

LA SUITE PROTEUS

Primeros pasos con EL TALLER IoT

labcenter Electronics
www.labcenter.com

Información sobre derechos y copyright

© Labcenter Electronics Ltd 1990-2020. 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.

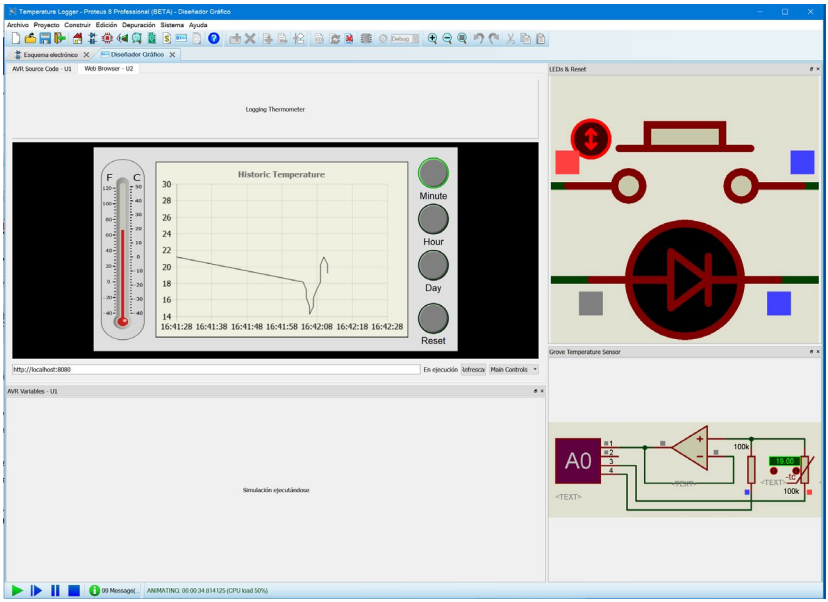
Se recuerda en particular a todos los usuarios que la simulación con éxito de un diseño utilizando el simulador PROSPICE no demuestra de manera concluyente que vaya a trabajar cuando se pasan los datos al proceso de fabricación. Siempre es conveniente hacer un prototipo de pruebas antes de mandar fabricar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

El Taller IoT[1] es un módulo de Proteus que sirve para desarrollar cuadros de mando remotos con los que monitorizar y/o gobernar equipos basados en el Arduino Yun o similares. El Taller IoT es una herramienta complementaria que se utiliza cuando desarrollamos aplicaciones para Arduino, bien utilizando el módulo Diseñador Gráfico de Proteus o el módulo código fuente (VSM).



Para trabajar con el taller IoT debemos seguir los siguientes pasos:

1. Crear un proyecto con un flujograma utilizando el modulo Diseñador Gráfico o un programa utilizando el módulo código fuente, eligiendo como equipo base un dispositivo

[1] IoT son las abreviaturas de 'Internet of the things', el internet de las cosas. Podemos encontrar más información al respecto en el enlace: https://es.wikipedia.org/wiki/Internet_de_las_cosas.

que soporte el uso del Taller IoT como, por ejemplo, el Arduino Yun.

2. Añadir las tarjetas de expansión, los módulos accesorios independientes y los módulos de sensores Grove a nuestro diseño utilizando la galería de periféricos del Diseñador Gráfico.
3. Colocar los mandos (selectores, pulsadores, indicadores, etc.) en el panel del cuadro de control distribuyéndolos a nuestro gusto. Todos estos mandos quedan enlazados dentro de nuestro proyecto de forma automática. Además, todos los métodos asociados a ese mando para facilitar el manejo del mismo desde nuestro flujograma o programa C++, estarán preparados para ser utilizados de forma muy sencilla utilizando la técnica 'coger y arrastar'.
4. Escribir el programa con la lógica de control. Todas las tarjetas de expansión y los mandos que seleccionamos en los pasos anteriores tienen asociados métodos en forma de bloques de flujograma (si estamos utilizando el Diseñador Gráfico) o funciones (si estamos utilizando la pestaña Código Fuente) que facilitan enormemente la escritura del programa.
5. Arrancar la simulación y depurar el código. Utilizando el botón 'Play' la ponemos en marcha y podemos utilizar el cuadro de mando remoto actuando con los equipos virtuales definidos en la pestaña diseño electrónico. Podemos fijar puntos de ruptura o realizar la ejecución del código paso a paso para facilitar la depuración.
6. Implementar nuestro trabajo en los equipos reales. Una vez que hemos comprobado que todo funciona en la simulación, podemos descargar el programa a los equipos de la vida real y utilizar nuestro teléfono móvil o tableta para monitorizar o controlar nuestro equipo Arduino.

1.2.- Equipos con los que se puede usar el Taller IoT.

Aunque en el momento de escribir esta ayuda se está trabajando en el desarrollo de otros equipos con los que se podrá utilizar el Taller IoT, por ahora se puede utilizar el Taller IoT en proyectos de Proteus

que utilizan equipos Arduino Yun o la tarjeta de expansión VFP Server (Yun).





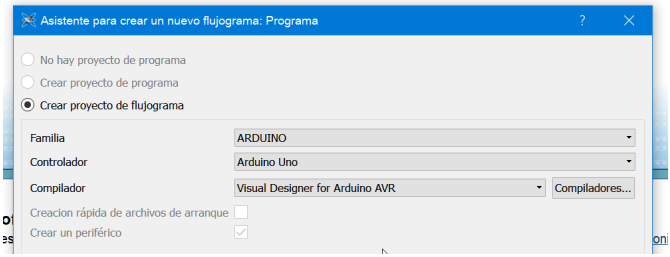
1.2.1.-Utilizar un equipo Arduino Yun.

Desde la pantalla de inicio utilizamos el botón 'Nuevo Flujograma' y seleccionamos como controlador la opción 'Arduino Yun'

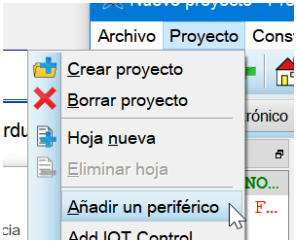


1.2.2.-Utilizar la tarjeta de expansión VFP Server (Yun).

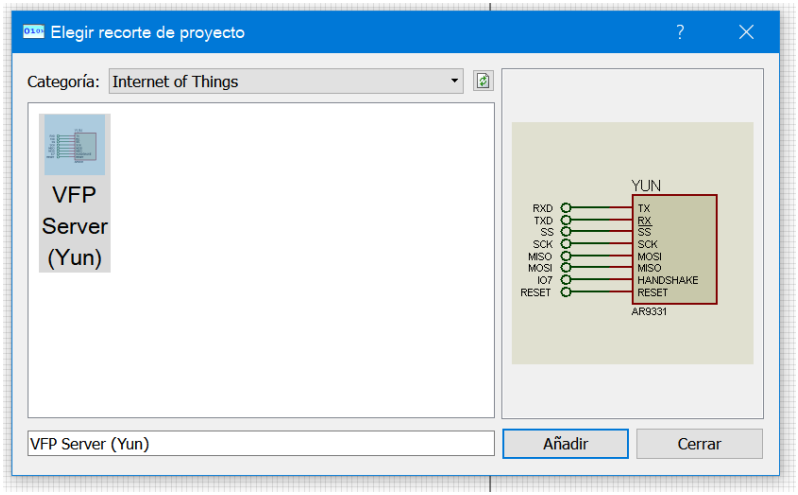
Desde la pantalla de inicio utilizamos el botón 'Nuevo Flujograma' y seleccionamos como controlador la opción 'Arduino One' o 'Arduino Mega'.



Y una vez creado el proyecto, desde la pestaña del flujograma seleccionamos la opción añadir un periférico del menú ‘Proyecto’.



En la ventana de diálogo que aparece, escogemos la categoría ‘Internet de las cosas’ y seleccionamos la tarjeta de expansión ‘VFP Server (Yun)’



El proceso a seguir varía muy poco si creamos un proyecto de código fuente y no deseamos usar el Diseñador Gráfico. En este caso en la pantalla de inicio debemos seleccionar 'Nuevo Proyecto' en lugar de 'Nuevo Flujograma'.

> Utilizar un dispositivo Arduino Yun es más sencillo que utilizar la tarjeta de expansión VFP Server (Yun), pero la tarjeta de expansión tiene mejor cobertura Wifi y puede ser acoplada a un Arduino Mega que dispone de más puertos de entrada y salida y más cantidad de memoria.

1.3.-Objetivo y alcance de este documento de ayuda.

El objetivo de este documento de ayuda es cubrir el uso de la herramienta Taller IoT de Proteus. El Taller IoT es un módulo complementario del Diseñador Gráfico y está pensado para ser utilizado mediante el uso de métodos asociados a los objetos situados en el árbol de proyecto. Todo ello supone la necesidad de disponer de los conocimientos básicos de cómo se utiliza el Diseñador Gráfico. Esta ayuda parte de que el lector los tiene. En la documentación de ayuda suministrada con Proteus se puede encontrar otro manual totalmente dedicado al uso del Diseñador Gráfico.

También es posible utilizar el Taller IoT como un módulo complementario del módulo VSM para Arduino. En este caso, supone la necesidad de disponer de los conocimientos básicos de cómo se utiliza el módulo VSM de Proteus. En la documentación de ayuda suministrada con Proteus se puede encontrar otro manual totalmente dedicado al uso de la simulación del código fuente de los microprocesadores.

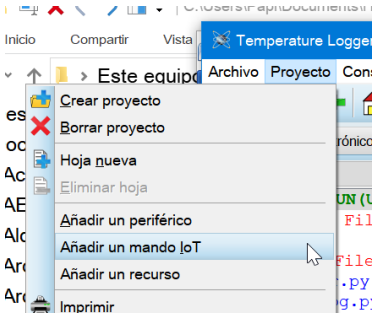


2.-Entorno de trabajo del Taller IoT: el editor de cuadro de mando remoto.

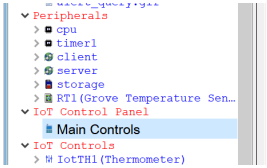
2.1.-Presentación.

En este apartado vamos a ver una introducción al editor de cuadros de mando remoto. Se puede acceder al editor de cuadros de mando desde el árbol del proyecto cuando uno o más mandos IoT se han añadido al proyecto en el que estamos trabajando.

Para añadir un mando IoT utilizaremos la opción 'Añadir un mando IoT' del menú 'Proyecto'.

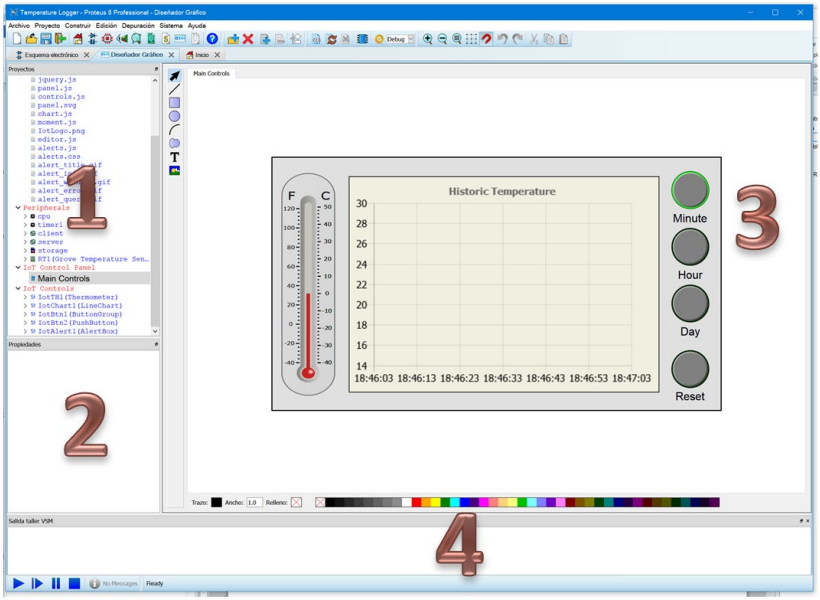


Una vez añadido un mando IoT, desde el árbol del proyecto, podemos abrir el editor pulsando con el ratón sobre el panel de control[2] que deseemos editar.



[2] Un proyecto puede usar más de un panel de control.

Podemos distinguir cuatro áreas principales en la ventana de trabajo cuando estamos utilizando el taller IoT



1. El árbol del proyecto (estructura de archivos).
2. Propiedades del mando (objeto seleccionado).
3. Ventana de edición del cuadro de mando remoto (área de dibujo).
4. Ventana de salida de resultados.

2.2.-El árbol del proyecto.

El árbol del proyecto se encuentra en el lado izquierdo de la pantalla. En el contexto del Taller de IoT sirve para llevar a cabo las siguientes tareas:

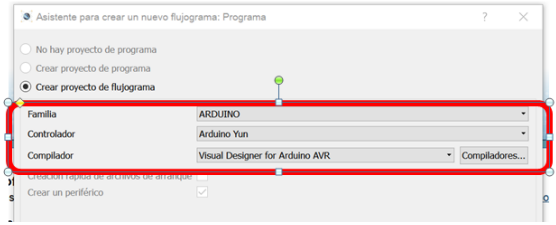
1. Conmutar entre la ventana de edición del cuadro de mando y la ventana del Diseñador Gráfico.
2. Utilizar el menú contextual para añadir mandos IoT al cuadro de mando.
3. ‘Coger y arrastrar’ mandos para posicionarlos en un panel del cuadro de mando.

4. Configurar las propiedades de los mandos utilizando el panel de propiedades de la zona inferior.

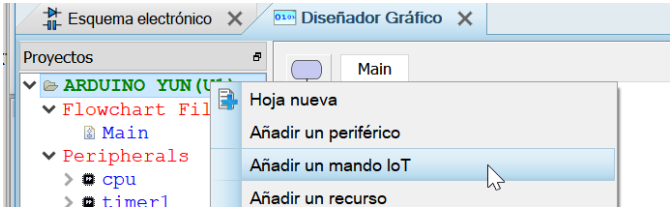
El siguiente ejemplo ilustra brevemente cómo llevar a cabo estas tareas.

2.3.-Añadir un pulsador al cuadro de mando.

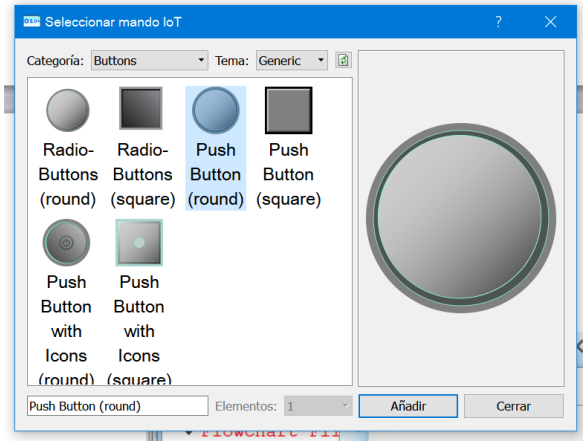
Empezaremos por crear un proyecto de flujograma seleccionando como dispositivo un Arduino Yun.



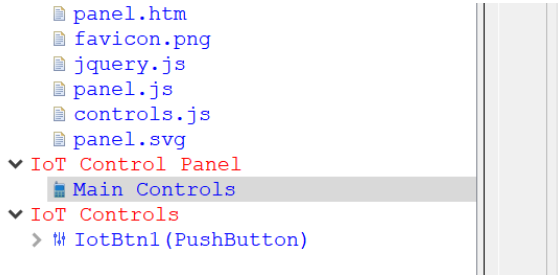
Utilizamos el botón derecho del ratón sobre el equipo Arduino Yun situado en la zona superior del árbol del proyecto y seleccionamos 'Añadir un mando IoT' en el menú contextual que aparece.



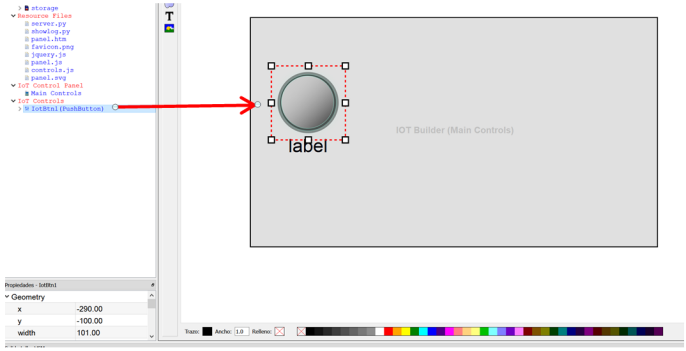
Seleccionamos la categoría 'Buttons' y dentro de ella el 'Push Button (round) y pulsamos el botón 'Añadir'.



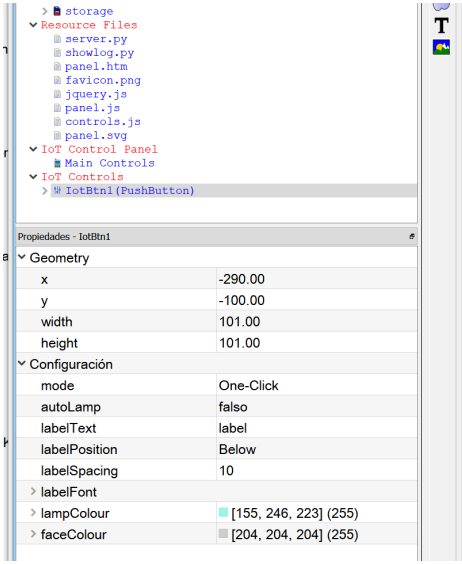
Hacemos doble click sobre el cuadro de mando remoto del árbol de proyectos para abrir la ventana del editor de cuadros de mando.

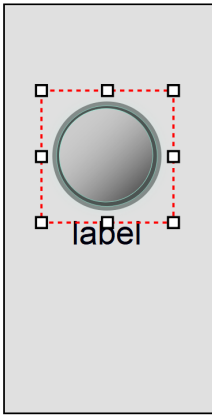


'Cogemos y arrastramos' el mando que acabamos de crear sobre el cuadro de mando.



Pulsamos con el ratón sobre el pulsador que acabamos de crear y podemos editar sus propiedades desde el panel de propiedades situado en la zona inferior izquierda.

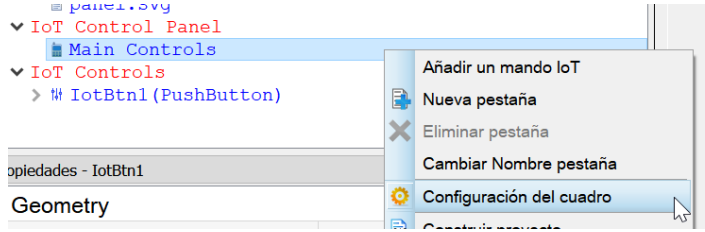




2.4.-La ventana de edición del cuadro de mando.

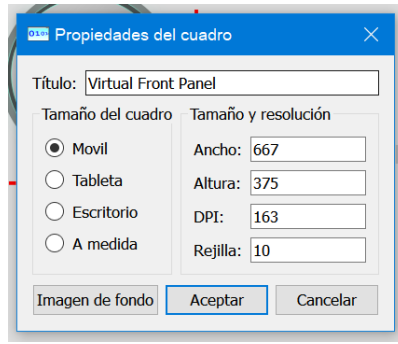
El área de trabajo del cuadro de mando viene determinada por el dispositivo que hemos seleccionado para mostrarlo: puede ser un navegador de un ordenador de sobremesa, un iPad, un iPhone, un Galaxy S6, etc. Para elegir el dispositivo o cambiarlo en un momento dado debemos ejecutar los siguientes pasos:

1.-Botón derecho del ratón sobre el panel de control IoT en el árbol del proyecto para abrir el menú contextual y seleccionar la opción ‘configuración del cuadro’.



Para que aparezca el panel de control IoT en el árbol del proyecto es necesario que nuestro proyecto contenga algún mando. Para añadir un mando IoT utilizaremos la opción 'Añadir un mando IoT' del menú 'Proyecto'.

2.- Podemos seleccionar el tamaño del cuadro desde el panel izquierdo de la ventana de diálogo y definir el tamaño (ancho y altura), resolución (dpi) y tamaño de la rejilla desde el panel derecho. Los valores introducidos deben ser el ancho y alto CSS (pixels CSS independientes del dispositivo).



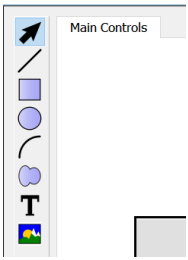
En la página https://mydevice.io/devices/ se encuentra una información muy útil con los tamaños de la mayoría de los dispositivos presentes en el mercado y los valores CSS equivalentes para utilizar a la hora de definir el tamaño del cuadro en el taller IoT.

3.-Cerrar la ventana de diálogo para ajustar el tamaño de cuadro de mando al tamaño adecuado según el dispositivo seleccionado. Hay que tener en cuenta que si ya hemos colocado algún mando sobre el panel y cambiamos de un dispositivo más grande a uno de menor tamaño, podría no haber suficiente espacio para los mandos previamente colocados. Por eso es aconsejable que la primera tarea

que hagamos siempre sea seleccionar el dispositivo que deseamos utilizar para visualizar nuestro cuadro de mando.

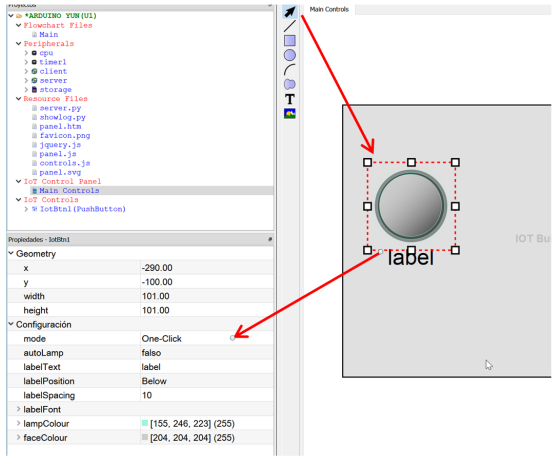
2.5.-Las herramientas de dibujo del cuadro de mando.

En el lado izquierdo de la ventana de edición del cuadro de mando se encuentra la barra de herramientas de dibujo.



La primera nos permite elegir el modo selección. El resto sirve para dibujar líneas, rectángulos, círculos, arcos, superficies de contorno libre, escribir texto e insertar imágenes. La forma de usar estas herramientas es el habitual en todas las herramientas de dibujo.

Después de colocar un gráfico o mando sobre el panel del cuadro de mando, podemos variar su posición o editar sus características desde el panel de propiedades situado en la zona inferior izquierda. Primero tenemos que elegir el modo selección (el primer icono de la barra de herramientas). Hacer click sobre el elemento gráfico. Y, finalmente, editar sus propiedades.

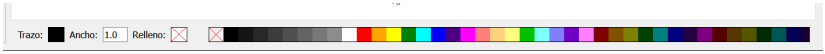


El uso más frecuente de las herramientas de dibujo es para crear algún recuadro y colocar algún texto para agrupar visualmente diferentes mandos por funcionalidades dentro del panel.

> Hay que tener en cuenta que la herramienta texto sólo sirve para crear etiquetas de texto estáticas sobre el panel del cuadro de mando. Si necesitamos utilizar un cuadro de texto para que el usuario pueda enviar órdenes a nuestro dispositivo debemos utilizar un mando IoT específico del tipo texto de entrada o terminal.

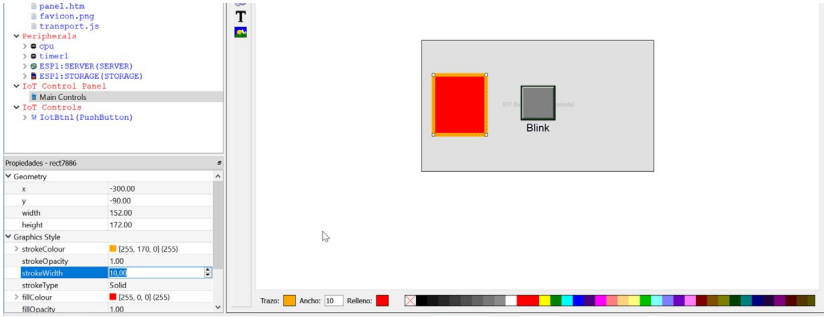
2.6.-La paleta de colores.

La paleta de colores se encuentra ubicada en la zona inferior de la ventana del editor. Se utiliza para fijar el color del trazo, el ancho y el relleno del siguiente objeto gráfico que coloquemos sobre el panel del cuadro.

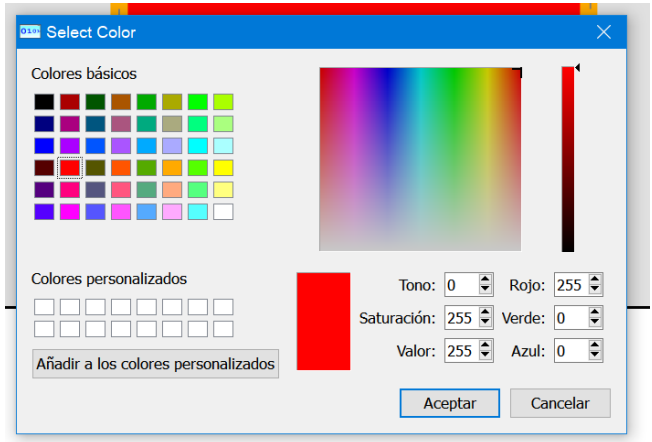


El trazo define el lápiz usado para dibujar el contorno. Tiene dos propiedades, el color y el ancho. Podemos cambiar el color del trazo haciendo doble click sobre el color actual y cambiar el ancho utilizando el cuadro de edición. La siguiente imagen muestra un

ejemplo de cómo hemos definido el color y el ancho del trazo de un gráfico del tipo rectángulo.



El relleno se utiliza para definir el color de fondo del objeto gráfico Se puede definir si el fondo es transparente (icono con un aspa roja sobre un fondo blanco) o de un determinado color. Podemos escoger el color de fondo desde la paleta situada en la zona inferior o haciendo doble click sobre el cuadro relleno para abrir la ventana de diálogo de selección de color.

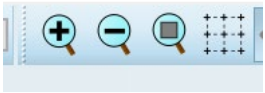


Debemos tener en cuenta que los cambios efectuados utilizando esta técnica fijan los valores por defecto y afectarán a los siguientes objetos gráficos que situemos sobre el panel del cuadro de control. Si deseamos cambiar el trazo o relleno sólo de un objeto gráfico

debemos seleccionarlo primero y cambiar los valores utilizando el panel de propiedades.

2.7.-Zoom.

Podemos aumentar o disminuir el zoom o ajustar automáticamente el tamaño para que incluya todo el panel, utilizando los iconos situados en la barra de herramientas superior.

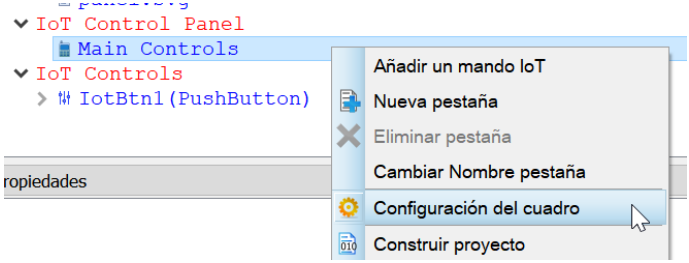


También es posible utilizar los atajos de teclado F6, F7 y F8 para llevar a cabo las mismas tareas.

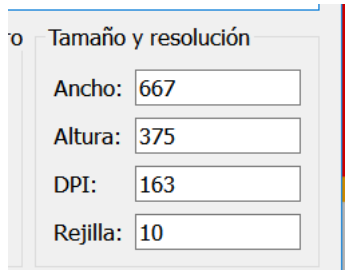
2.8.-La rejilla y el forzado.

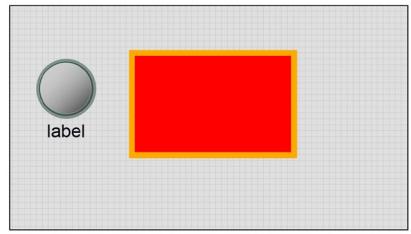
Podemos mostrar u ocultar una rejilla sobre el área de dibujo para que sirva de ayuda en el posicionamiento de nuevos mandos. También se puede forzar para que el cursor del ratón se posicione sobre un nodo de la rejilla actual.

Podemos definir la rejilla desde el menú contextual del panel de control utilizando la opción 'Configuración del cuadro'.

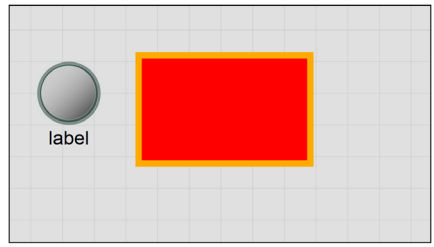


En la ventana de diálogo que aparece, podemos seleccionar el espaciado de la rejilla desde la opción 'Rejilla' del panel 'Tamaño y resolución'. Por defecto el valor es 10.



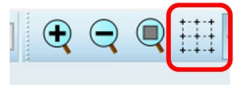


Si fijamos un valor más grande, la rejilla tendrá menos cuadros lo que significa menos puntos de forzado para colocar dispositivos.



Por el contrario, si elegimos valores más pequeños, tendremos más libertad para situar los objetos en el lugar que queramos pero será más difícil alinear los mandos de forma precisa. Nuestra sugerencia es dejar el valor por defecto al menos que tengamos que hacer algo que requiera cambiarlo.

Una vez configurado el tamaño de la rejilla, podemos mostrarla u ocultarla con el icono de la barra de herramienta superior.



3.-Tutorial 1: Encender y apagar un led.

3.1.- Objetivo.

Este tutorial pretende ser una introducción al proceso de diseñar, desarrollar e implementar un proyecto utilizando el Taller IoT. La funcionalidad del dispositivo final de este proyecto ha sido reducida tanto cómo es posible para lograr fijar el foco de atención del lector en el proceso de trabajo que va desde el arranque de Proteus hasta el control de nuestro equipo Arduino Yun usando el teléfono móvil.

En este tutorial asumimos que el lector está familiarizado con la programación mediante flujogramas utilizada por el Diseñador Gráfico. Si no fuera así, es recomendable consultar primero los diferentes tutoriales presentes en las ayudas de Proteus sobre el Diseñador Gráfico.

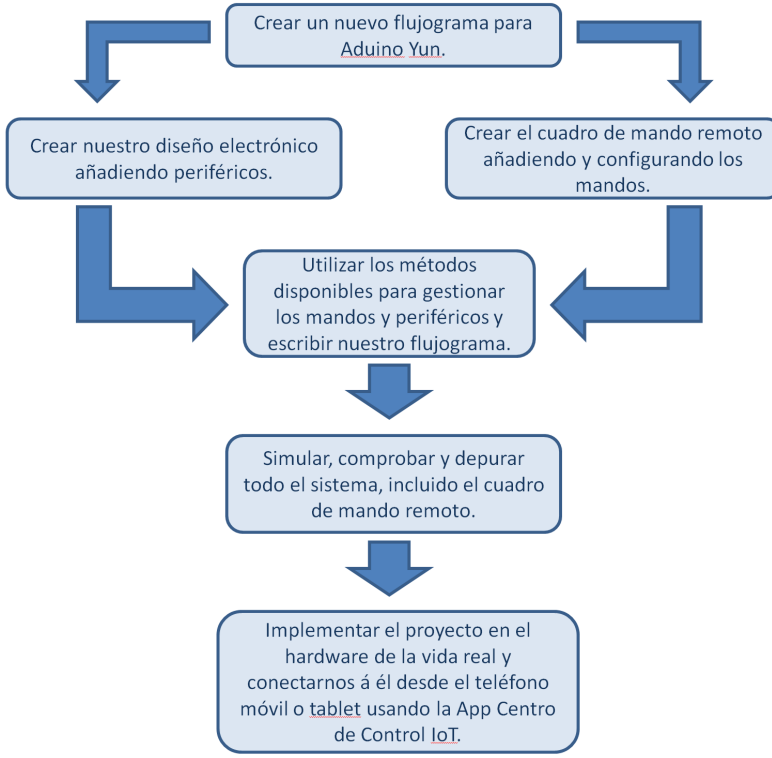
3.2.-Terminología utilizada.

Nos vamos a referir al equipo resultante al final del diseño con el término 'dispositivo'. El dispositivo puede ser tanto un 'equipo virtual' durante la fase de diseño del esquema electrónico y simulación, como un 'equipo real' al final de la implementación del proyecto sobre el hardware de la vida real. En el caso de este tutorial, el dispositivo va a ser un equipo Arduino Yun ampliado con las tarjetas de expansión, los módulos accesorios independientes y los módulos de sensor Grove que deseemos conectarle.

Y vamos a utilizar el término 'cuadro de mando remoto' para referirnos al equipo que utilizaremos para monitorizar y gobernar nuestro dispositivo a distancia. En el caso de este tutorial, el cuadro de mando remoto va a ser un cuadro de mando remoto virtual durante la fase de diseño y simulación, y un teléfono móvil o tableta al final de la implementación en la vida real.

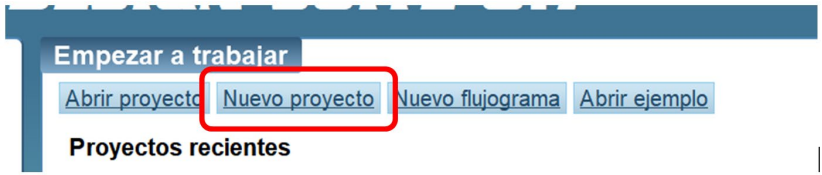
3.3.-El proceso de diseño del dispositivo.

El proceso del diseño de nuestro dispositivo, desde el comienzo del proyecto hasta la implementación final en equipos de la vida real, sigue una serie de pasos que se resumen en el diagrama de flujo siguiente, aunque el orden pueda variar según las circunstancias de cada proyecto.



3.4.-La configuración del proyecto.

Ejecutamos el asistente para crear un nuevo flujograma desde la pantalla de inicio de Proteus. Es necesario que indiquemos el nombre del proyecto y la carpeta donde se guardará.

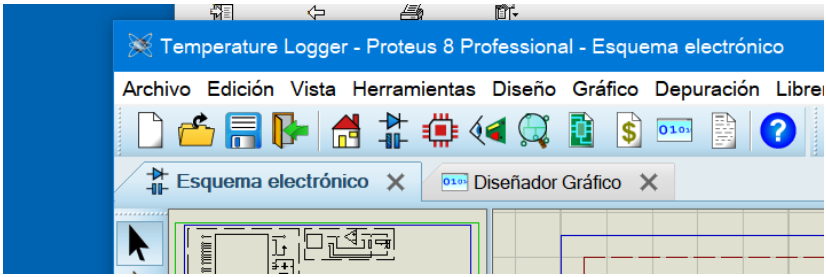


En la siguiente ventana de diálogo que aparece, creamos nuestro proyecto de flujograma indicando el dispositivo que queremos utilizar. En nuestro caso, debemos elegir la familia Arduino, el controlador Arduino Yun y el compilador Diseñador Gráfico para Arduino AVR.

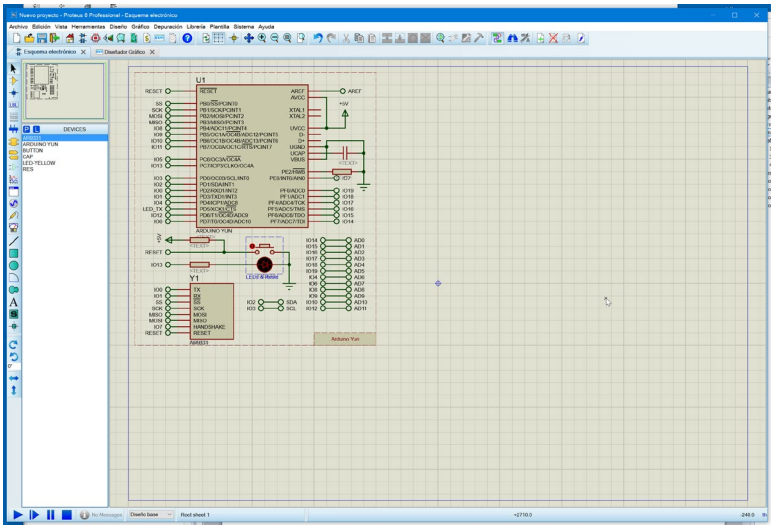


En cuanto lo hayamos hecho, se nos mostrará una ventana con el resumen de característica del proyecto que vamos a crear. Lo validamos y terminaremos esta primera fase de creación del proyecto.

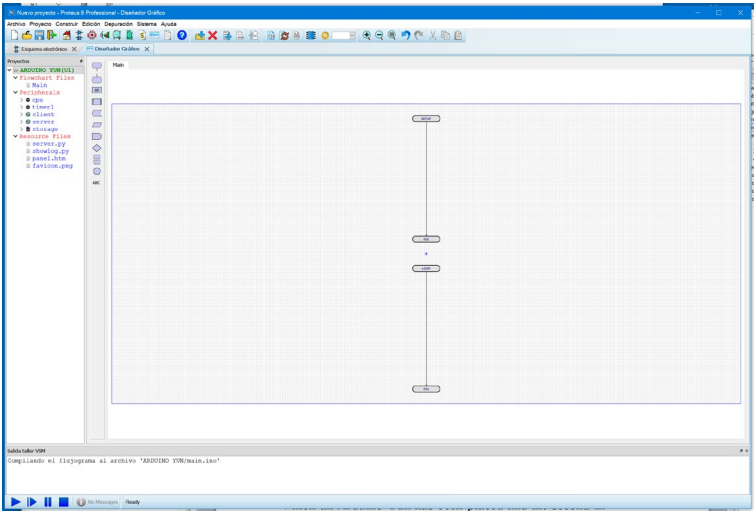
Al finalizar el asistente, ya tendremos disponibles dos pestañas de Proteus. En una encontraremos el esquema electrónico y en la otra el diseñador gráfico.



En la pestaña de esquema electrónico podremos encontrar nuestro Arduino Yun listo para trabajar con él.



Y en la pestaña del diseñador gráfico un armazón de programa básico con las subrutinas ‘loop’ y ‘setup’ para gobernar un equipo Arduino.

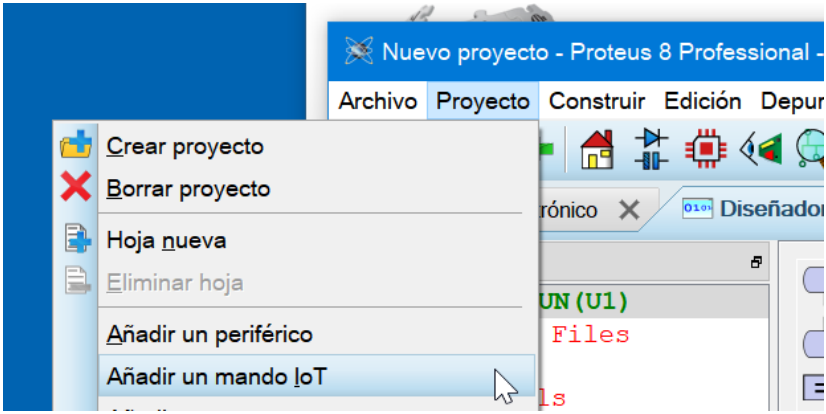


3.5.-Diseño del cuadro de mando remoto.

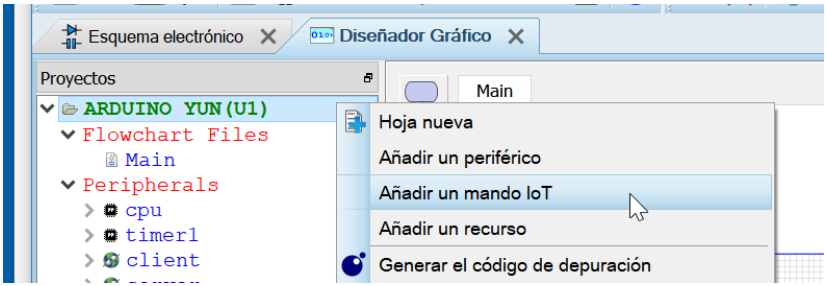
El cuadro de mando remoto es el equipo que utilizaremos para monitorizar y gobernar nuestro dispositivo. Es necesario que contenga los distintos mandos que serán necesarios para enviar órdenes y para monitorizar los datos recibidos de nuestro dispositivo.

Este primer tutorial tiene como objetivo un primer acercamiento a la herramienta. Un objetivo muy similar a los tutoriales del tipo 'hola mundo' con los que habitualmente se empieza a trabajar en cualquier lenguaje de programación. Por ese motivo, nuestro proyecto únicamente encenderá o apagará un led utilizando un pulsador situado en el cuadro de mando remoto.

Empezaremos añadiendo un mando IoT utilizando la opción 'añadir un mando IoT' del menú 'Proyecto'.

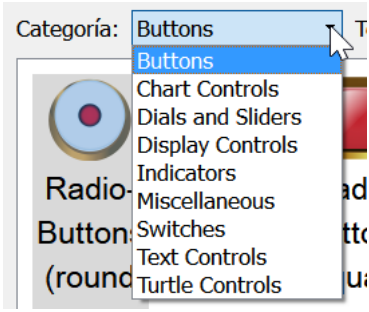


O usando la opción 'añadir un mando IoT' del menú contextual del equipo Arduino Yun del árbol del proyecto.

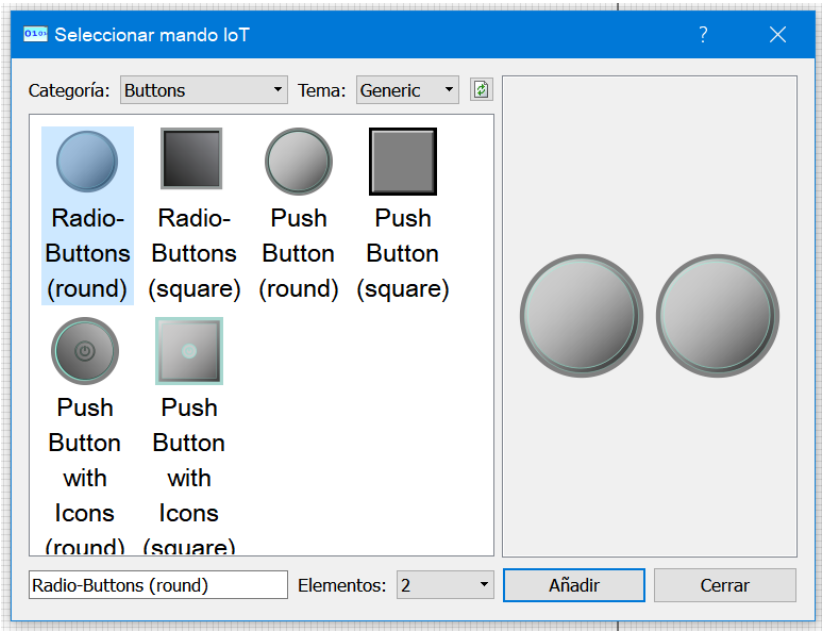


En la ventana de diálogo que aparece vamos a seleccionar nuestro pulsador. Tenemos a nuestra disposición un gran número de mandos de naturaleza muy diferentes para ser usados en nuestro cuadro de mando remoto. Para facilitar su selección, los diferentes mandos se agrupan en diferentes categorías y temas.

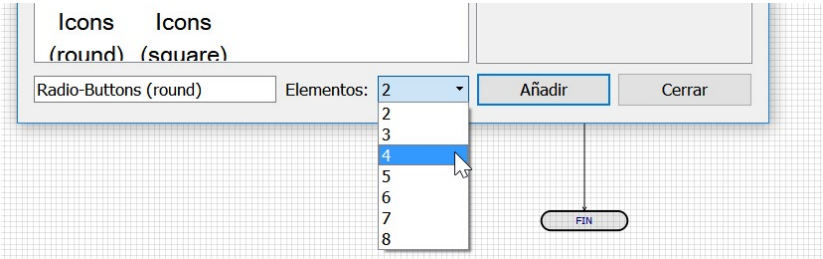
Las categorías agrupan mandos de la misma funcionalidad y los temas agrupan los mandos de un mismo aspecto (generic, metallic, modern, etc.).



Seleccionaremos la categoría ‘Buttons’ y dentro de ella el mando ‘Radio Buttons (round)’. Luego seleccionamos el tema ‘Generic’. Todavía no usaremos el botón ‘añadir’ ni ‘cerrar’.



Al haber seleccionado la opción ‘Radio Buttons (round)’, en la zona inferior nos pregunta el número de pulsadores que deseamos posicionar en nuestro cuadro de mando.



Si hubiéramos utilizado cualquiera de las opciones disponibles de ‘Push Button’ el sistema entiende que sólo deseamos posicionar un pulsador y este cuadro de diálogo aparecerá deshabilitado.

Volveremos a seleccionar el mando ‘Push Button (round)’, puesto que es el que necesitamos usar en nuestro proyecto. Para acabar, una vez seleccionado, pulsamos el botón ‘Añadir’.

Más adelante veremos que el mando seleccionado contiene una propiedad denominada ‘Auto Lamp’ que se utiliza para controlar el aspecto del pulsador. Cuando el pulsador está accionado, esta propiedad tiene el valor ‘True’ y el pulsador aparece con un halo de color. Cuando el pulsador está liberado, la propiedad toma el valor ‘False’ y el halo de color desaparece. La siguiente imagen muestra un ejemplo del pulsador sin accionar a la izquierda y accionado (con el halo de luz) a la derecha.

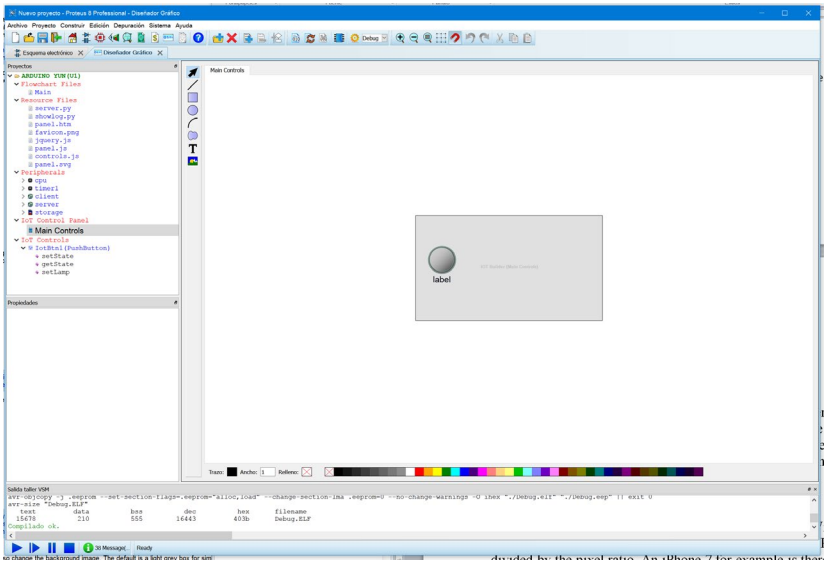


No debemos confundir esta propiedad que sólo sirve para controlar el aspecto visual del pulsador, con la lógica de control que llevemos a cabo en nuestro código para controlar las acciones que hay que hacer cuando se acciona o libera el pulsador. La propiedad ‘Auto Lamp’ sólo es una cuestión estética y no guarda relación con la programación.

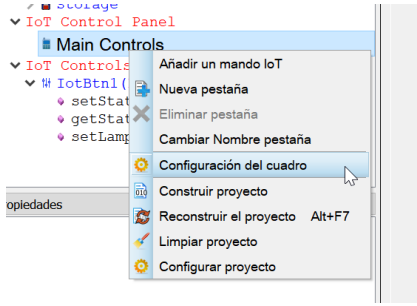
Al haber añadido un mando del tipo pulsador, podemos comprobar que se han producido algunos cambios importantes en el árbol del proyecto. Ahora existe un mando IoT (nuestro pulsador) situado al final de la estructura en una rama denominada ‘IoT Controls’. Y justo encima otra rama denominada ‘IoT Control Panel’ con un panel del cuadro llamado ‘Main Controls’. En la rama ‘Resources Files’ también tendremos un par de ficheros nuevos.

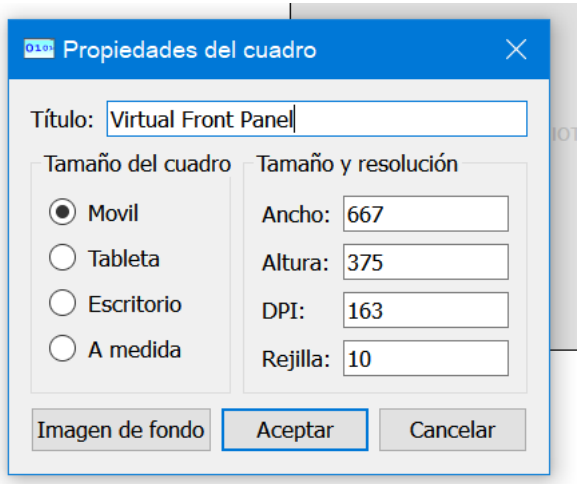


Si hacemos un doble click sobre la rama ‘Main Controls’ se abrirá la ventana de edición del panel del cuadro.



La ventana de edición muestra lo que será el cuadro de mando remoto de nuestro dispositivo. Por defecto está configurado para ser del tamaño de un teléfono móvil, pero podemos cambiarlo. Para ello pulsaremos el botón derecho del ratón sobre la rama del panel y en el menú contextual que aparece seleccionaremos la opción 'Configuración del cuadro'



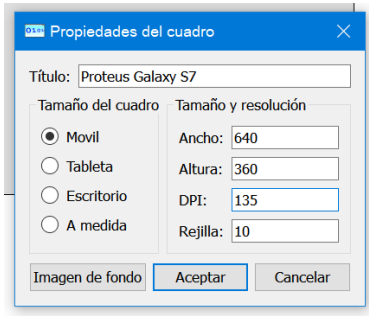


En la página https://mydevice.io/devices/ se encuentra una información muy útil con los tamaños de la mayoría de los móviles y tabletas presentes en el mercado y los valores CSS equivalentes para utilizar a la hora de definir el tamaño del cuadro en el taller IoT. Los valores introducidos para el ancho y la altura deben ser el ancho y alto CSS (columnas 4 y 5) que aparecen en esta tabla. Y el valor del campo DPI es el valor PPI físicas (columna 7) dividido por el ratio de pixels (columna 6).

name	phys. width	phys. height	CSS width	CSS height	pixel ratio	phys. ppi	CSS ppi
Apple iPhone X	1125	2436	375	812	3	458	288
Apple iPhone 6+, 6s+, 7+, 8+	1080	1920	414	736	3	401	288
Apple iPhone 7, iPhone 8	750	1334	375	667	2	326	192

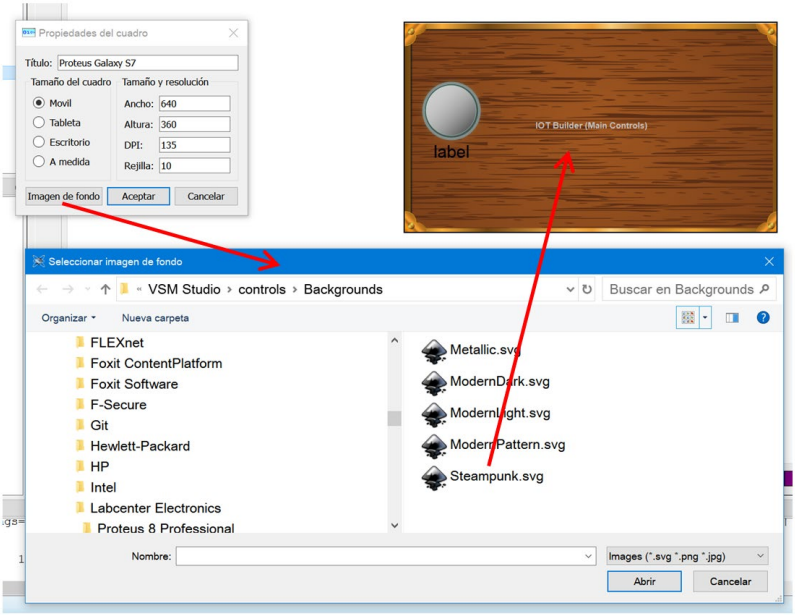
Por ejemplo, para el caso del Iphone 7 tendríamos unos valores de ancho y alto de 375 x 667. Y el valor de DPI sería 362/2 = 163.

En nuestro ejemplo queremos seleccionar un Galaxy S7 como el teléfono móvil que usaremos de cuadro de mando remoto de nuestro dispositivo. Así que introduciremos los siguientes valores.

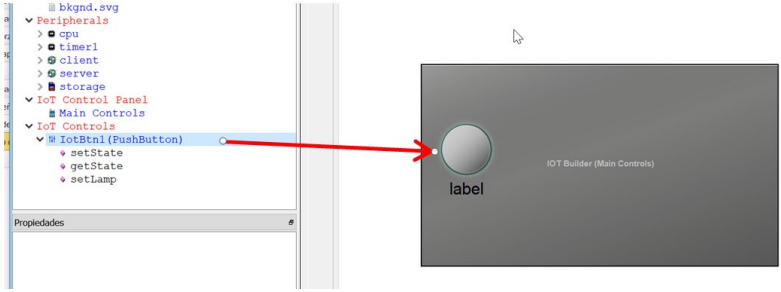


Hay que tener cuidado cuando cambiamos de una configuración de escritorio a una tableta o a un móvil a la mitad de un proyecto. Si cambiamos a un cuadro de mando remoto de dimensiones menores de las que partimos y ya tiene colocados mandos en el mismo, los mandos no se ajustan automáticamente al nuevo diseño y será un trabajo pesado volver a recolocar todos nuestros mandos en la distribución original. Incluso algunos mandos pueden quedar fuera del área de trabajo. La mejor opción es empezar por seleccionar nuestra configuración del cuadro de mando remoto en el momento de la creación del proyecto.

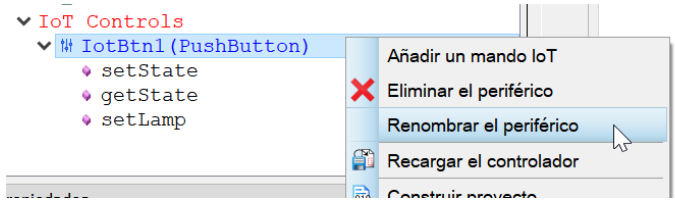
También podemos cambiar la imagen de fondo que se utilizará en nuestro cuadro de mando remoto. Por defecto se utiliza un color gris claro para simplificar. Pero es posible seleccionar otro fondo que se ajuste más con el tema general (aspecto) que hemos seleccionado para nuestro cuadro de mando remoto. Proteus se suministra con unas pocas imágenes de fondo de ejemplo, pero podemos añadir todas las que deseemos. Sólo tenemos que situar el archivo de extensión 'jpg', 'svg' o 'png' que deseemos utilizar en la carpeta 'C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\ VSM Studio\controls\Backgrounds' y seleccionarla en nuestro proyecto usando el botón 'imagen de fondo'. En el siguiente ejemplo hemos seleccionado la imagen 'Steampunk.svg'.

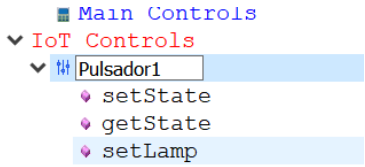


Para colocar nuestro pulsador sobre nuestro panel del cuadro, sólo tenemos que ‘coger y arrastrar’ el pulsador desde el árbol del proyecto al panel.

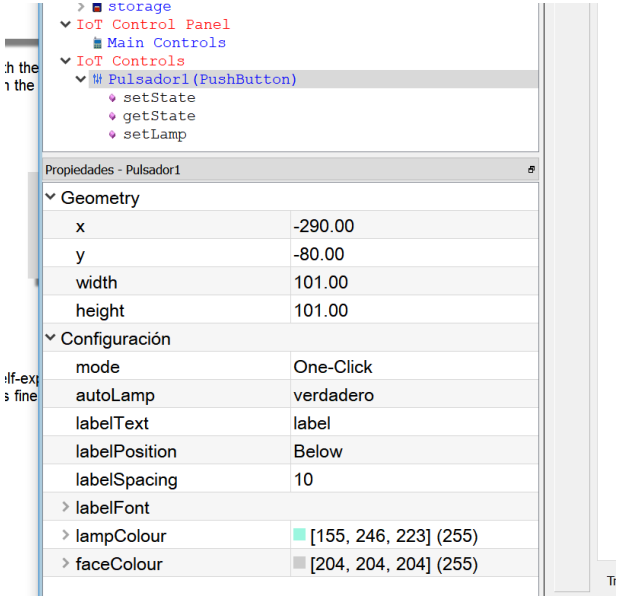


Para cambiar el nombre del pulsador, utilizaremos el botón derecho del ratón sobre la rama del pulsador en el árbol de proyecto para abrir el menú contextual y seleccionaremos la opción ‘Renombrar el periférico’



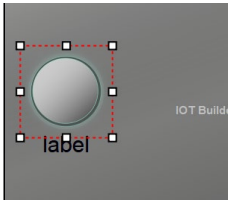


Proteus nos permite modificar prácticamente todo aquello que podamos necesitar configurar de nuestro pulsador utilizando el panel de propiedades que se encuentra debajo del panel del árbol del proyecto. En este panel se muestran todos los parámetros configurables del objeto que se encuentra actualmente seleccionado en la ventana de edición.

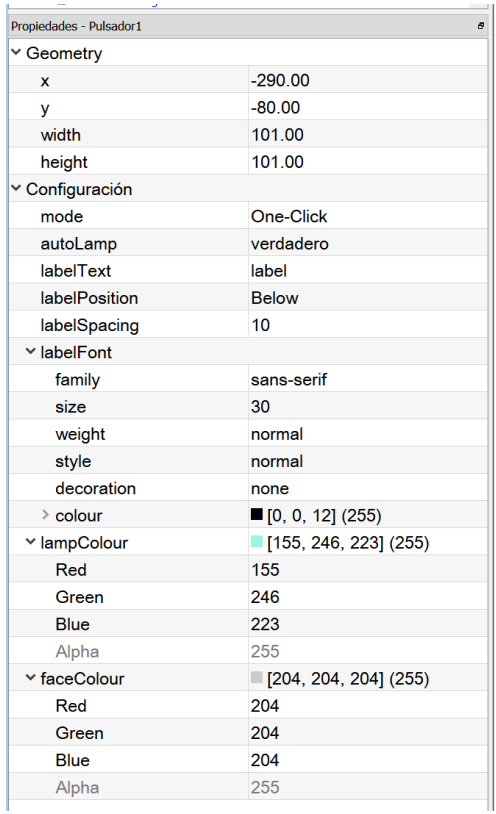


Si el pulsador no se encuentra seleccionado el panel de propiedades aparecerá en blanco. Es fácil saber qué mando está seleccionado en

cada momento porque aparece rodeado de un recuadro de trazos rojos con manejadores para ajustar su tamaño. Para seleccionar un determinado objeto sólo tenemos que pulsar con el ratón sobre él.



Las propiedades de un pulsador son tan evidentes por sí mismas que no es necesario detenernos mucho tiempo en este punto. La sección ‘Geometry’ controla tamaño y posición. La sección ‘Configuración’ controla el tipo de pulsador (para nuestro ejemplo elegir ‘one-click’ es lo más adecuado), la propiedad ‘autoLamp’ y unas pocas propiedades para controlar la etiqueta y los colores del halo de luz.



3.6.-La propiedad ‘AutoLamp’.

Si le asignamos un valor ‘verdadero’ a la propiedad ‘AutoLamp’, el pulsador funcionará automáticamente como un pulsador de modo ‘toggle’ (conmutador de dos posiciones) y el halo de luz se encenderá o apagará según el estado actual.

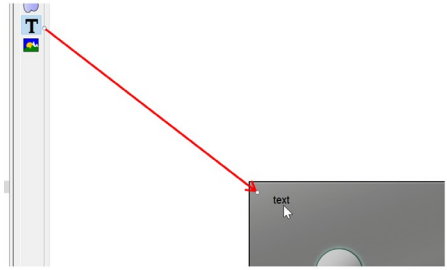
Si le asignamos un valor ‘falso’ necesitamos utilizar la función setLamp() en nuestro flujograma para controlar el halo de luz. Este método tiene la ventaja de que el halo es gobernado por el propio dispositivo y no por el gestor del aspecto del cuadro de mando remoto, lo que nos garantiza que el dispositivo y el panel de control usan la misma información de forma sincronizada en todo momento.

Si el código no utiliza para nada esta información y sólo se trata de un aspecto meramente ornamental para ayudar al usuario a identificar si ha actuado sobre el pulsador, asignar el valor ‘verdadero’ a la propiedad ‘autoLamp’ simplifica el trabajo de programación.

3.7.-Rótulos.

Una vez que hemos dejado nuestro pulsador a nuestro gusto, podemos añadir un rótulo o una imagen a nuestro panel del cuadro de mando remoto. Para ello podemos utilizar las herramientas disponibles en la barra lateral del área de trabajo.

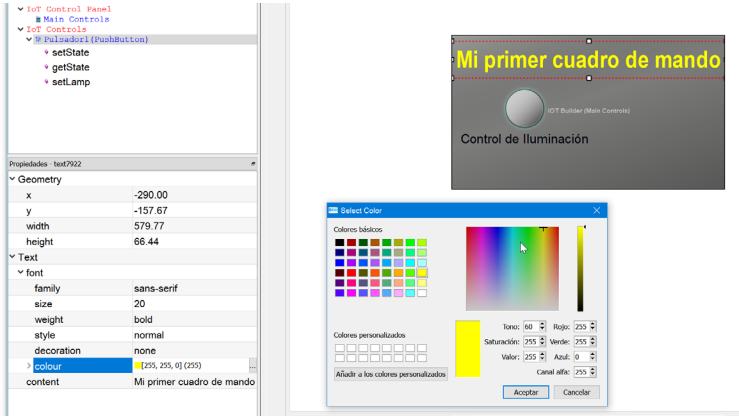
Por ejemplo, podemos seleccionar la herramienta ‘modo colocar texto’ y colocar el cursor del ratón donde deseamos que se coloque el rótulo.



Una vez posicionado el rótulo, podemos seleccionarlo y ajustarlo al tamaño que deseemos que tenga.

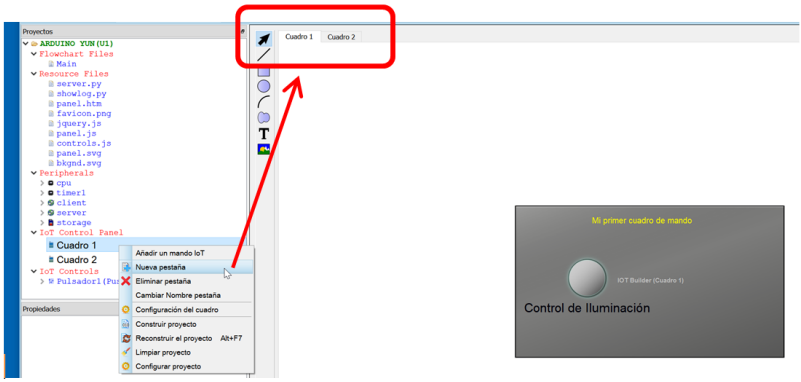


Y, para terminar con el rótulo, podemos cambiar la fuente, el color y el texto desde el panel de propiedades situado debajo del árbol del proyecto.

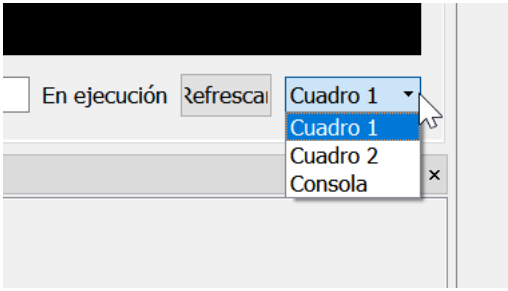


3.8.-Múltiples paneles.

Si necesitamos más de un panel, podemos crear nuevas pestañas desde el menú contextual de la rama de nuestro cuadro de mando remoto seleccionando la opción 'nueva pestaña'. Una vez creada podemos asignarle el nombre que deseemos.



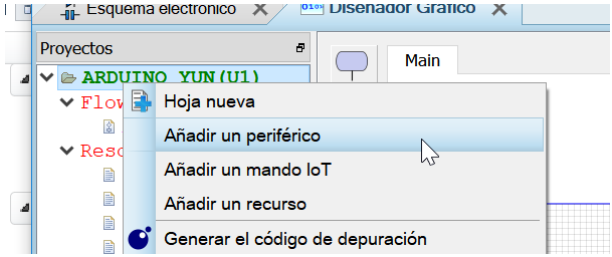
Cuando pongamos en marcha la simulación podremos elegir qué cuadro de mando remoto deseamos utilizar en la zona inferior de la ventana.



3.9.-Escritura del código.

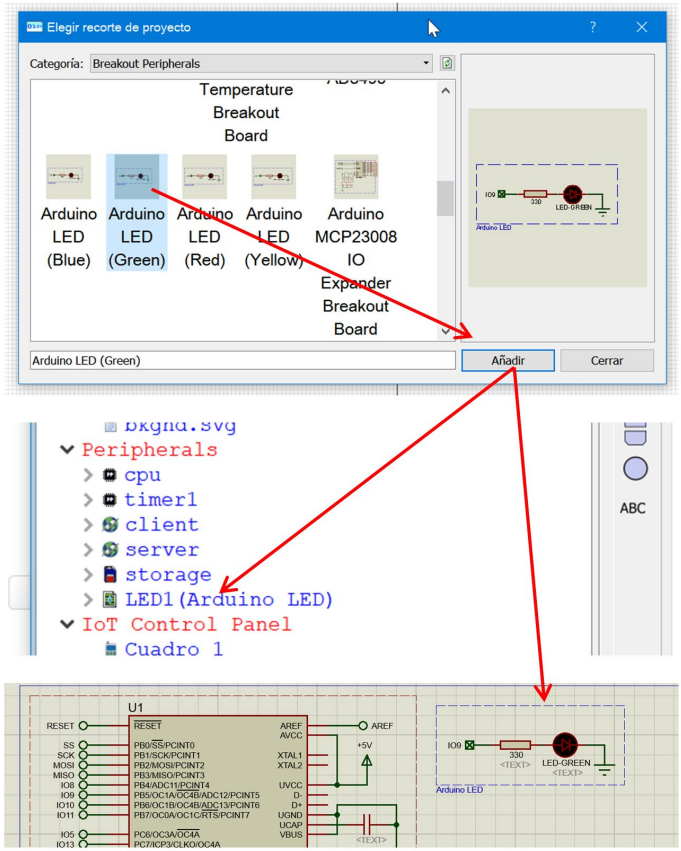
Una vez que hemos construido nuestro cuadro de mando y colocado el pulsador podemos empezar a gestionar cómo debe interactuar con nuestro dispositivo. Puesto que deseamos que cada vez que se actúe sobre el pulsador se encienda un led, lo primero que tenemos que hacer es añadir un led (del color que deseemos) a nuestro proyecto. Para hacerlo, lo tenemos que añadir en nuestro diseño electrónico. No debemos olvidar que el diseño electrónico es una versión virtual de lo que finalmente será el equipo en la vida real.

Podemos llevar a cabo esta tarea desde la opción ‘añadir un periférico’ del menú contextual de la rama raíz del árbol de proyectos o desde la opción ‘añadir un periférico’ del menú ‘Proyecto’ de la pestaña ‘Diseñador Gráfico’

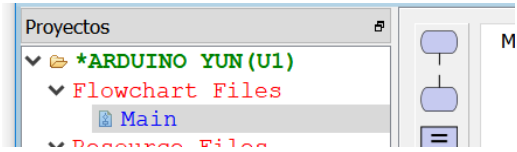


Con esta operación Proteus llevará a cabo dos tareas: añadirá un led (azul, verde, amarillo o rojo) a nuestro esquema en la pestaña diseño

electrónico y también añadirá una nueva rama a nuestro árbol de proyecto en la pestaña diseñador gráfico.

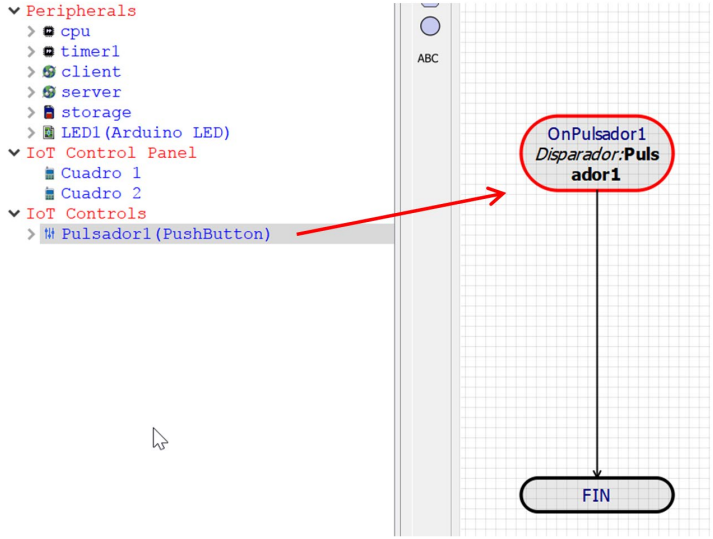


Para ir al flujograma, solo tenemos que hacer un doble click con el ratón sobre la rama ‘main’ del árbol del proyecto.



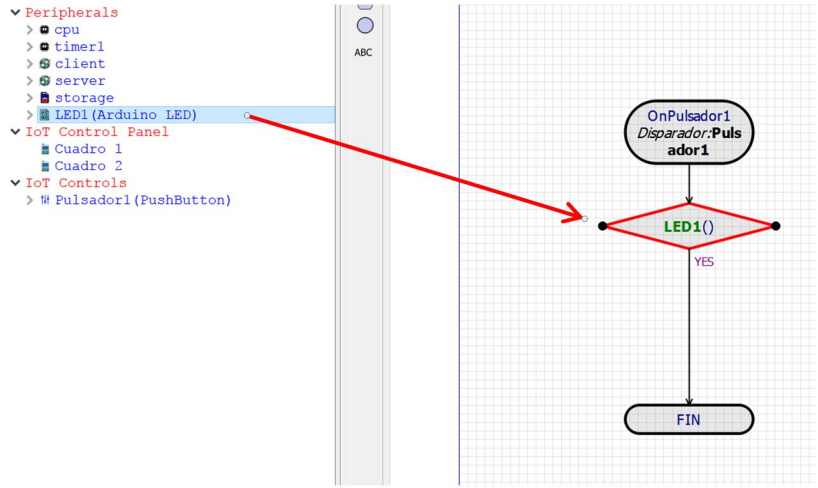
A continuación, tenemos que escribir el código que responda a cada actuación sobre el pulsador y que se encargará de encender o apagar el led. En el Diseñador Gráfico tenemos que crear un evento

que responda a cada actuación que el usuario lleve a cabo con el pulsador. Lo podemos hacer muy rápidamente 'cogiendo y arrastrando' desde la rama donde se encuentra el pulsador en el árbol de proyecto hasta el flujograma.



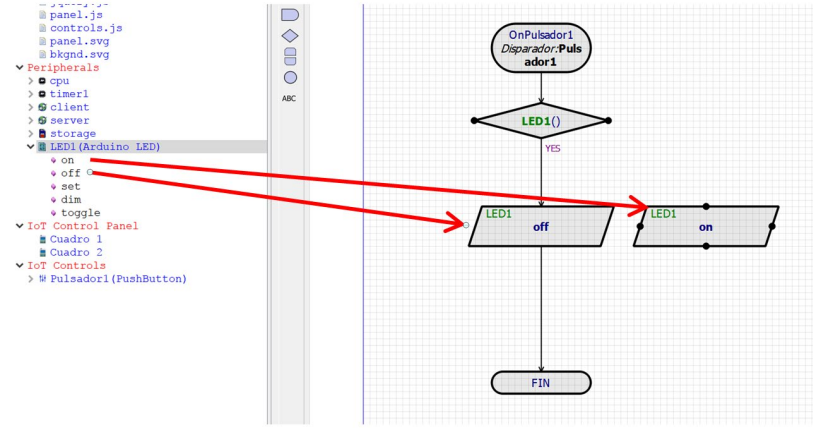
Este es el camino más corto para añadir una subrutina de respuesta al evento que se desencadena cuando el usuario actúa sobre el pulsador. Cada vez que el usuario actúe sobre el pulsador se ejecutará este trozo de código.

Como deseamos que el led se encienda o apague en función de su estado actual, tendremos que volver a utilizar la técnica 'coger y arrastar' para soltar en el flujograma la rama del led en el árbol de proyecto. La única precaución que tenemos que tener, es soltarlo dentro de la subrutina para crear el bloque de decisión tipo 'if' en el lugar adecuado.

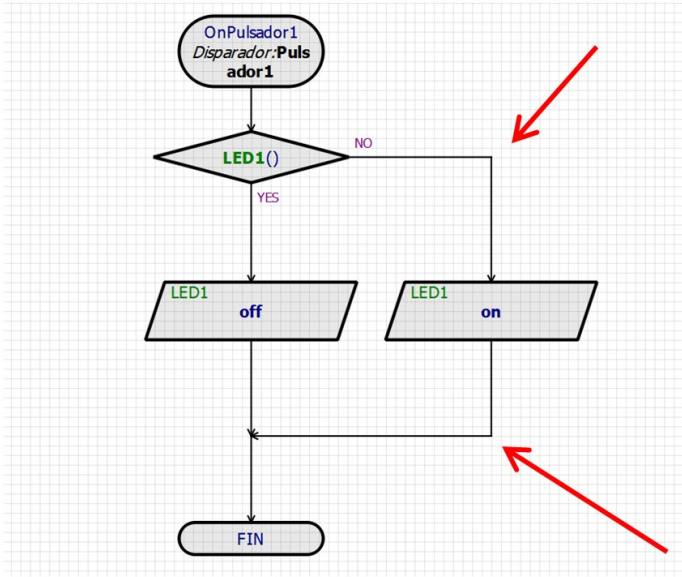


Acabamos de comprobar la potencia de trabajar con la técnica que nos brinda el Diseñador Gráfico. Conviene que analicemos con detalle lo que ha ocurrido para que dominemos completamente la técnica y la aprovechemos en todo su potencial. Cada vez que 'cogemos y arrastramos' uno de los periféricos que hemos añadido a nuestro proyecto (sin necesidad de seleccionar uno de los métodos asociados a este elemento), obtenemos la función asociada por defecto a dicho elemento. En el caso de un led, esta función por defecto es un bloque de decisión que abre dos ramas del flujograma dependiendo de si el led se encuentra en este momento encendido o apagado. Podemos encontrar toda la información referente a las funciones asociadas por defecto a cada elemento en la ayuda del Diseñador Gráfico.

Ahora que ya tenemos el bloque de decisión que vigila si el led está encendido o apagado, todo lo que necesitamos hacer es encender el led cuando se encuentra apagado o apagarlo en caso contrario. Para ello, expandimos la rama del led en el árbol del proyecto y 'cogemos y arrastramos' los métodos apropiados (on y off) al lugar indicado del flujograma.

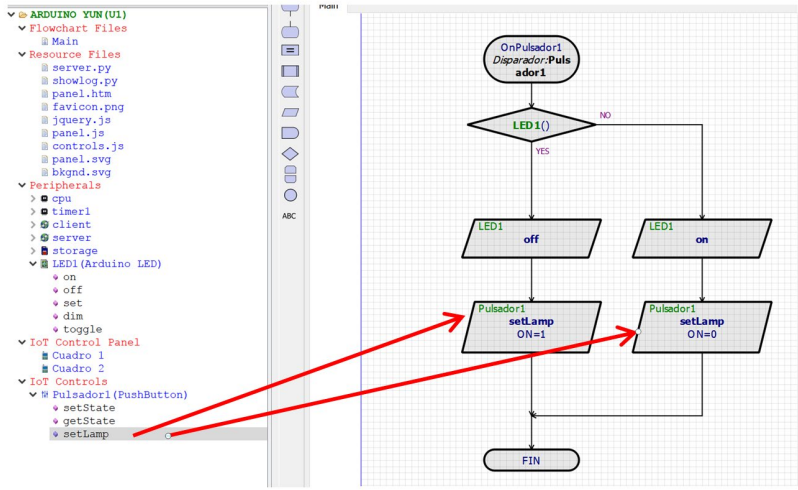


Para terminar añadimos las conexiones desde el bloque IF al bloque LED1-ON que enciende el led y desde éste al bloque FIN de la subrutina.



Como ‘toque final’, vamos a añadir una indicación en el cuadro de mando remoto que indique si el led se encuentra encendido o apagado. Lo podemos realizar fácilmente utilizando nuevamente la

técnica ‘coger y arrastrar’. Esta vez, usando el método ‘SetLamp’ asociado al pulsador como se muestra en la imagen siguiente.



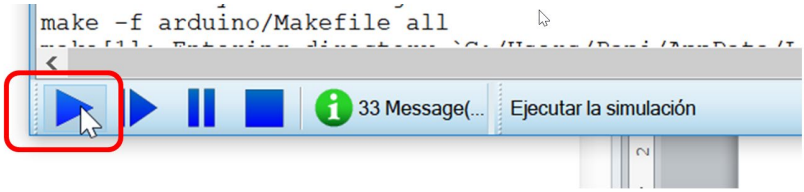
Y ya hemos terminado. Con cinco bloques de flujograma hemos escrito el código que nos permite controlador un led conectado a un equipo Arduino Yun desde nuestro teléfono móvil o tableta.

Nuestro siguiente paso será simular y comprobar el correcto funcionamiento del programa.

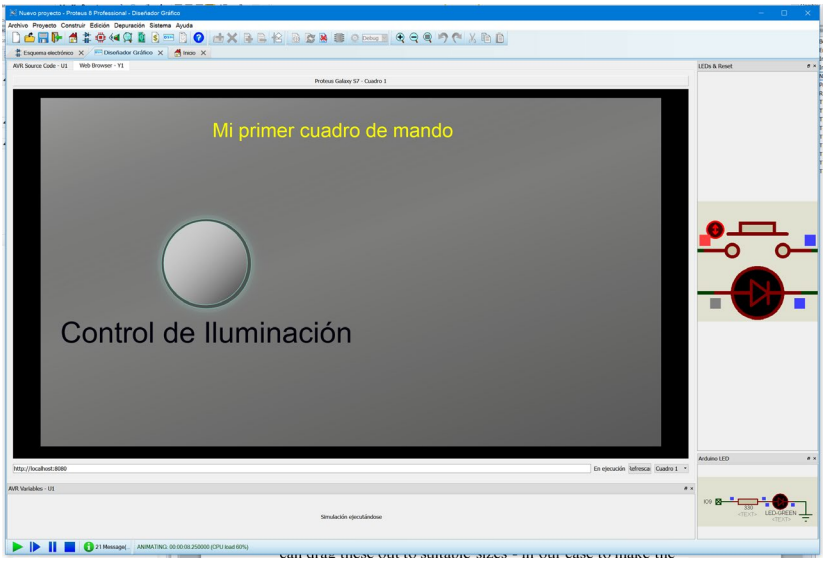
3.10.-Simulación y comprobación del código.

Podemos llevar a cabo una rápida comprobación de que no hemos cometido ningún error durante la escritura del código. Vamos a simular lo que ocurre cuando accionamos el pulsador desde nuestro cuadro de mando remoto.

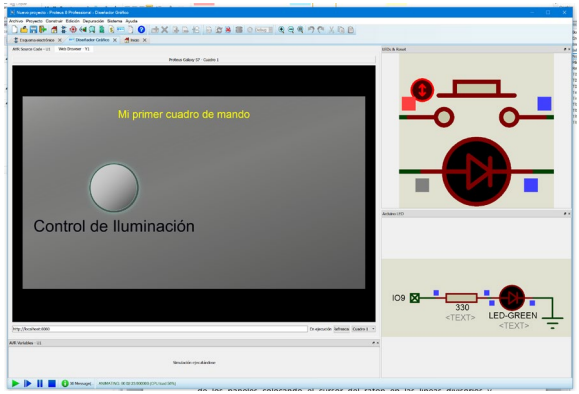
Como se hace siempre en Proteus, pondremos en marcha la simulación utilizando el botón ‘play’ del cuadro de control de la simulación situado en la zona inferior de la ventana de Proteus.



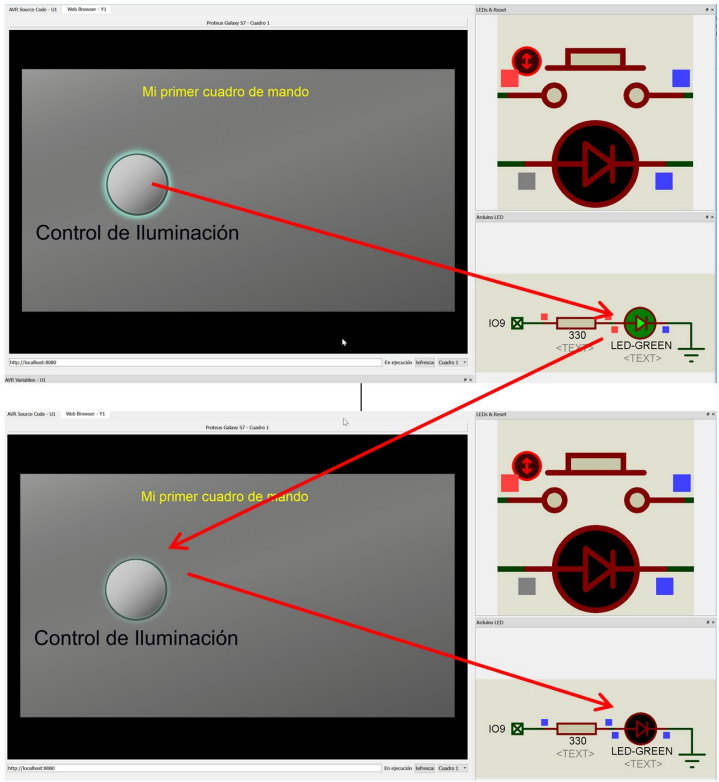
Al finalizar la compilación y empezar la ejecución de la simulación, en nuestra pantalla encontraremos varios paneles con los equipos virtuales. En la parte izquierda (y de mayor tamaño) se encuentra nuestro panel de control remoto. Y en la parte derecha, dos ventanas con dos trozos del esquema electrónico (el conjunto pulsador más led de la placa principal Arduino y la tarjeta de expansión con el led conectado).



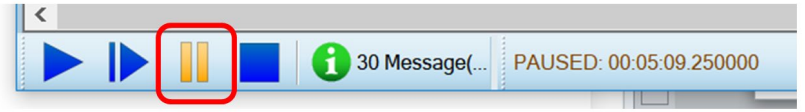
Para realizar una simulación más cómoda ajustaremos los tamaños de los paneles colocando el cursor del ratón en las líneas divisorias y arrastrando el cursor para dejarlos del tamaño deseado. En nuestro caso nos interesa agrandar el panel donde se visualiza la tarjeta de expansión del led.



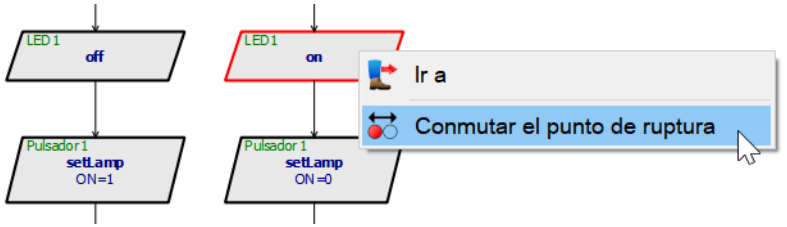
Todo lo que tenemos que hacer es accionar el pulsador y comprobar que el led se enciende. A continuación, volvemos a accionar el pulsador y comprobamos que el led se apaga.



Supongamos que hemos cometido algún error y el programa no funciona como esperamos y que necesitamos investigar cuál es el origen del error. Podemos presionar la tecla 'pausa' del cuadro de control de la simulación.

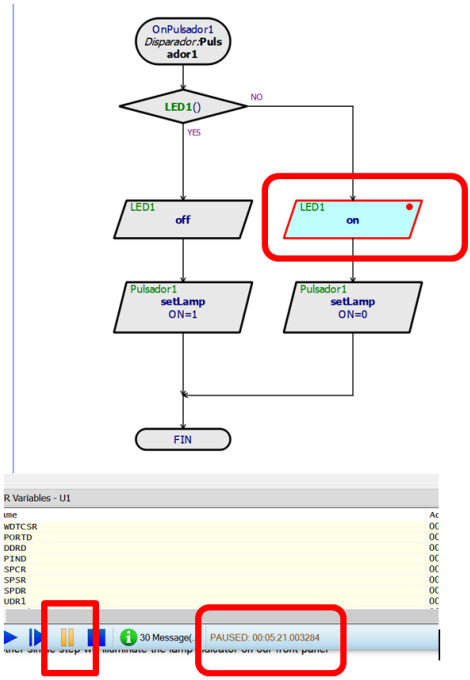


A continuación, vamos a fijar un punto de ruptura en nuestro flujograma. En este ejemplo, lo situaremos en el método que provoca el encendido del led desde el menú contextual de ese bloque de flujograma.



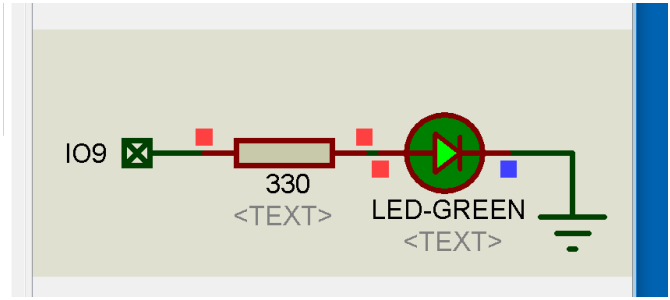
Volvemos a poner en marcha la simulación utilizando el botón play de nuestro cuadro de control de la simulación.

La simulación se volverá a poner en pausa de forma automática cuando se alcance el punto de ruptura (cuando generemos la orden de encender el led).



Podemos avanzar paso a paso en la ejecución de la simulación para poder llevar a cabo el proceso de depuración con detalle observando todas las repercusiones de cada bloque de código sobre el circuito.

En este ejemplo trivial, avanzar un salto, provocará el encendido del led.



Y el segundo salto, la iluminación del halo alrededor del pulsador situado en el cuadro de mando remoto.



Una vez finalizada la depuración y comprobado que todo funciona como nosotros esperábamos, ya podemos proceder a implementar el proyecto en equipos del mundo real.

> Por supuesto que este primer proyecto de ejemplo por el que hemos empezado es tan sencillo que no hemos sacado mucho provecho de las potentes herramientas de depuración que nos ofrece Proteus. Es posible encontrar información mucho más detallada de las posibilidades que brinda Proteus para la depuración en las ayudas disponibles.

3.11.-Uso de la app IoT Controller.

Proteus pone a nuestra disposición de forma gratuita una app para sistemas Android e iOS llamada 'IoT Controller' que es la forma más sencilla de construir un panel de mando remoto que podemos utilizar desde nuestro teléfono móvil o tableta.

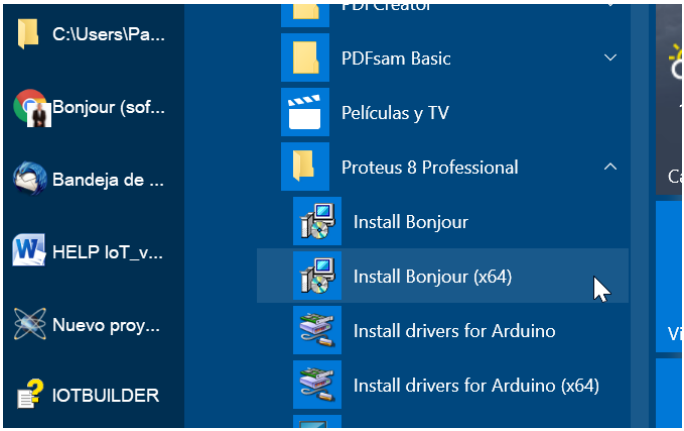
- Si utilizamos un dispositivo Android podemos descargar la APP desde Gooble Play buscando Proteus IoT Builder e instalándola. La herramienta es gratuita.

- Si utilizamos un dispositivo Apple podemos descargar la APP desde el Apple Store buscando Proteus IoT Builder e instalándola. La herramienta es gratuita.

La APP realiza las funciones de 'host' del panel frontal creado desde Proteus y también nos ayuda a localizarlo en nuestra red. Para encontrarlo se utiliza la utilidad de Apple llamada 'Bonjour' (https://es.wikipedia.org/wiki/Bonjour_(software)).

Esta utilidad ya se encuentra instalada en muchos equipos y es posible que el ordenador que estemos utilizando para ejecutar

Proteus ya la tenga instalada. Si no la tuviéramos instalada podemos proceder a hacerlo utilizando la opción disponible en el menú inicio de Windows dentro de la carpeta Proteus 8 Professional. Tendremos que seleccionar la opción de 32 o 64 bits según nuestro sistema operativo.

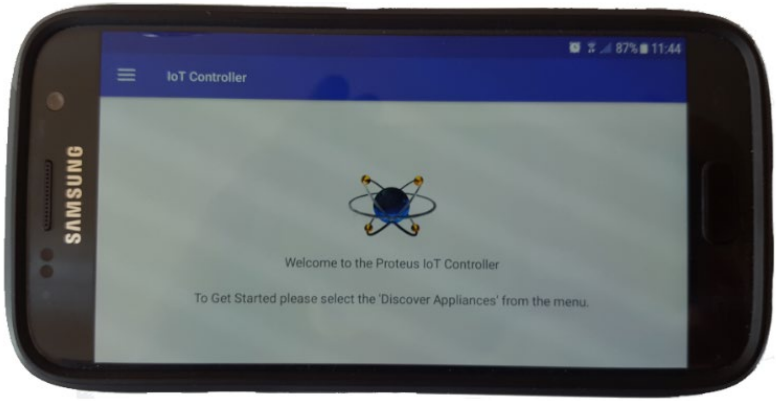


Una vez instalada, podemos utilizar la app para ejecutar nuestro cuadro de control remoto desarrollado con el taller IoT y controlar nuestro equipo virtual siguiendo estos pasos:

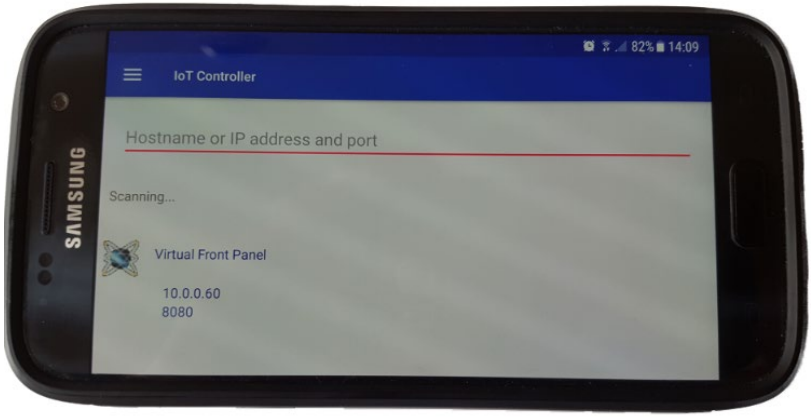
1. Comprobar que nuestro teléfono móvil o tableta está conectado en la misma red wifi a la que está conectado nuestro ordenador personal (PC).
2. Arrancar la simulación en Proteus.
3. Arrancar la app en nuestro teléfono móvil o tableta.



4. Una vez abierta, la app mostrará la ventana de inicio.

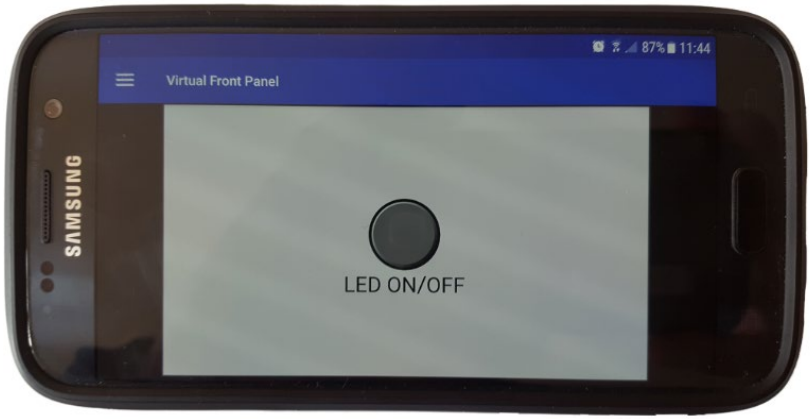


5. Abrir el menú y seleccionar la opción 'Discover appliances'. Buscaremos nuestro panel del cuadro en la lista de dispositivos encontrados y lo seleccionamos (en el ejemplo que se muestra en la imagen el panel se llama 'Virtual Front Panel', pero este nombre cambiará en función del nombre que le hemos dado en nuestro proyecto).



Podemos cambiar el nombre del panel desde la ventana de propiedades que se abre desde el menú contrextual de la rama 'Main Controls' del árbol de estructura del proyecto.

6. Ya tenemos que ver nuestro cuadro de mando remoto en nuestro teléfono móvil.



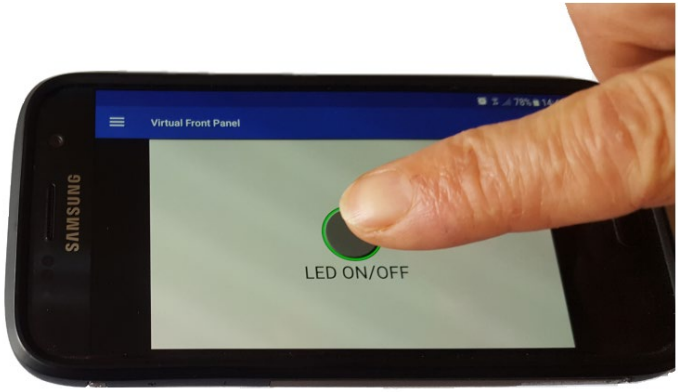
7. Al mismo tiempo, en nuestro cuadro de mando remoto virtual que se ejecuta en Proteus, encontraremos un mensaje avisando de que se ha perdido la conexión. Esto es normal, porque ahora la conexión está establecida con el teléfono móvil o tableta.

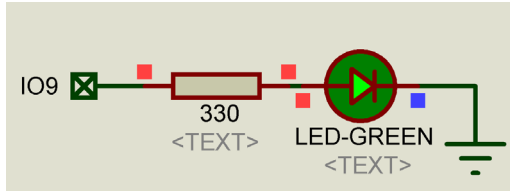


Como es lógico, la app no funcionará si nuestro ordenador personal (PC) no tiene wifi o no está conectado a una red que disponga de un router con red wifi.

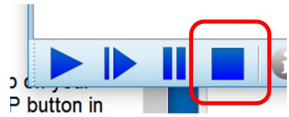
8. Podemos presionar el pulsador del cuadro de mando remoto de nuestro teléfono móvil o tableta. Tenemos que observar en la simulación de Proteus que el led se enciende y se apaga obedeciendo las órdenes. También

tenemos que ver que el halo de luz del pulsador se comporta como esperamos.





9. Detendremos la simulación utilizando el botón ‘Parar’ del cuadro de mando de la simulación. Es importante tener en cuenta que si cerramos la app en nuestro teléfono móvil o tableta no se detiene la simulación que se está ejecutando en nuestro ordenador personal PC.



3.12.-Implementación del proyecto en nuestro equipo de la vida real.

Una vez que hemos comprobado que la app que se ejecuta desde nuestro teléfono móvil o tableta y que todo funciona correctamente para monitorizar y controlar nuestro dispositivo, el último paso consiste en cargar nuestro código en el equipo de la vida real.

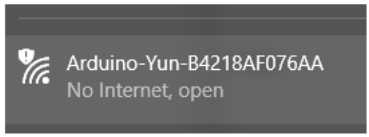
Podemos llevarlo a cabo utilizando comunicaciones Wireless SSH si nuestro ordenador está conectado a una red Wifi. O a través del puerto USB en caso contrario. El proceso es prácticamente idéntico en ambos casos. En este tutorial usaremos el primero de ellos.

> Una tarjeta de expansión Arduino Yun (incorporada a un controlador Arduino Uno o Arduino Mega) no puede programarse utilizando el puerto USB porque la conexión vía puerto serie ya está comprometida para otro uso. Sin embargo, un controlador Arduino Yun puede programarse vía SSH o vía USB.

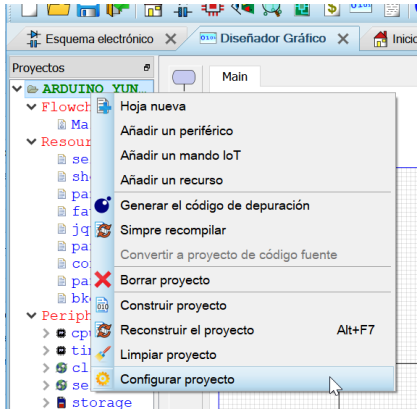
1. Encenderemos el controlador Arduino Yun conectándolo a la fuente de alimentación. Si se desea programar utilizando el puerto USB tenemos que asegurarnos de que el cable está conectado tanto al controlador Arduino Yun como al PC, porque en este caso el mismo cable realiza las dos funciones, alimentación y transmisión de datos.



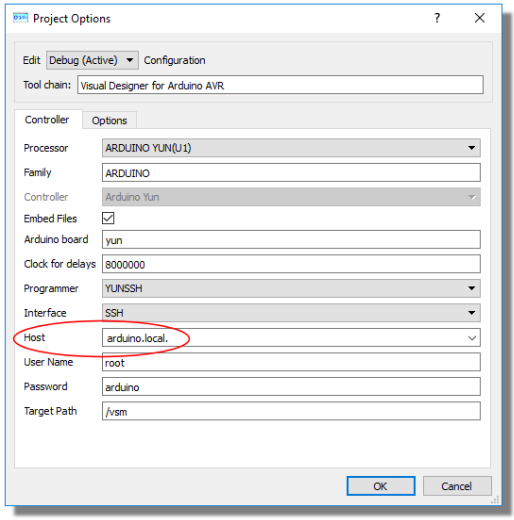
2. Esperamos unos treinta segundos para que el equipo Arduino Yun pueda arrancar completamente. Conectamos ambos equipos, el controlador Arduino Yun y nuestro ordenador personal PC a la misma red inalámbrica. Esta operación provocará la desconexión de internet, pero no nos preocupa porque no es necesario tenerla activa para el ejercicio que estamos llevando a cabo.



3. Seleccionamos la opción ‘Configurar proyecto’ del menú contextual de la rama principal del árbol de proyecto o desde el menú ‘Proyecto’ desde la pestaña del Diseñador Gráfico.

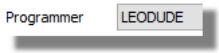


4. Si queremos programar el equipo utilizando Wifi, seleccionamos el programador YUNSSH y el interface SSH. El controlador Arduino Yun debería ser encontrado de forma automática y aparecer en el campo host como una opción disponible en la lista desplegable.



Por defecto el nombre del controlador Arduino Yun tiene la dirección MAC como un sufijo al final del nombre. Esta dirección MAC puede ser encontrada en la pegatina que viene en la zona trasera de la tarjeta.

5. Si estamos programando nuestro controlador Arduino Yun a través del puerto USB, tenemos que seleccionar el programador LEODUDE y seleccionar el puerto serie y la velocidad de transmisión de datos. Por defecto el Arduino Yun utiliza una velocidad de transmisión de datos de 57.600 baudios.



6. Utilizaremos el botón 'Enviar' de la barra de tareas de la zona superior para transmitir el código al controlador Arduino Yun.

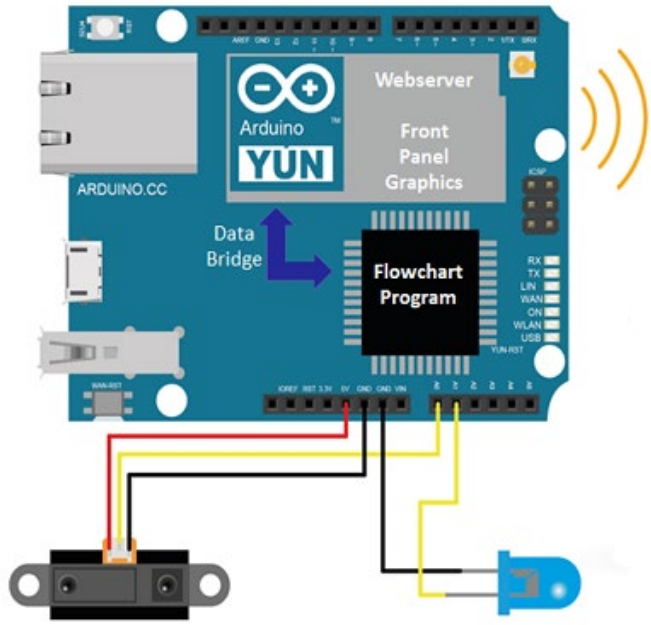


Si estamos utilizando el sistema de programación SSH el código fuente y los recursos asociados (webserver, gráficos de los paneles del mando remoto, etc.) se transmiten al microcontrolador Atheros. Este microprocesador, de forma automática, se encarga de ejecutar AVRDUDE y reprogramar la memoria donde se almacena el programa utilizando el interface ICP.

Si, por el contrario, estamos utilizando el sistema de programación USB, entonces el bootloader incorporado al microprocesador AVR se utilizará para transmitir todos los recursos utilizando el microcontrolador Atheros como puente. Finalmente será el AVRDUDE presente en el PC quien se encargue de reprogramar la memoria donde se almacena el programa. Este sistema puede ser significativamente largo en el tiempo en el caso de que el cuadro de mando remoto sea relativamente complejo.

Tanto si elegimos un camino o el otro, el resultado final será que el código del programa, el webserver y los gráficos necesarios para el

cuadro de mando remoto serán implementados en el controlador Arduino Yun, según se describe en el siguiente gráfico (el flujograma con el código del programa en el microprocesador AVR y el webserver y los gráficos del cuadro de mando remoto en el microprocesador Atheros).



3.13.-Ejecución final del proyecto.

Asumiendo que tenemos una tarjeta de expansión con un led conectada a nuestro controlador Arduino Yun en el pin correcto (IO10 en nuestro ejemplo) y que el controlador Arduino Yun está correctamente alimentado de corriente, para ejecutar nuestro proyecto en la vida real, solo tenemos que arrancar la APP en nuestro teléfono móvil o tableta, buscar el controlador Arduino Yun igual que hicimos en el caso de la simulación.

Si actuamos sobre el pulsador del cuadro de mando remoto de nuestro teléfono móvil podemos controlar el led de la vida real, exactamente igual que lo hicimos durante la simulación.



Con esto, completamos este primer tutorial intencionadamente simple para mostrar los rudimentos básicos de la técnica de programación de cuadros de mando remoto utilizando el taller IoT. En el segundo tutorial donde se construye un registrador de lecturas de temperatura podemos encontrar un proyecto mucho más útil y acercado a las necesidades de un proyecto real.

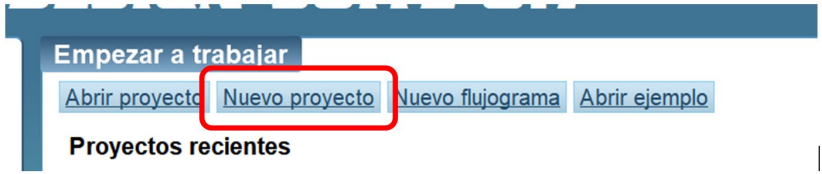
4.-Tutorial 2: Registrador de lecturas de temperatura.

En este segundo tutorial vamos a construir un registrador de lecturas de temperatura. El cuadro de mando remoto mostrará una gráfica con el histórico de medidas registradas, una serie de pulsadores que sirven para ajustar el rango de los datos visualizados y un termómetro que muestra la lectura actual. El equipo está basado en un Arduino Yun y un módulo Grove del tipo sensor de temperatura.

Aunque el proyecto es relativamente sencillo de construir utilizando el taller IoT, supone un nivel mayor de complejidad que el primer tutorial en el que llevábamos a cabo el control de encendido y apagado de un led. Para no caer en la repetición, en este segundo caso no vamos a detenernos en explicar las mismas técnicas básicas que ya vimos en el primer tutorial y las vamos a dar por supuestas, por lo que lo lógico sería que el lector empezara por aquél antes de comenzar con éste.

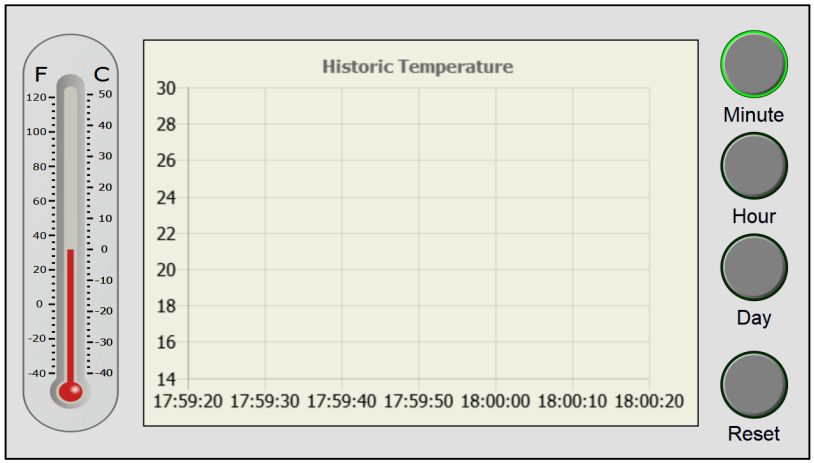
4.1.-Diseño del cuadro de mando remoto.

En primer lugar tenemos que crear un nuevo proyecto para Arduino Yun y el Diseñador Gráfico. Elegimos como compilador el Diseñador Gráfico, que significa que utilizaremos un flujograma para escribir nuestro programa.



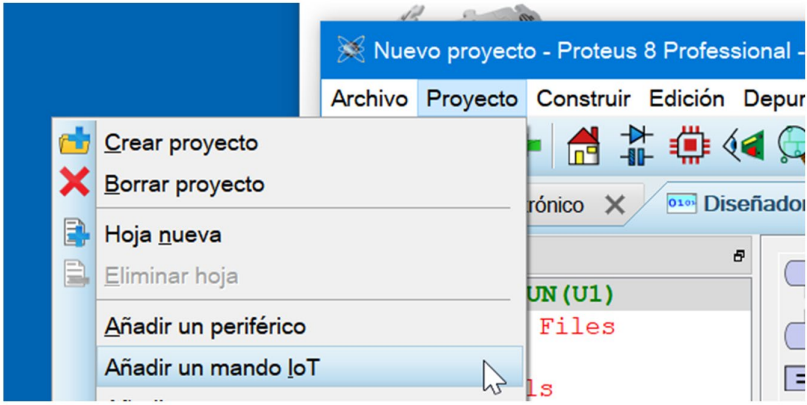


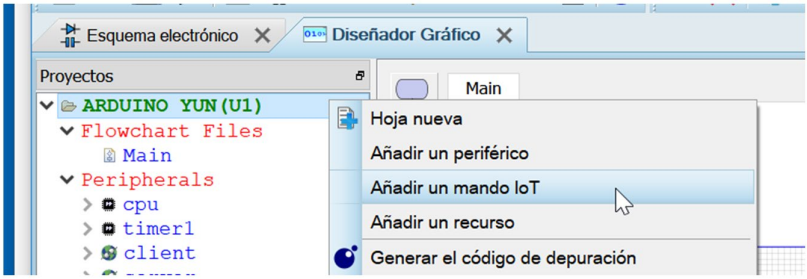
Nuestro objetivo es que el aspecto final de nuestro cuadro de mando sea como el que muestra la siguiente figura.



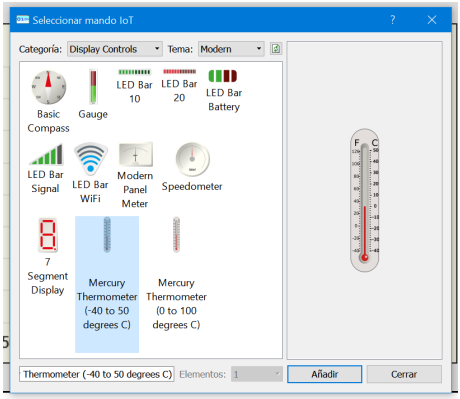
4.2.-Selección de los mandos.

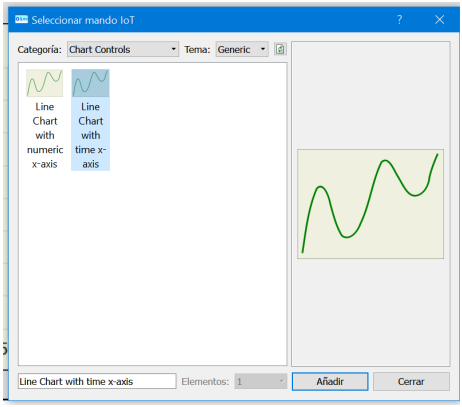
Todos los mandos que forman nuestro cuadro de mando remoto se añaden seleccionándolos desde la librería de mandos IoT. Para acceder a ella utilizaremos la opción 'Añadir un mando IoT' del menú Proyecto. O la opción del mismo nombre del menú contextual de la rama principal del árbol de estructura del proyecto.

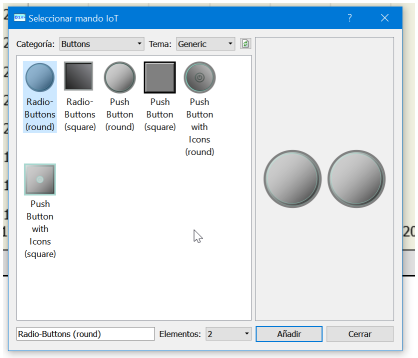




Queremos usar un mando que representa un termómetro clásico de mercurio (con un rango de -40 a 50ºC es suficiente) que encontraremos dentro de la categoría 'Display controls', un gráfico del tipo lineal con un eje-x que contenga los valores temporales que se encuentra en la categoría 'chart controls' y cuatro pulsadores que se encuentran en la categoría 'buttons'.

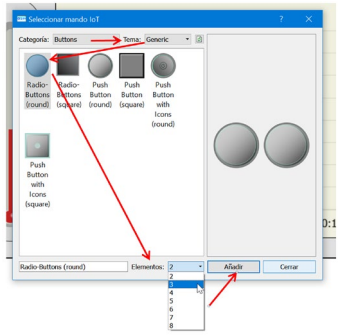




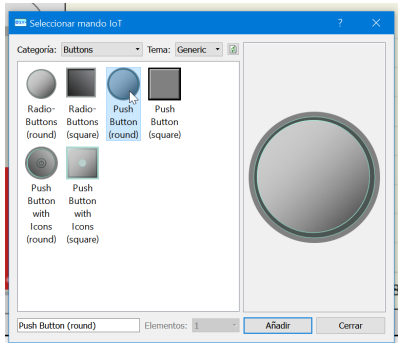


Vamos a detenernos en la selección de los pulsadores con más detalle porque vamos a utilizar un grupo de pulsadores de opciones

para cambiar el día, hora y minuto, en lugar de varios pulsadores independientes. El concepto es muy similar a los típicos botones de opciones (radio buttons) utilizados en muchos lenguajes de programación. Los pulsadores de opciones son mutuamente exclusivos entre sí, lo que significa que sólo se puede seleccionar uno de ellos al mismo tiempo. Además, los pulsadores de opciones se añaden todos en una sola operación formando un grupo. Para hacerlo, tenemos que seleccionar la opción 'Radio-buttons' e indicar el número de elementos que compone el grupo de pulsadores.



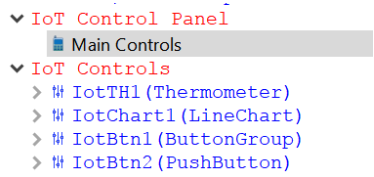
Además, para restaurar el gráfico a sus valores iniciales, vamos a utilizar un pulsador independiente convencional. Así que podemos seleccionar el pulsador del tipo 'push-button' y añadirlo a nuestro cuadro de mando de control de la forma habitual.



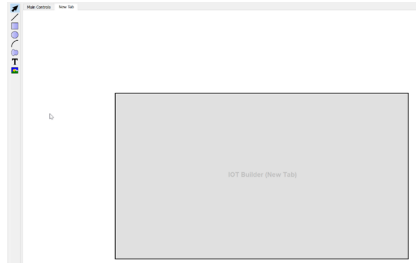
En este proyecto hemos utilizado el tema 'generic' pero, como vimos en el primer tutorial, se puede utilizar cualquier otro tema que deseemos entre los disponibles en la librería.

4.3.-Posicionamiento de los mandos.

Una vez que hemos añadido todos los mandos que necesitamos para el cuadro de mando remoto a nuestro proyecto, el siguiente paso consiste en ubicar los mandos en el panel. Nuestro árbol de estructura de proyecto debe tener un aspecto similar al que se muestra en la imagen siguiente.

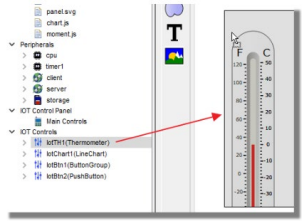


Haremos un doble-click en la rama 'IoT Control Panel' para abrir el editor de paneles de control.

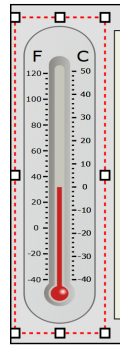


⚠ Como vimos en el primer tutorial, en primer lugar tenemos que configurar el tamaño de nuestro cuadro de mando remoto acorde al dispositivo que vamos a utilizar para visualizarlo (iPhone, Nexus, Galaxy, ordenador de escritorio, etc.). Este es el primer paso que debemos realizar siempre antes de empezar a colocar los mandos. Por defecto, el tamaño del panel es adecuado para utilizarse en cualquier teléfono móvil moderno en modo apaisado.

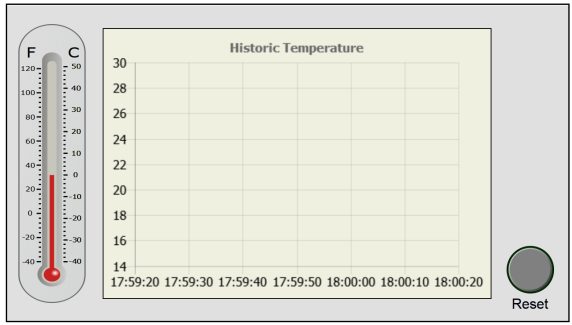
Empezaremos usando la técnica 'coger y arrastrar' para colocar el termómetro para colocarlo en el lado izquierdo de nuestro cuadro de mando remoto.



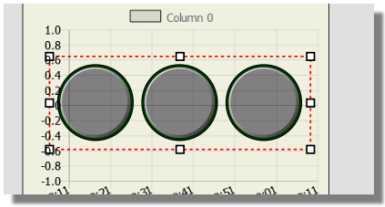
Puede ocurrir que el mando sea demasiado grande para poder ser colocado en el panel que hemos definido (hemos seleccionado el que nos ofrece Proteus por defecto). En este caso la solución es tan sencilla como usar los manejadores y reajustar el tamaño a nuestras necesidades. Si es necesario, también podemos cogerlo y moverlo a una nueva posición.



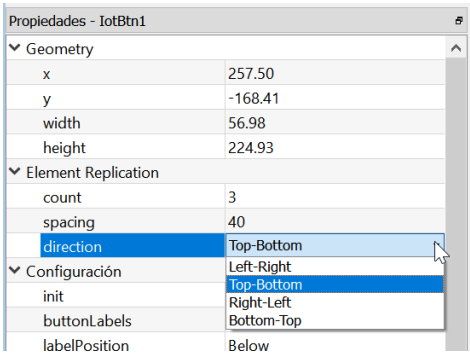
Repetiremos el proceso con el gráfico y el pulsador para limpiar (reset) los datos mostrados en el gráfico.



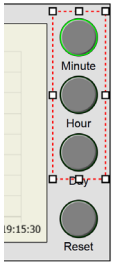
A continuación, colocamos el grupo de pulsadores en cualquier sitio del panel para trabajar con ellos. Por ejemplo, en nuestro caso, un buen sitio puede ser, aunque parezca algo extraño en principio, en el medio del gráfico que ya hemos colocado antes. Los pulsadores se colocan en línea horizontalmente.



Con el grupo de pulsadores seleccionado, desde el panel de propiedades buscamos la propiedad ‘direction’ del grupo ‘element replication’ y seleccionamos el valor ‘Top-Bottom’ para que los pulsadores se coloquen en línea verticalmente.



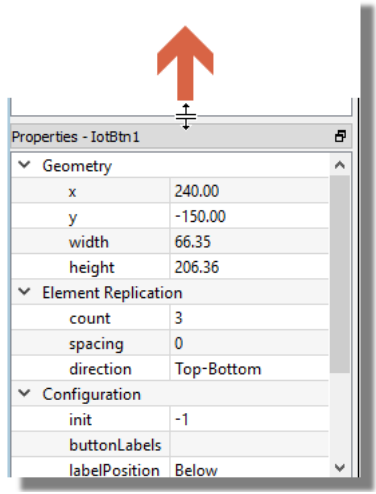
Sólo queda ajustar el tamaño del grupo de pulsadores y colocarlo justo encima del pulsador de iniciar.



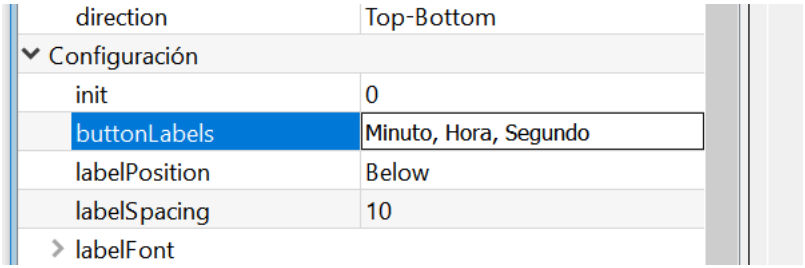
4.4.-Ajuste de las propiedades y alineación de los mandos.

A menos de que seamos unos virtuosos del diseño gráfico, lo más probable es que los pulsadores aparezcan un poco ‘feos’ en esta fase del diseño. Lo más probable es que los pulsadores estén más estrechos, más cortos o de tamaño distinto al pulsador de iniciar.

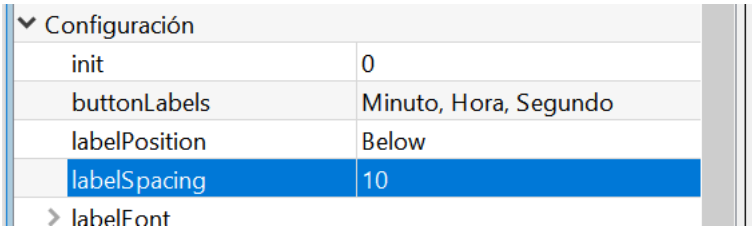
En este momento es de esperar que necesitemos ajustar el tamaño y posición de nuestro grupo de pulsadores. También aprovecharemos para colocar etiquetas debajo de cada uno de ellos para que el usuario pueda identificar de forma rápida su función. Para hacerlo, vamos a seleccionar el grupo de pulsadores y utilizar el panel de propiedades para llevar a cabo estas tareas de la forma más cómoda y sencilla posible. Podemos ajustar el tamaño del panel de propiedades para que nos resulte más cómodo trabajar con él.



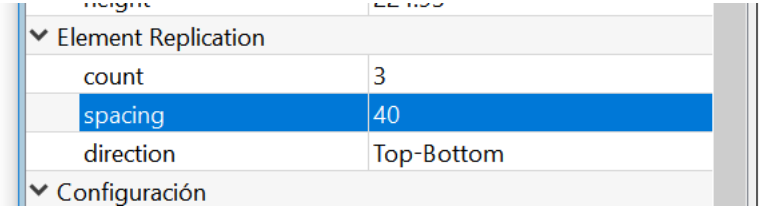
Utilizaremos la barra de desplazamiento vertical de la derecha para encontrar la sección ‘configuration’ e introduciremos las etiquetas de los pulsadores en forma de lista separada por comas utilizando la propiedad ‘buttonLabels’



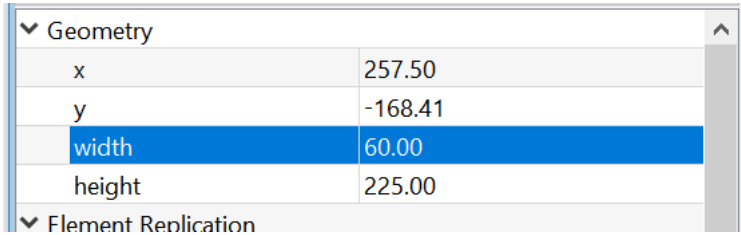
Vamos a necesitar dejar un poco más de espacio para las etiquetas entre el pulsador y el texto. Así que asignaremos un valor a la propiedad 'labelSpacing' de 10.



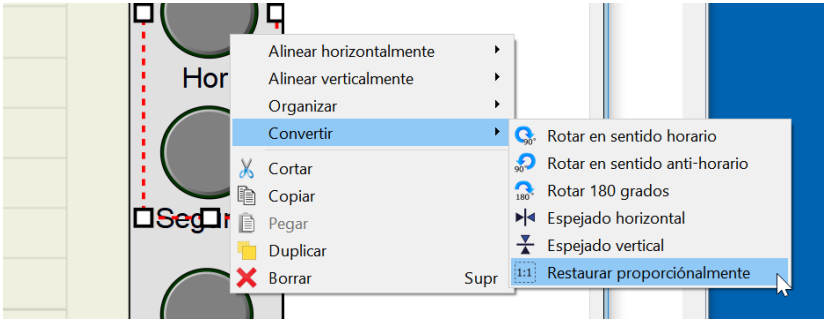
El próximo paso será ajustar la separación entre los pulsadores que forman el grupo. Lo haremos utilizando la propiedad 'spacing' dentro de la sección 'element replication'. Esta propiedad establece la distancia entre la zona inferior de un pulsador a la zona superior del siguiente. Un valor apropiado para nuestro ejemplo es 40.



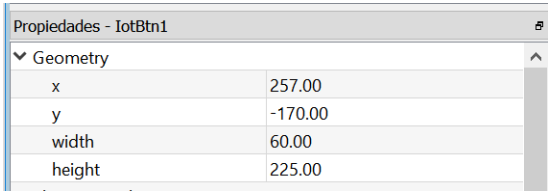
También será necesario ajustar el tamaño de los pulsadores. Podemos hacerlo utilizando el manejador de una de las esquinas y cambiando el tamaño de forma manual. Pero, casi seguro, que será más cómodo fijando el valor de las propiedades 'width' y 'height' con los valores 60 y 255. Las propiedades en el caso de grupo de pulsadores se refieren al tamaño total del grupo.



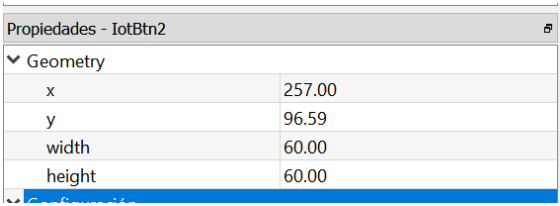
Si durante la manipulación manual del grupo de pulsadores estos quedarán muy deformados, podemos utilizar la opción del menú contextual 'Convertir->Restaurar proporcionalmente' para dejarlos a su forma original.



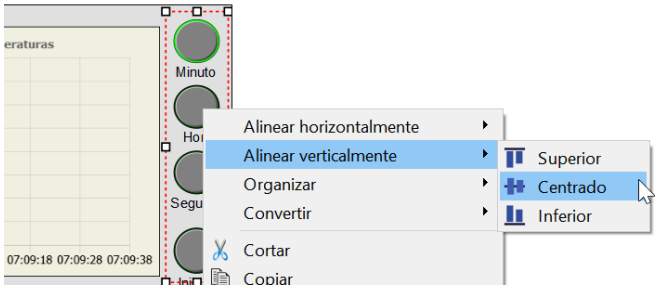
Podemos desplazar el grupo de pulsadores de forma manual hasta su posición definitiva o también podemos utilizar las propiedades 'x' e 'y' de la sección geometría para indicarlo cuando partimos de los requerimientos de un diseño previo que define exactamente la posición de cada elemento de nuestro cuadro de mando remoto.



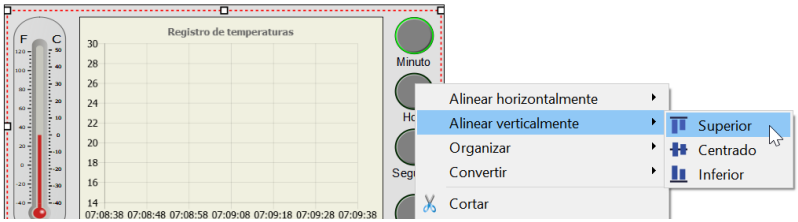
Pasemos ahora al pulsador que usaremos para reiniciar los datos del gráfico. Estableceremos el nombre de su etiqueta a 'Iniciar', el ancho y altura a 60 y el valor de 'x' al mismo que usamos en el grupo de pulsadores para que quede alineado correctamente con aquél.



También es posible alinear varios objetos manteniendo pulsada la tecla CTRL mientras vamos haciendo click con el ratón sobre los objetos que deseamos alinear. Una vez seleccionados podemos elegir desde el menú contextual las opciones alinear horizontalmente o alinear verticalmente.

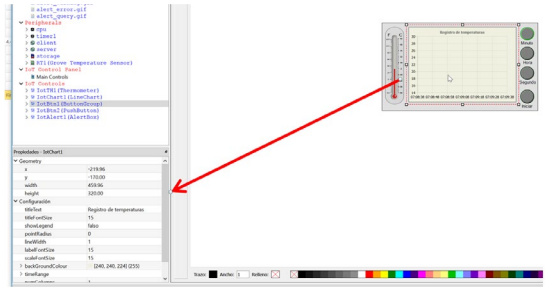


Podemos continuar con el uso de los comandos disponibles para alinear con la gráfica y el termómetro. Por ejemplo, si deseamos que todos ellos tengan un mismo valor para su posición respecto al eje ‘y’ el comando alinear verticalmente -> superior los dejaría perfectamente alineados.

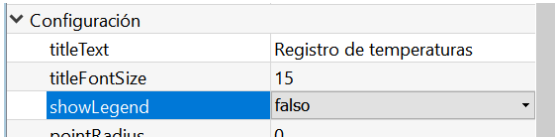


4.5.-Configuración de la gráfica.

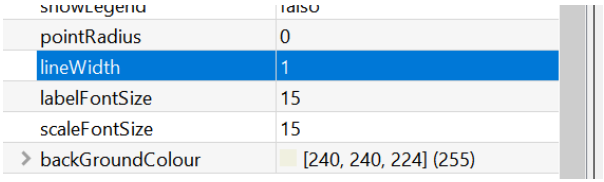
El mando ‘gráfica’ es un control muy flexible altamente configurable. En nuestro ejemplo, vamos a cambiar algunas de las opciones de configuración en tiempo de ejecución (por ejemplo, cambiaremos el eje de tiempos cada vez que el usuario utilice uno de los pulsadores existentes para modificar el rango del histórico mostrado). A parte de esas propiedades, este el momento adecuado para configurar algunas otras relacionadas con los aspectos estéticos del gráfico. El camino para llevarlo a cabo es el uso del panel de configuración previa selección del gráfico.



Podemos empezar cambiando el título del gráfico y, puesto que en este ejemplo sólo usaremos un único conjunto de datos, podemos prescindir de la leyenda que identifica cada línea del gráfico con el conjunto de datos que la origina.



También podemos seleccionar el ancho de la línea a un único pixel y cambiar el color de fondo por defecto a uno nuevo que se ajuste un poco más al diseño general del cuadro de mando remoto.

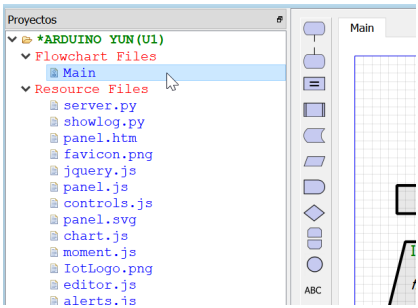


4.6.-Escritura del código (flujograma).

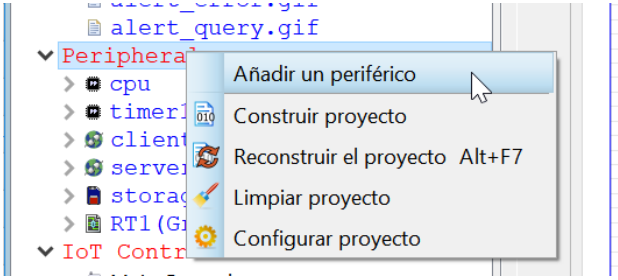
Una vez que nuestro cuadro de mando remoto ya tiene el aspecto que deseamos, es el momento de empezar a pensar sobre el programa que controlará todo el sistema.

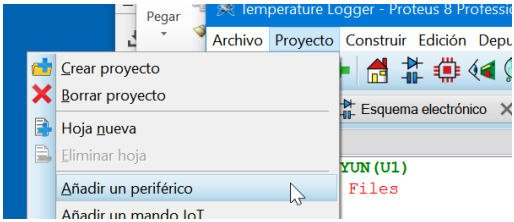
> En este tutorial, pasaremos por encima sobre las técnicas básicas del uso de los flujogramas. Si el lector no está familiarizado con ellas, puede consultar la ayuda del Diseñador Gráfico (disponible en el menú 'ayuda') para ver con todo detalle cómo se utiliza esta herramienta.

Para pasar de la ventana del editor del cuadro de mando remoto al editor del flujograma sólo tenemos que hacer un doble click en la rama correspondiente del árbol de estructura del proyecto.

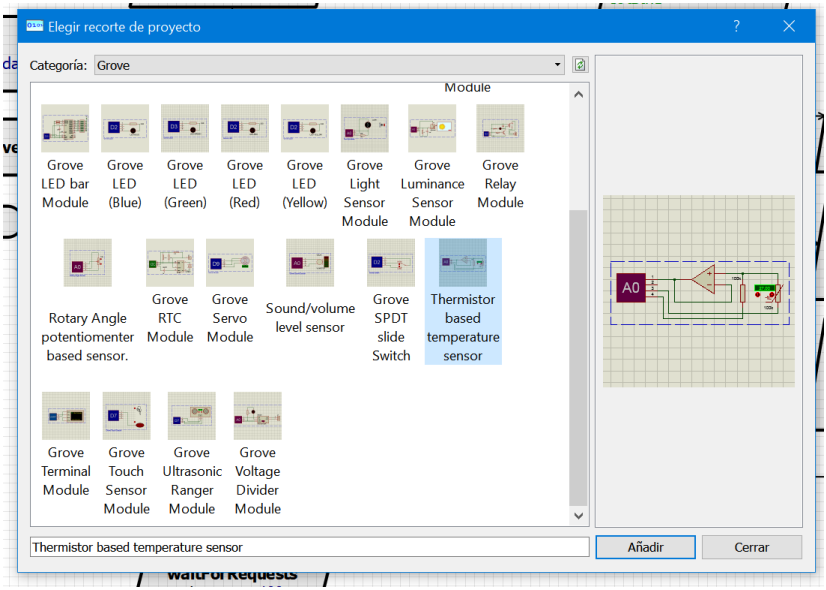


Antes de empezar con el código, vamos a añadir el módulo Grove con el sensor de temperatura que utilizaremos para obtener todas las medidas. Para llevar a cabo este paso seguiremos el camino habitual de utilizar el comando 'añadir un periférico' desde el menú contextual de la rama 'Peripherals' del árbol de estructura del proyecto o desde la opción del mismo nombre del menú 'Proyecto'.





Seleccionaremos desde la librería de periféricos el módulo 'Thermistor based temperatura sensor' situada en la categoría 'Grove'.



4.6.1.-Estructura del programa.

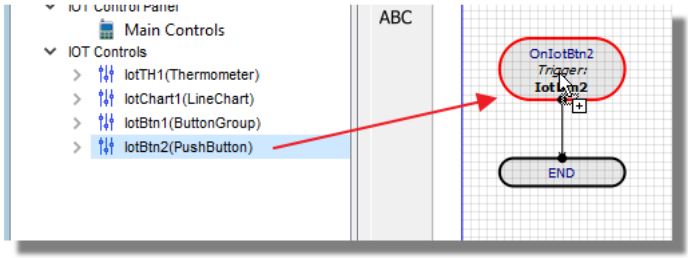
Como en todos los proyectos Arduino, nuestro programa necesita utilizar una fase de configuración inicial y una fase de ejecución en bucle en la que responderemos a las órdenes que pueda genera el usuario mediante el uso de los pulsadores.

Las órdenes generadas con los pulsadores pueden servir para cambiar el rango histórico de los datos mostrados en la gráfica (cuando el usuario usa uno cualquiera del grupo de pulsadores) o

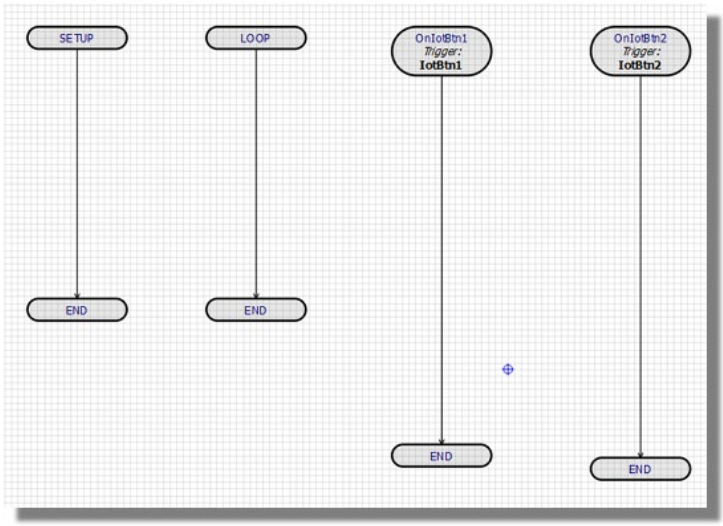
para borrar todos los datos acumulados (cuando el usuario usa el pulsador independiente de reset).

Por defecto, Proteus ya construyó para nosotros dos subrutinas vacías que se corresponden con las fases habituales de los códigos para Arduino encargadas de la configuración inicial (setup) y ejecución en bucle (loop).

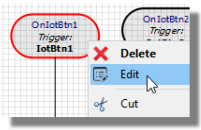
Para responder a la actuación del usuario sobre un pulsador concreto podemos utilizar un bloque de respuesta al evento situado en nuestro flujograma como ya vimos en el primer tutorial. Sólo tenemos que usar la técnica 'coger y arrastrar' desde la rama del árbol de estructura de proyecto de ese mando en concreto hasta el flujograma.

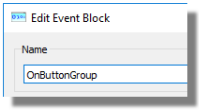


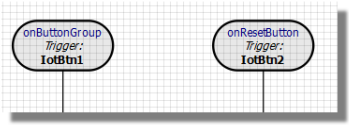
Tenemos que llevar a cabo esta operación tanto para el grupo de pulsadores como para el pulsador individual (reiniciar). Con unas pocas operaciones, tenemos que conseguir que nuestro flujograma tenga un aspecto similar al mostrado en la imagen siguiente



Es una buena práctica editar y cambiar los nombres de los gestores de los pulsadores para asignarles nombres que resulten más sencillos de recordar. Por ejemplo, en este ejemplo, hemos utilizado ‘onButtonGroup’ y ‘onResetButton’.







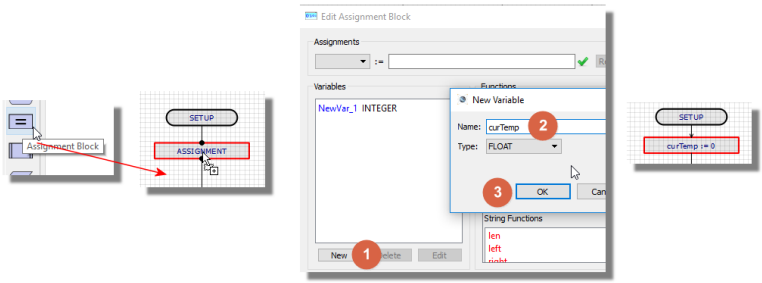
4.6.2.-Configuración (subrutina setup).

La subrutina ‘setup’ es el lugar donde se ejecutan las tareas relativas a la inicialización y configuración previas a la ejecución propiamente dicha de nuestro código.

Como nuestro proyecto utiliza un mando del tipo termómetro para mostrar la lectura de temperatura actual, necesitamos usar una variable para almacenar el valor leído.

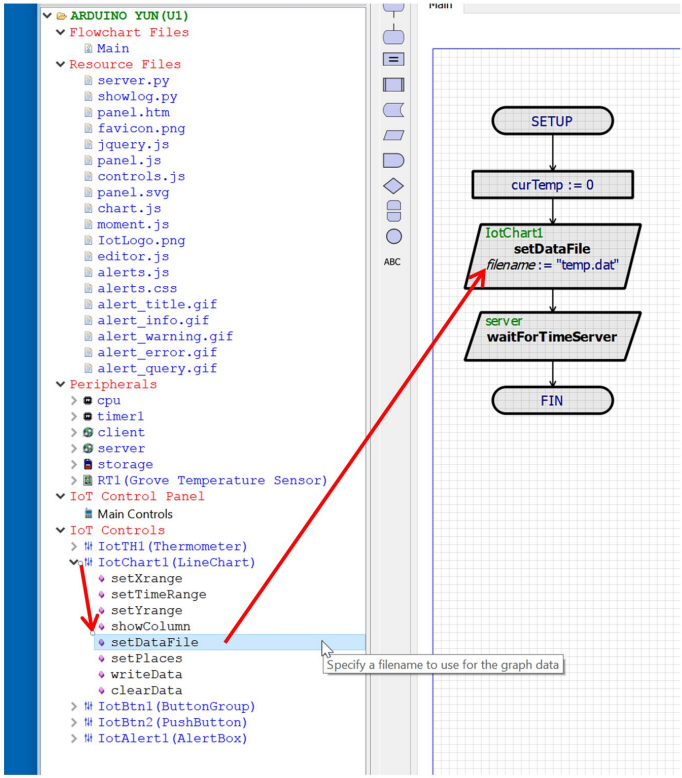
Usaremos la técnica ‘coger y arrastar’ para colocar un bloque del tipo asignación dentro de la subrutina ‘setup’, lo editaremos, crearemos

la variable con el nombre ‘curTemp’ del tipo flotante y le asignaremos el valor de cero.



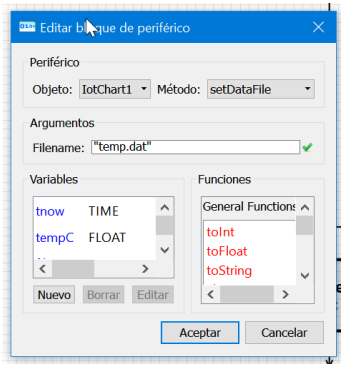
También necesitamos almacenar en algún sitio el histórico con los valores de todas las medidas de temperatura que hemos ido recogiendo. Para pequeños volúmenes de datos, esta tarea puede ser hecha de forma automática y almacenada como datos propios de una sesión en el microprocesador Atheros. Pero para volúmenes de datos mayores es necesario usar un fichero e indicar que los datos serán almacenados en este fichero. El fichero puede estar físicamente en el microprocesador Atheros o en una tarjeta SD.

Para elegir el fichero de datos donde se almacenarán los datos del histórico de medidas usaremos uno de los métodos disponibles asociados con el mando IoT ‘gráfica’. Expandimos esta rama del árbol de estructura de proyecto y usamos la técnica ‘coger y arrastar’ para llevar el método ‘setDataFile’ dentro de nuestro flujograma.

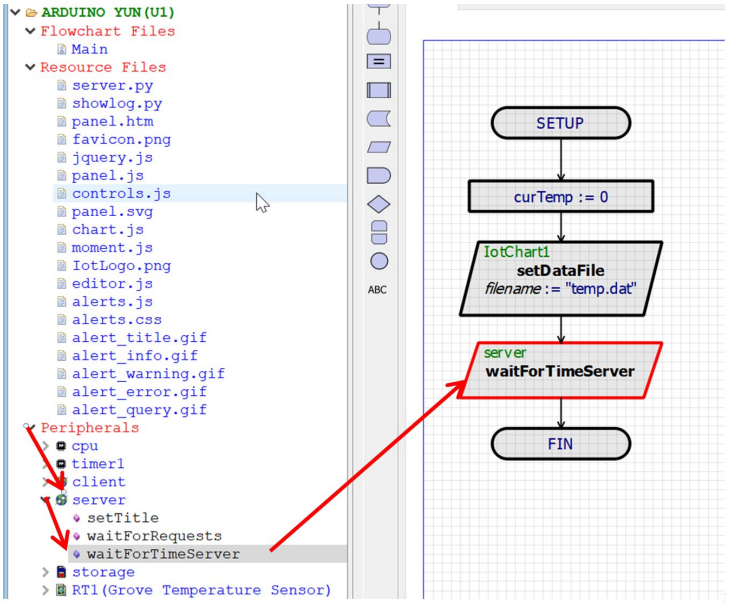


El fichero con el histórico por defecto se almacena en la carpeta VSM dentro del microprocesador Atheros. Es posible indicar que este fichero debe almacenarse en una tarjeta SD utilizando como prefijo del nombre del fichero la ruta completa a la tarjeta (v.g. /MMT/SDA1 en el caso de una tarjeta de expansión Yun o /MMT/SDB1 en el caso de un controlador Arduino Yun).

Una vez que hemos colocado el bloque del tipo ‘operación con un periférico’ en nuestro flujograma, es necesario editarlo para asignar el nombre al fichero.



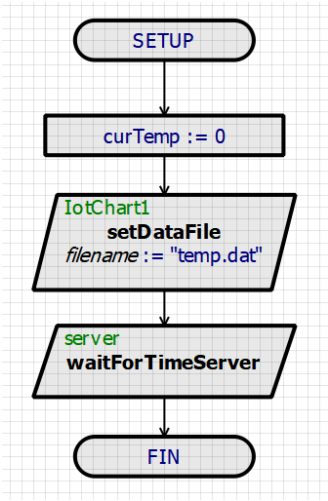
La siguiente tarea que necesitamos llevar a cabo antes de empezar a guardar los datos de las medidas de temperatura es comprobar y asegurarnos de que el 'timeserver' está funcionando correctamente. Para hacerlo expandimos la rama 'server' que está dentro de la sección 'Peripherals'. Seleccionamos el método 'waitForTimeServer' y usamos la técnica 'coger y arrastrar' para colocarlo dentro del flujograma. Este bloque no tiene ningún parámetro puesto que, en esencia, no es más que un delay especial que termina cuando el servidor confirma que el 'timeserver' está trabajando correctamente.



El ‘timeserver’ es un componente del Sistema operativo Linux que se está ejecutando en el microprocesador Atheros. Su función es buscar la hora actual en alguno de los servidores de hora existentes en internet. La función ‘waitForTimeServer’ se encarga de comprobar que el ‘timeserver’ está ejecutándose correctamente y que ha sido capaz de conectarse a algún servidor de hora de internet correctamente. De esta forma, estamos seguros de que la hora de nuestro sistema (base para la toma de medidas) es la correcta.

> ⚠ Si nuestro equipo Arduino Yun no estuviera conectado a internet por cualquier motivo, el ‘timeserver’ no podrá obtener una hora válida y este bloque del flujograma no terminará nunca su ejecución y no permitirá que nuestro programa avance y se ejecute correctamente. Seguro que el lector podrá modificar el código sin mucho esfuerzo si desea un funcionamiento diferente.

Después de realizar este último paso, la subrutina ‘setup’ completa debe tener el siguiente aspecto.



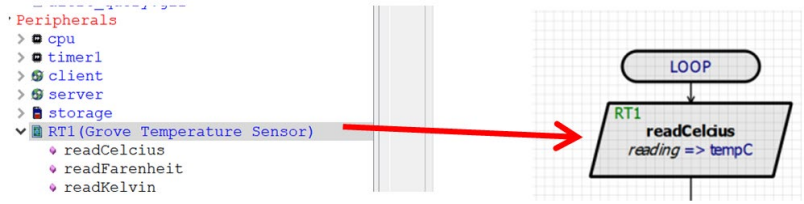
4.6.3.-Bucle cíclico (subrutina loop).

Una vez terminada la escritura de la subrutina de configuración tenemos que afrontar la tarea de definir lo que nuestro programa

debe hacer. Para una normal operación, nuestra estación registradora de temperaturas debe llevar a cabo varias operaciones:

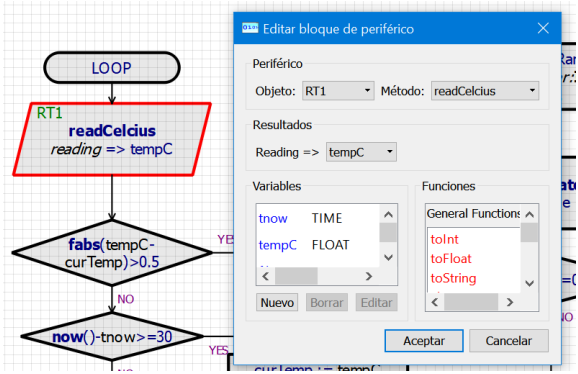
- Leer el valor de temperatura actual.
- Actualizar el mando termómetro con la última lectura.
- Actualizar los datos del gráfico con el histórico recopilado.
- Comprobar si ha habido un cambio de temperatura.

La primera de ellas es la más sencilla. Nuestro termómetro va a mostrar la información en grados Celsius, así que sólo tenemos que usar la técnica 'coger y arrastrar' con el método 'readCelcius' que cuelga de la rama 'RT1(Grove temperature sensor)' y colocarla dentro de la subrutina 'loop' en nuestro flujograma.



Elegir entre mostrar la temperatura en grados Farenheit o Celsius en el mando termómetro que estamos utilizando en nuestro cuadro de mando remoto es una opción configurable utilizando el panel de propiedades. Así que, si lo deseamos, es muy sencillo cambiar el termómetro para que muestre grados Farenheit y utilizar el método 'readFarenheit' asociado al sensor de temperatura para realizar la lectura en la unidad adecuada.

Proteus se ha encargado de forma automática de crear para nosotros una nueva variable de tipo flotante llamada 'tempC' y asignarle el valor de lectura recibida del sensor. Lo podemos comprobar editando las propiedades del bloque que acabamos de crear.



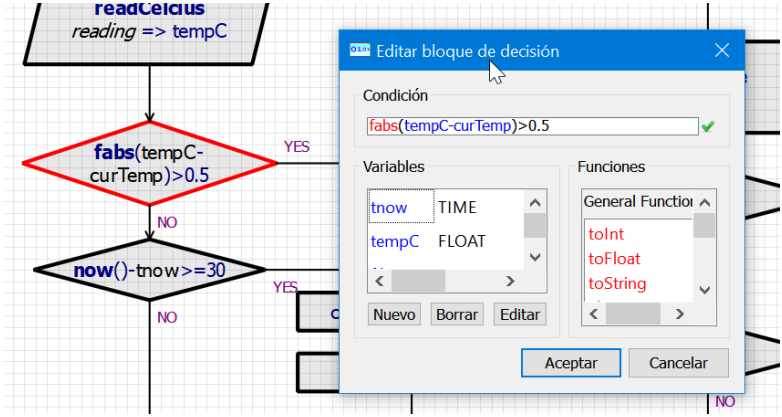
Puede parecer extraño que no utilicemos la variable ‘curTemp’ que habíamos creado en la subrutina ‘loop’ para almacenar el valor de la lectura de temperatura. La razón por la que no actuamos de esta manera es el tiempo que esto lleva y la optimización de todos nuestros recursos de hardware. Analicemos con cuidado la situación para entenderlo.

Nuestra subrutina ‘loop’ se va a ejecutar extremadamente rápido porque el código que contiene es pequeño. Eso significa que tenemos que pensar siempre en que vamos a estar llevando a cabo las tres tareas que hemos definido (y no sólo la lectura de temperatura) y las consecuencias que esto supone:

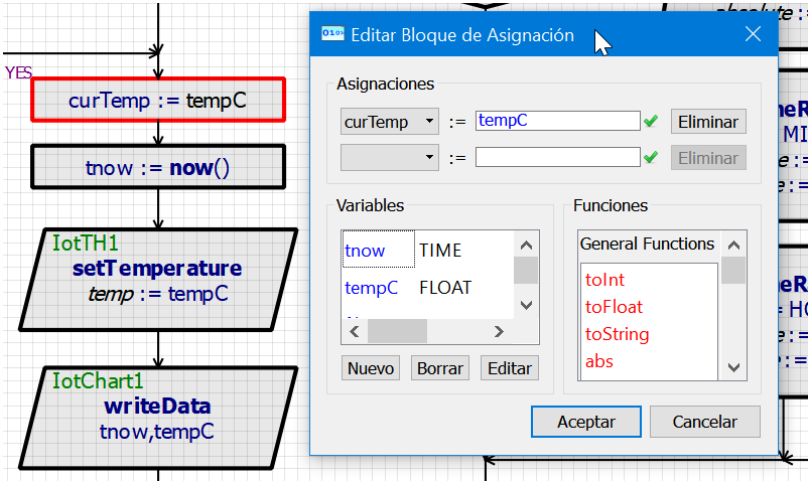
- Nuestro fichero histórico se va a rellenar muy rápidamente de una ingente cantidad de información.
- Vamos a tener un gran volumen de tráfico de información entre nuestro dispositivo Arduino Yun y el cuadro de mando remoto.
- Va a ser muy difícil atender a los comandos recibidos del usuario mediante los pulsadores situados en el cuadro de mando remoto porque estamos ejecutando un bucle a rápida velocidad que prácticamente ocupa toda la capacidad de procesamiento de nuestro microprocesador.

A la vista de lo cual, parece imprescindible contestar a la pregunta ¿cuándo es realmente necesario actualizar la información mostrada en el cuadro de mando remoto? La respuesta es, obviamente, sólo cada vez que la temperatura haya experimentado una variación. Así que conviene que añadamos a nuestro flujograma un bloque de

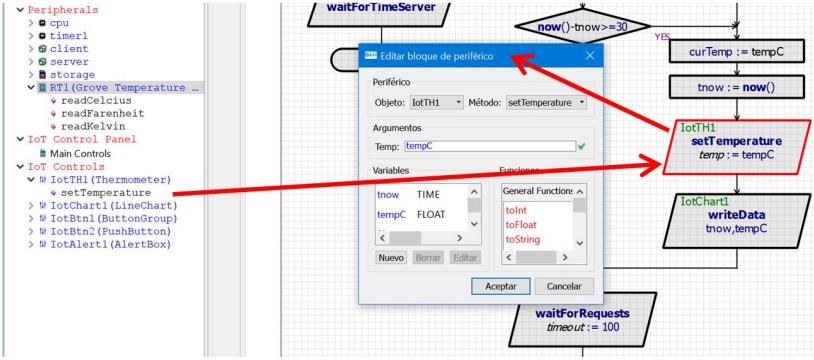
decisión que compruebe si el valor actual de lectura de temperatura es medio grado diferente al de la anterior lectura llevada a cabo.



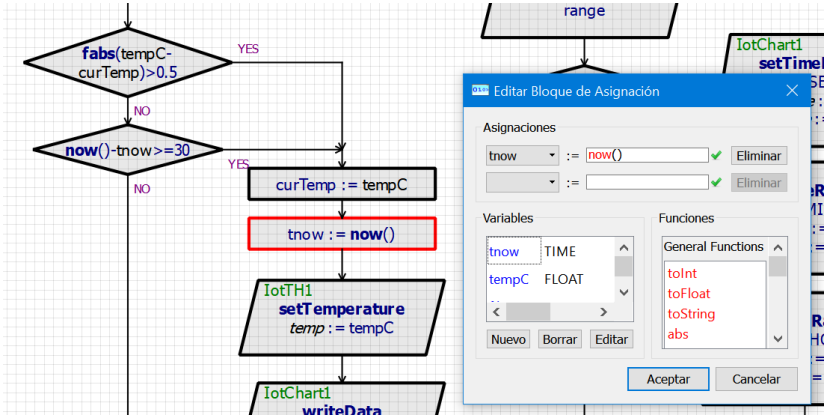
Si la temperatura ha experimentado el cambio que estamos evaluando, asignaremos el valor de la lectura recibida (tempC) a la variable que creamos en el bucle 'setup' (curTemp) con objeto de poder llevar a cabo una nueva comparación en la siguiente ejecución del bucle principal del programa. Para hacerlo usaremos un bloque de asignación y lo colocaremos en nuestro flujograma a la salida 'True' del bloque de decisión.



A continuación, actualizaremos el termómetro del cuadro de mando remoto con la lectura de temperatura. Podemos llevarlo a cabo usando la técnica ‘coger y arrastrar’ para mover el método ‘setTemperature’ de la rama ‘IotH1’ del árbol de estructura del proyecto hasta nuestro flujograma.

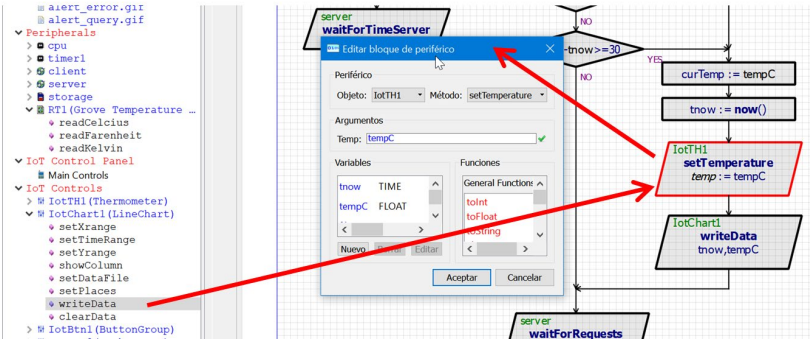


También tenemos que registrar el cambio de temperatura y anotar el momento en que se ha producido. Para poder hacerlo necesitamos saber cuál es la hora actual. Para conocerla, podemos utilizar el método ‘now()’ y almacenar su valor actual en una variable (que vamos a llamar ‘tnow’) utilizando un bloque de asignación.



El método now() se encarga de realizar una llamada al 'timeserver' y devolver un valor en segundos en formato EPOCH[3]. De esta forma nos aseguramos que no haya problemas con la datación de las lecturas.

Ya podemos utilizar el método 'writeData' del mando 'IotChart1' para añadir un nuevo punto a la gráfica del cuadro de mando remoto.

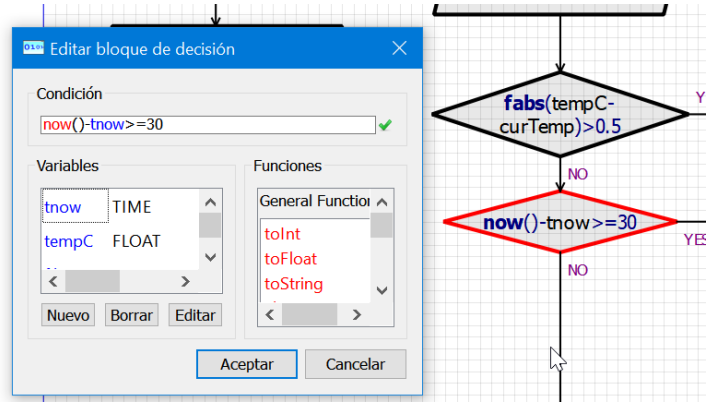


El método writeData se encarga de las dos tareas: actualizar la gráfica y añadir un registro al fichero que guarda el histórico de lecturas.

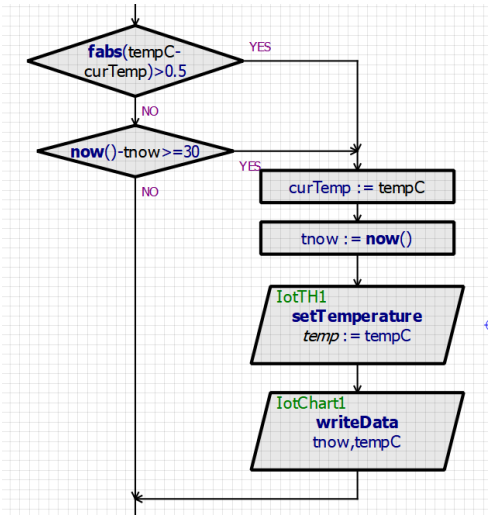
En situaciones normales, es deseable que cuando la temperatura no experimente cambios durante un determinado espacio de tiempo queramos seguir recogiendo en nuestro histórico una serie de datos de forma periódica para comprobar que todo funciona correctamente y que no se está experimentando una pérdida de lecturas. Por ejemplo, podemos fijar que cada 30 segundos volveremos a llevar a cabo una escritura en el histórico de lecturas independientemente de que la temperatura haya cambiado o no.

Podemos hacerlo utilizando la rama de nuestro flujograma que resulta de responder 'NO' a nuestro bloque de decisión que comprobaba si la lectura de temperatura era diferente a la grabada con anterioridad. Podemos poner un nuevo bloque de decisión en este punto que evalúe si han transcurrido 30 segundos desde la última escritura.

[3] Consultar https://es.wikipedia.org/wiki/Tiempo_Unix para obtener más información sobre el formato EPOCH utilizado por el sistema operativo Unix.



Si la respuesta a este nuevo bloque de decisión es NO, no tenemos que hacer nada más. Si la respuesta es SI, tendremos que realizar las mismas operaciones que hemos hecho cuando hubo un cambio de temperatura. Al final, el flujograma debe tener este aspecto.

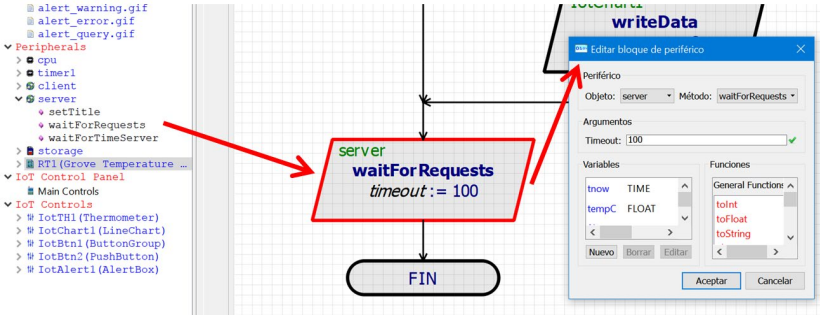


Terminaremos generando una pequeña pausa en la ejecución de nuestro programa que detenga momentáneamente la ejecución de la subrutina 'loop' y le otorgue un poco de tiempo a evaluar los comandos recibidos desde el cuadro de mando remoto pendientes de atender. Es de vital importancia que añadamos en nuestro

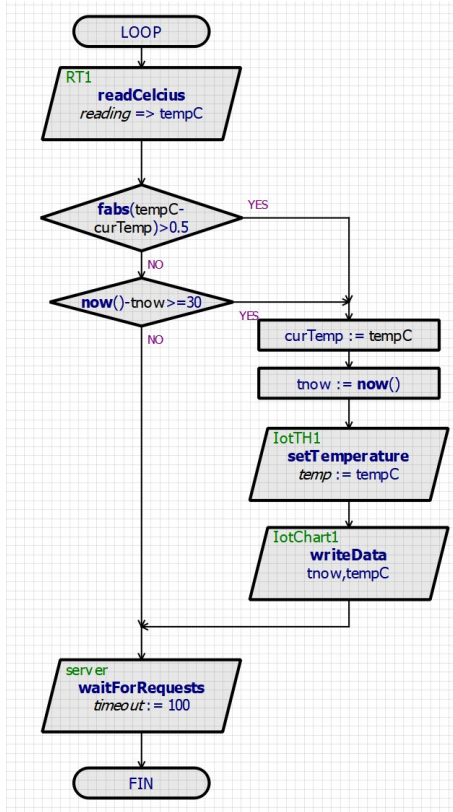
flujograma una llamada al método del servidor ‘WaitForRequest’ y que no utilicemos un bloque ‘delay’ convencional.

Un bloque ‘delay’ detiene completamente la ejecución de nuestro microprocesador y no le daríamos opción a evaluar las actuaciones que el usuario lleve a cabo mediante los pulsadores. Por el contrario, un método ‘waitForRequest’ permite al microprocesador responder a las interrupciones generadas por el sistema como, por ejemplo, la presión sobre un pulsador por parte del usuario.

Para evitar los problemas y posibilitar que nuestro controlador responda de forma satisfactoria a las órdenes del usuario, tendremos que incluir en nuestro flujograma una llamada al método ‘waitForRequest’.



Una vez llevadas a cabo todas estas tareas, nuestra subrutina ‘loop’ completa debe tener un aspecto como el mostrado en la imagen siguiente.



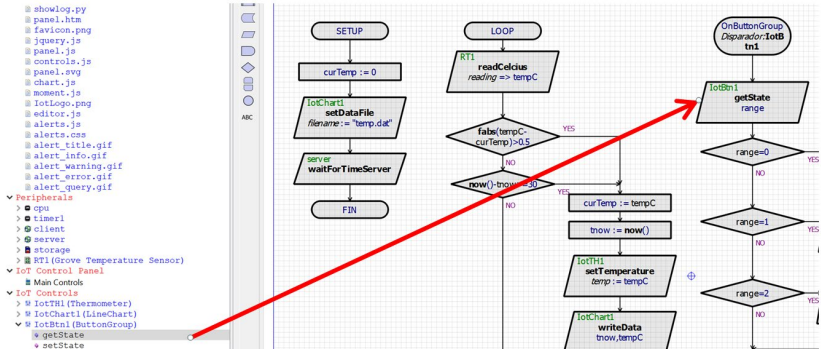
⚠ Terminamos este apartado recordando que suele ser un error frecuente en este tipo de aplicaciones confundir el tiempo de ejecución de nuestro código con el tiempo de actualización de los datos. Hay que tener siempre presente que la subrutina 'loop' se ejecuta extremadamente rápido (se compone de muy pocas líneas de código) y que las necesidades de actualización, grabado de históricos y refresco de gráficos se lleva a cabo a velocidades mucho más reducidas. El 'tiempo de la máquina' es absolutamente diferente del 'tiempo del ser humano'.

4.6.4.-Control de los eventos generados por el grupo de pulsadores.

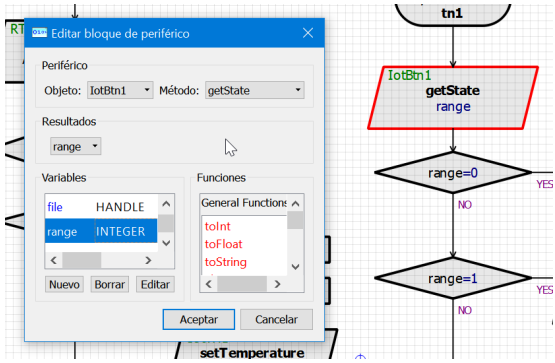
Una vez que ya hemos programado la lógica principal de nuestro dispositivo, llega el momento de planificar cómo vamos a responder a los diferentes requerimientos del usuario que van a llegar desde el cuadro de mando remoto.

Nos ocuparemos en primer lugar de la agrupación de pulsadores. Si recordamos, al principio habíamos creado una subrutina llamada 'OnButtonGroup' que habíamos destinado a gestionar los eventos asociados al grupo de pulsadores. Por lo tanto, cada vez que el usuario actúe sobre uno de estos pulsadores, el programa automáticamente saltará al primer bloque dentro de esta subrutina. Nuestra primera tarea consistirá en determinar cuál de los tres pulsadores ha sido activado (recuerde que sólo puede estar uno activo en cada momento) y en función del resultado configurar la gráfica para mostrar los valores almacenados durante un minuto, una hora o un día.

Empezaremos usando la técnica 'coger y arrastar' para posicionar el método 'getState' de la rama 'IotBtn1' dentro de la sección 'IoT Controls' desde el árbol de estructura de proyecto al comienzo de la subrutina 'OnButtonGroup'.

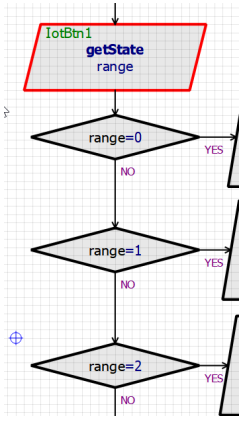


Editaremos el bloque que acabamos de posicionar y asignaremos el valor a una variable de tipo entero a la que llamaremos 'range'.

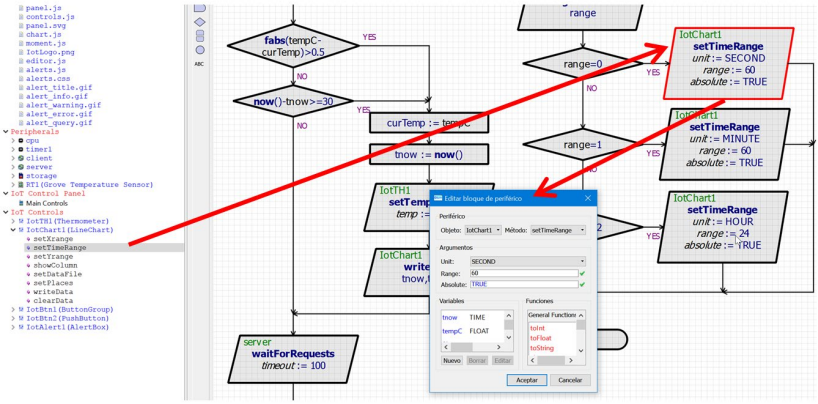


El método ‘getState’ asociado a un grupo de pulsadores devuelve un valor que indica la posición del pulsador actualmente activo. El rango de posiciones empieza en 0. En nuestro caso, el primer pulsador es el 0 (rotulado como segundos), el segundo es el 1 (rotulado como minutos) y el tercero es el 2 (rotulado como horas). Se ha configurado de esta manera para que sea posible en cualquier momento añadir o restar nuevos pulsadores a un grupo ya existente según las necesidades de cada proyecto.

Una vez que tenemos la información del pulsador activo almacenada en la variable ‘range’, sólo tenemos que hacer una evaluación por cada uno de los pulsadores posibles para comprobar el valor de la variable ‘range’ para cada caso y ajustar el eje-x de la gráfica acorde a la selección del usuario. Por cada pulsador, usaremos un bloque de decisión.

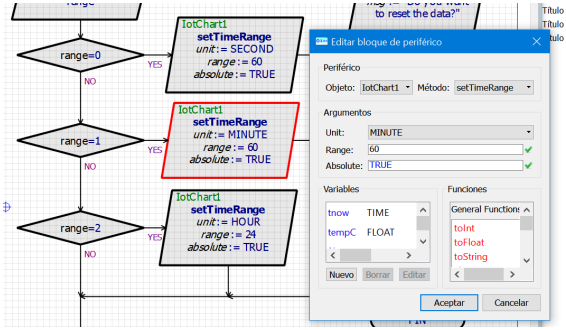


Para fijar el rango de la gráfica adecuado a cada selección tenemos que utilizar el método ‘setTimeRange’ de la rama ‘IotChart1’.

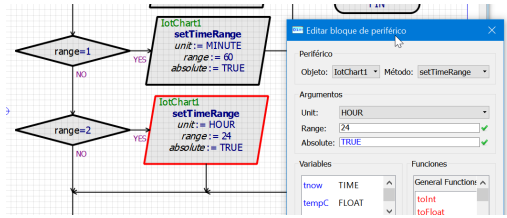


La configuración se logra seleccionando ‘second’ en el campo ‘unit’, el valor ‘60’ en el campo ‘range’ (60 segundos = 1 minuto) y el valor ‘True’ para el parámetro ‘Absolute’. El rango de tiempo seleccionado indica los datos que se visualizaran desde el momento actual hacia atrás en el tiempo en una cantidad igual a la indicada por el rango seleccionado. Si el campo ‘Absolute’ se asigna con un valor ‘False’ entonces los datos del tiempo se consideran relativos.

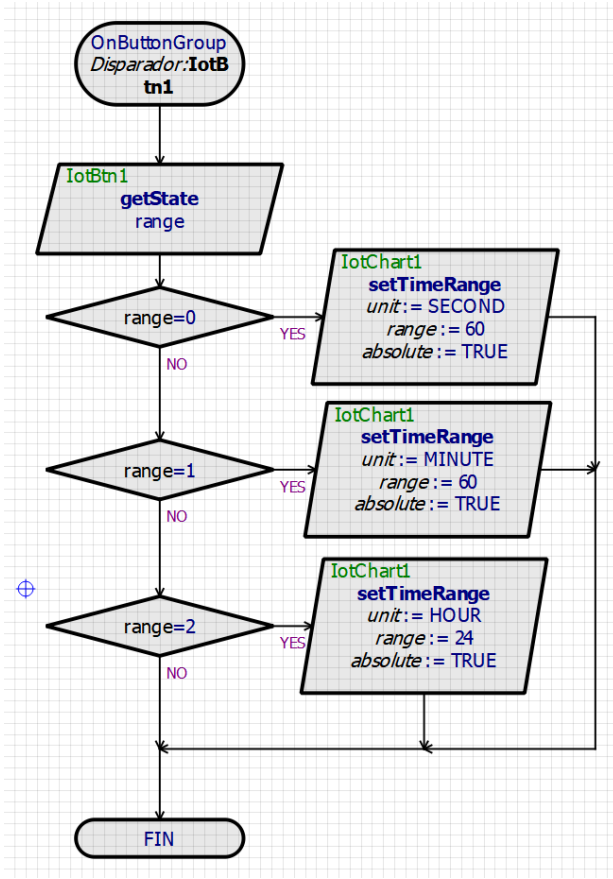
Tenemos que hacer la misma operación para el pulsador de los minutos (range = 1).



Y el para el pulsador de las horas (range = 2).



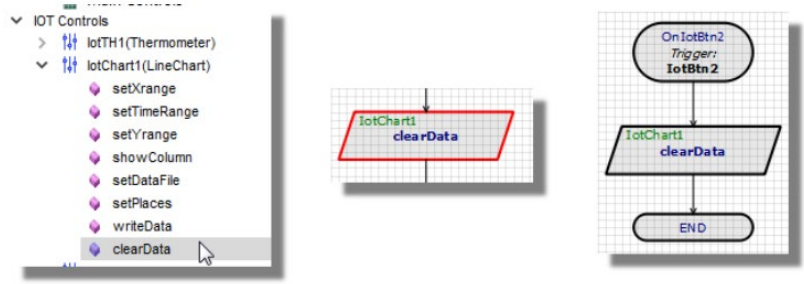
Al terminar, al subrutina que controla el grupo de pulsadores debe tener este aspecto.



4.6.5.-Control del pulsador ‘iniciar’.

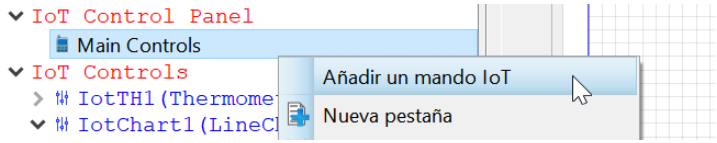
El último paso consiste en programar la respuesta que debemos tener cada vez que el usuario actúa sobre el pulsador de ‘iniciar’. Esta vez usaremos la subrutina ‘onResetButton’ que también creamos al principio de este tutorial para gestionar los eventos asociados con este pulsador.

En la rama ‘IotChart1’ del árbol de estructura de proyecto tenemos disponible el método ‘clearData’ que tendremos que ‘coger y arrastrar’ al comienzo de la subrutina.

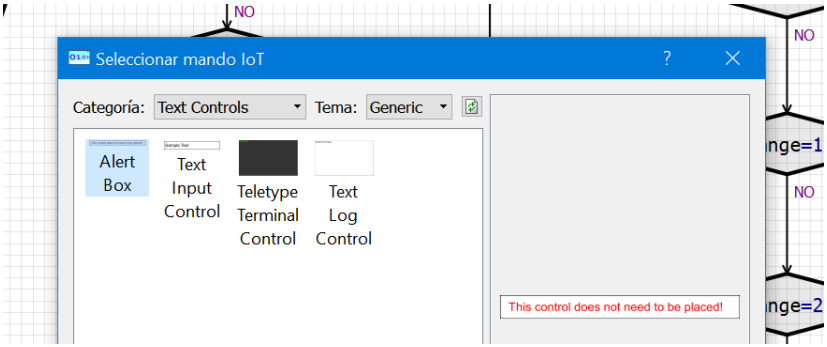


Puesto que cada vez que el usuario actúe sobre el pulsador ‘iniciar’ se van a borrar todas las lecturas almacenadas sin posibilidad de marcha atrás, parece razonable ofrecer al usuario una oportunidad de confirmar que realmente es ésta la operación que desea llevar a cabo y evitar posibles errores.

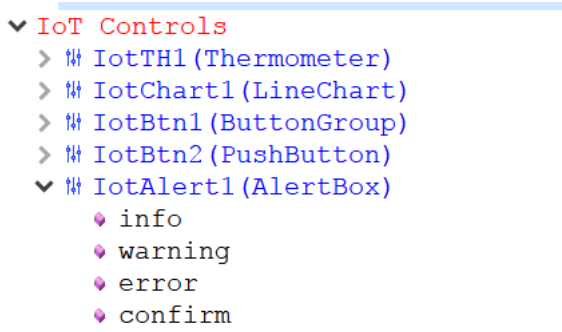
Afortunadamente, el taller IoT nos facilita una manera de llevarlo a cabo de forma muy sencilla utilizando un mando del tipo ‘AlertBox’. Para usar un mando de este tipo, tenemos que usar la opción ‘añadir un mando IoT’.



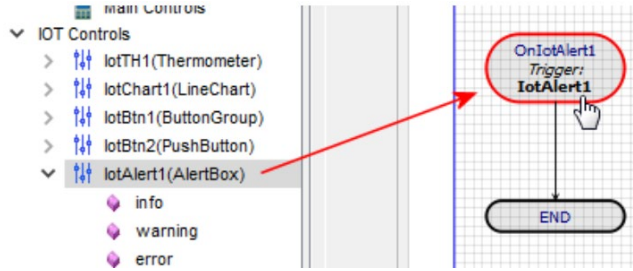
Y seleccionar un mando del tipo ‘Alert Box’ que se encuentra dentro de la categoría ‘Text Controls’.



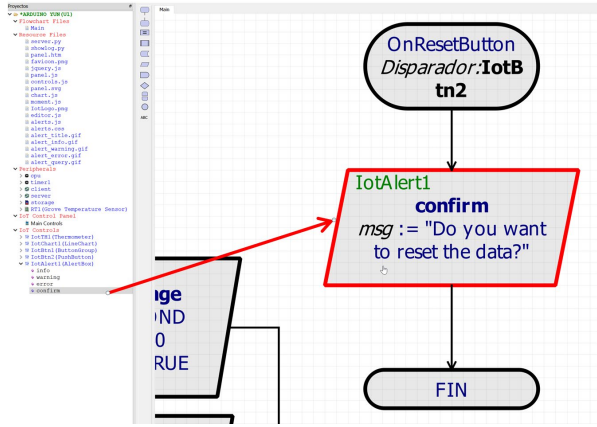
A diferencia de los mandos que hemos utilizado con anterioridad, este tipo de mando no es necesario ubicarlo sobre el panel del cuadro de mando remoto. Se creará una nueva línea dentro del árbol de estructura de Proyecto, de la que cuelgan todos los métodos disponibles como es habitual. En este caso tenemos eventos para mostrar un mensaje informativo, de aviso, de error o de solicitud de confirmación.



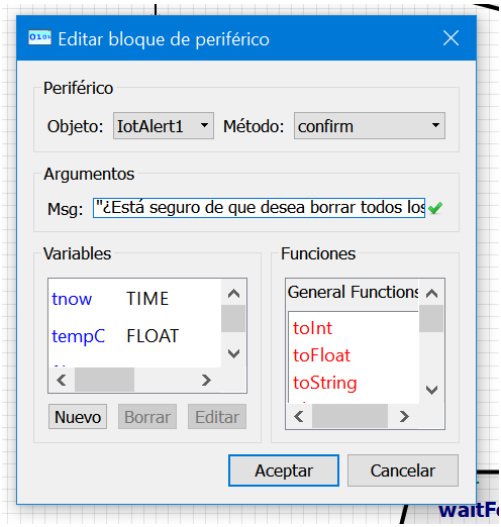
Usando la técnica ‘coger y arrastrar’ podemos llevar el mando ‘Iotalert1’ que acabamos de crear a cualquier sitio libre de nuestro flujograma y se creará automáticamente una subrutina para gobernar los mensajes de alerta. El nombre de la subrutina será ‘OnIotAlert1’.



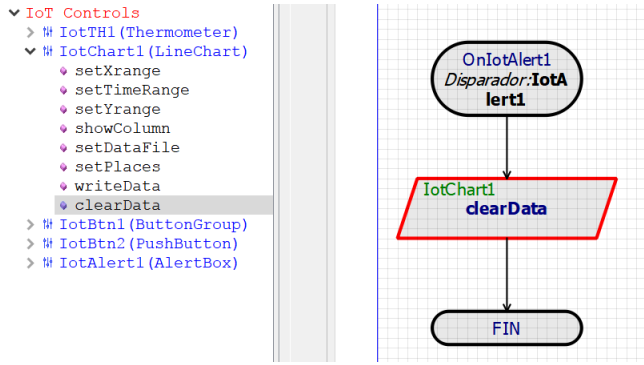
Como queremos configurar un mensaje para solicitar la confirmación del usuario antes de borrar el histórico, usaremos la técnica ‘coger y arrastrar’ para llevar el método ‘confirm’ asociado al pulsador dentro de la subrutina que controlaba los eventos del pulsador.



Podemos editar el bloque para ajustar el texto que se presentará al usuario a nuestro gusto.



Sólo nos queda limpiar el histórico de datos cuando el usuario haya confirmado que realmente quiere llevar a cabo esta acción. Lo haremos arrastrando el método 'clearData' de la rama 'IotChart1' del árbol de estructura del proyecto dentro de la subrutina 'onIotAlert1' que se ejecuta cada vez que se abre el cuadro de diálogo.



Es importante tener en cuenta que una ventana de aviso se implementa en dos fases. Primero tenemos que tener un método que se encarga de presentar la ventana al usuario con el mensaje y donde se le pide una respuesta. Y en segundo lugar tenemos que manejar una subrutina que responde al evento que se produce

cuando el usuario pulsa sobre el botón de aceptación mostrado en la ventana.

La subrutina de respuesta al evento solo se ejecuta cuando el usuario pulsa el botón de 'ok' y siempre se ejecuta los bloques dentro de esa rutina independientemente del texto de la pregunta que hayamos definido. Si existe la opción de 'cancelar' y el usuario pulsa este botón no se ejecuta esa subrutina.

No tenemos forma de responder a la pulsación del usuario sobre el botón 'Cancelar'. Lo único que ocurre, si lo hace, es que la subrutina de respuesta al evento se cancela (es decir, no se ejecuta) y el programa retorna al punto siguiente desde donde se llamó a esta subrutina, continuando la ejecución normal del código.

Puesto que la respuesta del usuario se gestiona desde una subrutina gobernada por un evento, significa que mientras el usuario decide la respuesta el programa principal sigue ejecutándose con normalidad. En nuestro caso, seguiríamos acumulando lecturas en el histórico si el usuario se tomara más de 30 segundos en responder.

> ⚠ En otros proyectos, deberá ser tarea del programador analizar si el programa debe seguir ejecutándose mientras el lector lee el mensaje y da su respuesta o si debe detenerse la ejecución del programa hasta obtener una respuesta.

4.6.6.-Sumario de las tareas llevadas a cabo para escribir el código.

Repasemos, a modo de recopilatorio, todas las tareas que fueron necesarias llevar a cabo para crear un histórico de las medidas de temperatura en una gráfica y visualizar la temperatura actual en un termómetro. Las fases serán muy parecidas en la mayoría de los proyectos que desarrollemos.

1. Crear el cuadro de mando remoto (interruptores, pulsadores, potenciómetros, gráficas, etc.) y controlarlos desde el flujograma usando las subrutinas de eventos. Usar la técnica 'coger y arrastrar' para usar los métodos asociados que necesitemos usar en nuestro flujograma y añadir los bloques de código que sean necesarios.

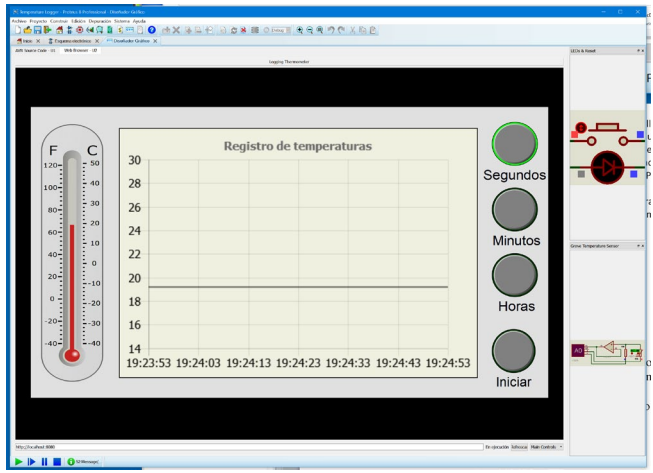
2. Analizar el tiempo que tarda en ejecutarse nuestra subrutina 'loop'. Cuán rápido se ejecuta, cuán rápido es necesario que se ejecute y cuánto tiempo hay que esperar para poder gestionar los eventos asociados con las órdenes recibidas desde el cuadro de mando remoto.
3. Utilizar siempre un método 'waitForRequest' en lugar del método 'delay' para permitir la respuesta a las órdenes de usuario.
4. En caso de duda de cómo hay que gestionar un periférico, conviene consultar los métodos asociados a este periférico que están disponibles en el árbol de estructura del proyecto.
5. En caso de duda de cómo hay que gestionar un mando de un cuadro de mando remoto, conviene consultar los métodos asociados a este mando que están disponibles en el árbol de estructura del proyecto.

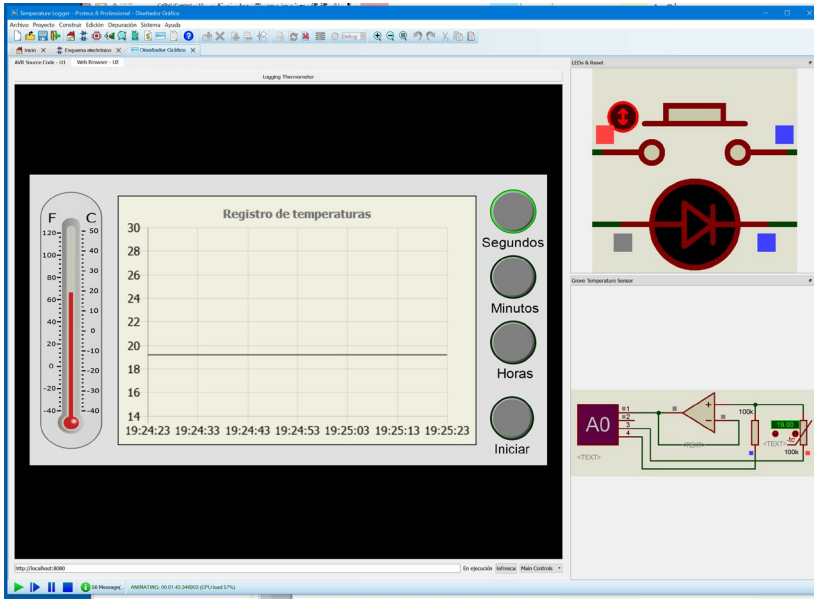
4.7.-Simulación del proyecto.

Una vez que hemos terminado de diseñar el cuadro de mando remoto y de escribir nuestro programa ha llegado el momento de hacer una prueba de cómo funciona todo el sistema. Podemos poner en marcha la simulación utilizando el botón 'play' del cuadro de control de la simulación situado en la zona inferior de la ventana de Proteus.



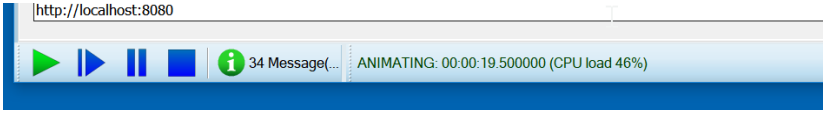
Proteus compilará todo el código comprobando si se ha llevado a cabo alguna modificación desde la última vez que se ejecutó la simulación y arrancará mostrando tres ventanas: en una tenemos el cuadro de mando remoto, en otra el pulsador de 'reinicio' que se encuentra situado en la placa del Arduino Yun y en la tercera el sensor de temperatura. Podemos ajustar el tamaño de las tres ventanas para ajustarlas a nuestra conveniencia. Por ejemplo, en este tutorial, tendría sentido agrandar la ventana donde se muestra el sensor de temperatura para poder simular las lecturas de forma más cómoda y observar los resultados.



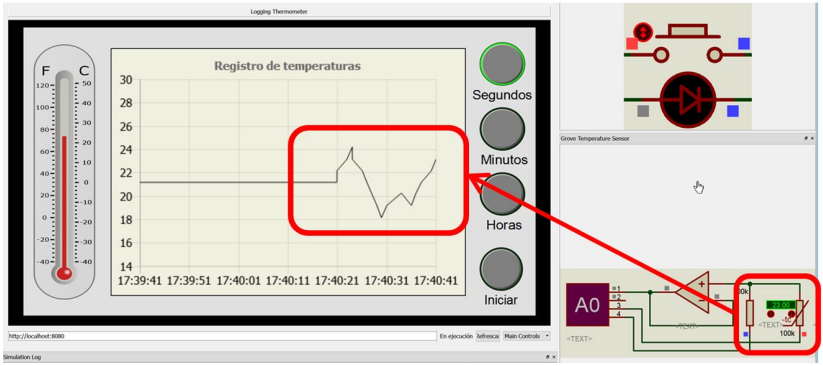


Es importante que nos fijemos que, por defecto, el grupo de pulsadores tiene seleccionado el rotulado como segundos y, por ese motivo, el eje-x del gráfico muestra un intervalo de un minuto y con marcas de 10 en 10 segundos. Además mostrará la hora actual correctamente.

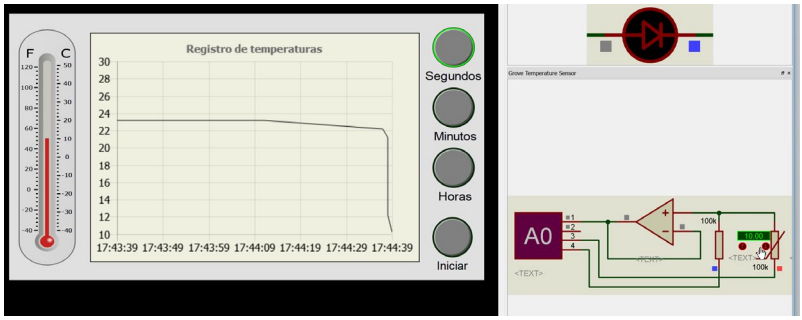
El gráfico avanzará automáticamente (se irá desplazando) cada vez que el tiempo avance un minuto. Es posible que si el ordenador donde se está llevando a cabo la simulación no tiene suficiente potencia la simulación no se desarrolle exactamente a la velocidad real. Pero, en este caso, Proteus muestra en la barra de estado el tiempo real de simulación y, aunque a 'cámara lenta' la simulación se realizará completa y sin pérdidas de información.



La primera prueba sencilla que podemos llevar a cabo es aumentar y reducir el valor de la lectura de temperatura que el dispositivo está recogiendo. Para hacerlo, debemos utilizar los dos botones del sensor de temperatura que se representa en el panel inferior derecha. Cada vez que modifiquemos el valor de la lectura de temperatura tenemos que observar que se modifica en el visor que se encuentra en el sensor y que la curva de registro de temperatura varía en consonancia.

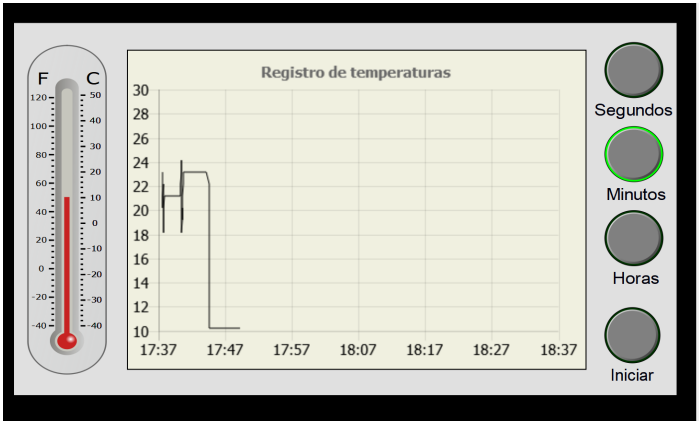


A continuación podemos mantener el ratón sobre el pulsador de descenso de la temperatura y observar lo rápido que la curva reacciona para mostrar las nuevas lecturas.

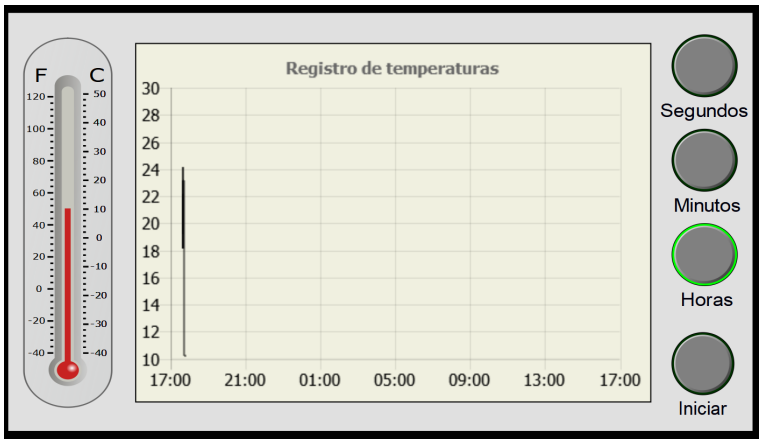


> Este es un buen momento para detenernos un momento a reflexionar, comparar y valorar las ventajas de realizar las pruebas de nuestros equipos utilizando la simulación de Proteus en lugar de tener que llevar a cabo las pruebas sobre equipos en la vida real. ¿Se imagina lo que nos costaría lograr que el sensor de temperatura registrara lecturas variadas utilizando un secador de pelo para aumentar la temperatura o un ventilador para reducirla y generar los cambios de temperatura deseados en su entorno de pruebas?

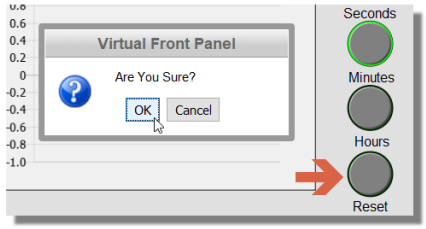
Después de más o menos un minuto, los datos se mueven y van desplazándose a la izquierda del gráfico. Es un buen momento para cambiar el rango de la gráfica a una hora (usando el pulsador de los minutos) y comprobar que el rango de tiempo es adecuado para visualizar los datos que hemos recogido y observar que toda la información se está almacenando correctamente.

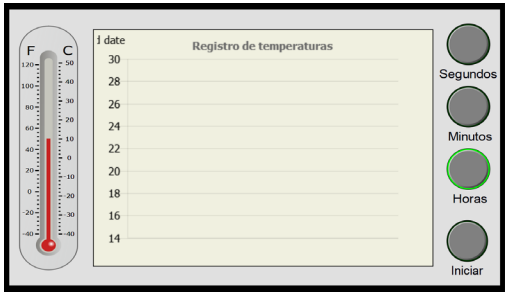


Lógicamente, si usamos el pulsador de horas, las lecturas registradas hasta ahora ocuparan mucho menos espacio dentro de la gráfica pues el rango de la gráfica será ahora de un día.



Por último, podemos comprobar que si usamos el pulsador iniciar, nuestros datos son eliminados y se muestra una gráfica vacía. Además, podemos comprobar que aparece la ventana de aviso para que el usuario tenga que confirmar esta operación.

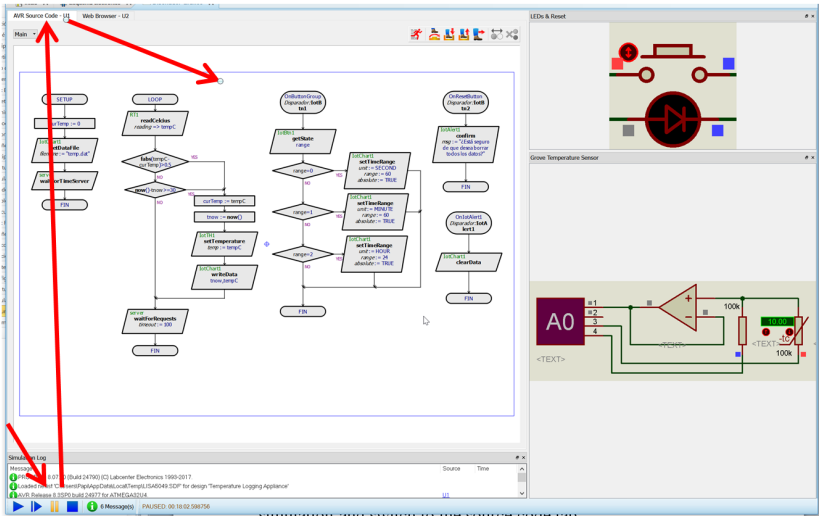




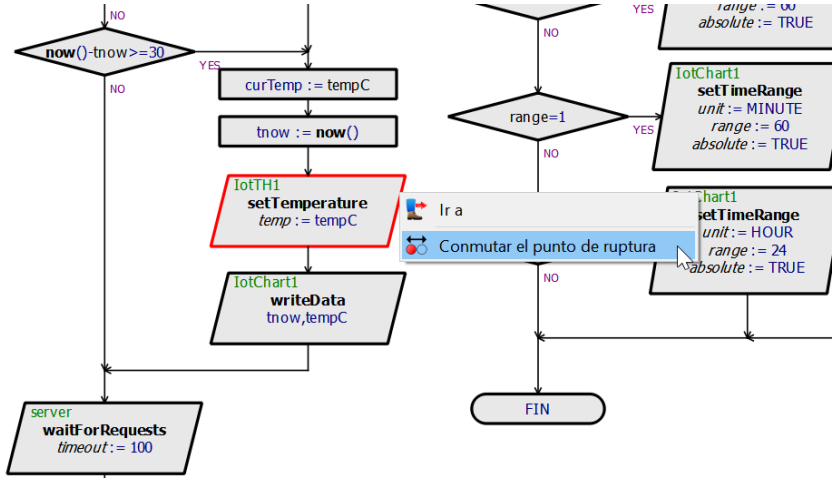
Como vimos en el primer tutorial, Proteus nos permite usar nuestro teléfono móvil o tableta para comprobar cómo funciona el cuadro de mando remoto en la vida real, aunque el dispositivo siga siendo simulado en Proteus. Nuestra última prueba en este segundo tutorial podría ser precisamente esa.

4.8.-Depuración básica del proyecto.

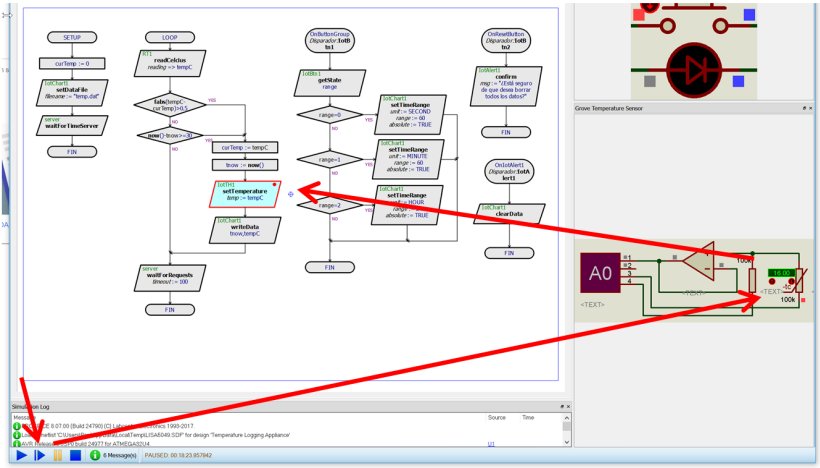
Después de simular el funcionamiento de nuestro proyecto podemos encontrarnos que alguna cosa no se lleva a cabo como esperábamos y que sea necesario investigar la causa del problema. Veamos las diferentes formas que Proteus pone a nuestra disposición de detener la ejecución del programa y ejecutarlo paso a paso para poder mirar con detalle cómo nuestro código se va ejecutando a medida que se desarrollan los hechos y poder descubrir que va mal.
Empecemos presionando el botón de pausa del panel de control de la simulación para detenerla y activar la pestaña donde se muestra el código fuente.



A continuación, colocamos un punto de ruptura en el bloque ‘setTemperature’ dentro de la subrutina ‘loop’ utilizando el menú contextual.



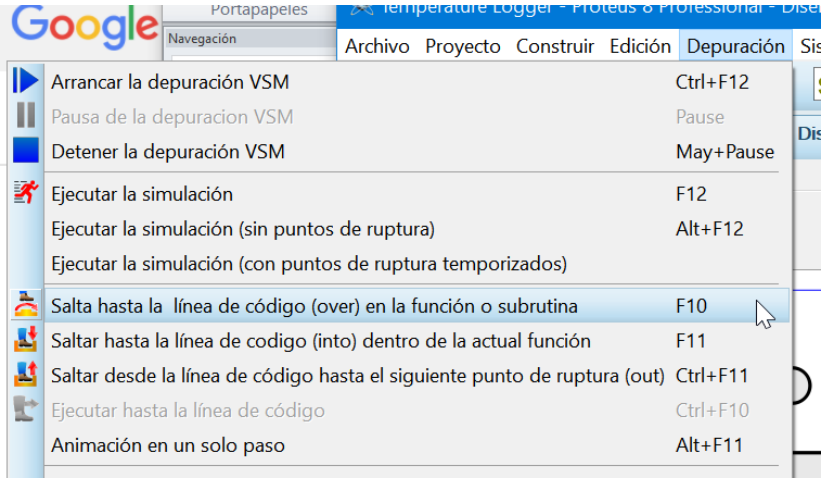
Ahora volvemos a poner la simulación en marcha y cambiamos la temperatura utilizando el sensor.



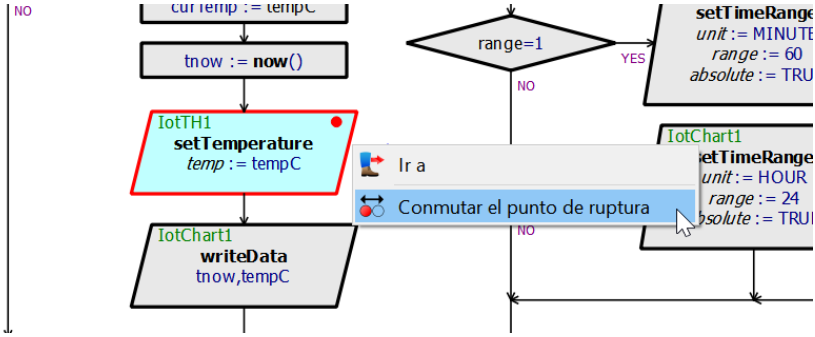
Comprobaremos que la ejecución del programa se detiene ella sola de forma automática y que el panel del código fuente vuelve a presentarse con el bloque de código que se acaba de ejecutar resaltado de otro color. Lógicamente se trata del bloque donde hemos puesto el punto de ruptura.

Ahora podemos ejecutar el programa paso a paso usando los botones de la barra de herramienta de depuración, las opciones del menú 'depuración' o los atajos de teclado correspondientes

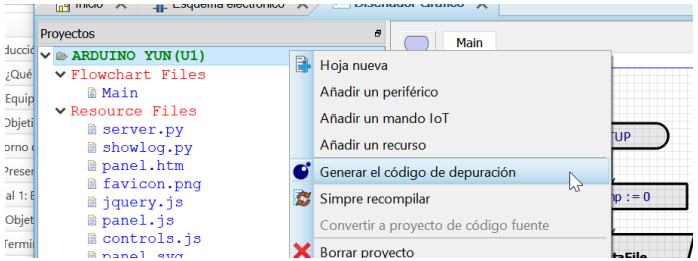




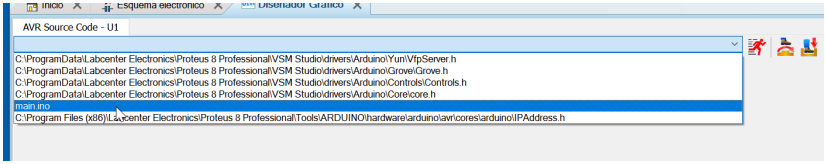
Por último, podemos eliminar el punto de ruptura cuando hayamos acabado la depuración paso a paso, utilizando de nuevo el menú contextual del bloque de código.



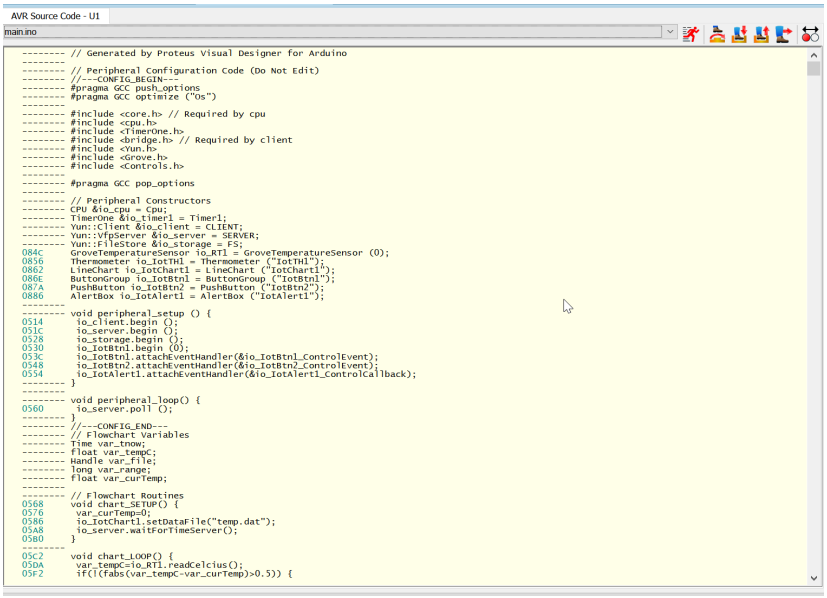
De la misma forma que hemos llevado a cabo la depuración utilizando el flujograma, es posible realizar la depuración cuando utilizamos la programación directamente en código fuente. Cuando la simulación esté detenida, abrimos el menú contextual de la rama principal (Arduino Yun) del árbol de proyecto usando el botón derecho del ratón y seleccionamos la opción 'Generar el código de depuración'.



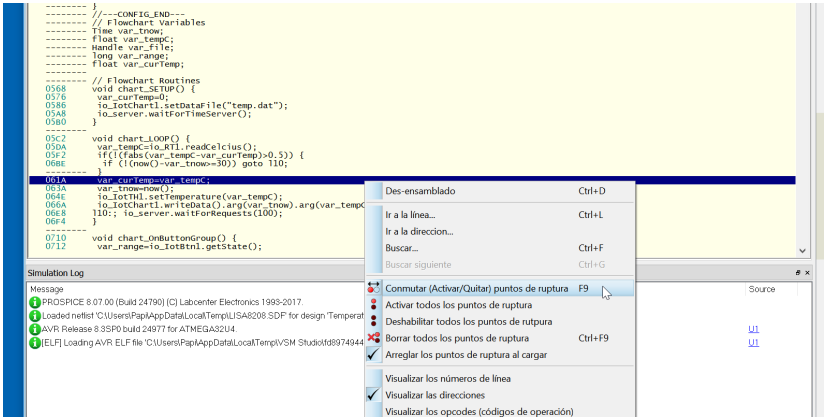
En la zona superior del panel del código fuente y justo a la izquierda de la barra de herramientas de depuración podemos abrir la lista desplegable y seleccionar el fichero 'main.ino'.



El código fuente en lenguaje C aparecerá en el panel del código fuente.



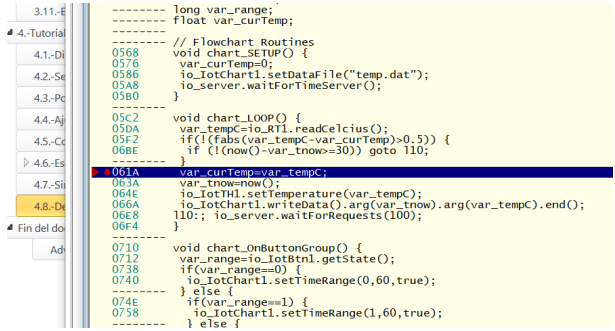
Podemos buscar la línea de código que se corresponde con el mismo bloque de flujograma donde pusimos un punto de ruptura en la prueba anterior: ‘var_curTemp = var_tempC;’ y con ayuda del menú contextual fijar un punto de ruptura en esta línea.



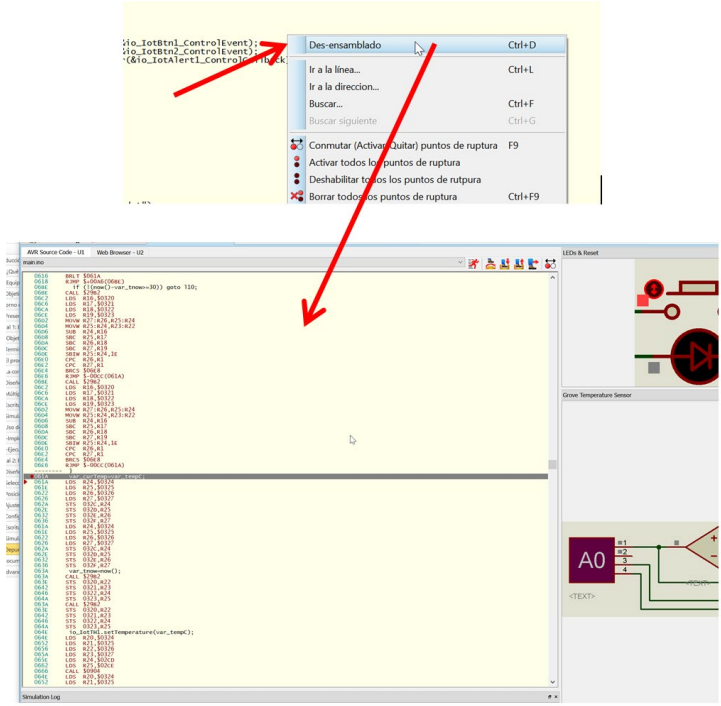
Una vez puesto el punto de ruptura, observaremos que se pone un punto rojo a la izquierda de esa línea.

```
05B0      }
--------
05C2      void chart_LOOP() {
05DA       var_tempC=io_RT1.readCelcius();
05F2       if(!(fabs(var_tempC-var_curTemp)>0.5)) {
06BE        if (!(now()-var_tnow>=30)) goto l10;
--------   }
061A       var_curTemp=var_tempC;
063A       var_tnow=now();
064E       io_IotTH1.setTemperature(var_tempC);
066A       io_IotChart1.writeData().arg(var_tnow).arg(var_tempC).end();
06E8      l10:; io_server.waitForRequests(100);
06F4      }
--------
0710      void chart_OnButtonGroup() {
0712       var range=io_IotBtn1.getState();
```

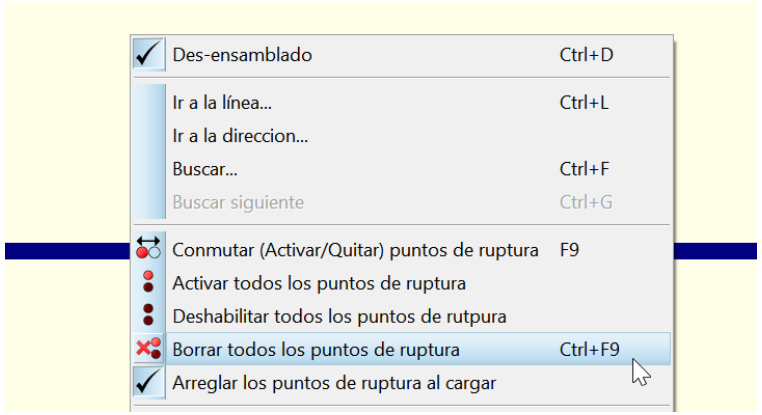
Podemos ejecutar la simulación y después de unos pocos segundos, cuando el cuadro de mando remoto se muestre, podemos volver a ajustar la temperatura utilizando los botones del sensor de temperatura. Podremos observar, que igual que en el caso anterior, la simulación se detiene y se muestra la línea de código que se acaba de ejecutar (precisamente sobre la que hemos puesto el punto de ruptura).



> Si lo deseamos podemos realizar la depuración sobre el código en ensamblador pulsando el botón derecho del ratón sobre el panel que muestra el código fuente en C y seleccionando la opción 'Des-ensamblado' del menú contextual.

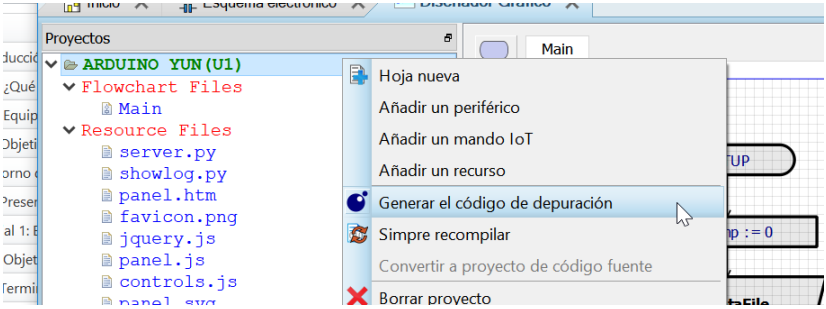


Podemos fijar tantos puntos de ruptura como deseemos. Y eliminarlos todos de un golpe desde el menú contextual seleccionado la opción 'Borrar todos los puntos de ruptura'.



Podemos pasar del ensamblador al lenguaje C volviendo a utilizar la opción 'Des-ensamblado' del menú contextual (es un conmutador y podemos ver su estado por la marca situada a la izquierda.

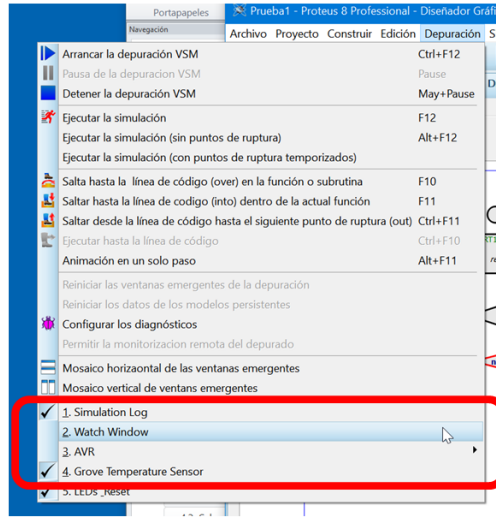
Si queremos volver a la depuración usando el flujograma, tenemos que detener la simulación. Cuando la simulación esté detenida, abrimos el menú contextual de la rama principal (Arduino Yun) del árbol de proyecto usando el botón derecho del ratón y seleccionamos la opción 'Generar el código de depuración' (también es un conmutador y, en este caso, la marca es un perfil que simula un recuadro en 3D alrededor del icono situado a la izquierda de la opción).

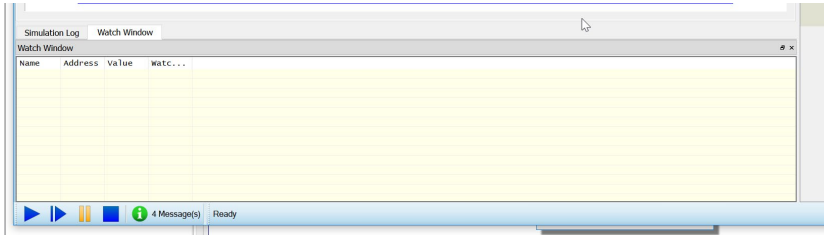


4.9.-Depuración avanzada del proyecto.

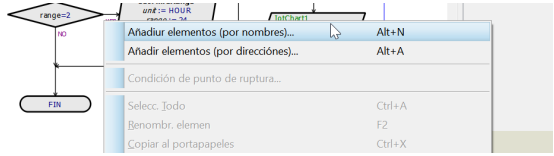
Si estamos familiarizados con los microprocesadores de la familia AVR y/o nos encontramos cómodos en el manejo del esquema electrónico de nuestro dispositivo, hay un par de técnicas útiles que pueden ayudarnos para controlar la simulación y llevar a cabo una depuración más avanzada.

La primera es utilizar la ventana de inspección para controlar el flujo del programa. Como siempre el primer paso es poner la depuración en pausa. A continuación podemos abrir la ventana de inspección utilizando la opción 'Watch window' del menú 'depuración'.



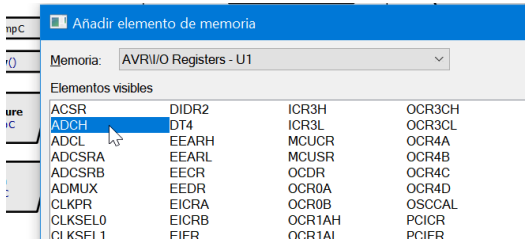


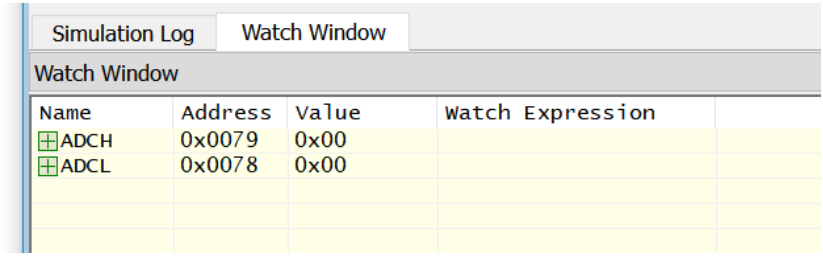
En este caso nos interesa mirar los valores del registro ADC porque el sensor de temperatura está conectado al microprocesador AVR utilizando el puerto ADC y su lectura se graba en este registro. Usaremos el botón derecho del ratón sobre la ventana de inspección y seleccionaremos la opción 'Añadir elementos (por nombres)' del menú contextual.



En la ventana emergente que aparece seleccionaremos el registro ADCH (el byte alto) haciendo un doble click sobre él y observaremos

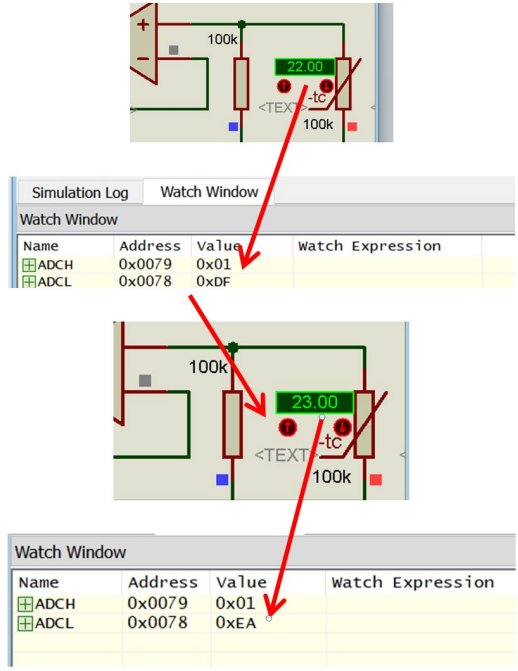
que se añade a la ventana de inspección. A continuación haremos lo mismo con el registro ADCL (el byte bajo).



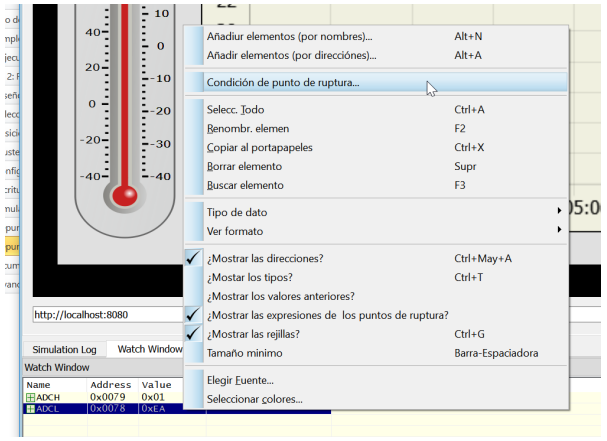


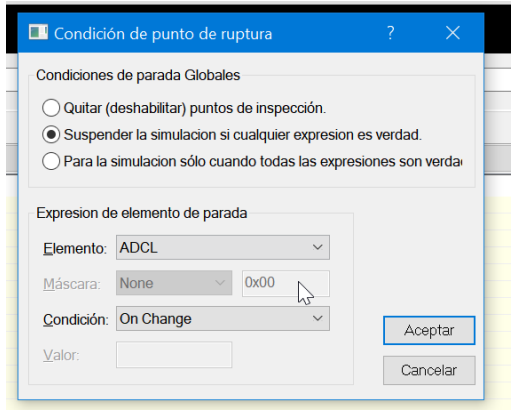
Podemos observar como el valor 'Value' del registro ADCL (byte bajo) se actualiza cuando cambiamos la temperatura.

> Para una temperatura de 22ºC el valor del registro es 0x01DF (479 en decimal) y para la temperatura de 23ºC es valor es 0x01EA (490 en decimal). Los dos primeros dígitos en hexadecimal se guardan en el registro H y los dos segundos en el registro L.



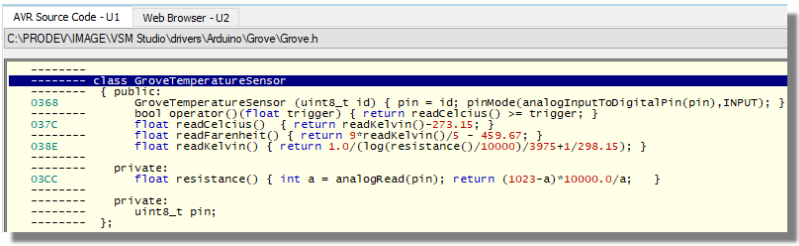
Podemos utilizar el botón derecho del ratón sobre el registro ADCL para abrir el menú contextual y seleccionar la opción ‘Condición de punto de ruptura…’. Y seleccionar la opción cuando se produzca un cambio.





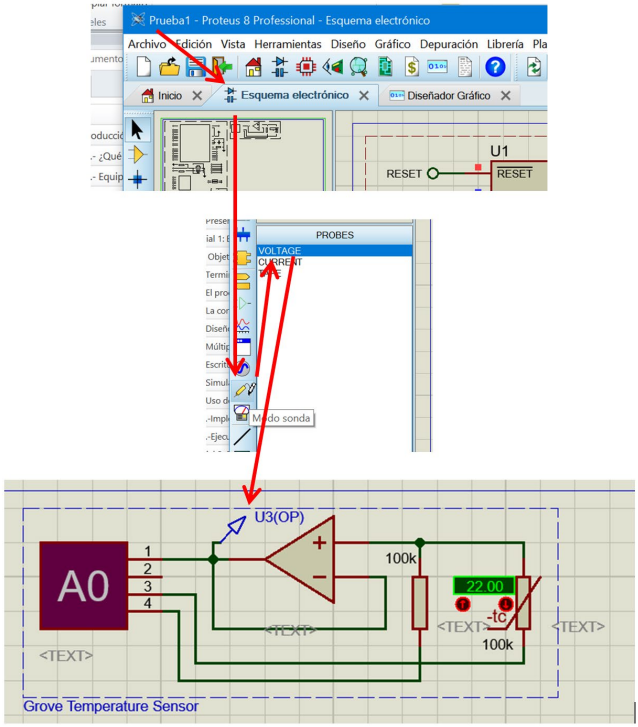
De esta forma, cada vez que se produzca un cambio de temperatura, el conversor ADC modificará el registro ADC con un nuevo valor y la simulación se detendrá.

Si deseamos entender con detalle lo que ocurre podemos conmutar al depurador del código fuente y repetir el experimento presionando la tecla F10 cuando se alcance el punto de ruptura. Con ello veremos el código que se encarga de hacer la conversión analógico-digital dentro del fichero grove.h que controla el sensor de temperatura seleccionando este fichero en el desplegable superior.

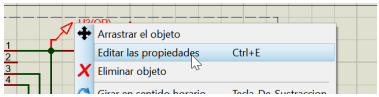


También podemos fijar condiciones de ruptura de la simulación enlazadas al hardware en lugar de al software. Por ejemplo, podemos poner una excepción enlazada a un cambio de voltaje en un determinado punto del circuito.

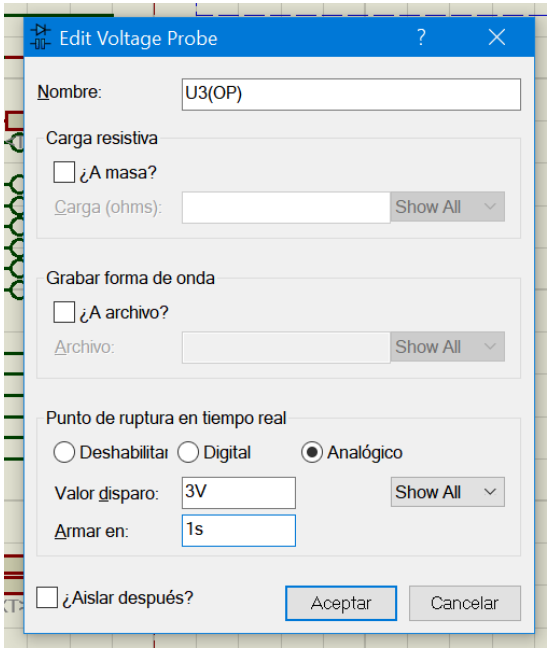
Nos cambiaremos a la pestaña del diseño electrónico, seleccionaremos una sonda de tensión y la posicionaremos a la salida del amplificador operacional del sensor de temperatura Grove.



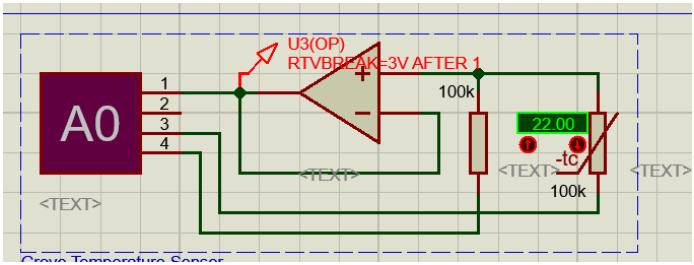
Seleccionamos la sonda que hemos colocado y con el botón derecho abrimos el menú contextual. Seleccionamos la opción ‘Editar las propiedades’



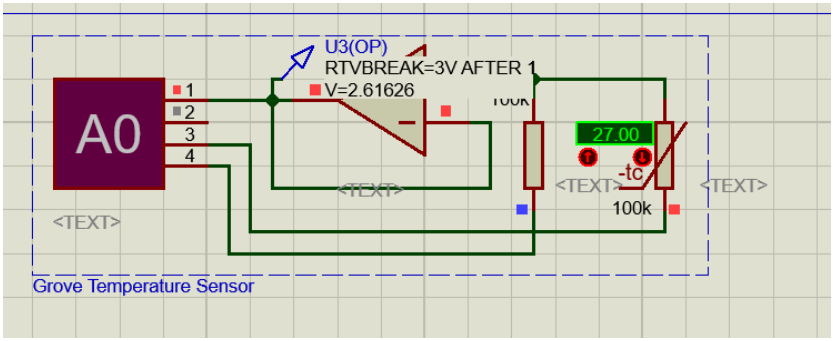
En la ventana de diálogo que aparece, seleccionamos la opción ‘punto de ruptura en tiempo real’, luego la opción ‘analógico’, un valor de disparo de 3V y armar en 1s (de esta forma evitamos los ruidos que se producen durante el arranque y estabilización del sistema).



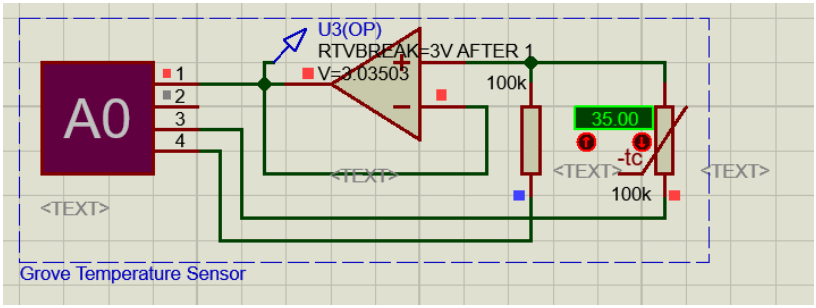
Al guardar los cambios, tenemos que ver que junto a la sonda aparecen las condiciones del punto de ruptura de hardware.



Volvemos a poner en marcha la simulación. A medida que subimos la temperatura, se incrementa el valor del voltaje a la salida del amplificador operacional.



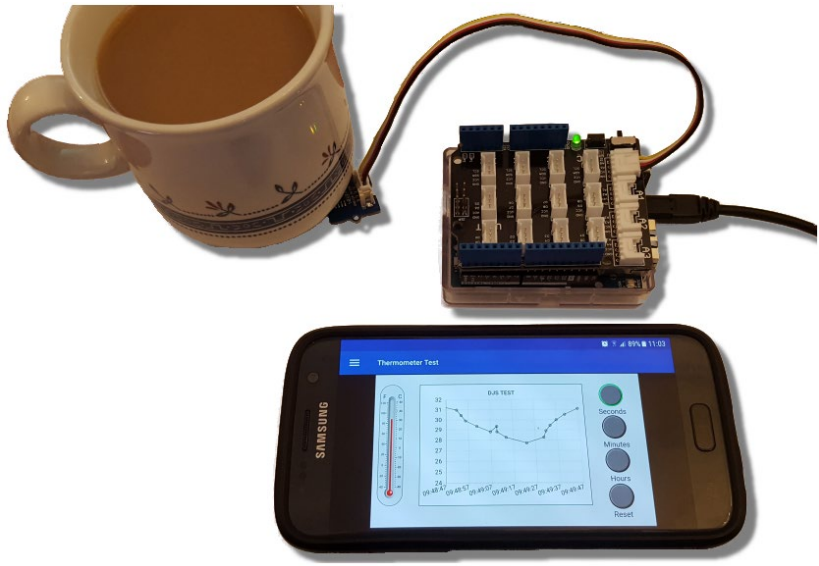
Cuando alcancemos la temperatura de 35ºC el valor de tensión a la salida del amplificador operacional alcanzará los tres voltios y la simulación se detendrá.



Estos sólo son dos ejemplos de las posibilidades de depuración que nos brinda Proteus y obviamente cada uno de ellos ha realizado el mismo trabajo. Esto ha sido provocado porque nuestro tutorial es intencionadamente sencillo y nuestro objetivo sólo es mostrar las técnicas de depuración disponibles. Las completas herramientas de depuración profesional integradas en Proteus están totalmente disponibles para lograr establecer condiciones complejas y de muy variado tipo para depurar prácticamente cualquier proyecto con el que estemos trabajando.

4.10.-Implemenación del proyecto en nuestro equipo de la vida real.

Todo lo que hemos visto que sucede durante la simulación realizada con Proteus podemos comprobar que se corresponde con la ejecución real si cargamos el código en nuestro equipo Arduino Yun y utilizamos el cuadro de mando remoto desde un teléfono móvil.



5.-Tutorial 3: Un proyecto desarrollado usando código C.

Para los programadores más experimentados, el taller IoT es una magnífica herramienta que puede ser utilizada perfectamente en proyectos desarrollados utilizando el lenguaje de programación C++. Puesto que nuestros proyectos utilizarán para su funcionamiento equipos Arduino, para poder llevar a cabo este tipo de proyectos con Proteus es necesario disponer de la licencia VSM Taller IoT para Arduino y alguna de las licencias siguientes de Proteus:

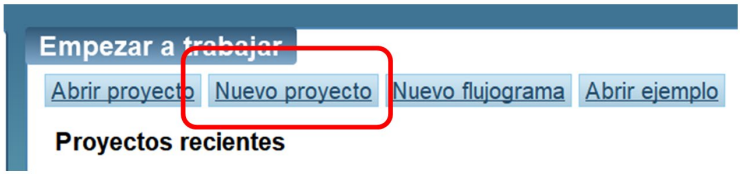
- VSM Diseñador Gráfico para Arduino.
- VSM Atmel AVR.
- VSM Arduino AVR.
- Platinium Edition[4]

El proceso para construir un proyecto para Arduino utilizando la programación en C++ es muy parecido al que vimos en los tutoriales donde desarrollamos proyectos usando el Diseñador Gráfico, aunque hay un par de diferencias importantes referentes al asistente para crear un proyecto nuevo y al diseño del cuadro de mando que vamos a estudiar con detalle en este apartado.

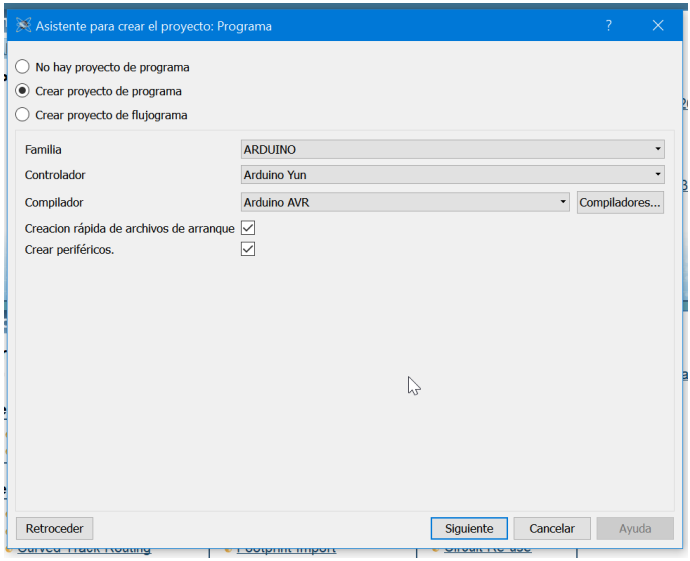
5.1.-El asistente para crear un nuevo proyecto.

Cuando creamos un nuevo proyecto lo más cómodo es usar el asistente utilizando el botón 'Nuevo Proyecto' situado en la pantalla de inicio de Proteus.

[4] Esta licencia es la única que incluye la licencia VSM Taller IoT para Arduino y no es necesario adquirirla por separado.



Podemos ir aceptando las opciones que el asistente nos ofrece por defecto hasta llegar a la ventana donde nos pregunta si queremos crear un proyecto con programa. Debemos elegir un proyecto de programa y no un proyecto de flujograma como vinimos haciendo en los tutoriales anteriores. Como familia elegiremos 'Arduino', como controlador el 'Arduino Yun' y como compilador debemos elegir 'Arduino AVR'. Además debemos comprobar que hemos seleccionado las dos casillas de verificación para crear los archivos de arranque de forma rápida y los periféricos.

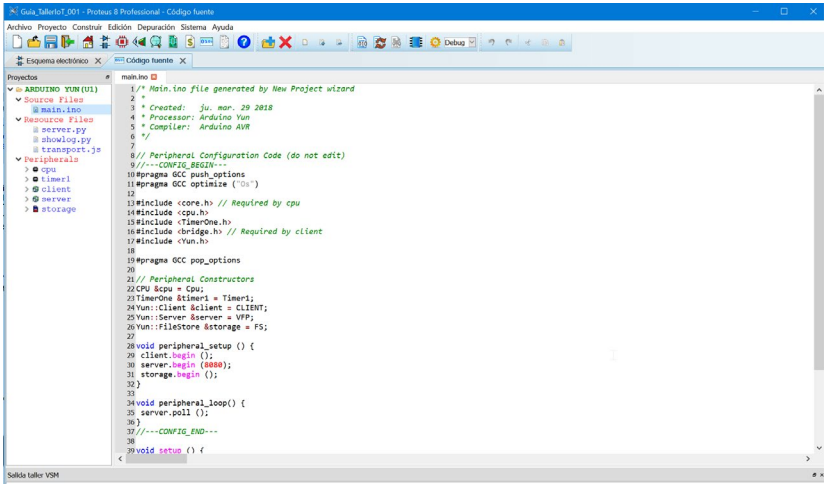


Marcar la opción 'crear periféricos' es imprescindible en este punto porque al seleccionarla posibilitamos que más tarde las diferentes tarjetas de expansión, los módulos accesorios independientes y los módulos de sensores Grove incluidos en Proteus estén disponibles para ser utilizadas en el proyecto. Con esta opción marcada al incluir uno de estos complementos en nuestro proyecto, se asocian de forma

automática todos los controladores de cada tarjeta o módulo al proyecto. De esta manera se facilitan todos los mecanismos de ayuda para utilizarlos cómodamente. Si dejamos desmarcada esta casilla no resultará tan sencillo el uso del taller IoT en nuestro proyecto.

El resto de las opciones que el asistente nos ofrece por defecto nos servirán perfectamente para este tutorial.

Al finalizar el asistente, habrá creado para nosotros un proyecto de código fuente utilizando VSM Studio con una importante cantidad de código en lenguaje C++ generado de forma automática y que podemos manejar desde la pestaña código fuente.



Los lectores que hayan trabajado anteriormente con las MFC (Microsoft Foundation Clasess) u otros entornos de programación similares encontrarán esta forma de trabajo familiar. La clave más importante de esta forma de trabajo es no alterar, cambiar o mover el código generado de forma automática.

Este código primario que se ha generado de forma automática será utilizado por los diversos controladores de dispositivos suministrados por Labcenter con Proteus y se irá actualizando de forma automática a medida que vayamos añadiendo nuevos periféricos y módulos a nuestro proyecto. Por este motivo es necesario que dejemos la gestión de esta parte del código al propio software Proteus.

```
// Peripheral Configuration Code (do not edit)
//---CONFIG_BEGIN---
#pragma GCC push_options
#pragma GCC optimize ("Os")

#include <core.h> // Required by cpu
#include <cpu.h>
#include <TimerOne.h>
#include <bridge.h> // Required by client
#include <Yun.h>

#pragma GCC pop_options

// Peripheral Constructors
CPU &cpu = Cpu;
TimerOne &timer1 = Timer1;
Yun::Client &client = CLIENT;
Yun::Server &server = VFP;
Yun::FileStore &storage = FS;

void peripheral_setup () {
 client.begin ();
 server.begin (8080);
 storage.begin ();
}

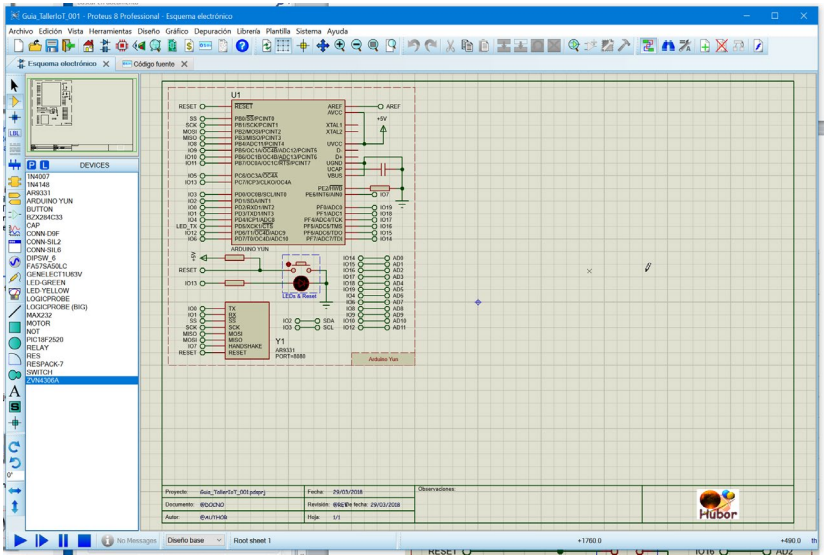
void peripheral_loop() {
 server.poll ();
}
//---CONFIG_END---
```

Debemos ignorar todo este código extra generado y centrarnos en el uso de las dos subrutinas principales, setup() y loop(), para incluir en ellas todo el código que deseemos añadir a nuestro proyecto.

```
void setup () {
peripheral_setup();
// TODO: put your setup code here, to run once:
}

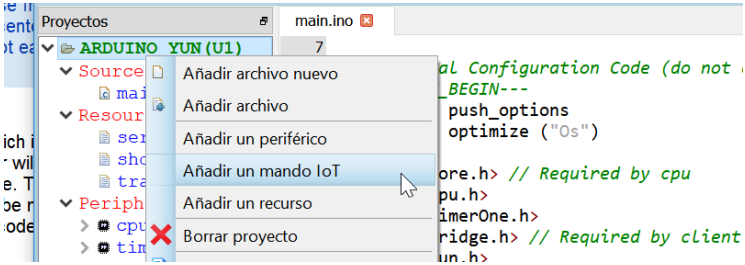
void loop() {
peripheral_loop();
// TODO: put your main code here, to run repeatedly:
}
```

En la pestaña ‘esquema electrónico’ encontraremos que el asistente ha colocado para nosotros una placa Arduino Yun.

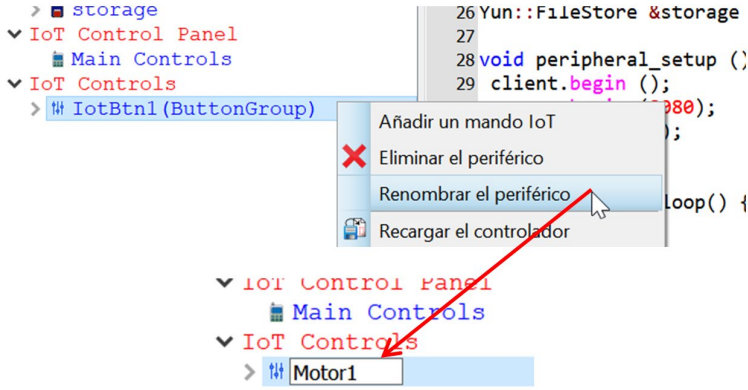


5.2.-El diseño del cuadro de mando remoto.

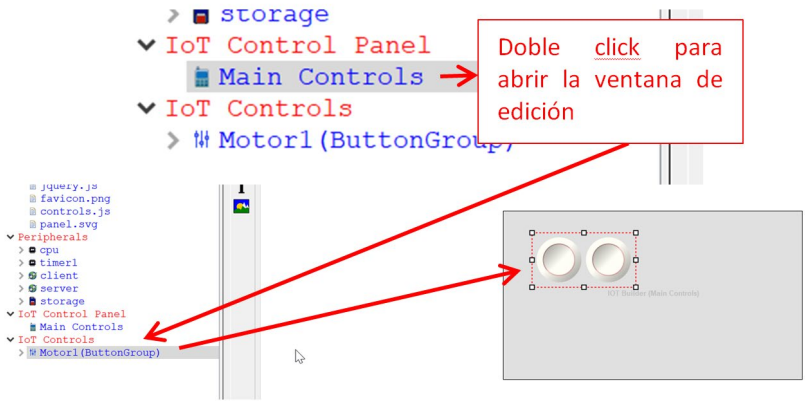
El diseño del cuadro de mando remoto es idéntico al que hemos descrito en los dos tutoriales anteriores cuando usábamos el Diseñador Gráfico. Podemos añadir mandos IoT utilizando el menú contextual de la rama principal del árbol de proyectos.



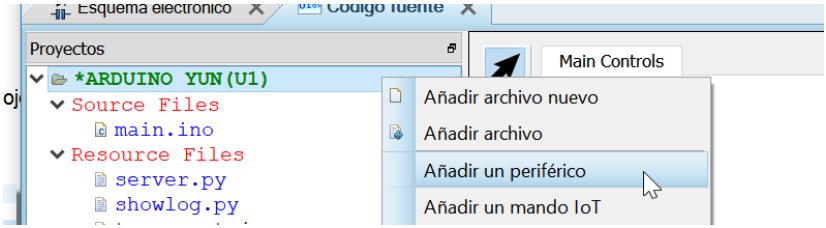
Es una buena práctica darle siempre un nombre al mando creado que nos ayude a su identificación cuando lo usemos a lo largo del desarrollo del proyecto. Por ejemplo, si añadimos un pulsador, podemos cambiarle el nombre desde el menú contextual y darle uno que refleje el uso que le hemos asignado.

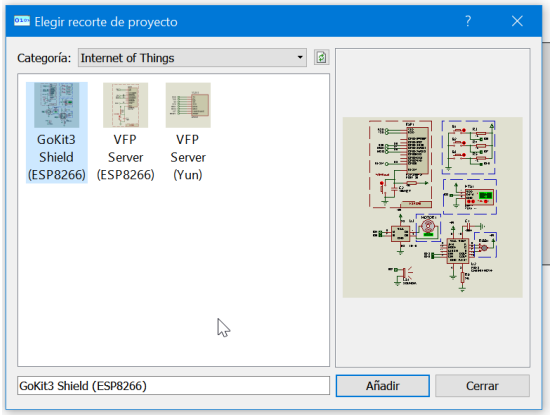


Haciendo un doble click sobre la rama del árbol de proyectos del cuadro de mando remoto, abriremos el editor de los cuadros de mando y podemos ir situando nuestros mandos sobre él utilizando la técnica ‘coger y arrastrar’.

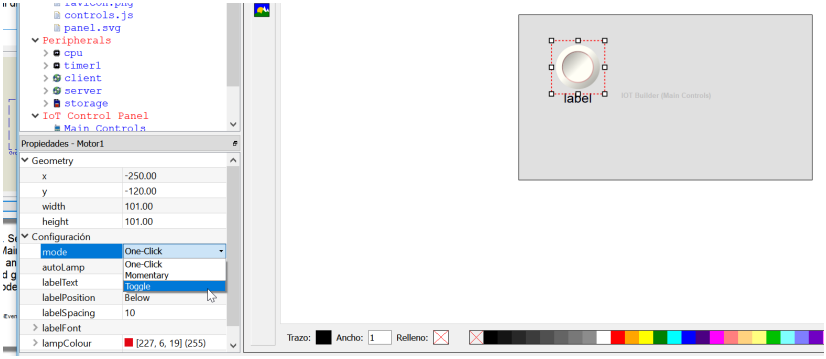


También podemos añadir tarjetas de expansión y módulos a nuestro equipo Arduino utilizando la opción ‘añadir un periférico’ de menú contextual de la rama principal del árbol de proyecto.

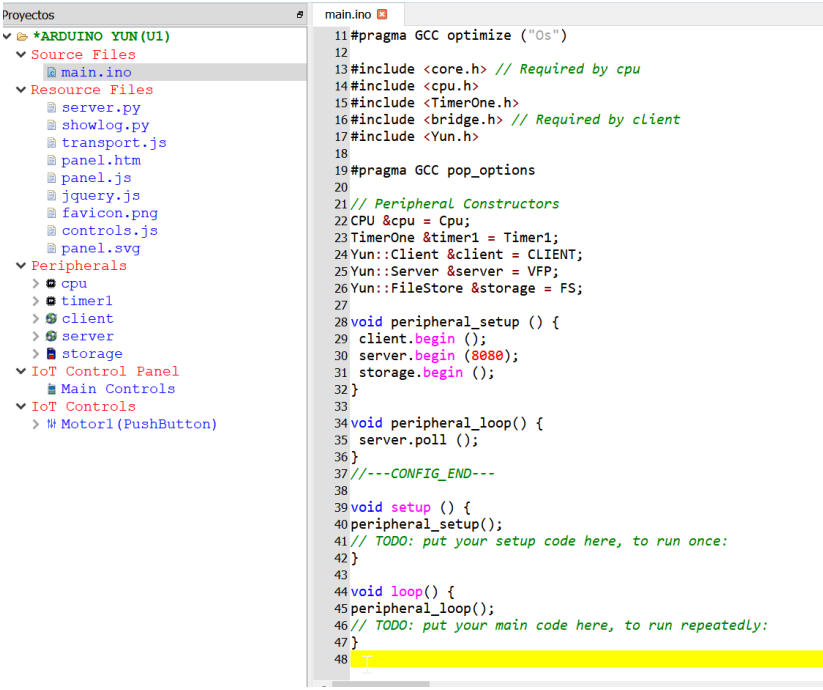




Al añadir nuevos periféricos o mandos de nuestro panel de control ya podemos utilizar los controladores asociados. Así, por ejemplo, para crear la subrutina que gestione el evento que se produce cada vez que el usuario presiona sobre un pulsador podemos hacerlo de forma sencilla. Seleccionamos el pulsador y en la ventana de propiedades elegimos el modo 'toggle'.



Luego haremos un doble click sobre la rama del fichero principal del código (main.ino) para abrir el editor del código y colocaremos el cursor del ratón sobre algún lugar libre, por ejemplo, detrás de la función peripheral_loop().

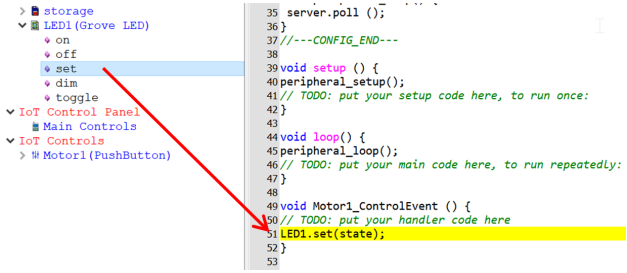


A continuación, cogemos y arrastramos el mando del pulsador que habíamos creado y lo soltamos sobre la ventana de código. Con este simple movimiento, obtenemos un 'armazón' básico de la función que se encargará de gestionar el evento que se lanza al actuar sobre el pulsador. Ya podemos escribir el código que deseemos.

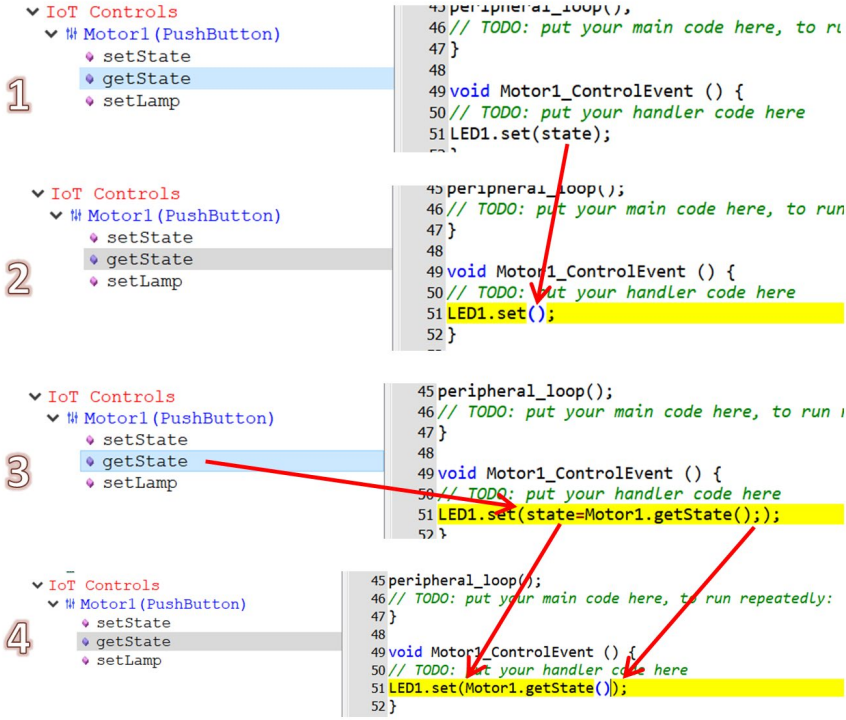
```
45 peripheral_loop();
46 // TODO: put your main code here, to run repeatedly:
47 }
48
49 void Motor1_ControlEvent () {
50 // TODO: put your handler code here
51 }
52
```

> ⚠ Es importante no cambiar los nombres de las funciones que se generan de forma automática o cuando ejecutemos el código del proyecto este no funcionará correctamente. La excepción a esta regla se produce cuando le cambiamos el nombre a un mando o periférico después de que hayamos generado alguna función asociada a ese mando o periférico. En este caso, después de cambiar el nombre al mando o periférico, debemos cambiar el nombre de todas las funciones de gestión de eventos asociadas con él para que coincidan el prefijo de la función con el nombre del mando o periférico.

Para este tutorial queremos encender o apagar el led al cambiar el estado del pulsador. Así que desde la línea del periférico LED 'cogemos y arrastramos' la función set y la colocamos dentro de la función Motor1_ControlEvent() que acabamos de crear y que gestiona los eventos del pulsador

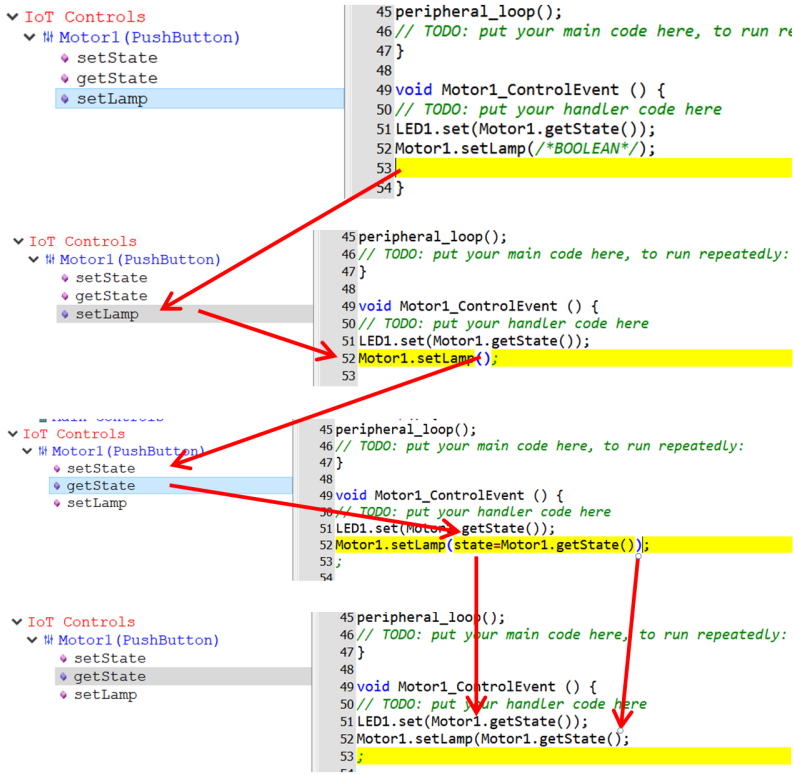


Después de situar la función set() asociada a led, necesitamos añadir una variable booleana que nos ayude a saber si tenemos que encender o apagar el led. La forma más sencilla de hacerlo es insertar el método getState() asociado al pulsador que nos devuelve esta información dentro de la función LED1.set() que acabamos de colocar borrando las palabras que sobran ('state', '=' y ';').



Al no haber configurado el método autoLamp() asociado al pulsador, tenemos que gestionar el halo luminoso que indica el estado del pulsador desde el propio código.

Podemos 'coger y arrastar' el método setLamp del pulsador y colocador detrás del código y volver a insertar el método getState() dentro.



De esta manera nos aseguramos que el led está encendido sincronizado con el halo del pulsador.

Y esto es prácticamente todo lo que haremos en este tutorial. Hay muchas otras cosas que podríamos hacer y funcionalidades que podríamos añadir, pero básicamente con este código conmutaremos el estado del LED situado en el módulo Grove asociado al Arduino Yun cuando actuemos en el pulsador del cuadro de mando remoto (por ejemplo, ejecutándolo desde un teléfono móvil).

Este es el momento adecuado para que el lector, si lo considera necesario, revise los dos tutoriales anteriores para ejecutar las fases de simulación, depuración e implementación de nuestro proyecto. Estas tareas se llevan a cabo de forma casi idénticas en el caso de un proyecto con código fuente u otro con un diagrama de flujo.

6.-La App para móviles y tabletas.

Hoy en día, prácticamente todo el mundo tiene un teléfono inteligente en el que se pueden ejecutar aplicaciones (apps). Por ese motivo Labcenter ha desarrollado la app 'Proteus IoT Builder' que está disponible tanto en la tienda 'Apple app store' como en la tienda 'Google Play store'.

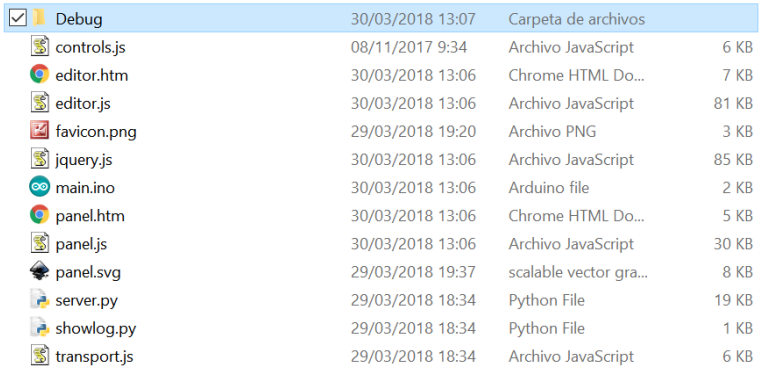
En el epígrafe 3.11 se muestra la instalación y utilización de esta app con todo detalle.

7.-Diseño avanzado de cuadros de mando remoto.

7.1.-Estructura interna del fichero de un cuadro de mando remoto.

El taller de IoT contiene potentes herramientas de edición y se suministra con librerías de mandos de diferentes temas que nos permiten diseñar cuadros de mando remoto de muy diversos aspectos y funcionalidades. Sin embargo, podemos encontrarnos con situaciones en la que necesitamos diseñar mandos diferentes a los suministrados. Podemos llevar a cabo esta tarea usando algunos de los paquetes de diseño gráfico existentes en el mercado.

Los mandos que posicionamos en los paneles son gráficos SVG controlados mediante la utilización de clases escritas en lenguaje Javascript. El fichero principal (llamado panel.htm) contiene un armazón básico escrito en javascript que utiliza los ficheros panel.svg, panel.js y controls.js archivados en el servidor.



Todo el cuadro de mando remoto, independientemente de que contenga una o muchas pestañas, se guarda en un solo fichero llamado panel.svg. A los datos básicos incluidos en el formato svg se

le añade una información adicional complementaria que se guarda dentro del propio fichero panel.svg en el espacio de nombres llamado vfp. Esta información adicional se incluye cumpliendo el estándar svg, así que a nuestros efectos el fichero panel.svg es simplemente un fichero con formato svg que puede ser editado utilizando cualquier editor de dibujo svg.

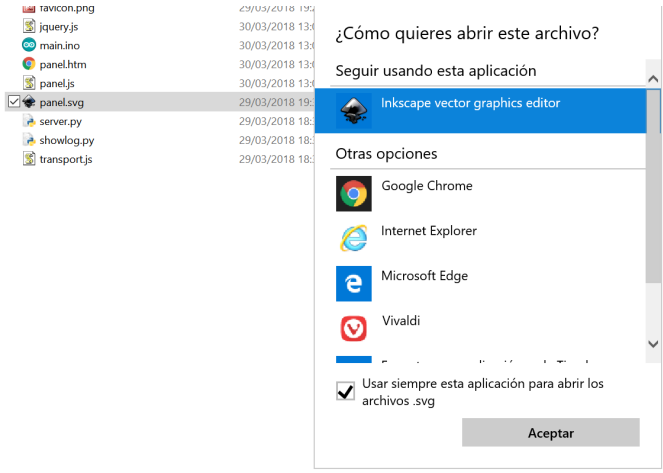
7.2.-Edición avanzada de un fichero de cuadro de mando remoto utilizando inkscape.

Aunque es posible utilizar cualquier editor de ficheros SVG de propósito general para editar un fichero de cuadro de mando remoto, hay razones poderosas para utilizar InkScape. Es una herramienta gratuita, es la utilizada por el equipo de Labcenter para crear los mandos suministrados con Proteus y Labcenter ha comprobado que funciona correctamente y sin problemas.

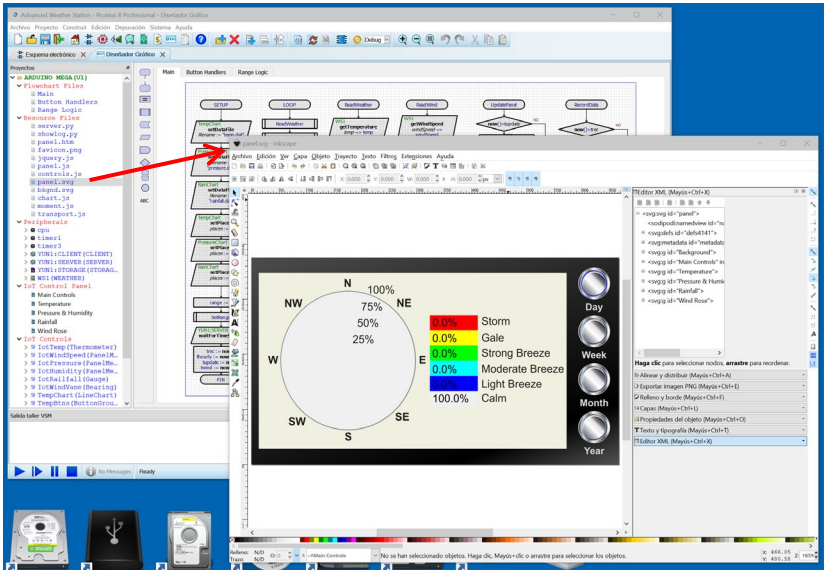
Hay varias situaciones en las que podemos tener que editar un fichero de cuadro de mando remoto usando InkScape por varias razones:

- Cuando es necesario añadir a nuestro cuadro de mando remoto gráficos complejos que usen gradientes de color que es una característica no soportada por el editor incluido en Proteus.
- Para recolocar algunos de los mandos de nuestro cuadro de forma más sofisticada con ayuda de herramientas de forzado o guías más potentes que las soportadas por el edito incluido en Proteus.
- Cuando deseamos hacer pequeños cambios en la apariencia de algunos de los mandos suministrados con Proteus.

En este tutorial vamos a suponer que ya tenemos hecha desde el sistema operativo la asociación de los ficheros con extensión svg con el editor de InkScape.



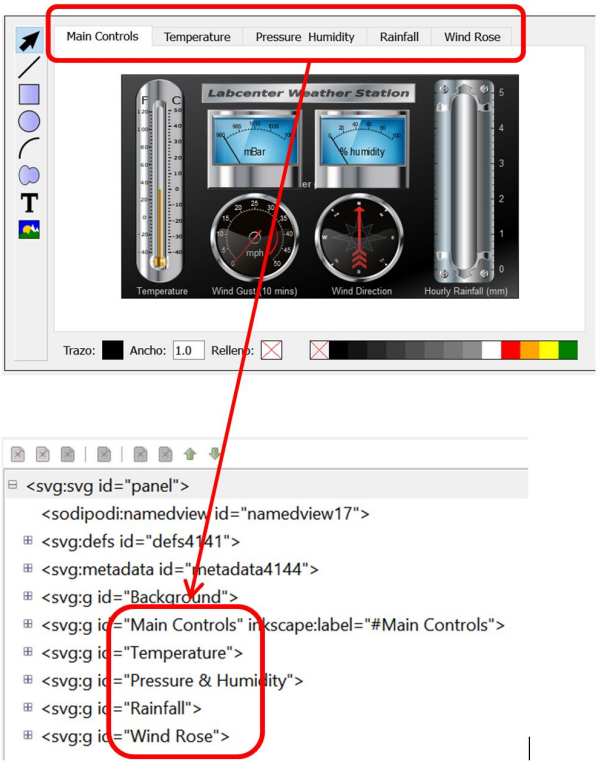
Una vez que la asociación está lista en nuestro ordenador, desde el árbol de proyecto, si buscamos el fichero 'panel.svg' y hacemos un doble click sobre él, automáticamente se abrirá la herramienta InkScape[5] y podemos editar el archivo.



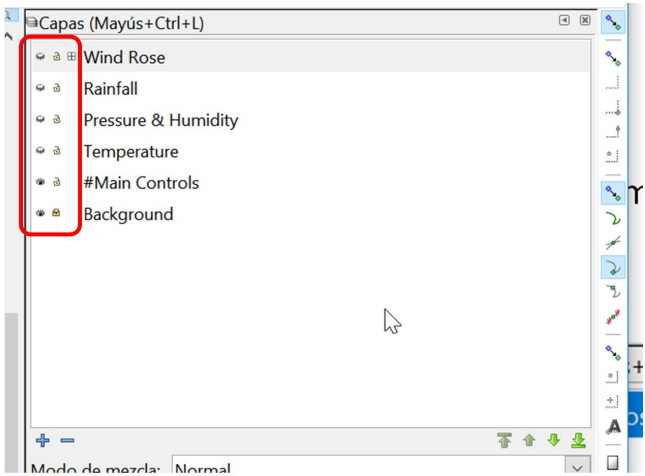
[5] Dependiendo de la capacidad de nuestro ordenador personal PC el tiempo que tarda en abrirse InkScape puede ser más o menos largo.

Antes de seguir adelante, es muy importante tener en cuenta que al editar el fichero usando InkScape es posible realizar tareas o cometer errores que destruyan e inutilicen nuestro cuadro de mando remoto diseñado desde el editor de Proteus. La cuestión clave para evitar problemas es no alterar la estructura de grupos de mandos contenida en el fichero svg. Dicho de forma abreviada, nunca tenemos que desagrupar los mandos, porque si lo hiciéramos, tendríamos que volver a recolocarlos de forma manual desde el editor de Proteus.

Al abrir por primera vez el fichero panel.svg desde InkScape, podemos comprobar que cada pestaña de nuestro cuadro de mando remoto está guardada como una capa independiente. Cada capa se corresponde con uno de los grupos de más alto nivel dentro de la estructura de un archivo svg.



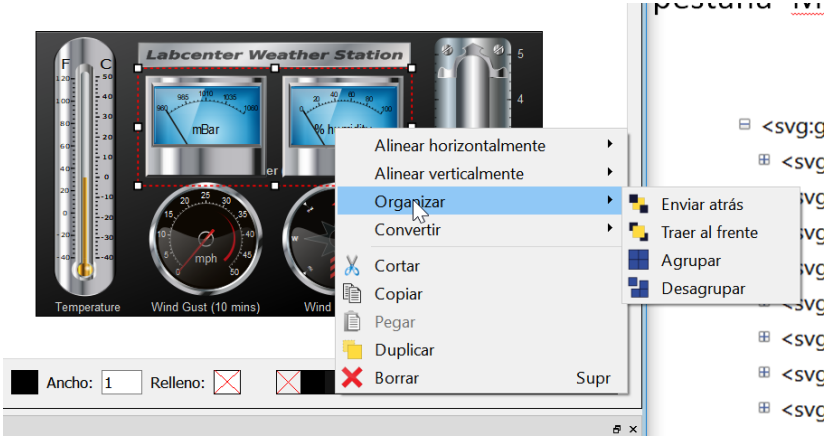
También debemos controlar la gestión de las capas que hace InkScape para asegurarnos de cuáles están visibles y cuales ocultas. Si la capa que contiene una determinada pestaña está como no visible InkScape no la presenta. Pero podemos fácilmente elegir qué capas se ven y cuáles no en cada momento utilizando la ventana de diálogo 'capas' de InkScape.



Cada mando de cada pestaña aparece en el segundo nivel de agrupación de la estructura del archivo svg. Por ejemplo, colgando de la rama de la pestaña 'Main Controls' aparecen todos los mandos situados en ella.

```
<svg:g id="Main Controls" inkscape:label="#Main Controls">
  <svg:g id="IotTH1">
  <svg:g id="IotSpeedo1">
  <svg:g id="IotMeter1">
  <svg:g id="IotMeter2">
  <svg:text id="text5859">
  <svg:text id="text3301">
  <svg:text id="text2771">
  <svg:text id="text5020">
  <svg:g id="g8291">
  <svg:g id="IotGauge1">
  <svg:g id="IotWindDial1">
```

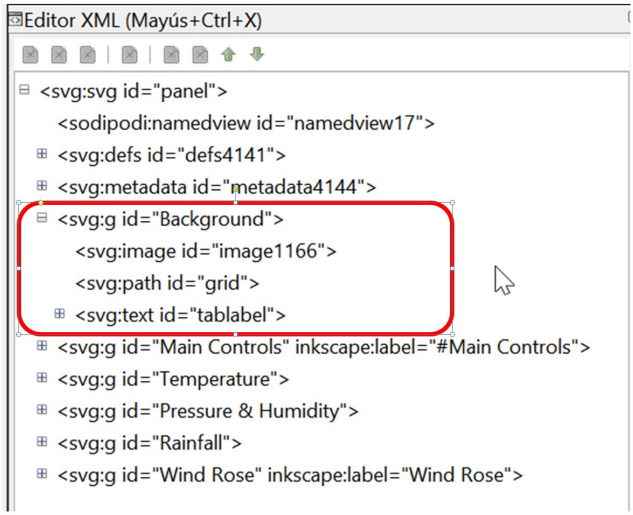
Si hubiéramos utilizado la herramienta agrupar desde el editor de Proteus, encontraríamos en el árbol de estructura del fichero svg, los mandos agrupados en una misma rama de la estructura del fichero.



Con sólo entender este funcionamiento básico de la forma en que se estructura un fichero svg y que seamos cuidadosos para no cambiarla, una vez que terminemos de editar el fichero en InkScape, podremos guardar nuestro trabajo y reutilizarlo perfectamente en Proteus con toda su funcionalidad intacta.

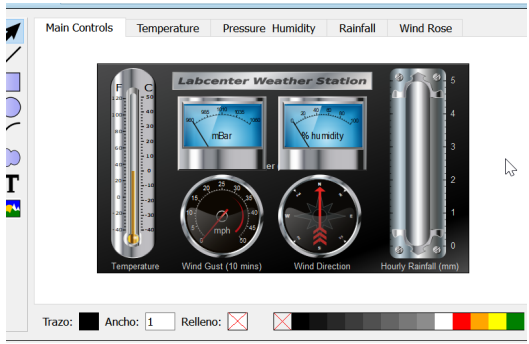
Si queremos añadir más gráficos o nuevos textos a nuestro cuadro de mando remoto podemos hacerlo usando las herramientas que pone a nuestra disposición InkScape.

Si los colocamos dentro de la capa que se utiliza para el fondo (la llamada background), estos gráficos y textos aparecerán en todas las pestañas. Si por el contrario, queremos que sólo aparezca en una determinada pestaña, tendremos que colocarlos dentro de la capa de esa pestaña en concreto (por ejemplo la capa ‘Main Controls’ de nuestro ejemplo).

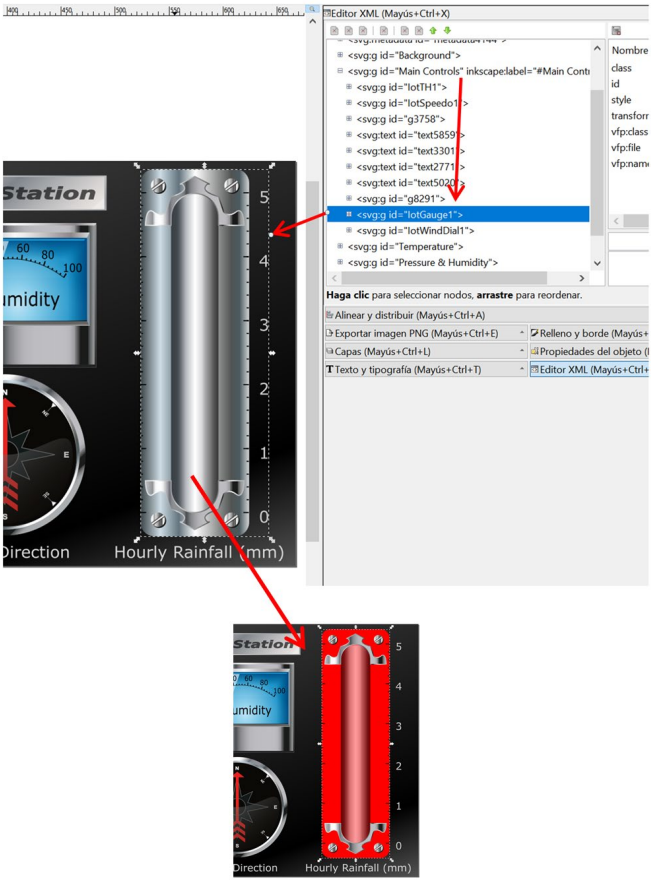


Si lo que deseamos hacer es cambiar el aspecto de un mando en concreto, podemos hacerlo fácilmente seleccionando dicho elemento en el árbol de estructura del fichero.

Veamos un ejemplo. En nuestro panel de mando, en la pestaña 'Main Controls' tenemos un mando que marca el nivel lluvias recogido.

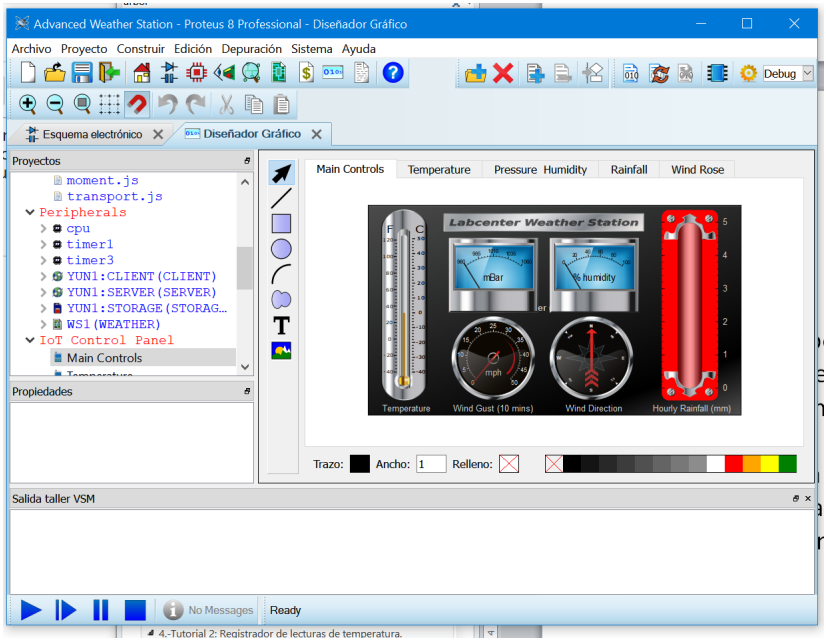


Para editarlo en InkScape, buscamos la rama de la pestaña (Main Controls), el mando deseado (IoTGauge1), pulsamos con el ratón sobre él y quedará seleccionado en la ventana de edición. A partir de este momento, podemos usar todas las herramientas que pone a nuestra disposición InkScape para cambiar su aspecto.



Podemos cambiar el trazo, el relleno, la fuente o cualquier otro atributo de todos elementos gráficos que componen nuestro cuadro de mando remoto.

En cuanto guardemos el fichero en InkScape, Proteus detectará automáticamente los cambios y actualizará el aspecto de nuestro cuadro de mando mostrando las modificaciones llevadas a cabo.



⚠ Es muy importante que no desagrupemos los diferentes mandos que componen nuestro cuadro de mando remoto, porque si lo hacemos el fichero svg modificado no trabajará al ser recargado en Proteus.

7.3.-Edición avanzada de un mando.

La edición de un mando es sustancialmente diferente a editar el cuadro de mando remoto. Para poder crear o editar un nuevo mando es muy importante entender perfectamente cuál es su organización interna y cómo funcionan.

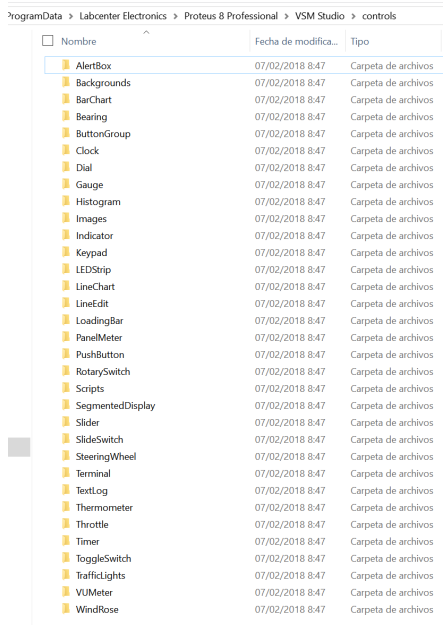
Las personas interesadas en crear sus propios mandos pueden enviar una solicitud por correo electrónico a support@labcenter.com y Labcenter les hará llegar toda la información técnica necesaria para poder hacerlo.

Este apartado está dedicado para aquellos que sólo necesiten realizar cambios menores en el aspecto de los mandos ya existentes en Proteus.

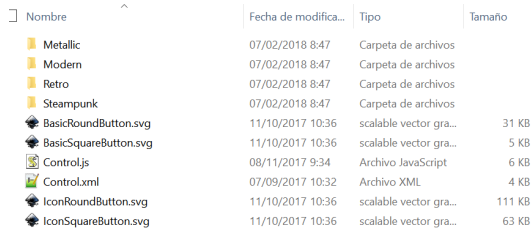
En este punto, vamos a asumir que tenemos abierto el editor del taller IoT y que ya hemos creado un cuadro de mando remoto en el que hemos colocado un pulsador. Hemos recorrido los distintos pulsadores disponibles en Proteus y ninguno de ellos muestra el aspecto que nos gustaría para nuestro cuadro de mando remoto. De hecho, vamos a suponer que queremos usar un pulsador que muestre una cara feliz que, obviamente, no viene suministrado con Proteus y, por lo tanto, tenemos que personalizar por nosotros mismos.

Desde el navegador de archivos de Windows iremos a la carpeta donde se encuentran los mandos que se usan en Proteus. La dirección de la carpeta, si hemos aceptado la carpeta que por defecto nos ofrece Proteus cuando hemos instalado el programa, es C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\VSM Studio\controls.

En esta carpeta encontraremos organizados en subcarpetas los diferentes mandos organizados por su función. En nuestro caso, como queremos cambiar el aspecto de un pulsador, nos tendremos que fijar en la carpeta PushButton.

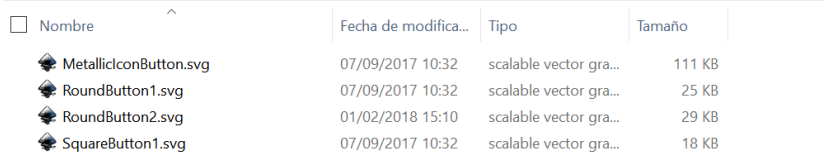


Dentro de esa carpeta encontraremos los mandos del tipo pulsador agrupados a su vez por temas.



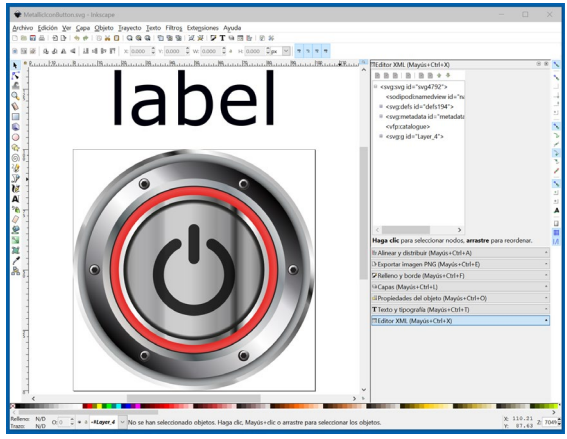
Nombre	Fecha de modifica...	Tipo	Tamaño
Metallic	07/02/2018 8:47	Carpeta de archivos	
Modern	07/02/2018 8:47	Carpeta de archivos	
Retro	07/02/2018 8:47	Carpeta de archivos	
Steampunk	07/02/2018 8:47	Carpeta de archivos	
BasicRoundButton.svg	11/10/2017 10:36	scalable vector gra...	31 KB
BasicSquareButton.svg	11/10/2017 10:36	scalable vector gra...	5 KB
Control.js	08/11/2017 9:34	Archivo JavaScript	6 KB
Control.xml	07/09/2017 10:32	Archivo XML	4 KB
IconRoundButton.svg	11/10/2017 10:36	scalable vector gra...	111 KB
IconSquareButton.svg	11/10/2017 10:36	scalable vector gra...	63 KB

Y dentro de la carpeta de cada tema, los diferentes mandos disponibles. Por ejemplo, dentro del tema ‘metallic’ encontramos varios ficheros de extensión svg cada uno de los cuales contiene la información de cada uno de los pulsadores disponibles en ese tema.



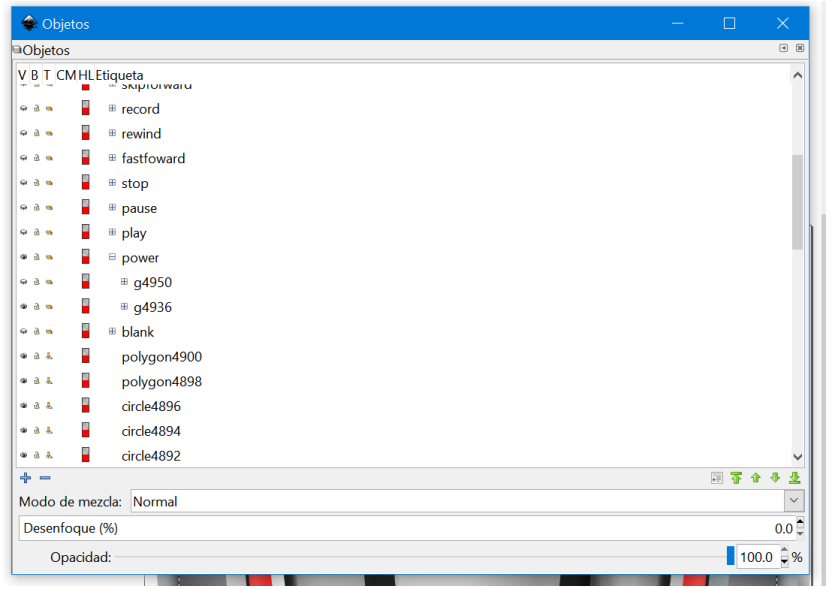
Nombre	Fecha de modifica...	Tipo	Tamaño
MetallicIconButton.svg	07/09/2017 10:32	scalable vector gra...	111 KB
RoundButton1.svg	07/09/2017 10:32	scalable vector gra...	25 KB
RoundButton2.svg	01/02/2018 15:10	scalable vector gra...	29 KB
SquareButton1.svg	07/09/2017 10:32	scalable vector gra...	18 KB

Vamos a abrir el fichero ‘MetallicIconButton.svg’. Si tenemos asociada la extensión de archivo ‘svg’ con el programa ‘InkScape’ al hacer doble click sobre él, se abrirá la herramienta para editarlo.



Abriremos la ventana que muestra el árbol con la estructura del archivo svg (menú objeto->objetos). Puede ser necesario expandir

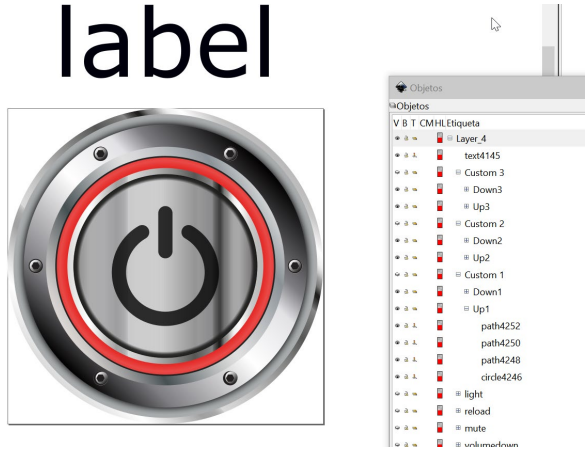
varias ramas para poder ver todos los grupos y capas que contiene el archivo.



⚠ Es muy importante que no desagrupemos los objetos que componen nuestro mando, porque si lo hacemos el fichero svg modificado no trabajará al ser recargado en Proteus.

ℹ Una buena técnica cuando llevemos a cabo el aspecto de un mando, es ocultar la(s) capa(s) que contiene los gráficos que queremos eliminar y crear la(s) capa(s) que necesitemos para mostrar en ella(s) la nueva iconografía. Así, en cualquier momento podemos volver a dejar el mando cómo estaba y comprobar siempre, mostrando y ocultando las capas necesarias, como evoluciona nuestro trabajo.

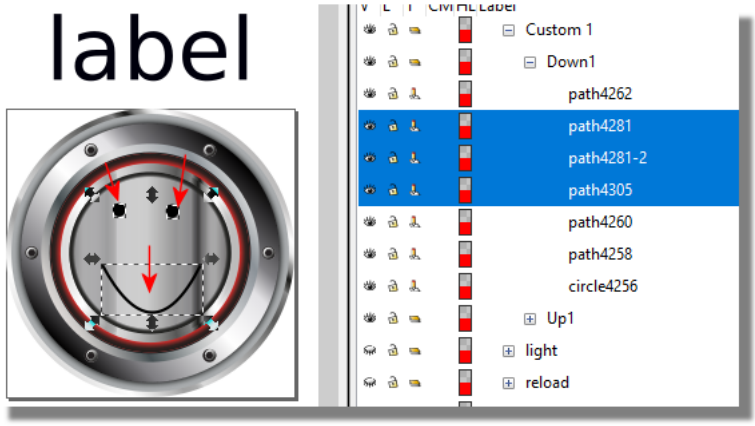
Buscaremos una de las capas 'Custom' (la 1, la2 o la 3) que son las que deseamos modificar dentro del árbol de la estructura del archivo. Dentro de cada una de ellas encontraremos dos grupos más que se corresponden con las imágenes que se muestran en los diferentes estados posibles del pulsador (actuado o no actuado).



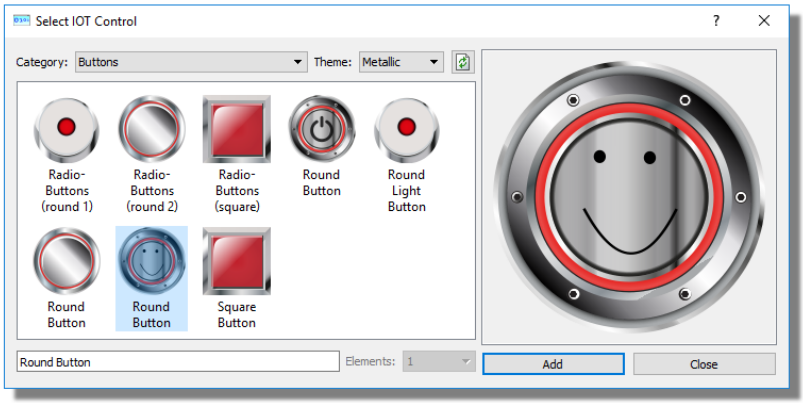
Volvemos a expandir uno cualquiera de ellos, por ejemplo el Down1, para ver su contenido. Vamos a añadir en el gráfico un rectángulo, una línea, un arco o cualquier otro elemento de dibujo que deseemos. Tantos como necesitemos para conseguir el aspecto que deseamos para nuestro nuevo pulsador.

Una vez creados podemos copiarlos en este grupo o en cualquier otro y moverlos hasta colocarlos dentro del lugar adecuado en la estructura del archivo. Si los colocamos en las primeras posiciones dentro del grupo, veremos los nuevos objetos gráficos añadidos cuando el pulsador sea actuado. Podemos añadir más de un objeto, pero sin agruparlos.

En la siguiente imagen, podemos ver nuestro trabajo con flechas que indican los gráficos añadidos y esos mismos elementos seleccionados en el árbol de estructura del archivo en la derecha.



Guardaremos nuestro trabajo como un nuevo fichero svg. Es muy importante que guardemos nuestros nuevos mandos dentro de la carpeta que almacena todos los ficheros svg siguiendo las convecciones que explicamos antes para que aparezcan de forma coherente cuando busquemos un nuevo mando al usar el taller IoT: carpetas organizadas por funcionalidad del mando y dentro de ella carpetas organizadas por el tema.



⚠ Cualquier cambio que llevemos a cambio en los mandos que se encuentran dentro de las carpetas creadas durante la instalación de Proteus que contienen los mandos serán sobre-escritas en cualquier nueva actualización de Proteus que instalemos en nuestro ordenador. La manera de

preservar nuestros diseños es utilizar carpetas personalizadas donde ubiquemos nuestros diseños que serán respetados en el proceso de actualización de la versión de Proteus.

8.-Mandos básicos.

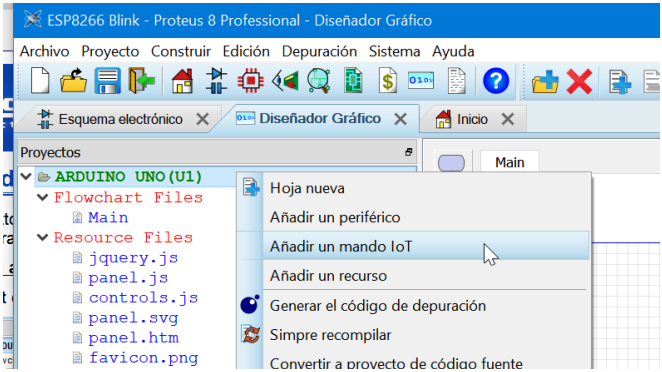
8.1.-Pulsadores.

8.1.1.-Descripción.

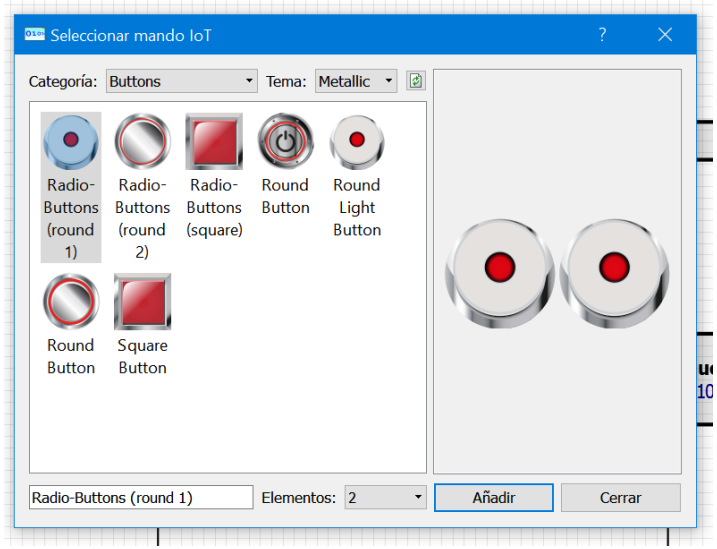
El pulsador es probablemente el mando que más tengamos que utilizar en nuestros proyectos y, sin duda, uno de los más versátiles de todos los disponibles en la librería de mandos del taller IoT. En este apartado vamos a ver los diferentes tipos de pulsador y las opciones de configuración que están disponibles.

En todos los casos, independientemente del tipo y de la configuración que elijamos, el procedimiento para añadir un pulsador en nuestro proyecto es el mismo.

Primero abriremos el menú contextual de nuestro equipo Arduino Yun en el árbol de proyectos con el botón derecho y seleccionaremos la opción 'Añadir un mando IoT'



A continuación comprobaremos que tenemos seleccionados la categoría de los pulsadores y el tema que deseamos utilizar en los dos desplegables situados en la parte superior de la ventana.



El tema determina los diferentes estilos de pulsador disponibles pero no su funcionalidad. Sólo fija su aspecto y no su comportamiento. De esta forma es posible diseñar un cuadro de mando remosto con un aspecto moderno, clásico, retro, etc. de acuerdo con nuestras preferencias.

Por último, elegiremos entre los tipos de pulsadores disponibles. En la mayoría de los casos podemos elegir entre los pulsadores de selección (radio buttons), pulsadores convencionales (push buttons) y pulsadores grabados (push buttons with icons).

Los pulsadores de selección son un grupo de pulsadores que son excluyentes entre ellos (sólo podemos tener uno seleccionado de cada vez). Son interesantes de usar cuando queremos mostrar todas las opciones disponibles destacando la seleccionada actualmente. Cuando elegimos los pulsadores de selección debemos indicar el número de pulsadores que contendrá el conjunto en la casilla situada en la parte inferior de la ventana de diálogo y a continuación usar el botón añadir.

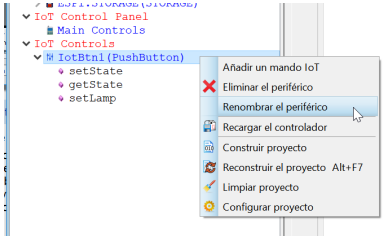
Los pulsadores grabados son pulsadores normales en los que podemos incluir un icono seleccionable para ayudar a identificar su función. Por ejemplo, encendido, arranca, parada, etc. Si queremos

utilizar un pulsador de este tipo sólo tenemos que seleccionarlo y usar el botón añadir.

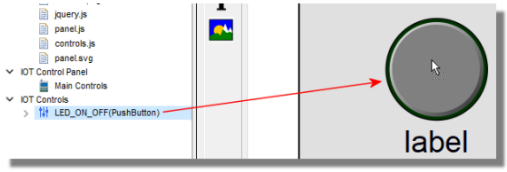
Llamamos pulsadores estándar a todo el resto de pulsadores que no han quedado definidos en las dos categorías anteriores y que pueden estar disponibles dentro de cada tema con diferentes formas (circulares, cuadrados, etc.). Si queremos utilizar un pulsador de este tipo sólo tenemos que seleccionarlo y usar el botón añadir.

8.1.2.-Configuración.

El pulsador puede ser configurado en tiempo de diseño antes de que esté colocado en nuestro cuadro de mando remoto y después de que lo hayamos posicionado. Lo más aconsejable es que antes de llevarle a su ubicación, le asignemos un nombre que nos ayude a identificar su función. Para hacerlo, en el árbol de la estructura del proyecto, en el apartado 'IoT Controls' buscaremos nuestro pulsador y en el menú contextual seleccionaremos la opción 'Renombrar el periférico'

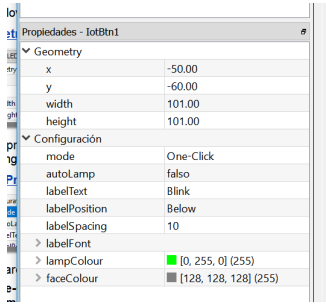


Una vez que lo hayamos renombrado podemos colocarlo sobre el panel del cuadro en la posición que deseemos utilizando la técnica 'coger y arrastar'.



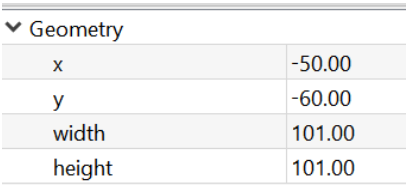
Una vez que tengamos situado nuestro pulsador sobre el panel del cuadro haciendo un click sobre él, podemos seleccionarlo. Cuando lo

tengamos seleccionado se muestra el panel de propiedades en la parte izquierda inferior.



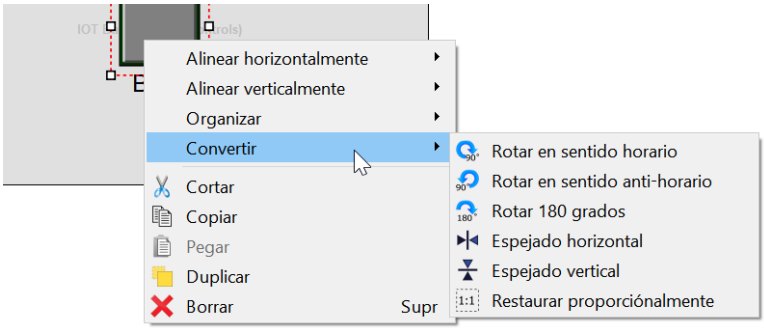
8.1.2.1.-El grupo de propiedades 'Geometry'.

En prácticamente todos los mandos, encontraremos disponibles las cuatro funcionas básicas x, y, width y height, dentro de un apartado llamado 'Geometry'.



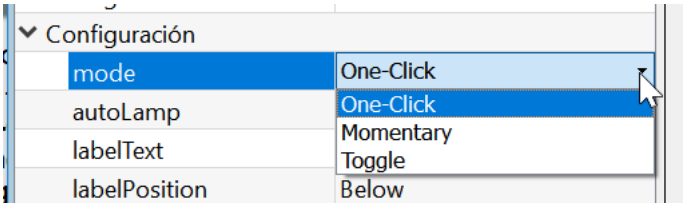
Estas propiedades sirven para colocar y alinear el pulsador. Por ejemplo, podemos elegir varios pulsadores con el mismo valor en la variable 'x' para que estén todos ellos alineados verticalmente.

'Width' y 'height' nos permiten seleccionar el ancho y alto respectivamente de nuestro pulsador. Podemos usarlos para asegurarnos que varios pulsadores tienen exactamente el mismo tamaño. Si seleccionamos sólo uno de ellos, nos puede resultar muy útil la opción del menú contextual del pulsador 'Restaurar proporcionalmente' que ajusta el tamaño de forma automática para mantener las proporciones del modelo original.



8.1.2.2.-La propiedad 'Mode'

La propiedad 'mode' nos permite seleccionar entre tres opciones: pulsador (one-click), instantáneo (momentary) y conmutador (toggle).



La opción pulsador (one-click). Cuando seleccionamos este modo, nuestro pulsador tiene un único estado: pulsado. El controlador del evento es invocado después que la opción de pulsar ha concluido. Si mantenemos el pulsador accionado no pasa nada hasta que lo liberamos dejando de accionarlo.

La opción instantáneo (momentary). Cuando hemos elegido este modo, nuestro pulsador también tiene un único estado: pulsado. El controlador del evento es invocado en cuanto actuamos sobre el pulsador y, de nuevo, cuando lo liberamos. Si mantenemos el pulsador accionado ocurre algo nada más accionarlo y, aunque sigamos presionándolo, podremos observar que vuelve a su estado original de liberado de forma automática.

La opción conmutador (toggle). Cuando seleccionamos este modo, nuestro pulsador tiene dos estados: pulsado y liberado. Cada actuación sobre el pulsador provoca el cambio de estado. El controlador del evento es invocado cada vez que se actúa sobre el pulsador y para saber en cuál de los dos estados se encuentra el

pulsador utilizaremos el método gestState() que nos devuelve un 1 o un 0.

Es conveniente seleccionar el modo del pulsador en función del uso que vayamos a darle. Veamos unos ejemplos. Para un pulsador de reinicio (reset) la mejor opción es el modo pulsador, porque cada vez que el usuario actúe sobre el botón deseamos reiniciar la aplicación. Para un pulsador utilizado para subir el volumen, la mejor opción será el modo instantáneo que responde nada más pulsarlo. Por último, si queremos usar el pulsador para que la primera vez que actuemos sobre él se encienda un led y con la segunda pulsación se apague (es decir, para cambiar el estado del led con cada nueva pulsación que efectuemos), nuestra opción será el modo conmutador.

8.1.2.3.-La propiedad 'autoLamp'.

Todos los pulsadores tienen un anillo luminoso a lo largo de su contorno que puede encenderse cuando el pulsador se encuentra actuado. Con ello se logra facilitar al usuario una información visual útil y vistosa. El comportamiento del anillo luminoso se gestiona utilizando la propiedad 'autolamp'. El comportamiento dependerá del valor de la propiedad 'mode' de pulsador que hayamos seleccionado.

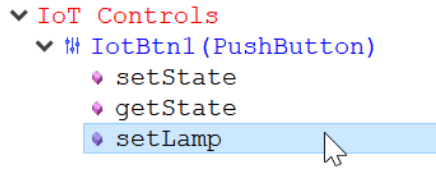
En los pulsadores modo pulsador (one-click) y modo instantáneo (momentary), el anillo se iluminará cuando el pulsador esté accionado y se apagará al liberarlo.

En los pulsadores modo conmutador (toggle) el anillos se iluminará con la pulsador se accione y permanecerá iluminado hasta que con una nueva pulsación lo desactivemos.

De lo dicho se deduce que el uso de la propiedad 'autolamp' tiene más sentido cuando se utiliza un pulsador tipo conmutador porque el anillo luminoso sirve para informar al usuario de forma intuitiva del estado actual del pulsador (actuado o liberado). De todas formas, también puede ser una información útil en los otros dos casos porque ayuda al usuario a confirmar de que realmente ha actuado sobre el pulsador cuando se usa el panel de control.

Es importante tener en cuenta que la propiedad 'autolamp' por defecto toma el valor de apagado y, por lo tanto, no se enciende al anillo luminoso al actuar sobre el pulsador. Aunque puede parecer

que este comportamiento es extraño, la razón de ello es que el anillo luminoso se puede encender o apagar también mediante programación usando el método setLamp().

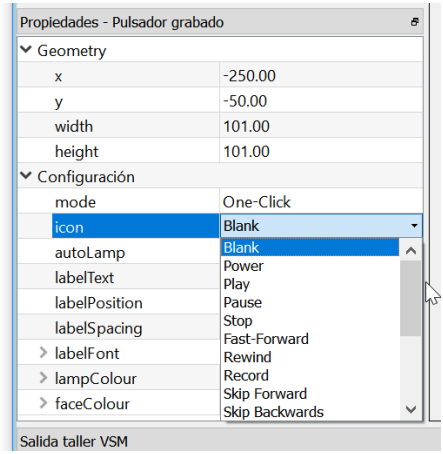


Utilizar el método setLamp() es más aconsejable desde el punto de vista del programador, aunque nos lleve más tiempo implementar su gestión. La principal razón de usarlo es que de esta manera garantizamos que siempre el estado del anillo luminoso se corresponde exactamente con el estado que el controlador está valorando en cada momento para ese pulsador.

Así, por ejemplo, por un error en el código o en las comunicaciones entre el panel de control y el autómata, el controlador puede estar trabajando como si el pulsador se encontrara liberado (porque no se ha recibido correctamente la orden desde el cuadro de mando remoto) y, sin embargo, mostrarse como actuado en el panel de control remoto.

8.1.2.4.-La propiedad 'icon'.

Esta propiedad solo está disponible para los pulsadores grabados (los que pueden llevar un icono). En este caso, podemos utilizar la propiedad 'icon' para establecer el icono que será utilizado en el pulsador. Podemos seleccionar entre la lista de iconos por defecto o, con un poco más de trabajo, añadir un icono realizado por nosotros a medida.



Para añadir un icono propio necesitamos editar un fichero svg. No es un proceso sencillo, pero se explicó la manera de hacerlo en el apartado 7 diseño avanzado de cuadros de mando remoto.

8.1.2.5.-El grupo de propiedades 'label'.

El uso de estas propiedades son tan obvias que su propio nombre auto define perfectamente su significado y utilización.

labelText	label
labelPosition	Below
labelSpacing	10
▼ labelFont	
family	sans-serif
size	30
weight	normal
style	normal
decoration	none
▶ colour	[0, 0, 12] (255)

Nos permiten asignar el texto de la etiqueta, la posición que ocupará la etiqueta con respecto al pulsador, el espacio de separación entre la etiqueta el pulsador y las características de las fuente tipográficas utilizadas para la etiqueta.

8.1.2.6.-La propiedad 'lampColour'.

Esta propiedad nos permite especificar el color del anillo luminoso del pulsador.

▶ lampColour	[155, 246, 223] (255)

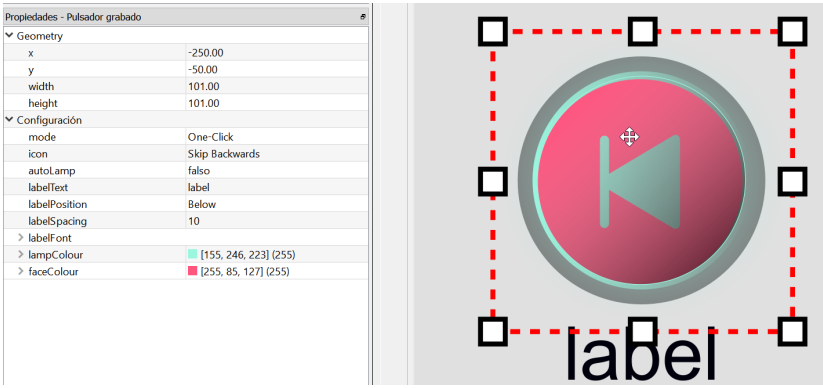
8.1.2.7.-La propiedad 'faceColour'.

Esta propiedad nos permite especificar el color del fondo del pulsador.

▶ faceColour	[255, 85, 127] (255)

La siguiente imagen muestra un pulsador con una selección de colores muy contrastada para que sea fácil identificar qué partes del

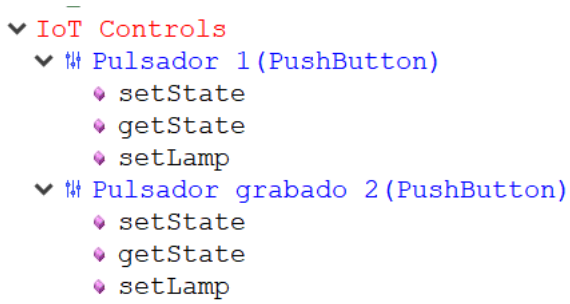
pulsador controlamos con cada una de las dos propiedades disponibles para gestionar el color (lampColour y faceColour).



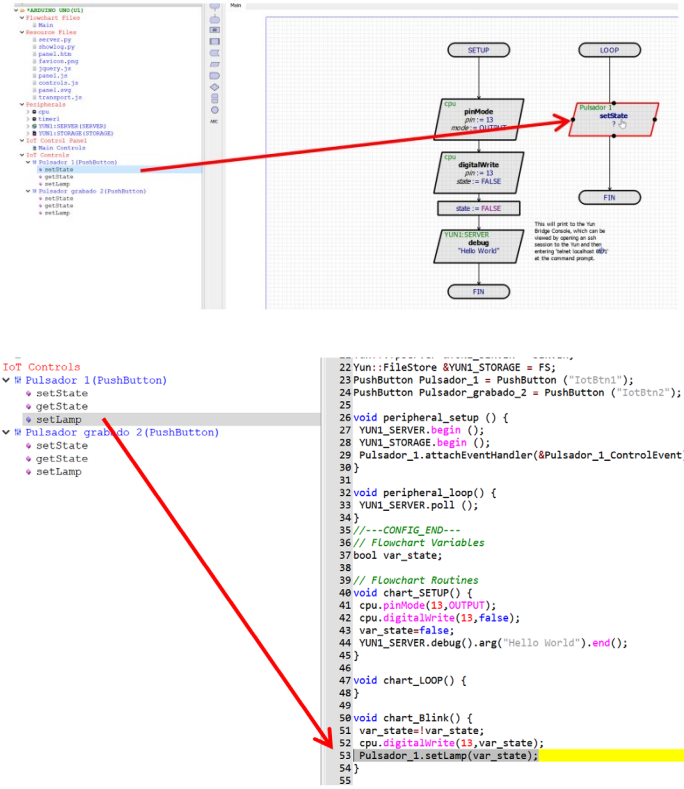
8.1.3.-Métodos de control.

Después de configurar el pulsador es necesario poder controlarlo desde nuestro código de programación. Proteus pone a nuestra disposición tres métodos, salvo en el caso de los pulsadores de selección (radio buttons) que sólo disponen de dos.

Los métodos están disponibles en el árbol de estructura de proyecto colgando de cada rama de pulsador que hayamos colocado en nuestro cuadro de mando remoto.



Podemos usar estos métodos en nuestro programa usando la técnica ‘coger y arrastar’ desde el árbol de estructura del proyecto hasta nuestro flujograma (en cuyo caso se creará un bloque de programación) o hasta nuestro código C (en cuyo caso se creará la estructura básica de una función).



8.1.3.1.-El método setState().

Este método sólo está disponible en el pulsador tipo conmutador (toggle) porque es el único que tiene dos estados. Se utiliza para definir el estado por defecto del pulsador (cerrado o abierto) cuando el usuario no ha llevado a cabo ninguna actuación sobre él. Por ejemplo, podemos seleccionar un pulsador del tipo conmutador que sirva para seleccionar si se graba o no en la bitácora el histórico de recogida de datos. En este caso, podemos seleccionar con esta función si por defecto se procede a la grabación de los datos en la bitácora o, si por el contrario, no se arranca la aplicación grabando los datos.

8.1.3.2.-El método getState().

Este método también está sólo disponible en el pulsador del tipo conmutador (toggle). Devuelve el estado actual del pulsador (actuado o liberado) como un valor de tipo booleano.

8.1.3.3.-El método setLamp().

Este método sirve para encender o apagar el halo luminoso del pulsador mediante código. De esta manera se puede indicar al usuario de forma intuitiva el estado actual del pulsador desde el controlador Arduino.

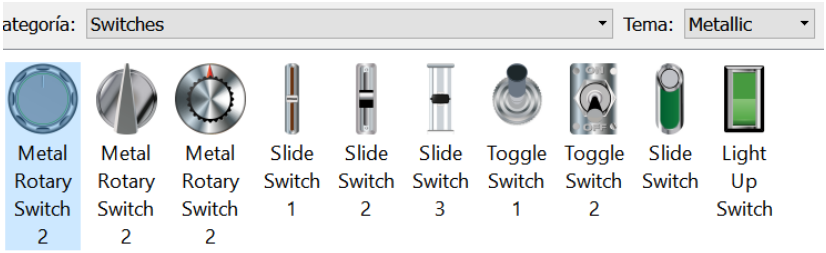
Si hemos configurado la propiedad 'autoLamp' a verdadero desde la página de propiedades del pulsador, no necesitamos utilizar este método, porque será el propio cuadro de mando remoto quien gestione el halo luminoso.

En el apartado 8.1.2.3 se muestra con detalle el análisis de las posibilidad de usar la propiedad autoLamp en conjunción con el método setLamp().

8.2.-Interruptores.

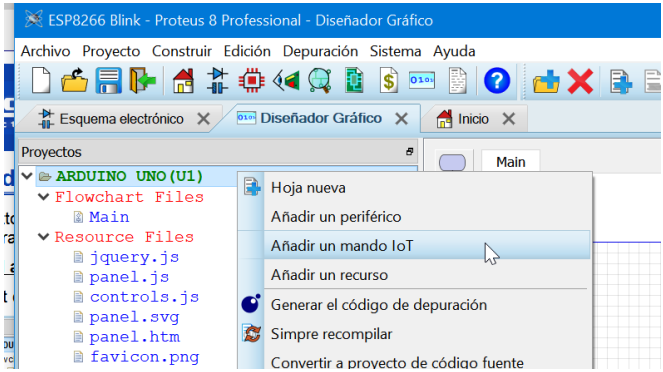
8.2.1.-Descripción.

En función del tema que hayamos seleccionado, tenemos hasta tres tipos posibles de interruptores: interruptores simples de dos estados (toggle switch), un interruptor de actuador deslizante de múltiples estados (slide switch) y un interruptor rotatorio (rotary switch).

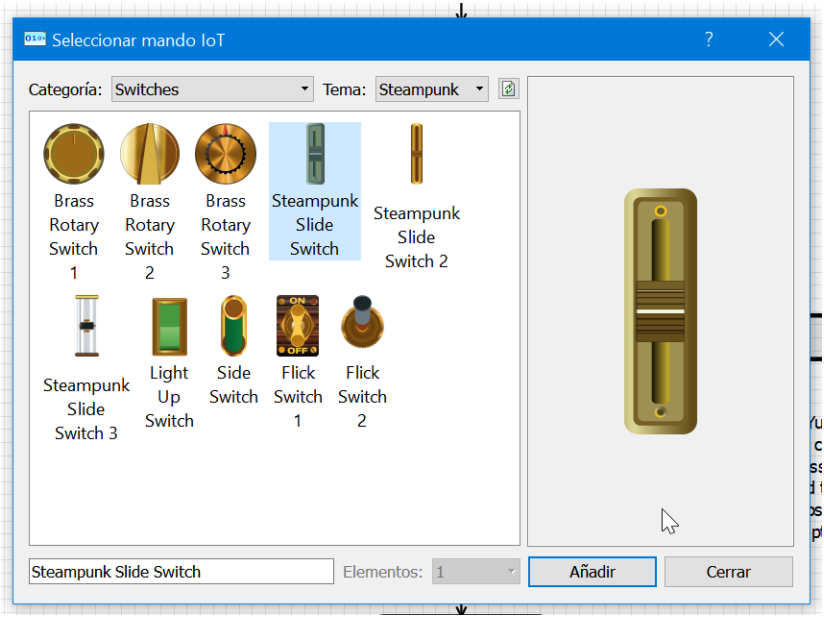


En todos los casos, independientemente del tipo y de la configuración que elijamos, el procedimiento para añadir un interruptor en nuestro proyecto es el mismo.

Primero abriremos el menú contextual de nuestro equipo Arduino Yun en el árbol de proyectos con el botón derecho y seleccionaremos la opción 'Añadir un mando IoT'



A continuación comprobaremos que tenemos seleccionados la categoría de los interruptores y el tema que deseamos utilizar en los dos desplegables situados en la parte superior de la ventana.

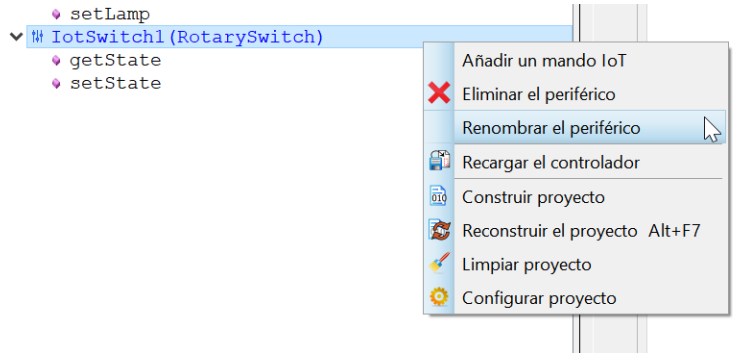


Por último, elegiremos entre los tipos de pulsadores disponibles y usaremos el botón ‘añadir’.

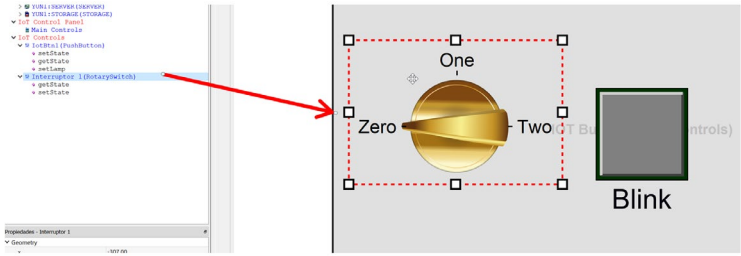
8.2.2.-Configuración

El interruptor puede ser configurado en tiempo de diseño antes de que esté colocado en nuestro cuadro de mando remoto y después de que lo hayamos posicionado. Lo más aconsejable es que antes de llevarle a su ubicación, le asignemos un nombre que nos ayude a identificar su función.

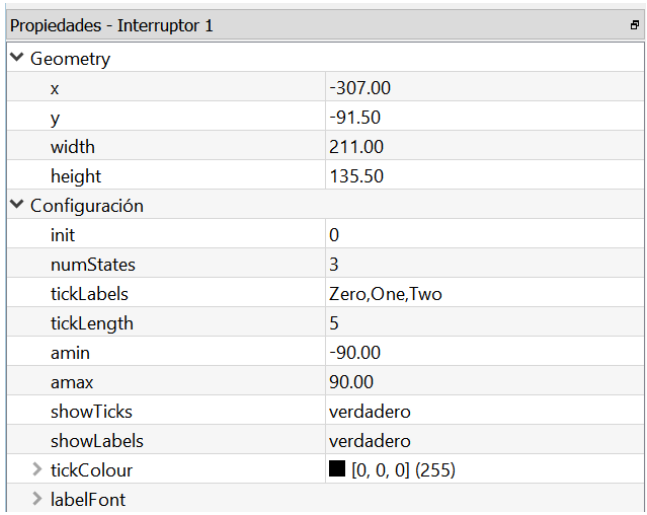
Para hacerlo, en el árbol de la estructura del proyecto, en el apartado ‘IoT Controls’ buscaremos nuestro interruptor y en el menú contextual seleccionaremos la opción ‘Renombrar el periférico’



Una vez que lo hayamos renombrado podemos colocarlo sobre el panel del cuadro en la posición que deseemos utilizando la técnica ‘coger y arrastar’.

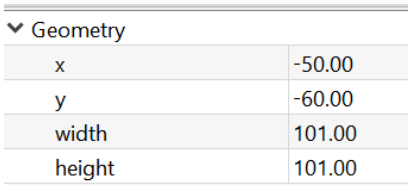


Una vez que tengamos situado nuestro interruptor sobre el panel del cuadro haciendo un click sobre él, podemos seleccionarlo. Cuando lo tengamos seleccionado se muestra el panel de propiedades en la parte izquierda inferior.



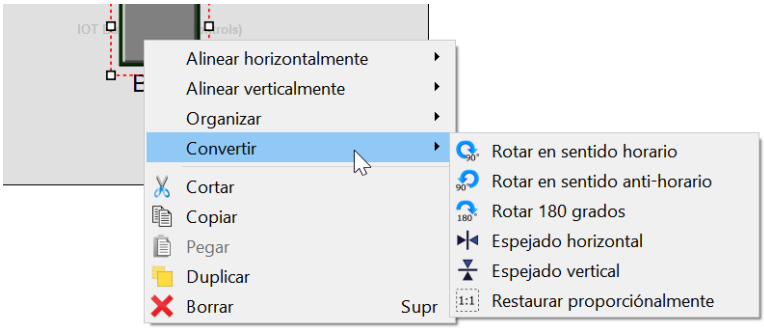
8.2.2.1.-El grupo de propiedades 'Geometry'.

En todos los interruptores, encontramos disponibles las cuatro funcionas básicas x, y, width y height, dentro del apartado llamado 'Geometry'.



Estas propiedades sirven para colocar y alinear el interruptor. Por ejemplo, podemos elegir varios pulsadores con el mismo valor en la variable 'x' para que estén todos ellos alineados verticalmente.

'Width' y 'height' nos permiten seleccionar el ancho y alto respectivamente de nuestro interruptor. Podemos usarlos para asegurarnos que varios interruptores tienen exactamente el mismo tamaño. Si seleccionamos sólo uno de ellos, nos puede resultar muy útil la opción del menú contextual del interruptor 'Restaurar proporcionalmente' que ajusta el tamaño de forma automática para mantener las proporciones del modelo original.



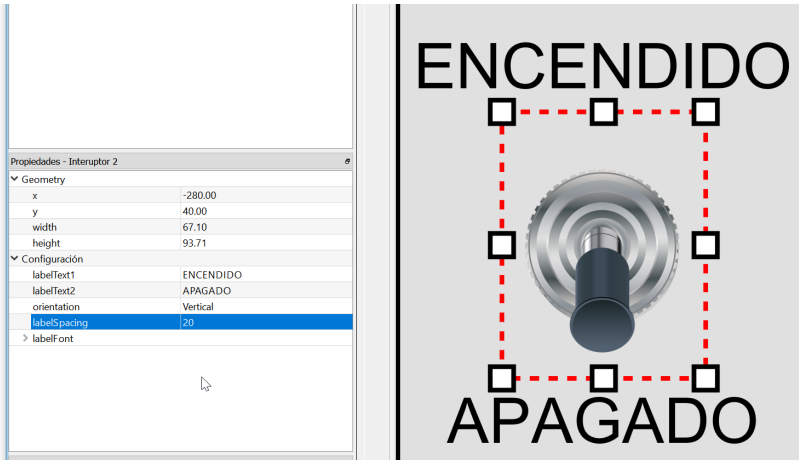
8.2.2.2.-El grupo de propiedades 'label'.

El uso de estas propiedades permite gobernar las etiquetas que ayudan a identificar la función de cada una de las posiciones del interruptor.

Con la propiedad labelFont podemos elegir el formato de la fuente utilizado para las etiquetas.

labelFont	
family	sans-serif
size	20
weight	normal
style	normal
decoration	none
colour	[0, 0, 0] (255)

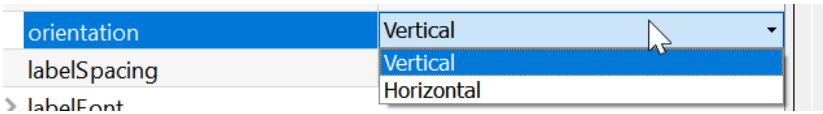
Las propiedades labelText1 y labelText2 están disponibles en el caso de los interruptores simples de dos posiciones y sirven para rotular la función de cada posición.



La propiedad ‘labelSpacing’ nos permite fijar la distancia entre el pulsador y las etiquetas.

labelSpacing	20

Con la propiedad ‘orientation’ podemos elegir la orientación de la etiqueta.

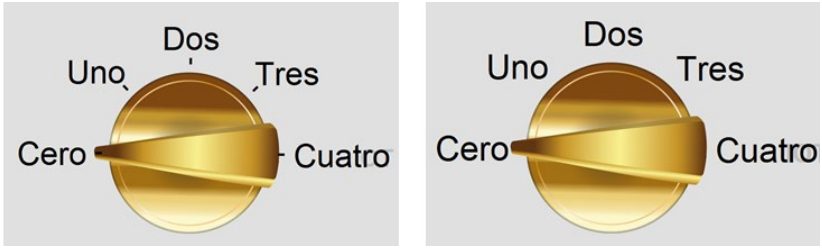


En los interruptores que pueden tener múltiples estados (interruptor de actuador deslizante e interruptor rotatorio) tenemos propiedades adicionales para el control de las etiquetas.

La propiedad ‘showLabels’ se usa para indicar si se muestran o no las etiquetas de las funciones. Y la propiedad ‘showTicks’ indica si se muestra o no una pequeña marca junto a cada posición del interruptor para ayudar a identificarla.

showTicks	verdadero
showLabels	verdadero

La siguiente imagen muestra un interruptor con las marcas y sin ellas para comprobar la diferencia.



La propiedad ‘tickLenght’ fija la longitud de la marca.

tickLength	4

La propiedad ‘tickLabels’ nos permite poner nombres a las diferentes etiquetas de cada uno de los estados posibles del interruptor. Los diferentes rótulos se separan mediante el uso de comas.

tickLabels	Cero, Uno, Dos, Tres, Cuatro

La propiedad ‘numStates’ fija el número de estados posibles que puede adoptar el interruptor.

numStates	5

8.2.3.-métodos de control.

Los métodos de control nos permiten acceder y controlar cada interruptor en tiempo de ejecución. Proteus pone a nuestra disposición dos métodos: getState() y setState().

8.2.3.1.-El método getState().

Devuelve el valor de la posición actual del interruptor como un valor entero a contar desde cero. Por ejemplo, si el interruptor sólo tiene dos estados, devolverá 0 ó 1. Si es un conmutador de cinco posiciones, tendrá cinco estados, 0, 1, 2, 3 y 4.

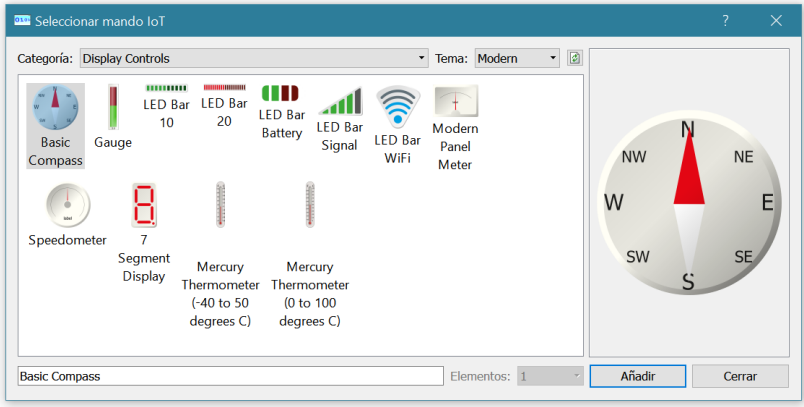
8.2.3.2.-El método setState().

Permite cambiar desde el código el estado de un interruptor y fuerza el aspecto del mismo para indicar la nueva posición seleccionada.

8.3.-Indicadores analógicos.

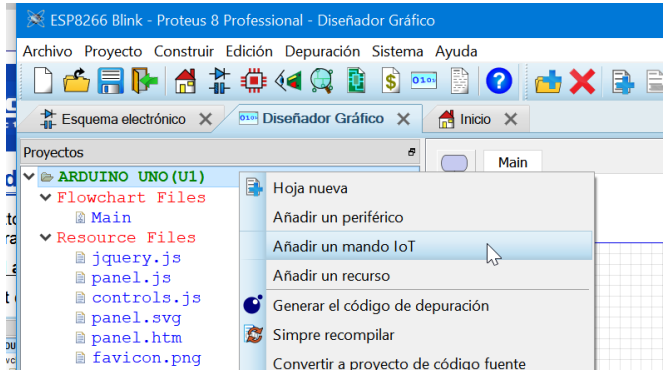
8.3.1.-Descripción.

En Proteus disponemos de seis tipos básicos de controles de lectura analógica: controles de aguja (velocímetro, panel de medida), display de 7 segmentos, termómetros de mercurio, barras de led (para valores numéricos o porcentajes), barras Gauge.

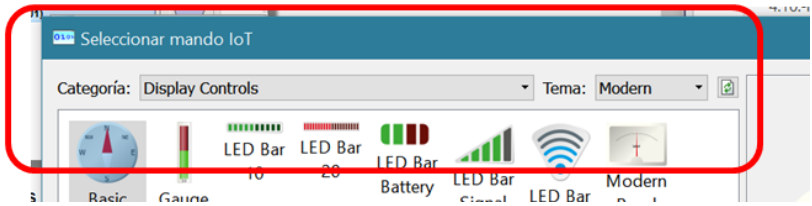


En todos los casos, independientemente del tipo y del tema que elijamos, el procedimiento para añadir un interruptor en nuestro proyecto es el mismo.

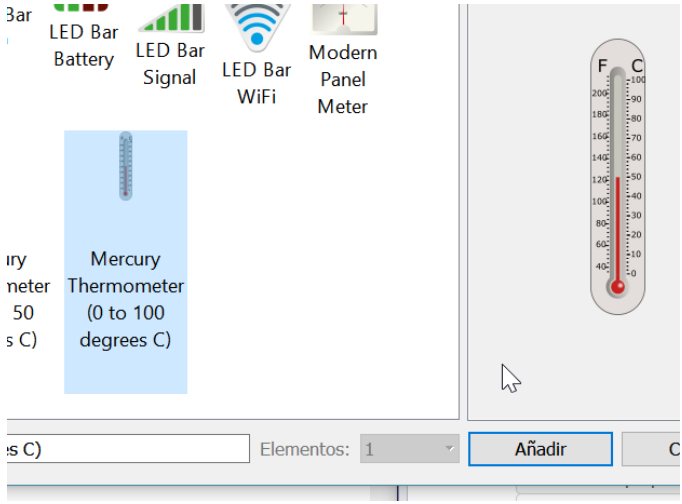
Primero abriremos el menú contextual de nuestro equipo Arduino Yun en el árbol de proyectos con el botón derecho y seleccionaremos la opción 'Añadir un mando IoT'



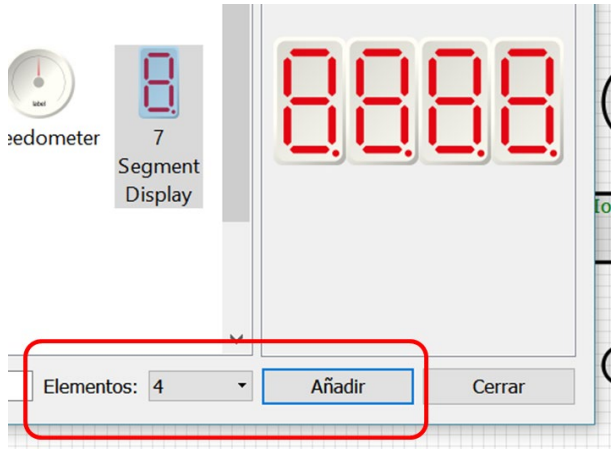
A continuación comprobaremos que tenemos seleccionados la categoría de los indicadores de lecturas analógicas y el tema que deseamos utilizar en los dos desplegables situados en la parte superior de la ventana.



Si estamos incorporando un control del tipo controles de aguja o termómetro sólo tenemos que seleccionarlo y presionar el botón ‘Añadir’.

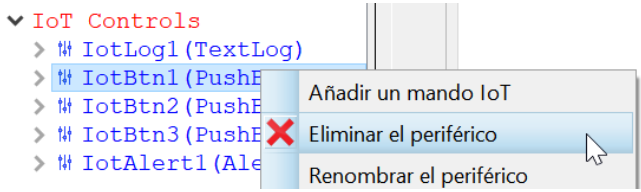


Si estamos incorporando un control del tipo display de 7 segmentos tenemos que elegir el número de elementos (dígitos) que formará el control y presionar el botón ‘añadir’



8.3.2.-Configuración

La primera tarea que tenemos que llevar a cabo una vez que hayamos seleccionado el control es asignarle un nombre que nos facilite la identificación de la tarea para la que lo vamos a usar. Lo podemos llevar a cabo utilizando el menú contextual asociado al control.



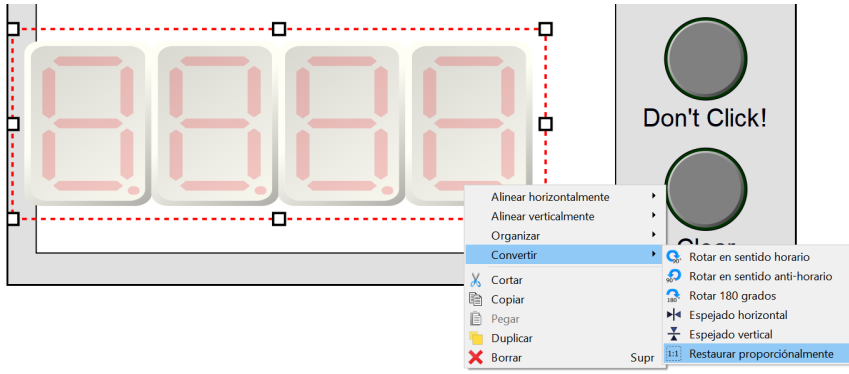
8.3.3.-métodos de control.

8.3.3.1.-Las propiedades comunes

Encontraremos varias propiedades comunes a todos los controles independientemente de su tipo. En primer lugar las que se encuentran en el grupo 'Geometry' que nos permiten posicionar y elegir el tamaño de nuestro control.

Geometry	
x	-50.00
y	-60.00
width	101.00
height	101.00

A menudo resulta más sencillo redimensionar el control utilizando el ratón para estirar los 'manejadores' del gráfico aunque se deforme. Al tener nuestro tamaño aproximado podemos seleccionar desde el menú contextual del control la opción 'Restaurar proporcionalmente' dentro del grupo 'Convertir' para volver a dejar el control proporcionado en altura y anchura.



Podemos cambiar la etiqueta asociada al control seleccionando las propiedades que se encuentran bajo el apartado 'Configuración'. Algunos controles disponen de la propiedad 'textVisible' para ocultar la etiqueta.

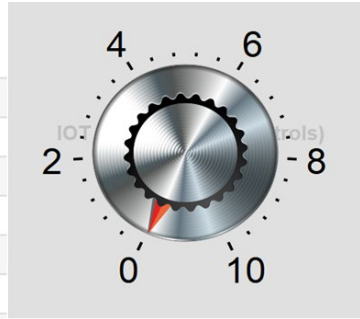
Configuración	
units	Percentage
init	0.00
LEDColourGroup1	[0, 255, 0] (255)
LEDGroupArea1	30.00
LEDColourGroup2	[255, 255, 0] (255)
LEDGroupArea2	40.00
LEDColourGroup3	[255, 0, 0] (255)
LEDoffColour	[96, 96, 96] (255)
textFont	
textVisible	verdadero

8.3.3.2.-Propiedades de los controles del tipo mando rotatorio.

En función del tema seleccionado podemos escoger entre diferentes tipos de mando rotatorio aunque las opciones de configuración son iguales para todos ellos.

Podemos seleccionar el rango de los valores numéricos que aparecen utilizando las propiedades ‘min’ y ‘max’. A continuación podemos elegir el número de divisiones entre esos rangos con la propiedad ‘numDivs’. Por ejemplo, si elegimos un rango entre 0 (min=0) y 10 (max=10) con un número de divisiones igual a 5 (numDivs=5) obtendremos incrementos de dos unidades de una posición a otra.

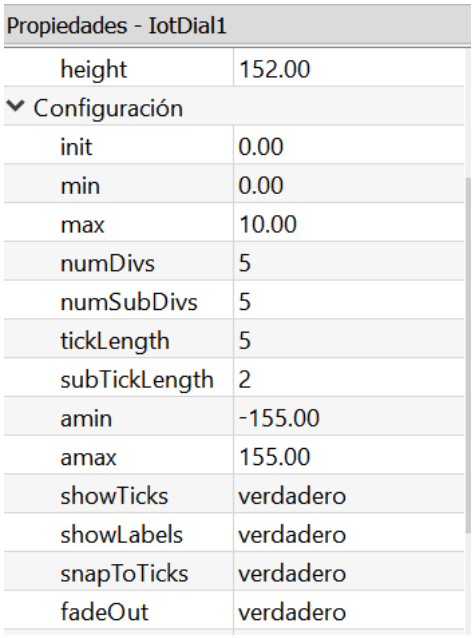
Configuración	
init	0.00
min	0.00
max	10.00
numDivs	5
numSubDivs	5



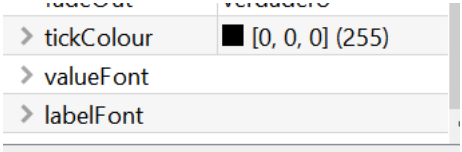
El otro aspecto configurable respecto al rango de posiciones que nos permite un mando de tipo rotatorio se refiere a las subdivisiones.

Podemos elegir el número de saltos que existirá entre cada una de las divisiones principales, utilizando la propiedad numSubDivs. En el ejemplo anterior podemos observar que la propiedad numSubDivs vale 5 y por ese motivo cada marca división está dividida a su vez en cinco subdivisiones.

También podemos controlar el aspecto de las marcas de divisiones y subdivisiones. Podemos fijar la longitud de los trazos de las divisiones con la propiedad 'tickLength', la longitud de los trazos de las subdivisiones con la propiedad 'subTickLength' y si se tienen que mostrar o no con la propiedad 'showTicks'.



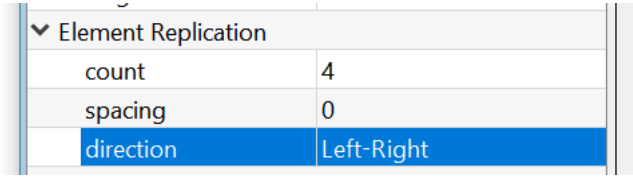
Es posible seleccionar el color y la fuente de los rótulos de cada marca de la escala usando las propiedades 'tickColour' y 'valueFont'.



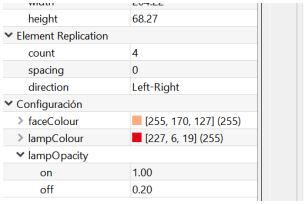
Se puede habilitar/deshabilitar en algunos mandos el efecto de animación de la aguja indicadora (por ejemplo en el velocímetro) configurando la propiedad 'animateNeedle'. Este efecto añade un mayor realismo a la animación del mando cuando se utiliza para mostrar aceleraciones o efectos relacionados con el 'momento' de una fuerza.

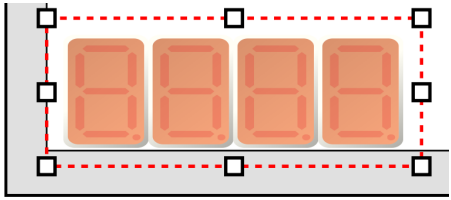
8.3.3.3.-Propiedades de los controles del tipo siete segmentos.

Este tipo de controles permite configurar los diferentes elementos que componen el control y para los 'segmentos luminosos' que componen cada elemento. Podemos cambiar el número de elementos, el espacio entre ellos y la dirección en que se leerán.



Respecto a los segmentos luminosos podemos seleccionar el color del fondo, el color del segmento y el grado de transparencia cuando se encuentra apagado o encendido. De esta manera controlamos totalmente el aspecto de cada display.





8.3.3.4.-Propiedades de los controles del tipo termómetro.

Los controles del tipo termómetro de mercurio son muy sencillos y, aparte de la configuración ya mencionada del apartado Geometría tienen una única propiedad que nos permite seleccionar si las unidades se expresan en grados Celsius o Farenheit.

8.3.3.5.- Propiedades de los controles del tipo barra de leds y contadores de progreso.

Estos controles son capaces de devolver el valor proporcional en tantos por ciento basado en cálculos matemáticos llevados a cabo en el código. Pueden representar valores entre 0 y 100.

También están disponibles las propiedades 'init' que nos permite fijar el valor inicial de la barra de leds, 'barColour' que nos permite seleccionar el color de la barra de leds y la propiedad 'animateBar' que nos permite seleccionar si la barra de progreso está animada o no.

8.3.3.6.-Propiedades de los controles del tipo WiFi, medición de señal o carga de batería.

La propiedad 'units' nos posibilita la elección entre mostrar un porcentaje del total o un número de pasos. Si seleccionamos porcentaje se mostrarán tantos leds encendidos como se correspondan con el tanto por ciento del valor total proporcionalmente al número de leds que contenga el control. Cada posición de la barra de led sólo se iluminará cuando el valor medido se encuentre por encima de la mitad de los valores fijados para la posición anterior y ésta. La única excepción a esta regla es la primera de las posiciones que se iluminará en cuanto el valor medido supere el cero.

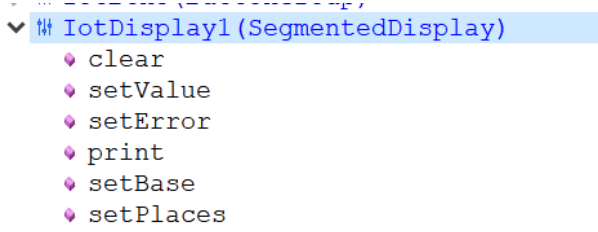
La propiedad 'LEDonColour' nos permite elegir el color del led activo y la propiedad 'LEDoffColour' el color del led inactivo.

8.3.3.7.-Propiedades de los controles tipo barra de gauge.

Estos controles disponen de las siguientes propiedades específicas: 'tickDis' fija la distancia de la barra de indicaciones con respecto al centro del control., 'labelDis' fija la distancia de las unidades de medida del centro del control, 'emptyColour' nos permite elegir l color de la barra Gauge cuando está vacía y 'gaugeColour' el color de la barra Gauge cuando está llena.

8.3.4.-Métodos de control.

Los controles de indicación analógica pueden ser gobernados por el software mediante la utilización de los métodos que el Diseñador Gráfico pone a nuestra disposición.



8.3.4.1.-Display de 7 segmentos.

En el caso de los displays de 7 segmentos podemos usar los siguientes métodos.

'Clear' apaga todos los segmentos.

'setValue' nos permite indicar el valor numérico que se mostrará en el display.

'setError' muestra una 'E' en todos los dígitos del display para indicar que se ha producido un error.

'print' muestra el valor de una variable numérica, booleana o de texto en el display.

'setBase' nos permite configurar si el control mostrará los valores en decimal, binario, octal o hexadecimal.

'setPlaces' nos permite elegir el número de dígitos decimales que se visualizará en el display.

8.3.4.2.-Termómetros.

En el caso de los termómetros, disponemos del método 'setTemperature' para especificar el valor de la temperatura a mostrar por el control.

8.3.4.3.-Mandos rotatorios.

La propiedad 'setLabel' permite fijar el nombre del mando. De esta forma es posible sobrescribir cualquier etiqueta que se hubiera especificado en el momento del diseño en función de las necesidades de la programación.

La propiedad 'setValue' se utiliza igual que en el caso de los displays de 7segmentos.

8.3.4.4.-Barras de progreso, barra de leds y barras Gauge.

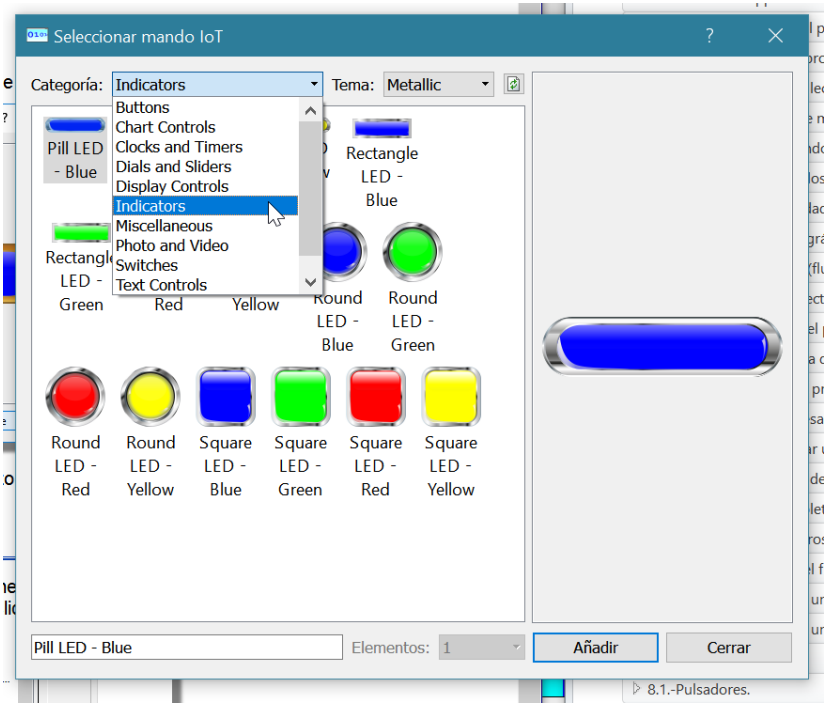
En estos mandos podemos utilizar la propiedad 'setValue' igual que en el caso de los displays de 7 segmentos.

8.4.-Pilotos.

8.4.1.-Descripción.

El grupo ‘indicators’ incluye mandos de tipo pilotos luminosos de varias formas y tamaños que pueden ser utilizados en nuestro panel de mando remoto para indicar la situación de los dispositivos. Todos ellos comparten el mismo conjunto de propiedades.

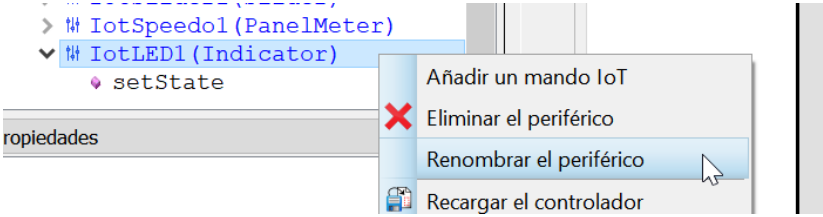
Para añadir un mando de este tipo, seguiremos el mismo camino que para todos los mandos y que ya hemos descrito con anterioridad.



8.4.2.-Configuración

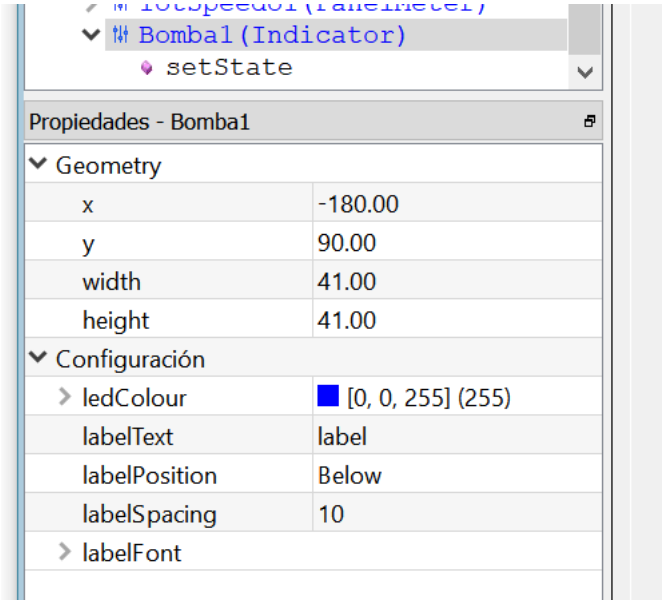
La configuración de los pilotos puede llevarse a cabo en tiempo de diseño cuando el mando es colocado en el cuadro de mando. Como siempre, es recomendable empezar asignando un nombre adecuado

al piloto que nos ayude a identificarlo de forma sencilla indicando su función desde el árbol de estructura de proyectos.



Una vez que le hayamos cambiado el nombre, ya podemos 'cogerlo y arrastrarlo' para situarlo sobre el cuadro de mando en el lugar que deseemos.

Una vez que lo hayamos colocado, podemos seleccionarlo y modificar sus propiedades desde el panel situado en la zona izquierda.



Las propiedades que utilizaremos con mayor frecuencia son las que se encuentran en el apartado 'Geometry' que permiten fijar su tamaño y posición.

El resto de propiedades disponibles son: ‘ledColour’ permite fijar el color del piloto, ‘labelText’ permite definir el rótulo de la etiqueta que acompaña al piloto para indicar su función, ‘labelPosition’ nos deja fijar la posición relativa de la etiqueta respecto al piloto e incluso ocultarla, con ‘labelSpacing’ podemos elegir la distancia entre la etiqueta y el piloto y, por último, ‘labelFont’ nos deja configurar la fuente utilizada para la etiqueta.

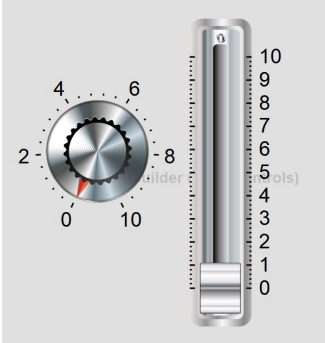
8.4.3.-métodos de control.

Los métodos de control disponibles nos permiten gestionar el piloto en tiempo de ejecución desde nuestro código fuente. Para un piloto sólo existe un método asociado llamado ‘setValue’. Es de tipo booleano y establece si el piloto estará encendido o apagado.

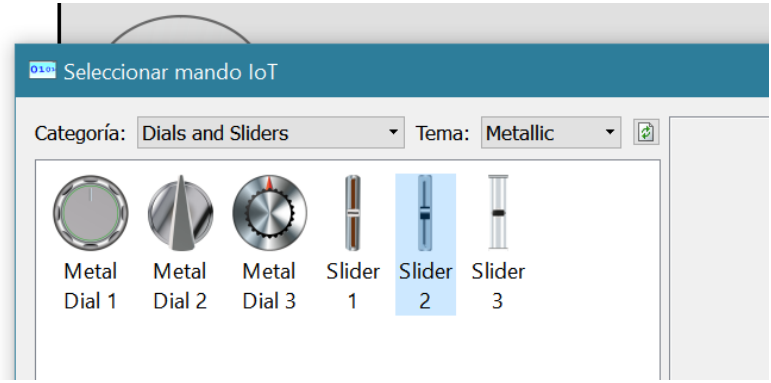
8.5.-Mandos rotatorios y Deslizantes.

8.5.1.-Descripción.

Las librerías suministradas con Proteus incluye tanto los mandos rotatorios como los mandos deslizantes.



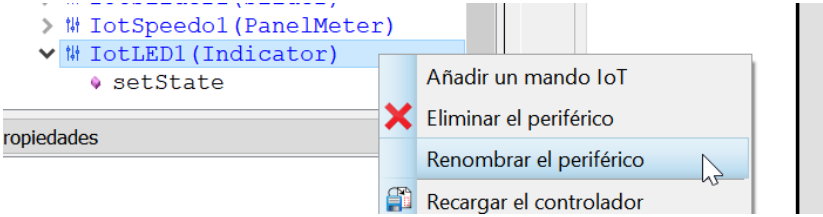
Para añadir un mando de este tipo, seguiremos el mismo camino que para todos los mandos y que ya hemos descrito con anterioridad.



8.5.2.-Configuración

La configuración de los pilotos puede llevarse a cabo en tiempo de diseño cuando el mando es colocado en el cuadro de mando. Como siempre, es recomendable empezar asignando un nombre adecuado

al piloto que nos ayude a identificarlo de forma sencilla indicando su función desde el árbol de estructura de proyectos.



Una vez que le hayamos cambiado el nombre, ya podemos 'cogerlo y arrastrarlo' para situarlo sobre el cuadro de mando en el lugar que deseemos

> Si deseamos utilizar una etiqueta para indicar la función de un mando rotatorio o deslizante tenemos que realizarlo colocando un elemento de tipo 'texto' sobre el nuestro cuadro de mando remoto junto al mando, porque este tipo de mandos no incluye esta funcionalidad.

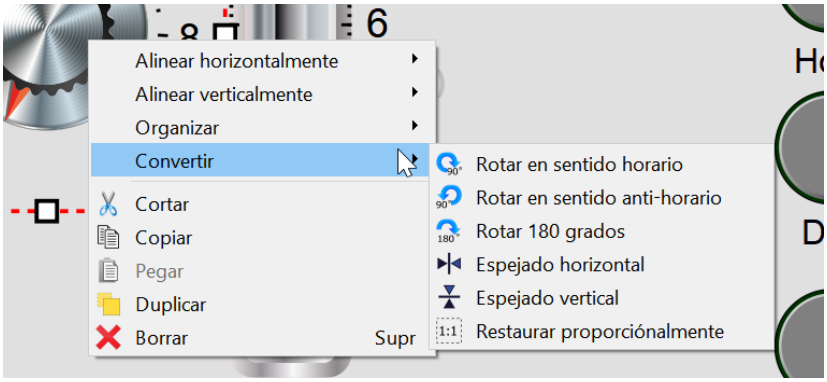
8.5.2.1.-Las propiedades del apartado 'Geometry'.

Las propiedades que utilizaremos con mayor frecuencia son las que se encuentran bajo el apartado 'Geometry' y que nos permiten fijar el tamaño y posición del mando.

Geometry	
x	-177.00
y	-103.36
width	155.00
height	152.00

Por ejemplo, podemos fijar el valor de la propiedad 'x' a toda una serie de mandos para alinearlos verticalmente.

Las propiedades 'width' y 'height' nos permiten fijar el tamaño del mando, aunque suele ser más cómodo fijar el tamaño utilizando el ratón y al final usar la opción del menú contextual 'Restaurar proporcionalmente' que se encuentra en el apartado 'Convertir' para restablecer su proporción.



8.5.2.2.-Mandos rotatorios.

La propiedad 'init' fija el valor inicial que marcará el mando.

Las propiedades 'max' y 'min' selecciona el rango de valores que puede mostrar el mando.

Las propiedades 'numDivs' y 'numSubDivs' fija el número de saltos entre las marcas principales y secundarias.

Las propiedades 'tickLength' y 'subTickLength' fija el tamaño de las marcas principales y secundarias.

Las propiedades 'showTicks' y 'showLabels' determina la visibilidad de las marcas y de los rótulos de las marcas.

La propiedad 'snapToTicks' fuerza el ajuste automático a la marca más cercana cuando se actúa sobre el mando. De esta forma se posibilita un manejo más preciso del mando en tiempo de ejecución.

La propiedad 'fadeOut' establece si la lectura fuera de una posición marcada se visualiza de forma permanente o desaparece pasado un breve espacio de tiempo después de que se haya llevado a cabo un cambio en la posición del mando en tiempo de ejecución.

Las propiedades 'tickColour', 'valueFont' y 'labelFont' establecen el color, la fuente y el tamaño de las etiquetas y de las posiciones de marca del mando rotatorio.

8.5.2.3.-Mandos deslizantes.

Los mandos deslizantes tienen las mismas propiedades que los mandos rotatorios, más un par de ellas que son específicas de este tipo de mandos.

La propiedad ‘orientation’ permite elegir si el mando se muestra en posición horizontal o vertical. Es preferible utilizar esta opción a utilizar la herramienta de dibujo ‘girar’ porque de esta manera nos aseguramos que las etiquetas y marcas girarán y se ubicarán correctamente.

Las propiedades ‘valuePosition’ y ‘labelPosition’ fijan si el valor y la etiqueta se colocan en función de la posición y orientación del mando deslizante.

8.5.3.-métodos de control.

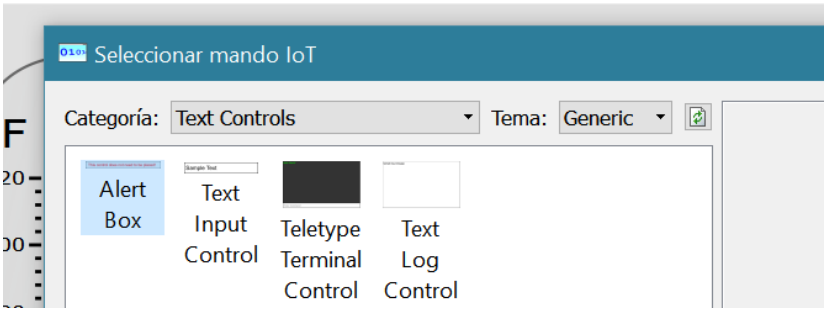
Los métodos de control disponibles nos permiten gestionar el mando en tiempo de ejecución desde nuestro código fuente.

Para los mandos rotatorios y deslizantes tenemos dos métodos asociados. Uno llamado ‘setValue’ fija el valor indicado por el mando. El otro llamado ‘getValue’ lee el valor indicado por el mando en su actual posición.

8.6.-Paneles de alarma.

8.6.1.-Descripción.

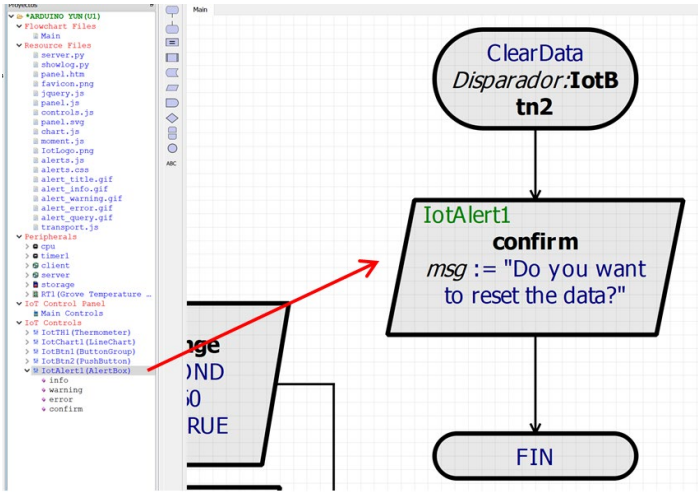
Los paneles de alarma se utilizan para presentar un cuadro de mensaje emergente al usuario en tiempo de ejecución con información relevante y con un botón para solicitar la confirmación de lectura. A diferencia del resto de controles no es necesario colocarlos en nuestro cuadro de mando remoto, aunque si es necesario incorporarlo al proyecto y que aparezca en el árbol de estructura del proyecto. Esta última operación se realiza siguiendo el mismo método de siempre.



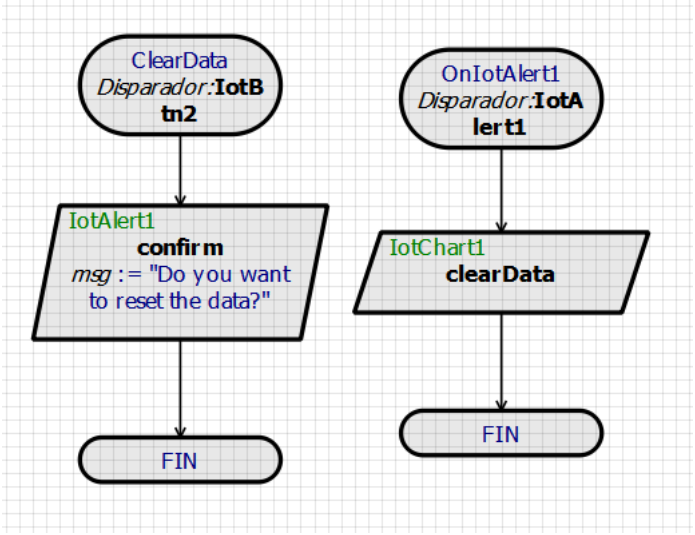
El procedimiento para utilizarlos se lleva a cabo en tres pasos.

Añadiremos el mando a nuestro proyecto en la forma habitual.

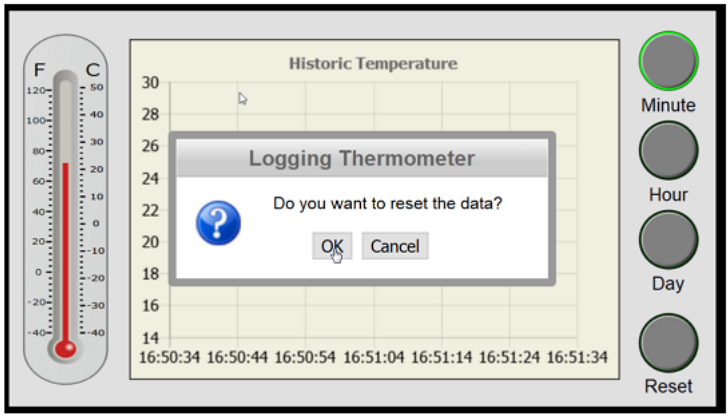
'Cogeremos y arrastraremos' el método de control asociado con el mando del tipo que deseemos (información, aviso, error, confirmación) y lo colocaremos en el lugar de nuestro flujograma que deseemos. Escribiremos el mensaje que deseemos que se muestre.



Establece el código para definir el evento que lanzará el proceso y gestionar la respuesta del usuario cuando pulse con el ratón en el botón correspondiente de la ventana emergente que aparece.



El flujograma de la imagen fuerza la presentación del panel de alarma en el cuadro de mando remoto (teléfono móvil, tableta o navegador) como se muestra en la siguiente imagen.



En los proyectos de ejemplo suministrados con Proteus se encuentra uno llamado 'Arduino Temperature Logging' que contiene un ejemplo de utilización de un panel de alarma que puede ser una buena forma de comprobar el funcionamiento de este tipo de controles.

8.6.2.-Configuración

No hay propiedades a configurar en tiempo de diseño para este tipo de controles.

8.6.3.-métodos de control.

Los cuatro métodos asociados a este tipo de controles no necesitan de muchas explicaciones porque sólo determinan el tipo de panel de alarma que se mostrará. El aspecto más importante al usarlos en nuestro flujograma es asegurarnos de dos cosas:

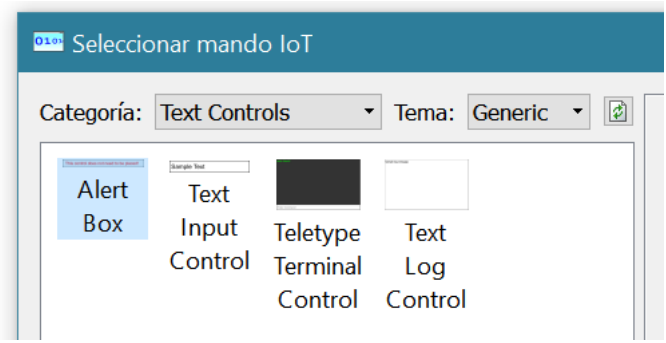
Que la ejecución del programa de control de nuestro equipo continúa mientras un panel de alarma esté activo y mostrándose.

Que el evento asociado sólo se ejecuta cuando se detecta la respuesta afirmativa por parte del usuario.

8.7.-Cuadros de texto y terminales de texto.

8.7.1.-Descripción.

Proteus pone a nuestra disposición varios mandos de introducción de texto. El más simple consiste de una única línea de texto. Pero también tenemos otros que permiten múltiples líneas para su uso como un terminal de texto o como un histórico de eventos y alarmas. En cualquier caso, los utilizaremos siguiendo el mismo método que hemos visto en todos los casos anteriores.



8.7.2.-Configuración

8.7.2.1.-Mandos tipo entrada de texto.

Están disponibles las siguientes propiedades.

Las incluidas en el Grupo geometry que sirven para posicionar y dimensionar el mando.

La propiedad ‘readOnly’. Cuando su valor es falso, permite que el texto se introduzca por parte del usuario y que sea leído utilizando una cadena de texto desde el código. Cuando es verdadero no se puede modificar el texto por parte del usuario y sólo sirve para presentar mensajes al usuario.

La propiedad ‘placeHolder’ establece el texto a mostrar.

La propiedad ‘maxLenght’ Fija el tamaño máximo de la cadena de texto a mostrar.

8.7.2.2.-Mandos tipo terminal de texto.

Están disponibles las siguientes propiedades.

Las incluidas en el Grupo geometry que sirven para posicionar y dimensionar el mando.

La propiedad ‘textFont’ permite establecer las propiedades de la fuente utilizada (tamaño, color, familia).

La propiedad ‘placeHolder’ establece el texto a mostrar.

La propiedad ‘maxLenght’ Fija el tamaño máximo de la cadena de texto a mostrar.

La propiedad ‘clsCmd’ fuerza a que se necesite una instrucción para limpiar la pantalla de un texto escrito con anterioridad.

8.7.2.3.-Mandos tipo histórico.

Están disponibles las siguientes propiedades.

La propiedad ‘textFont’ permite establecer las propiedades de la fuente utilizada (tamaño, color, familia).

La propiedad ‘warnColour’ estable el color para escribir mensajes de tipo aviso.

La propiedad ‘errorColour’ estable el color para escribir mensajes de tipo error.

La propiedad ‘backgroundColour’ estable el color de fondo de la ventana.

8.7.3.-métodos de control.

8.7.3.1.-Mandos tipo entrada de texto.

Están disponibles los siguientes métodos.

El método ‘setText’ posibilita la escritura de un determinado texto y permite que el programador pueda aceptarlo o esperar a que el usuario lo edite.

El método ‘getText’ lee el valor introducido.

El método ‘clear’ elimina el texto actual.

El método ‘setError’ muestra un determinado mensaje en letras rojas debajo del texto principal. Se utiliza principalmente para poder mostrar mensajes al usuario cuando introduce un valor en un formato o de un tamaño diferente al esperado por el código.

8.7.3.2.-Mandos tipo terminal de texto.

Están disponibles los siguientes métodos.

El método ‘print’ muestra un texto con la cadena de texto indicada. Sucesivas llamadas a este método imprimirá la nueva cadena de texto donde haya terminado la anterior, en la misma línea si el tamaño lo permite.

El método ‘pintln’ muestra un texto con la cadena de texto indicada y genera un salto de línea antes para que la cadena de texto aparezca en una nueva línea.

El método ‘setBase’ establece la base para la codificación de valores numéricos (decimal, binario, octal o hexadecimal).

El método ‘setPlaces’ indica el número de posiciones decimales utilizadas para mostrar un número de tipo coma flotante.

El método ‘setPrompt’ establece una cadena para utilizarla como información o pregunta cuando se quiere solicitar entrada de texto al usuario. Sobrescribe el valor introducido en la propiedad ‘placeHolder’.

El método ‘getCommand’ devuelve el valor introducido por el usuario dentro de una variable o una cadena vacía si no introdujo nada.

El método ‘cls’ limpia la ventana.

8.7.3.3.-Mandos tipo histórico.

Están disponibles los siguientes métodos.

El método ‘setFile’ fija el nombre del fichero donde se almacenan los datos.

El método ‘setBase’ establece la base para la codificación de valores numéricos (decimal, binario, octal o hexadecimal).

El método ‘setPlaces’ indica el número de posiciones decimales utilizadas para mostrar un número de tipo coma flotante.

El método ‘info’ escribe un mensaje de tipo informativo en la pantalla.

El método ‘warning’ escribe un mensaje de tipo aviso en la pantalla.

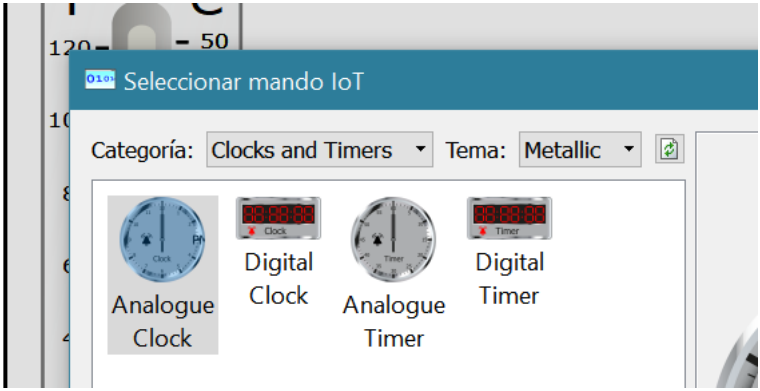
El método ‘error’ escribe un mensaje de tipo error en la pantalla.

El método ‘clear’ limpia la ventana.

9.-Mandos avanzados.

9.1.-Relojes y temporizadores.

Proteus ofrece varios mandos que pueden ser utilizados para mostrar relojes y temporizadores en nuestro cuadro de mando remoto y que nos facilitan trabajar con variables de tiempo en nuestra aplicación.



9.1.1.-Funcionamiento de los relojes y temporizadores.

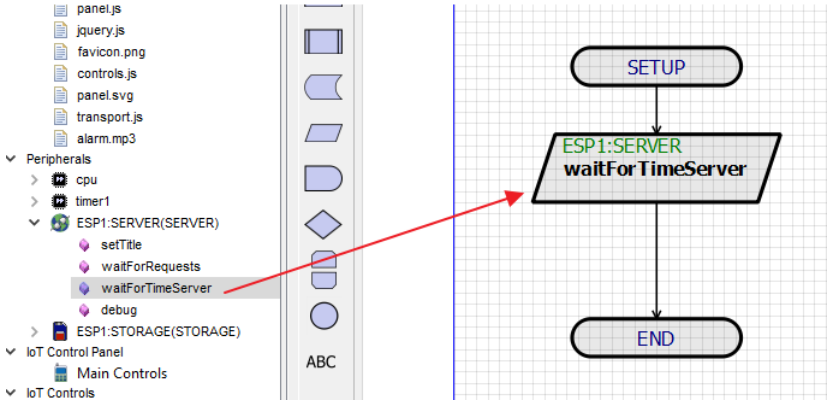
A menos de que esté familiarizado con el uso de relojes y temporizadores, es muy importante que lea detenidamente este apartado para comprender como ambos, el código y el interface de usuario, trabajan con las variables de tipo temporal.

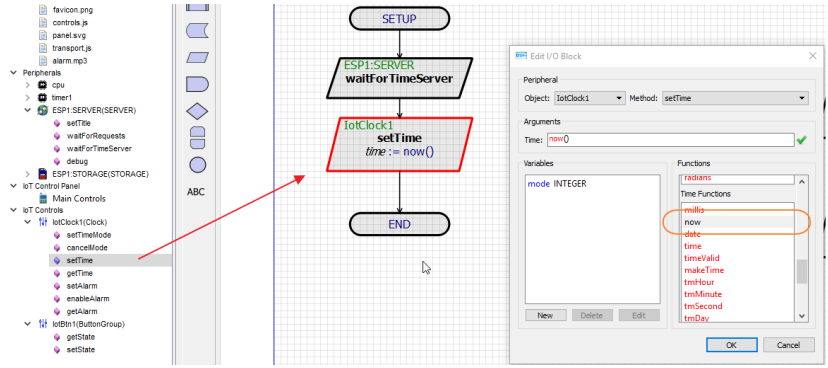
9.1.1.1.-Establecer y leer el tiempo.

Se pueden usar dos métodos para fijar una determinada hora.

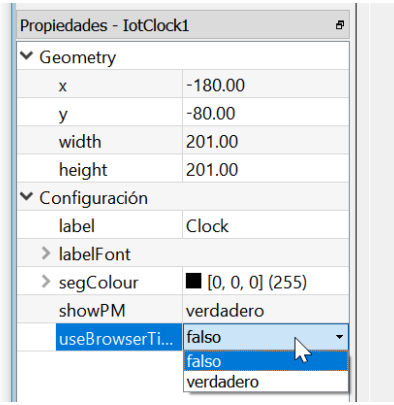
Utilizar el servidor Linux. En este caso, usaremos el método 'waitForTimeServer()' para activar este servicio. Este método normalmente se encuentra situado dentro de la rutina de configuración (setup). Al llamarla le indicamos al microprocesador AVR que utilice el chip 'Atheros' (que a su ve usa el sistema

operativo Linux que lleva incorporado) como proveedor de la hora actual. A partir de ese momento, podemos utilizar el método 'now()' tantas veces cómo deseemos para leer la hora actual y, por ejemplo, asignarla a una determinada variable.





Utilizar el navegador. En este caso recibimos la hora directamente del navegador y será la misma que utilice el interface de usuario. Esto se lleva a cabo en tiempo de diseño utilizando la propiedad 'useBrowseTime' del panel de propiedades.

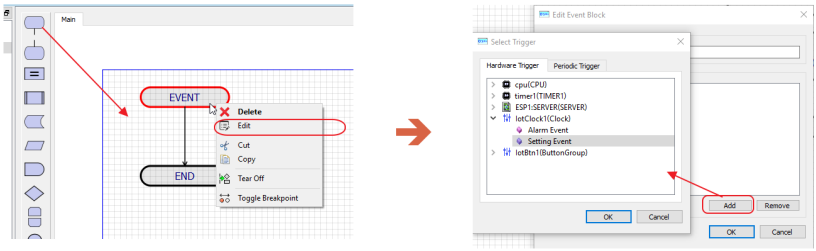


⚠ Cuando seleccionamos la opción de utilizar el navegador como servidor de tiempo, la aplicación (nuestro código) no tiene necesariamente que saber la hora. Por este motivo el primer método es mucho más seguro porque el hardware envía directamente a nuestro interface de usuario la información de la hora.

9.1.1.2.-Eventos temporales.

Cada vez que deseemos fijar un temporizador tendremos que usar un evento que se encargue de llevar a cabo una determinada acción. Entre otras cosas, con ello podemos pasar la información de la hora actual del interface de usuario al dispositivo si hemos utilizado el navegador como servidor de tiempos. Aunque es muy importante tener en cuenta que el dispositivo no dispondrá de esta información si el navegador no está activo.

Es posible crear una respuesta a un determinado evento utilizando una subrutina de tipo evento y eligiendo como disparador un reloj (setting event). También es posible elegir como disparador una alarma (alarm event). Si seleccionamos esta segunda opción, el procedimiento asociado al evento se ejecutará cuando se alcance el tiempo fijado (cuando el contador de tiempo alcance el valor prefijado).



En los proyectos de ejemplo suministrados con Proteus se encuentra uno llamado 'Arduino IoT Alarma Clock' que contiene un ejemplo de utilización de temporizadores que puede ser una buena forma de comprobar el funcionamiento de este tipo de controles.

9.1.2.-Relojes.

9.1.2.1.-Configuración.

Las propiedades disponibles para los mandos tipo reloj son:

El grupo de propiedades 'Geometry' para controlar la posición y tamaño del mando.

La propiedad 'label' para fijar el texto que mostrará la etiqueta con el nombre del reloj. La propiedad 'labelFont' para configurar el formato, color, estilo de la fuente utilizada para la etiqueta.

La propiedad 'useBrowserTime' para establecer como fuente para obtener la hora actual el navegador o el servidor de tiempo Linux. Consultar más arriba en este tutorial el epígrafe que explica cómo establecer y leer el tiempo.

La propiedad 'segColour' se utiliza sólo en los relojes de tipo digital y permite seleccionar el color de los segmentos de los dígitos utilizados para mostrar la hora.

La propiedad 'showPM' se utiliza sólo en los relojes de tipo analógico y permite elegir si se mostrará o no el indicador 'PM' para diferenciar si estamos mostrando la hora de antes o después del mediodía en relojes de 12 horas.

9.1.2.2.-Métodos de control.

Podemos utilizar los siguientes métodos con los mandos tipo reloj:

El método ‘setTimeMode’ nos permite pasar una determinada hora al mando reloj. Si utilizamos el valor 0 para el argumento ‘mode’ estamos fijando la hora del reloj. Si utilizamos un valor 1 a 8 para el argumento ‘mode’ estamos estableciendo una alarma para esa hora. Por ejemplo, setTimeMode(0) establece la hora, setTimeMode(1) fija la hora de la alarma 1, setTimeMode(2) fija la hora de la alarma 2 y así sucesivamente.

El uso de este método se puede ver en el proyecto ejemplo suministrado con Proteus llamado ‘Alarm Clock’, donde se utilizan varios pulsadores en conjunción con el método SetTimeMode para establecer la hora y dos alarmas del reloj.



El método ‘cancelMode’ cancela la orden del usuario que se haya introducido utilizando el método ‘SetTimeMode’.

El método ‘setTime’ establece la hora mostrada en el reloj a una valor específico, normalmente la hora real usando la función now(). El argumento se pasa expresado en segundos contados a partir del 1 de Enero de 1970 como es habitual en muchos sistemas operativos.

El método ‘getTime’ lee la hora actualmente mostrada por el reloj.

El método ‘setAlarm’ fija la hora y minutos para cada una de las alarmas disponibles.

El método ‘enableAlarm’ conmuta una determinada alarma de encendida a apagada o viceversa.

El método ‘getAlarm0 lee la hora fijada para una determinada alarma.

9.1.3.-Temporizadores.

> En los proyectos de ejemplo suministrados con Proteus se encuentra uno llamado ‘Countdown Timer’ que contiene un ejemplo de utilización de temporizadores que puede ser una buena forma de comprobar el funcionamiento de este tipo de controles.

9.1.3.1.-Configuración.

Las propiedades disponibles para los mandos tipo reloj son:

El grupo de propiedades ’Geometry’ para controlar la posición y tamaño del mando.

La propiedad ‘label’ para fijar el texto que mostrará la etiqueta con el nombre del reloj. La propiedad ‘labelFont’ para configurar el formato, color, estilo de la fuente utilizada para la etiqueta.

La propiedad ‘dir’ almacena la dirección (cuenta atrás o cuenta hacia adelante) del contador para el temporizador. Por defecto se establece una cuenta atrás. Así que si establecemos la propiedad ‘period’ a 10 segundos, el temporizador comenzará a contar en 10 segundos e irá retrocediendo hasta alcanzar el valor 0. Si estableciéramos una cuenta hacia adelante empezaría a contar desde 0 hasta alcanzar el valor 10.

La propiedad ‘mode’ puede tener dos valores. ‘Oneshot’ representa un contador simple a menos que por código detengamos y reanudemos la cuenta. Mientras que ‘Repeat’ genera alarmas de forma periódica a determinados periodos de tiempo. Podemos usar estos temporizadores como disparadores de una determinada

subrutina para establecer que se ejecute de forma periódica. Por ejemplo, crear una entrada en una bitácora de histórico cada 30 segundos.

La propiedad ‘period’ establece las horas, minutos y segundos para un temporizador. Se puede utilizar usando métodos para poder programar este valor desde el código.

La propiedad ‘enableAlarm’ permite anular la alarma cuando una determinada condición se cumple. También hay métodos disponibles para poder programar este funcionamiento desde el código.

La propiedad ‘segColour’ fija el color de los segmentos de los dígitos utilizados para mostrar la hora.

9.1.3.2.-Métodos de control.

Podemos utilizar los siguientes métodos con los mandos tipo reloj:

El método ‘start()’ arranca el temporizador.

El método ‘stop()’ detiene el temporizador.

El método ‘restart’ reinicia un temporizador detenido en el punto donde se detuvo.

El método ‘setPeriod’ estable el tiempo que contará el temporizador.

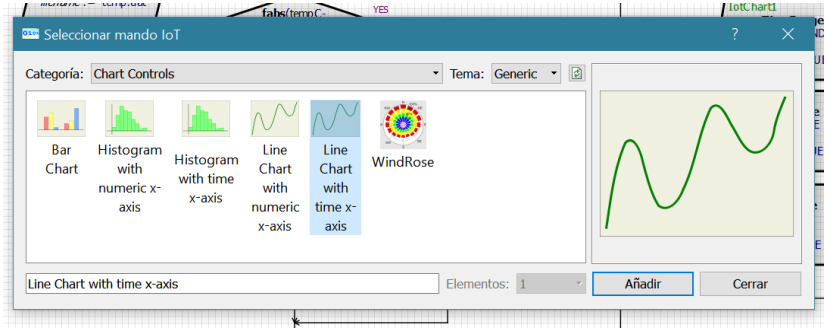
El método ‘enableAlarm’ activa o desactiva la alarma. Si se desactiva y utilizamos un disparador para arrancar una determinada subrutina cuando se alcance el tiempo fijado por el temporizador, ésta no se ejecutará. Por defecto el valor es activo.

El método ‘getTime’ devuelve el valor actual del temporizador.

9.2.-Gráficas lineales.

9.2.1.-Descripción.

La gráfica lineal es un mando que permite registrar, almacenar y presentar información. Podemos utilizar una gráfica líneal con una base temporal o numérica de la forma habitual que hemos visto en todos los casos anteriores.



Si optamos por usar una base temporal y tener como eje-x valores de hora, es imprescindible usar en nuestro código el método 'waitForTimeServer()' en nuestra subrutina de configuración (setup), para que la aplicación sea capaz de conocer la hora actual en cada momento.

> En los proyectos de ejemplo suministrados con Proteus se encuentra uno llamado 'Arduino Temperature Logging Appliance' que contiene un ejemplo de utilización de una gráfica lineal que puede ser una buena forma de comprobar el funcionamiento de este tipo de controles.

9.2.2.-Configuración.

Las propiedades disponibles para los mandos tipo gráfica lineal son:

'tittleText' establece el valor de la etiqueta asociada a la gráfica.

'FontSizes' fija el tamaño de la fuente utilizada para la gráfica.

'showLegend' muestra u oculta la leyenda con la explicación de los datos mostrados. Si no se utiliza la leyenda puede ser difícil para el

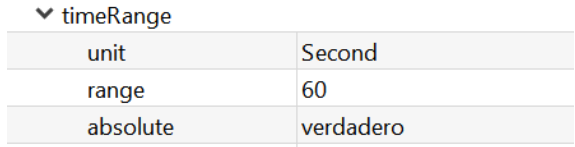
usuario identificar qué magnitud se representa con cada línea, pero si se utiliza puede verse mermado el espacio disponible para mostrar la gráfica.

‘pointRadius’ permite establecer el tamaño de los puntos utilizados en la gráfica. Si le asignamos un valor igual a cero no se mostrarán puntos y sólo aparecerá una línea simple.

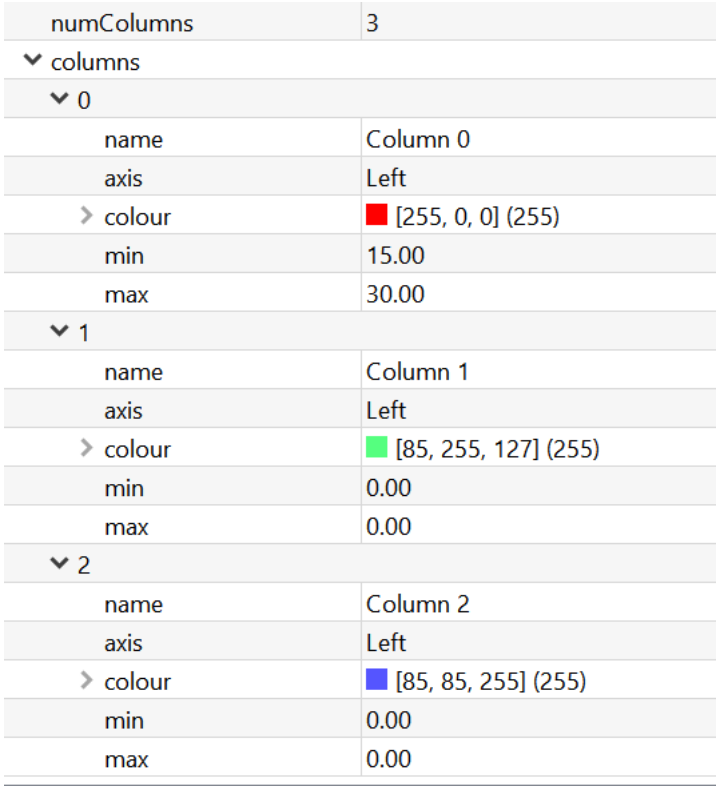
‘lineWidth’ configura el grosor de la línea utilizada en la gráfica. Si le asignamos un valor igual a cero no se mostrará la línea y sólo se mostrará los puntos de lectura.

‘backgroundColour’ establece el color de fondo de la gráfica.

‘timeRange’ se usa para establecer que el eje-X muestre los valores del tiempo. Sólo se puede usar esta propiedad en los mandos de gráfica lineal de tipo base de tiempos. ‘unit’ y ‘range’ fijan el rango de tiempos que mostrará la gráfica (segundos, minutos, horas). Si ‘absolute’ está fijado a verdadero, la gráfica empezará a mostrar datos desde el momento en que arranca la aplicación a funcionar. Si el valor está fijado a falso, la gráfica mostrará datos desde las 00:00.



‘numColumns’ permite fijar el número de columnas disponibles. Dependiendo del valor que asignemos a esta propiedad, podemos entonces configurar cada una de las columnas con su aspecto particularizado.



9.2.3.-Métodos de control.

Podemos utilizar los siguientes métodos con los mandos tipo gráfica lineal:

El método 'setXRange' nos posibilitar fijar el rango de valores del eje-X. Sólo funciona en las gráficas lineales de base numérica aunque la propiedad está disponible para los dos tipos.

El método 'setTimeRange' sirve para fijar el rango de valores del eje-Y a una determinada hora. La propiedad 'unit' determina el tipo de unidad utilizada (horas, minutos, segundos' y con esta propiedad cuántas de esas unidades se mostrarán. Si la propiedad 'absolute' está fijada a verdadero la gráfica mostrará la hora actual en que arrancó la aplicación. Si está fijada a falso el rango de tiempos empezará en 00:00.

El método ‘setYRange’ permite establecer el rango de unidades del eje-Y en un lateral de la gráfica. Si este rango supera el valor máximo o mínimo establecidos en las propiedades ‘max’ o ‘min’ la gráfica se ajustará automáticamente.

El método ‘showColumn’ muestra u oculta las líneas.

El método ‘setDataFile’ configura el fichero donde el mando escribirá y leerá los datos. Hay que introducir el nombre del fichero como una cadena de texto.

El método ‘setPlaces’ establece el número de posiciones decimales utilizado para mostrar los datos.

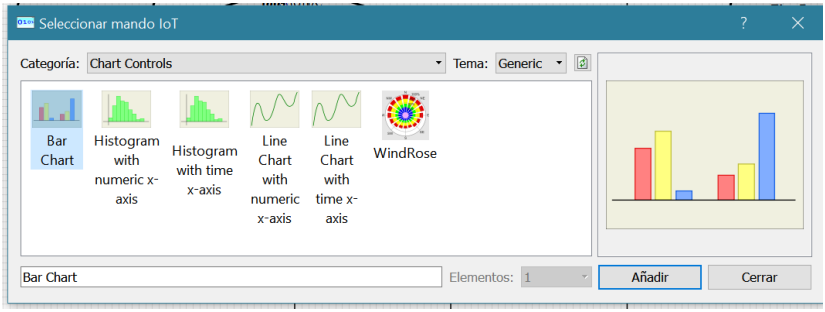
El método ‘writeData’ fuerza la escritura de un dato en la gráfica. No fuerza la actualización de la gráfica, pero almacena el valor en el fichero asociado.

El método ‘clearData’ limpia la gráfica y elimina el fichero de datos asociado.

9.3.-Gráficas de barras.

9.3.1.-Descripción.

La gráfica lineal es un mando que permite registrar, almacenar y presentar información que permite capturar diferentes conjuntos de datos y almacenarlos en diferentes ficheros. Los diferentes conjuntos de datos pueden ser mostrados unos al lado de otros para poder realizar comparaciones entre ellos. No es posible acceder a los ficheros de datos directamente y es necesario usar métodos de tipo escritura y lectura para interactuar con ellos. El mando ofrece muchas posibilidades para configurar el aspecto de las barras y la leyenda utilizadas.

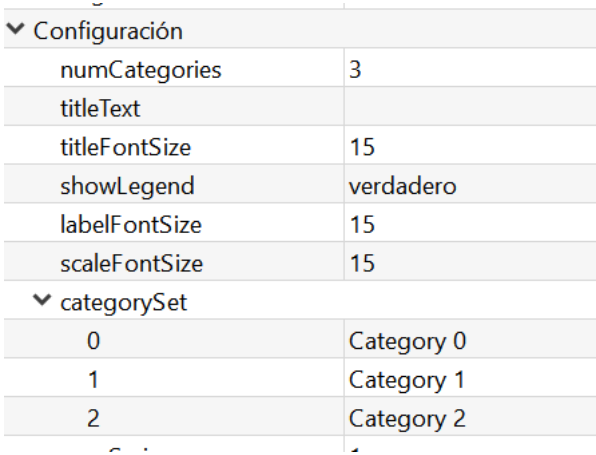


9.3.2.-Configuración.

Las propiedades disponibles para los mandos tipo gráfica de barras son:

‘numCategories’ permite fijar el número de categorías disponibles. Una categoría es un conjunto de datos para un analizar un determinado objetivo. Las categorías no interactúan entre sí a menos que se indique expresamente y pueden contener datos tomados en periodos de tiempo diferentes.

‘CategorySet’ permite establecer un nombre diferente para cada una de las categorías existentes.



Configuración	
numCategories	3
titleText	
titleFontSize	15
showLegend	verdadero
labelFontSize	15
scaleFontSize	15
categorySet	
0	Category 0
1	Category 1
2	Category 2

‘numSeries’ fija el número de series que contendrá la gráfica. Una serie es un conjunto de datos de una determinada categoría. Es muy importante distinguir entre el comportamiento de la serie 0 y el resto de series posibles de utilizar en el gráfico. La serie 0 se mantiene en la memoria y se puede manipular desde el código utilizando los métodos disponibles como, por ejemplo, setData()’. Para poder optimizar la utilización de los recursos disponibles, el resto de las series se almacenan en ficheros y sólo se pueden manipular utilizando los métodos WriteSeries() y loadSeries(). Por lo tanto si deseamos disponer de múltiples conjuntos de datos en nuestra gráfica el programa puede complicarse en tanto que es necesario controlar el cambio de datos a incluir en las otras series y llenar el gráfico con ellos.

setSeries() permite fijar el color de cada serie y de las etiquetas asociadas.

‘tittleText’ establece el valor de la etiqueta asociada a la gráfica para utilizarla como título.

‘tittleFont’ ‘scaleFontSize’ ‘labelFontSize’ fija el tamaño de la fuente utilizada para la gráfica.

‘showLegend’ muestra u oculta la leyenda con la explicación de los datos mostrados. Si no se utiliza la leyenda puede ser difícil para el usuario identificar qué magnitud se representa con cada línea, pero

si se utiliza puede verse mermado el espacio disponible para mostrar la gráfica.

'minY' y 'minX' fijan los límites de los valores de los ejes. El gráfico se autoajusta si los valores leídos exceden los límites configurados en estas propiedades.

9.3.3.-Métodos de control.

Podemos utilizar los siguientes métodos con los mandos tipo gráfica de barras:

El método 'showSeries' permite mostrar u ocultar las series. Resulta útil cuando se visualizan diferentes cantidades de datos. No es posible crear una serie con este método y el número de series se configura en tiempo de diseño, pero podemos visualizarlas u ocultarlas utilizando este método.

El método 'setYRange' permite establecer el rango de unidades del eje-Y en un lateral de la gráfica desde el código.

El método 'setData' nos permite añadir cualquier valor a la serie 0. Sólo se puede manejar los datos de la serie 0 usando este método. Cuando se trabaje con múltiples series, lo típico será:

- Trabajar con la serie 0.
- Usar el método writeDate() para guardar la serie 0.
- Usar el método loadData() para cargar la serie 0 dentro de otra serie, por ejemplo, la serie 1.
- Usar el método resetSeries() y empezar a trabajar de nuevo con la serie 0.

Sólo la serie 0 se mantiene en la memoria para preservar los recursos disponibles del microprocesador AVR.

El método 'incrementData' permite incrementar la serie 0 de cualquier categoría con un valor establecido.

El método 'decrementData' es el método inverso de 'incrementData'. Permite eliminar un valor establecido de la serie 0 de cualquier categoría.

El método ‘resetSeries’ permite borrar los datos de una serie individual. Aplica a todas las categorías.

El método ‘writeSeries’ escribe todas las series 0 en un determinado fichero que indicaremos usando una variable de cadena de texto. Si el fichero existe lo sobrescribe. De esta manera podemos reutilizar varias veces un mismo fichero.

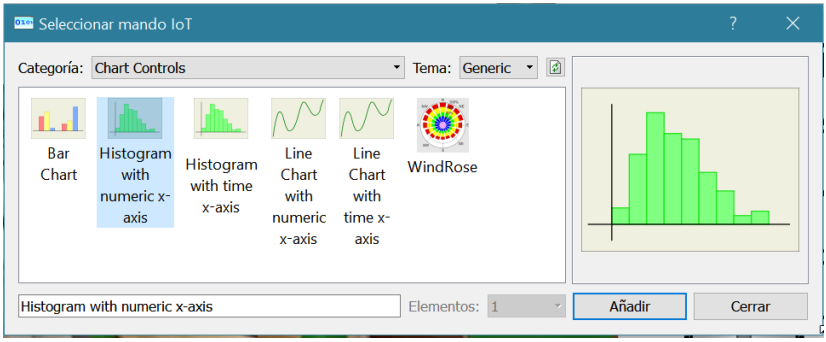
El método ‘loadSeries’ se utiliza como el paso siguiente a método ‘writeSeries’ y nos permite leer los datos guardados en un fichero y usarlo como una serie. El caso más simple es escribir los datos de la serie 0, borrarlos y más tarde aplicar esos datos a la serie 2.

El método ‘clearData’ borra todos los ficheros de datos y limpia la gráfica.

9.4.-Gráficas de Históricos.

9.4.1.-Descripción.

La gráfica de histórico o histograma es un mando que permite registrar, almacenar y presentar información. Puede parecer muy similar a la gráfica de barras, pero su comportamiento se asemeja más a la gráfica lineal. Puede ser de tipo base de tiempos o del tipo base de valores.



9.4.2.-Configuración.

Las praopiedades disponibles para los mandos tipo gráfica de histórico son:

‘tittle’ establece el valor de la etiqueta asociada a la gráfica.

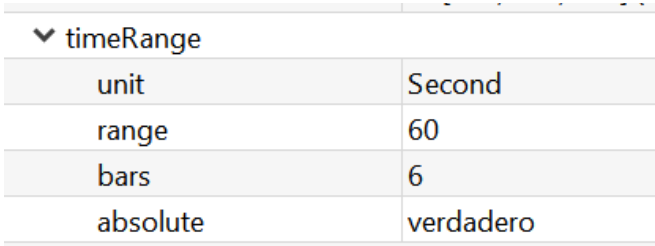
‘tittleFont’ ‘scaleFontSize’ ‘labelFontSize’ fija el tamaño de la fuente utilizada para la gráfica.

‘Background’ y ‘bar colour’ permiten elegir los colores del fondo y de las barras de la gráfica.

‘X range’ e ‘Y range’ sirven para fijar el rango mínimo y máximo de cada eje y el número de barras que se mostrarán.

‘timeRange’ se usa para establecer que el eje-X muestre los valores del tiempo. Sólo se puede usar esta propiedad en los mandos de gráfica de históricos de tipo base de tiempos. ‘unit’ y ‘range’ fijan el rango de tiempos que mostrará la gráfica (segundos, minutos, horas). Si ‘absolute’ está fijado a verdadero, la gráfica empezará a

mostrar datos desde el momento en que arranca la aplicación a funcionar. Si el valor está fijado a falso, la gráfica mostrará datos desde las 00:00. 'bars' permite indicar el número de barras visualizadas.



timeRange	
unit	Second
range	60
bars	6
absolute	verdadero

9.4.3.-Métodos de control.

Podemos utilizar los siguientes métodos con los mandos tipo gráfica de histórico:

El método 'setXRange' nos posibilitar fijar el rango de valores del eje-X. Sólo funciona en las gráficas lineales de base numérica aunque la propiedad está disponible para los dos tipos.

El método 'setTimeRange' sirve para fijar el rango de valores del eje-Y a una determinada hora. La propiedad 'unit' determina el tipo de unidad utilizada (horas, minutos, segundos' y con esta propiedad cuántas de esas unidades se mostrarán. Si la propiedad 'absolute' está fijada a verdadero la gráfica mostrará la hora actual en que arrancó la aplicación. Si está fijada a falso el rango de tiempos empezará en 00:00.

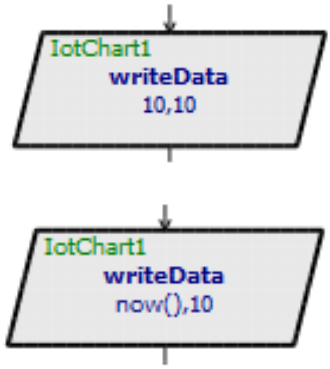
El método 'setYRange' permite establecer el rango de unidades del eje-Y en un lateral de la gráfica. Si este rango supera el valor máximo o mínimo establecidos en las propiedades 'max' o 'min' la gráfica se ajustará automáticamente.

El método 'showColumn' muestra u oculta las columnas.

El método 'setDataFile' configura el fichero donde el mando escribirá y leerá los datos. Hay que introducir el nombre del fichero como una cadena de texto.

El método ‘setPlaces’ establece el número de posiciones decimales utilizado para mostrar los datos.

El método ‘writeData’ fuerza la escritura de un dato en la gráfica. Si hemos configurado un fichero de datos almacena el valor en el fichero asociado. A continuación se muestra un bloque de flujograma para un grafico de base numérica y otro de base de tiempos para comparar y ver las diferencias entre ambos.

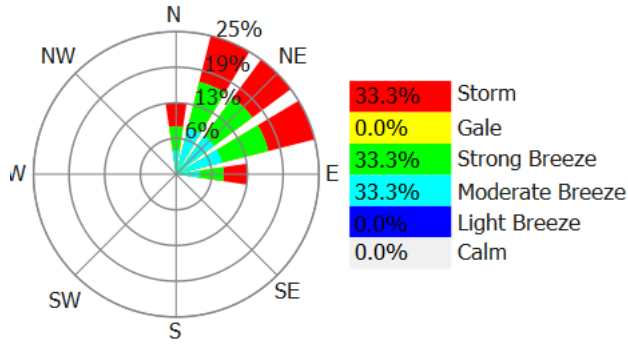


El método ‘clearData’ limpia la gráfica y elimina el fichero de datos asociado.

9.5.-Gráficas ‘rosa de los vientos’.

9.5.1.-Descripción.

El gráfico ‘rosa de los vientos’ ha sido diseñado de forma específica para mostrar la información de velocidades y direcciones del viento.



En los proyectos de ejemplo suministrados con Proteus se encuentra uno llamado ‘Advanced weather station’ que contiene un ejemplo de utilización de gráfica rosa de los vientos que puede ser una buena forma de comprobar el funcionamiento de este tipo de controles.

9.5.2.-Configuración.

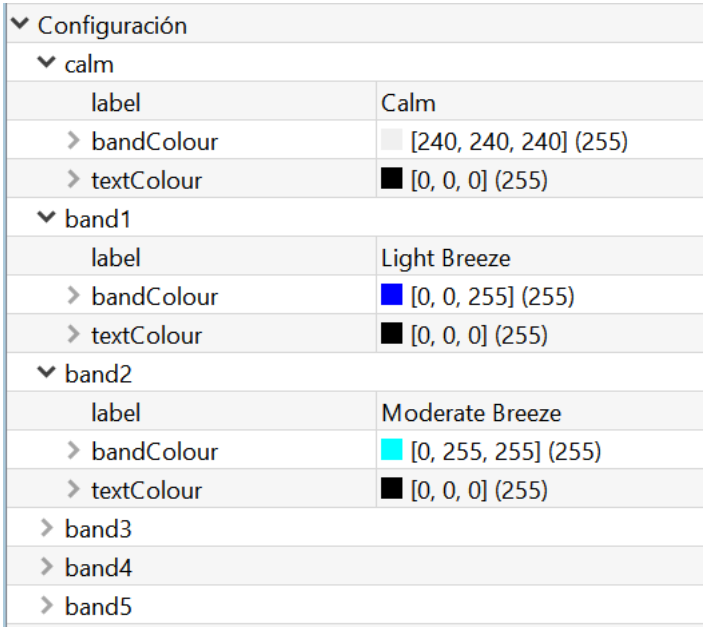
9.5.2.1.-Propiedades de los ‘pétalos’

La rosa de los vientos utiliza ‘pétalos’ para mostrar la dirección y fuerza del viento. Puesto que la ausencia de viento (denominada normalmente como ‘calma’) no tiene dirección se muestra dentro del círculo situado en la zona central.

Cada rango de velocidades del viento se muestra utilizando un color diferente, colocando los vientos más fuertes en los círculos más externos y los más débiles en la zona más interna.

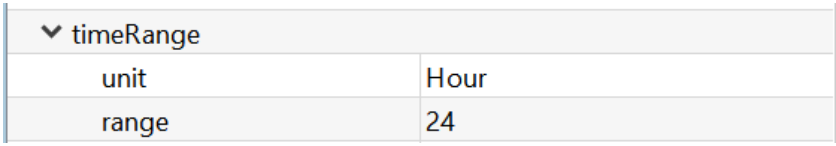
El usuario puede configurar los colores utilizados para cada pétalo editando su banda y el nombre utilizado para cada una de ellas.

Puesto que no siempre el texto en negro se puede visualizar bien sobre todos los colores también es posible cambiar el color del texto de cada banda de forma individual.



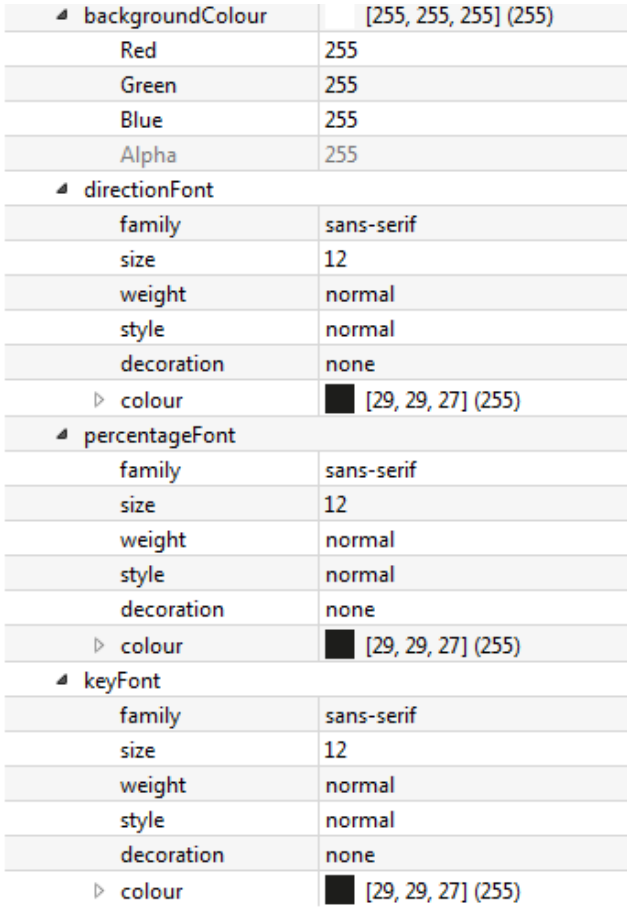
9.5.2.2.-propiedades relacionadas con el tiempo medido.

La propiedad ‘timeRange’ nos permite configurar qué cantidad de datos se muestra en la gráfica. En este ejemplo se muestra datos de un día (24 horas). La propiedad ‘unit’ tiene cuatro opciones: minutos, horas, días o meses. La propiedad ‘range’ nos permite introducir la cantidad de unidades de ese tipo que se mostrará en la gráfica.



9.5.2.3.-Propiedades de color y tipografía.

Estas propiedades permiten configurar el aspecto estético de la gráfica definiendo colores y fuentes.



backgroundColour	[255, 255, 255] (255)
Red	255
Green	255
Blue	255
Alpha	255
directionFont	
family	sans-serif
size	12
weight	normal
style	normal
decoration	none
colour	[29, 29, 27] (255)
percentageFont	
family	sans-serif
size	12
weight	normal
style	normal
decoration	none
colour	[29, 29, 27] (255)
keyFont	
family	sans-serif
size	12
weight	normal
style	normal
decoration	none
colour	[29, 29, 27] (255)

9.5.2.4.-Propiedades del archivo de datos.

La propiedad ‘filename’ sirve para darle un nombre al fichero donde se almacenarán los datos. No podemos ver o acceder a los datos directamente, pero el uso de nombres diferentes permite tener más de una gráfica del tipo ‘rosa de los vientos’ en nuestro cuadro de

mando remoto y que cada una de ellas guarde sus datos en un fichero distinto.

9.5.2.5.-Propiedad 'bands'.

Con esta propiedad fijamos el rango de velocidades de viento que comprendidos en cada banda. En el ejemplo mostrado, hasta 4 se considera calma, de 5 a 13 sería la segunda banda, de 14 a 25 la tercera banda, de 26 a 39 la cuarta banda y de 40 a 55 la quinta banda.

filename	windrose.dat
bands	4,13,25,39,55

9.5.3.-Métodos de control.

> En los proyectos de ejemplo suministrados con Proteus se encuentra uno llamado 'Advanced weather station' que contiene un ejemplo de utilización de gráfica rosa de los vientos es la mejor forma de aprender a usar los métodos de control asociados.

El método 'setTimeRange' sirve para fijar el rango de tiempos mostrados.

El método 'StoreReading' almacena los datos de fuerza y dirección del viento en una matriz temporal. Los datos almacenados en esta matriz no se incorporarán al fichero de datos principal hasta que se utilice el método 'StoreRecord'

El método 'StoreRecord' traspasa los datos de la matriz temporal al fichero de datos principal y los 'estampa', es decir que les asigna una marca que fija la hora de lectura.

El método 'clearData' borra todos los datos y deja la gráfica vacía sin mostrar nada.

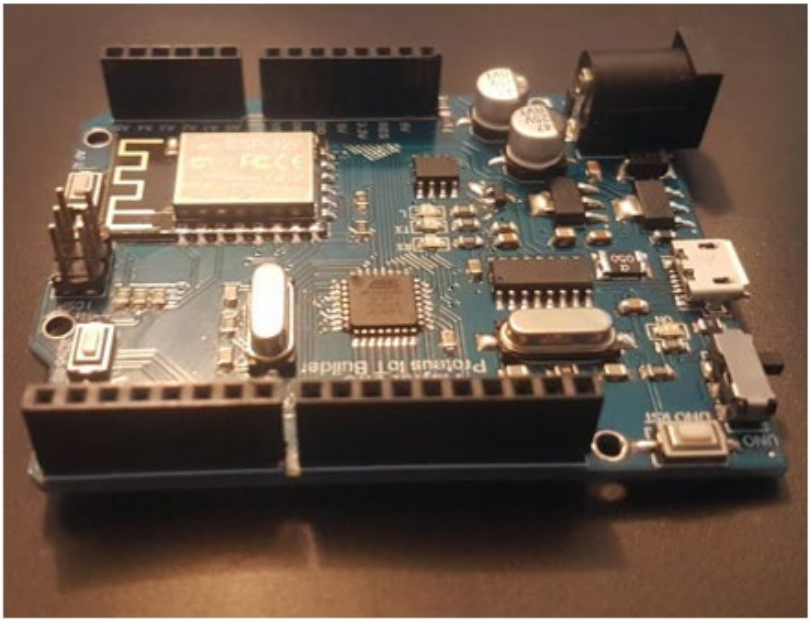
El método 'reload' actualiza la gráfica.

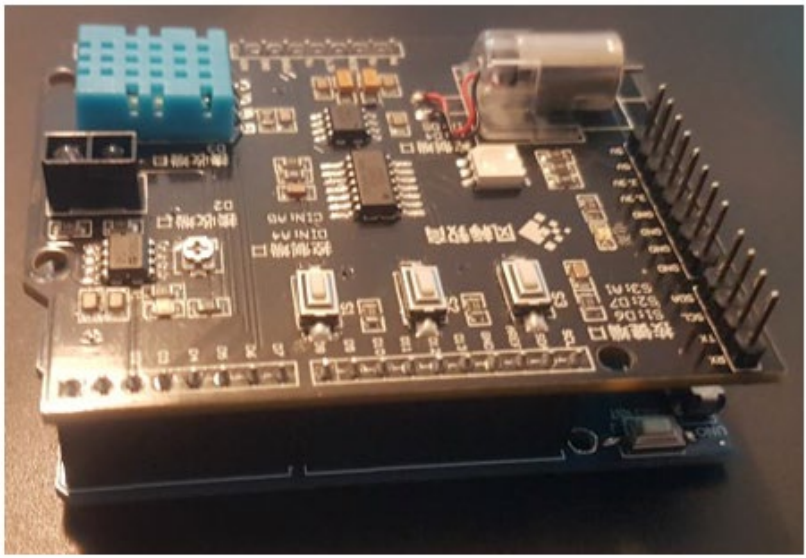
10.-Proteus Playkit

10.1.-Introducción.

Proteus Playkit es un equipo de hardware diseñado específicamente para ser utilizado con el taller IoT y la pestaña Diseñador Gráfico. Ha sido diseñado específicamente para ofrecer al sector educativo una plataforma de hardware a un precio asequible, que utiliza módulos enchufables, destinada a que los estudiantes puedan disfrutar de una rampa de aprendizaje con la herramienta Proteus muy rápida.

El Playkit completo está compuesto por dos tarjetas: una placa base con el controlador y una placa de periféricos que se enchufa encima de la primera.





10.1.1.-La placa base.

La placa base es, básicamente, un controlador Arduino Uno con un microprocesador ESP8266 para las comunicaciones WIFI conectado mediante la UART (transmisor receptor asíncrono universal).

Sus principales características son:

- Formato (tamaño y pines de los enchufes) según el estándar Arduino UNO.
- Microprocesador ATmega328P.
- Totalmente compatible con los módulos de expansión y componentes periféricos tipo Grove.
- Microprocesador ESP8266 pre-programado para conectarlo vía WIFI y listo para usarlo con el Taller IoT de Proteus.
- Modos de trabajo seleccionables mediante micro-interruptores para elegir entre modo UNO o modo Taller IoT.

10.1.2.-La placa de periféricos.

La placa de periféricos es una placa de extensión estándar Arduino que permite de forma sencilla la utiliza de módulos enchufables pre-conectaos existentes en el mercado.

Sus principales características son:

- 3 pulsadores integrados.
- Led tricolor.
- Detector de humedad DHT11.
- Motor DC.
- Detector de infrarrojos.

Las principales ventajas de usar el módulo Proteus Playkit sobre otras alternativas Arduino presentes en el mercado son su bajo coste y completa compatibilidad. Tanto el Arduino Yun como el Seeduino Cloud usan el ATmega32U4 que tienen tarjetas de expansión de periféricos diferentes a la que usa el ATmega328P, por lo que los módulos diseñados para Aduino Uno no funcionan con estos equipos.

11.-Control de acceso.

11.1.-Introducción.

El taller IoT incluye la capacidad de configurar y utilizar diferentes niveles de acceso al panel de control, permitiendo una funcionalidad mayor para aquellos usuarios que se han identificado al sistema de forma positiva.

Para habilitar el control de acceso hay que llevar a cabo tres pasos para configurar nuestro panel de control inicial.

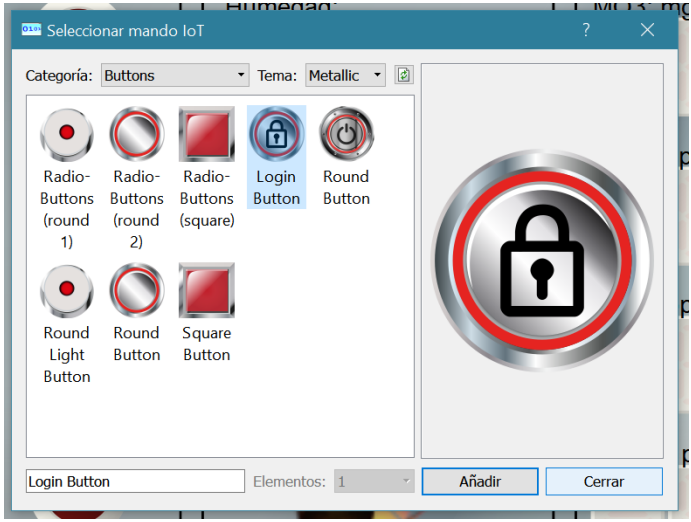
1. Añadir el mando de tipo ‘login button’ y configurar la clave de acceso.
2. Definir el comportamiento del resto de los mandos cuando el usuario no se ha autentificado correctamente.

3. Gestionar los eventos ‘login’ y ‘logout’ desde nuestro programa.

11.2.-El mando login button.

Configuraremos este mando siguiendo estos pasos:

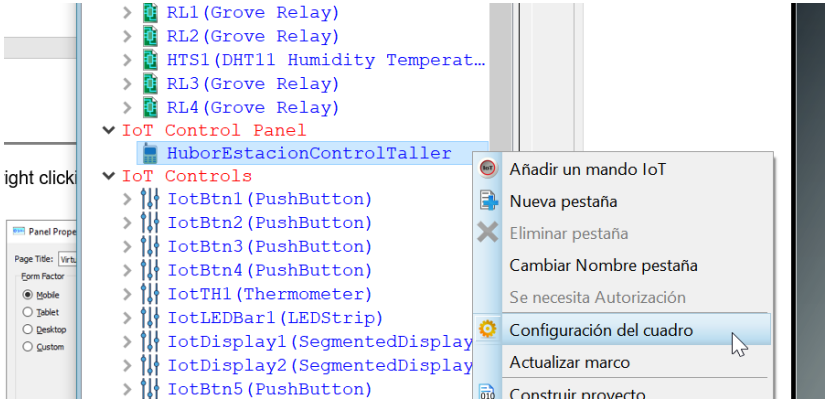
- Añadir un mando del tipo ‘login button’ desde la ventana de selección de mandos IoT. Para abrirla usaremos la opción ‘añadir un mando IoT’ del menú ‘Proyecto’. Lógicamente para ello previamente tenemos que tener al menos un panel de mando creado.

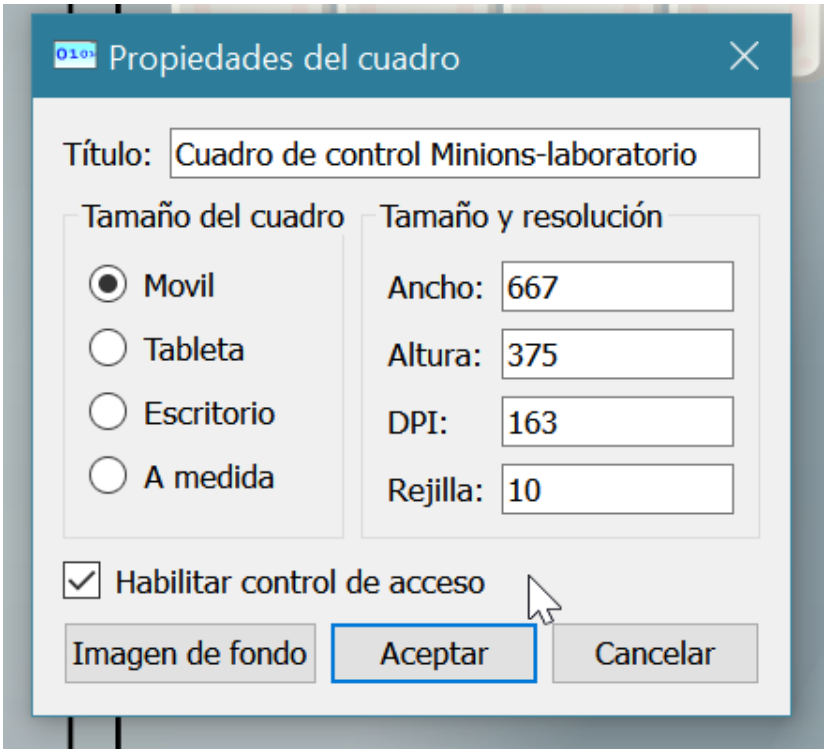


- Colocar el mando tipo ‘login button’ en el panel de mando, seleccionarlo y definir la clave de usuario que deseamos utilizar desde la ventana de edición de sus propiedades.

Propiedades - Iotlogin1	
Geometry	
x	-110,00
y	10,00
width	51,00
height	51,00
Configuración	
password	password123
loginText	Identificarse
logoutText	Salir
loginPrompt	Introduzca la clave de usuario
logoutPrompt	¿Desea salir?
errorPrompt	La clave es incorrecta
labelPosition	Below
labelSpacing	10
labelFont	
family	sans-serif
size	16

- Activar el control de accesos en nuestra aplicación. Lo haremos usando el botón derecho del ratón sobre el panel de mando en el árbol de estructura del proyecto y seleccionando la opción panel -> configuración del cuadro -> habilitar control de acceso.





11.3.-Funcionamiento del control de accesos.

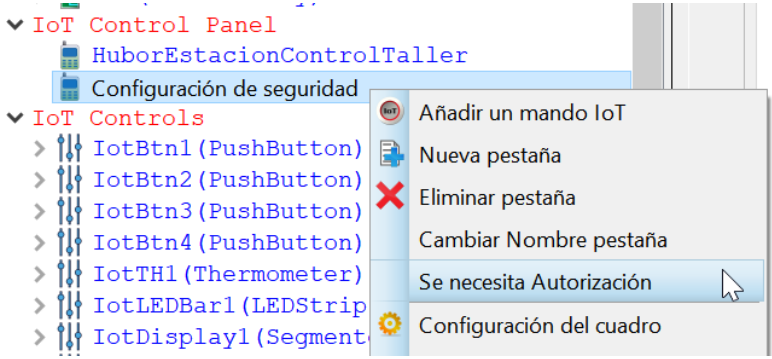
Una que hemos definido que vamos a habilitar el control de acceso necesitamos definir el comportamiento de las distintas pestañas y mandos que componen nuestro panel de mando.

11.3.1-Pestañas.

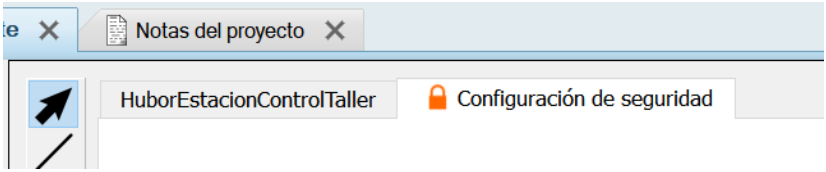
Una pestaña requiere que el usuario se haya identificado positivamente en el sistema para ser visible. En caso contrario permanecerá oculta incluyendo todos los mandos que hayamos ubicado en ella.

Para habilitar la necesidad de requerir autentificación en una pestaña tenemos que seleccionar la pestaña en cuestión en el árbol de

estructura del proyecto, utilizar el botón derecho del ratón para abrir el menú contextual y elegir la opción ‘se necesita autorización’



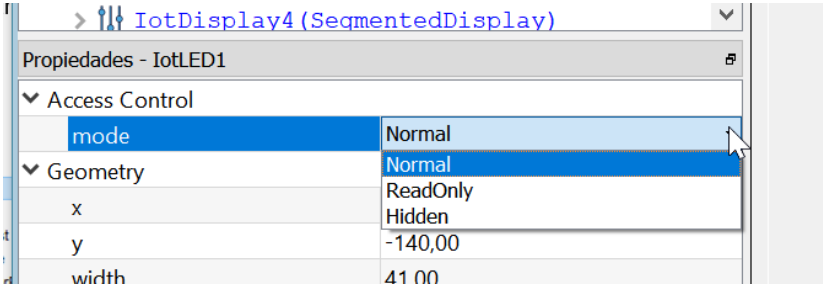
La pestaña aparece con un icono con candado para facilitar el reconocimiento visual de las pestañas que necesitan autorización para acceder a ellas.



11.3.2.-Mandos.

Todos los mandos tienen un comportamiento estándar por defecto cuando el usuario no se ha identificado positivamente. Esto, en la práctica, significa que todos son visibles y activos.

Podemos cambiar el comportamiento de cada uno de ellos desactivándolos u ocultándolos utilizando la propiedad ‘mode’ de la sección ‘Access Control’ en el panel de propiedades.

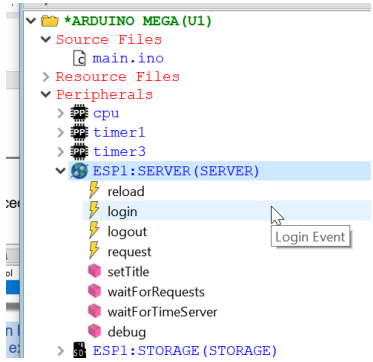


Si elegimos ‘Normal’ el mando será visible y activo. Si elegimos ‘Read Only’ el mando será visible pero no activo. Y si elegimos ‘Hidden’ el mando no será visible y, lógicamente, tampoco activo.

Esta es la situación en que el mando se encontrará al abrir la aplicación y antes de que el usuario se hay identificado positivamente. Una vez que el usuario se haya identificado positivamente el mando estará visible y activo.

11.3.3.-Escribiendo el código.

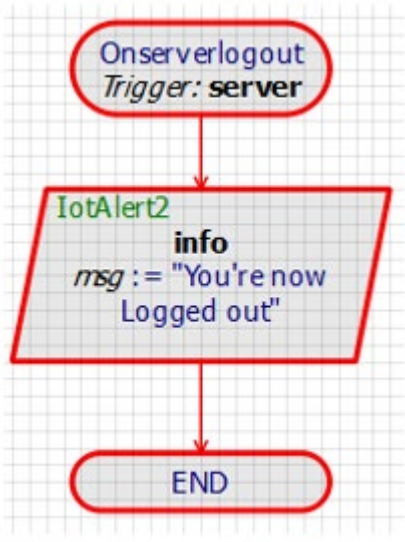
Una vez que el panel de mando está configurado ya tenemos un sistema listo y funcional. Los usuarios se pueden identificar utilizando el mando ‘login button’ para introducir la clave y las propiedades del panel de mando estarán disponibles según el nivel de autorización. De todas formas, muchas veces es útil que podamos controlar mediante código cuando el usuario se ha conectado o desconectado. Podemos hacerlo utilizando dos eventos de servidor ‘login’ y ‘logout’.



Podemos ‘coger y arrastrar’ estos eventos hasta el flujograma de la manera habitual. El evento ‘login’ se activa automáticamente y se ejecuta cada vez que el usuario se identifica en el sistema. De la misma manera el evento ‘logout’ se ejecuta cuando se desconecta del sistema.

Para saber si un usuario se ha identificado positivamente o no, podemos utilizar una variable (por ejemplo, isAdmin) que se inicializa al valor ‘false’ en la subrutina ‘setup’, se pone a ‘true’ con el evento ‘login’ y se vuelve a poner a ‘false’ con el evento ‘logut’.

O, simplemente, podemos mostrar un mensaje para avisar al usuario que se ha identificado correctamente añadiendo un módulo específico.



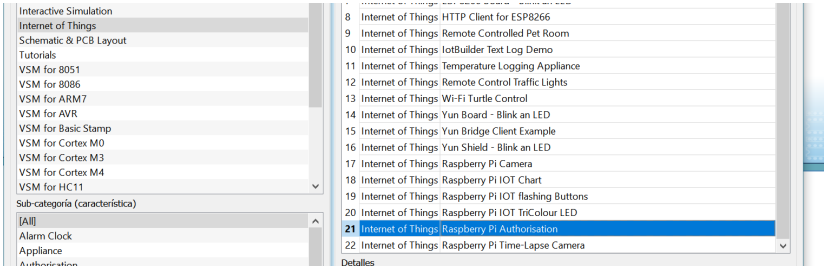
11.4.-Ejecutando la simulación.

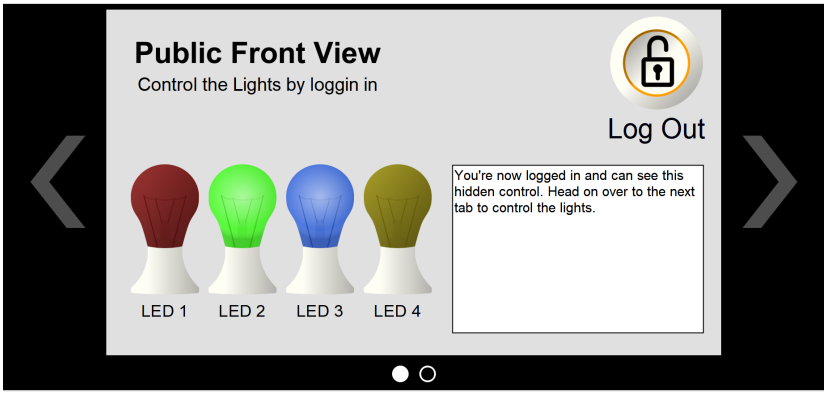
Podemos utilizar el mando ‘Play’ para ejecutar la simulación. Tendríamos que ver aquellas pestañas que hayamos autorizado a ver sin necesidad de que el usuario se haya identificado y no poder ver las que necesiten de identificación positiva del usuario.

De la misma forma, los mandos con necesidad de identificación positiva del usuario estarán ‘inactivos’ o no se verán en función de su configuración.

Si pulsamos sobre el mando de identificación se abrirá una ventana emergente donde podemos introducir la clave de acceso. Todas las pestañas ocultas y mandos ocultos se visualizarán. De la misma forma, todos los mandos inactivos pasarán a activos.

Está disponible un proyecto de ejemplo en la categoría ‘Internet of things’ con el nombre ‘Raspberry Pi Authorisation’ para poder aprender a utilizar esta funcionalidad.





--Página dejada intencionadamente en blanco para sus anotaciones—