simulación y depuración para afianzar nuestro conocimiento de las técnicas que pone a nuestra disposición Proteus para controlar la simulación y cómo podemos depurar problemas con nuestro código. El control del motor sólo es una excusa para lograr este objetivo y no se trata de aprender cómo se controla un motor.

Como siempre, la simulación se controla utilizando el panel de control de la simulación situado en la zona inferior izquierda de la pantalla principal de Proteus.



El panel contiene los cuatro botones habituales en una simulación: play, saltar un paso, pausa y stop.

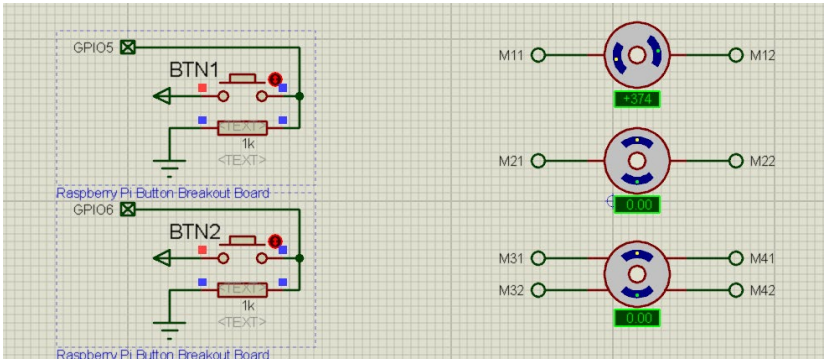
- El botón 'PLAY' arranca la simulación o la vuelve a poner en marcha si previamente la habíamos puesto en pausa.
- El botón 'SALTAR UN PASO' ejecuta la simulación avanzando 'un salto' a través de la ejecución del código de nuestro proyecto y lo pone en pausa. Un paso en el Diseñador Gráfico equivale a la ejecución de un bloque de código.
- El botón 'PAUSA' congela la ejecución de la simulación y nos permite visualizar los valores de las variables, voltajes, corrientes y resto de elementos implicados. Este botón no detiene la simulación al estilo que lo hace el botón stop y nos permite retomarla pulsando los botones play o saltar un paso.
- El botón 'STOP' detiene la simulación completamente y la abandona. Es necesario detener la simulación para poder editar nuestro proyecto (tanto el código como el esquema electrónico).

10.3.3.1.-Ejecución paso a paso básica.

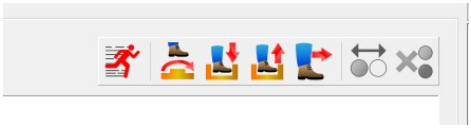
Arrancaremos la simulación simplemente pulsando el botón 'PLAY'. Proteus compila el código si es necesario y entra en el modo simulación. En la barra de estado podemos observar que empieza a contar el tiempo de simulación.

Presionando el pulsador BTN1 haremos que el motor empiece a girar. Dejemos que el motor gane un poco de velocidad y presionemos el pulsador BTN2 para provocar la parada del motor. El motor irá frenándose poco a poco. Si todo funciona bien y observamos este comportamiento del motor, ya podemos detener la simulación utilizando el botón 'STOP'. Si el programa no ha funcionado de esta manera, debe comprobar todos los pasos

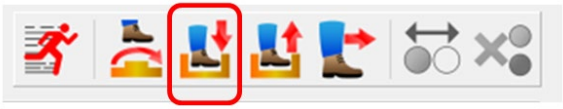
anteriores de este tutorial y asegurarse de que todo se ha hecho según las instrucciones.



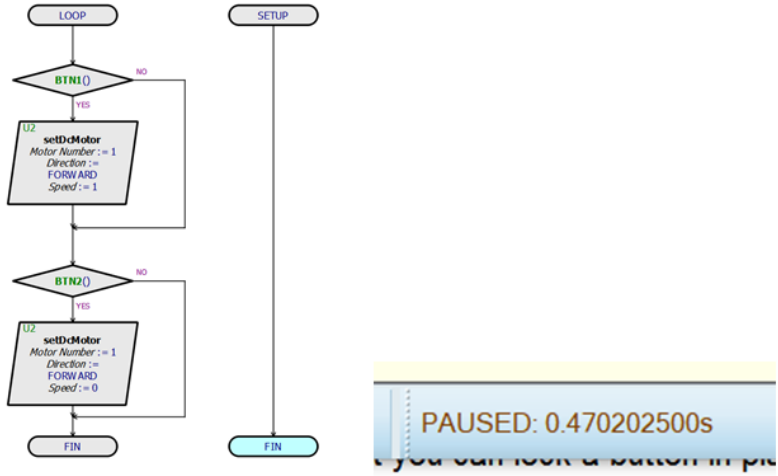
Ahora vamos a volver a arrancar la simulación, pero en esta ocasión utilizando el botón ‘SALTAR UN PASO’. Al hacerlo así, el programa se quedará en pausa ejecutando el primer bloque del bucle de configuración (setup). Podemos observar que en la pestaña diseñador gráfico aparece la barra de herramientas de depuración.



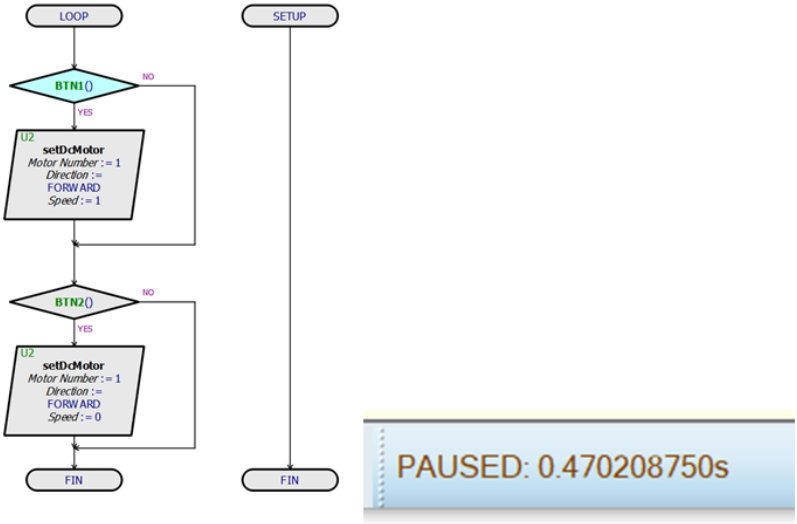
Vamos a utilizar el botón ‘Saltar Dentro’ que fuerza la ejecución de un bloque de nuestro flujograma.



La simulación se llevará a cabo y se detendrá al alcanzar el bloque FIN de subrutina. Podemos observar que está iluminado de color azul y que el tiempo de simulación ha avanzado un poco menos de un segundo.

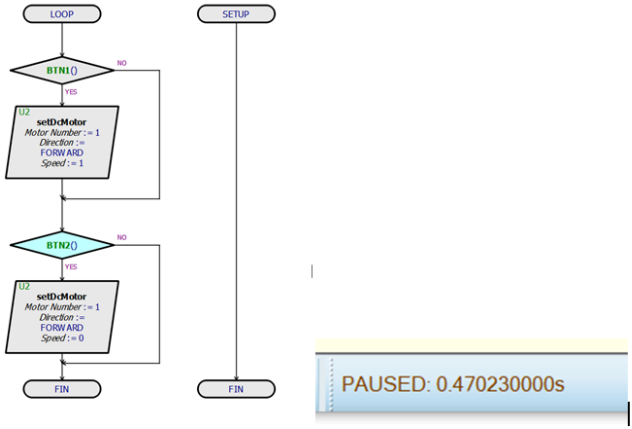


Si volvemos a pulsar el botón ‘Saltar Dentro’ el programa volverá a avanzar de nuevo un bloque. En este caso estará iluminado el bloque de decisión que controla el primer pulsador. Observe que el tiempo de simulación apenas ha avanzado.



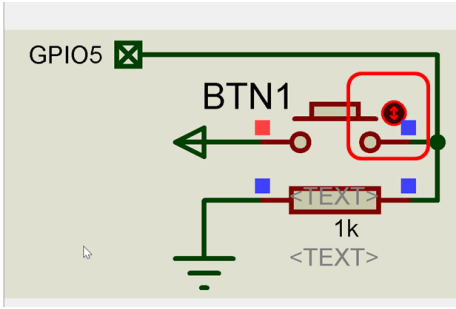
Si continuarámos con la ejecución de nuestras simulación con el pulsador BTN1 actuado el código avanzará por la salida YES del

bloque de decisión. Y si lo hiciéramos con el pulsador BTN1 liberado el código avanzará por la salida NO del bloque de decisión. Puesto que el pulsador BTN1 no está actuado si damos una nueva órden de 'Saltar Dentro' veremos que avanzamos al siguiente bloque de decisión saltando el bloque de control del motor.

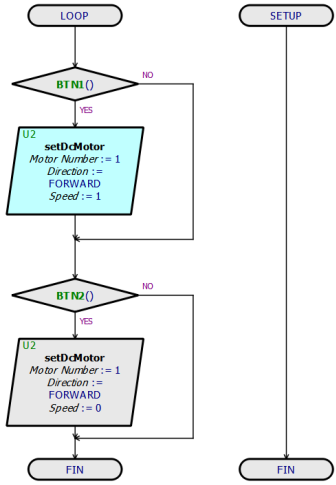


Podemos generar nuevas órdenes 'Salta Dentro' y comprobar cómo se va ejecutando nuestro código sin entrar nunca en los bloques de control del motor.

Con la simulación pausada vamos a utilizar el accionador de animación asosciado al pulsador BTN1 que tenemos que tener disponible en la ventana activa emergente y pulsaremos el botón 'SALTAR UN PASO' del cuadro de mandos de la simulación para que Proteus lea el cambio introducido.



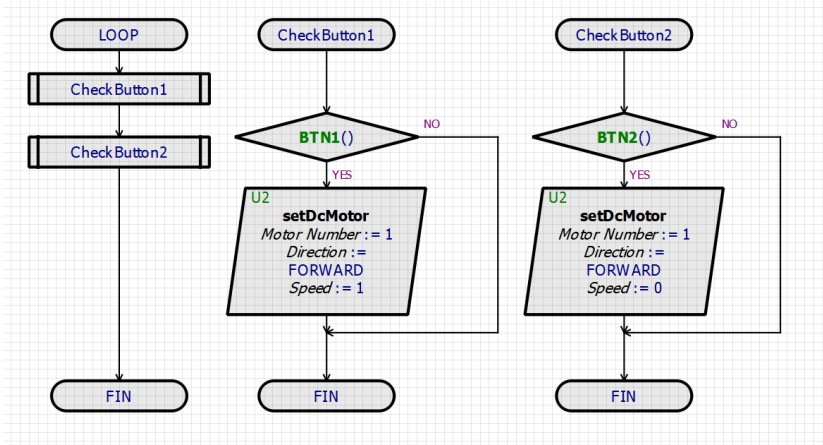
Al acturar tenemos que observar que el pulsador queda accionado, incluso aunque dejemos de actuar con el ratón sobre él. De esta forma podemos retormar la simulación y observar que ahora sí se activa y ejecuta el bloque de control del motor.



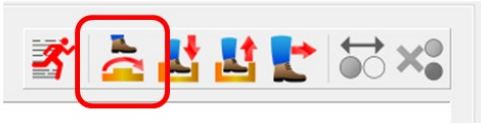
10.3.3.2.-Ejecución paso a paso con subrutinas.

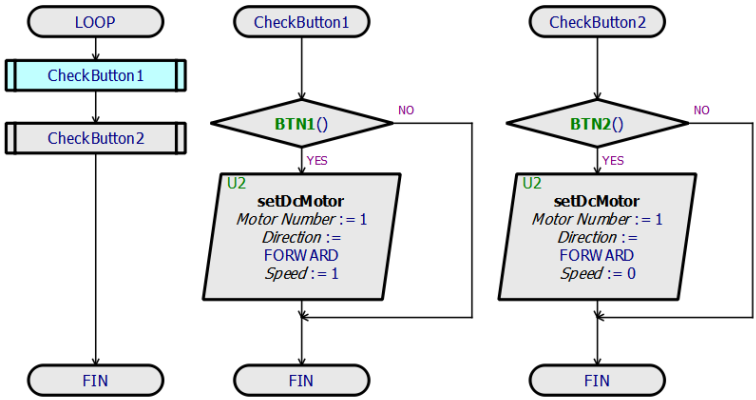
Veamos ahora cómo la ejecución paso a paso y los comandos de depuración trabajan cuando nuestro código contiene subrutinas. Básicamente significa que navegaremos entrando y saliendo de las rutinas utilizando el registro de 'la pila de programa'.

Para poder comprobar cómo funciona necesitamos tener subrutinas en nuestro código. Así que vamos a modificar nuestro flujograma para separar en dos subrutinas diferentes el control de arranque y el control de parada. En el bucle principal (loop) colocaremos los bloques de llamada a cada una de las subrutinas. Nuestro nuevo flujograma debe quedar como el que se muestra en la imagen siguiente.



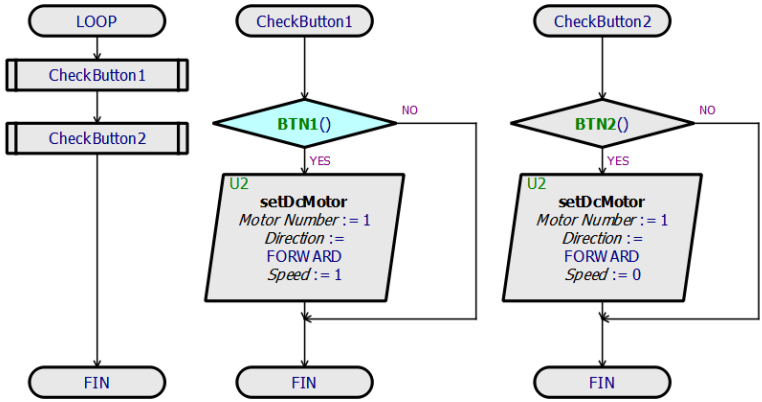
En primer lugar dejemos el pulsador BTN1 activos como vimos en el apartado anterior. Vamos a dar sucesivas órdenes 'Saltar hasta' hasta que logremos el bloque de llamada a la subrutina 'CheckButton1' quede iluminado.



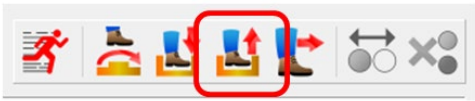


Si utilizamos ejecutamos varias veces la orden ‘Salta hasta’ veremos que cada vez el flujograma se ejecuta hasta alcanzar el siguiente bloque dentro del mismo bucle principal (y no parece pasar por dentro de las subrutinas). Volvamos a dejar iluminado el bloque de llamada a la subrutina ‘CheckButton1’.

Ejecutemos ahora varias veces la orden ‘Saltar dentro’. Veremos que ahora los bloques se van ejecutando de uno en uno incluído los que se encuentran dentro de la subrutina. Como tenemos el pulsador BTN1 accionado vemos que se ilumina el bloque de control de la llamada al motor que lo acelera y sin embargo nunca se ilumina el bloque de control de la llamada al motor que lo frena. Sigamos dando órdenes ‘Saltar dentro’ hasta que se quede iluminado el bloque de decisión del pulsador BTN1.



Ya vimos que si le damos una nueva orden ‘Saltar dentro’ se iluminará el bloque de control de llamada al motor. Sin embargo, probemos a dar la orden ‘Saltar fuera’



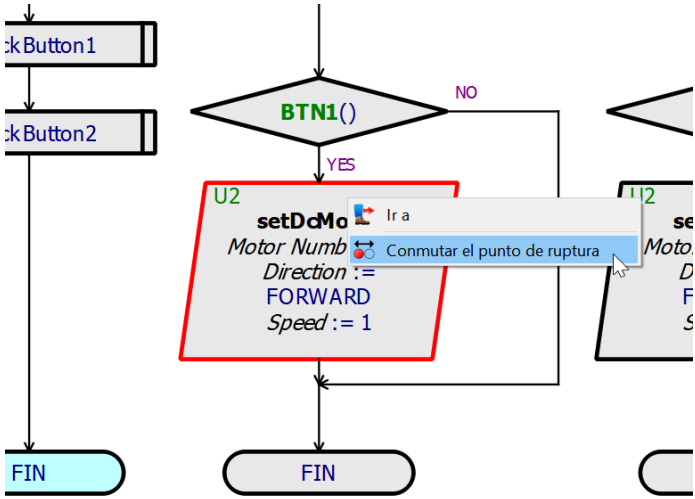
En lugar de iluminarse el bloque de control de llamada al motor, se ilumina el bloque de llamada a la subrutina del pulsador BTN2. La razón es que le estamos ordenando que ejecute todos los pasos

necesarios hasta abandonar la subrutina y volver al proceso padre que le llamó.

Tómese un tiempo para practicar esta técnicas y asegurarse de que entiende bien la diferencia entre los tres tipos de órdenes 'saltar hasta', 'saltar dentro' y 'saltar fuera'.

10.3.3.3.-Puntos de ruptura.

Los puntos de ruptura nos permiten detener la ejecución del código cuando alcanzamos un bloque de código determinado. Para poner o quitar un punto de ruptura, debemos seleccionar el bloque deseado, abrir su menú contextual con el botón derecho del ratón y usar la opción de menú de conmutación.

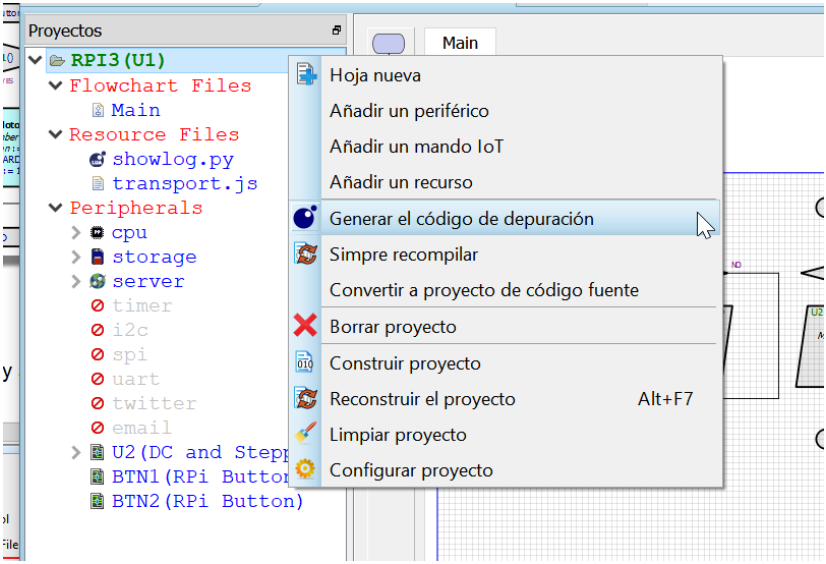


El bloque en cuestión mostrará un punto rojo para indicar que tenemos situado en él un punto de ruptura.

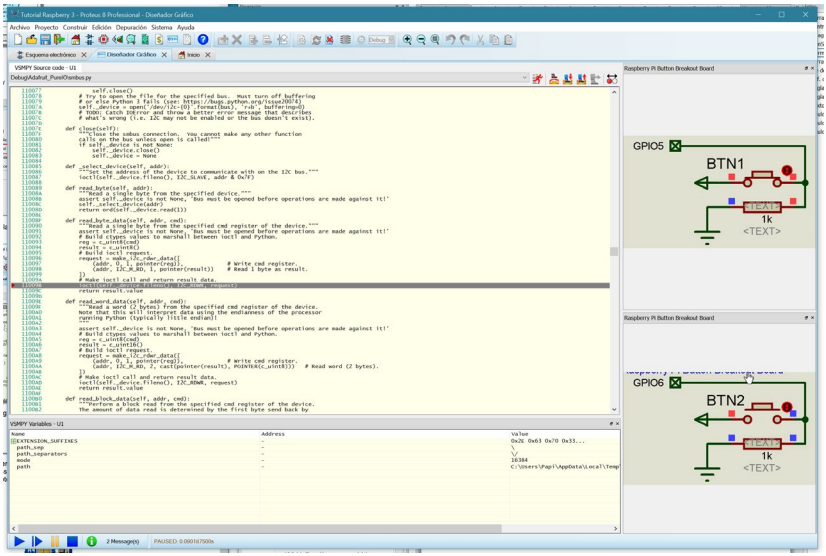
Si ponemos en marcha la simulación del programa con la tecla PLAY se ejecutará hasta que alcance el bloque con un punto de ruptura donde se detendrá. Una vez que el programa entre en pausa podemos volver a utilizar las herramientas que vimos en el punto anterior para ejecutar paso a paso nuestro código o volver a usar la tecla PLAY para que el programa se ejecute normalmente hasta alcanzar un nuevo punto de ruptura.

10.3.3.4.-Depuración del código.

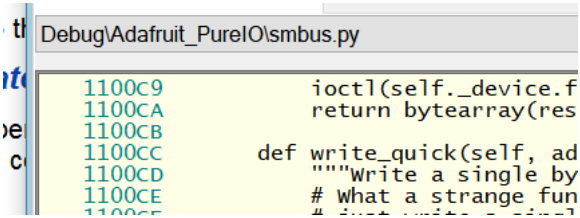
Si queremos observar con mayor profundidad lo que está ocurriendo en nuestro código podemos consultar directamente el código en lenguaje Phyton y no usar el flujograma. Para hacerlo debemos tener la simulación detenida y situar el ratón en la rama principal de nuestro árbol de estructura de proyecto, abrir el menú contextual con el botón derecho del ratón y seleccionar la opción 'generar el código de depuración'.



Ahora si ponemos en marcha la simulación con el botón 'SALTAR UN PASO' ya no podremos ver nuestro flujograma y veremos el código en Phyton. La primera vez que pulsemos el botón 'SALTAR UN PASO' sólo veremos una página en blanco. Es posible que tengamos que dar varias órdenes 'SALTAR UN PASO' para llegar a ejecutar una porción de código que se pueda mostrar.

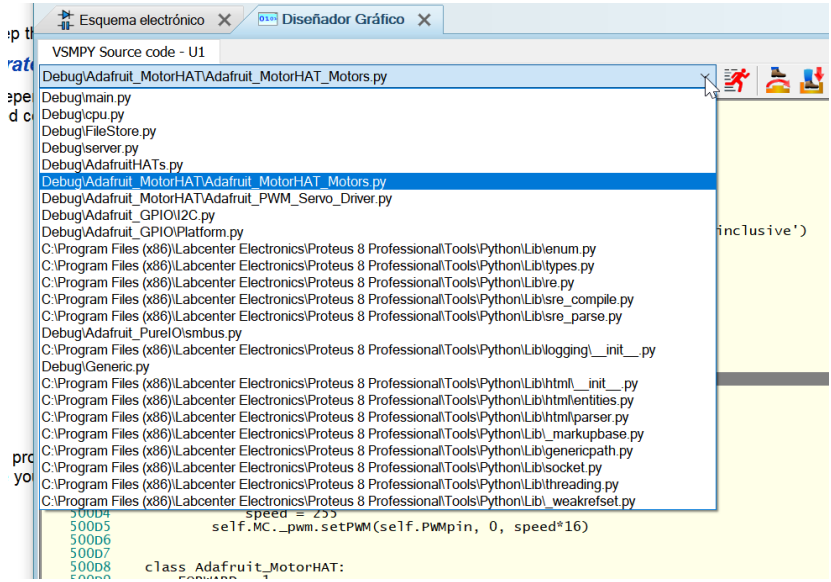


En la zona superior veremos el nombre del archivo Phyton en cuestión que se está visualizando.



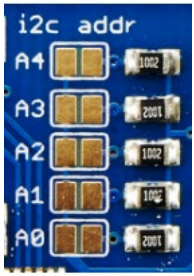
Si ahora usamos la herramienta ‘Saltar hasta’ notaremos que se ejecuta el código hasta la línea siguiente y no hasta el bloque de flujograma siguiente como ocurría cuando visualizábamos el flujograma.

Si usamos el desplegable, podemos ver todos los ficheros utilizados en nuestro proyecto.



10.3.4.-Direccionamiento de las tarjetas de expansión.

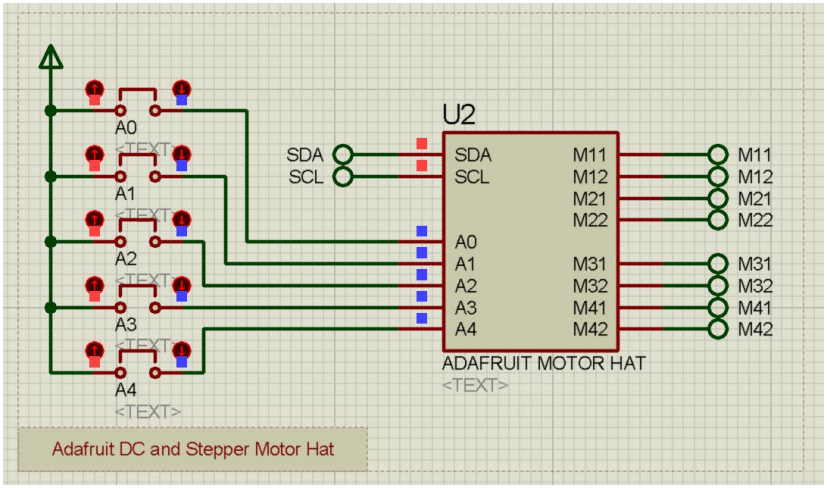
Algunas de las tarjetas de expansión son apilables y pueden utilizarse más de una del mismo tipo al mismo tiempo. Un ejemplo son las tarjetas de expansión de control de motor Adafruit. Cuando utilicemos más de una tarjeta de expansión del mismo tiempo necesitamos cambiar la dirección del bus i2c que usa cada una de ellas para comunicarse con el microprocesador. Normalmente esta operación se lleva a cabo utilizando puentes.



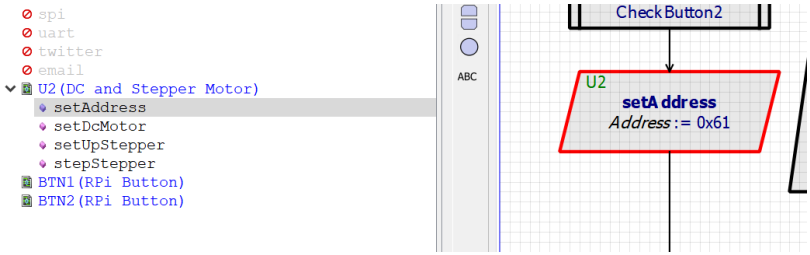
La tarjeta de expansión tiene una dirección base asignada. Por ejemplo para la tarjeta de expansión de control de motores Adafruit la dirección base es 0x60. Los puentes funcionan en código binario.

Así que si cerramos el puente A0 incrementamos en una unidad la dirección base, obteniendo la dirección de tarjeta 0x61. Si cerramos el puente A1 obtendremos la dirección de tarjeta 0x62. Y si cerramos los puentes A0 y A1 al mismo tiempo tenemos la dirección 0x63. Por lo tanto podemos tener hasta 32 direcciones diferentes.

El modelo de simulación de la tarjeta de expansión de control de motor Adafruit también incluye los puentes para que podamos tener y simular varias tarjetas de expansión.



En el árbol de estructura del proyecto, cada tarjeta de expansión de control de motor Adafruit dispone de un método llamado 'setAddrese' que sirve para indicar qué tarjeta vamos a gestionar.



11.-Los robot-tortuga.

11.1.-Definición de robot-tortuga.

Los robot-tortuga son muy populares y trabajar con ellos resulta divertido. Por ese motivo su uso se está extendiendo en los centros educativos y en el mundo de los aficionados. Aunque sus posibilidades son muy amplias, normalmente los proyectos para los que se utilizan caen dentro de una de estas tres categorías:

1. Seguir el trazado de una línea.
2. Evitar obstáculos en el camino.
3. Recorrer un laberinto.

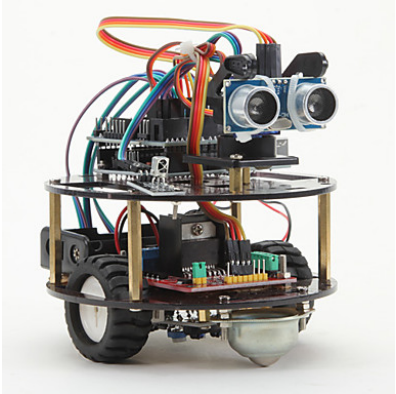
Proteus pone a nuestra disposición un completo modelo eléctrico de un robot-tortuga que incluye motores y sensores gobernados por una placa Arduino. Y nos lo presenta dentro de un mundo de 'realidad virtual' intencionadamente simple pero, a la vez, muy flexible en el que es posible ver la evolución de nuestro robot-tortuga y simular todos sus movimientos.

Controlar nuestro robot-tortuga utilizando el Diseñador Gráfico se simplifica enormemente mediante la utilización de los "métodos de alto nivel" que podemos incorporar dentro del flujograma que contiene la lógica de control. Alternativamente, es posible obtener una mayor flexibilidad y escribir programas tan complejos como deseemos utilizando la programación en el lenguaje C disponible con los equipos Arduino. Y, por supuesto, independiente del modo de programación seleccionado, Proteus pone a nuestra disposición un amplio y potente conjunto de herramientas de depuración y simulación con lo que podemos asegurarnos de que nuestro programa se comportará como realmente deseamos que lo haga, antes de proceder a cargar el código dentro de nuestro robot-tortuga del mundo real.

Actualmente, Proteus incorpora dos modelos completos para la simulación de dos robot-tortuga con amplia presencia en el mercado: Funduino y Zumo.

11.1.1.-El robot-tortuga Funduino.

El robot-tortuga Funduino, es un sencillo robot que utiliza una placa Arduino Uno, una tarjeta que controla el motor que mueve las ruedas y tres sensores del tipo 'busca líneas'. En la zona superior, en la 'cabeza' del robot, se encuentra un sensor de ultrasonidos y un motor paso a paso que permite hacer que la cabeza con los sensores gire a uno y otro lado. Todos estos dispositivos tienen sus correspondientes modelos en Proteus lo que nos permite escribir flujogramas o código en lenguaje C y realizar la simulación completa para comprobar que nuestro programa está correctamente escrito.

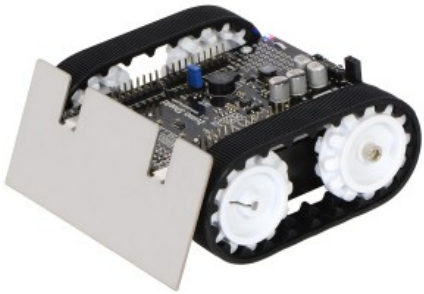


Puesto que el robot-tortuga Funduino ha sido el más barato de su tipo que pudimos encontrar (aproximadamente unos 50€ al momento de escribir esta guía), no debe sorprendernos que tenga algunas limitaciones. En concreto, los motores que gobiernan las ruedas no tienen codificadores rotatorios que nos informen de su posicionamiento exacto y los sensores 'busca líneas' son digitales.

11.1.2.-El robot-tortuga Zumo.

El robot-tortuga Zumo, es una plataforma robotizada que ha sido diseñada para ser controlada utilizando una placa Arduino. Contiene dos micro motores con reductora mediante engranaje metálico con los que se gobiernan las dos bandas de rodadura, una pala de acero inoxidable estilo 'bulldozer', un conjunto de seis sensores de infrarrojos en la zona frontal que posibilitan el seguimiento de líneas o la detección de bordes, un zumbador para la reproducción de sonidos y músicas sencillos, un acelerómetro de 3 ejes, un

magnetómetro y, finalmente, un giroscopio para ayudar en la detección de impactos y como ayuda para la orientación.

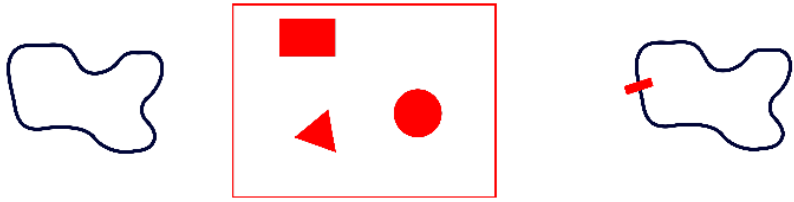


El zumo es un robot-tortuga más avanzado que el Funduino y con él podemos realizar proyectos mucho mejores de robots del tipo 'seguimiento de línea' y 'recorrido de laberintos'. Pero como no dispone del sensor de ultrasonidos es peor para realizar robots capaces de evitar obstáculos presentes en el camino.

11.2.-El entorno virtual: creación de un mapa con obstáculos.

Con el fin de poder comprobar el código de nuestros programas, el robot-tortuga necesita un entorno en el cual pueda operar durante la ejecución de la simulación. Aunque en el mercado existen muchos motores de simulación de entornos físicos muy complejos, la meta de la simulación es comprobar y depurar los códigos y uno de sus objetivos debe ser mantener la simplicidad de uso. Por ese motivo, nuestro mundo virtual es un simple dibujo en MS Paint u otra herramienta similar sobre las que aplicamos una serie de reglas muy sencillas:

- Cada pixel representa un milímetro.
- Un pixel negro debe ser rastreado y seguido.
- Un pixel rojo representa un obstáculo y debe ser evitado.
- Un pixel verde representa un punto de ruptura y deberá poner en pausa nuestro programa.

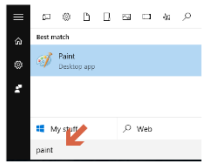


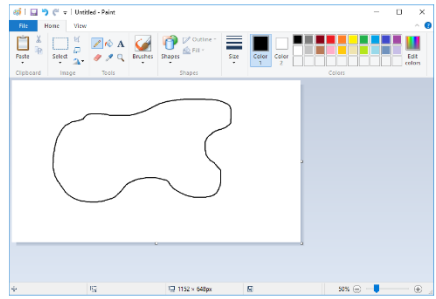
En la imagen se muestran tres ejemplos. De izquierda a derecha, un ejemplo de ruta a seguir por el robot, un ejemplo de obstáculos a evitar y un ejemplo de ruta a seguir con un obstáculo a evitar en medio del camino.

11.2.1.-Creación y utilización de un entorno de simulación.

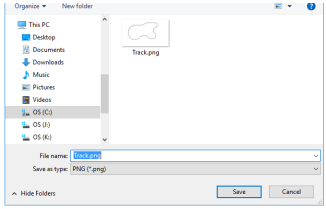
Para crear el entorno de simulación debemos llevar a cabo los siguientes pasos:

1. Dibujar la ruta que deseamos que siga el robot-tortuga con los obstáculos que queremos que aparezcan en el camino. Recordar que la escala del dibujo es 1 pixel equivale a 1mm, así que dibujar una línea de un ancho de 5 pixeles significa que estamos dibujando una línea de 5mm de ancho en la vida real. Es importante tener en cuenta este aspecto, porque el ancho de la línea a seguir afectará al algoritmo utilizado para programar el seguimiento (v.g. comprobar que nunca deben estar todos los sensores al mismo tiempo sobre la línea). Así que es importante dedicar un momento a la reflexión cuidadosa de este punto.

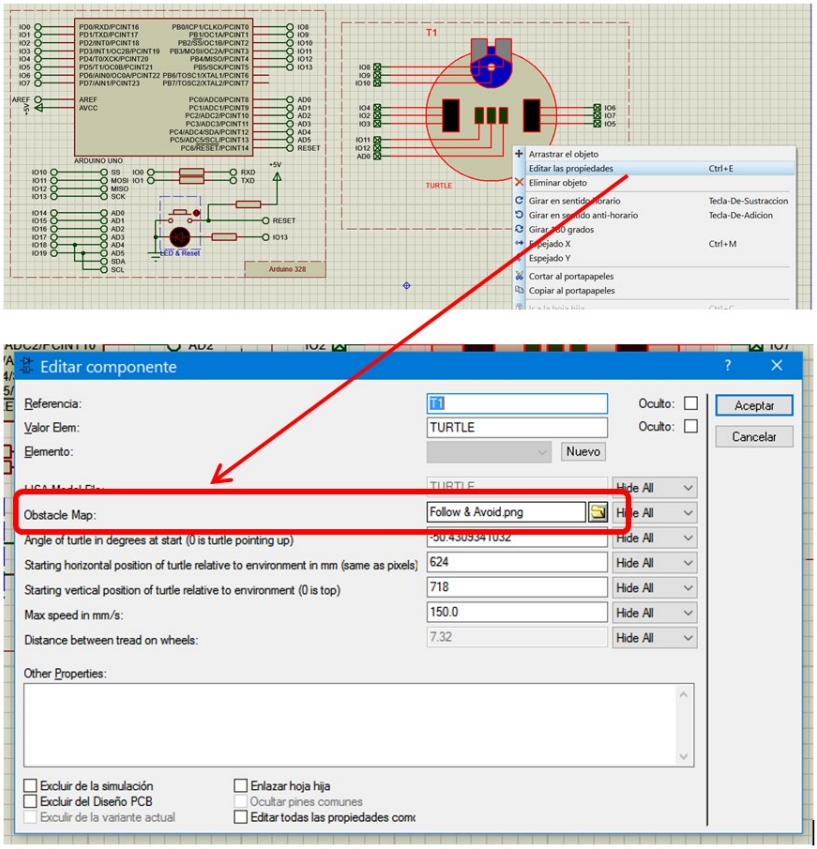




2. Almacenar el gráfico dibujado como un fichero de formato PNG y guardarlo dentro de la carpeta en la que estemos archivando nuestro proyecto Proteus.



3. Editar el componente tortuga dentro del diseño electrónico de nuestro proyecto para especificar el fichero gráfico que hará las veces de entorno de trabajo en el argumento de nombre ‘mapa de obstáculos’.



4. Ejecutar la simulación.

11.2.2.-Colocación del robot–tortuga en el mundo virtual.

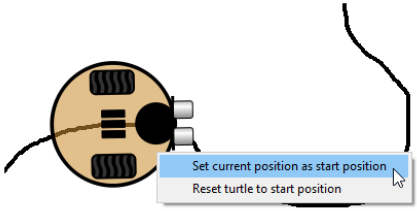
El robot-tortuga puede ser ‘cogido’ y ‘colocado’ en cualquier lugar que deseemos dentro del mapa de obstáculos utilizando el botón izquierdo del ratón.

Es posible rotar el robot-tortuga manteniendo pulsada la tecla CTRL al tiempo que se utiliza el botón izquierdo del ratón.

Podemos seleccionar la posición inicial poniendo la simulación en pausa, colocar el robot-tortuga en el lugar deseado y, por último,

utilizando el botón del ratón para señalar la nueva posición de arranque.





11.3.-Tutorial: un robot-tortuga salva-obstáculos.

11.3.1.-Objetivo del tutorial.

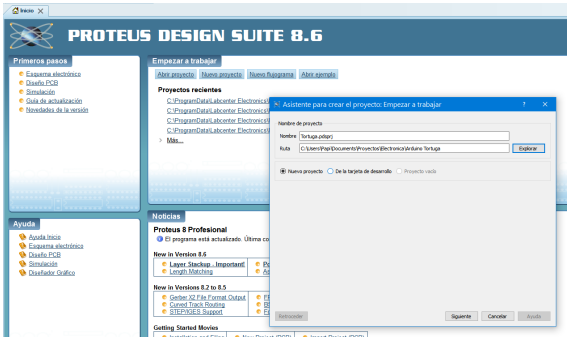
El objeto de este tutorial es mostrar el desarrollo completo de un proyecto construido utilizando el Diseñador Gráfico que utilice un robot-tortuga Funduino. El código programará el funcionamiento de un robot salva-obstáculos. A continuación realizaremos las pruebas de funcionamiento utilizando las herramientas de simulación. Y finalmente cargaremos el programa en un hardware real.

> Aunque hemos realizado el tutorial utilizando un robot-tortuga Funduino, el lector no tendrá dificultad en usar un robot-tortuga Zumo y cambiar el objetivo para realizar un proyecto de un seguidor-de-una-línea. Los principios del desarrollo de un proyecto son iguales en ambos casos.

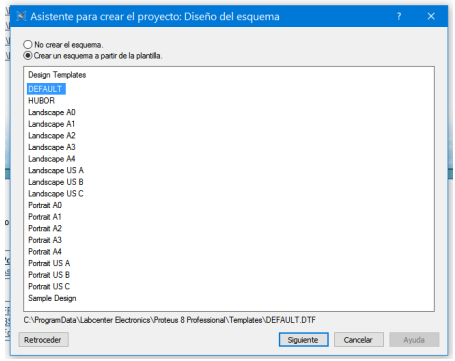
11.3.2.-Configuración del proyecto.

Nuestra primera tarea debe consistir en crear un nuevo proyecto y añadir un robot-tortuga virtual. Para ello seguiremos los siguientes pasos.

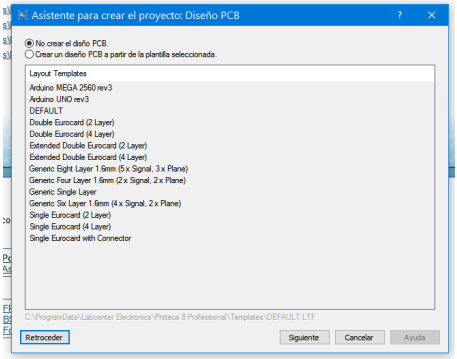
1. Abrimos el asistente para crear un nuevo proyecto desde la página de inicio de Proteus, le asignamos un nombre y definimos la carpeta donde lo queremos crear.



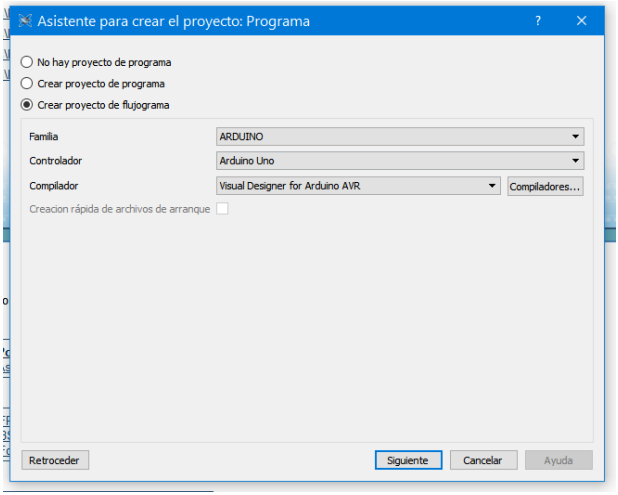
2. Seleccionamos la plantilla que deseemos para nuestro esquema electrónico. Por ejemplo, la que viene por defecto puede ser una buena opción.



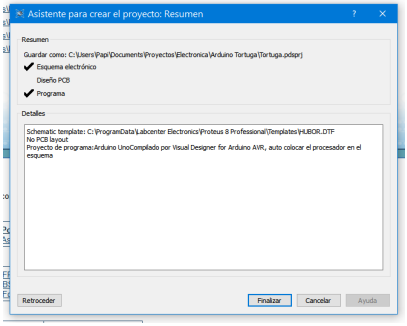
3. Si disponemos de una licencia de Proteus del tipo PCB que nos permite crear una placa de circuito impreso, tendremos que seleccionar la opción ‘No crear el diseño PCB’ en la siguiente ventana de diálogo. Si no disponemos de este tipo de licencia, esta ventana de diálogo no nos aparecerá y pasaremos directamente al paso siguiente.



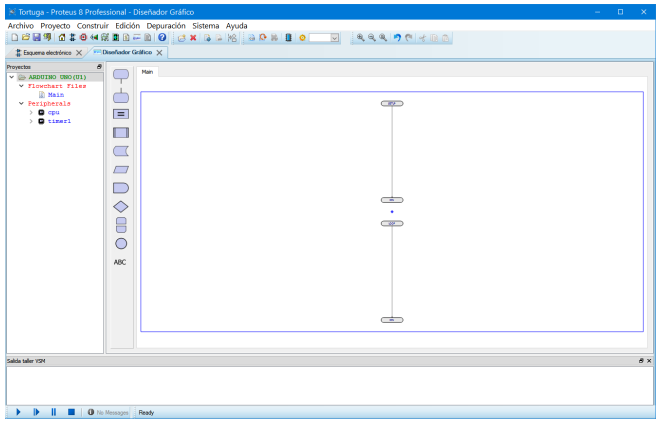
4. Crearemos un proyecto de flujograma, seleccionaremos como familia de controladores la opción ‘Arduino’, como controlador ‘Arduino Uno’ y como compilador ‘Visual Designer for Arduino AVR’.



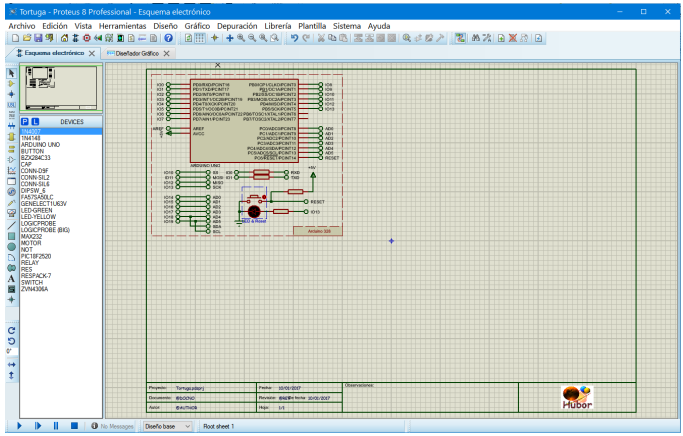
5. El asistente nos ofrecerá un resumen de todas las opciones que hemos elegido y daremos al botón finalizar para completar la operación de creación de nuestro proyecto.



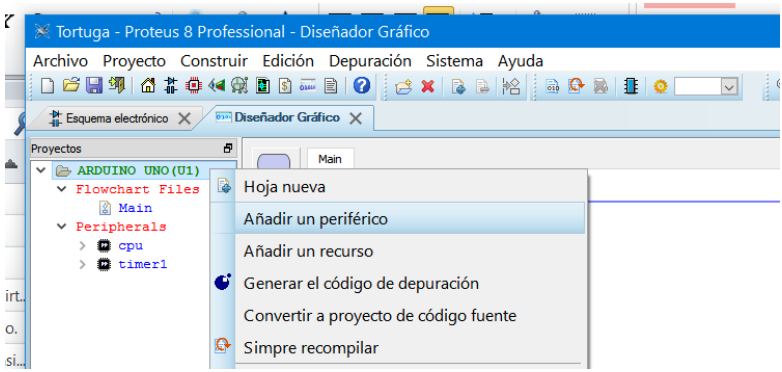
6. Al terminar todo este proceso ya tendremos un proyecto de Proteus. En la pestaña del 'Diseñador Gráfico' podremos encontrar un armazón básico del flujograma con un código para los microprocesadores de la familia Arduino con las subrutinas 'setup' y 'loop'.



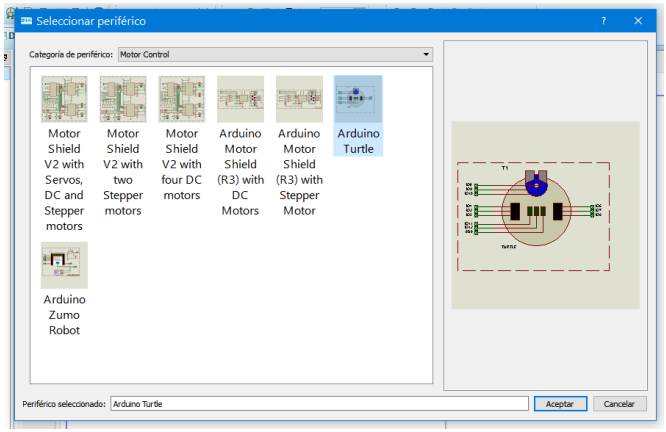
7. En la pestaña ‘Esquema electrónico’ sólo aparecerá una placa Arduino Uno colocada.



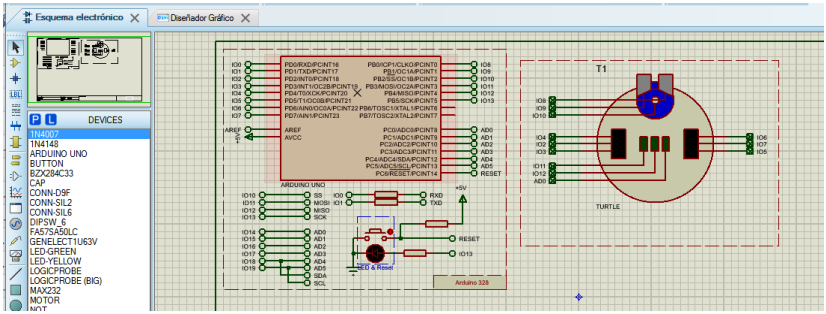
8. Ahora ya podemos añadir nuestro robot-tortuga a nuestro proyecto. Para hacerlo iremos a la pestaña del diseñador gráfico, apuntaremos con el cursor del ratón sobre el proyecto ‘Arduino Uno (U1)’ y abriremos el menú contextual con el botón derecho. En el menú contextual que aparece, seleccionaremos la opción ‘Añadir periférico’.

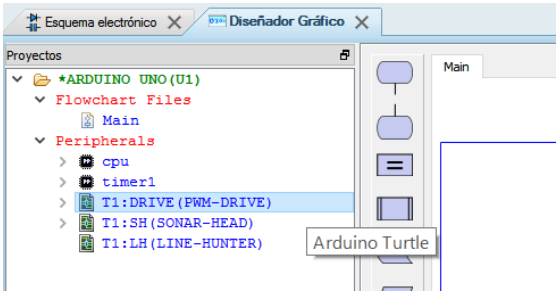


9. Como categoría de periférico seleccionamos ‘Motor control’ y como periférico el ‘Arduino Turtle’.

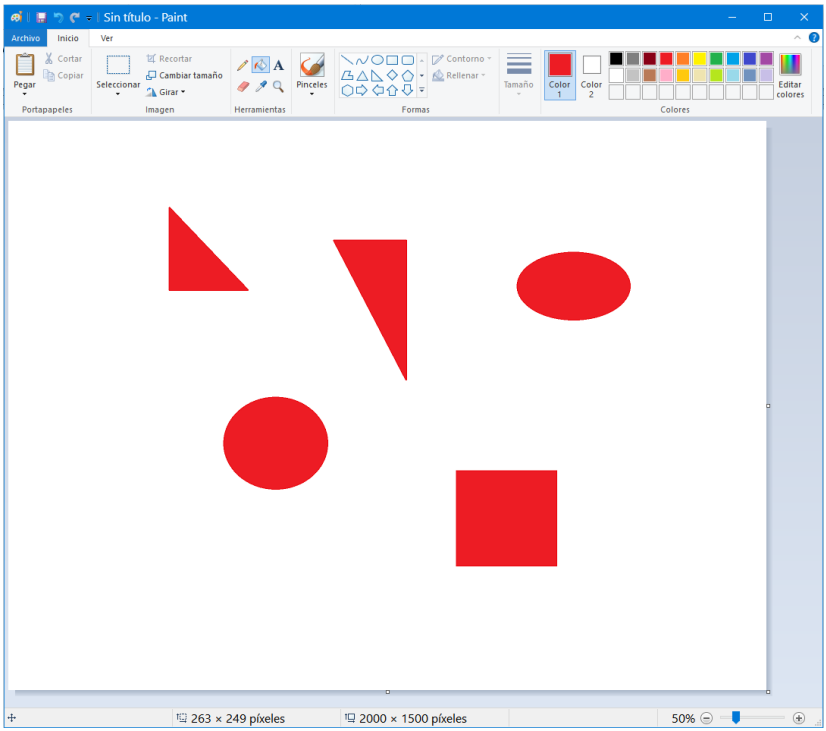


10. En la pestaña ‘Diseño electrónico’ ya podremos ver el módulo periférico de nuestro robot-tortuga (T1) y en la pestaña ‘Diseñador Gráfico’ ya veremos las funciones asociados con él (todos empiezan por T1) para gestionar un motor pwm, un sensor de ultrasonidos y un sensor del tipo seguidor de líneas.

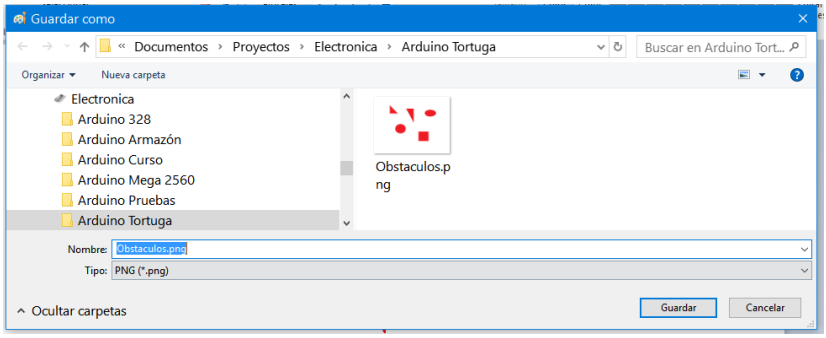




11. A continuación necesitamos crear el mapa con los obstáculos que formará nuestro entorno virtual para simular sobre él el funcionamiento de nuestro robot-tortuga. Para llevarlo a cabo, abriremos la herramienta MS Paint que podemos encontrar en el menú de inicio de Windows, dentro de la carpeta 'Accesorios de Windows'. Crearemos un dibujo donde situaremos una serie de figuras geométricas de color rojo para simular los obstáculos. Las dos reglas básicas que tenemos que tener en cuenta para crearlo es que cualquier objeto de color rojo se considerará un obstáculo y que cada pixel del dibujo representa un milímetro en la vida real. Así que una buena idea es definir un dibujo de 2.000 x 1.500 pixels que representa una habitación de 2 x 1,5 metros del mundo real que es perfecta para simular un campo de juego. A continuación dibujaremos una serie de figuras geométricas de color rojo para indicar cuáles son los obstáculos.

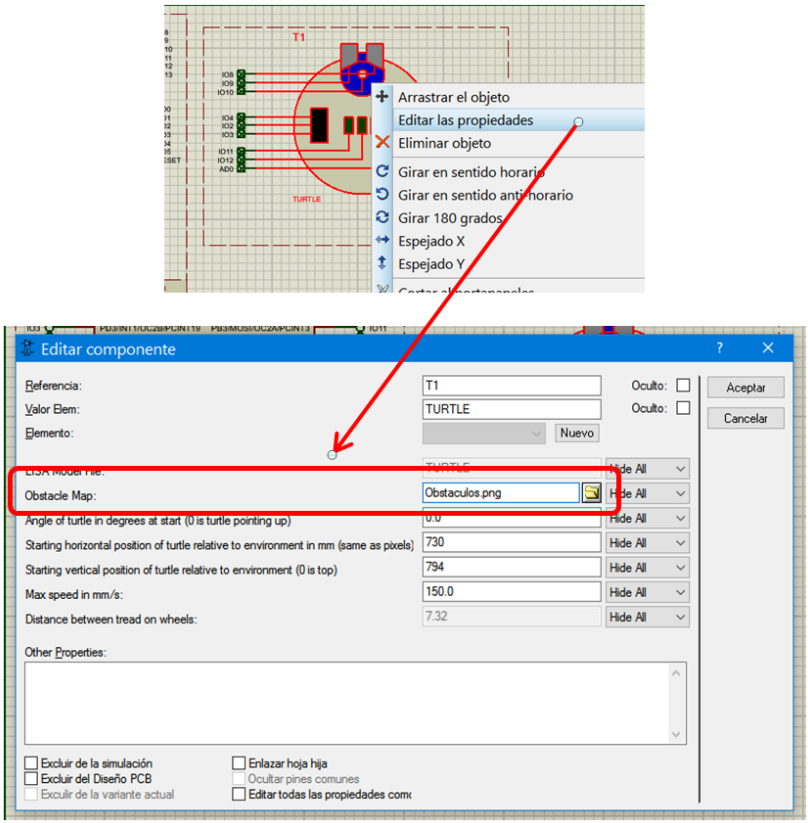


12. Una vez dibujado, debemos guardar nuestro 'entorno virtual' como un fichero del tipo PNG. Le daremos el nombre 'obstaculos.png' y lo archivaremos en la carpeta dónde se encuentra nuestro proyecto de Proteus.



Si estuviéramos haciendo un robot del tipo ‘seguidor de líneas’ tendríamos que utilizar un dibujo que contenga una línea negra que indicará cuál es la ruta que deseamos que siga nuestro robot-tortuga. En este caso, el marco del dibujo no debe tener un tamaño fijo, sino que debe ser lo suficientemente grande para contener toda la ruta dibujada pero sin ser excesivamente mayor, puesto que el ‘mundo virtual’, en este caso, no necesita ser mucho mayor que el recorrido que debe seguir nuestro robot-tortuga.

13. Para terminar, debemos indicar a Proteus cuál es el mapa de obstáculos que vamos a utilizar en nuestra simulación con el robot-tortuga. Para hacerlo, seleccionamos en la pestaña ‘esquema electrónico’ nuestro robot-tortuga y en el menú contextual seleccionamos el fichero del dibujo Obstáculos.png que acabamos de crear.



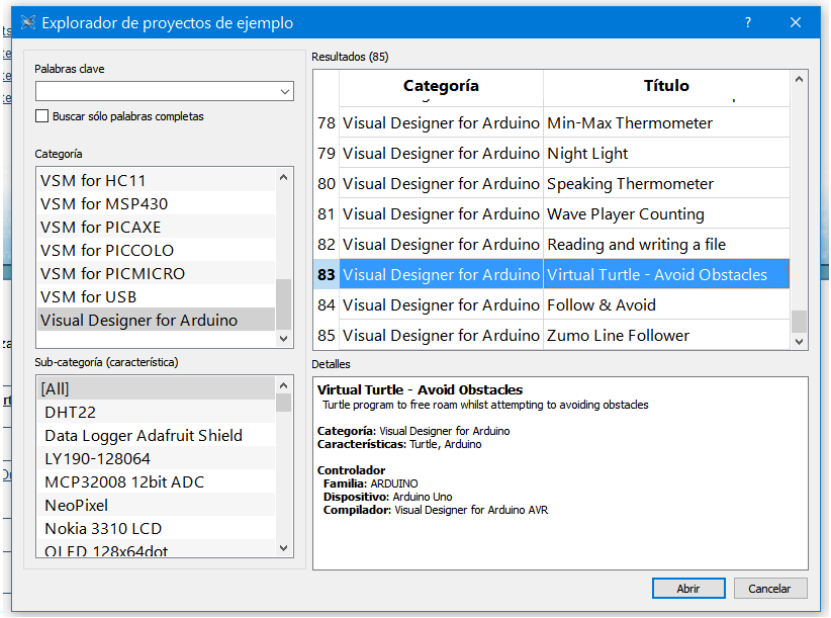
11.3.3.-Diseño del flujograma.

Cuando hemos añadido un robot-tortuga Funduino a nuestro proyecto automáticamente Proteus ha puesto a nuestra disposición una serie de métodos de alto nivel que podemos utilizar en el flujograma para controlar el funcionamiento de nuestro robot-tortuga. Estos métodos serán vistos con posterioridad en este mismo documento con todo detalle en el epígrafe 9.5 dedicado en exclusividad a este fin. Ese epígrafe es una potente herramienta de ayuda que debe servir como referencia en nuestro trabajo de programación.

En este punto, asumimos que el lector ya está familiarizado a trabajar con los flujogramas y el Diseñador Gráfico, puesto que en

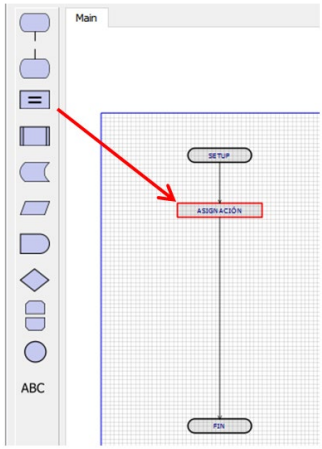
los primeros epígrafes de este documento, ya hemos desarrollado estos conceptos con detalle.

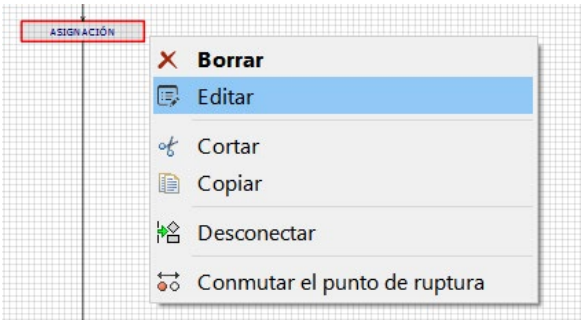
> El lector puede encontrar entre los proyectos de ejemplo suministrados con Proteus un robot-tortuga 'salva obstáculos' completo. Si ya se encuentra familiarizado con las técnicas de programación usando el Diseñador Gráfico puede saltar hasta la sección dedicada a la simulación del proyecto.

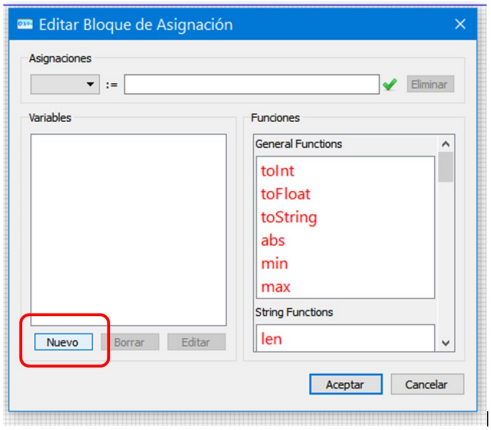


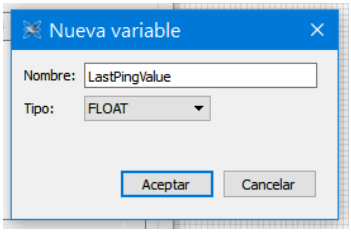
El algoritmo básico de nuestro código consiste en enviar constantemente una ráfaga de ondas de sonar (utilizando el método 'ping'), compararlo con los valores recibidos con anterioridad (para comprobar si nos encontramos atascados) y, cuando no nos encontramos atascados, medir la distancia al posible obstáculo que se encuentre enfrente del robot-tortuga (para comprobar si es necesario girar nuestro robot-tortuga en este momento para evitar el obstáculo).

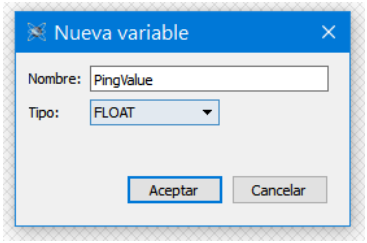
Nuestro primer paso consistirá en crear algunas variables de trabajo que vamos a utilizar en nuestro código e inicializarlas. Esto lo llevaremos a cabo dentro de la subrutina ‘setup’.

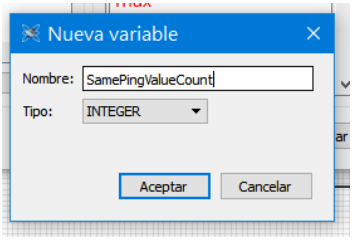


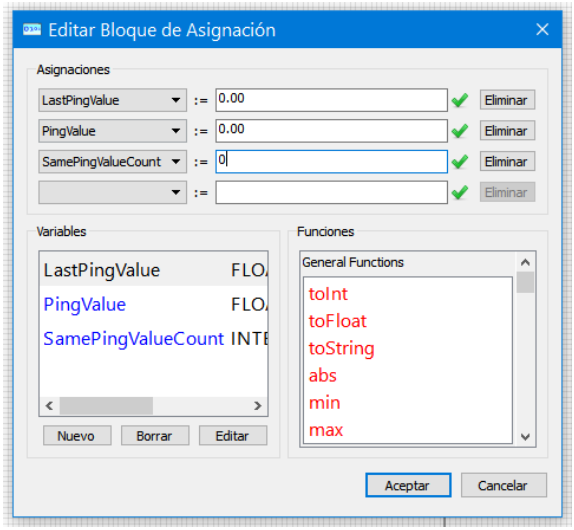


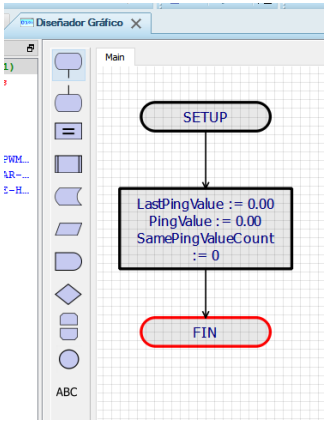

Editar Bloque de Asignación
Asignaciones
:=
Eliminar
Variables
Funciones
General Functions
toInt
toFloat
toString
abs
min
max
String Functions
len
Nuevo
Borrar
Editar
Aceptar
Cancelar


Nueva variable
Nombre:
LastPingValue
Tipo:
FLOAT
Aceptar
Cancelar


Nueva variable
Nombre:
PingValue
Tipo:
FLOAT
Aceptar
Cancelar

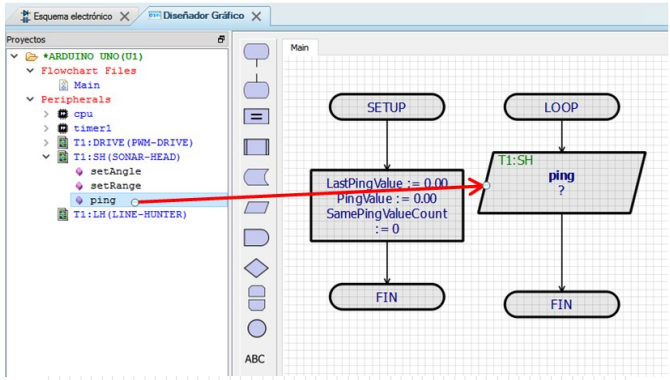

Nueva variable
Nombre:
SamePingValueCount
Tipo:
INTEGER
Aceptar
Cancelar

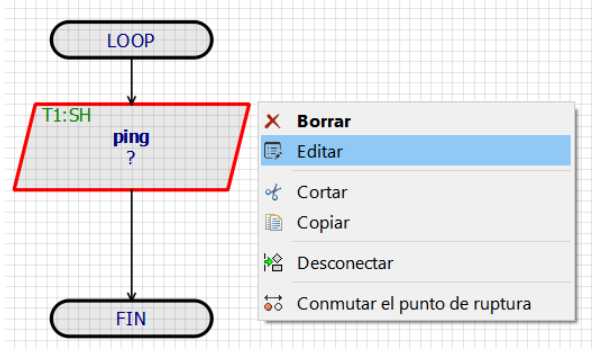


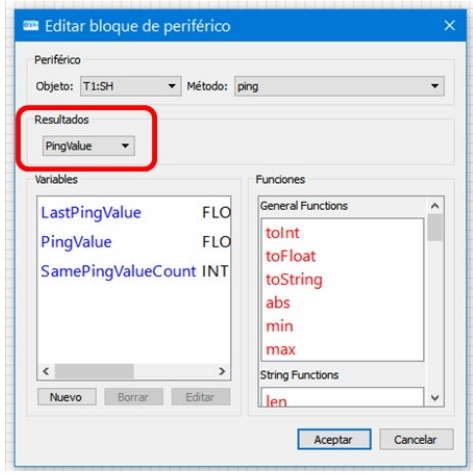


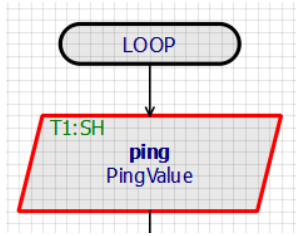
En el proceso tenemos que crear tres variables: LastPingValue de tipo float, PingValue de tipo float y SamePingValueCount de tipo integer.

En la zona superior del bucle ‘loop’ queremos colocar el método ‘ping’ asociado al sensor de ultrasonidos. Para hacerlo ‘cogemos y arrastramos’ el método desde el árbol de estructura del proyecto y asignamos el resultado de ejecutar el método a la variable de nombre ‘PingValue’.

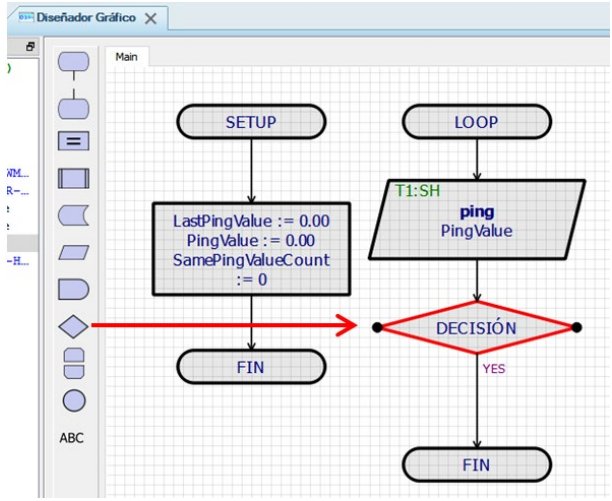

Esquema electrónico
Diseñador Gráfico
Proyectos
*ARDUINO UNO(U1)
Flowchart Files
Main
Peripherals
cpu
timer1
T1:DRIVE(PWM-DRIVE)
T1:SH(SONAR-HEAD)
setAngle
setRange
ping
T1:LH(LINE-HUNTER)
Main
SETUP
LOOP
LastPingValue := 0.00
PingValue := 0.00
SamePingValueCount := 0
T1:SH
ping
?
FIN
FIN
ABC

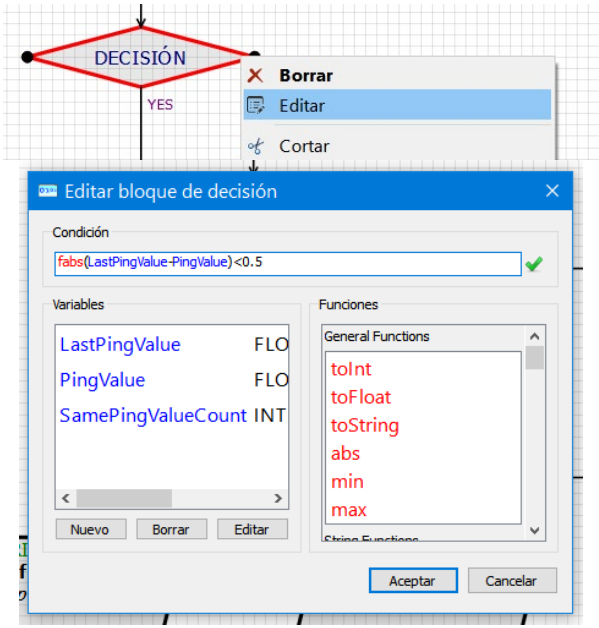

LOOP
T1:SH
ping
?
FIN
Borrar
Editar
Cortar
Copiar
Desconectar
Conmutar el punto de ruptura

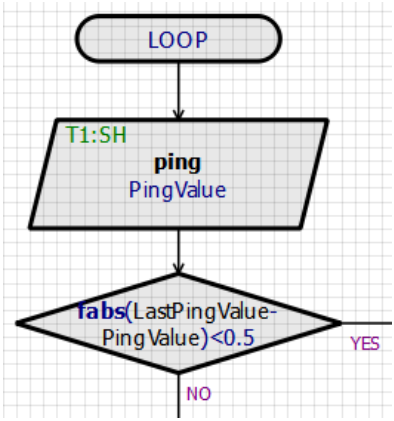

Editar bloque de periférico
Periférico
Objeto: T1:SH
Método: ping
Resultados
PingValue
Variables
LastPingValue FLO
PingValue FLO
SamePingValueCount INT
Nuevo
Borrar
Editar
Funciones
General Functions
toInt
toFloat
toString
abs
min
max
String Functions
len
Aceptar
Cancelar



El resultado de ejecutar este método que acabamos de crear es la base de nuestra toma de decisiones. Nuestra primera comprobación consiste en ver si el valor recibido es aproximadamente el mismo que el recibido en la última lectura realizada. En caso de obtener una respuesta afirmativa, significa que nos encontramos a la misma distancia del obstáculo. Para realizar esta comprobación necesitamos utilizar un bloque del tipo decisión, restar el valor de lectura actual al valor de la lectura anterior y, si la diferencia es menor de 0,5, significará que la distancia al obstáculo no ha variado. Es importante tener en cuenta que al usar la función fabs() incluida en la librería de funciones matemáticas devuelve un valor de tipo float. Llevemos a cabo esta tarea en nuestro flujograma.

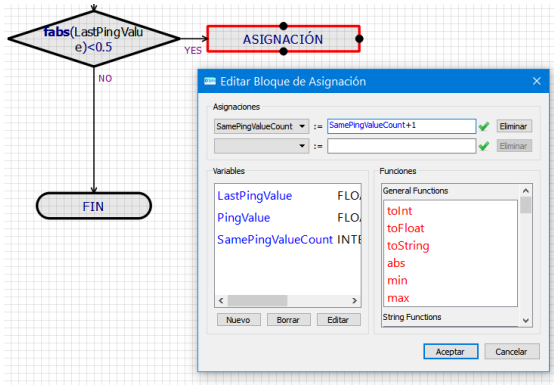


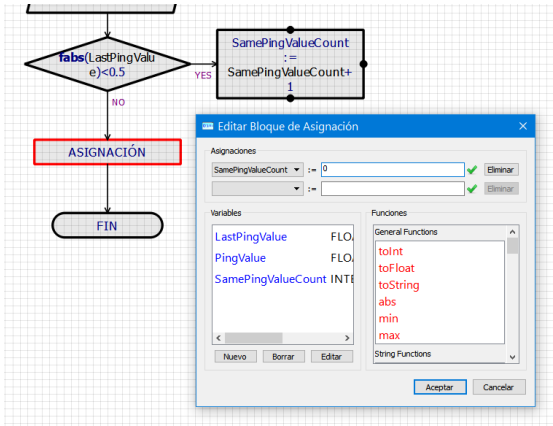


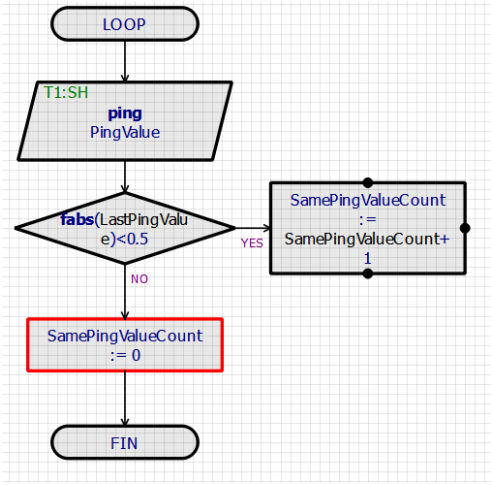


Con este bloque bifurcamos el flujo de ejecución de nuestro flujograma en dos ramas, una con el código que se ejecutará cuando la distancia medida al obstáculo sea la misma y otra en caso contrario.

Si la respuesta al bloque de decisión es afirmativa, con un bloque de asignación incrementamos el valor de nuestra variable ‘SamePingValueCount’. Si la respuesta es negativa reiniciamos el valor de esta variable a cero con otro bloque de asignación.

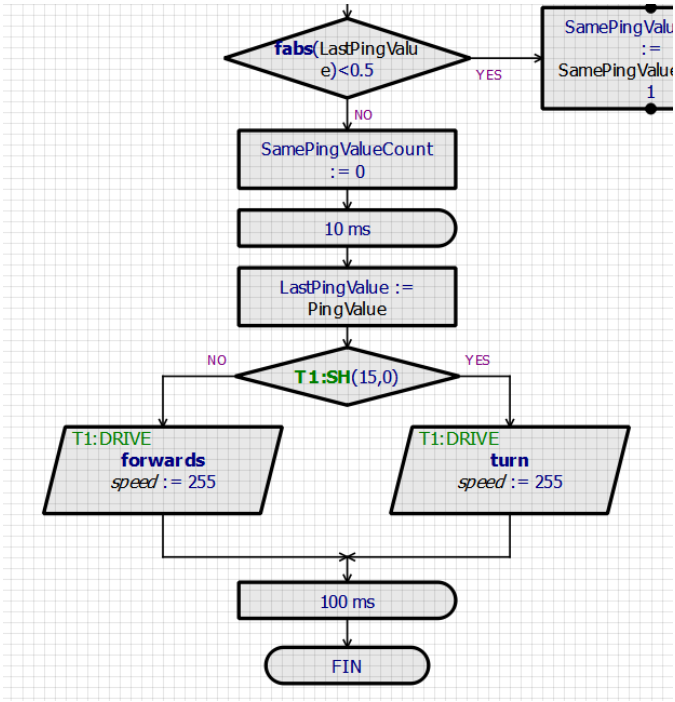






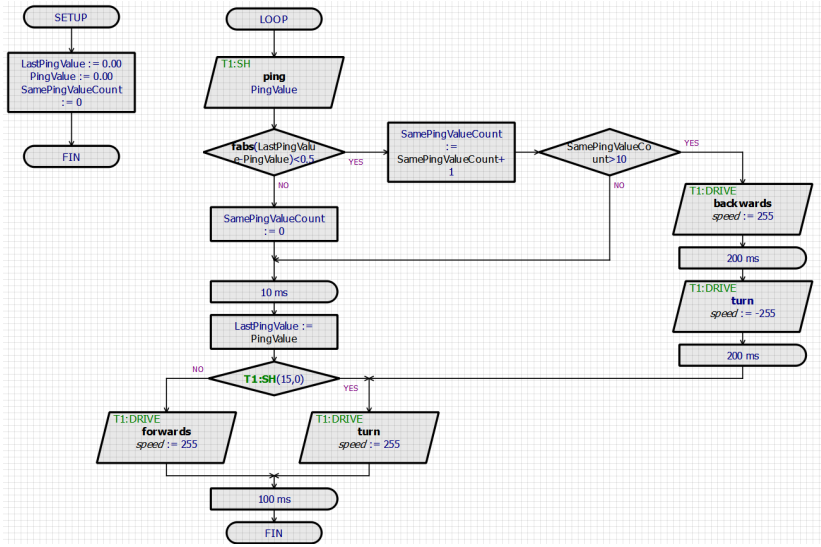
En el caso de que no nos encontremos atascados nuestro trabajo es muy simple. Sin entrar en detalles, después de asignar a la variable 'SamePingValueCount' el valor de cero, tenemos que comprobar si nos encontramos a menos de 15cm. del obstáculo. En caso afirmativo debemos girar a la derecha y en caso contrario debemos seguir hacia adelante. Debemos añadir una pequeña pausa al final, para permitir que nuestro robot-tortuga avance un poco antes de realizar una nueva iteración del bucle.

Al final, esta rama de nuestro flujograma debe quedar cómo se muestra en la figura siguiente.

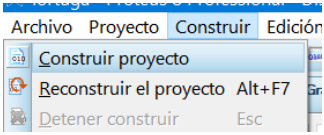


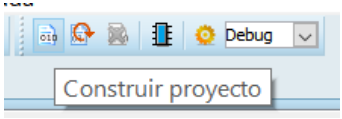
En el caso de que si estuviéramos atascados, necesitamos, después de incrementar la variable 'SamePingValueCount' en una unidad, comprobar si nos encontramos a la misma distancia del obstáculo en repetidas ocasiones. En caso afirmativo, tendremos que retroceder un poco y girar a la derecha un poco antes de retomar la ejecución de nuevo al final del bucle principal de nuestro programa.

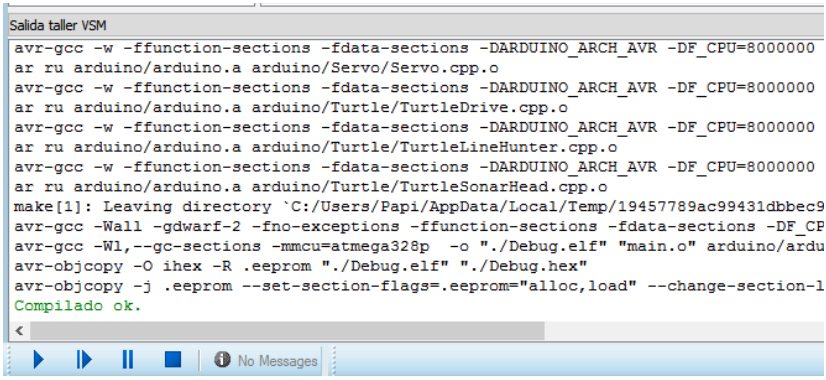
Al final, con esta segunda rama, nuestro flujograma debe quedar cómo se muestra en la figura siguiente.



Y, con esto, la escritura de nuestro código ya está completa. Ya podemos compilarlo y comprobar que no se produce ningún error. Para hacerlo usaremos la opción 'Construir proyecto' del menú 'Proyecto' o el icono equivalente situado en la barra de herramientas.



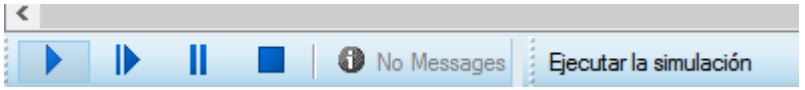




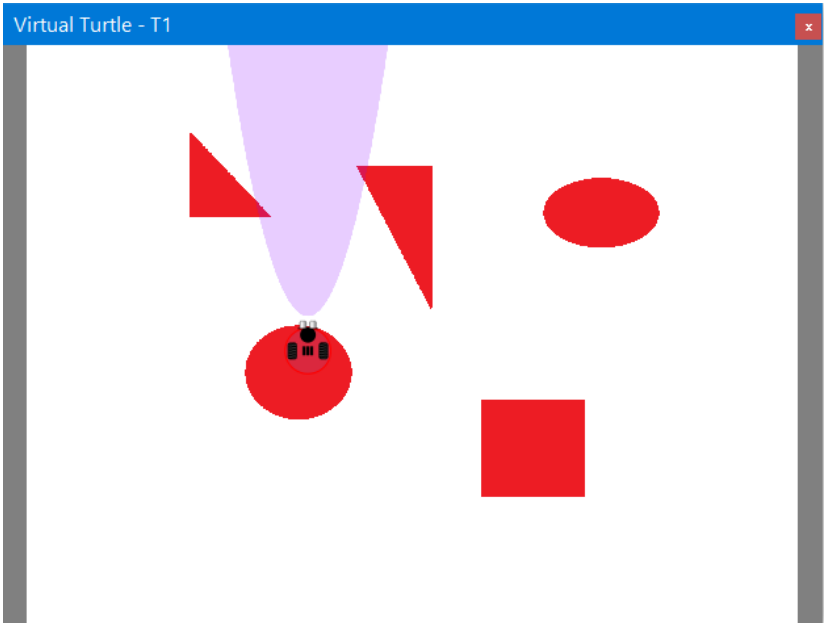
Si todo ha ido bien, tenemos que poder ver el mensaje ‘Compilado ok’ en la ventana de salida taller VSM.

11.3.4.-Simulación y depuración.

Para poner en marcha la simulación utilizaremos el botón ‘Play’ del cuadro de mandos.



l hacerlo, aparecerá una ventana emergente donde se muestra el mapa de obstáculos que hemos creado con anterioridad (en el epígrafe 9.2) con un pequeño robot-tortuga situado dentro del área de trabajo. El cono de color púrpura muestra el rango de alcance del sensor de ultrasonidos.



Podemos pinchar sobre el robot-tortuga y desplazarlo a cualquier lugar del área de trabajo. El robot se pondrá en marcha y su comportamiento vendrá determinado por el código que acabamos de escribir.

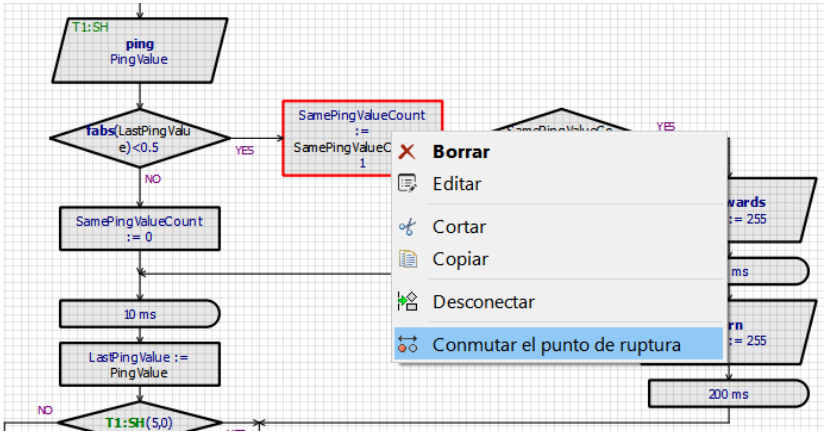
Podemos probar a detener la simulación y cambiar la distancia de comprobación desde los 15cm. anteriores a 5cm. y volver a poner de nuevo la simulación en marcha. Veremos que el robot-tortuga ahora se acerca más a los objetos que hacen la función de obstáculo antes de girar.

Por supuesto, podemos poner la simulación en pausa en cualquier momento con el botón correspondiente del cuadro de mandos.

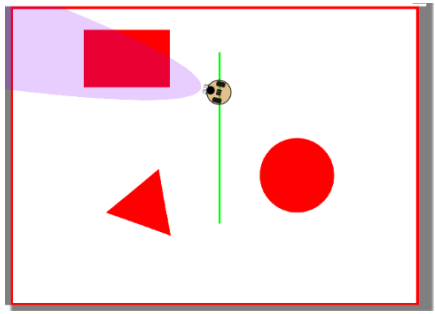
11.3.4.1.-Utilización de los puntos de ruptura.

Si necesitamos utilizar puntos de ruptura para facilitar la depuración de nuestro código podemos llevarlo a cabo de diferentes maneras:

1. Utilizando el botón derecho del ratón para abrir el menú contextual de cualquiera de los bloques del flujograma.



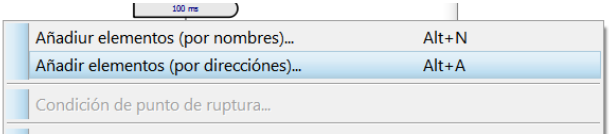
2. Dibujando un objeto de color verde en el mapa de obstáculos.

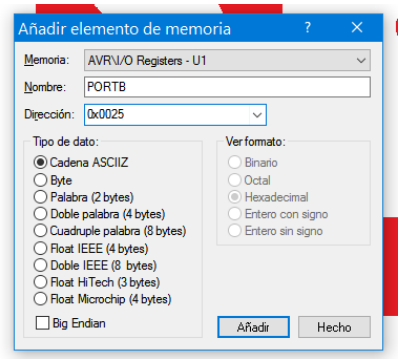


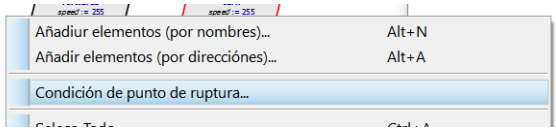
3. Estableciendo una determinada condición utilizando una ventana de inspección con la opción ‘Watch windows’ del menú ‘depuración’.

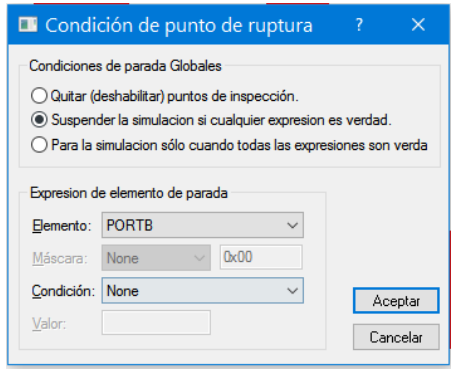


En la ventana de inspección, con el botón derecho del ratón abrimos el menú contextual. Primero añadimos el elemento que queremos usar para el punto de ruptura y luego creamos el propio punto de ruptura.







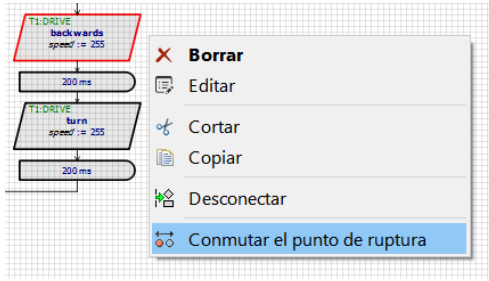


Independientemente de cuál es la condición que provocará la ejecución del punto de ruptura, lo importante es que nosotros podemos con ayuda de esta técnica investigar el comportamiento de nuestro robot-tortuga en un determinado punto. Gracias a las facilidades que nos ofrece Proteus disponemos de una extraordinaria ventaja para llevar a cabo la simulación de nuestro robot-tortuga enteramente bajo nuestro control. Por ejemplo, podemos saltar a lo largo del código y nuestro robot-tortuga se comportará exactamente como lo haría en ese punto en la vida real (los motores no pierden el momento de fuerza, por ejemplo).

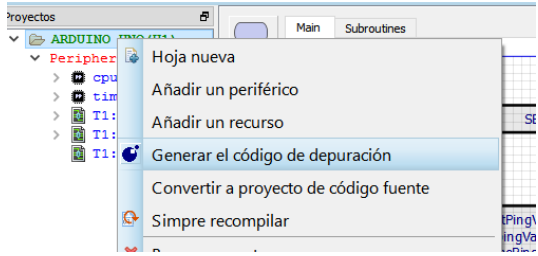
11.3.4.2.-Ejecución ‘paso a paso’.

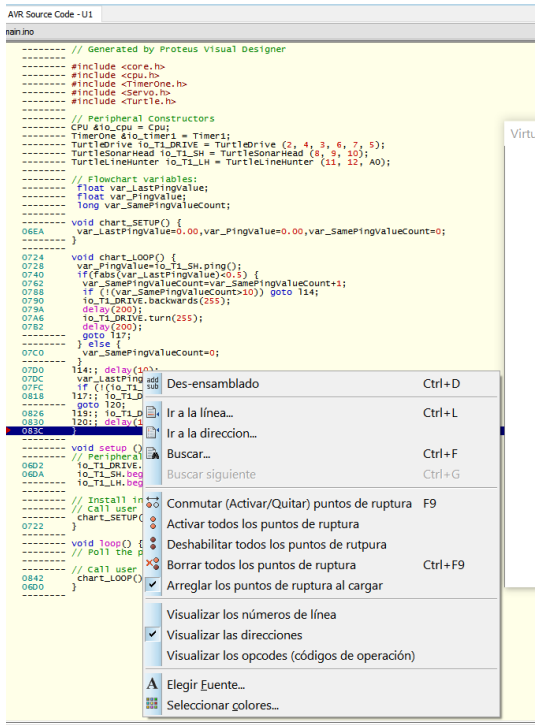
Proteus nos permite ejecutar el código ‘saltando’ a lo largo del flujograma de nuestro proyecto de diferentes formas:

1. Desde el flujograma utilizando el botón derecho del ratón para abrir el menú contextual de cualquier bloque y crear un punto de ruptura.



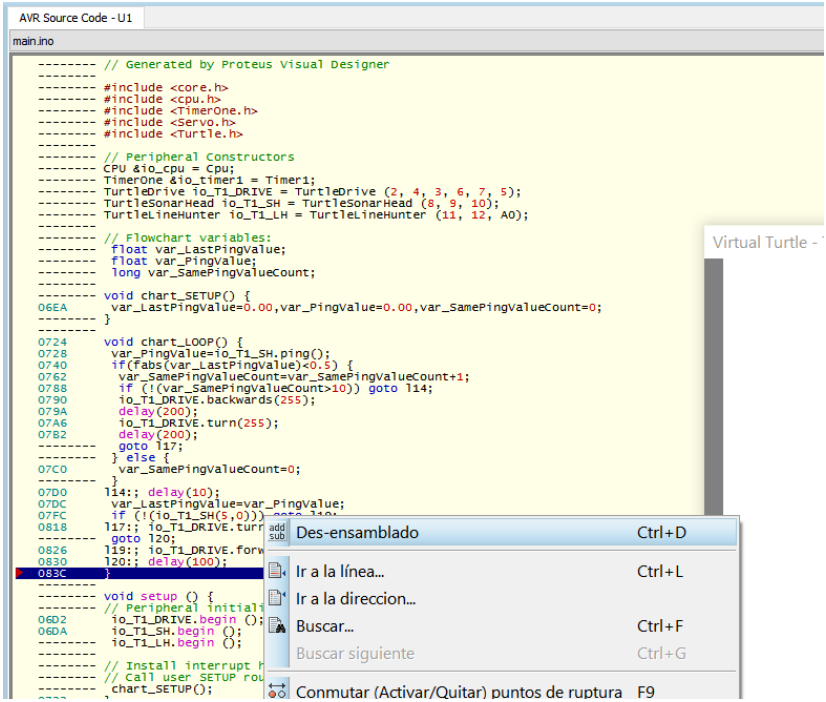
2. Desde el código fuente, deteniendo la simulación y desde el árbol de estructura del proyecto con ayuda del menú contextual generar el código de depuración. A continuación arrancar la simulación y utilizar la ventana de código fuente para incluir un punto de ruptura en el punto deseado.



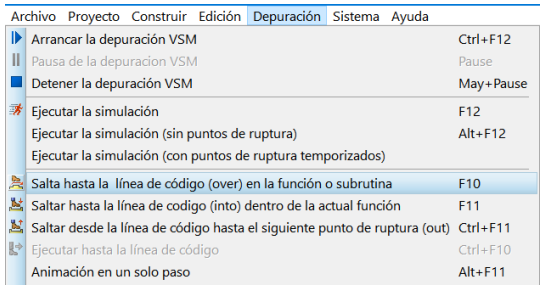


⚠ En este punto debemos asegurarnos de que usamos la opción ‘generar el código de depuración’ y no confundirse con la opción ‘convertir a proyecto de código fuente’. Esta segunda opción del menú contextual convierte un proyecto de flujograma en uno de código fuente en lenguaje C y ésta opción una vez ejecutada no puede ser revertida.

3. Desde el código en ensamblador. Para visualizarlo debemos seleccionar primero el comando ‘Des-ensamblado’ del menú contextual desde la ventana de código fuente.



Una vez hecho podemos utilizar todos los comandos disponibles de ejecución desde el menú depuración.



11.4.-Carga del programa en un robot-tortuga real.

Proteus ofrece el soporte para utilizar el programador AVRDUDE y poder de esta forma cargar nuestro código directamente en un robot-tortuga del mundo real desde el propio entorno de trabajo de Proteus.

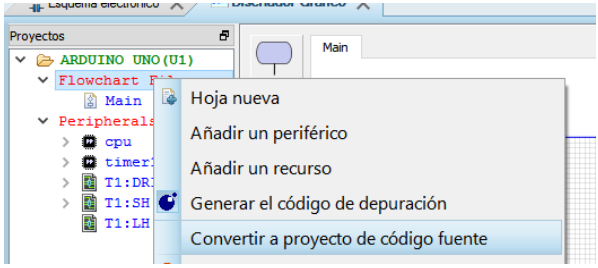
La forma de realizarlo se ha mostrado con todo detalle en el epígrafe 7 de esta ayuda.

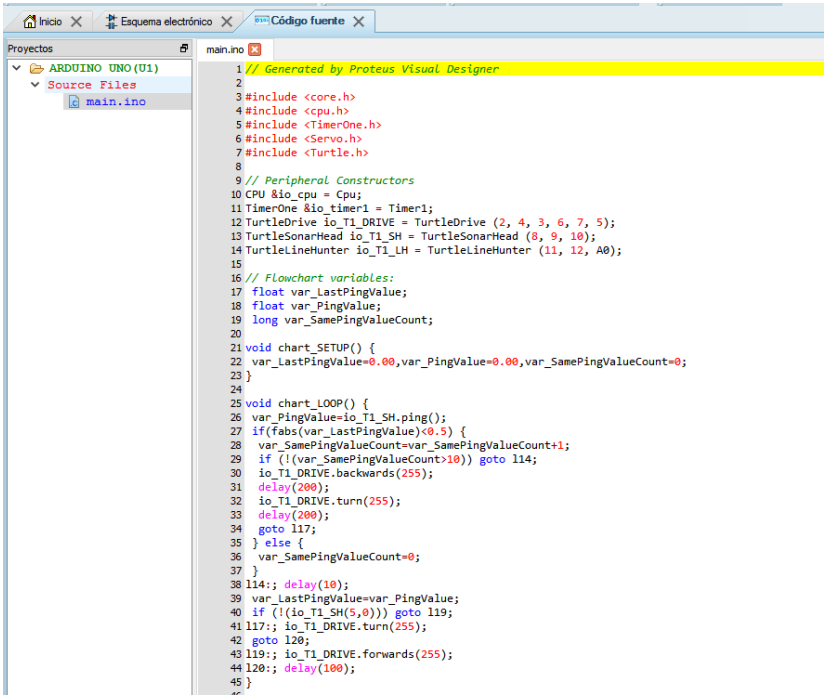
11.5.-Referencia de comandos del Diseñador Gráfico relativos a los robot-tortuga.

Cuando trabajamos utilizando el Diseñador Gráfico, una gran parte de la complejidad de las tareas de gobierno del robot-tortuga se puede llevar a cabo utilizando los métodos de alto nivel que abstraen el comportamiento del robot de la electrónica asociada, permitiéndonos concentrar el foco de nuestra atención en los algoritmos de control.

Puesto que cada robot-tortuga está construido utilizando diferentes componentes, los métodos de control que proporciona el Diseñador Gráfico deben ser diferentes para cada tipo de robot-tortuga para adaptarse al hardware de cada uno de ellos. En este epígrafe vamos a verlos en profundidad.

Los usuarios más avanzados podrán encontrar útil convertir sus proyectos a código fuente utilizando el menú contextual asociado desde el árbol de estructura del proyecto y modificar su código utilizando el lenguaje C.





⚠ Hay que tener muy presente que cuando se utiliza la opción 'convertir a proyecto de código fuente' para convertir un proyecto de flujograma en uno de código fuente en lenguaje C, ésta opción, una vez ejecutada, no puede ser revertida.

Esta opción puede ofrecer diversas ventajas a los usuarios, como la capacidad de acceder a los diferentes periféricos (v.g. el giroscopio o el acelerómetro) a nivel de registro, aunque, indudablemente, será una operación complicada que exige un mayor conocimiento técnico. En última instancia Proteus simula el equipo del mundo real, lo que significa que el usuario puede desarrollar sus propios códigos en la forma en la que él se encuentre más cómodo. En concreto, con este sistema se posibilita que librerías de terceras partes o ejemplos de código fuente disponibles en internet, puedan ser añadidos, utilizados, simulados y depurados en cualquier proyecto utilizando el interface de trabajo que ofrece Proteus.

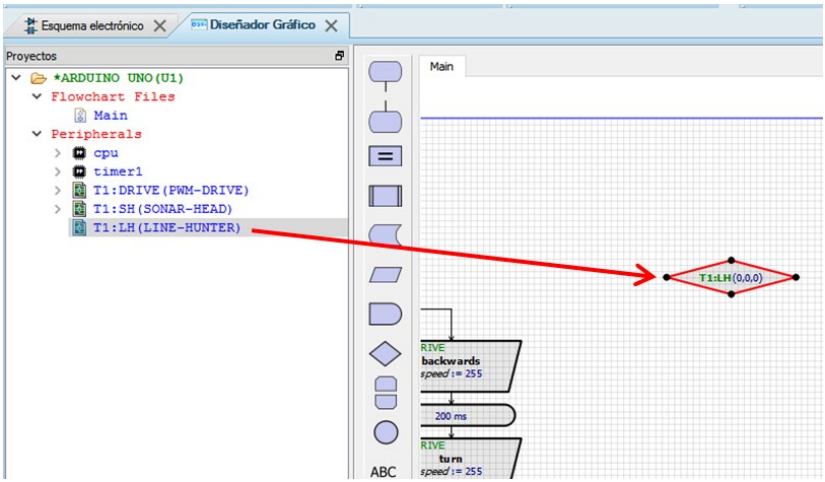
11.5.1.-Comandos del robot-tortuga Funduino.

El control de un robot-tortuga Funduino se lleva a cabo utilizando diferentes funciones y métodos que podemos agrupar en tres categorías diferentes: los que controlan los sensores del tipo 'seguidor de línea' para detectar la posición respecto a una línea, los que controlan los sensores de ultrasonidos situados en la cabeza del robot-tortuga para detectar obstáculos y los que controlan los motores de corriente continua que mueven las ruedas.

11.5.1.1.-Control de los sensores 'seguidor de línea'.

En la zona inferior del robot-tortuga Funduino están ubicados tres sensores. Cada uno de ellos proporciona una señal digital a la placa de control en función de si están detectando o no la línea.

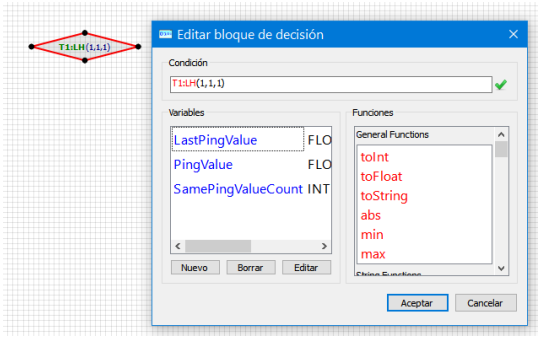
En el Diseñador Gráfico controlamos los tres sensores con un único bloque de control del tipo decisión que utiliza una función LH. Para utilizar un bloque de decisión que llame a la función, sólo tenemos que coger y arrastrar la función LH(LINE-HUNTER) asociada al periférico desde el árbol de estructura del proyecto y soltarlo en el flujograma.



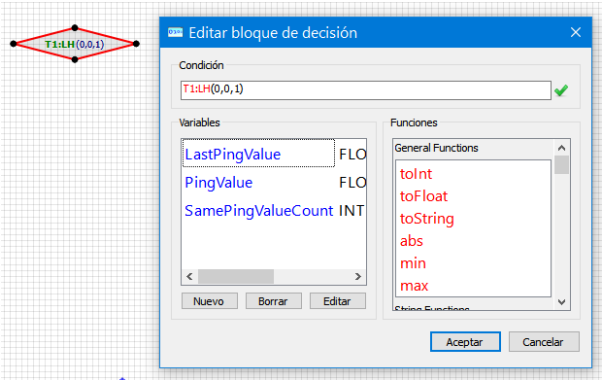
Una vez posicionado el bloque de decisión asociado a la función 'LH (LINE-HUNTER)' dentro del flujograma podemos editarlo para comprobar las lecturas de los sensores y generar la condición que

debe cumplirse para ejecutar una rama u otra del flujograma. La función LH recibe tres argumentos con la que indicamos la condición que debe cumplir cada uno de los tres sensores. Si el argumento vale 1, el estado de ese sensor debe ser TRUE. Si el argumento vale 0, el estado de ese sensor debe ser FALSE. Y si el argumento vale -1, el estado de ese sensor no se tendrá en cuenta.

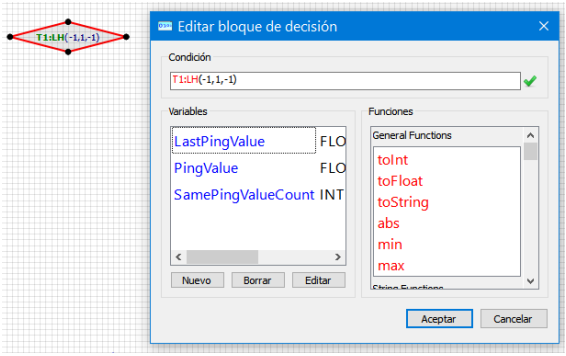
Veamos unos ejemplos para entender mejor el uso de la función LH. Si queremos comprobar que los tres sensores se encuentren sobre la ruta que estamos siguiendo, tendremos que llamar a la función LH pasando como argumentos tres unos: LH(1,1,1). El bloque de condición devolverá TRUE cuando los tres sensores se encuentren situados sobre la línea. De esta forma, en el código podremos controlar posibles desviaciones y realizar correcciones para evitar oscilaciones del robot-tortuga.



Si deseamos controlar la necesidad de realizar un giro brusco hacia la derecha, puede resultarnos útil vigilar que el sensor de la derecha se encuentre sobre la línea y que los otros dos la hayan perdido. En este caso llamaríamos a la función LH pasándole como parámetros los valores 0, 0 y 1.



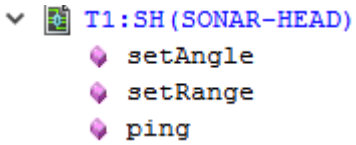
Un tercer caso posible es que nos interese simplemente conducir a nuestro robot-tortuga para que avance siguiendo la línea en cuyo caso, puede servirnos con vigilar únicamente el sensor del centro, ignorando el estado de los otros dos sensores. En este caso, los parámetros pasados a la función LH tendrán los valores -1, 1 y -1.



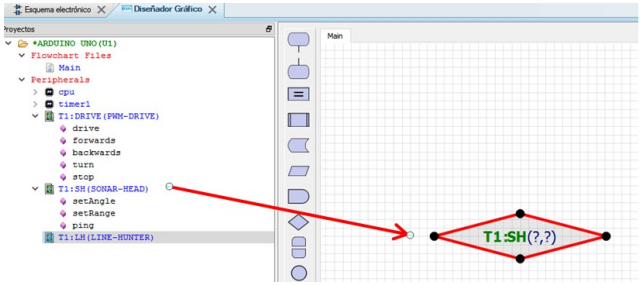
11.5.1.2.-Control del sensor de ultrasonidos.

El sensor de ultrasonidos emite ráfagas de ondas cortas (pings) que rebotan en los obstáculos y capta dichos rebotes calculando el tiempo que ha transcurrido. De esta forma determina la distancia a la que se encuentra el objeto. El sensor está ubicado en la parte superior del robot-tortuga en una unidad giratoria. Esta configuración nos permite fijar en primer lugar la dirección hacia la que apunta el sensor y luego comprobar si existen obstáculos dentro de esa dirección a una determinada distancia (siempre dentro del rango de

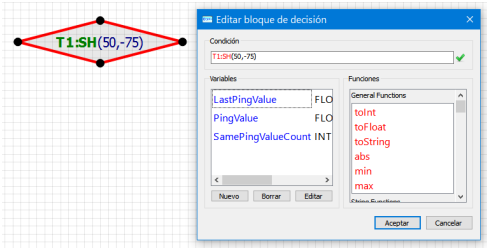
alcance del sensor de ultrasonidos). El Diseñador Gráfico nos proporciona una función para controlar el sensor de ultrasonidos llamada SH() y tres métodos: setRange, setAngle y ping.



La función SH(SONAR-HEAD) es un bloque del tipo decisión que nos permite comprobar el estado del dispositivo. Se utiliza, como es habitual, cogiendo y arrastrando la función desde el árbol de estructura del proyecto al flujograma.

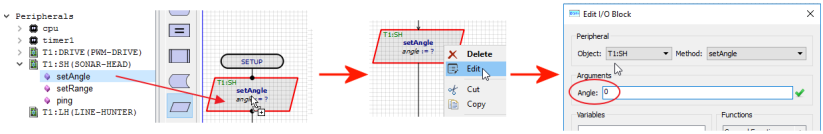


La función SH() recibe dos parámetros. El primero indica la distancia y el segundo el ángulo vigilado. La función devuelve TRUE cuando se detecta un obstáculo dentro de la distancia especificada y en el ángulo de visión especificado. Por ejemplo, la función SH(50,-75) devolverá el valor TRUE cuando detecte un objeto a menos de 50 centímetros situado a la izquierda del robot-tortuga.

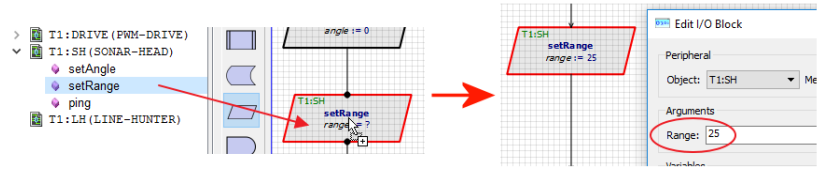


El método setAngle nos posibilita definir la posición de la cabeza sobre la que va montada el sensor de ultrasonidos. En la mayoría de los casos se fijará dentro de la rutina setup() de configuración del

proyecto con una valor igual a 0 (que apunte hacia adelante). Más tarde, a lo largo de la ejecución del programa se podrán ejecutar más comandos para variar esta orientación original.

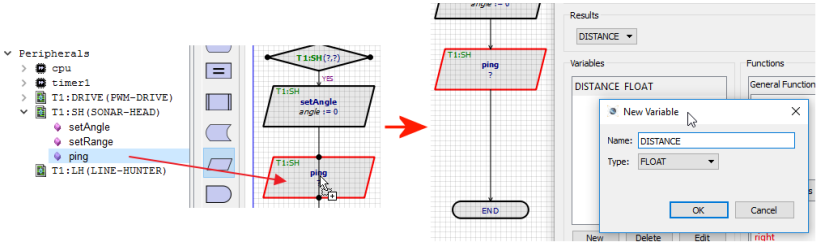


El método setRange el rango máximo dentro del cual deseamos detectar obstáculos. Desde el punto de vista de funcionamiento del sensor lo que estamos definiendo es el tiempo máximo que debemos esperar por la recepción del rebote del ping generado. Si el rebote no se alcanza en ese periodo de tiempo, no hay un obstáculo en nuestra ruta. En la mayoría de los proyectos se fijará este valor dentro de la rutina setup() de configuración. Más tarde, a lo largo de la ejecución del programa se podrán ejecutar más comandos para variar esta distancia original.



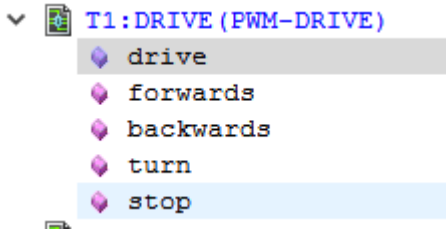
El método ping fuerza la generación de una onda de ultrasonidos, convierte el tiempo que se tarda en recibir el rebote en su equivalente en centímetros y devuelve el valor. Si el valor devuelto es -1 significa que ningún objeto se ha detectado dentro de la distancia máxima de alcance del sensor. Cuando el método se coloca en el flujograma, es necesario editarlo e indicar una variable de trabajo donde se escribirá el valor devuelto y que sirve para consultar el valor recibido.

> ⚠ La variable creada para recibir el valor devuelto por el método ping debe ser del tipo float. Si no, no aparecerá en el desplegable de variables disponibles para almacenar el valor.

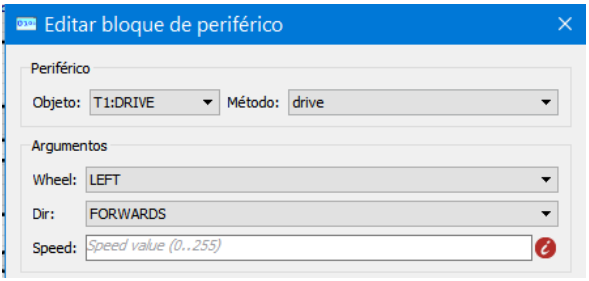


11.5.1.3.-Control de los motores.

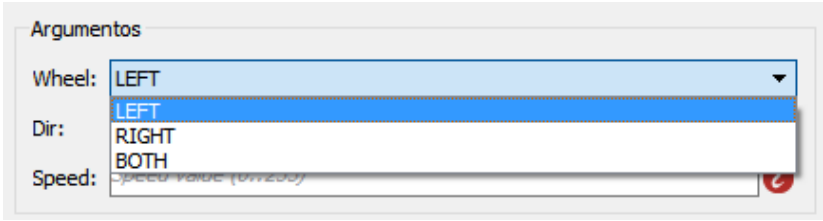
La función drive() nos proporciona el control sobre las ruedas izquierda y derecho unidas al motor. El gobierno se lleva a cabo mediante cinco métodos que nos ayudan a gestionar la dirección del desplazamiento y la velocidad.

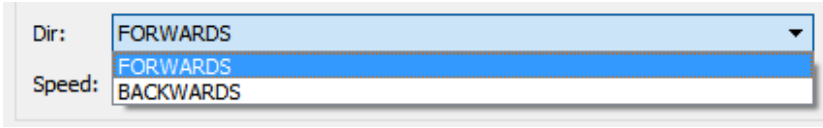


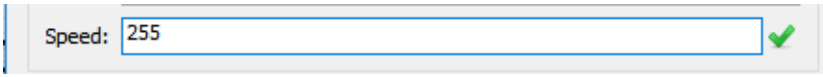
El método drive nos da un control exhaustivo sobre las dos ruedas. Tiene tres argumentos: 'Wheel', 'Dir' y 'Speed'.



El primero indica que ruedas vamos a controlar (la izquierda, la derecha o ambas), el segundo la dirección del desplazamiento (adelante o atrás) y la tercera la velocidad del movimiento.

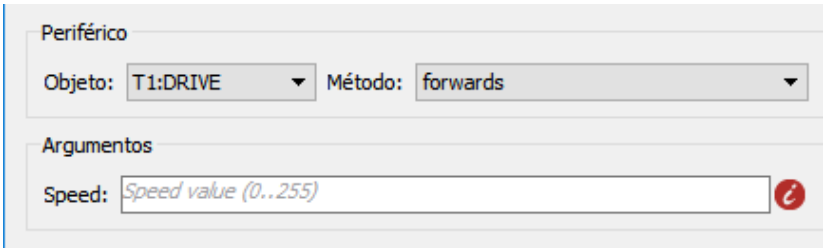




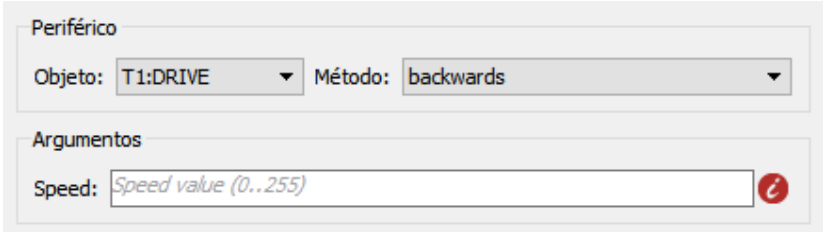


La velocidad es un entero entre 0 y 255. Físicamente se corresponde con el ciclo útil de la señal PWM que se utiliza para gobernar el motor.

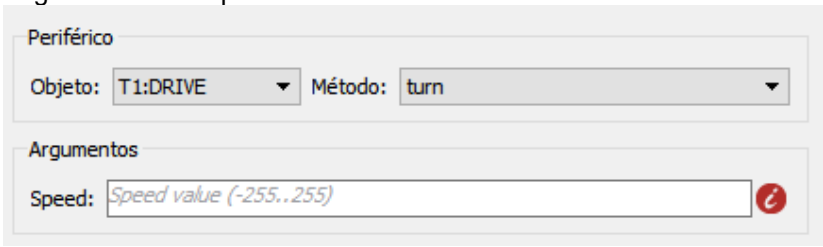
El método forwards es una simplificación del anterior donde se sobreentiende que se gobiernan las dos ruedas y que el movimiento se produce hacia adelante, así que sólo nos pide como argumento la velocidad.



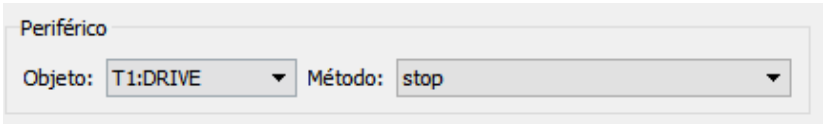
El método backwards es el equivalente cuando queremos que el movimiento se produzca hacia atrás. Igualmente sólo pide como argumento la velocidad.



El método turn es otra simplificación que facilita la ejecución de un giro del robot-tortuga con sólo especificar la velocidad de giro. Los valores negativos hacen girar al robot-tortuga a la izquierda y los valores positivos hacia la derecha. En la práctica el efecto que se produce es el giro de una rueda hacia adelante y de la otra hacia atrás a la misma velocidad. Así que puede entenderse este método como giro brusco, mientras que sería posible realizar un giro suave, haciendo girar una sola rueda mientras la otra permanece detenida. El giro suave sólo puede llevarse a cabo utilizando la función drive.



El método stop provoca la parada del robot-tortuga y no necesita argumentos.



11.5.2.-Comandos del robot-tortuga Zumo.

El control de un robot-tortuga Zumo utilizando el Diseñador Gráfico se lleva a cabo utilizando diferentes funciones y métodos que podemos agrupar en tres categorías diferentes: los que controlan los sensores del tipo ‘seguidor de línea’ para detectar la posición respecto a una línea, los que controlan el giroscopio y la brújula para

posicionar en un punto u orientar el movimiento al robot-tortuga y los que controlan los dos motores que mueven las bandas de rodadura.

11.5.2.1.-Control de los sensores 'seguidor de línea'.

El robot-tortuga Zumo trae montados una matriz de seis sensores que emiten rayos infrarrojos y que son capaces de captar el reflejo recibido de los rayos infrarrojos al rebotar sobre la superficie sobre la que está rodando el robot-tortuga para, diferenciando los diferentes contrastes que se producen dependiendo del color de la superficie, poder realizar las funciones de sensores 'seguidor de línea' o de sensores 'detectores de bordes'.

Cada sensor se compone de un emisor de rayos infrarrojos y un fototransistor que es capaz de leer cuánta luz se refleja de vuelta. A diferencia del caso del robot-tortuga Funduino, el Zumo está diseñado para permitir el uso de algoritmos de control del tipo PID (proporcional, integral y derivativo) en la detección y seguimiento de líneas puesto que la señal es analógica.

El modelo del robot-tortuga Zumo suministrado con Proteus incluye dos métodos: readLinePos y readRowValue.

El método readLinePos no necesita parámetros y devuelve como valor un entero entre 0 y 6.000 que indica la posición de la línea debajo de los seis sensores de infrarrojos.

El método utiliza la lectura de los valores recibidos de cada uno de los seis sensores de infrarrojos que componen el sistema. El valor que proporciona cada sensor es una medida de la cantidad de luz reflejada recibida utilizando unidades de medida abstractas en un rango entre 0 y 1.000. El valor más alto se corresponde con la menor cantidad de luz reflejada (la que refleja una superficie negra o la que se recibe cuando se detecta un borde: cuando la superficie está situada tan lejos del robot-tortuga que no es posible que el rayo se refleje).

La posición estimada del robot con respecto a la línea se calcula utilizando una media ponderada de los valores recibidos de cada sensor multiplicado por mil. Si el resultado de la media ponderada es cero, significa que la línea está directamente situada debajo del sensor 0. Si el resultado es 1.000 indica que la línea está

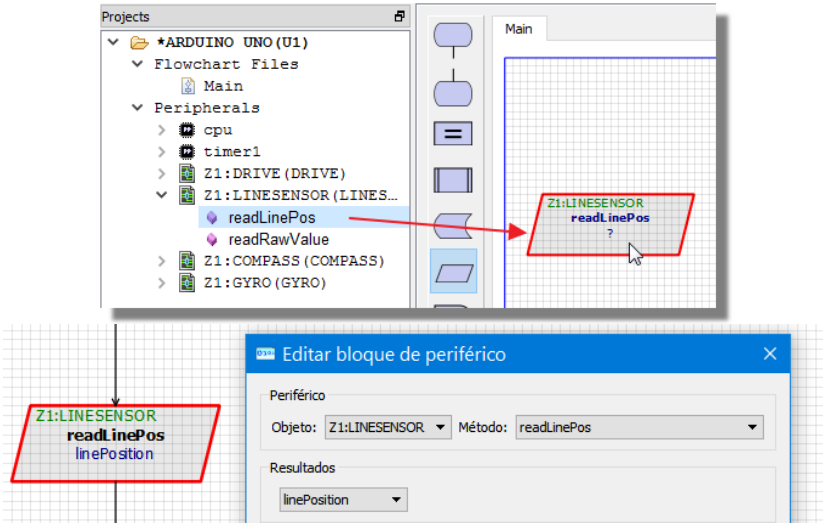
directamente situada debajo del sensor 1. Y así sucesivamente. Valores intermedios significan que la línea se encuentra entre dos de los sensores.

La fórmula utilizada para obtener la media ponderada es la siguiente:

$$\frac{0 * ValorSensor0 + 1.000 * ValorSensor1 + 2.000 * ValorSensor2 + 3.000 * ValorSensor3 + 4.000 * ValorSensor4 + 5.000 * ValorSensor5}{ValorSensor0 + ValorSensor1 + ValorSensor2 + ValorSensor3 + ValorSensor4 + ValorSensor5}$$

Al aplicar esta fórmula, el resultado obtenido nos indica la posición de la línea. Si el valor recibido es 0 la línea se encuentra justo debajo del sensor situado más a la izquierda. Si el valor es 1.000 la línea se encuentra justo debajo del segundo sensor empezando por la izquierda. El valor 2.500 significa que la línea está justo en el medio del tercer y cuarto sensor empezando por la izquierda. Un valor de 6.000 significa que la línea está justo debajo del sensor situado más a la derecha.

Para utilizar el método solo tenemos que 'cogerlo y arrastrarlo' desde el árbol de estructura del proyecto hasta el flujograma.



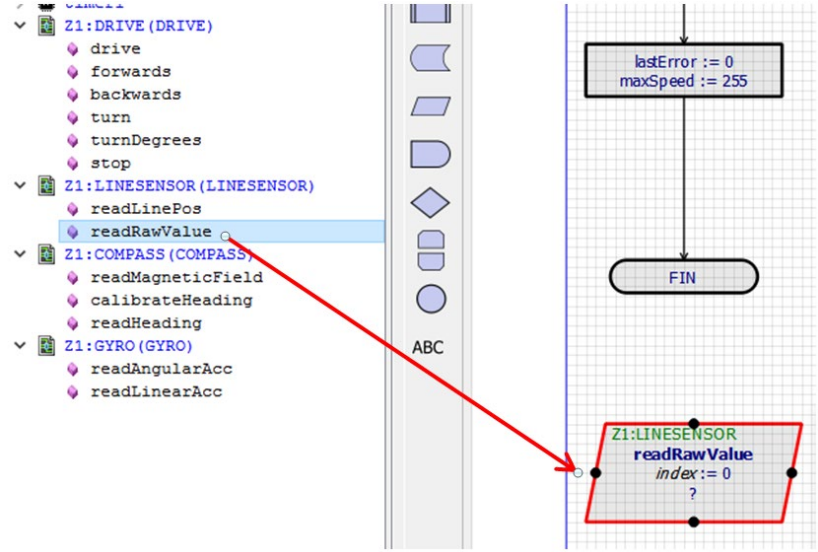
El método readLinePos almacena el valor leído en una variable de trabajo de tipo entero que previamente hemos tenido que crear para que aparezca en el desplegable de resultados.

El método readRawValue utiliza un argumento llamado 'index' que se utiliza para indicar el sensor del que deseamos recibir la lectura. Puede tomar los valores comprendidos entre 0 y 5, siendo el cero el sensor situado más a la izquierda y cinco el situado más a la derecha.

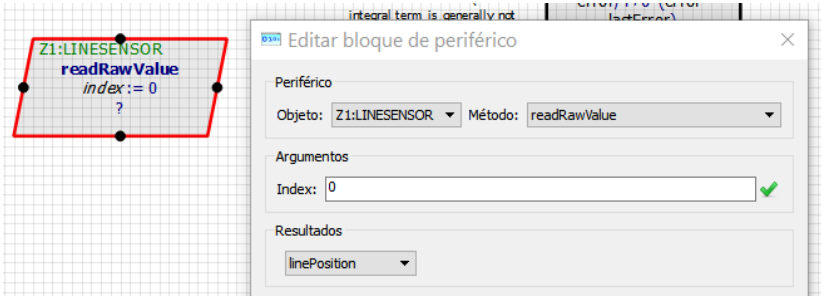
El método devuelve como valor un entero entre 0 y 1.000 que se corresponde con la medida de la cantidad de luz reflejada recibida. El valor más alto se corresponde con la menor cantidad de luz reflejada (la que refleja una superficie negra o la que se recibe cuando la superficie está situada tan lejos del robot-tortuga que no es posible que el rayo se refleje).

Los valores recibidos no son medidas calibradas. Eso significa que hay que tratar los datos cuidadosamente, porque los valores recibidos pueden cambiar cuando ejecutemos nuestro proyecto en el mundo real en función de las condiciones ambientales: luz ambiente, contrastes de la pista de rodadura, etc.

Para utilizar el método solo tenemos que 'cogerlo y arrastrarlo' desde el árbol de estructura del proyecto hasta el flujograma.



El método readRawValue almacena el valor leído en una variable de trabajo de tipo entero que previamente hemos tenido que crear para que aparezca en el desplegable de resultados.



11.5.2.2.-Control del sensor brújula.

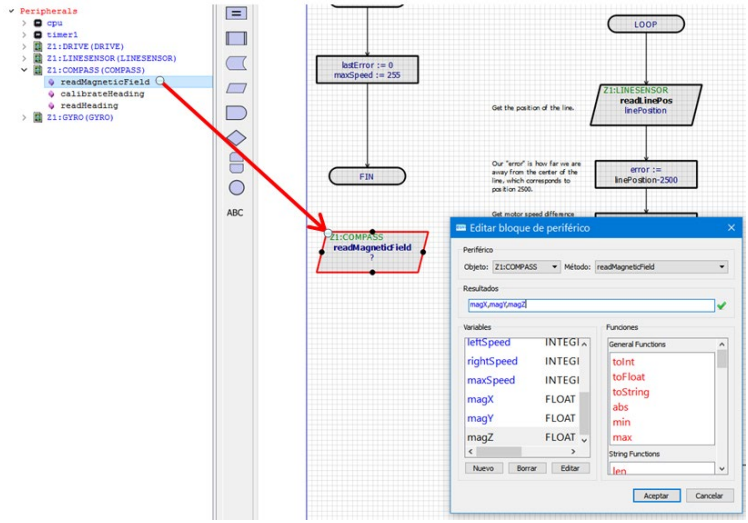
El robot-tortuga Zumo incluye un sensor del tipo brújula que es un integrado que realiza las funciones de acelerómetro de tres ejes y magnetómetro. El integrado incluido en el robot-tortuga Zumo que realiza las funciones de sensor brújula es el LSM303D. Para controlarlo el Diseñador Gráfico proporciona tres métodos.

La función readMagneticField utiliza tres parámetros del tipo punteros a variables del tipo coma con punto flotante: magX, magY y magZ. Cada uno de ellos devuelve la componente sobre el eje correspondiente del campo magnético medido en unidades Gauss.

Al utilizar la función se indican los tres punteros que deseamos utilizar y que son utilizados por la rutina subyacente para devolver en ellos los tres componentes del campo magnético. Hay que tener muy en cuenta que un integrado del tipo brújula es muy inexacto tanto en términos relativos como absolutos. Por ejemplo, es imposible determinar la desviación del vector respecto al origen y al norte sin llevar a cabo operaciones de calibración.

> ⚠ Un dispositivo de lectura del tipo brújula recibe un gran número de interferencias generadas por los motores, las baterías, la propia placa de circuito impresa o por otros dispositivos que se encuentren a su alrededor. Sin embargo, después de calibrado puede ser utilizado como un sistema de orientación con una moderada exactitud en la mayoría de los entornos.

Para utilizar la función solo tenemos que 'cogerlo y arrastrarlo' desde el árbol de estructura del proyecto hasta el flujograma. A continuación utilizando el menú contextual editaremos sus propiedades para indicar las tres variables, separándolas con comas, que vamos a utilizar para recibir los datos. Las variables deben haber sido creadas con anterioridad y ser del tipo flotante.



La función calibrateHeading no recibe ningún parámetro ni devuelve ningún valor y se utilizar para calibrar el sensor brújula y averiguar la posición del norte y calcular el centro de coordenadas.

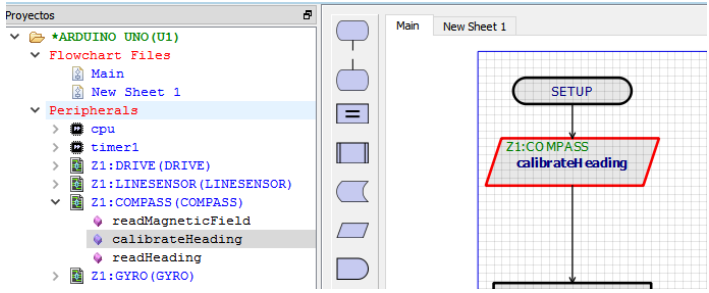
> Antes de llamar a este método hay que colocar el robot-tortuga Zumo apuntando al norte.

La función se utiliza para hacer girar el robot-tortuga Zumo sobre el sitio en el que se encuentra e ir realizando múltiples lecturas del campo magnético. Durante el proceso de tomar lecturas, se guarda el valor mínimo y el valor máximo encontrados y con ellos se calcula el centro. El giro continúa hasta que no se encuentran nuevos valores por debajo del mínimo guardado ni por encima del máximo durante un periodo de 500 milisegundos. Como el robot-tortuga Zumo está mirando al norte antes de empezar la ejecución de este método, se utiliza la primera lectura obtenida para calcular el ángulo del vector que apunta al norte.

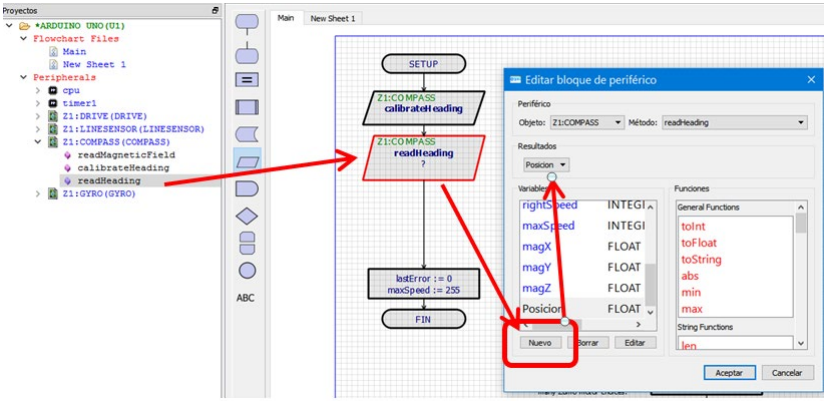
El método readHeading no recibe ningún parámetro y devuelve un único valor de tipo float donde se puede leer la posición a la que está apuntando el robot-tortuga Zumo. El valor recibido está dentro del rango -180,0 a +180,0 grados. El valor 0 identifica al norte.

El método requiere haber utilizado previamente la función calibrateHeading(). Si no se ha hecho devuelve el valor -1.

Veamos un ejemplo de utilización del sensor brújula. El primer paso será llevar a cabo la calibración. Para ello arrastramos la función calibrateHeading y la colocamos en el flujograma. El lugar habitual de colocar esta función es dentro de la subrutina 'setup' de configuración que se ejecuta al arrancar el robot-tortuga Zumo.



A continuación usaremos el método readHeading en el lugar de nuestro flujograma dónde queramos conocer la posición de nuestro robot-tortuga. Para ello arrastramos la función dentro de nuestro flujograma. Necesitamos crear la variable de trabajo de tipo float para recibir la posición de nuestro robot-tortuga.



Un valor recibido igual a 0 indica el Norte, un valor de -90 indica el Oeste, un valor de 180 el Sur y un valor de 90 indica el Este. Todos los valores restantes del intervalo -180 a 180 indican las diferentes orientaciones posibles.

11.5.2.3.-Control del sensor giroscopio.

El robot-tortuga Zumo a partir de la versión 1.2 incluye también un integrado del tipo L3GD20H-3 que es un giroscopio de tres ejes y que puede ser utilizado para ejecutar rotaciones. En conjunción con el sensor LSM3030D proporciona una medida del movimiento de nuestro robot-tortuga muy efectiva.

El Diseñador Gráfico incluye dos métodos para obtener la información del sensor giroscopio. Uno devuelve la descomposición de la aceleración angular sobre los tres ejes en grados centígrados por segundo y el otro la de la aceleración lineal en unidades G.

El método readAngularAcc utiliza tres parámetros del puntero a variable de tipo float. Los utiliza para devolver el valor de las tres componentes del vector aceleración angular sobre cada uno de los ejes: angX, angY, angZ.

El método readLinearAcc utiliza tres parámetros del puntero a variable de tipo float. Los utiliza para devolver el valor de las tres componentes del vector aceleración lineal sobre cada uno de los ejes: angX, angY, angZ.

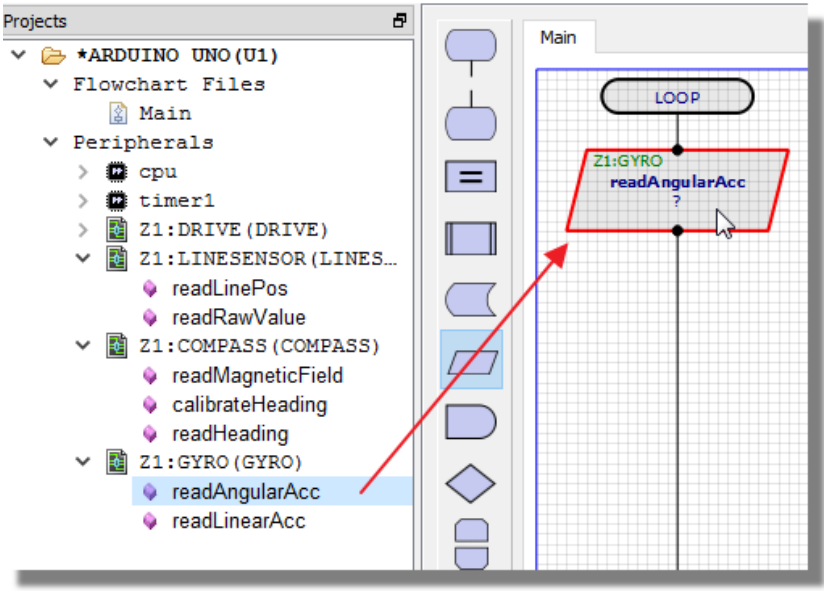
La medida real de la aceleración lineal la proporciona el dispositivo LSM303 mientras que la aceleración angular se lee desde el dispositivo L3GD20H. La aceleración lineal es difícil de medir en un entorno de simulación cuando todo ocurre durante un período tan corto de tiempo. Por ese motivo Proteus implementa el funcionamiento de este método de la siguiente manera:

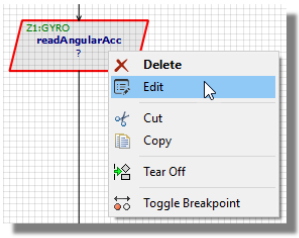
El modelo no simula la aceleración porque en cuanto un robot-tortuga se ha puesto en movimiento obedeciendo una orden se establece la velocidad de forma instantánea de forma tan rápida que no hay puntos para realizar el modelado correctamente. Esto es un problema en la obtención del valor de la aceleración lineal. Así que la modelación se ha realizado observando el cambio experimentado en la velocidad de un fotograma a otro de la simulación y limitando el cambio a un máximo de 1G para la aceleración medida durante el movimiento y a 10G para el caso de producirse una colisión.

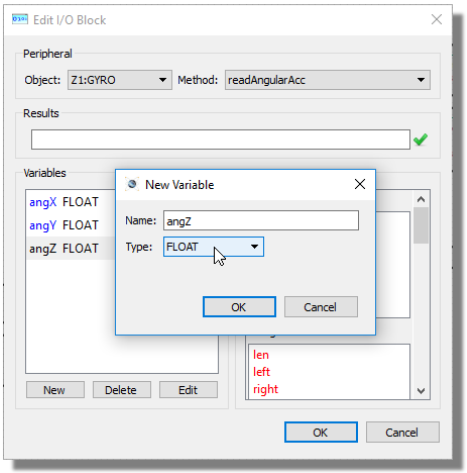
En la práctica, Labcenter asume que la aceleración lineal puede ser utilizada dentro de bucles con objeto de comprobar posibles colisiones. De todas formas, agradecemos aportaciones de los

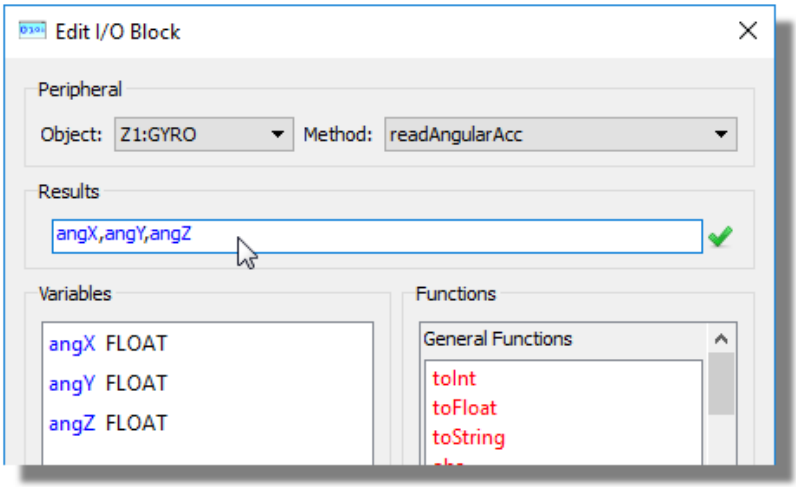
lectores que nos expliquen cómo trabaja y si el modelo es correcto o tiene fallos.

La forma de utilizar estos dos métodos es similar a la que utilizamos para el resto de métodos: coger y arrastrar el método dentro del flujograma y crear las variables necesarias.





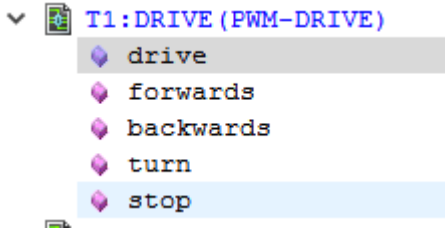




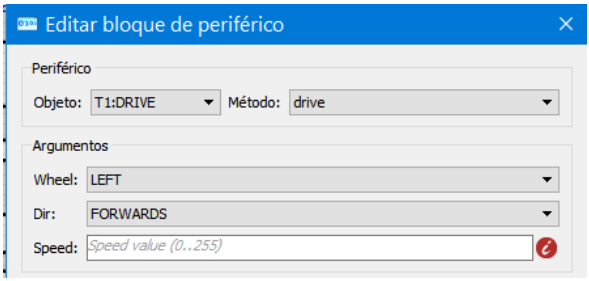
11.5.2.4.-Control de los motores.

El robot-tortuga Zumo integra dos motores con reductora gobernados mediante el uso de un integrado DRV8835. El Diseñador Gráfico proporciona varias funciones para controlarlos.

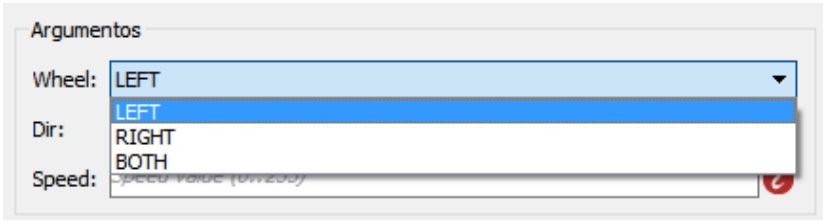
La función drive() nos proporciona el control sobre las ruedas izquierda y derecho unidas al motor. El gobierno se lleva a cabo mediante cinco métodos que nos ayudan a gestionar la dirección del desplazamiento y la velocidad.

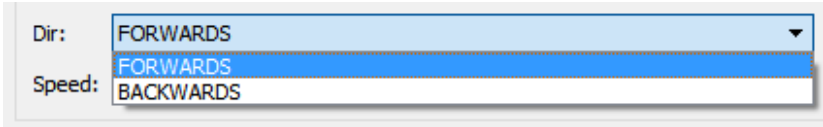


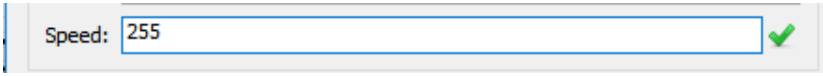
El método drive nos da un control exhaustivo sobre las dos ruedas. Recibe tres argumentos: ‘Wheel’, ‘Dir’ y ‘Speed’.



El primero indica que ruedas vamos a controlar (la izquierda, la derecha o ambas), el segundo la dirección del desplazamiento (adelante o atrás) y la tercera la velocidad del movimiento.

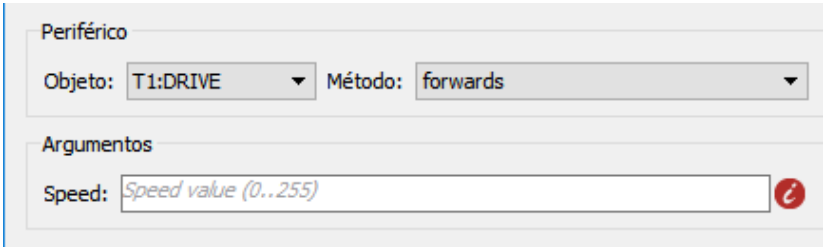




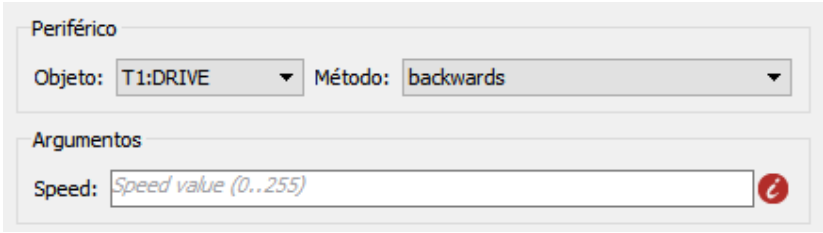


La velocidad es un entero entre 0 y 255. Físicamente se corresponde con el ciclo útil de la señal PWM que se utiliza para gobernar el motor.

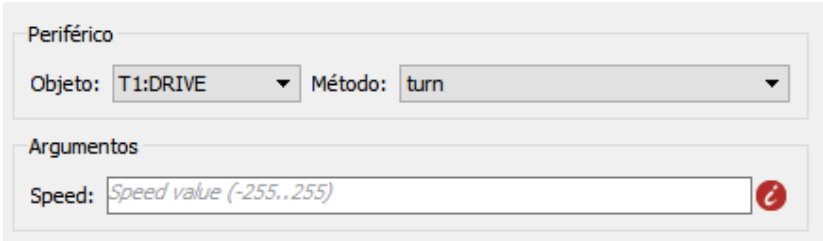
El método forwards es una simplificación del anterior donde se sobreentiende que se gobiernan las dos ruedas y que el movimiento se produce hacia adelante, así que sólo nos pide como parámetro la velocidad.



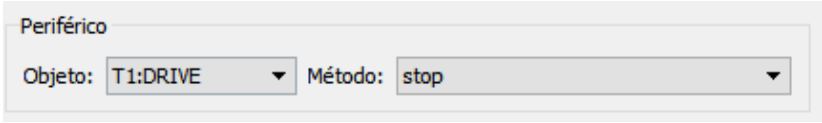
El método backwards es el equivalente cuando queremos que el movimiento se produzca hacia atrás. Igualmente sólo pide como parámetro la velocidad.



El método turn es otra simplificación que facilita la ejecución de un giro del robot-tortuga con sólo especificar la velocidad de giro. Los valores negativos hacen girar al robot-tortuga a la izquierda y los valores positivos hacia la derecha. En la práctica el efecto que se produce es el giro de una rueda hacia adelante y de la otra hacia atrás a la misma velocidad. Así que puede entenderse este método como giro brusco, mientras que sería posible realizar un giro suave, haciendo girar una sola rueda mientras la otra permanece detenida. El giro suave sólo puede llevarse a cabo utilizando la función drive.



El método stop provoca la parada del robot-tortuga y no necesita parámetros.



11.6.-La simulación de la mecánica de un robot-tortuga.

Es muy importante tener presente que la simulación de los robot-tortuga que realiza Proteus es muy fiel desde el punto de vista eléctrico pero no es una simulación real desde el punto de vista del mundo físico.

El control del movimiento del robot-tortuga se realiza utilizando un bucle abierto y, por lo tanto, la distancia recorrida en un periodo concreto de tiempo no es calculada de forma determinista. Todos los efectos físicos asociados al movimiento como la inercia, el momento, la fricción y la carga de la batería tienen un importante impacto en la distancia recorrida por el robot-tortuga en el mundo real. Por lo tanto en ningún caso debe aceptarse que los resultados de la simulación coincidirán con los obtenidos en el mundo real.

> Durante la simulación de un robot-tortuga se puede calcular la distancia recorrida utilizando el giroscopio que es una potente ayuda para escribir el código, pero no es posible predecir la distancia recorrida durante un determinado espacio de tiempo con antelación.

Por ejemplo, un robot-tortuga puede recorrer 20 centímetros desplazándose sobre una mesa y sólo recorrer 10 centímetros en el mismo tiempo si se desplaza sobre una alfombra. Puede perder el 20% de su potencia de movimiento cuando las baterías se van descargando. Incluso puede no llegar a moverse en absoluto si el ciclo de trabajo del control PWM no es capaz de proporcionar suficiente energía para vencer la resistencia que se produce por el rozamiento con la superficie de rodadura durante el arranque.

De forma similar, puede producirse diferencias significativas en el comportamiento del robot-tortuga cuando funciona como un 'seguidor de líneas' dependiendo de las condiciones medioambientales de iluminación y en función del contraste entre la línea y el fondo del tapiz.

--Página dejada intencionadamente en blanco para sus anotaciones—