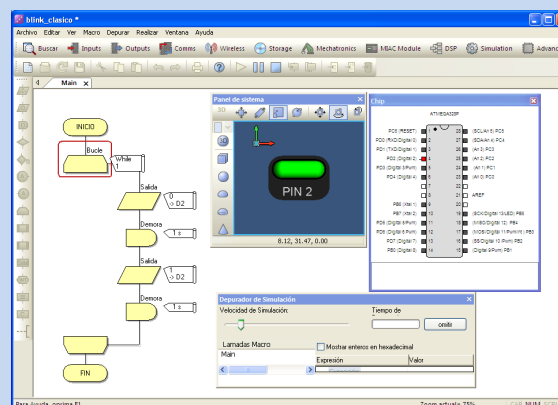
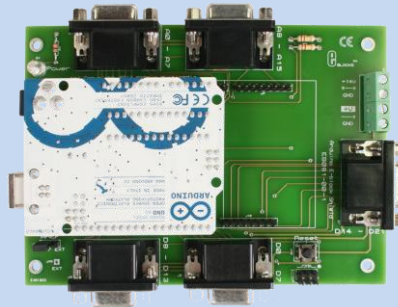


Flowcode + Arduino



Simulación y programación de aplicaciones con
Arduino, E-Blocks y Flowcode 6



Prof. José Manuel Ruiz Gutiérrez
Noviembre 2013

Índice

1. Introducción
2. Descripción del módulo EB081
3. Configuración de la tarjeta Arduino para ser reconocida por Flowcode.
4. Blink
5. Blink con Macro
6. Blink Tiempo variable
7. Botón
8. Alarma básica
9. Monitorización de funcionamiento con Alarma
10. Ejemplo Contador
11. Contador de impulsos de entrada
12. Función lógica AND
13. Salida PWM
14. Control de un motor con tres velocidades
15. Termostato

1. Introducción

Este manual pretende explicar las posibilidades didácticas y metodológicas del software [Flowcode 6](#) de la empresa [Matrix Multimedia](#) para la programación de la Plataforma Open Hardware [Arduino](#) en sus diversas formas de presentación y con los diversos microcontroladores PIC de la familia [Atmega](#).

La herramienta **Flowcode** lleva en el mercado los suficientes años para haberse convertido en un referente mundial dentro del grupo de Herramientas para la Programación de PICs. La oferta de versiones es muy amplia y el fabricante ha liberado versiones que pueden trabajar con **Arduino** manteniendo restricciones con otras PICs pero que son suficientes para poder trabajar en el ámbito educativo.

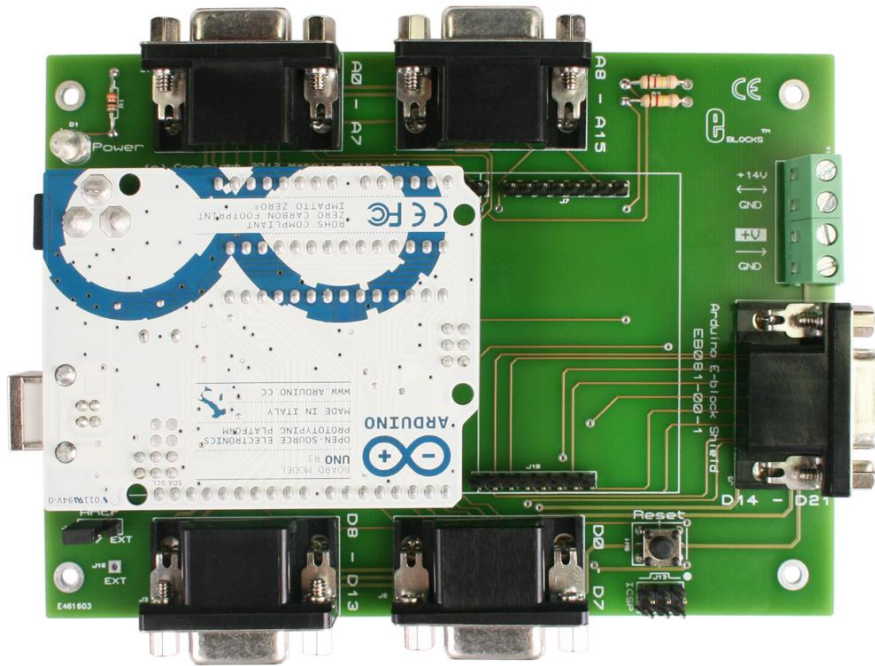


La versión con la que voy a realizar este manual es **Flowcode 6** que representa el último producto en la línea de software. Esta versión es muy adecuada para la docencia y la investigación dado que, a la potencialidad en lo que refiere a la programación gráfica de una PIC, se ha sumando, en este caso, un motor gráfico de simulación que incluye una amplia librería de objetos en 2D y 3D con los que podemos realizar simulaciones muy potentes y atractivas.

He realizado todas las prácticas con la tarjeta **Arduino UNO Rev3** y buena parte de los [E-Blocks](#) que tiene en su catálogo Matrix Multimedia, los cuales me han permitido de manera cómoda y sencilla realizar las simulaciones.

La tarjeta básica E-Block que he utilizado como base es la **EB081** que es un E-Block shield adaptado para recibir sobre él una tarjeta estándar **Arduino** tanto en versión UNO como en versión MEGA.

En este manual no explicaré de manera profunda las posibilidades de la simulación, pero quede dicho que estas son muchas y podrían ser materia para abordar en un segundo manual.



Tarjeta de la serie E-Blocks EB081

Esta tarjeta es muy interesante porque permite conectar a nuestro prototipo todas las tarjetas de la serie E-Blocks con lo que prácticamente sin tener que realizar apenas cableado podemos disponer configuraciones muy diversas.

Las tarjetas que he utilizado y propongo para configurar un kit básico de trabajo son:

EB081 Shield para adaptación de **Arduino** a al sistema E-Blocks

EB003 Modulo de sensores

EB004 Modulo de salidas de LED

EB005 Módulo display

EB007 Módulo de entradas de pulsadores

EB038 Módulo de salida de Relés

EB011 Control de Motores

EB059 Modulo para Servos

EB016 Modulo protoboard para montaje de componentes

2. Descripción del módulo EB081

El módulo **EB081** tiene dispuestos sus conectores “macho” para recibir una tarjeta **Arduino** tipo UNO o tipo MEGA, lo cual es muy sencillo y cómodo para trabajar. Los pines de **Arduino** se extienden a los puertos “conectores” del tipo **E-Blocks** que se indican en la siguiente figura.



Tarjeta **Arduino** conectada sobre EB081

2. Power LED
3. Conector E-blocks port - A0 to A7
4. Conector E-blocks port - A8 to A15
5. Conector E-blocks port - D0 to D7
6. Conector E-blocks port - D8 to D13
7. Conector E-blocks port - D14 to D21

8. Conexión de alimentación
9. Conexiones 5V, 3V3 y VCC
10. Pulsador Reset
11. ICSP header
12. Analógica VREF
13. Conector **Arduino** USB

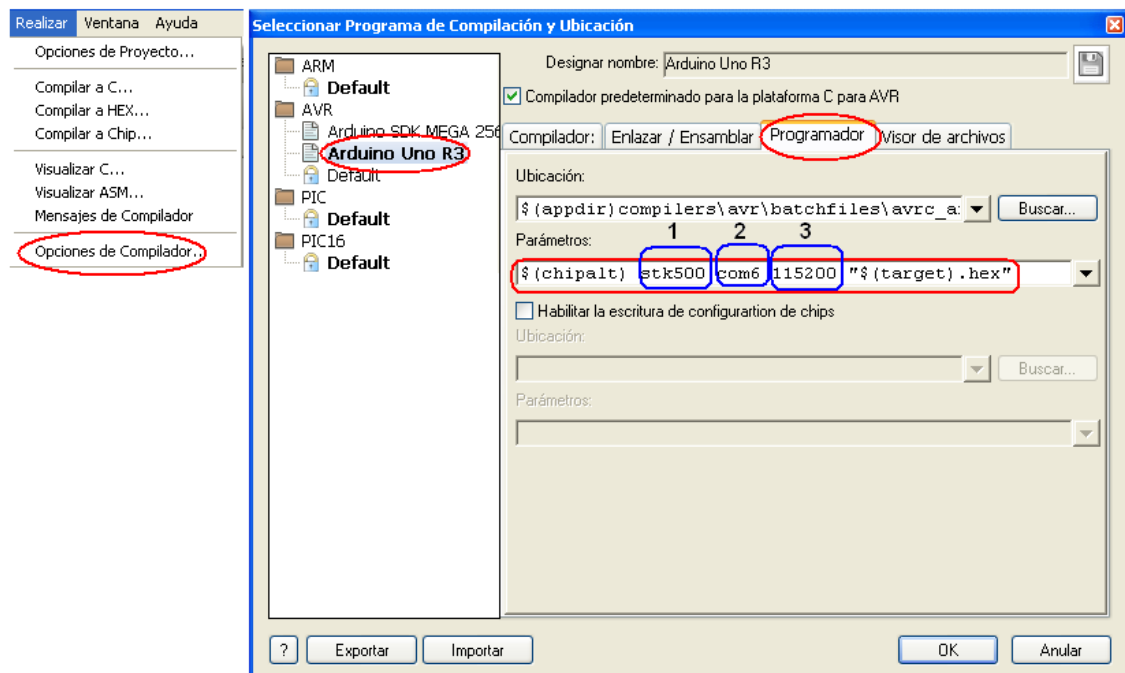
Las correspondencias de los pines de **Arduino** con los conectores son las siguientes:

3. Configuración de la tarjeta Arduino para ser reconocida por Flowcode.

Con el fin de evitar problemas y errores a la hora de realizar la conexión de **Arduino** con **Flowcode** es muy importante que configuremos correctamente los parámetros de nuestra tarjeta para que la compilación y la descarga de los “*bitcodes*” de nuestro programa sobre la PIC de **Arduino** no de ningún problema.

Para realizar la configuración

1. Debemos abrir la opción del menú **Realizar->Opciones de Compilador**



- 1 Representa el tipo de Tarjeta **Arduino**
- 2 Representa el COM por el que nos comunicamos con **Arduino**
- 3 Representa la velocidad de comunicación

2. A continuación ponemos estos parámetros para distintas tarjetas **Arduino**.

Los parámetros para los distintos tipos de **Arduino** son los siguientes:

Tipo de tarjeta	Parámetros	
	(1) Código	(3) Velocidad
Arduino Uno	stk500	115200
Arduino Duemilanove or Nano w/ ATmega328	stk500	57600
Arduino Diecimila, Duemilanove, or Nano w/	stk500	19200
Arduino Mega 2560	stk500v2	115200
Arduino Mega (ATmega1280)	stk500	57600
Arduino Mini	stk500	19200
Arduino Fio	stk500	57600
Arduino BT w/ ATmega328	stk500	19200
Arduino BT w/ ATmega168	stk500	19200
LilyPad Arduino w/ ATmega328	stk500	57600
LilyPad Arduino w/ ATmega168	stk500	19200
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328	stk500	57600
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168	stk500	19200
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328	stk500	57600
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168	stk500	19200
Arduino NG or older w/ ATmega168	stk500	19200
Arduino NG or older w/ ATmega8	stk500	19200

Las etapas básicas para realizar la descarga de la aplicación desde **Flowcode** a **Arduino** son: **Compilar, Ensamblar, Descargar**

Para la ejecución de estas etapas se ejecuta un fichero .bat que contiene las instrucciones para el compilador, ensamblador y modulo de descarga del código que se encuentran en la carpeta .. **compilers\avr\batchfiles** de la aplicación **Flowcode**.

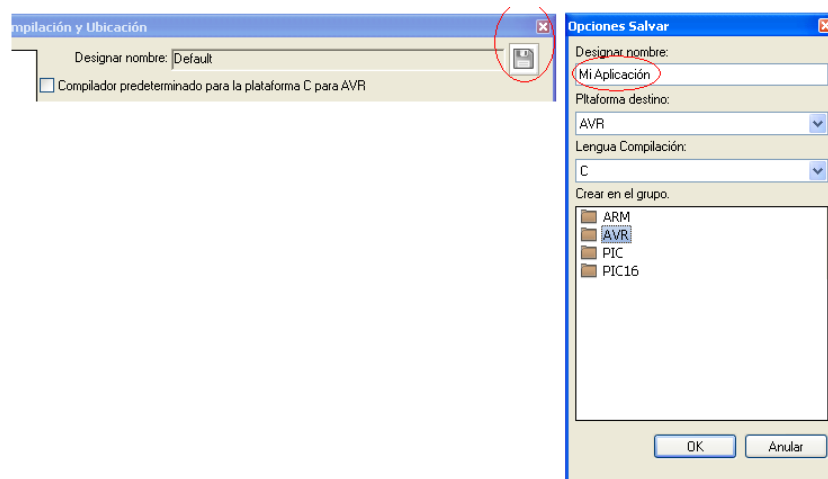
Los ficheros que se ejecutan son:

Para la Compilación: **avra.bat**

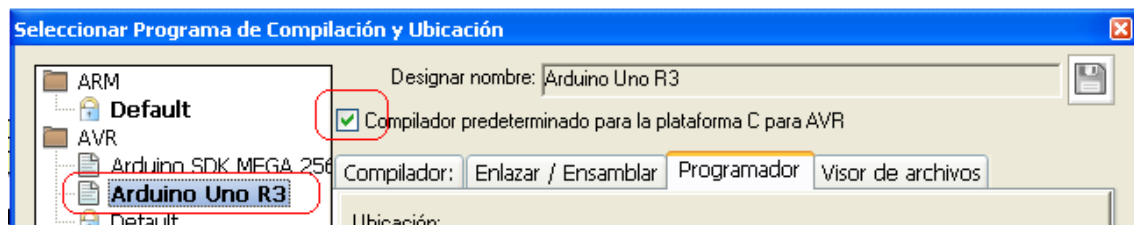
Para el ensamblado: **avrb.bat**

Para la Programación(descarga) del código: **avrc_ArduinoA.bat**

Si nos fijamos bien podremos crear distintas configuraciones que podremos aplicar a distintas tarjetas, bastará pinchar sobre la carpeta AVR y mediante el menú contextual (botón derecho) añadir una nueva configuración que se guardará con ese nombre una vez que nosotros hayamos puesto en cada una de las opciones (compilar, ensamblar y descargar) los parámetros.



Luego podremos definir como “predeterminada la que deseamos tener como tal:



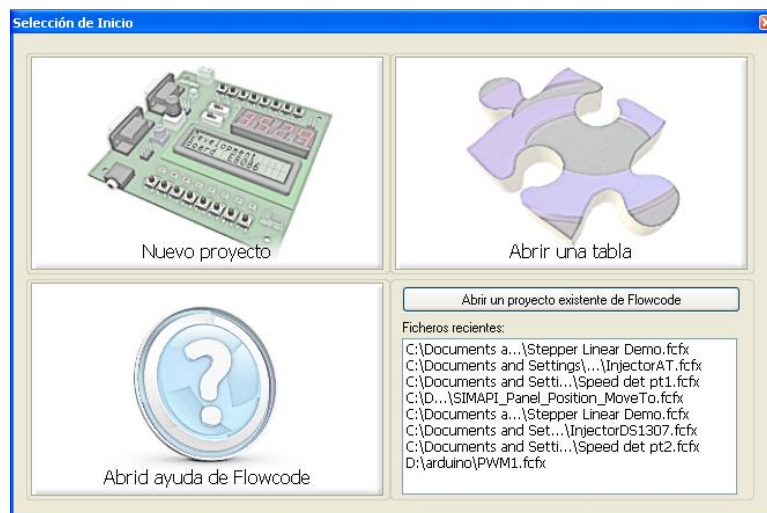
4. Blink

Vamos a realizar nuestra primera aplicación con **Arduino +Flowcode**. Se trata del clásico ejemplo Blink en el que activamos y desactivamos con un tiempo de cadencia determinado una salida digital de **Arduino**.

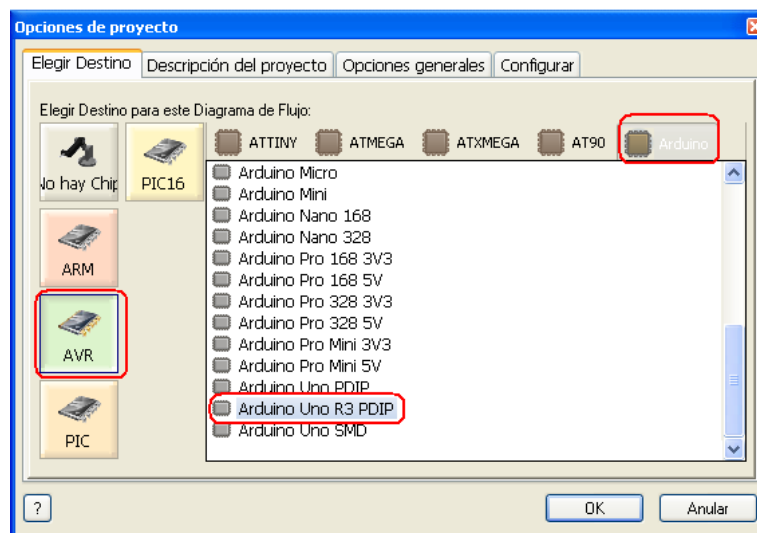
Describiremos en este primer ejemplo de manera más detallada las etapas para realizar la programación, descarga y simulación de este sencillo ejemplo.

Descripción del proceso:

1. Una vez que ejecutamos **Flowcode** se abre la pantalla de inicio y se nos pide que elijamos lo que deseamos hacer. Nosotros seleccionamos “Nuevo proyecto”

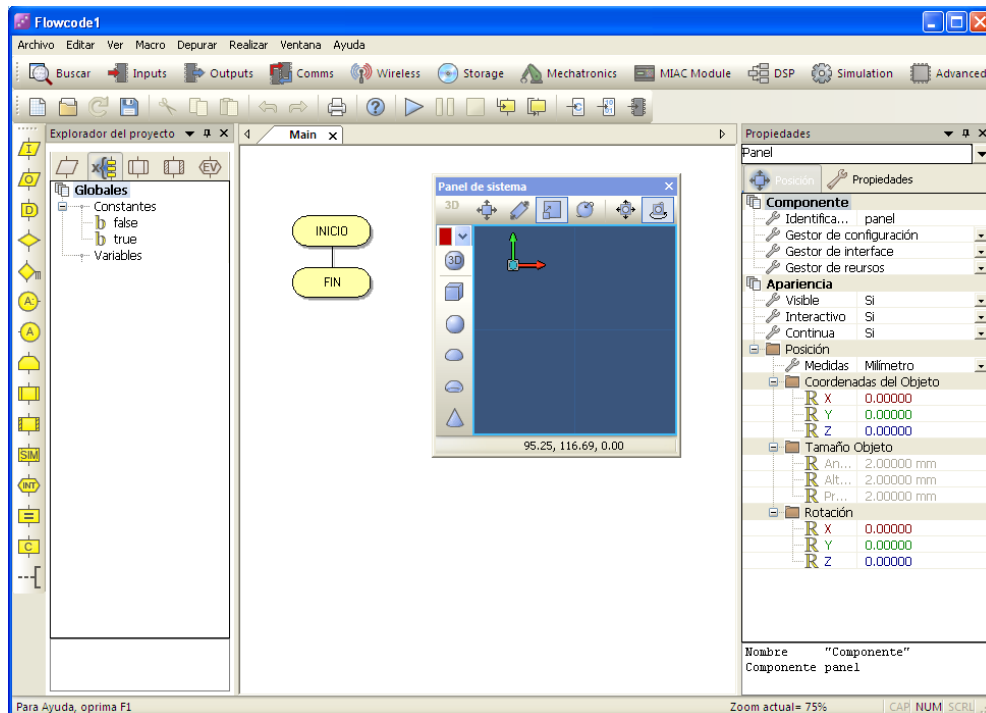


2. A continuación seleccionamos el microcontrolador en la pantalla siguiente

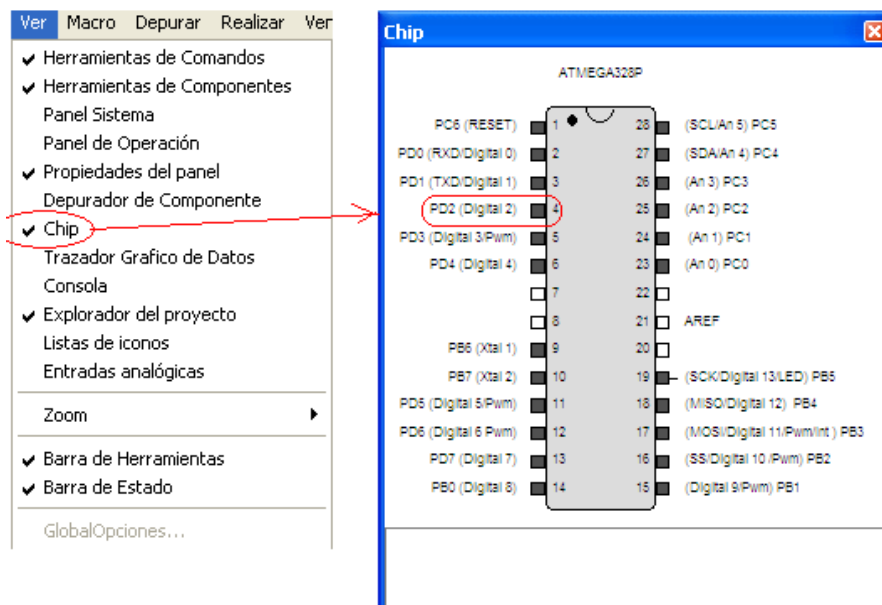


Seguidamente podremos, si lo deseamos, hacer una descripción de nuestro proyecto “Pestaña Descripción de Proyecto” y no tocaremos nada en las otras pestañas.

3. Una vez realizadas estas sencillas opciones se abre la pantalla del entorno y ya estamos en disposición de realizar nuestro trabajo.



Queremos encender y apagar un led que estará conectado al **PIN 2** de **Arduino** que se corresponde con el **PIN D2** en la nomenclatura de **Flowcode**, tal como se indica en la ventana de Pin. Esta ventana se puede hacer visible en la opción Ver.



Queremos que ese pin **PIN Digital 2** se active y desactive cada segundo.

4. El siguiente paso será crear el algoritmo que realice esta función. Para ello vamos colocando los elementos de la barra de bloques de programación de la izquierda en el área de trabajo.

Los bloques que integraremos serán:

Un bloque tipo “**Bucle**”:



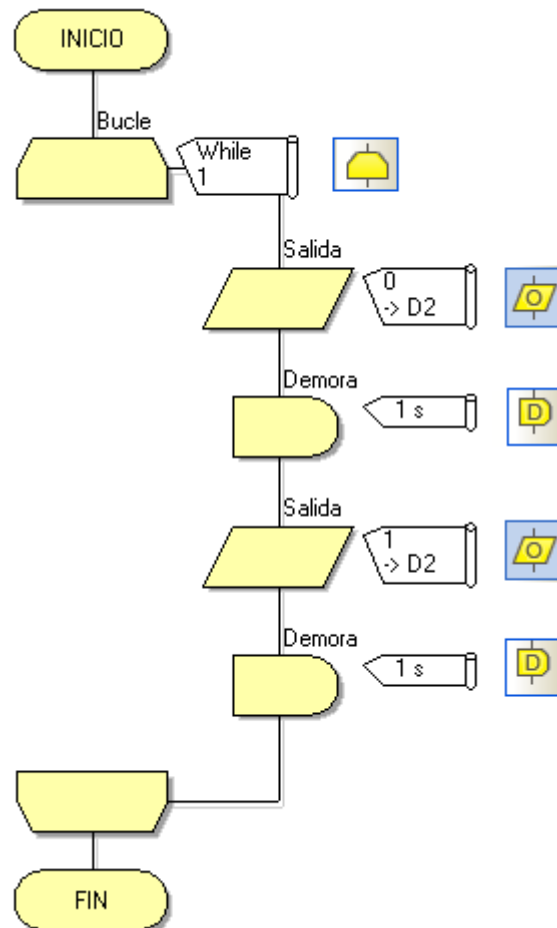
Dos bloques tipo “**Salida**”:



Dos bloques tipo “**Demora**”:

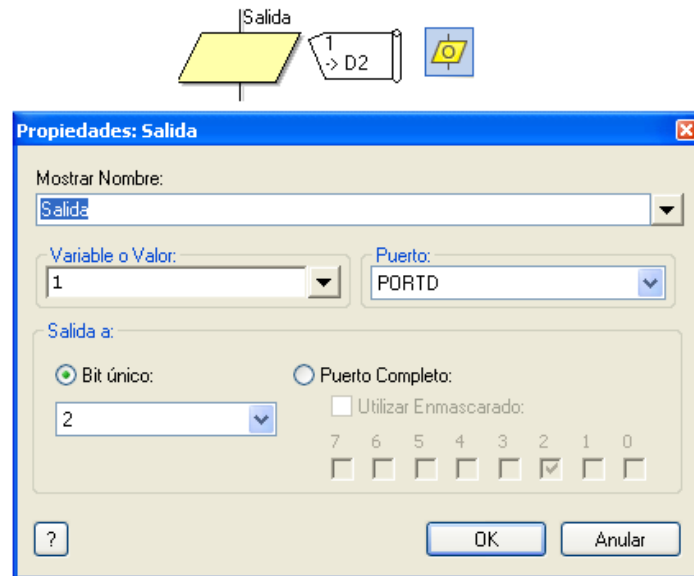


El diagrama de flujo de nuestro algoritmo de la figura. Téngase en cuenta que todo el desarrollo de las funciones esta dentro de una estructura tipo “**Bucle**”.

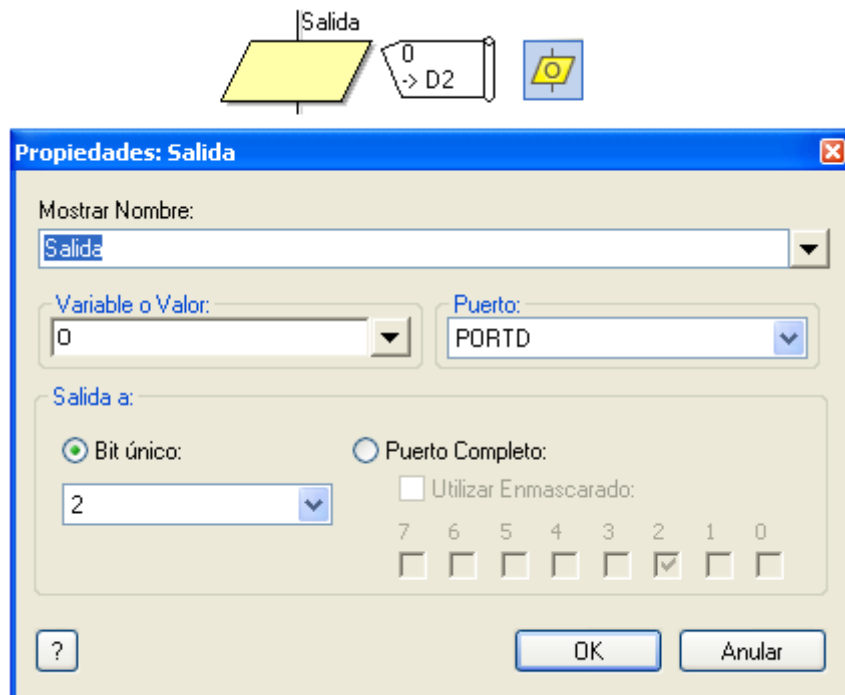


Los bloques tienen que ser parametrizados de acuerdo a la salida que deseamos y con los tiempos de demora correspondientes. A continuación se muestran las imágenes de las ventanas de parámetros de estos bloques:

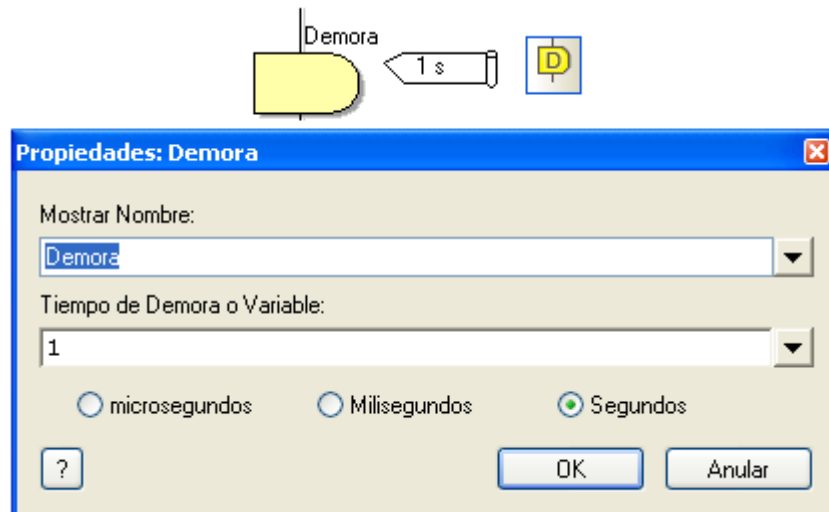
Con este bloque activamos con el valor “1” la salida digital **PIN2** que se corresponde con el puerto **PORTD D2**




Con este bloque activamos con el valor “0” la salida digital **PIN2** que se corresponde con el puerto **PORTD D2**



Las demoras se programaran para un tiempo de 1000 ms

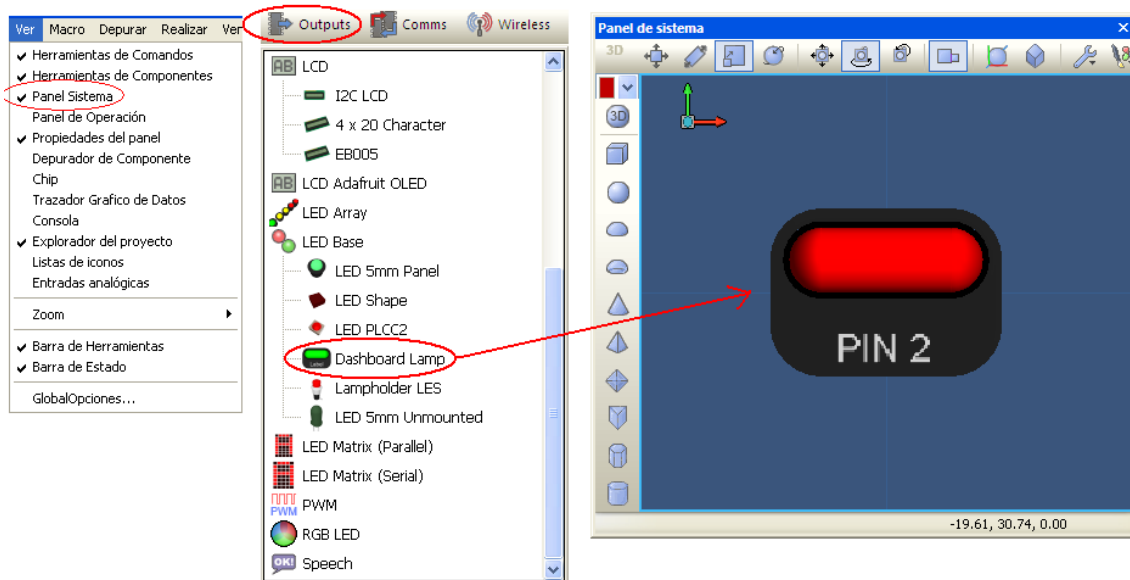


5. El siguiente paso es configurar el panel de Sistema en el que aparecerá un indicador del estado de la salida que hemos programado cuando pongamos el sistema en

modo Simulación. 


Para esta operación seleccionamos en el menú “Ver” la opción “Panel Sistema” y se muestra el panel.

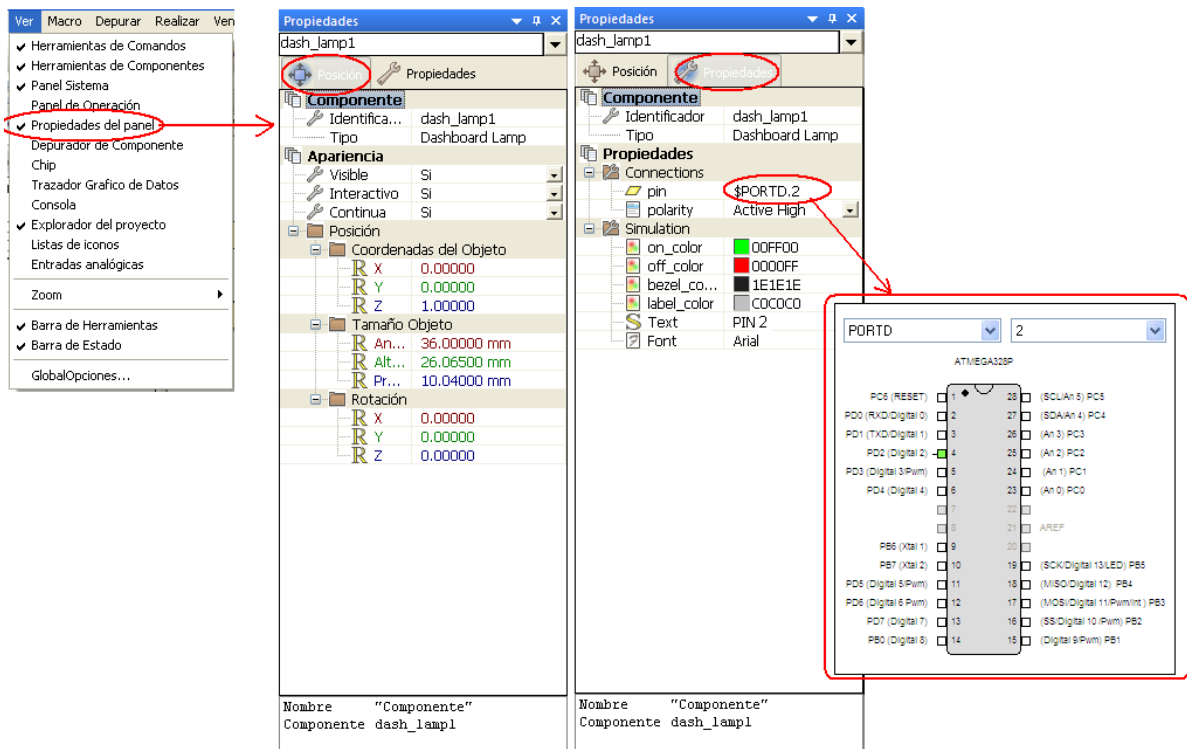
Seguidamente seleccionamos el objeto “Dashboard Lamp” de la librería de elementos “Outputs” y lo arrastramos al panel.



El elemento seleccionado debe ser parametrizado y para ello nos vamos a la ventana propiedades de elemento y allí seleccionamos el puerto y pin de salida.

Bastará para esta operación mostrar la ventana de propiedades mediante la opción **Ver->Propiedades de Panel** y desplegada esa ventana seleccionamos la pestaña de

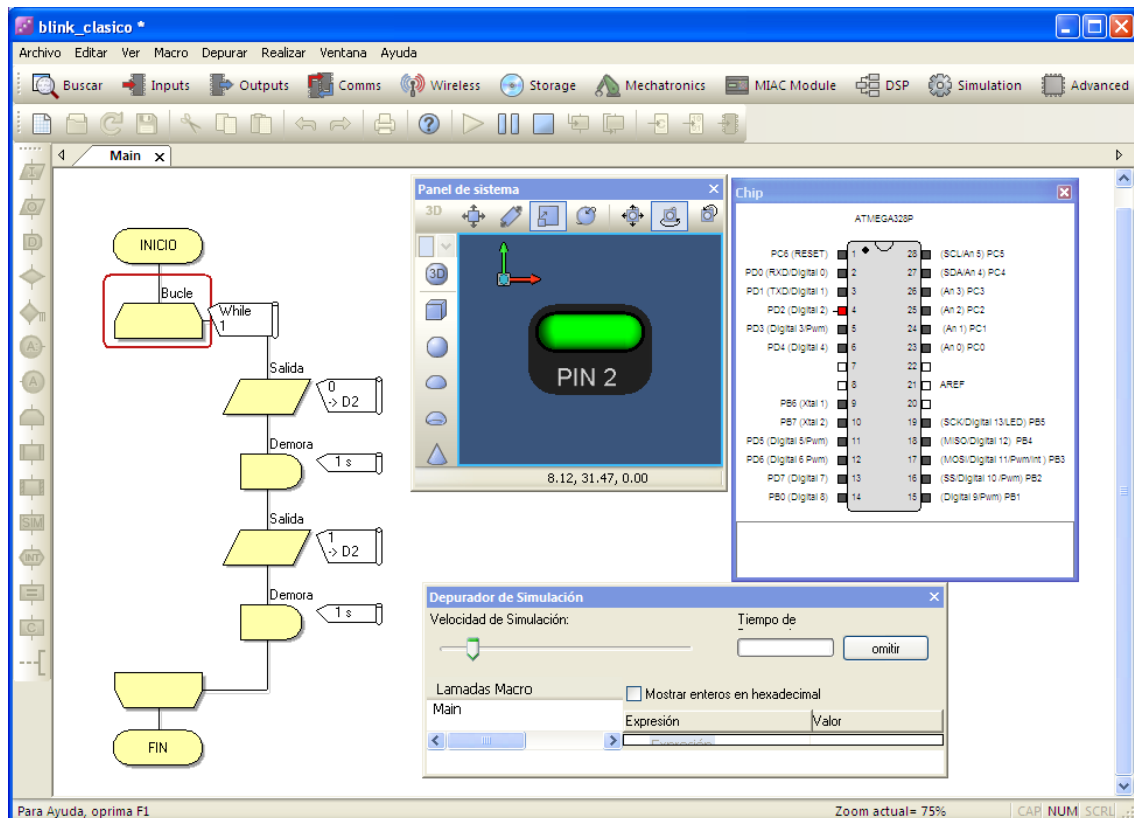
“Propiedades”  y allí pinchando en la parte de *Connection->pin* aparece una ventana en la que se muestra el chip **ATMEGA328P** y seleccionamos el puerto y el pin en las ventanas que aparecen.



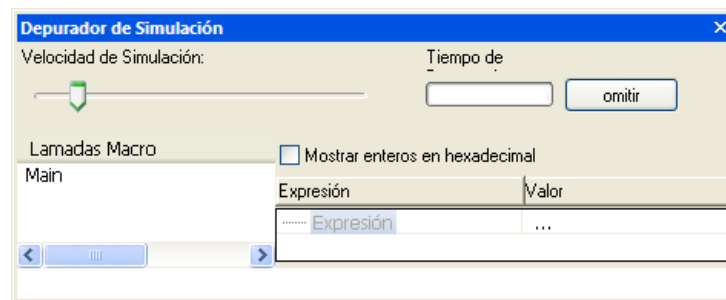
Si todo ha ido bien estaremos en disposición de realizar el test de simulación de nuestro proyecto para lo cual bastará colocar la ventana de **Panel Sistema** y la de **Chip** de forma cómoda y visible y le pulsamos en el botón de simulación.



Si todo ha ido bien veremos que nuestro Led parpadea dentro de la ventana “**Panel de sistema**” y también veremos que el **PIN 2** de la ventana de “**Chip**” se activa y desactiva.

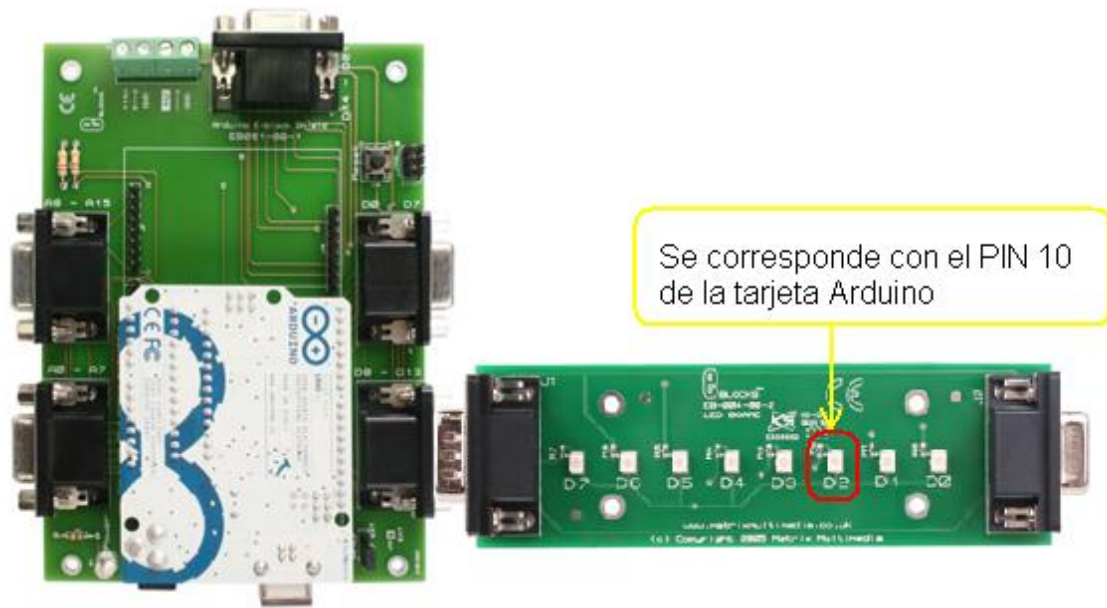


También aparecerá una ventana llamada “**Depurador de Simulación**” que permite, entre otras cosas, modificar la velocidad de simulación. Si hacemos más lenta la simulación podremos ver cómo se van ejecutando las instrucciones en el diagrama de flujo lo cual nos permitirá aprender con facilidad como actúa.



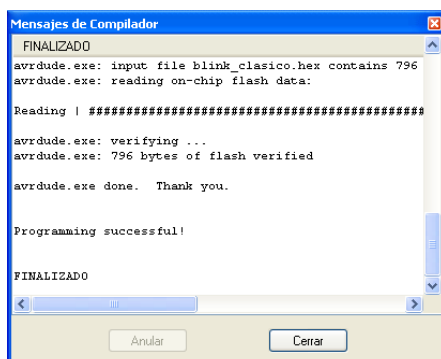
6. Una vez que hemos comprobado el funcionamiento en modo simulación tenemos que realizar la programación física sobre la tarjeta **Arduino** para ello dispondremos las tarjetas **EB081** que es el shield en donde colocaremos Arduino y la tarjeta de salidas tipo Led **EB004** que es la que vemos como se activa realmente el led conectado al **PIN2** de **Arduino** y al **PIN D2** de la tarjeta de led que se insertará en el conector **D0 – D7** del shield **ED081**.

En la imagen siguiente vemos el montaje.



No olvidemos que en este caso la tarjeta de simulación de Leds de salida los diodos D0 y D1 no se utilizarán normalmente porque se ocupan en los canales de comunicación Tx y Rx.

7. La siguiente y última fase es descargar el programa sobre **Arduino**. Para ello se deberá realizar la compilación ensamblado y carga del programa con la ayuda de los botones correspondientes del menú de **Flowcode**.

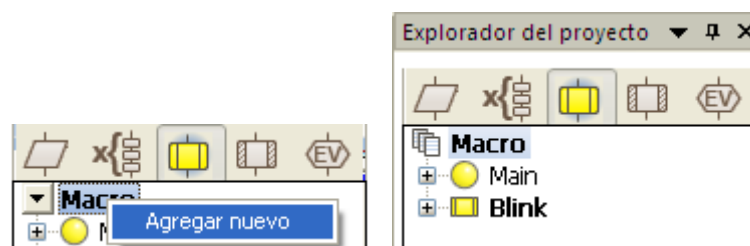


Primero se compila (1), para ello deberemos salvar previamente el diagrama. Después se crea el fichero HEX (hexadecimal) (2) y por último se descarga sobre **Arduino**. Si todo ha ido bien aparecerá una ventana en la que se nos dará cuenta de que todo fue bien.


5. Blink con Macro

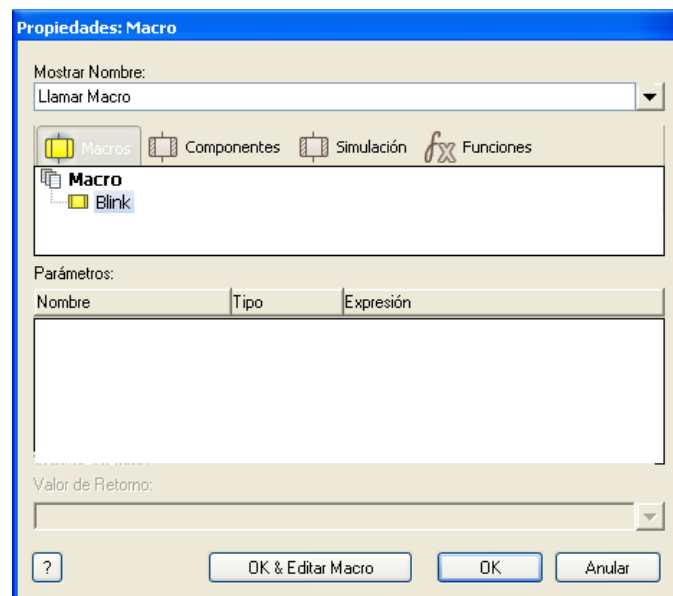
A continuacion vamos a realizar un ejemplo que es igualo que el anterior pero en el que hemos introducido una macro que se encarga de realizar la cinmutacion de la salida con demoras entteela activiacion y deactivacion.

Para crear una macro bastará con ir a la ventana del “Explorador de Proyecto” y en la pestaña de macros pular sobre macro y “Agregar nuevo”. Ponemos el nombre de la macro, y quedará creada automaticamenbte. Luego pinchando sobre el cion de madro “Blink” aparecera en el area de trabajo los dos bloques **INICIO** y **FIN** y colaoramos entte ambos los bloques que realizan el encendido y apagado del **PIN10** de **Arduino** (**PUERTOB 2**)

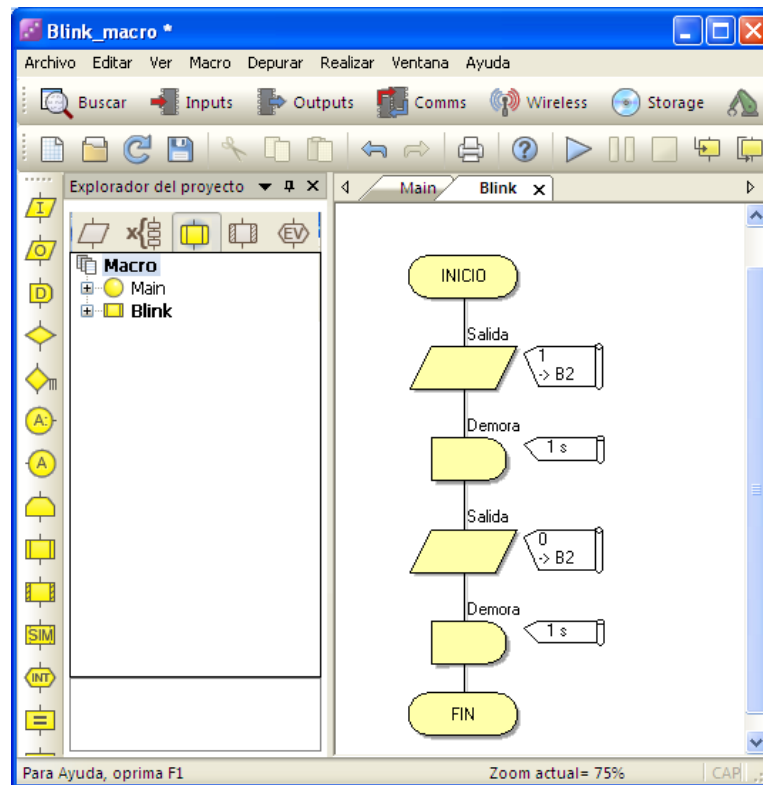


La macro **Blink** se añade en una pestaña del area de trabajo llamada “**Blink**”.

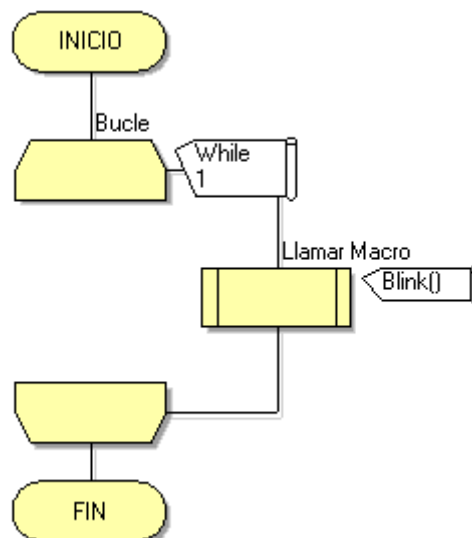
Para ewscribir el programa principal pulsamos sobre la pestaña “**Main**” y escribimos colcoamos el “**Bucle**” y el bloque de funcion “Llamar Macro”  desigando el nombre Blink en la venata de parametros de este bloque



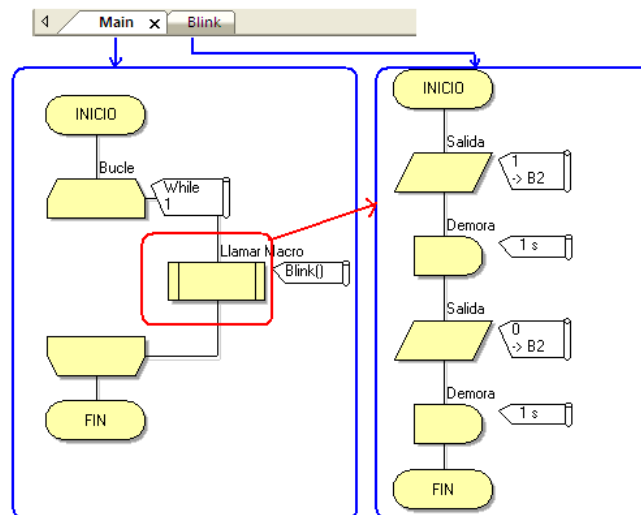
En la siguiente figura vemos el organigrama de la macro Blink una vez termiando de construir



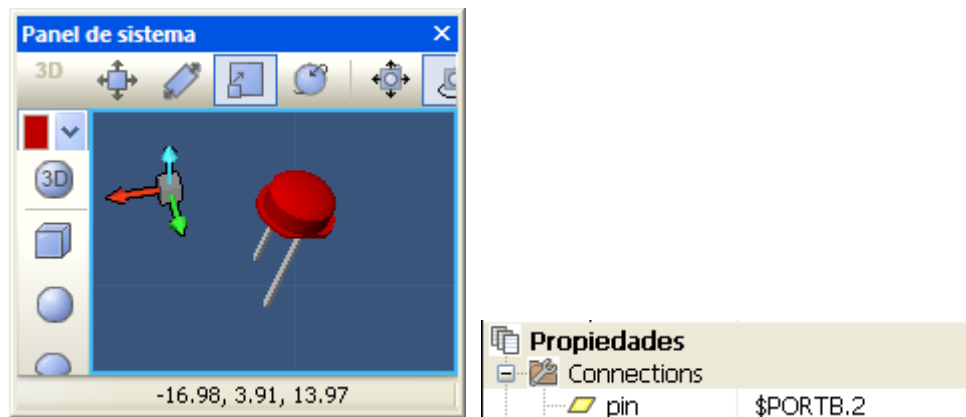
En la siguiente imagen tenemos el organigrama del programa principal “**Main**”. No olvidemos que la activación será en el **PIN10** de **Arduino** a que se corresponde con el **PORTB 2 (PB2)** del Chip **ATMEGA 328**



En la ejecución del programa “**Main**” se realiza la llamada a la macro “**Blink**” tal como se indica en la figura.



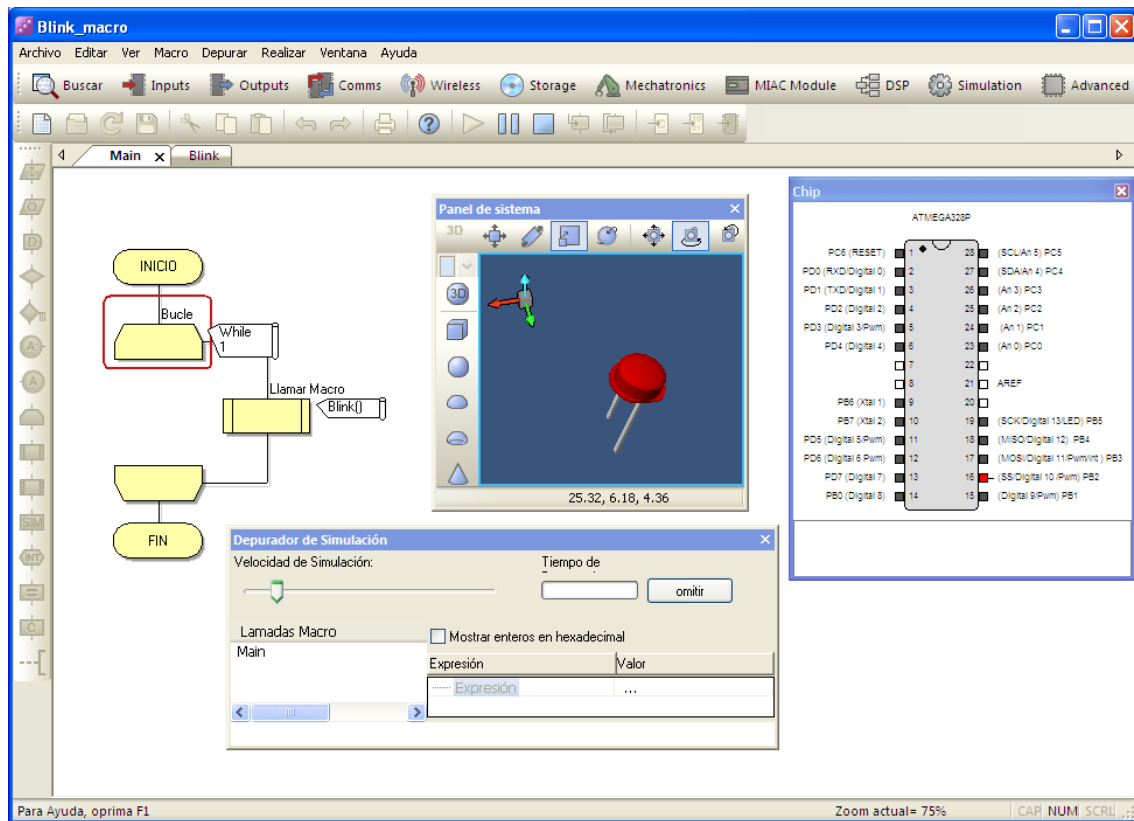
Para construir el “**Panel Sistema**” hemos recurrido a la librería de elementos de salida “**Outputs**” y de ella se ha seleccionado un led del tipo “**LED 5mm Unmounted**” al que en la ventana de propiedades le hemos asociado al **PIN \$PORTB.2**



Una vez realizada esta operación pasamos a simular la aplicación comprobando que funciona tal y como la hemos pensado.

En la siguiente imagen vemos la pantalla en modo simulación.

Cuando se ha probado que está bien nuestro diseño pasamos a las fases de compilar, montar y descargar la aplicación sobre la tarjeta **Arduino** montada sobre la tarjeta **EB081**.



El montaje físico es el que se muestra en la figura siguiente.

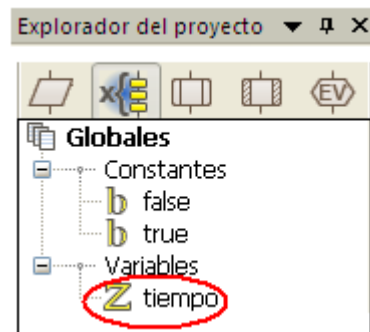


6. Blink Tiempo variable.

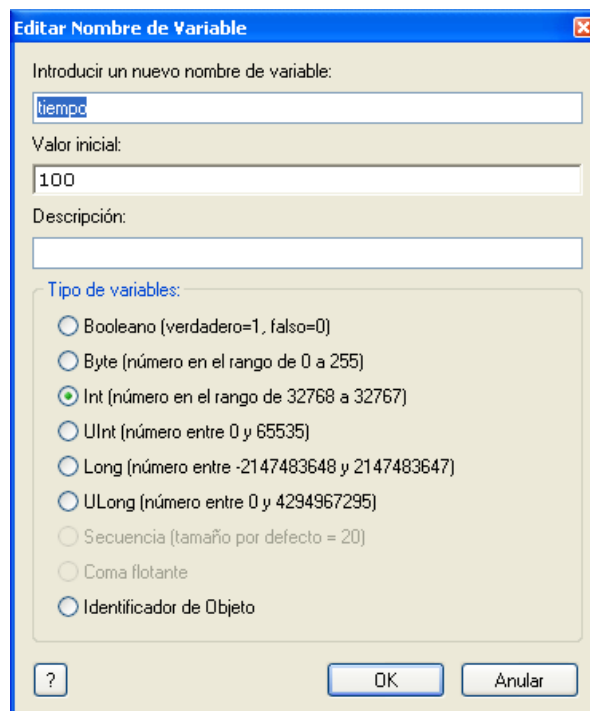
Con esta aplicación vamos a incorporar una lectura de valor analógico de la tarjeta **Arduino** que se tomara como variable de tiempo en el los bloques de demora en el encendido y apagado del led que colocamos en el **PIN10** de **Arduino**.

El valor analógico lo tomaremos del canal **A1** que mediante la tarjeta de simulación **EB003**.

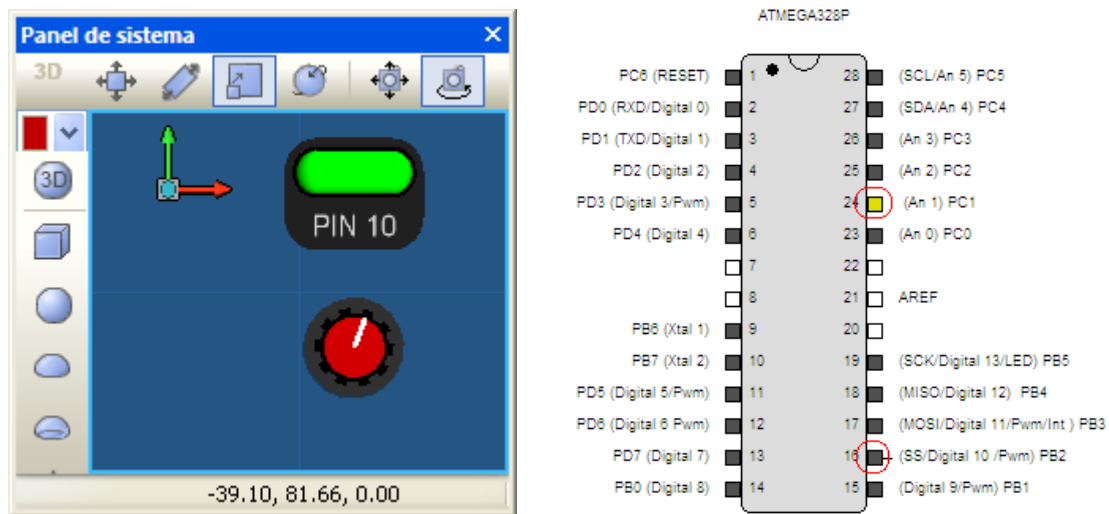
La inclusión de una variable nos permitirá leer y entregar el valor correspondiente. La variable recibirá el nombre de **“Tiempo”**



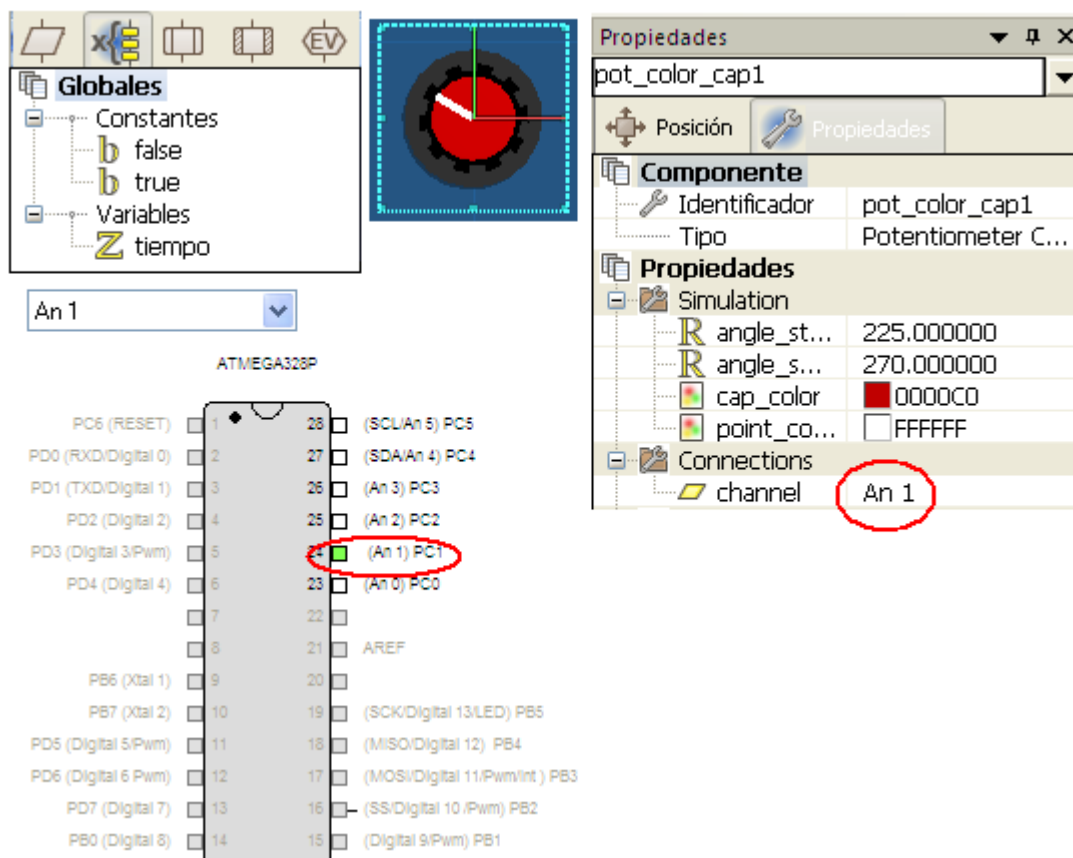
Esa variable tiempo se creará simplemente pulsando el botón derecho del rato estando situados en **“Variables”** seleccionado **“Agrega nuevo”**. Se definirá como una variable de tipo Byte. Se pondrá el valor 100 por defecto. Esta variable recogerá el valor del canal **AN1**



En este caso vamos a construir primero el “**Panel Sistema**” incluyendo en el dos elementos: un LED y un Slider (potenciómetro). Con el primero simularemos el **PIN10** de salida de **Arduino** y con el Slider la variable de entrada analógica **AN1**

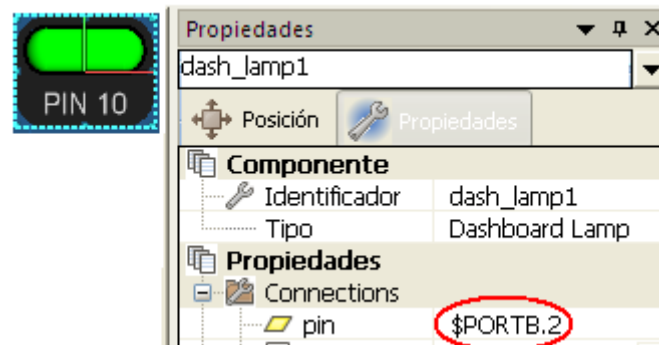


En la ventana de “Propiedades”, cuando seleccionamos el “Slider” aparecen las propiedades de este y seleccionamos en este caso en el apartado “**Connection**” y **Channel** el valor **An1** en el menú que nos aparece de los terminales (puertos) del Chip.



De esta manera queda adscrito el canal An1 al Slider.

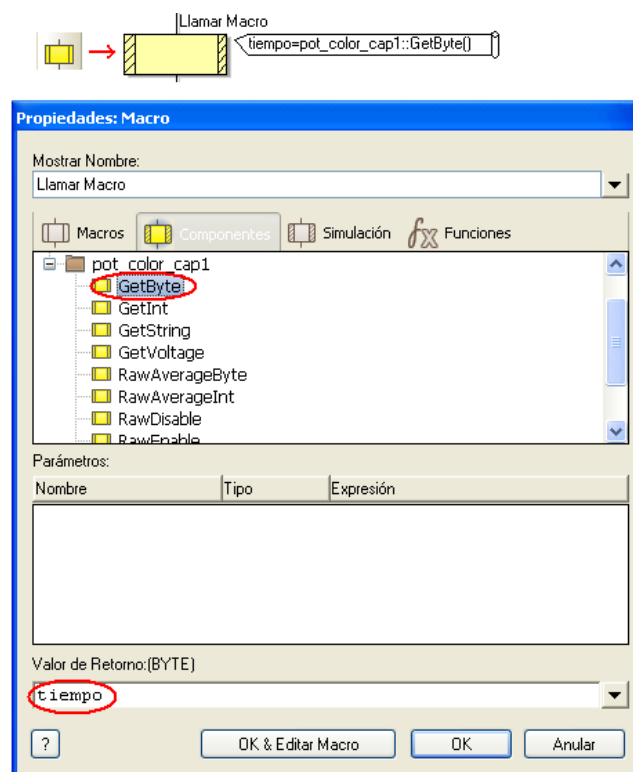
El siguiente elemento es Led



En este caso se asociará al **PIN 10** de **Arduino** que es el **PORTB.2** del Chip de **Flowcode ATMEGA328**.

Realizadas estas operaciones nos dispondremos a construir el diagrama de flujo de nuestra aplicación. Recordemos que todas las funciones se encuentran dentro de una estructura “**Bucle**”.

Pondremos un bloque tipo “**Macro de componente**” en el que podremos realizar la asociación de la variable “**tiempo**” al objeto “**pot_color_cap1**” que es el slider que acabamos de colocar en el “panel Sistema” el cual a su vez esta unido al **PINA1** de **Arduino** An 1 (en el chip **Flowcode**).



Seguidamente pondremos unos tras otros los siguientes bloques:

Salida (poner 0 en PB2)

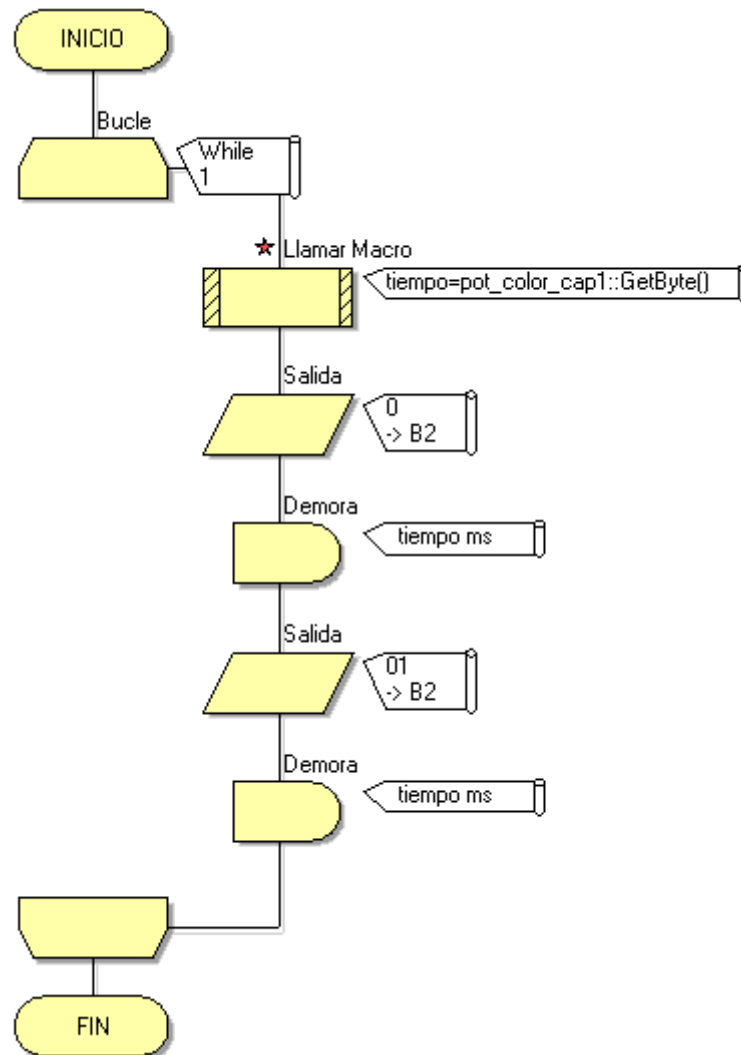
Demora (tiempo ms)

Salida (poner 1 en PB2)

Demora (tiempo ms)

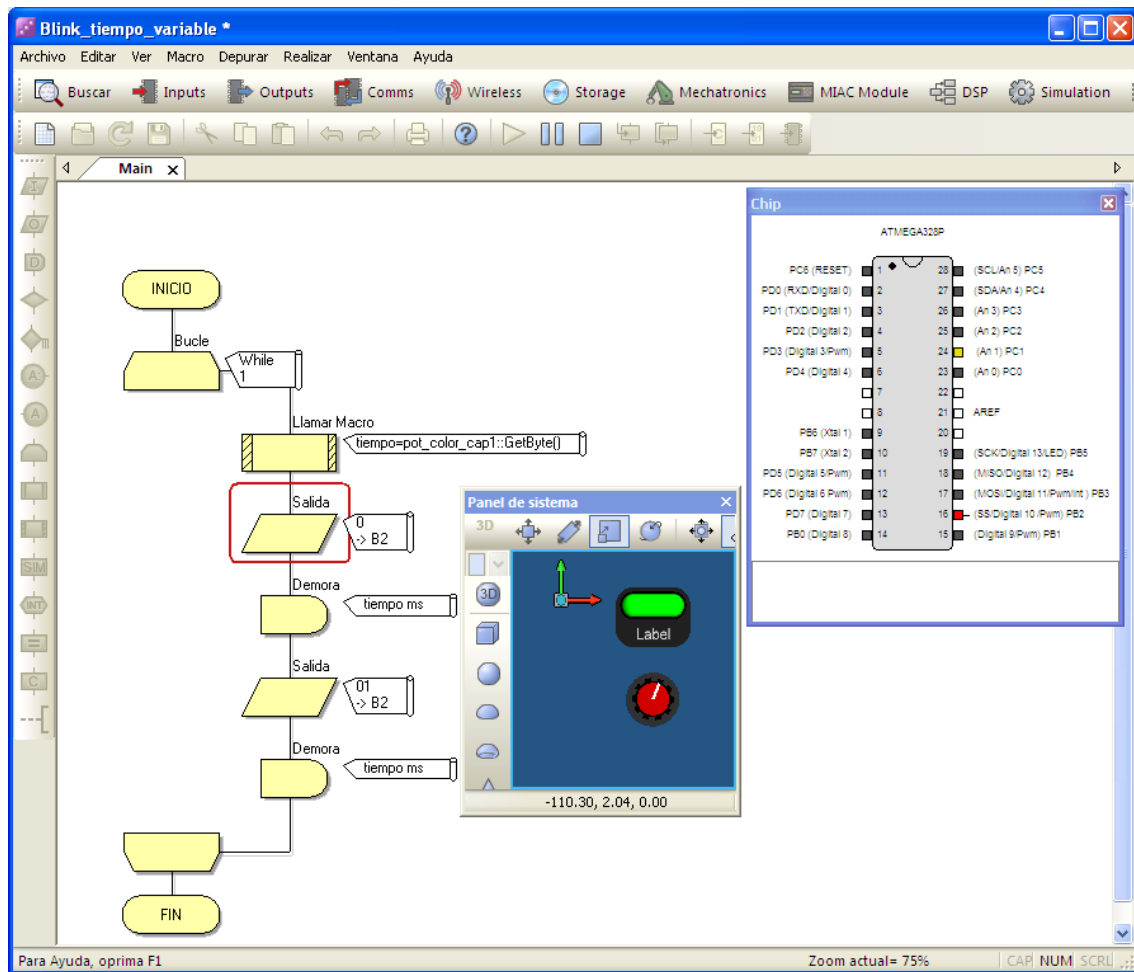
Tal como se en el siguiente el diagrama de flujo ya terminado.

No olvidemos que el tiempo de demora ser no un valor numérico sino la variable “tiempo”.



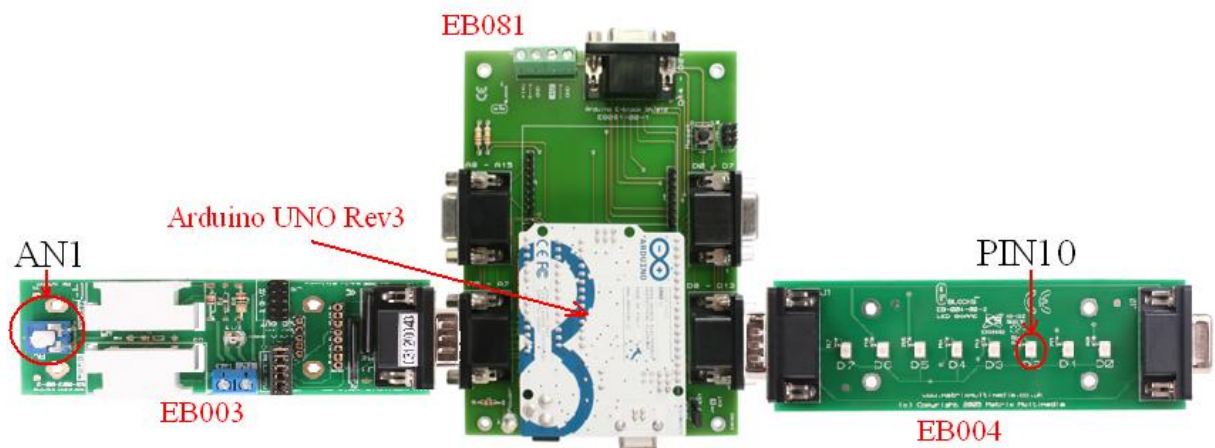
Lo que corresponde a continuación es realizar la simulación del montaje. Para ello pulsamos el correspondiente botón y vemos como al variar la posición del slider varia el tiempo de encendido y apagado del LED. Podemos hacer más lenta la ejecución y ver como los bloques se van ejecutando en el diagrama de flujo.

En la siguiente imagen vemos el aspecto del sistema mientras se está realizando la simulación



Comprobado el correcto funcionamiento de nuestro sistema descargamos la aplicación sobre la tarjeta **Arduino**.

En el montaje obsérvese que se ha colocado la tarjeta de sensores analógicos para poder usar el Potenciómetro de esta que está colocado en el canal AN1.



7. Botón

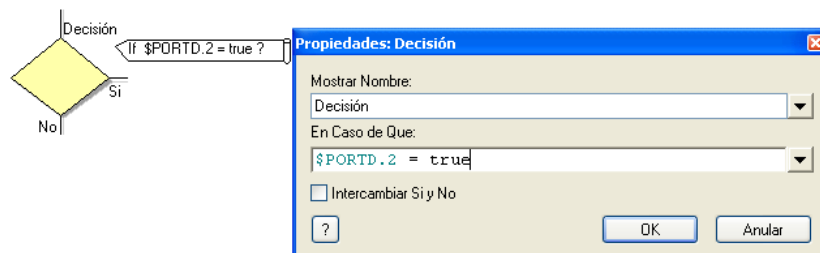
La siguiente práctica consiste en la activación de una salida digital dependiendo del estado de una entrada digital.

La designación de estas señales será:

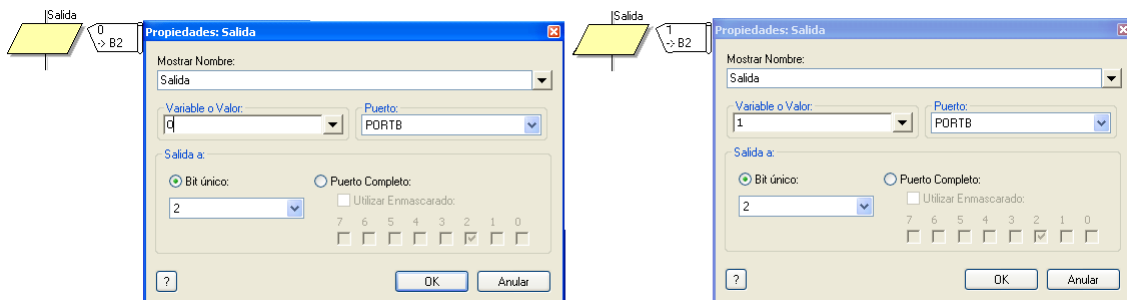
Entrada digital: **PIN 2 (Arduino)** PORTD.2 o PD2 (**Flowcode**)

Salida digital: **PIN 10 (Arduino)** PORTB 2 o PB2(**Flowcode**)

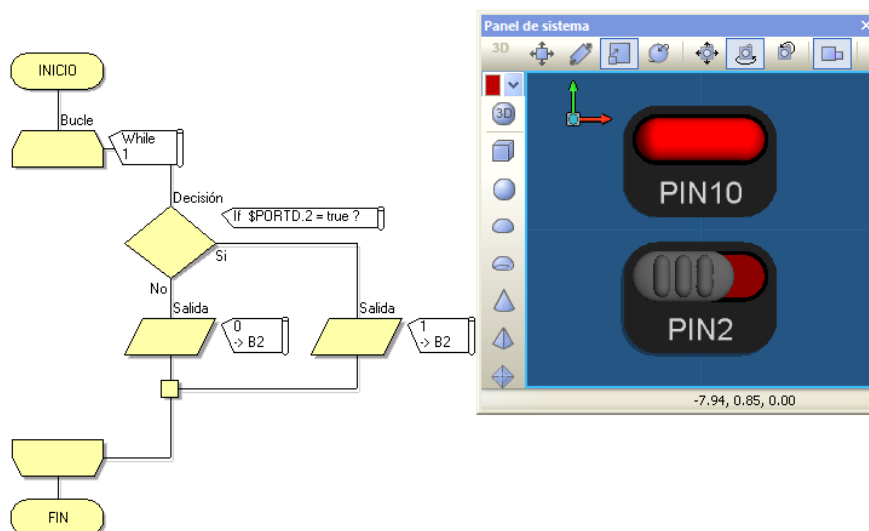
El diagrama de flujo es muy sencillo. Dentro, como siempre, de **“Bucle”** pondremos un condicional **“Decisión”** en el que se preguntará por el estado de la variable de entrada del Puerto \$PORTD.2



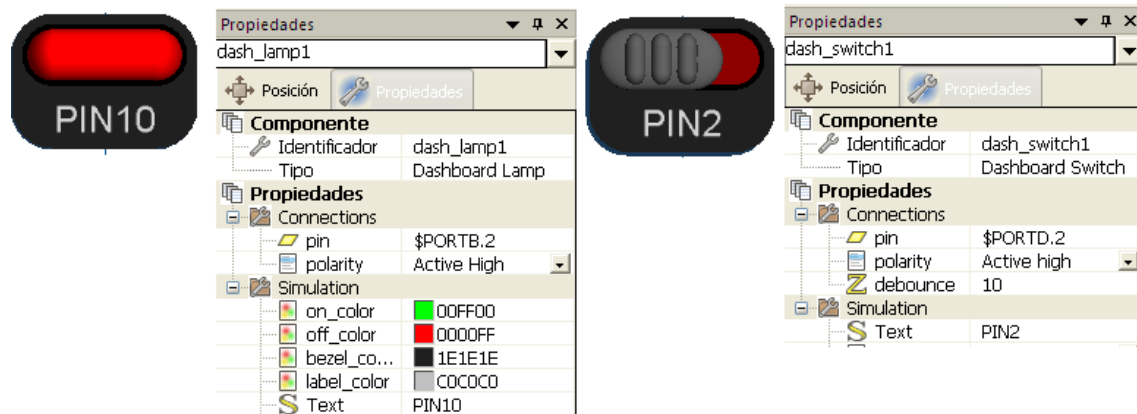
Seguidamente colocaremos los bloques de “Salida” con la señal de puerto PB2 en valor “0” y valor “1”



Finalmente el organigrama quedará como se muestra en la siguiente figura

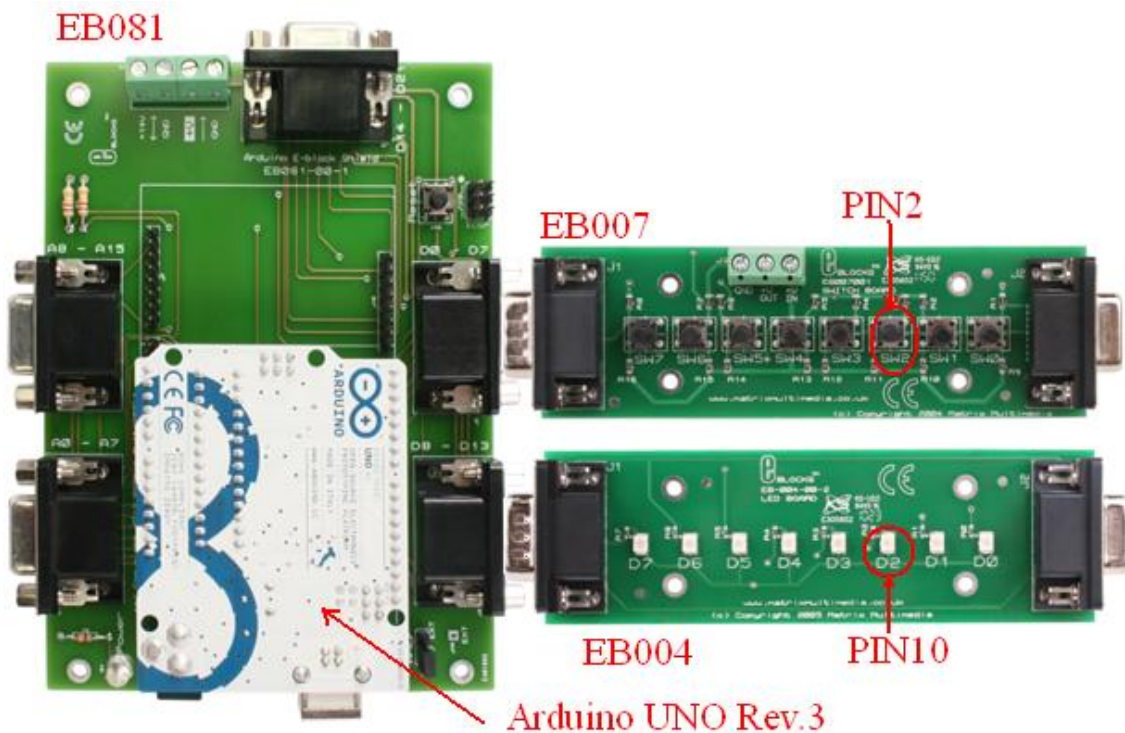


Para el Panel de Sistema se han colocado un Led (dash_lamp1) y un Interruptor (dash_switch1) sacados de las librerías “Inputs” y “Outputs” respectivamente



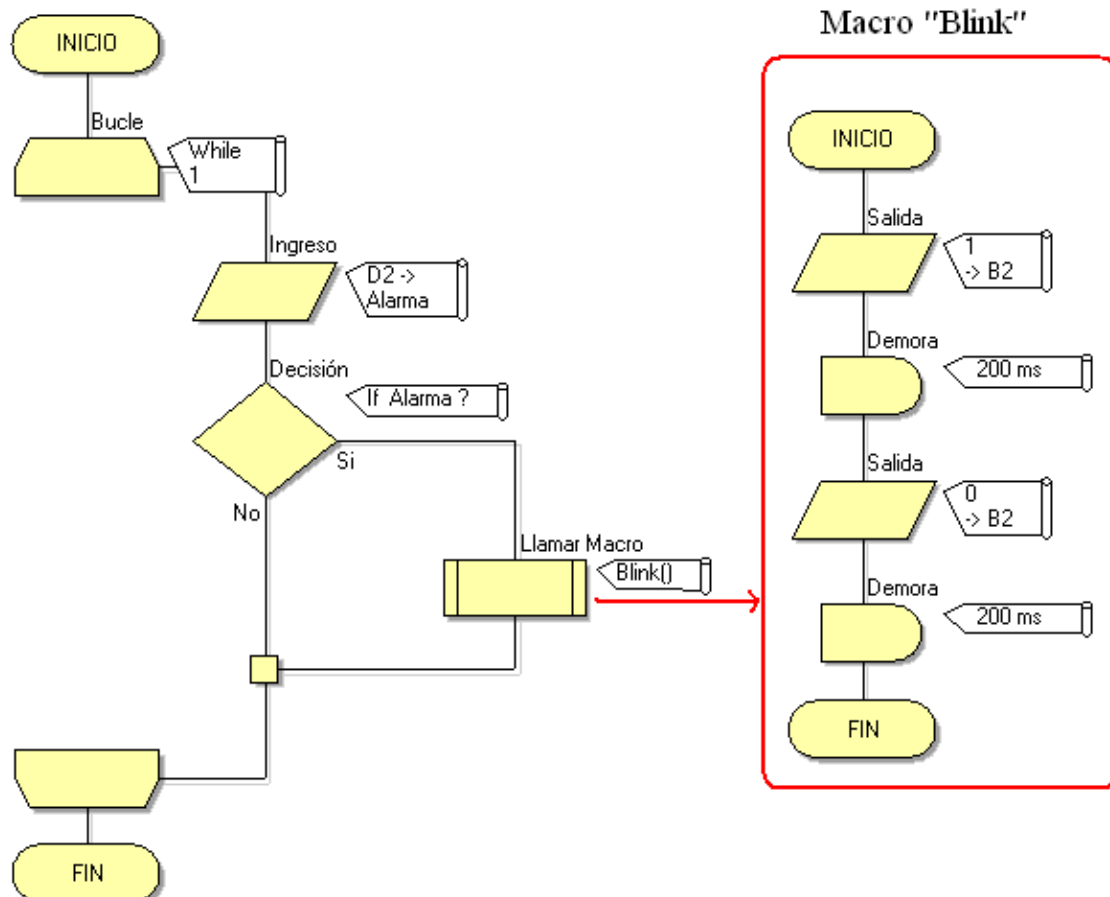
A estos dos elementos se les asociaran los pines correspondientes \$PORTB.2 y \$PORTD.2 respectivamente y las etiquetas PIN10 y PIN2

Lo siguiente será comprobar el funcionamiento, descargar sobre la tarjeta **Arduino** el código y verificar que todo está correcto a nivel físico del sistema.



8. Alarma básica

A continuación vamos a diseñar una alarma sencilla que consistirá en el parpadeo de una salida digital **PIN10** cuando se pulse una entrada digital que esta vez en lugar de ser directamente el pin será una variable que habremos creado previamente.



En los diagramas de flujo que se muestran queda fácilmente aclarado el algoritmo de esta programa. Cuando se activa la entrada **D2** a la que hemos llamado Alarma se cumple la condición del bloque “**Decisión**” y se invoca la macro **Blink** que activa y desactiva la salida **B2** cada 200 ms.

En la siguiente imagen vemos los elementos que hemos definido en nuestra aplicación:

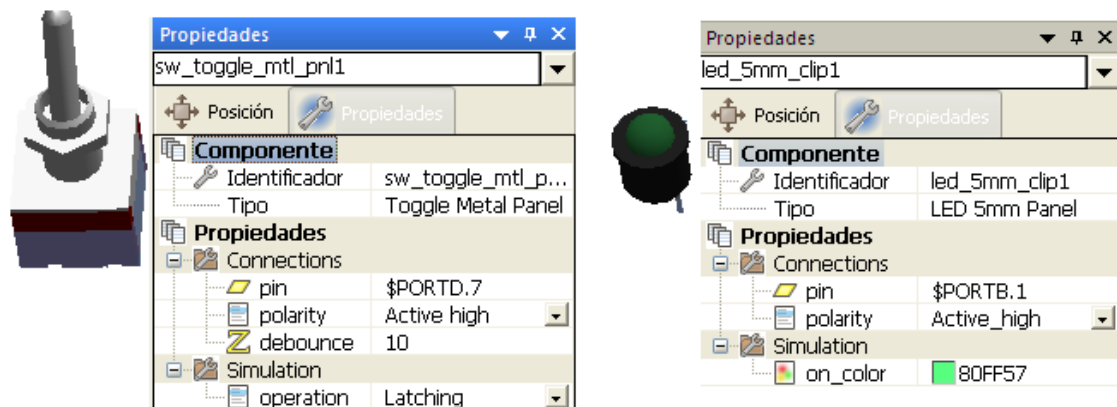
- Una Variable llamada **Alarma**
- Una Macro llamada **Blink**
- Un componente llamado **led_5mm_clip1**
- Un componente llamado **sw_toggle_mtl_pnl1**



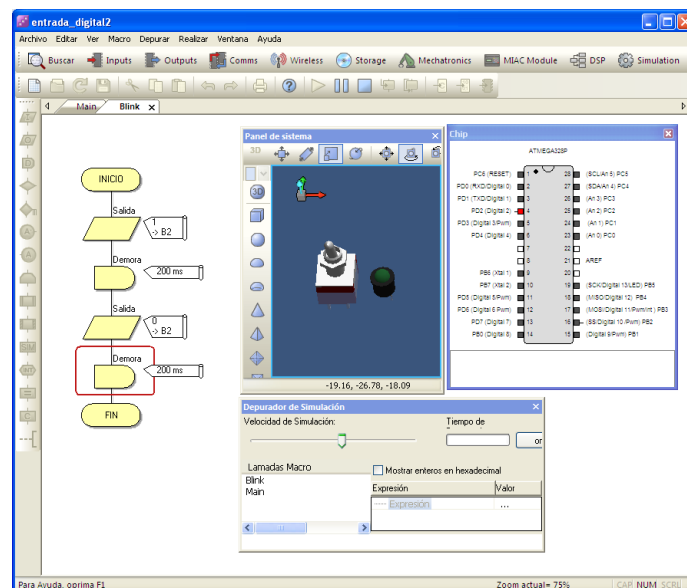
Los parámetros de los componentes son los que se muestran en la siguiente figura:

led_5mm_clip1 -> pin (\$PORTB.1)

sw_toggle_mtl_pnl1 -> pin (\$PORTD.2)

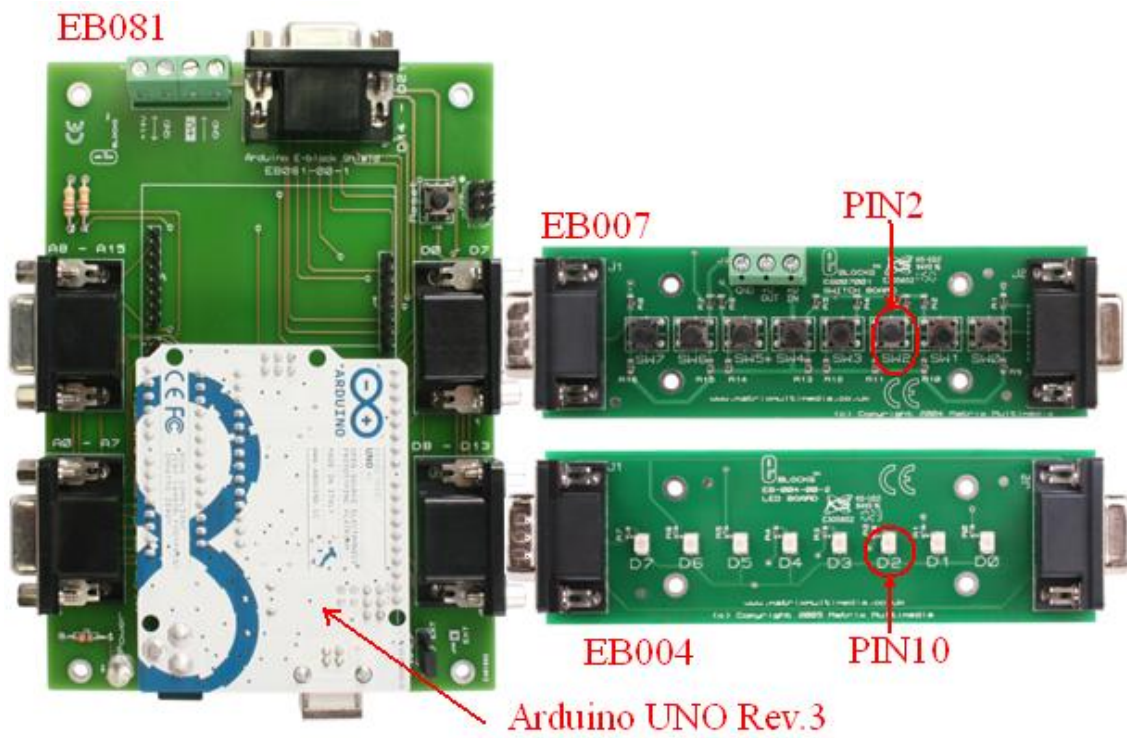


Una vez montado simulamos la aplicación.



Se descarga el programa sobre la tarjeta **Arduino** y se prueba que todo va bien.

El montaje es el siguiente.



9. Monitorización de funcionamiento con Alarma

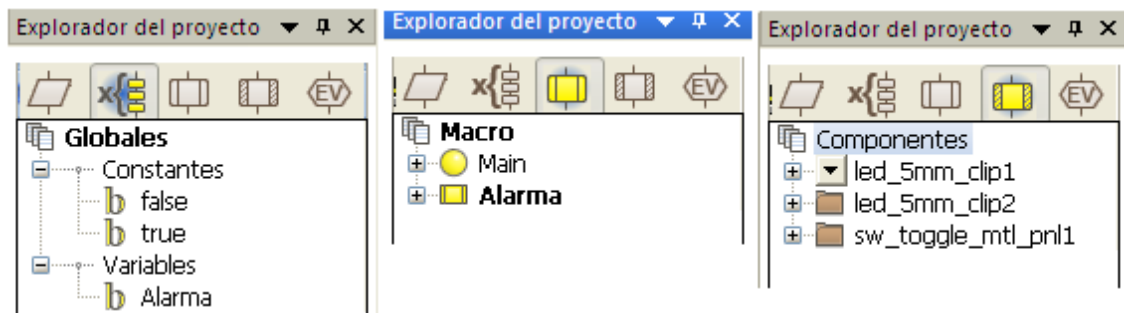
Este nuevo ejemplo es una variante del anterior. Se trata de gobernar una salida que indica el estado de buen funcionamiento de un sistema con una entrada que indica un estado de alarma.

Cuando por el pin de entrada de alarma parece un “1” se desactiva la salida que indica buen funcionamiento y se pone intermiten otra salida indicando que existe una alarma.

Las variables serán:

- Se define una variable de estado que llamamos “**Alarma**” de tipo booleano.
- El **Pulsador de Alarma** se establecerá en el **PIN 4** que se corresponde con el **PORTD.4**
- La salida **Funcionamiento** será el **PIN 10** que se corresponde con el **PORTB.2**.
- La salida intermitente **Salida Alarma** será el **PIN 9** que se corresponderá con **PORTB.1**

A continuación se muestran los elementos que debemos crear: La variable, una macro llamada Alarma y los componentes del Panel de Sistema.

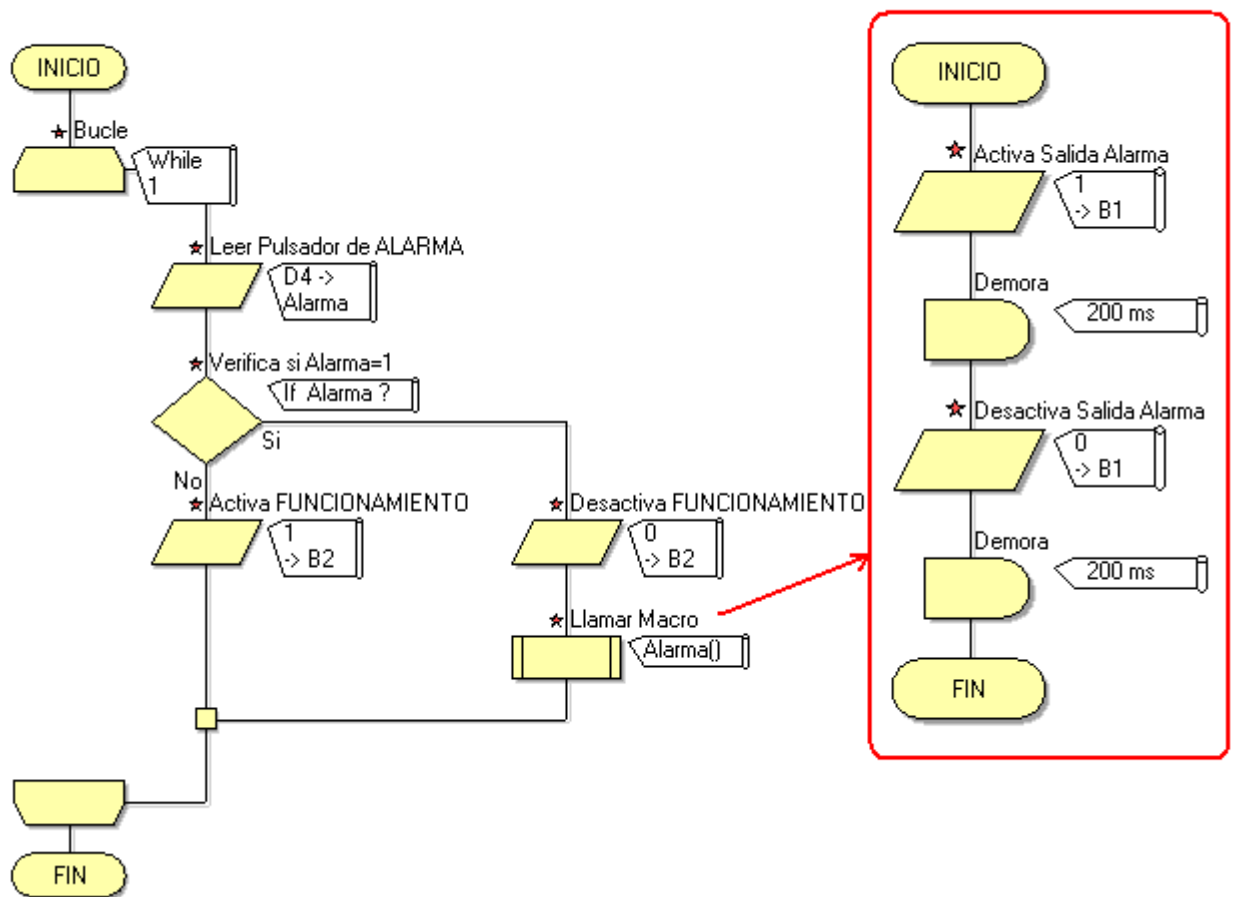


El programa es muy parecido al anterior. En el bloque principal “**Main**” se establece el “**Bucle**” dentro del cual lo que haremos será testear el estado de la entrada “**Pulsador de Alarma**” si su valor es “0” se activará la salida “**Funcionamiento**” y si es “1” (alarma) se desactiva la salida “**Funcionamiento**” y se ejecuta la macro “**Alarma**” en la que se ponen en estado intermitente “**Salida Alarma**”.

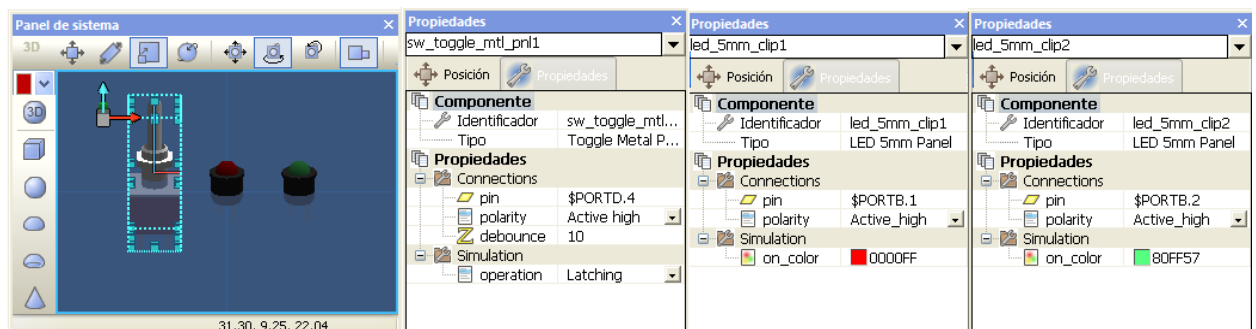
En la simulación se han dispuesto dos Leds para indicar las salidas y un interruptor para indicar la entrada de Alarma.

Recordemos que los pines que se utilizan de **Arduino** son el **PIN10**, **PIN9** y **PIN43**.

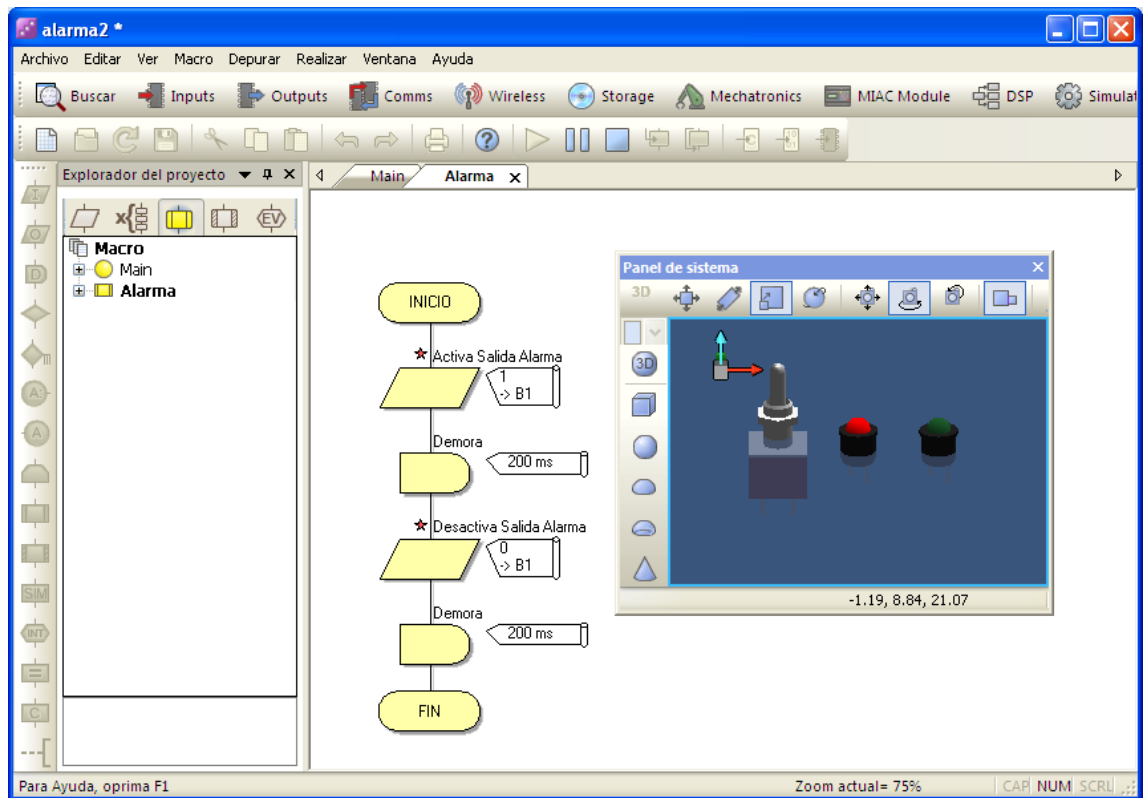
En la siguiente imagen se muestran los diagramas de flujo del programa **Main** y de la macro **Alarma**



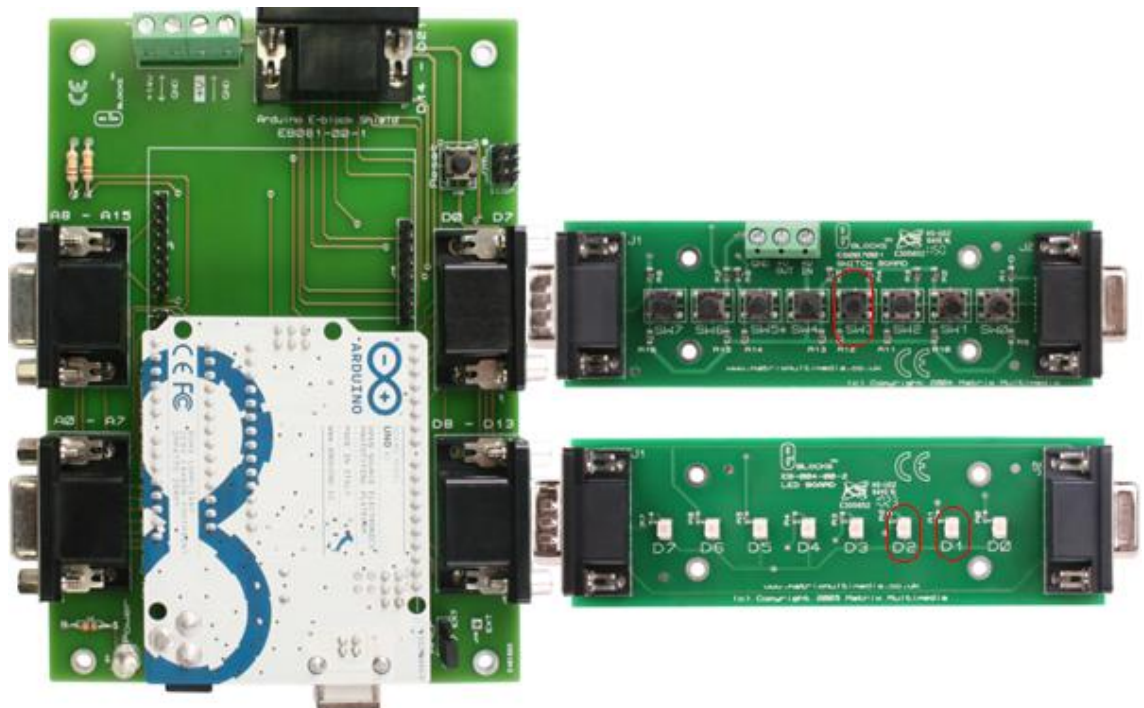
La configuración de los elementos de simulación es la que se muestra en la siguiente imagen



Una vez realizada la programación se procederá a la simulación de la aplicación comprobando que funciona de acuerdo a como la hemos pensado.



El montaje con las tarjetas E-Blocks es el siguiente.



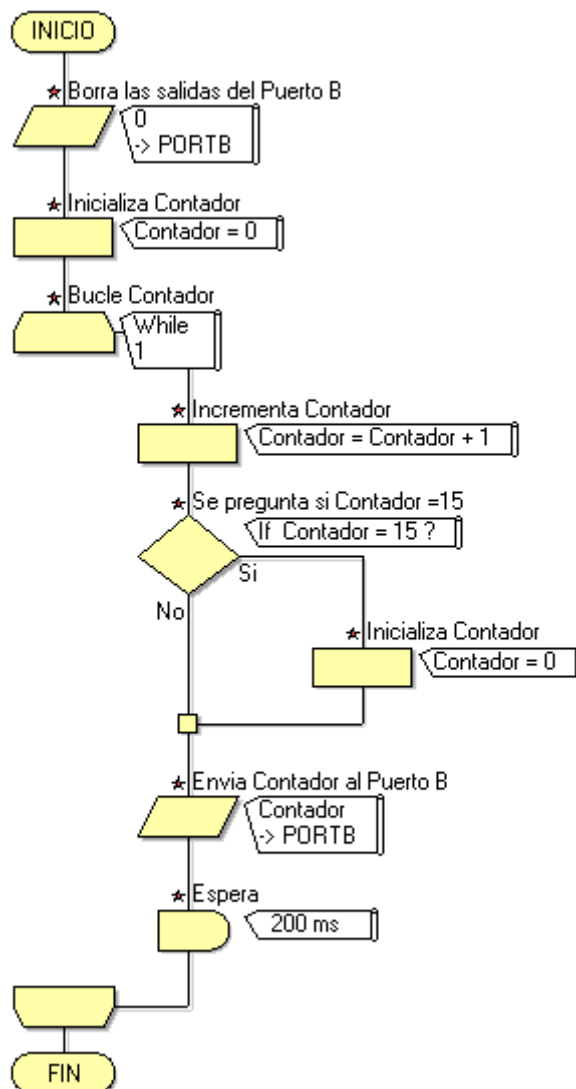
10. Ejemplo Contador

Vamos a realizar un contador que ponga su estado de cuenta, codificado en binario, en el puerto de salida para gobernar los pines digital de Arduino PIN8, PIN9, PIN10, PIN11 que son los cuatro bits de menor peso del puerto PORTB.

Se creará una variable llamada “Contador” que será la que se lleva a las salidas del puerto.

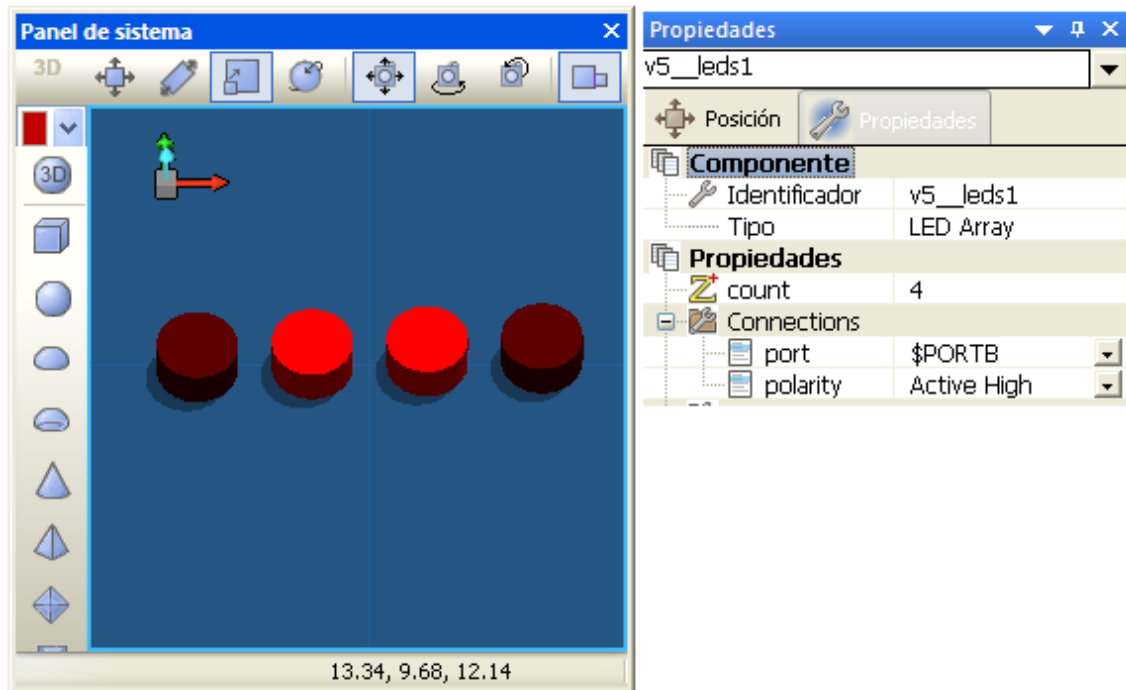


Veamos las instrucciones que debemos colocar en nuestro diagrama de flujo

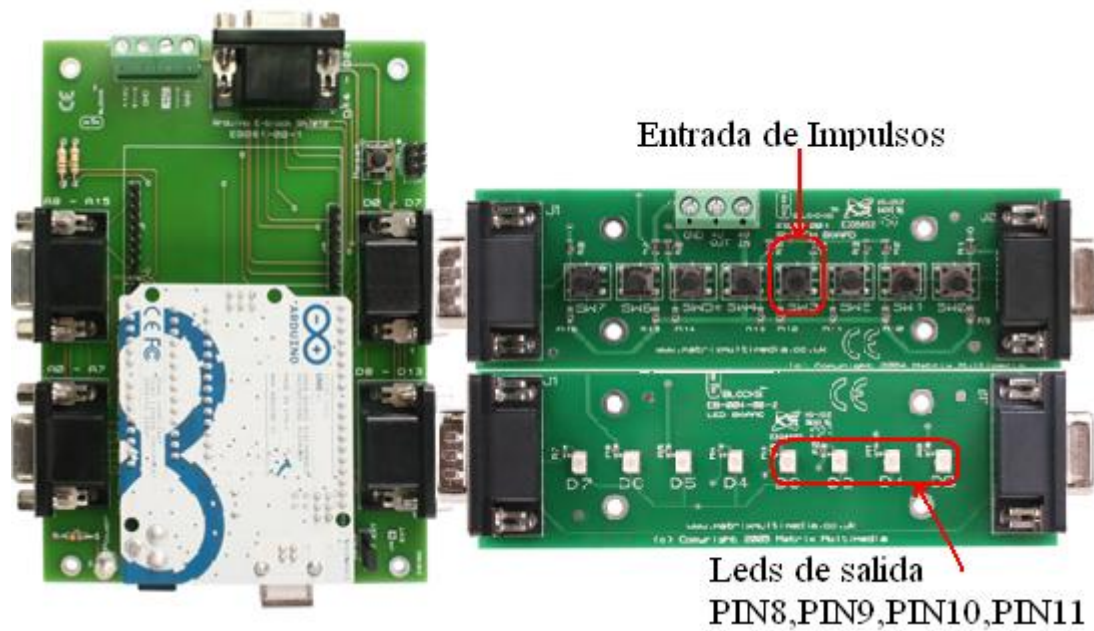


Lo primero que hacemos es poner a cero el estado de las salidas de **PORTB**, bastará con enviar a el el dato “0”. Seguidamente ponemos a “0” el valor de la variable Contador y a continuación preguntaremos si el valor de la variable es igual a 15. Si se cumple la condición se pondra a cero la variable (Contador=0) seguidamente se sacará el valor por el puerto **PORTB**. Para finalizar ponemos un retardo para poder ver bien el encendido y apagado de los leds de salida.

En la siguiente imagen se puede distinguir el componente que hemos insertado en el “Panel Sistema”. Se trata de un array de 4 leds que asociamos al puerto **PORTB** para poder ver, en modo simulación el esatdo del puerto. Vemos tambien la ventana de parámetros tal como queda configurada.



Realizamos la simulacion del ejemplo y posteriormente descargamos el programa sobre la tarjeta Arduino, que presentara la siguiente configuracion de conexiones.

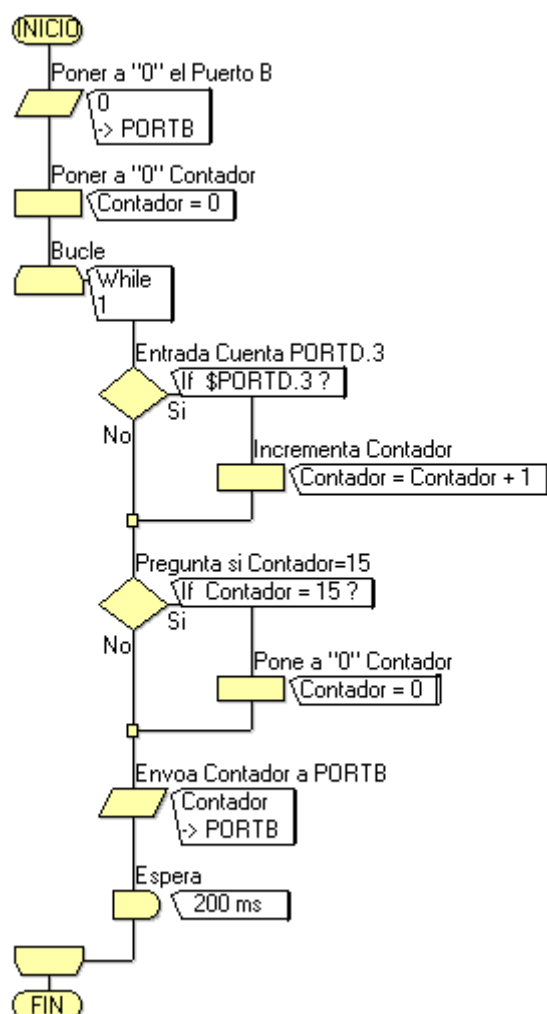


11. Contador de impulsos de entrada.

Vamos a realizar una variante del ejemplo anterior que consistirá en contar de 0 a 15 los impulsos que entran por una de los terminales digitales de Arduino.

Las definiciones serán las siguientes:

- Variable de cuenta = **Contador**
- Entrada de Impulsos de cuenta el **PIN3** de Arduino (**PORTD.3** del Chip de **Flowcode**).
- Los valores de la variable Contador saldrán en forma binaria por los cuatro primeros bits del puerto PORTB que se corresponden con los pines **PIN8,PIN9,PIN10,PIN11**.

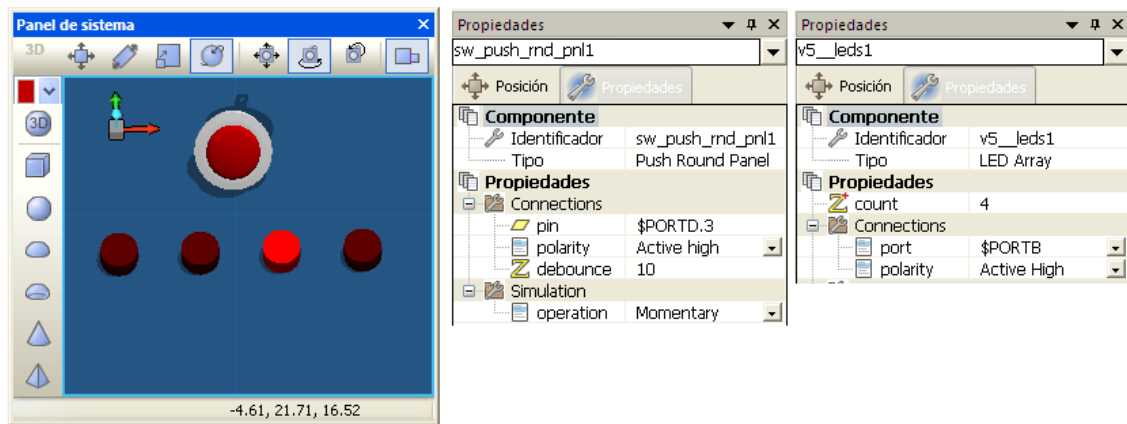


Las variaciones con respecto al ejemplo anterior son muy pocas, únicamente que en este caso lo que hacemos es condicionar el incremento de la variable Contador al estado de la entrada PIN3, de tal forma que si esta entrada esta en nivel “1” entonces se incrementa la variable en caso contrario no se incrementa.

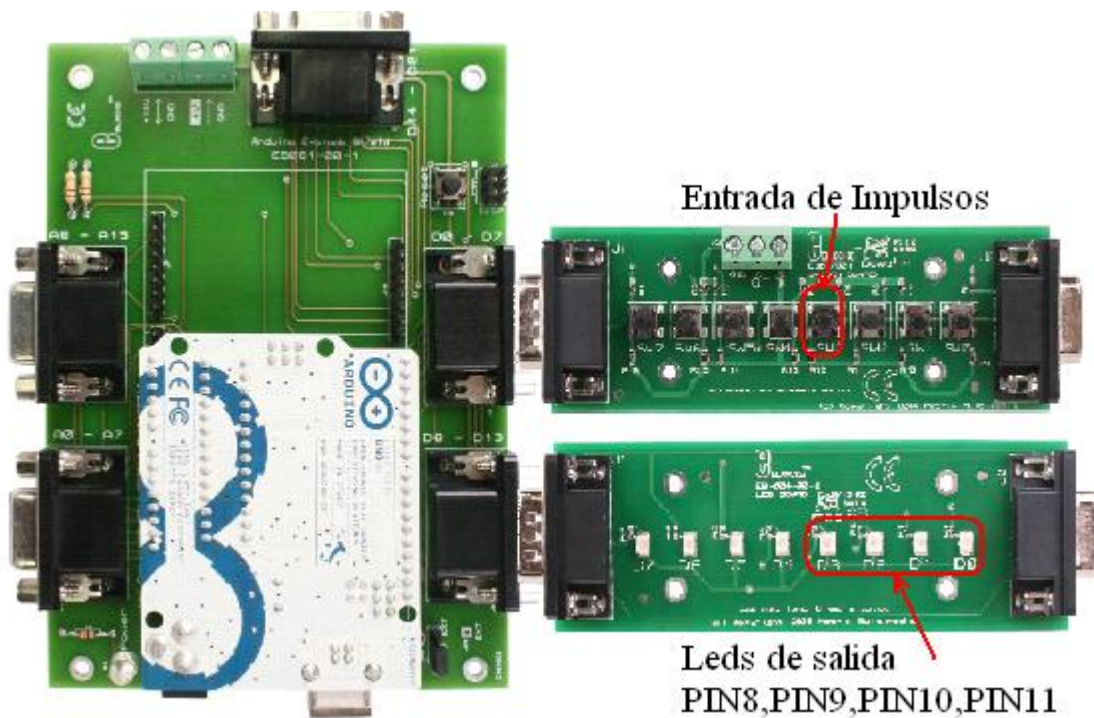
En el caso de que se mantenga en estado de “1” la entrada el contador ira contando sin parar.

Sobre este ejemplo se podrían realizar otras variantes como por ejemplo la cuenta hacia delante o hacia a tras, la puesta a cero mediante un pulsador, etc.

A continuación, en el siguiente dibujo, se muestran los elementos que esta vez se colocaran en la venta de simulación.



El montaje físico de las tarjetas es el siguiente.



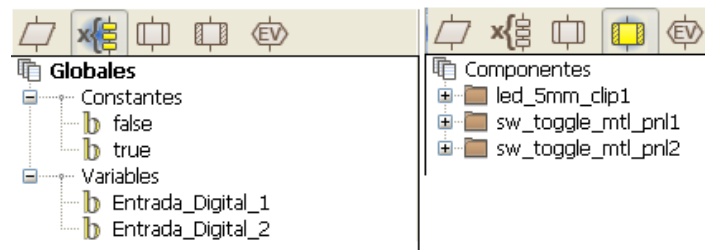
12. FUNCION LÓGICA AND

Este ejemplo nos permite comprender como establecer funciones lógicas entre valores de entrada.

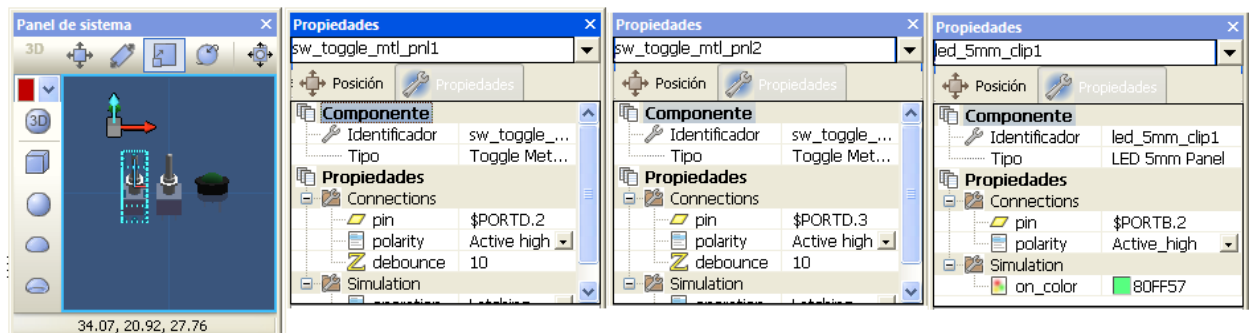
Vamos a definir dos entradas digitales y una salida digital. Las señales son:

- **Entrada_Digital_1 PIN 2** que se corresponde con el pin **PORTD.2**
- **Entrada_Digital_2 PIN 3** que se corresponde con el pin **PORTD.3**
- **Salida Digital de la Función PIN 10** que se corresponde con el pin **PORTB.2**

Seguidamente se muestran las variables y los elementos de simulación de esta aplicación

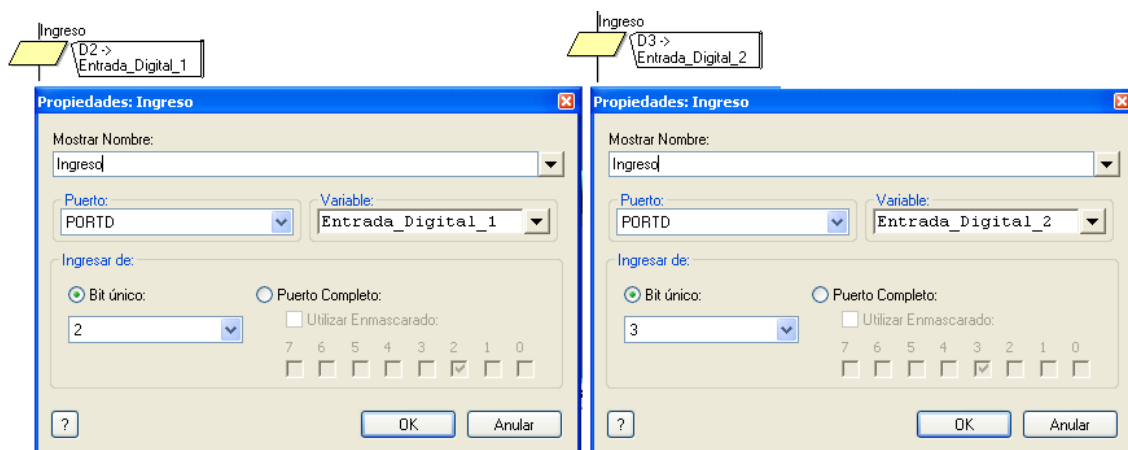


En la figura vemos la configuración de los elementos de simulación



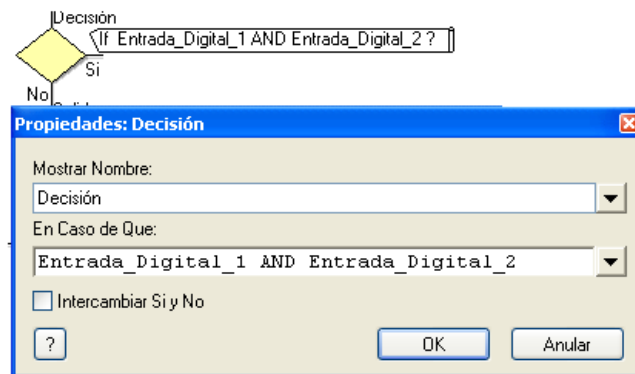
El diagrama de flujo de la aplicación es el siguiente.

Las entradas se recogen mediante el bloque de función “Ingreso” que se configura con

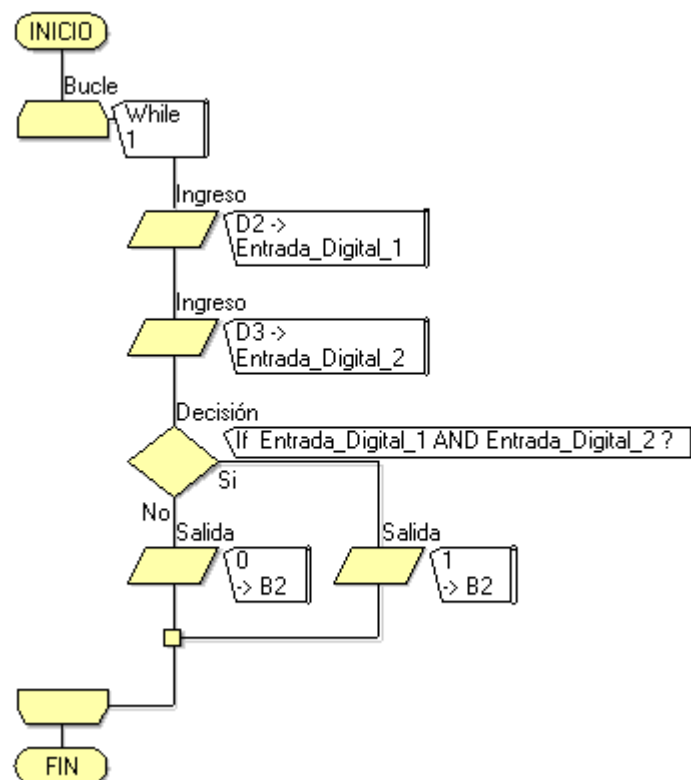


La instrucion de “Decisión” tiene asociada la condición

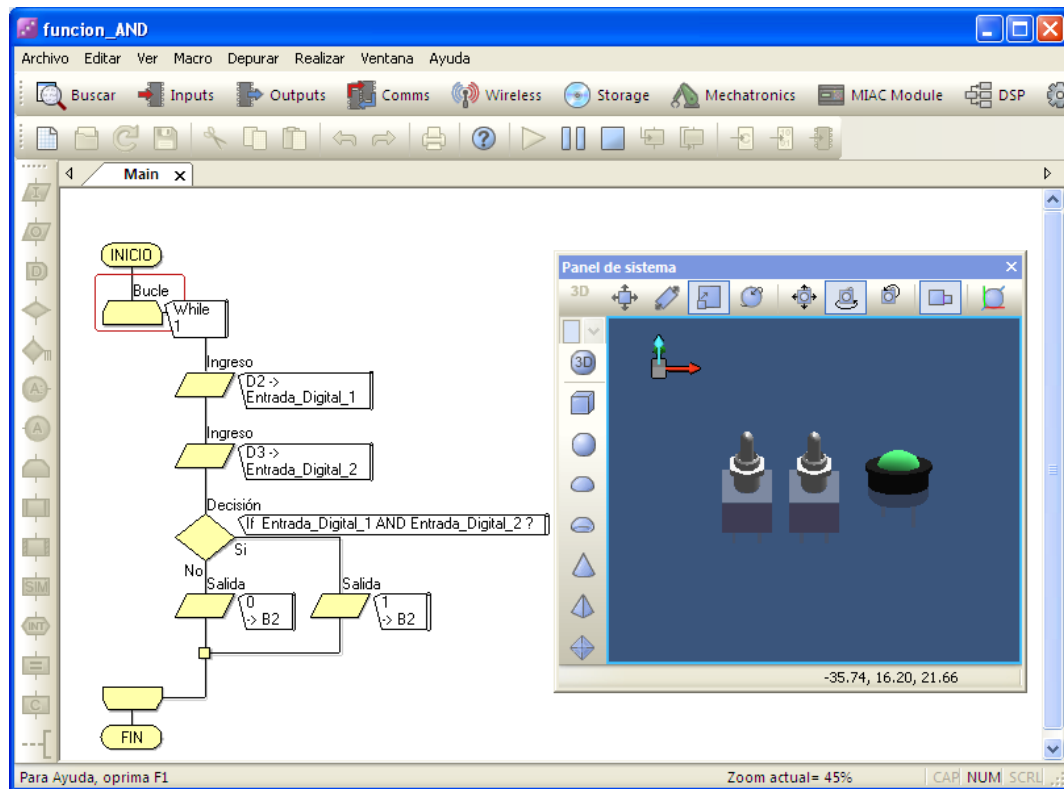
Entrada_Digital_1 AND Entrada_Digital_2



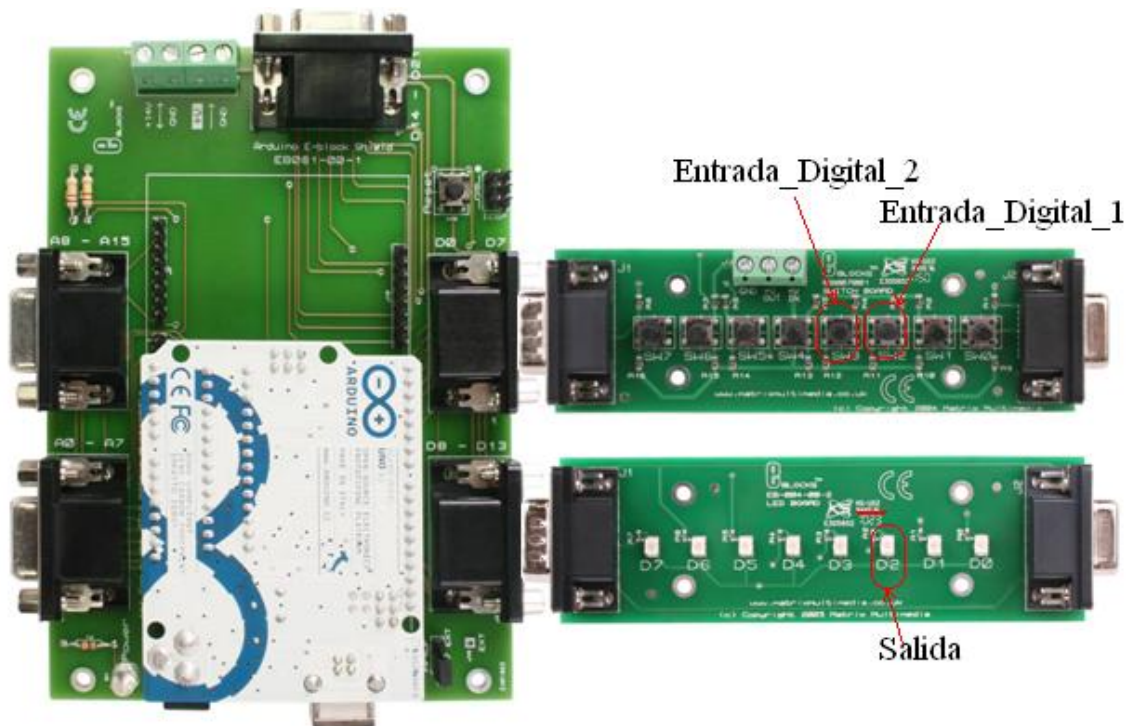
A continuación se muestra el diagrama de flujo completo



La ventana siguiente es una vista general de la simulación.



El montaje de las tarjetas E-Blocks es el siguiente.

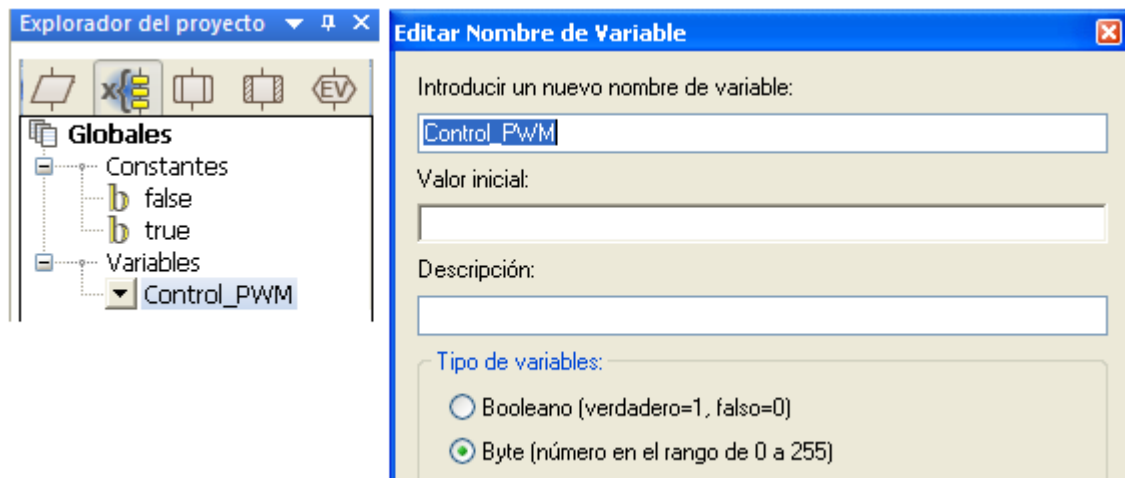


13. Salida PWM

Vamos a crear una sencilla aplicación en la que gobernaremos una salida de tipo PWM de **Arduino**. El valor numérico con el que controlaremos la salida se recogerá de un canal de entrada analógica de **Arduino**.

Flowcode incorpora una macro que nos facilitará la programación de esta salida analógica en formato PWM. Este bloque se encuentra en la librería “Outputs”.

Definiremos una variable llamada **Control_PWM** a la que se asociará el valor de control del canal PWM.

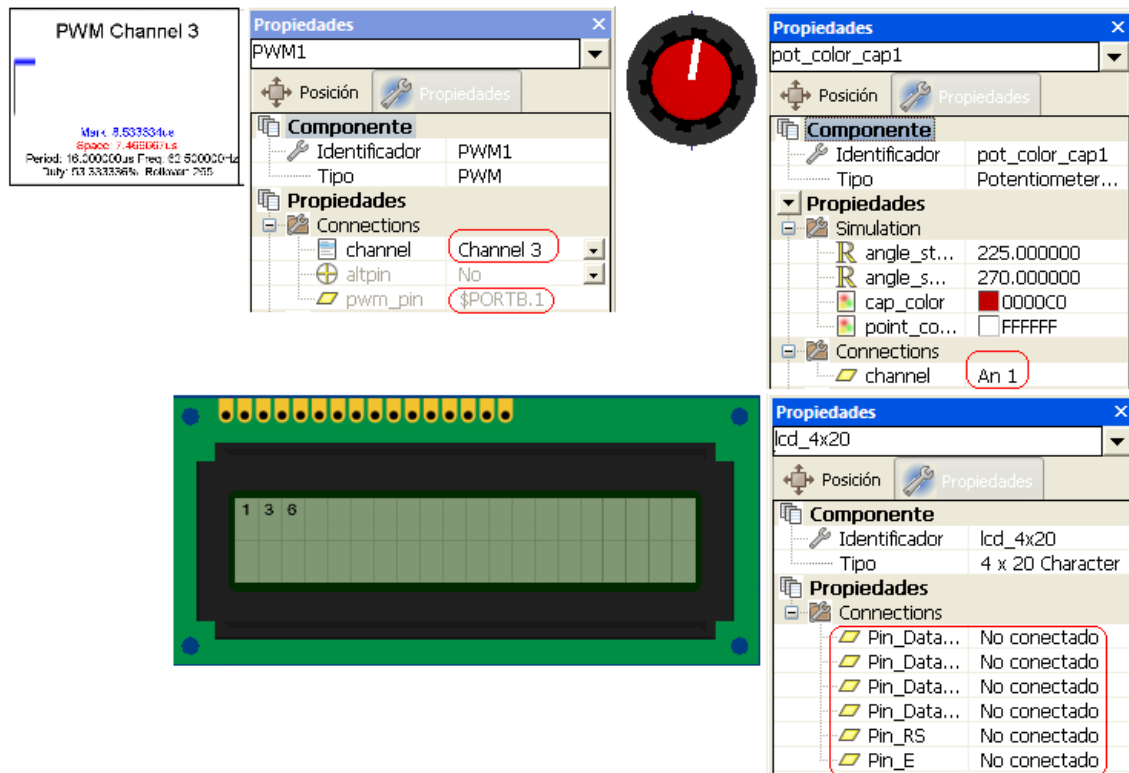


El control de la salida PWM se realizará mediante la entrada analógica **AN1** y el pin de salida analógica PWM será el **PIN9** que se corresponde con el pin de **Flowcode** **PORTB.1** y que equivale al **Channel3** del objeto de librería PWM.

Se han colocado los siguientes objetos en el “**Panel de Sistema**”: Un **potenciómetro** (pot_color_cap1), un **Display tipo CD** (cd_4x20) y el propio objeto **PWM**.

En la siguiente imagen vemos sus parámetros.

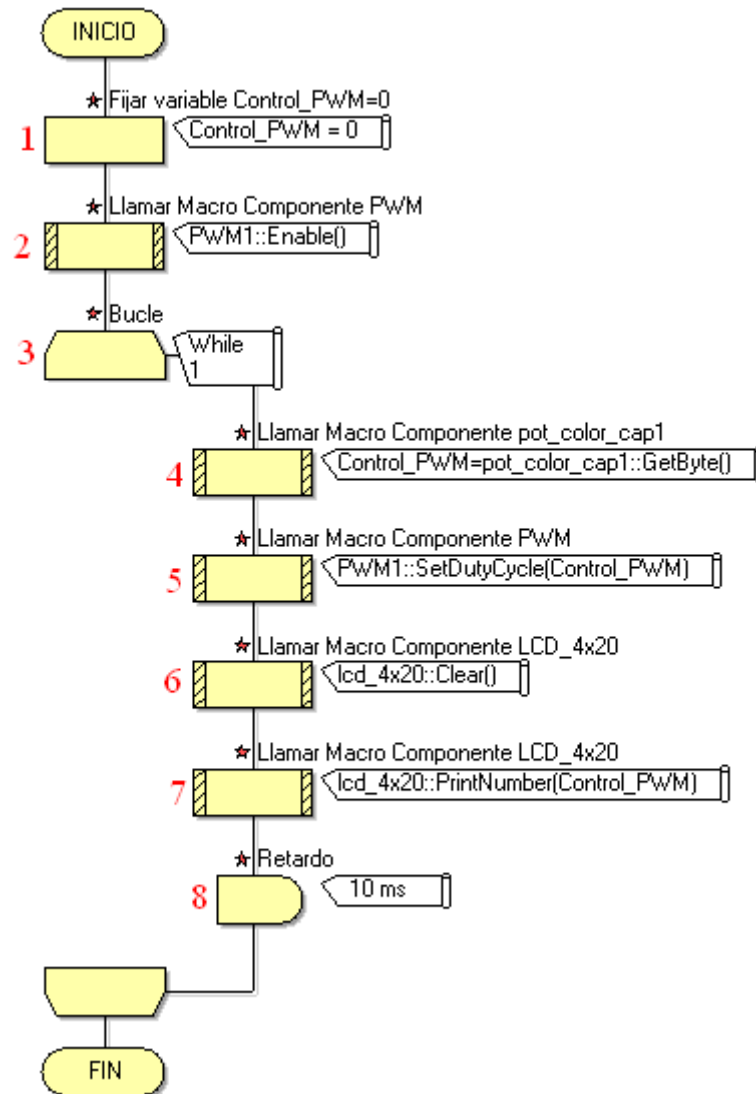
El objeto potenciómetro nos permite definir el canal analógico que vamos a leer en la tarjeta **Arduino** **AN1** y el bloque PWM en el que hemos seleccionado el Canal3 nos permite definir como salida PWM el **PIN9**.



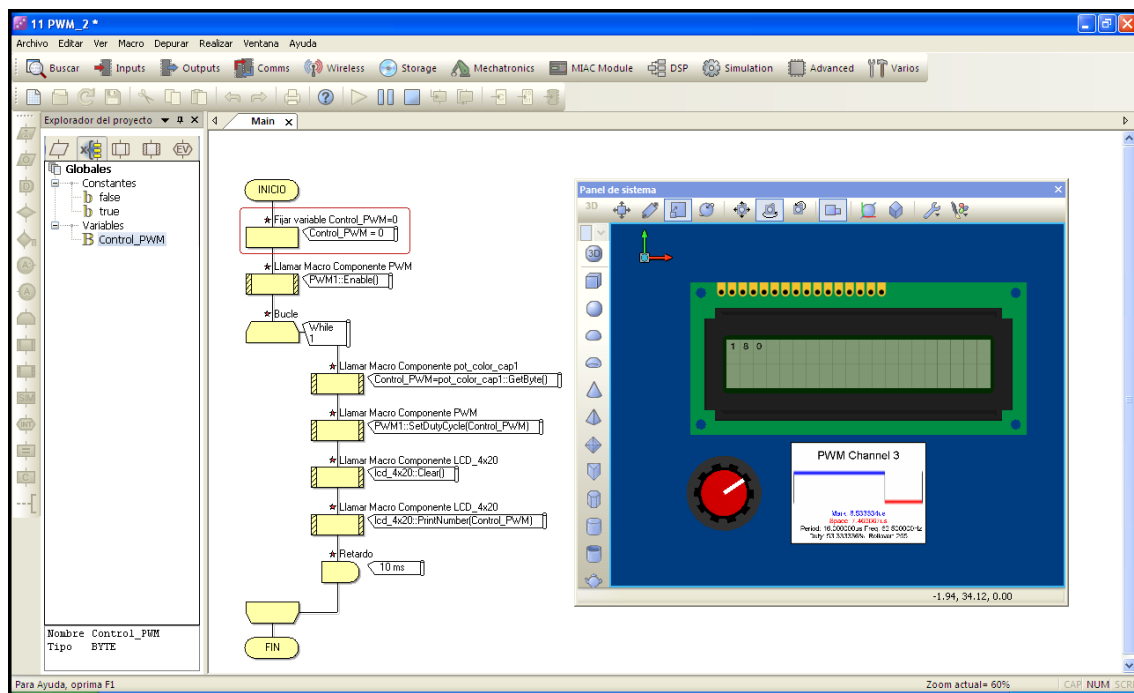
La programación de la aplicación es muy sencilla.

1. Colocaremos en primer lugar un bloque para asignar un valor por defecto a la variable **Control_PWM** que es la que recoge el valor de activación de la señal de salida PWM (Bloque de Calculo al que hemos etiquetado como "Fijar bvariable Control_PWM=0")
2. Seguidamente colocamos un bloque de **Macro** en el que activamos el componente **PWM** (bloque Macro al que hemos llamado *Lllamar Macro Componente PWM*) seleccionando el parámetro **Enable**.
3. Una vez realizadas estas operaciones entraremos en el **Bucle** y allí llamamos a un bloque **Macro** que se encargara de designar el potenciómetro a la variable **Control_PWM** que ya hemos asociado al pin **AN1** cuando hemos definido el objeto para colocarlo en el **Panel de Sistema**.
4. En el siguiente bloque **Macro** (*Lllamar Macro Componente pot_color_cap1*) configuramos el parámetro **SetDutyCycle** del objeto **PWM** a la variable **Control_PWM**.
5. Con este nuevo bloque **Macro** (*Lllamar Macro Componente PWM*) asociamos el parámetro **SetDutyCycle** a la variable **Control_PWM**
6. Este bloque Macro (*Lllamar Macro Componente LCD_4x20*) y borra lo escrito anteriormente (**Clear**)

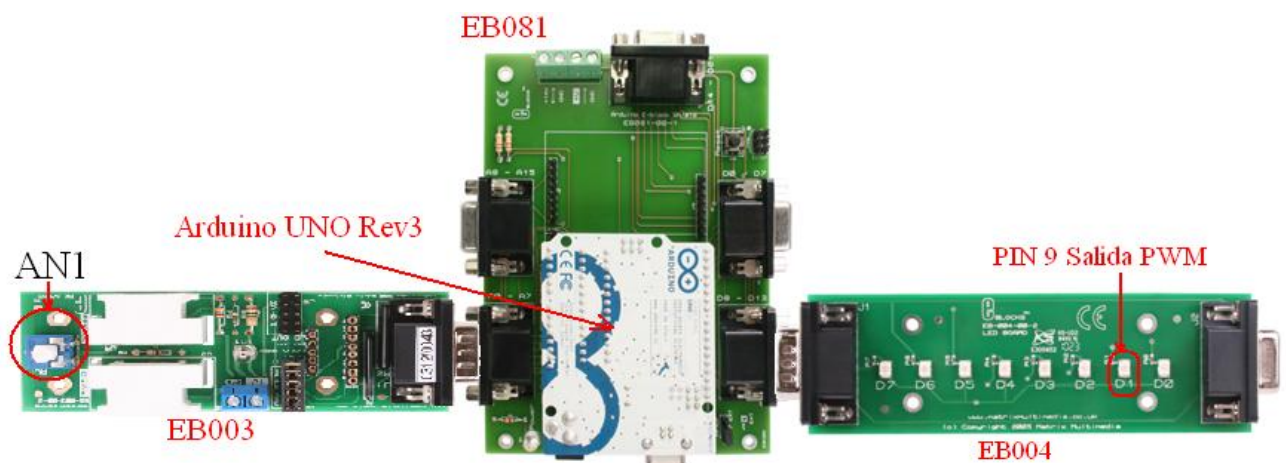
7. Este bloque Macro (*Llamar Macro Componente LCD_4x20*) realiza la escritura del valor de la variable **Control PWM** en el display
8. Par finalizar ponemos un bloque Demora que introduce un retardo de 10ms.



En la simulación podremos observar en el componente PWM el trazado de la señal PWM comprobando el factor de ciclo.



El montaje de las tarjetas E-Blocks será el siguiente.

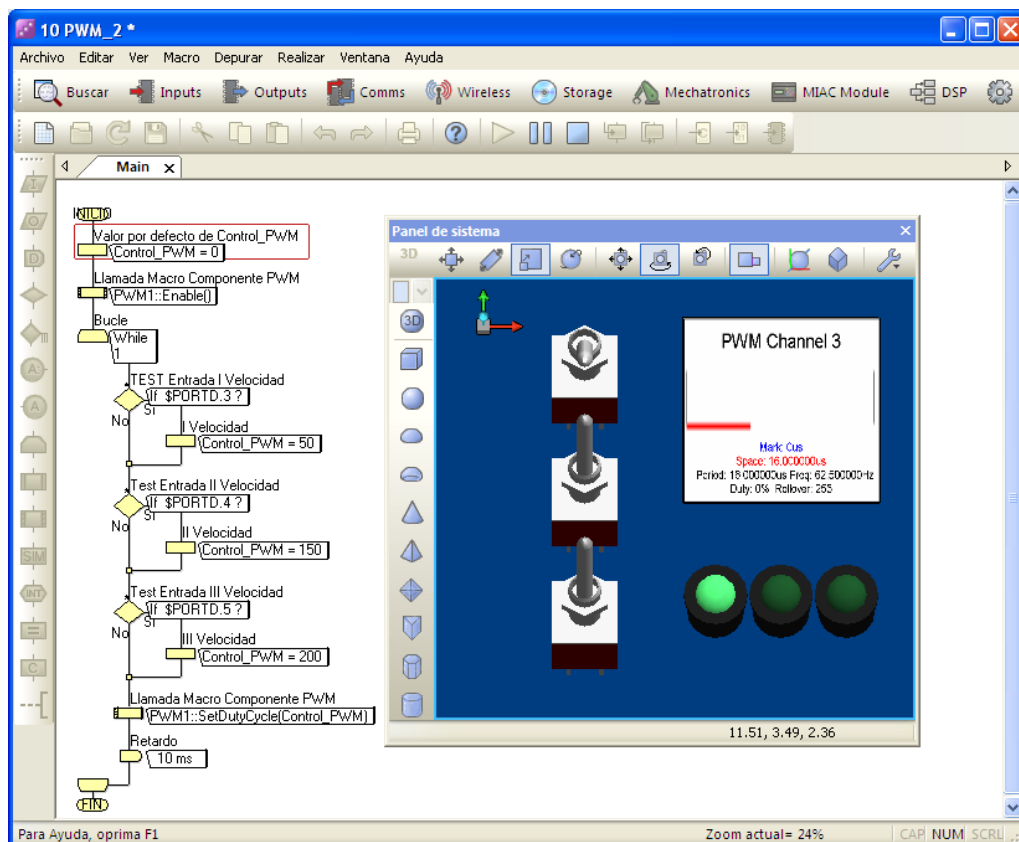


14. Control de un motor con tres velocidades.

Con el siguiente ejemplo pretendemos controlar un pin de salida analógica con tres niveles de salida que servirán para gobernar un motor eléctrico de c.c. con tres velocidades respectivamente.

Para nuestro ejemplo vamos a contar con tres interruptores conectados en las entradas correspondientes a los pines **PIN3**, **PIN4** y **PIN5** de **Arduino** y la salida será el **PIN9** que como sabemos es una salida analógica de tipo PWM.

Cada uno de las entradas será testeada mediante un bloque de tipo condicional y si se cumple que esté activado asignará un valor a la variable de control de la salida PWM que la llamaremos **Control_PWM**.



Las velocidades serán:

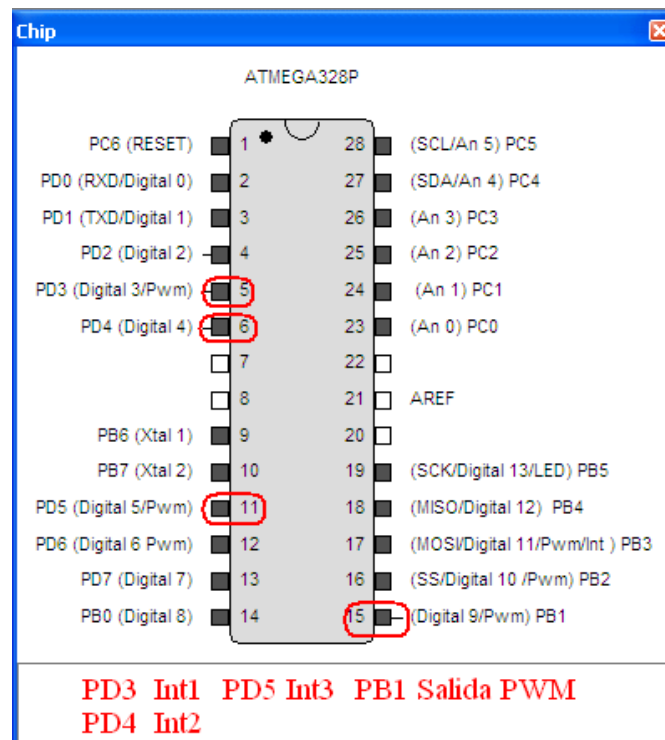
I Velocidad (Int1) PIN3 Control_PWM= 50

II Velocidad (Int2) PIN4 Control_PWM= 150

III Velocidad (Int3) PIN5 Control_PWM= 200

El Panel Sistema de nuestra aplicación tendrá tres interruptores, uno por cada velocidad, tres Leds indicadores de la velocidad y el elementos PWM.

En la siguiente figura vemos la designación de pines.



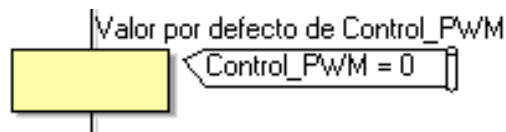
El Panel de Sistema se configurará con los tres interruptores (*sw_toggle_mtl_pnl1*, *sw_toggle_mtl_pnl2*, *sw_toggle_mtl_pnl3*) que simularan las entradas correspondientes. Se colocarán tres diodos leds (*led_5mm_clip1*, *led_5mm_clip2*, *led_5mm_clip3*) para señalar el estado de estas entradas y finalmente aparece el objeto **PWM1**



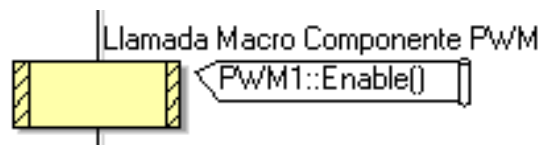
A continuación se muestra el diagrama de flujo de la aplicación. Vemos que, mediante tres condicionales, preguntamos el estado de las entradas y en cada caso se asigna un valor a la variable **Control_PWM** que es la que finalmente es enviada a la salida **PWM1**.

La configuración del algoritmo es la siguiente:

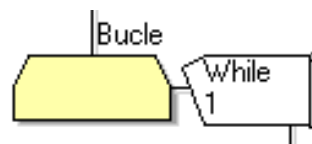
1. Colocaremos en primer lugar un bloque para asignar un valor por defecto a la variable **Control_PWM** que es la que recoge el valor de activación de la señal de salida PWM (Bloque de Calculo al que hemos etiquetado como “*Fijar variable Control_PWM=0*”).



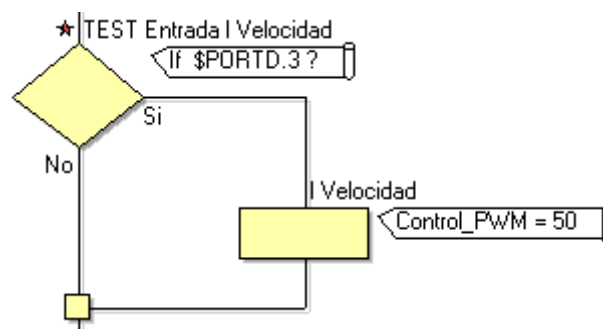
2. Seguidamente colocamos un bloque de **Macro** en el que activamos el componente **PWM** (bloque Macro al que hemos llamado *Llamar Macro Componente PWM*) seleccionando el parámetro **Enable**.



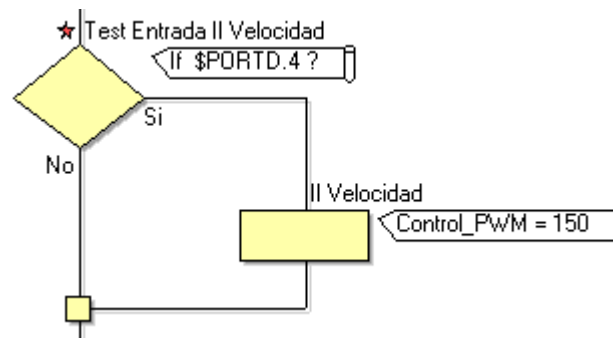
3. Una vez realizadas estas operaciones entraremos en el **Bucle** y allí llamamos a un bloque **Macro** que se encargara de controlar el funcionamiento del Componente PWM.



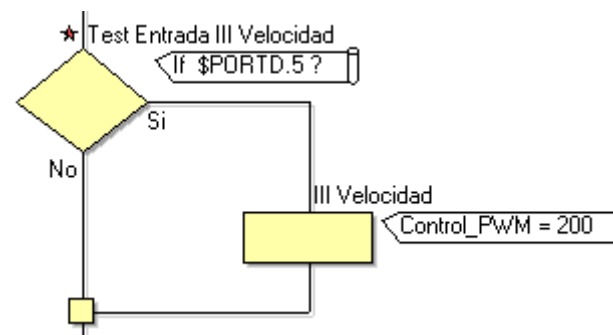
4. Se pregunta si el interruptor **Int1 (PIN3)** esta activado y si lo esta se asigna a **Control_PWM** el valor 50



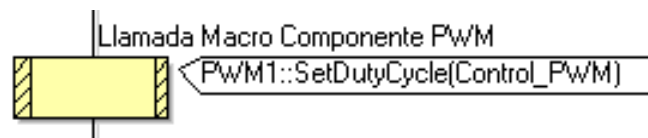
5. Se pregunta si el interruptor **Int2 (PIN4)** esta activado y si lo esta se asigna a **Control_PWM** el valor 150



6. Se pregunta si el interruptor **Int1 (PIN5)** esta activado y si lo esta se asigna a **Control_PWM** el valor 200

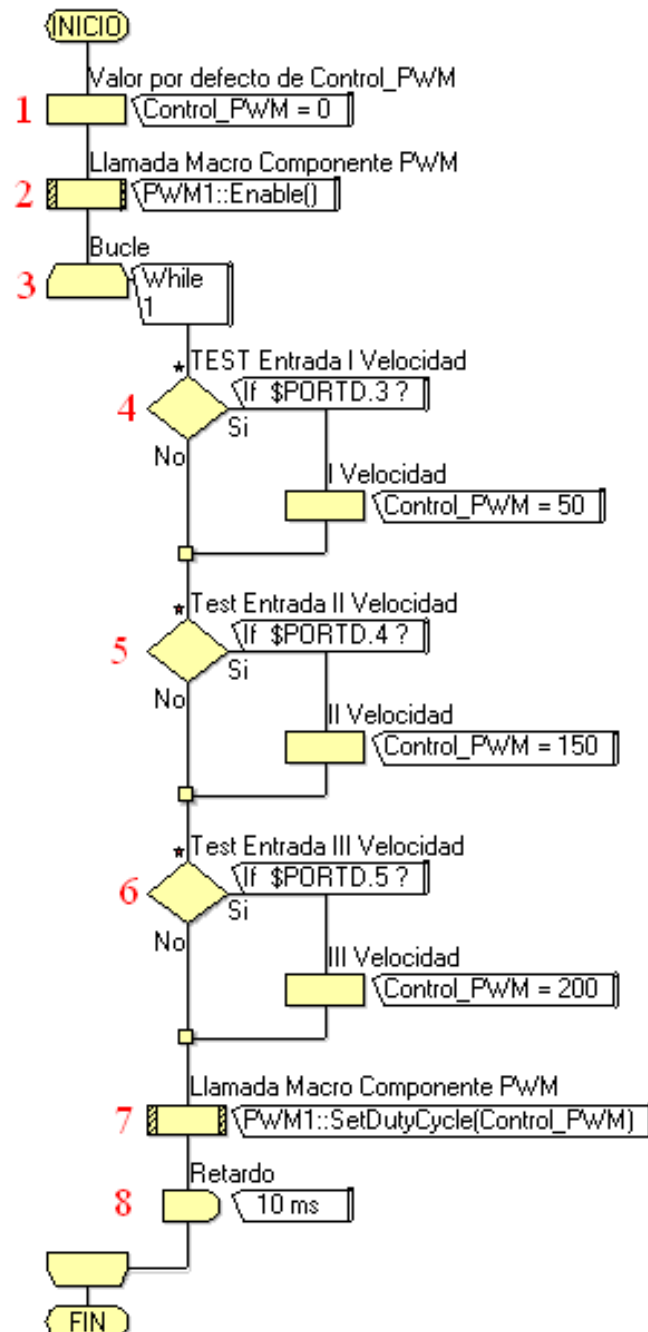


7. Con este nuevo bloque **Macro (Llamar Macro Componente PWM)** asociamos el parámetro **SetDutyCycle** a la variable **Control_PWM**



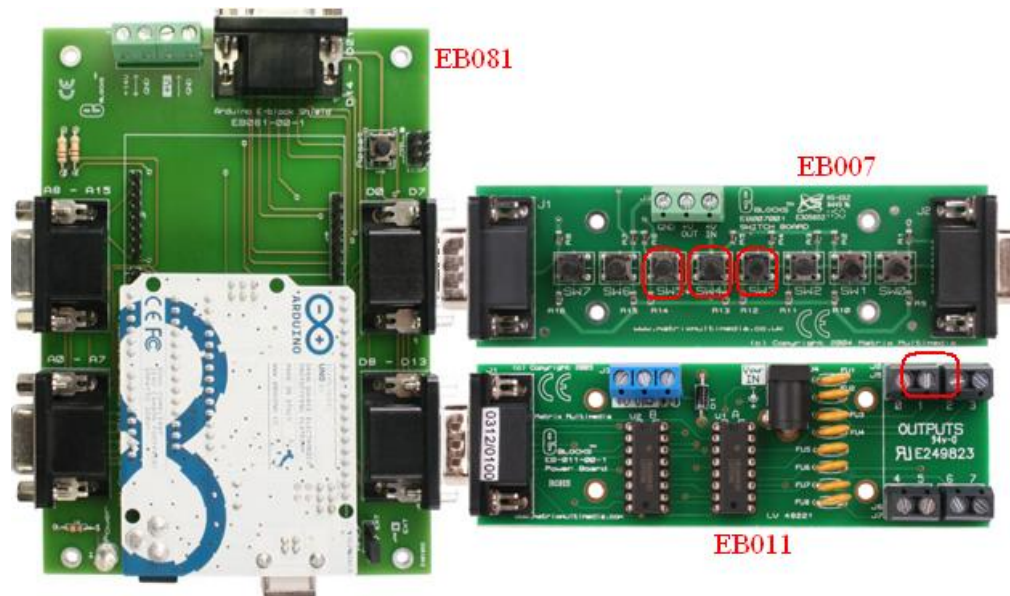
8. Par finalizar ponemos un bloque Demora que introduce un retardo de 10ms.





Este es diagrama de flujo completo.

Este es el montaje de la aplicación con los B-Blocks.



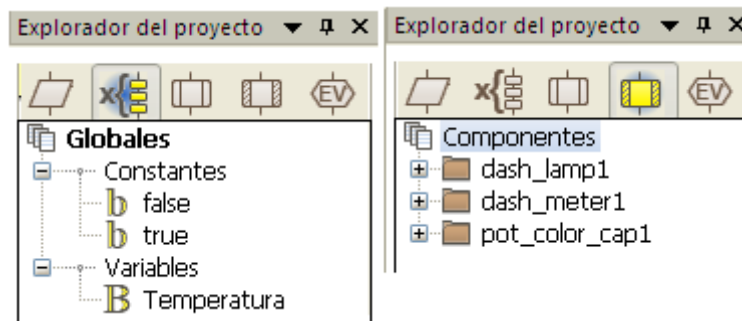
15. Termostato

Vamos a construir una aplicación mediante la cual se pretende controlar un elemento calefactor mediante un relé. Tomaremos una señal de consigna que recibiremos de un termostato y por otro lado leeremos la señal de una sonda. Dependiendo del resultado de la comparación de ambas señales gobernaremos una salida.

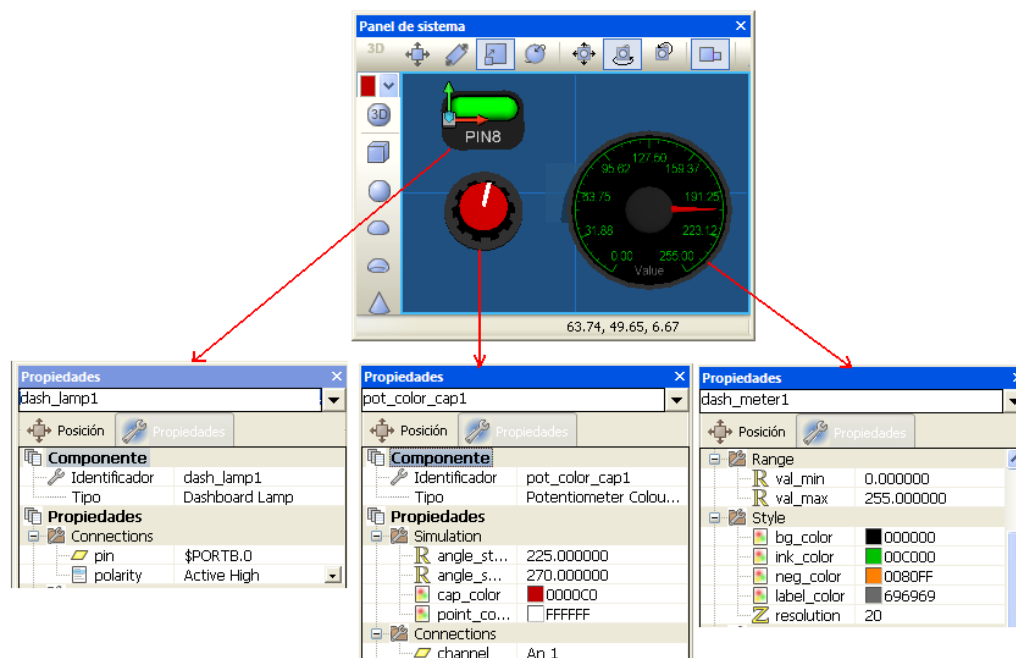
Se creará una variable que será la que recoja el valor de la temperatura: **Temperatura**

La designación de entradas y salidas es la siguiente:

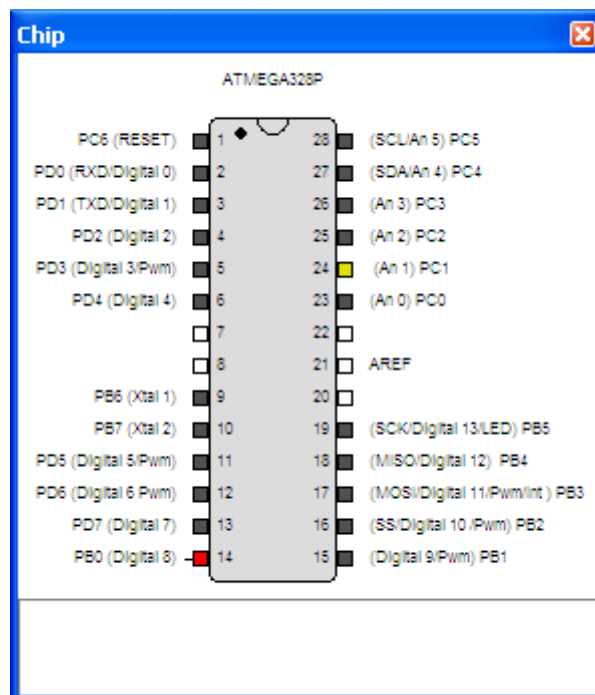
- Consigna termostato **AN1 (PC1)**
- Salida del relé del calefactor **PIN8 (PB0)**



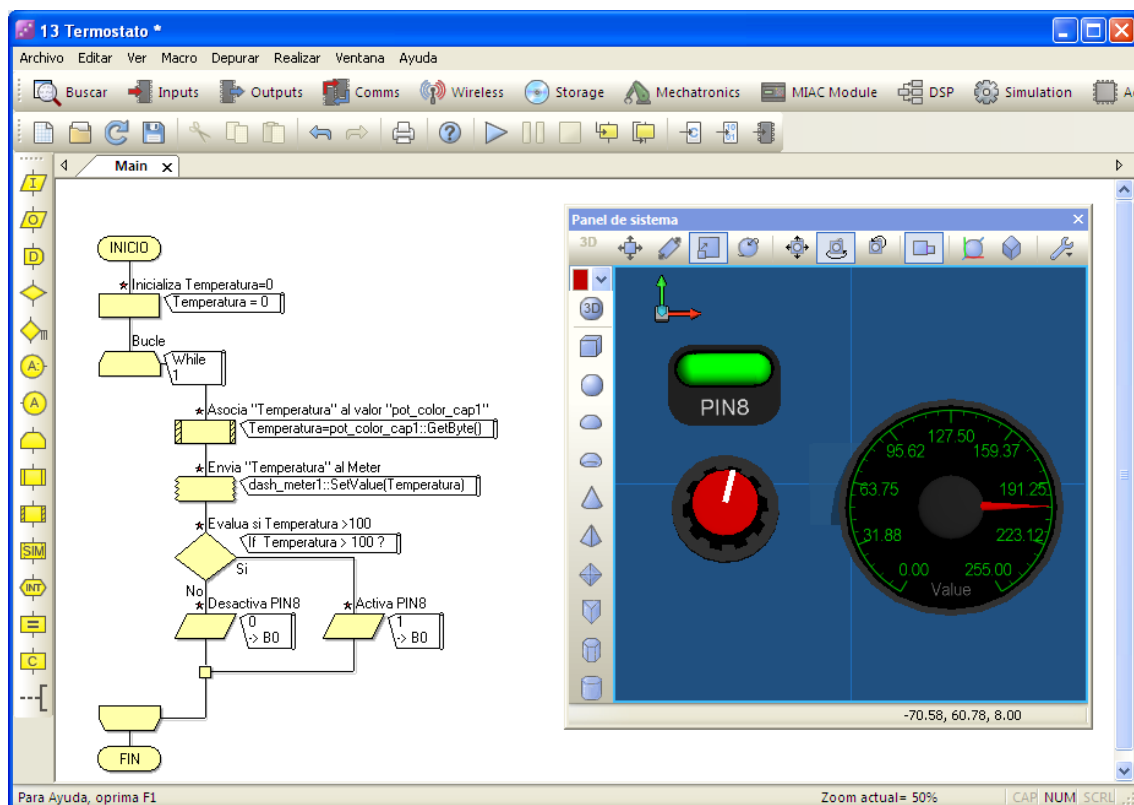
El Panel de sistema para la simulación se construirá colocando tres elementos: Un led (*dash_lamp1*), un potenciómetro (*pot_col_cap1*) y un visualizador analógico (*dash_meter1*). Los parámetros de estos componentes son los que aparecen en la siguiente figura.



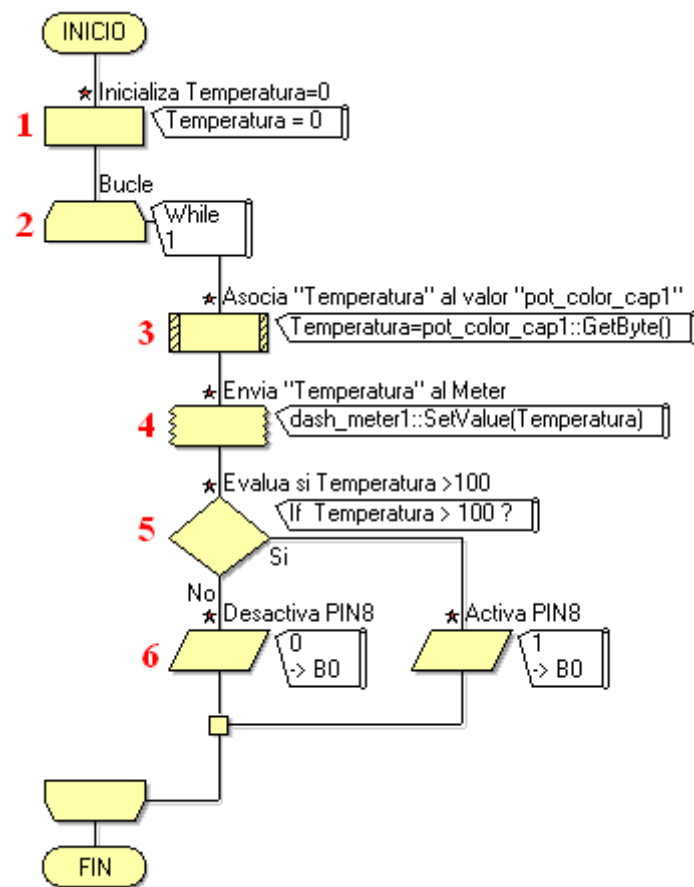
La designación de los pines en el Chip es la que se muestra en la figura.



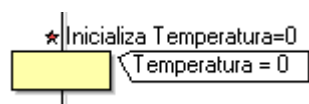
El aspecto de la aplicación será el que se muestra a continuación.



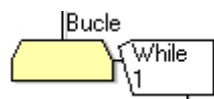
Las etapas y bloques que se deben insertar en el diagrama de flujo son los siguientes:



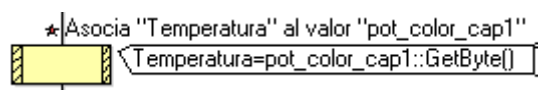
1. Colocaremos en primer lugar un bloque de **Cálculo** para asignar un valor por defecto a la variable **Temperatura** (Bloque de Calculo al que hemos etiquetado como “*Inicializa Temperatura=0*”).



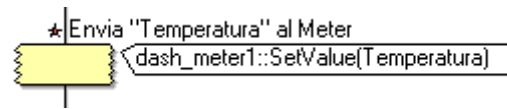
2. Entraremos en el **Bucle**.



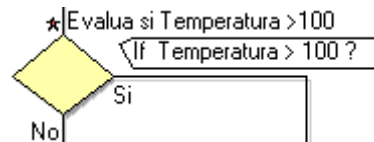
3. Mediante una **Macro de Componente** se asocia el valor de **Temperatura** al parámetro *GetByte* del potenciómetro.



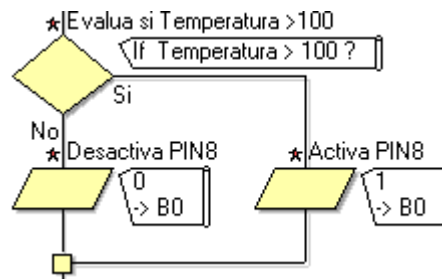
4. Mediante una **Macro de Simulación** enviamos el valor **Temperatura** al medidor analógico (parámetro *SetValue*).



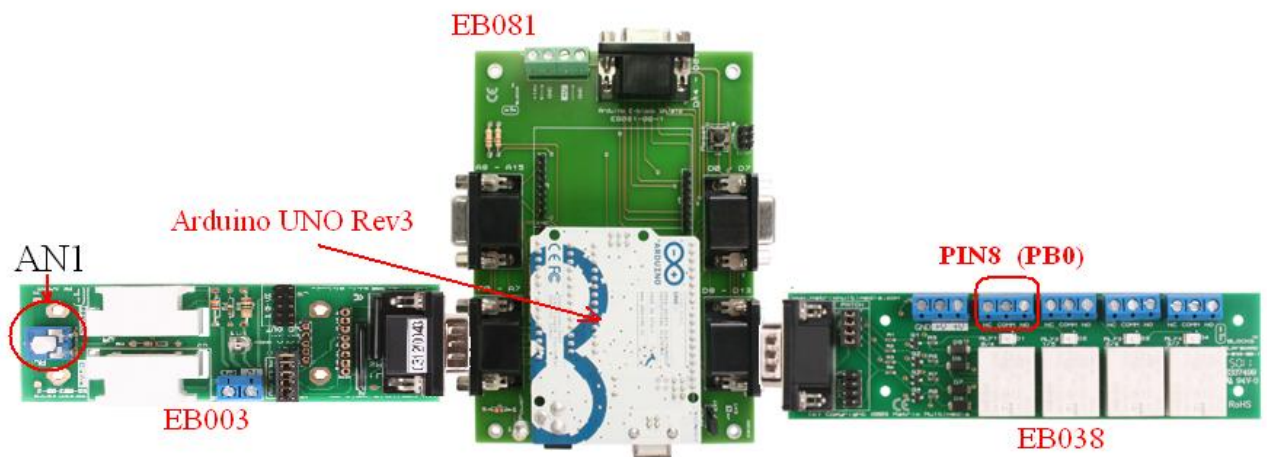
5. Con la ayuda del bloque Decisión, preguntando si **Temperatura > 100** estableceremos la activación o desactivación de la salida.



6. En este nivel aparecen dos bloques de **Salida** que ponen en "0" o en "1" la salida **B0 (PIN8)** dependiendo de si se cumple o no la condición establecida en el bloque anterior.



Finalmente se muestra el montaje con los E-Blocks correspondientes



Agradecimiento: Quiero expresar mi agradecimiento a la firma **Matrix Multimedia Ltd** <http://www.matrixmultimedia.com> por haberme facilitado la utilización tanto de su software **Flowcode 6** como de los bloques de la serie **E-Blocks** que he enumerado anteriormente. Especial reconocimiento a la persona: Carl Hegarty R&D Manager de Matrix Multimedia