

## Tema 4: Entorno de Desarrollo

### Sistemas Digitales Basados en Microprocesadores

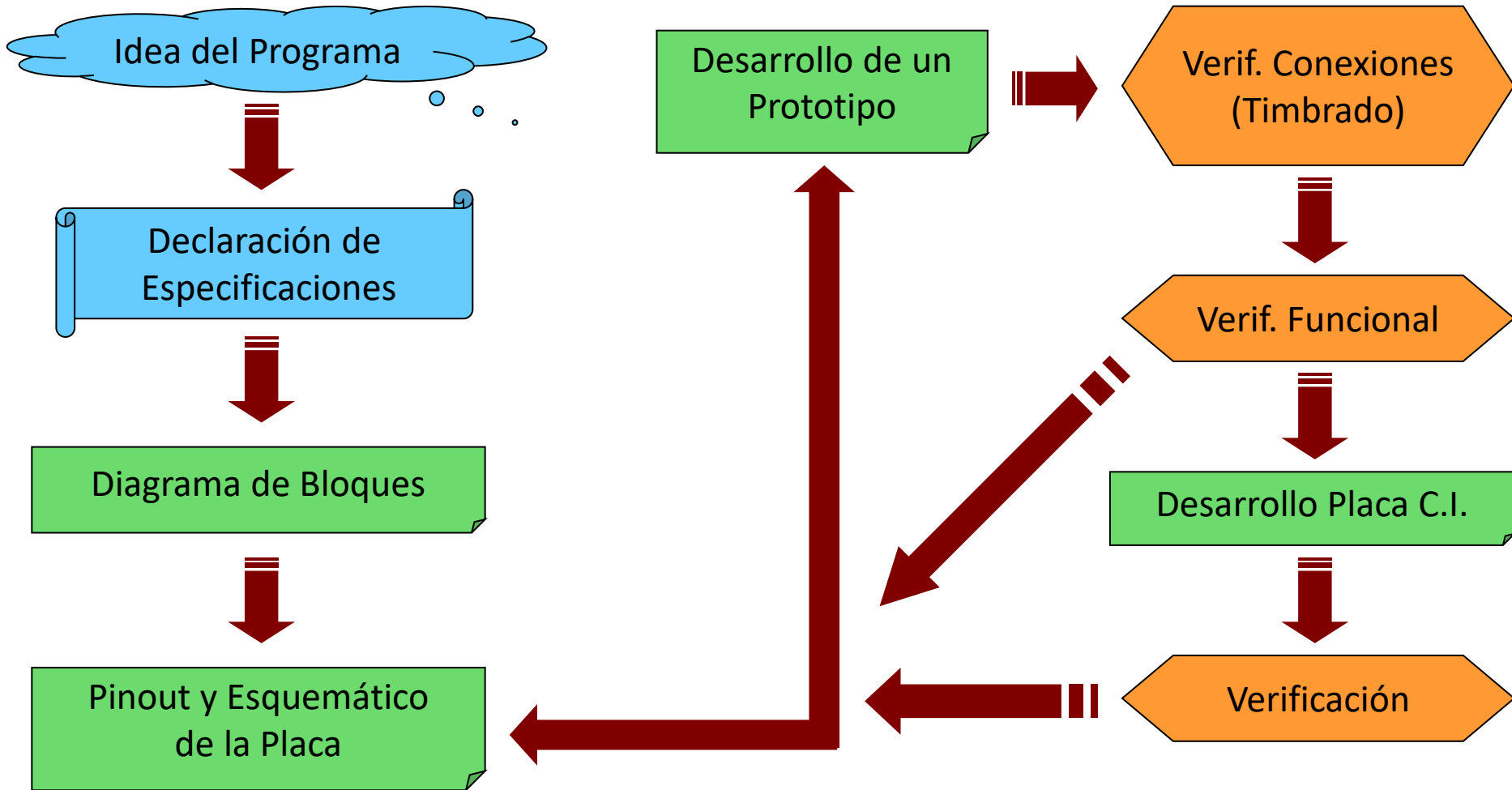
Universidad Carlos III de Madrid  
Dpto. Tecnología Electrónica

*Nota: Las figuras utilizadas para ilustrar las características y funcionalidades del microcontrolador del curso y del entorno de desarrollo se han obtenido de la documentación técnica disponible en <https://www.st.com/>*

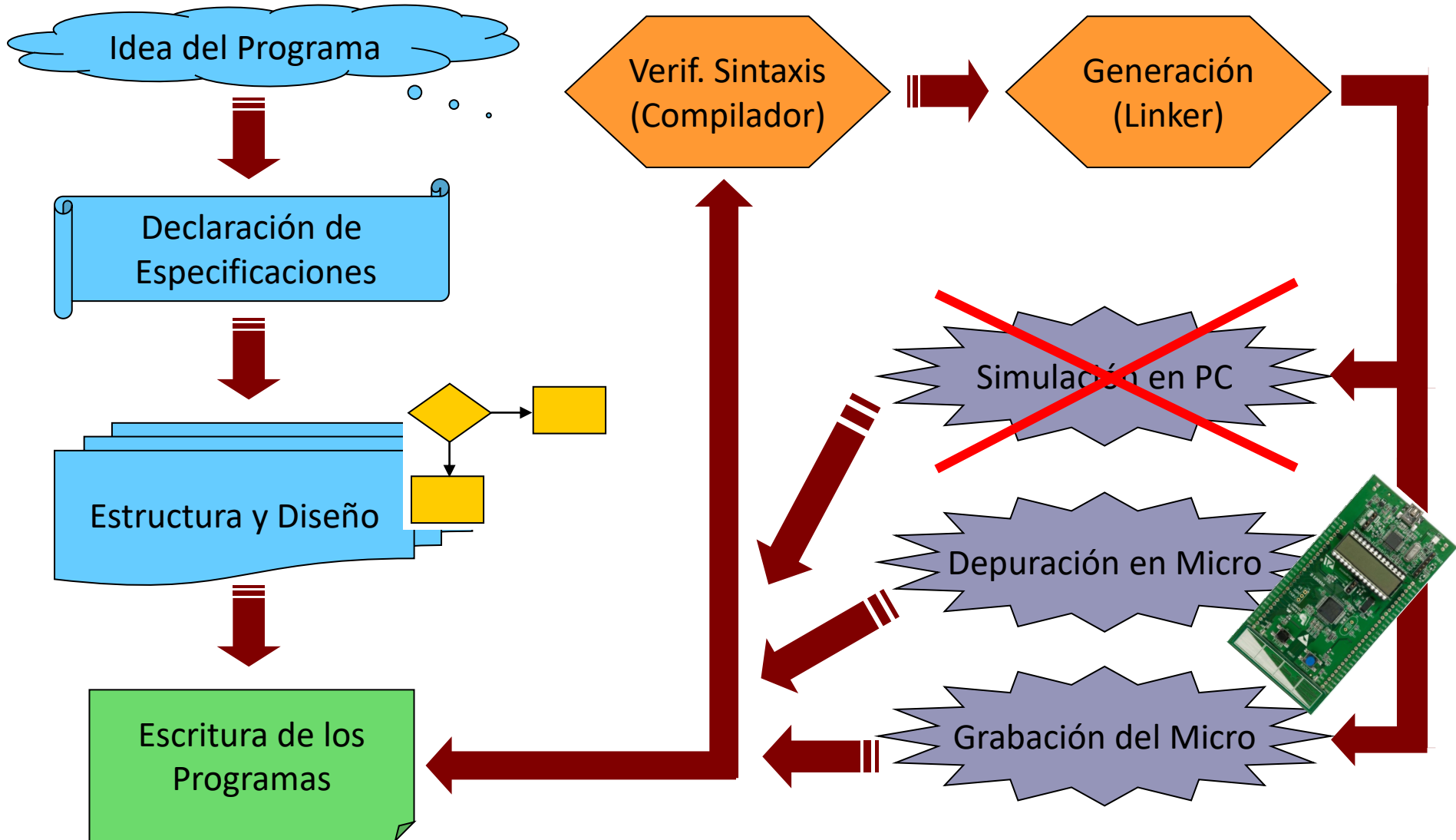
- Ciclo de Desarrollo
- Diagramas de Flujo
- La placa de Desarrollo STM32L-DISCOVERY
- El entorno de trabajo
- Pasos para la creación de un proyecto
- Pasos para la depuración de un proyecto
- Peculiaridades de la Programación en C en Microcontroladores
- Recomendaciones de Uso de la Placa de Desarrollo

# Ciclo de Desarrollo

# Ciclo de Desarrollo Hardware



# Ciclo de Desarrollo Software



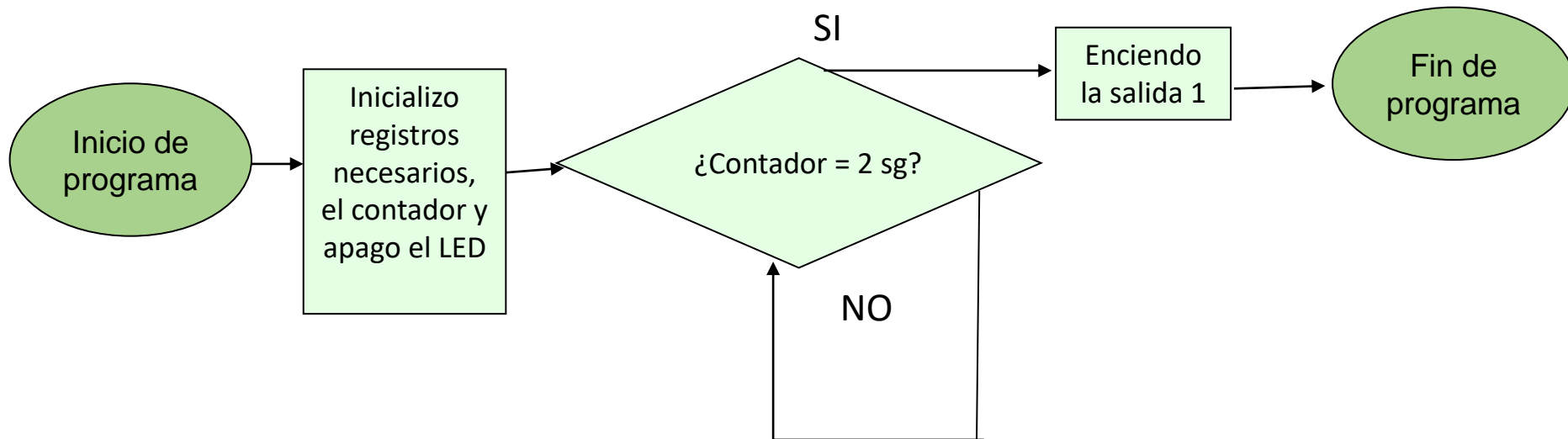
# Diagramas de Flujo

# Diagramas de Flujo

- Son representaciones del funcionamiento de un programa
  - De forma genérica – independiente de la arquitectura
    - Nunca pueden contener referencias a registros de la arquitectura utilizada, ni a instrucciones de la CPU
  - Que muestran la solución al problema planteado
  - Que tienen que servir de guía, tanto al programador como a los posibles programadores que tengan que tocar ese programa
- Se pueden escribir a distintos niveles de detalle/abstracción
  - El nivel de detalle que debe ser utilizado dependerá de la situación
- Tradicionalmente se utilizarán sólo los símbolos sencillos:
  - Elipse o círculo, para indicar una etiqueta
  - Rectángulo, para indicar un proceso
  - Rombo, para indicar una decisión

# Ejemplo

Programa que espera 2 segundos para encender un LED en la salida digital PA1 del micro y luego la deja encendida para siempre





# La placa de Desarrollo STM32L- DISCOVERY



## STM32L-DISCOVERY

### STM32L ultralow power discovery board

Data brief

#### Features

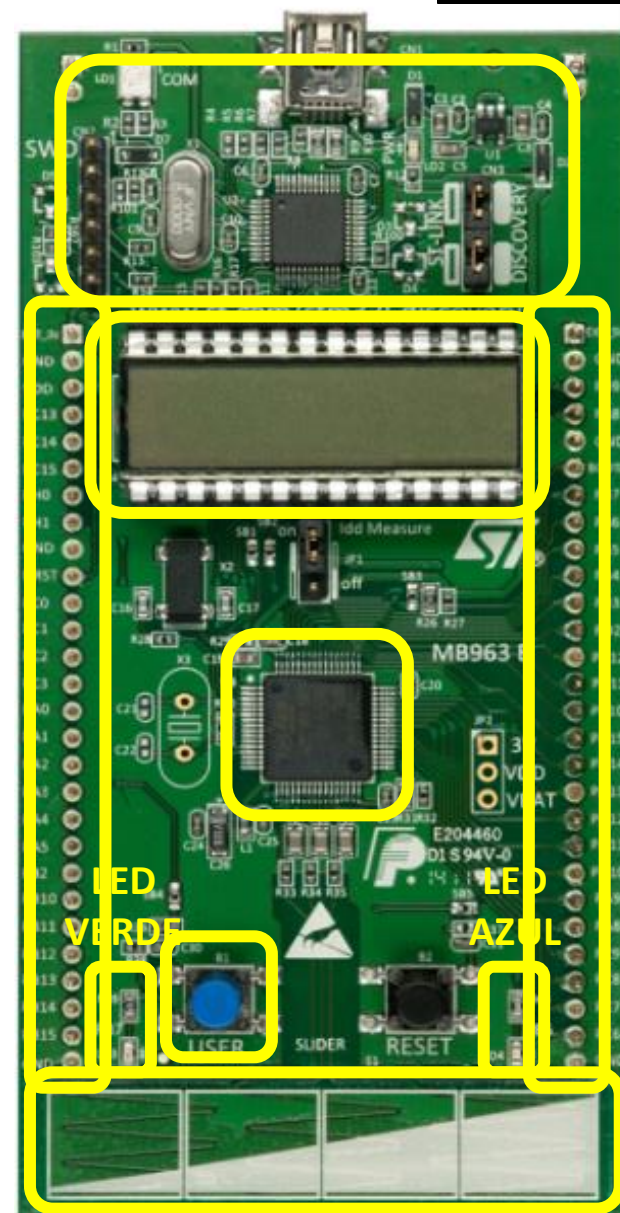
- STM32L152RBT6 microcontroller featuring 128 KB Flash, 16 KB RAM, 4 KB EEPROM, in an LQFP64 package
- On-board ST-Link/V2 with selection mode switch to use the kit as a standalone ST-Link/V2 (with SWD connector for programming and debugging)
- Board power supply: through USB bus or from an external 3.3 or 5 V supply voltage
- External application power supply: 3 V and 5 V
- I<sub>DD</sub> current measurement
- LCD
  - DIP28 package
  - 24 segments, 4 commons
- Four LEDs:
  - LD1 (red/green) for USB communication
  - LD2 (red) for 3.3 V power on
  - Two user LEDs, LD3 (green) and LD4 (blue)
- Two pushbuttons (user and reset)
- One linear touch sensor or four touchkeys
- Extension header for LQFP64 I/Os for quick connection to prototyping board and easy probing



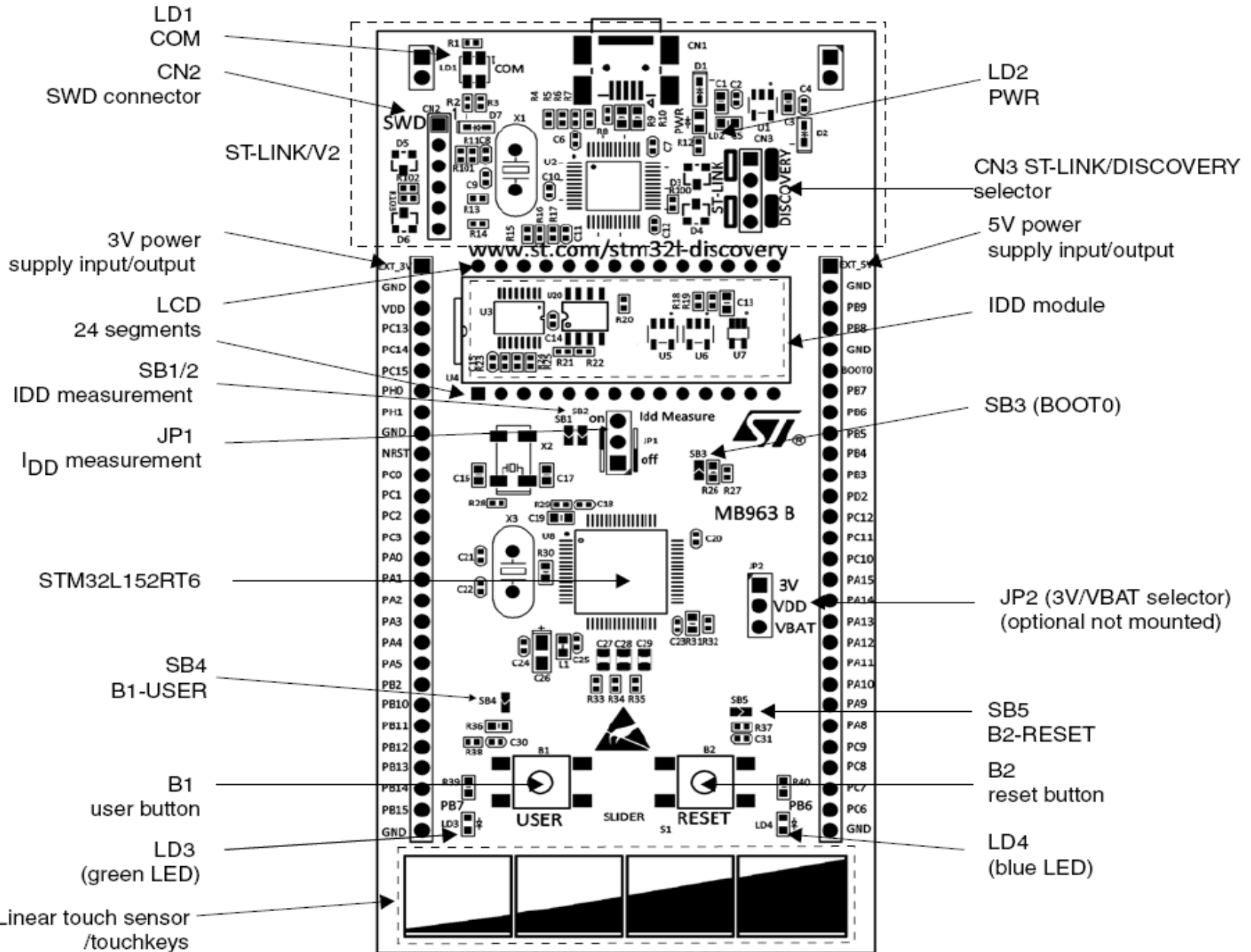
# STM32L-Discovery



- La placa de desarrollo tiene las siguientes funcionalidades:
  - Microcontrolador STM32L152RB
  - Interfaz de depuración ST-LINK/V2 incluido (conectado al ordenador a través de Mini-USB)
  - Una pantalla LCD de 24 segmentos y 4 comunes
  - 4 LEDs
    - 2 de ellos programables por el usuario (LED\_VERDE, LED AZUL)
  - Un sensor táctil lineal, con posibilidad de ser utilizado como 4 teclas individuales
  - Botón programable por el usuario (USER)
  - 2 Puertos de expansión a placa adicional (P1 y P2)



# D. de Bloques y Layout

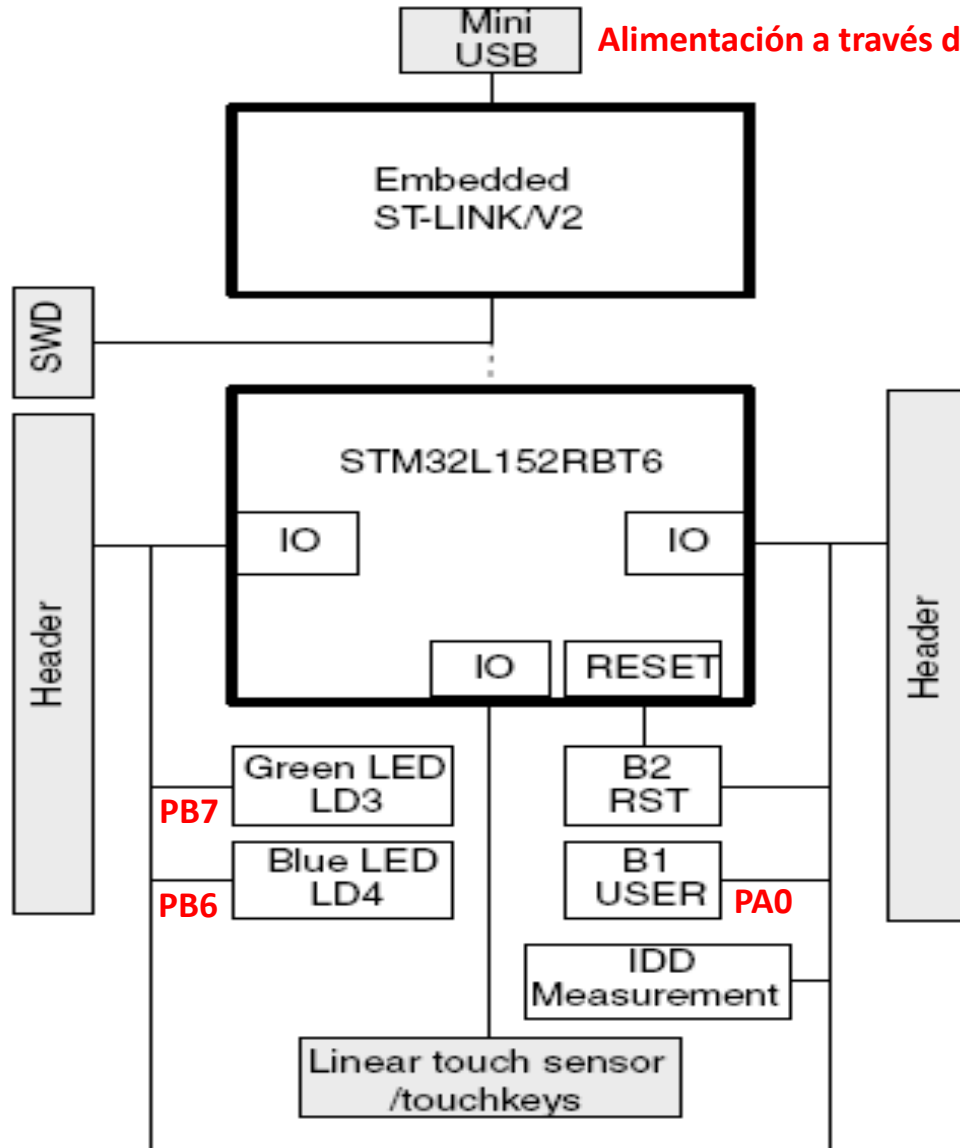


# Diagrama de Bloques y Layout

Alimentación a través del USB

- PUERTO P1:**
- 04 – PC13
  - 15 – PA0  
ADC\_IN0  
TIM2\_CH1\_ETR
  - 19 – PA4  
ADC\_IN4  
DAC\_OUT1
  - 20 – PA5  
SPI1\_SCK  
ADC\_IN5  
DAC\_OUT2  
TIM2\_CH1\_ETR

- PUERTO P2:**
- 07 – PB7  
I2C1\_SDA  
TIM4\_CH2  
USART1\_RX
  - 08 – PB6  
I2C1\_SCL  
TIM4\_CH1  
USART1\_TX
  - 12 – PD2  
TIM3\_ETR
  - 13 – PC12
  - 19 – PA12  
USBDP  
SPI1\_MOSI
  - 20 – PA11  
USBDM  
SPI1\_MISO



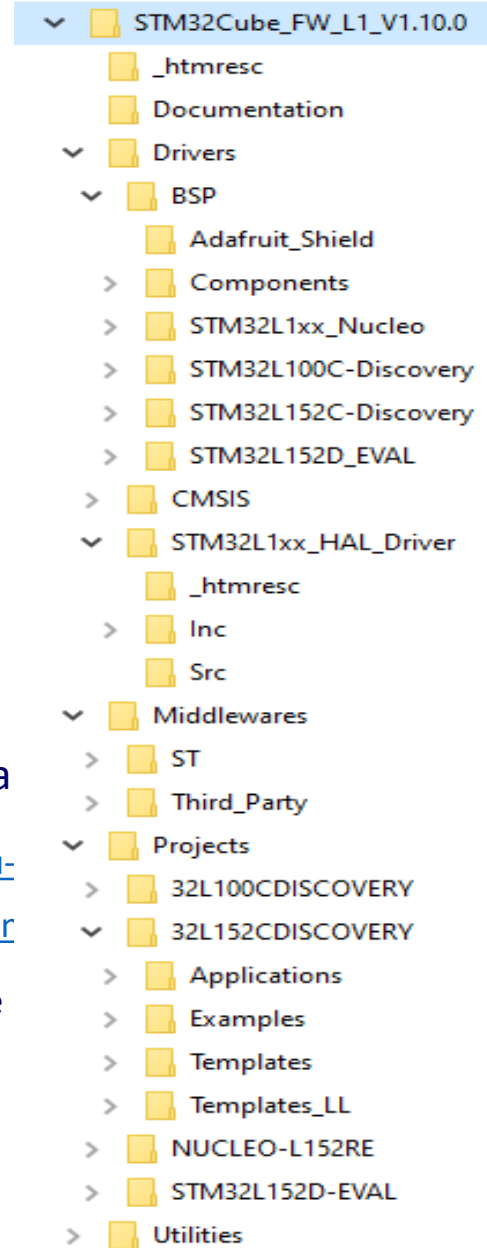
LCD (24 Segments, 4 Commons)

# Instalación de STM32CubeIDE



# Instalación

- Accede a la página:  
[https://www.st.com/content/st\\_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-ides/stm32cubeide.html](https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-ides/stm32cubeide.html)
- Selecciona la versión a descargarte: Debian, Linux genérico, macOS, Windows, etc.
- Si no tenías cuenta, ábretela y entra en ella
- Descarga el software en tu ordenador (más de 600MB)
- Descomprime y ejecuta el fichero de instalación
  - Indica que quieres instalar todos los sistemas de depuración, en especial el ST-LINK
  - Permite al sistema que instale los drivers correspondientes
  - Esto creará un icono en el escritorio
- También será necesario que se descargue las bibliotecas para la familia STM32L1 (denominado STM32CubeL1)
  - [https://my.st.com/content/my\\_st\\_com/en/products/embedded-software/mcu-mpu-embedded-software/stm32-embedded-software/stm32cube-mcu-mpu-packages/stm32cubel1.license=1603395499153.product=STM32CubeL1.version=1.10.0.html#overview](https://my.st.com/content/my_st_com/en/products/embedded-software/mcu-mpu-embedded-software/stm32-embedded-software/stm32cube-mcu-mpu-packages/stm32cubel1.license=1603395499153.product=STM32CubeL1.version=1.10.0.html#overview)
  - Esto contendrá, tanto las bibliotecas HAL (véase el tema 9), como las de uso de los periféricos incluidos en la placa STM32L-DISCOVERY (es decir, el BSP)



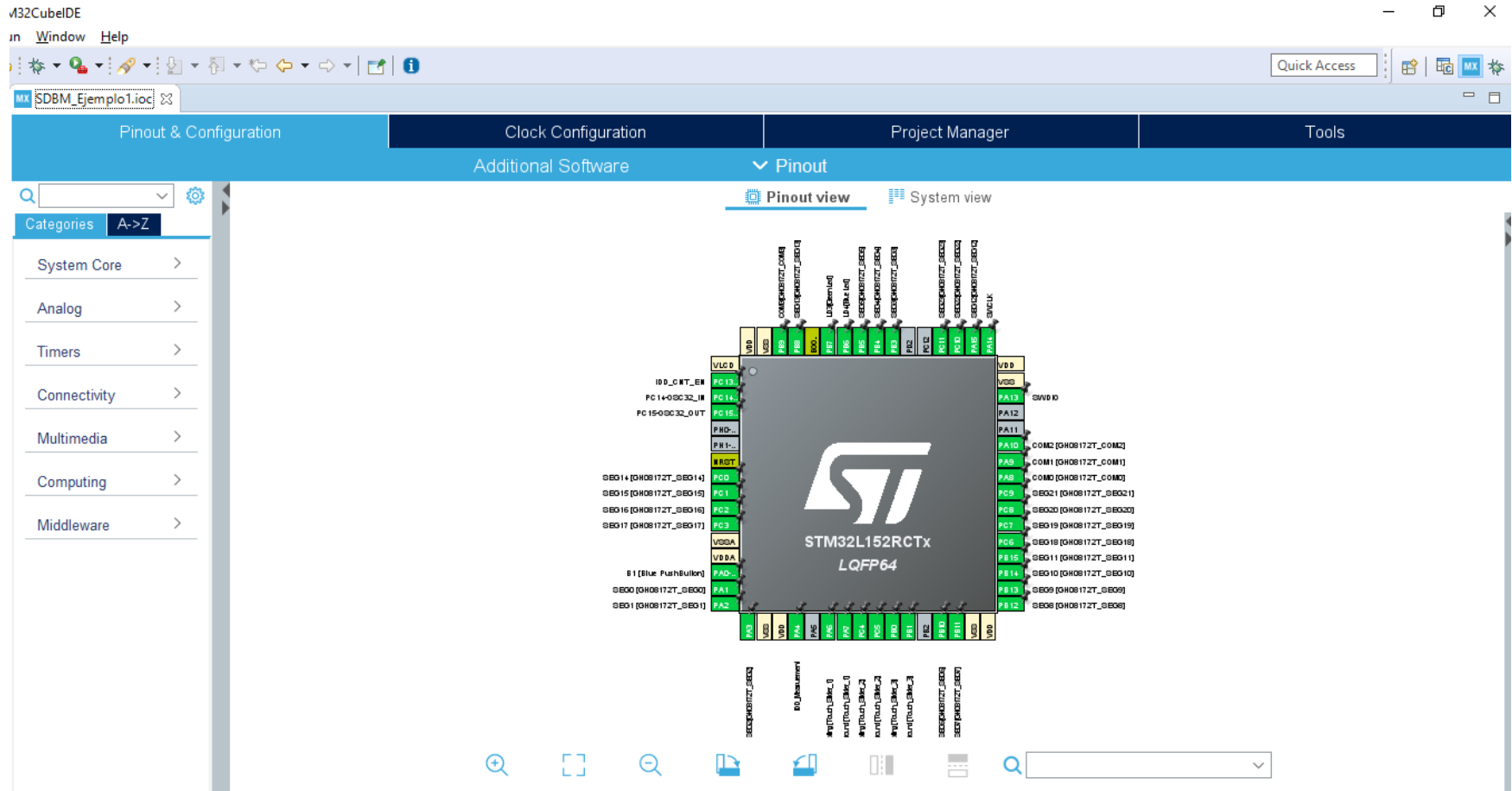
# Ejecución por primera vez

- Arranque la aplicación por primera vez:
  - Selecciona un directorio para la aplicación (puede ser el de por defecto)
  - Si pide permisos de acceso, concédalos
  - Si le comunica que hay actualizaciones, instálelas
  - En la pantalla inicial se encontrará con documentación y un tutorial
  - Intente crear un proyecto por primera vez (New -> STM32 Project), y una vez cargado el selector de dispositivos, seleccione la “Board” 32L152C DISCOVERY
    - Pulse Next
    - Póngale un nombre al proyecto
      - Puede elegir una nueva ubicación, o la de por defecto que puso antes
    - Seleccione un lenguaje (de momento C)
    - Seleccione que va a ser un ejecutable y que va a trabajar con STM32Cube
    - Inicialice los periféricos a su modo por defecto
    - Abra la perspectiva de CubeMX



# Vista de la Perspectiva CubeMX

- Le aparecerá la siguiente pantalla, en la cual ya podrá empezar a trabajar



# Trabajando con la Perspectiva de Cube



- Una vez que ha llegado a ver la pantalla anterior, puede empezar a configurar las funcionalidades del microcontrolador y de la placa que le interesen para su proyecto
- Para saber lo que debe modificar aquí, tendrá que haber determinado previamente:
  - Qué periféricos va a utilizar
  - Qué pines va a usar y con qué funcionalidad
  - Qué configuración va a querer utilizar para cada uno de esos pines y periféricos
- En esta perspectiva se puede:
  1. Configurar los pines y periféricos (lo necesitará hacer a lo largo del curso)
  2. Seleccionar el reloj del microcontrolador (se recomienda no hacerlo hasta tener un amplio conocimiento)
  3. Gestionar el proyecto (las opciones por defecto suelen ser las correctas)
  4. Grabar los cambios, para que el CubeIDE genere el código (o las modificaciones al código)
- Todo esto se verá en las siguientes transparencias mediante un ejemplo

# Configuración de pines y periféricos



- La pestaña de configuración de pines y periféricos se puede dividir en 3 zonas. Más adelante en el curso se explicará en más detalle.

STM32CubeMX Untitled: STM32L152RBTx - STM32L-DISCOVERY

File Window Help

Home / STM32L152RBTx - STM32L-DISCOVERY / Untitled - Pinout & Configuration

GENERATE CODE

### Pinout & Configuration

Options

Categories: A-Z

System Core

- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- TS
- WWDG

Analog

Timers

Connectivity

Multimedia

Computing

Middleware

### Clock Configuration

Additional Softwares

Pinout

### Project Manager

### Tools

#### GPIO Mode and Configuration

Configuration

Group By Peripherals

- GPIO
- Single Mapped Signals

Pin Name	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-up/...	Maximum out...	User Label	Modified
PA0-WKUP1	n/a	n/a	External Even...	No pull-up an...	n/a	B1 [Blue Pus...	<input checked="" type="checkbox"/>
PB6	n/a	Low	Output Push ...	No pull-up an...	Very Low	LD4 [Blue Led]	<input checked="" type="checkbox"/>
PB7	n/a	Low	Output Push ...	No pull-up an...	Very Low	LD3 [Green L...	<input checked="" type="checkbox"/>
PC13-WKUP2	n/a	Low	Output Push ...	No pull-up an...	Very Low	IDD_CNT_EN	<input checked="" type="checkbox"/>

Vista de los detalles del periférico seleccionado en el árbol

#### Pinout view

System view

Vista del chip para configurar funcionalidad de los pines

Select Pins from table to configure them. Multiple selection is Allowed.

# Configuración de reloj

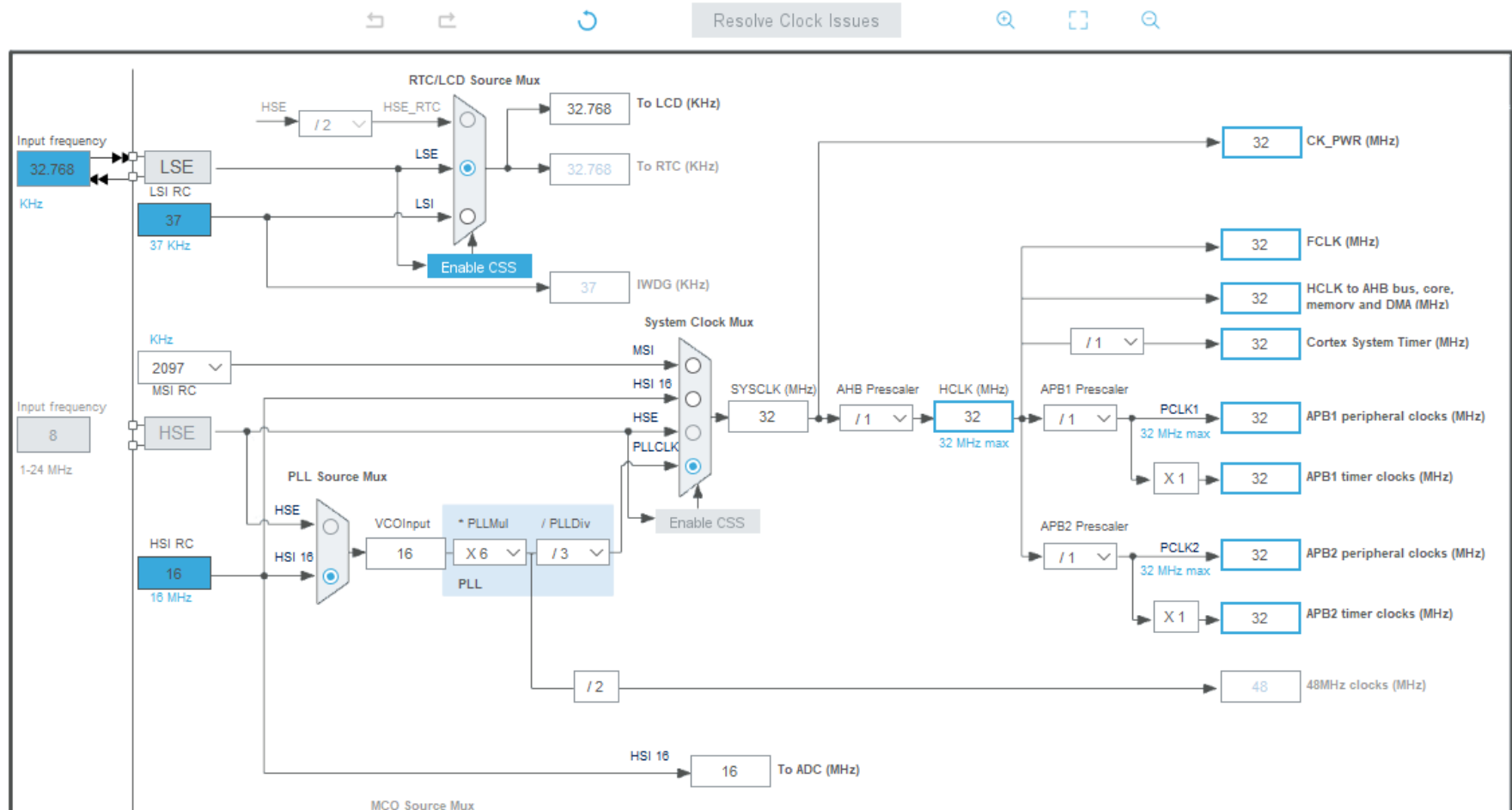
- La pestaña de configuración hay que dejarla por defecto, es decir, como en la figura, para tener reloj de 32MHz partiendo del HSI.

Pinout & Configuration

Clock Configuration

Project Manager

Tools



# Creación de la Estructura Básica de Trabajo

# Estructura Básica de Trabajo

- A lo largo del curso se van a hacer múltiples ejercicios, tanto de clase como de Laboratorio
- Cada uno de esos ejercicios será un Proyecto distinto
- Es preferible crear una estructura básica de trabajo, para así tenerlo todo ordenado
- Cree en el disco duro de su ordenador una carpeta para la asignatura (por ejemplo, D:\SDBM\
  - Se recomienda que NO se utilice el escritorio, ya que las rutas son mucho más largas y muchas veces con caracteres complejos (por ejemplo, espacios)
  - Puede ser interesante que al CubelDE se le indique que esa va a ser la carpeta de trabajo por defecto del entorno.
- Descomprima el paquete de bibliotecas del STM32CubeL1, y copie el BSP de la placa de desarrollo en la carpeta de la asignatura
  - El BSP es la carpeta STM32L152C-Discovery que se encuentra en la ruta \Drivers\BSP\ del paquete STM32CubeL1
  - Dentro de esa carpeta, se encontrarán 6 ficheros (entre ellos 2 .c y 2 .h)
  - Aquí se encuentra la biblioteca de uso del LCD, que se utilizará en casi todos los ejemplos
- A partir de aquí, siga los pasos de generación de proyecto que se indican a continuación

# Pasos para la creación de un proyecto

# Pasos para crear un Proyecto

- Hagámoslo de forma práctica, a través de un ejemplo:
  - El programa irá sacando secuencialmente, mensajes en el LCD, con un tiempo de espera apreciable (por ejemplo, 1 segundo), entre paso y paso:
    - Paso 1:
      - LCD = “UNO”
    - Paso 2:
      - LCD = “DOS”
    - Paso 3:
      - LCD = “TRES”
    - Paso 4:
      - LCD = “MAMBO”
- Antes de ponerse a trabajar, es necesario tener claro qué periféricos y qué características de los mismos se van a necesitar
  - En este caso sólo va a ser el LCD



# Uso del LCD de la DISCOVERY

- Si al crear el proyecto se ha seleccionado la placa 32L152CDISCOVERY, y se han inicializados los periféricos por defecto, no se debería hacer nada.
- En caso de haber manipulado algo, es importante saber que la configuración del LCD en la perspectiva de CubeMX:
  1. En el árbol se secciona Multimedia
  2. Se pincha en LCD
  3. A priori la configuración por defecto es correcta. Si no, cámbiala para que se cumpla lo de la figura:
    - Mode = 1/4 Duty Cycle
    - Pines por defecto
    - Clock Prescaler = 1
    - Clock Divider = 16
    - Bias Selector = 1/4
    - Voltage source = Internal
    - Contrast Control = 2.60V
    - Dead Time = No dead time
    - High Drive = Disable
    - Pluse ON Duration = 0 pulse
    - Blink Mode = Disabled
    - Blink Frequency fLCD/8
  - De a Grabar para re-generar

The screenshot shows the STM32CubeMX interface with the 'LCD Mode and Configuration' window open. The 'Mode' section is set to '1/4 Duty Cycle'. The 'Configuration' section shows the following parameters:

Parameter	Value
Clock Prescaler	1
Clock Divider	16
Duty Selection	1/4
Bias Selector	1/4
Multiplex mode	Disable
Voltage Source Selection	Internal
Contrast Control	2.60V
Dead Time Duration	No dead Time
High Drive	Disable
Pulse ON Duration	0 pulse
Blink Mode	Disabled
Blink Frequency	fLCD/8

# Uso del LCD de la DISCOVERY

- Para poder utilizar el LCD, hay que añadir el BSP al proyecto.
- Esto se puede hacer de varios modos, aunque el más cómodo es el de añadir la carpeta del BSP (la que se ha copiado en nuestra carpeta de trabajo como STM32I152C-Discovery) en las rutas de búsqueda del proyecto:
  - Seleccione *“Project”* -> *“Properties”*
  - Vaya a *“C/C++ General”* y desde allí a *“Paths and Symbols”*
  - En la pestaña *“Includes”*, pulse *“Add...”*, después *“File system...”* y seleccione la carpeta donde está el BSP
  - En la pestaña *“Source Location”*, pulse *“Link Folder...”*, haga click en *“Link to folder in the file system”*, luego *“Browse...”* y seleccione la misma carpeta donde está el BSP
- Además, tiene que incluir los archivos .h de esa biblioteca dentro del main.c, en la sección:

```
/* USER CODE BEGIN Includes */  
#include "stm321152c_discovery.h"  
#include "stm321152c_discovery_glass_lcd.h"
```
- A partir de ahí puede usar funciones como:
  - `BSP_LCD_GLASS_Init();`
  - `BSP_LCD_GLASS_BarLevelConfig(0);`
  - `BSP_LCD_GLASS_Clear();`
  - `BSP_LCD_GLASS_DisplayString((uint8_t *)“texto”);`

# Programando el Microcontrolador

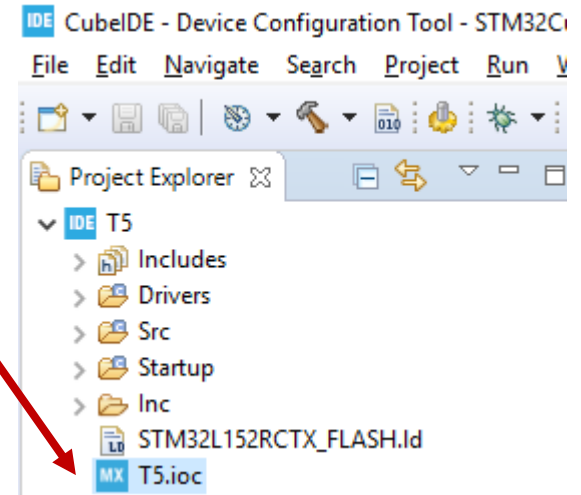
- Todo programa de microcontrolador se basa en 2 partes fundamentales de la función main():
  - Una parte de inicialización de los periféricos, variables, etc.
    - Se ejecutará una única vez al principio del programa
  - Una parte de código metida dentro de un bucle infinito, que representará la funcionalidad continua
    - Ese bucle infinito se realiza con un `while(1) { ... }`
- Por lo tanto su trabajo constará en escribir dos partes de código:
  - Una de inicialización:
    - Se escribirá en la función main(), justo después de la línea
      - `/* USER CODE BEGIN 2 */`
  - Una de funcionamiento del programa:
    - Se escribirá justo después de la línea que contiene la llave que se abre tras el `while(1)`
      - Antes de la línea `/* USER CODE END WHILE */`
- **Es muy importante que escriba el código en esas secciones, porque así, si se modifica a posteriori algo en el Cube y se regenera el proyecto, no se borrará**

# Crear un proyecto

- Arrancar el CubeIDE
- Seleccionar File -> New -> STM32 Project
- Esto arranca el “Target Selector”
  - Elije la placa a utilizar (por ejemplo 32L152C-DISCOVERY)
  - Pulsa Next
  - Pon un nombre de Proyecto
  - De momento elija lenguaje C, binario Executable y Tipo de Proyecto STM32Cube
  - Pulse “Finish”
  - Diga que inicialice a modo por defecto
  - Permita que cree la perspectiva STM32CubeMx
- El proyecto se inicia con la perspectiva de CubeMX

# Crear un proyecto

- Haz los cambios que crea necesarios en la perspectiva de CubeMX y de a grabar, para que se genere el código
- Si a lo largo del desarrollo del proyecto, quiere volver a la perspectiva de CubeMX, simplemente seleccione en el árbol del proyecto el fichero <Nombre proyecto>.IOC
- El código generado tiene el aspecto de las siguientes transparencias
  - SÓLO DEBE ESCRIBIR CÓDIGO ENTRE LAS LÍNEAS COMENTADAS DE **USER CODE BEGIN** Y **USER CODE END** (es decir, en lo marcado en azul)
    - Cada sección está pensada para un tipo de código distinto
  - Si escribe en cualquier otro sitio, al re-generar el código el compilador lo borrará



# Esquema del código generado

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */
```

**Descripción y Título del Proyecto (informativo)**

```
/* Includes -----*/
#include "main.h"
```

```
/* Private includes -----*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
```

**#include's del usuario**

```
/* Private typedef -----*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
```

**Definición de tipos de datos propios**

```
/* Private define -----*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
```

**#define's del usuario**

```
/* Private macro -----*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
```

**Macros del usuario**

```
/* Private variables -----*/
ADC_HandleTypeDef hadc;
LCD_HandleTypeDef hlcd;
```

```
/* USER CODE BEGIN PV */
/* USER CODE END PV */
```

**Variables globales del usuario**

```
/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC_Init(void);
static void MX_LCD_Init(void);
static void MX_TS_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
```

**Prototipos de funciones del usuario (implementadas, p. ej. en bloques 0 o 3)**

```
/* Private user code -----*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
```

**Funciones definidas por el usuario (por ejemplo, RAls o Callbacks)**

```
int main(void)
{
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
```

**Código previo a la inicialización del micro (p. ej. variables del main)**

```
/* MCU Configuration -----*/
/* Reset of all peripherals, ... */
HAL_Init();
```

**Código de inicialización previo a la configuración del reloj**

```
/* USER CODE BEGIN Init */
/* USER CODE END Init */
```

```
/* Configure the system clock */
SystemClock_Config();
```

```
/* USER CODE BEGIN SysInit */
/* USER CODE END SysInit */
```

**Código de inicialización del sistema**

# Esquema del código generado

```
/* Initialize all configured peripherals */
```

```
MX_GPIO_Init();
MX_ADC_Init();
MX_LCD_Init();
MX_TS_Init();
```

Inicialización y configuración del sistema posterior a la configuración automática de periféricos

```
/* USER CODE BEGIN 2 */
/* USER CODE END 2 */
```

```
/* Infinite loop */
```

Código del programa principal (funcionamiento cíclico del sistema)

```
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
```

Código adicional del usuario dentro o fuera del bucle principal, pero dentro del main()

```
...
/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Otro lugar para funciones definidas por el usuario (por ejemplo, RAIs o Callbacks)

```
...
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
```

- Es aconsejable entrar en Window->Preferences, para cambiar el comportamiento del editor:
  - En General->Editors->Text Editors seleccione la opción “Insert spaces for tabs” y ponga 2 en la “Displayed tab width”
  - Para ser compatible con el código generado por CubeMX, cambie las preferencias de indentación en C/C++ -> “Code Style” -> “Formater”, y allí:
    - Seleccione como “Active profile” el “BSD/Allman”
    - Pulse Edit y en Indentación cambie “Tab policy” a “Spaces only”, e “Indentation size” a 2 igual que “Tab size” a 2
    - Para grabar los cambios, debe cambiar el nombre del Perfil por otro cualquiera (por ejemplo, SDBM)
    - Pulse Apply y luego OK
    - Asegúrese que el nuevo perfil es el que está seleccionado como activo
    - Pulse “Apply and Close” para cerrar las preferencias.



# Compilar y Depurar

- Una vez haya escrito el código, puede pasar a Compilarlo (“Build”)
- Cuando ya no tenga errores, pase a Depurarlo (“Debug” -> STM32 MCU C/C++ Application -> OK)
  - Si le pide confirmación para cambiar de perspectiva, pulse “Switch”
    - Puede seleccionar que recuerde la decisión para que no le vuelva a aparecer este diálogo
  - Aquí utilice las múltiples funcionalidades disponibles:
    - Ejecución paso a paso
    - Puntos de ruptura
    - Examen de variables
    - Etc.

# El Código del Ejemplo

- Este proyecto va a utilizar:
  - El LCD
    - Deberá incluir el BSP de la placa STM32L152C-Discovery
  - Una biblioteca propia
    - Que se explicará su creación e uso en las siguientes transparencias
- De momento, veamos el código a escribir en el archivo main.c
- La inicialización es:

```
/* USER CODE BEGIN 2 */  
BSP_LCD_GLASS_Init();  
BSP_LCD_GLASS_BarLevelConfig(0);  
BSP_LCD_GLASS_Clear();  
/* USER CODE END 2 */
```

# El Código del Ejemplo

- La fase de ejecución continua es:

```
/* USER CODE BEGIN WHILE */
while (1)
{
    BSP_LCD_GLASS_Clear();
    BSP_LCD_GLASS_DisplayString((uint8_t *) "UNO");
    espera(5000000);
    BSP_LCD_GLASS_Clear();
    BSP_LCD_GLASS_DisplayString((uint8_t *) "DOS");
    espera(5000000);
    BSP_LCD_GLASS_Clear();
    BSP_LCD_GLASS_DisplayString((uint8_t *) "TRES");
    espera(5000000);
    BSP_LCD_GLASS_Clear();
    BSP_LCD_GLASS_DisplayString((uint8_t *) "MAMBO");
    espera(5000000);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

# Creación de una biblioteca propia

- Como se puede ver en el código, hay una función que no está definida: `espera()`
  - Esa función será de utilidad en varios proyectos futuros, por lo que se va a crear una biblioteca de funciones propias
- Los siguientes pasos muestran cómo hacerlo, pero no se mostrará el código ya que es un ejercicio propuesto
- Como la biblioteca va a ser utilizada en muchos proyectos, es preferible crearla manualmente en un directorio fácilmente accesible y luego añadirla al proyecto
  - Aunque formalmente no es muy recomendable, puede resultar cómodo incluir dicha biblioteca dentro del mismo directorio del BSP de la placa
    - Si hace esto, recomendamos que se cambie el nombre de la carpeta, para identificar que es una carpeta que no sólo tiene lo suministrado por el fabricante.
- Empezamos creando con un editor de textos cualquiera, un fichero `.c` y un `.h` cualquiera (pueden estar vacíos)
  - Se recomienda que se haga en la carpeta de trabajo, para así encontrarlo fácilmente en todos aquellos proyectos que se necesite

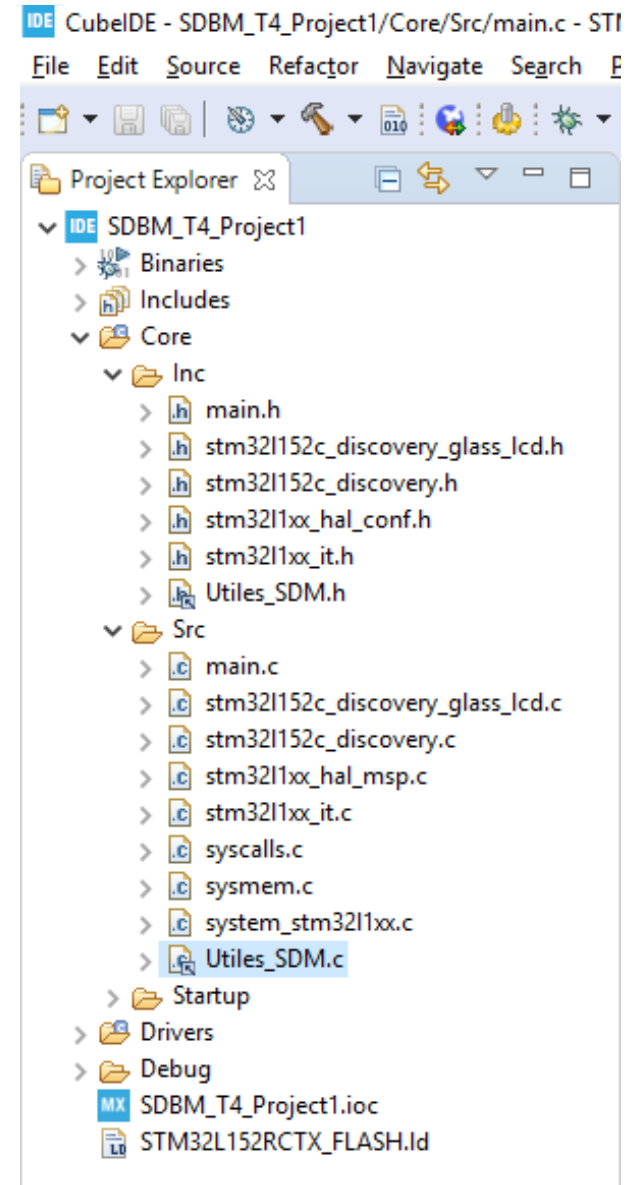
# Creación de una biblioteca propia

- Añada la biblioteca al proyecto de la misma forma que ha añadido la del BSP de la placa
  - Si al final ha optado por crearla en la misma carpeta que el BSP, ya será accesible sin hacer nada más.
- Por ejemplo, si ha llamado a la biblioteca “Utiles\_SDM”, el árbol del proyecto debe tener este aspecto
  - Incluyendo la importación de la biblioteca de la Discovery
- El siguiente paso es añadir el fichero .h de la biblioteca a la sección de Includes de usuario del main.c

```

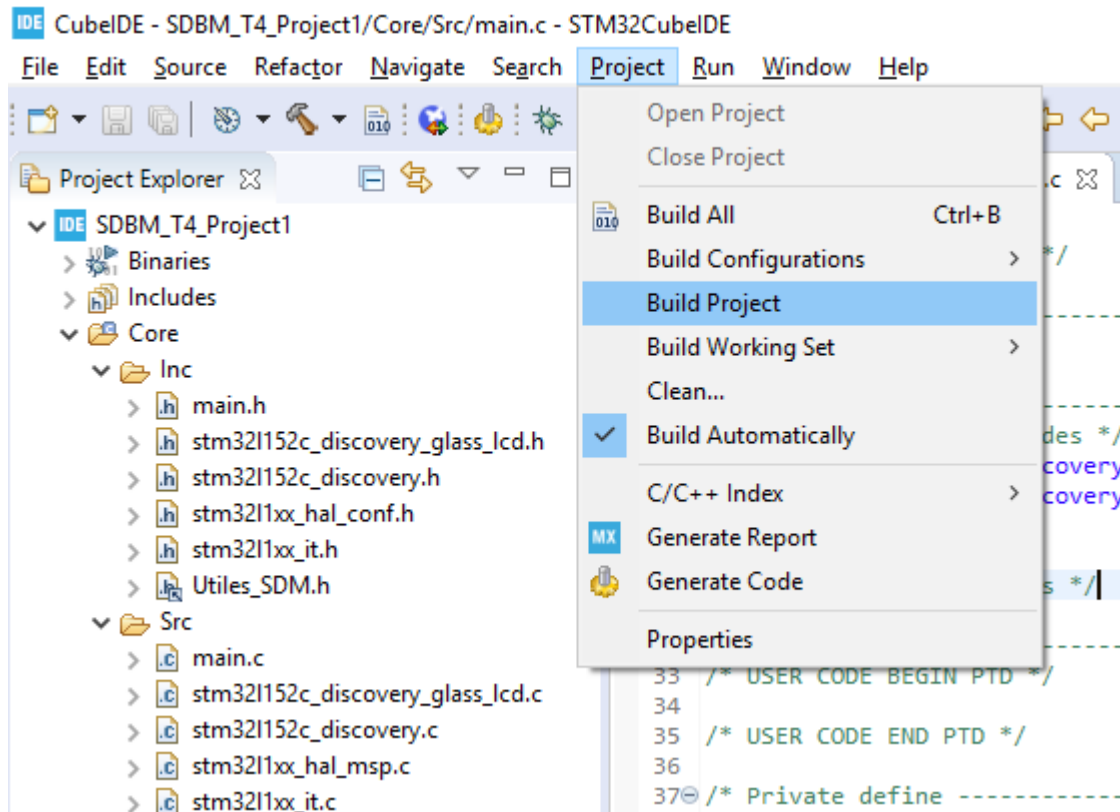
/* USER CODE BEGIN Includes */
#include "stm321152c_discovery.h"
#include "stm321152c_discovery_glass_lcd.h"
#include "Utiles_SDM.h"
/* USER CODE END Includes */

```



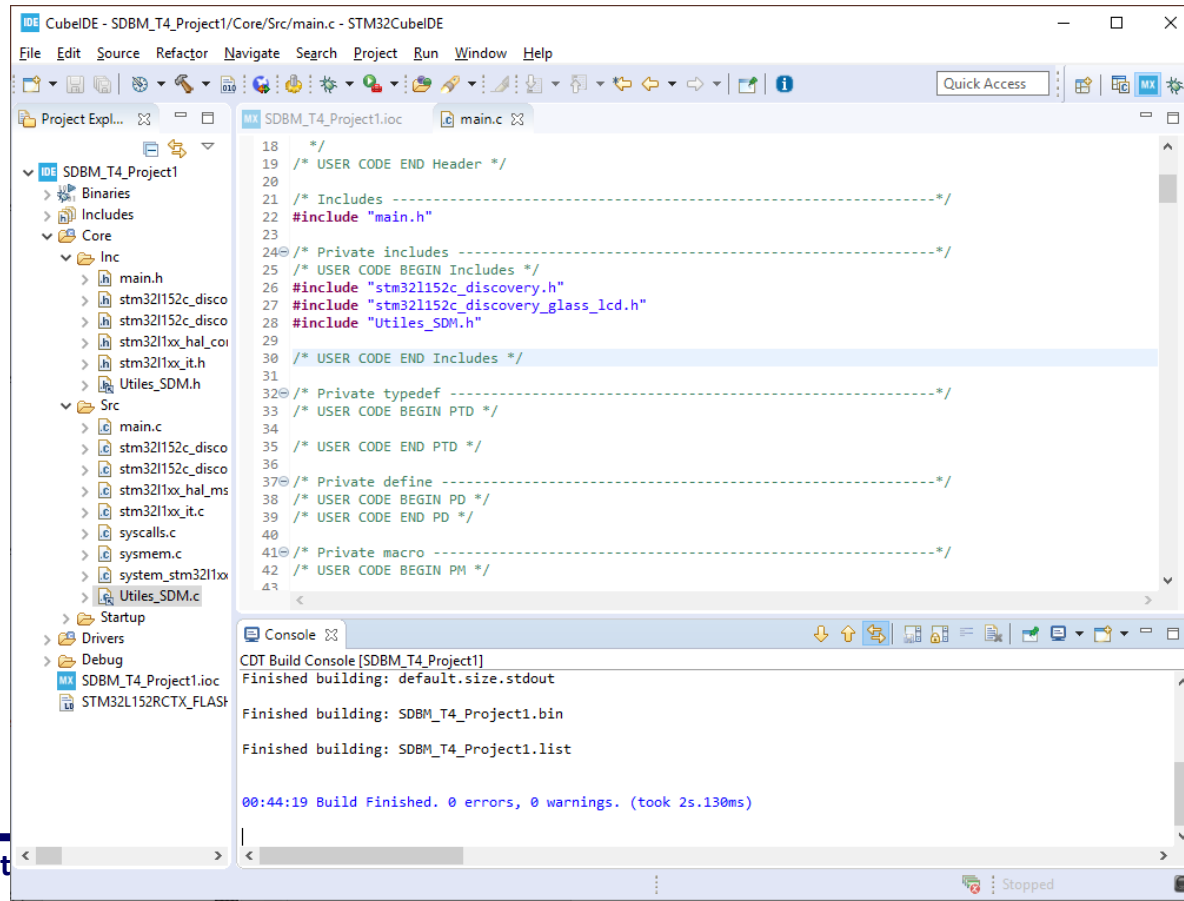
# Compilación

- El proyecto ya está listo para compilar
- Seleccione Project -> Build Project



# Compilación


- Durante la compilación verá aparecer mensajes en la ventana inferior del programa, y finalmente un mensaje diciendo el número de errores y avisos (warnings) resultantes.
- Una vez conseguido 0 errores, está listo para ejecutar el proyecto



# Pasos para la depuración de un proyecto



# Depuración de un Proyecto

- Una vez compilado correctamente, conecte la placa al USB y ejecute el depurador con el botón: 
  - Es posible que la primera vez le salgan unas ventanas que le pregunten por el modo de depuración. De ser así seleccione:
    - STM32 Cortex-M C/C++ Application
    - Acepte las opciones por defecto del depurador
  - También es posible que al intentar cargar el código en la placa, le diga que tiene que actualizar el firmware del ST-LINK
    - Acepte la actualización
    - Si le dice que no está disponible la placa, sin salirse del programa, desenchufe el USB, vuelva a enchufarlo y vuelva a intentar actualizar
- Una vez arrancado el modo depurador, la pantalla será como la de la siguiente transparencia, con el código y también una nueva pantalla a la derecha para ver valores de variables y registros
- Como puede ver, el programa se ha ejecutado hasta llegar a la función main(). La flecha azul indica el punto del programa donde se encuentra la ejecución

# Depuración de un Proyecto



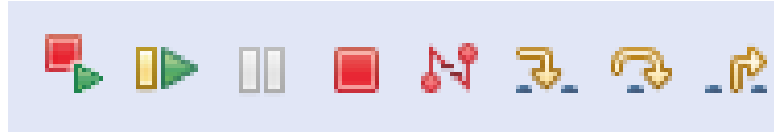
The screenshot shows an IDE window titled "CubelDE - SDBM\_T4\_Project1/Core/Src/main.c - STM32CubelDE". The main editor displays the source code for `main.c` with a breakpoint set at line 84, `HAL_Init();`. The code includes various initialization functions for peripherals like GPIO, ADC, LCD, and TS. The Project Explorer on the left shows the project structure, including the debug configuration "SDBM\_T4\_Project1 Debug [STM32 Cortex-M C/C++ Application]". The Console window at the bottom displays the message "Download verified successfully".

```
73  */
74  int main(void)
75  {
76  /* USER CODE BEGIN 1 */
77
78  /* USER CODE END 1 */
79
80
81  /* MCU Configuration-----*/
82
83  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
84  HAL_Init();
85
86  /* USER CODE BEGIN Init */
87
88  /* USER CODE END Init */
89
90  /* Configure the system clock */
91  SystemClock_Config();
92
93  /* USER CODE BEGIN SysInit */
94
95  /* USER CODE END SysInit */
96
97  /* Initialize all configured peripherals */
98  MX_GPIO_Init();
99  MX_ADC_Init();
100 MX_LCD_Init();
101 MX_TS_Init();
102 /* USER CODE BEGIN 2 */
103 BSP_LCD_GLASS_Init();
104 BSP_LCD_GLASS_BarLevelConfig(0);
105 BSP_LCD_GLASS_Clear();
106
107 /* USER CODE END 2 */
```

Console: SDBM\_T4\_Project1 Debug [STM32 Cortex-M C/C++ Application] ST-LINK (ST-LINK GDB server)  
Download verified successfully

# Depuración de un Proyecto

- Para depurar, se pueden utilizar las siguientes opciones, representadas por los siguientes botones:



- Parar la ejecución y volver a lanzar el programa
  - Continuar la ejecución del programa (también accesible por F8)
  - Pausar la ejecución del programa
  - Terminar la depuración (Ctrl + F2)
  - Desconectar la placa
  - Step Into (ejecutar el paso siguiente y si es una función, entrar a depurar la función) (F5)
  - Step Over (ejecutar el paso siguiente y si es una función, ejecutarla sin entrar a depurar) (F6)
  - Step Return (ejecutar el resto de la función y detenerse al salir de la función) (F7)
- Además, puede poner un punto de ruptura (breakpoint, es decir, pausa la ejecución cuando llega a ese punto) haciendo doble clic en cualquier punto a la izquierda del número de línea.
    - Aparecerá un círculo
    - Si se vuelve a hacer doble clic, se quita el breakpoint.

# Depuración de un Proyecto

- Cuando va ejecutando paso a paso, o ha llegado a un breakpoint, en la ventana de la derecha aparecen las variables locales
- También puede pasar el ratón por encima de una de las variables, para poder observar su valor y propiedades
- Si quiere, puede editar el valor, poniendo el nuevo valor en lugar del actual

The screenshot shows a debugger window with the following components:

- Code Editor:** Displays C code for a function `espera(int tiempo)`. The line `(i=0; i<tiempo; i++);` is highlighted in green, indicating the current execution point.
- Expression Window:** A table showing the current value of the variable `tiempo`.
 

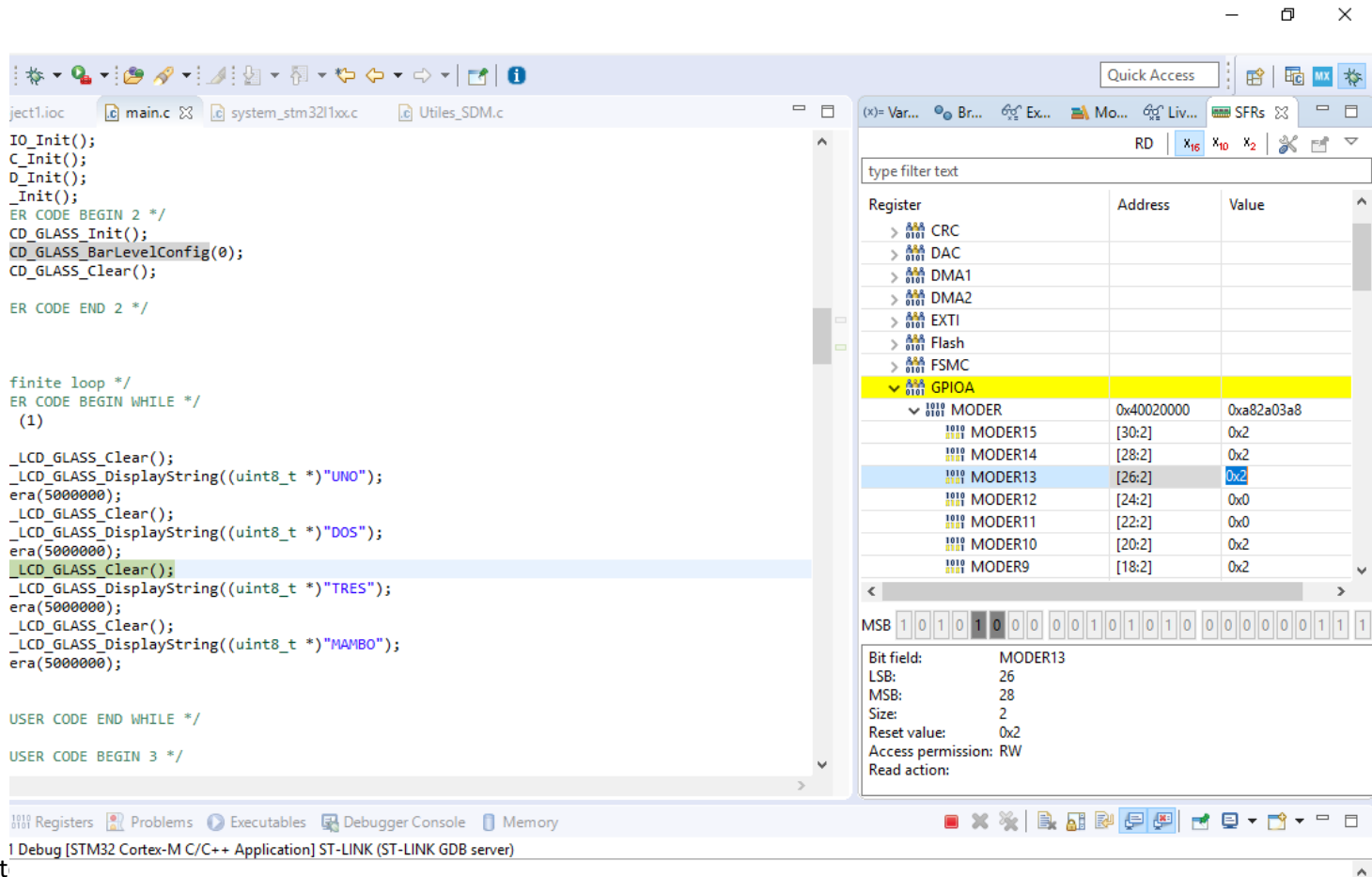
Expression	Type	Value
(*)= tiempo	int	5000000
- Variable Details:** A pop-up window for the variable `tiempo` showing its properties:
 

Name	tiempo
Details	5000000
Default	5000000
Decimal	5000000
Hex	0x4c4b40
Binary	1001100010010110100000
- Local Variables Window:** A table on the right side of the debugger showing local variables.
 

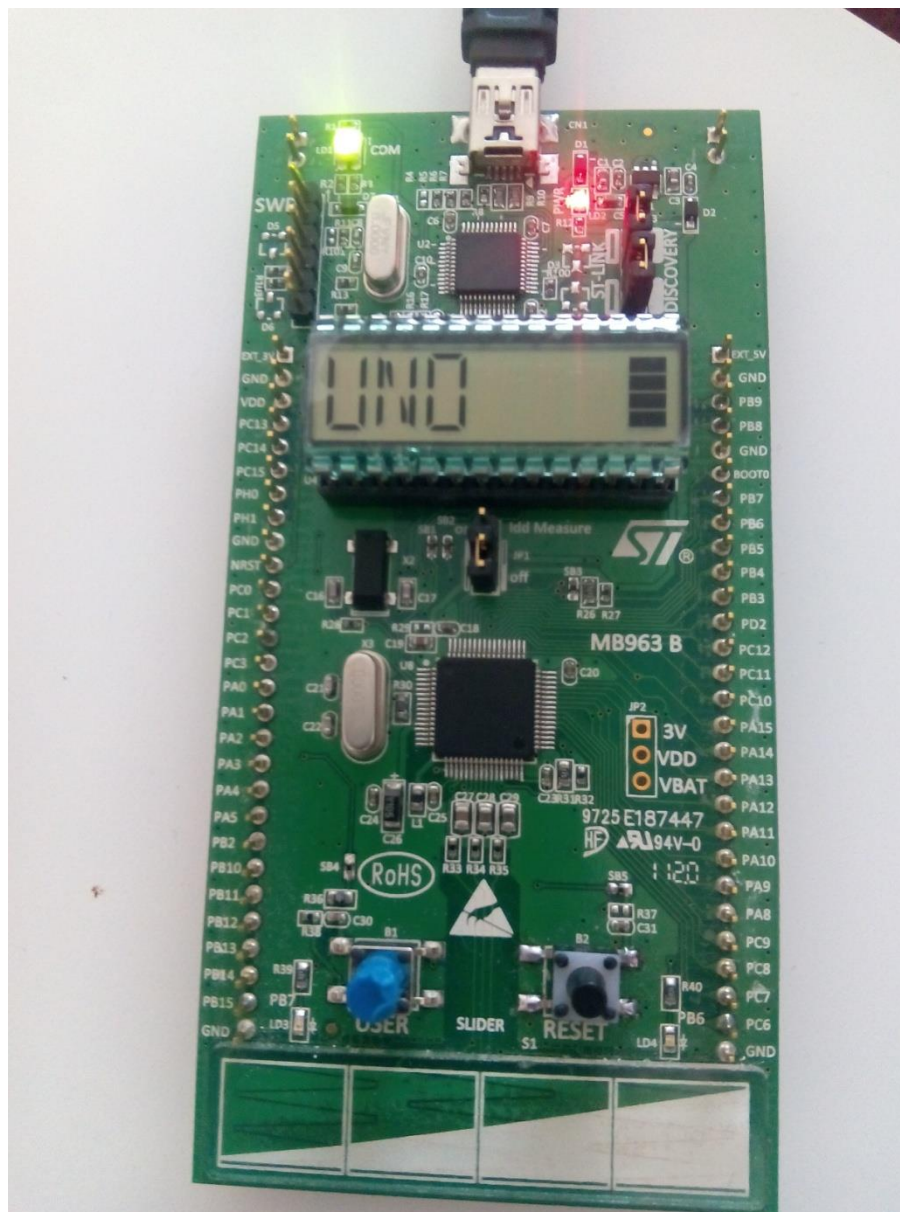
Name	Type	Value
(*)= tiempo	int	5000000
(*)= i	int	67112463

# Depuración de un Proyecto

- Más adelante observará que en la ventana de la derecha, hay pestañas con distintas funcionalidades, y que una de ellas permite acceder a los registros del microcontrolador (SFRs)



# Prueba del Proyecto Explicado



# Peculiaridades de la Programación en C en Microcontroladores

# Peculiaridades del C en Micros

- Introducción
- Ubicación de las definiciones de registros
- Definición de variables
  - Estáticas
  - Con el tamaño justo
- Paso de parámetros



- El Lenguaje C utilizado es ANSI-C, es decir, no tiene de por sí, ninguna variación a cualquier otra plataforma
- Sin embargo sí que es necesario mantener unas determinadas prácticas adicionales, para hacer una programación satisfactoria
- La razón es que NO estamos programando una aplicación en un PC, donde los recursos se podrían considerar limitados
  - Aquí se plantea una arquitectura con UNOS RECURSOS MUY LIMITADOS, tanto en cantidad de memoria, como en potencia de cálculo
  - Además se trata de una arquitectura que no tiene por qué tener un teclado, una pantalla o una conexión a internet, por lo que el desarrollador tiene que tener muy presente la arquitectura con la que trabaja
- En las siguientes transparencias se ilustran algunas recomendaciones

- Use el tamaño de variable que más se ajuste a sus necesidades
  - Si una variable va a tener solo valores entre el 0 y el 5, use un **unsigned char**, en lugar de un **int** (se pasa de usar 1 byte a 4 bytes)
  - Si la variable no va a tener valores negativos, use **unsigned**, para limitar los valores (y evitar problemas de cálculo posteriores)
- Salvo que sea estrictamente necesario, no utilice asignación dinámica de memoria, sino estática
  - La asignación dinámica de memoria implica:
    - Utilización de métodos adicionales como **malloc()** que consumen tiempo y recursos
    - Que cada trozo de memoria asignado, debe tener también espacio de reserva para los punteros de asignación (cada puntero son 4 bytes)
    - Que la programación tenga que ser más cuidadosa para no desbordar la memoria o acceder a partes reservadas de la memoria
      - En concreto, un control bastante elevado del uso de punteros
  - Por ejemplo, si va a usar una variable para el mensaje del LCD, y sabe que dicho mensaje es como mucho de 6 caracteres, use
    - **unsigned char message[6];**
    - Y acceda de forma estática al tercer carácter con **message[2];**

- No utilice funciones externas de las que desconozca su verdadero funcionamiento, o que estén sobredimensionadas a sus necesidades
  - Por ejemplo, se suele cometer el error de pensar que `printf (message)` va a mostrar el contenido de `message` por pantalla pero:
    - ¿Seguro que se va a mandar por pantalla?
      - No, se va a mandar por puerto serie
    - ¿Cuanta memoria ocupa utilizar `printf ()`?
      - Muchísima más que la necesaria, ya que no sólo es el envío de caracteres, sino que también contiene la forma de formatear la cadena `message` (por ejemplo los parámetros %)
- Pase los parámetros complejos por referencia, en lugar de por valor
  - Al pasar los parámetros por referencia, sólo se copia el puntero al dato, pero el dato no se duplica en la función.
  - Sin embargo, al pasarlo por valor, el contenido del parámetro se copia como una nueva variable en la ejecución de la función
  - Por defecto, el lenguaje C pasa todos los parámetros complejos por referencia, pero una mala programación puede forzar a pasarlos por valor

- Utilice siempre variables locales
  - De esta forma, cuando se salga de la función, se destruye la variable, liberando la memoria
  - Si crea una variable compleja dentro de una función, acuérdesese de eliminarla coherentemente antes de salir de la función
- No utilice cálculos de excesivo coste, cuando no es necesario. En concreto, intente evitar cálculos con decimales
  - La aritmética del microcontrolador es una aritmética entera, por lo que cualquier cálculo decimal supondría el tener que incluir en el código, las rutinas correspondientes para hacer el cálculo en punto fijo o en punto flotante a partir de la ALU entera
    - Esto conlleva un aumento del tamaño del programa, así como de su tiempo de computación, enorme
  - Por ejemplo, si en un problema va a trabajar con temperaturas, con precisión de un decimal, entre -20 y +50 grados, no trabaje con decimales, sino considere trabajar con enteros entre -200 y +500, que en este caso cabrían en una variable de tipo **short**
- También recuerde que si usa aritmética entera  $x/y = 0$  siempre que  $x$  sea menor que  $y$ , por lo que debería multiplicar  $x$  por algún valor antes de dividir

# Otras Recomendaciones

- Si hay parámetros que pueden llegar a cambiar de proyecto a proyecto, utilice #define
  - Los #define se ubican normalmente en la parte inicial de un fichero .c, o en los ficheros .h
  - Al cambiar el valor, automáticamente se cambia en todos los sitios donde se haya utilizado el identificador utilizado en el #define
  - Además el cambio se hace en momento de compilación, por lo que no implica un coste computacional
  - Por ejemplo:

```
#define MAXIMO 500;  
  
...  
If (valor > MAXIMO) Error();  
Else Procesa(valor);  
  
...  
Porcentaje = valor*100/MAXIMO;
```

# Particularidades del C para STM32L152



- Los registros ya se encuentran definidos en:
  - `stm32l152xb.h`
- Esta biblioteca ya se encuentra incluida en los proyectos, por lo que no hay que hacer nada adicional para usarla

# Recomendaciones de Uso de la Placa de Desarrollo

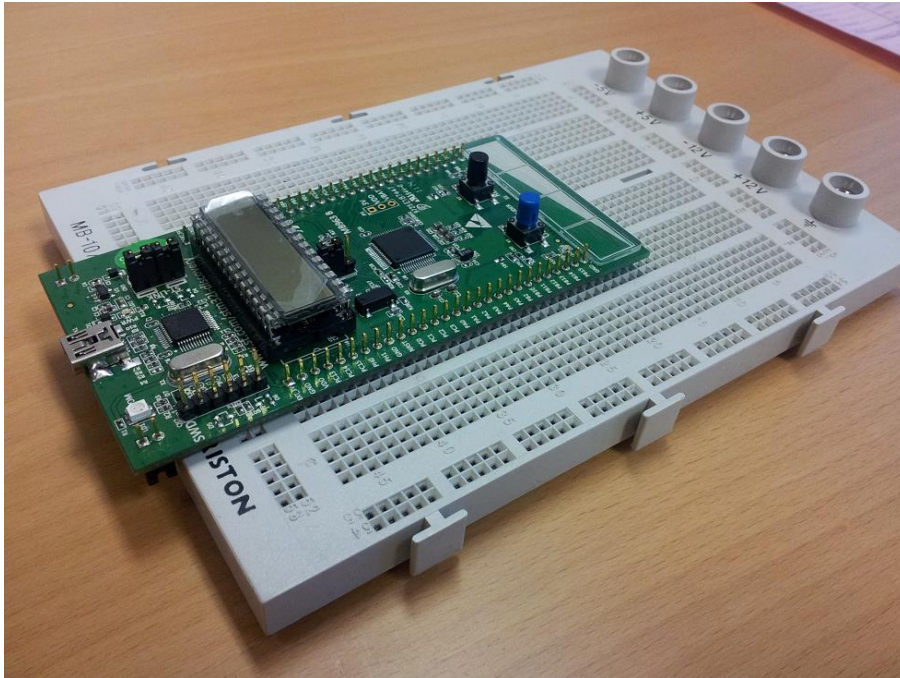
# Recomendaciones para el uso de la placa



- Para poder utilizar mucho mejor la placa STM32L1-Discovery, es aconsejable:
  - Pincharla en una protoboard (o en un conjunto de ellas), de forma que los pines no se cortocircuiten y además dejen huecos para conectar cables
  - Meter el conjunto de la protoboard, la placa, así como las conexiones realizadas, en una caja, para su transporte sin que se suelten las conexiones
- En la siguiente transparencia se puede ver el detalle de conexión, así como un ejemplo de uso



# Inserción de la placa en una protoboard



# Ejemplo de Uso

