



Blink Sketch using HAL libraries

rev1.0 24/03/2020

GOAL

Toggle a GPIO Output using HAL Libraries

PREREQUISITES

Software needed:

- STM32IDE

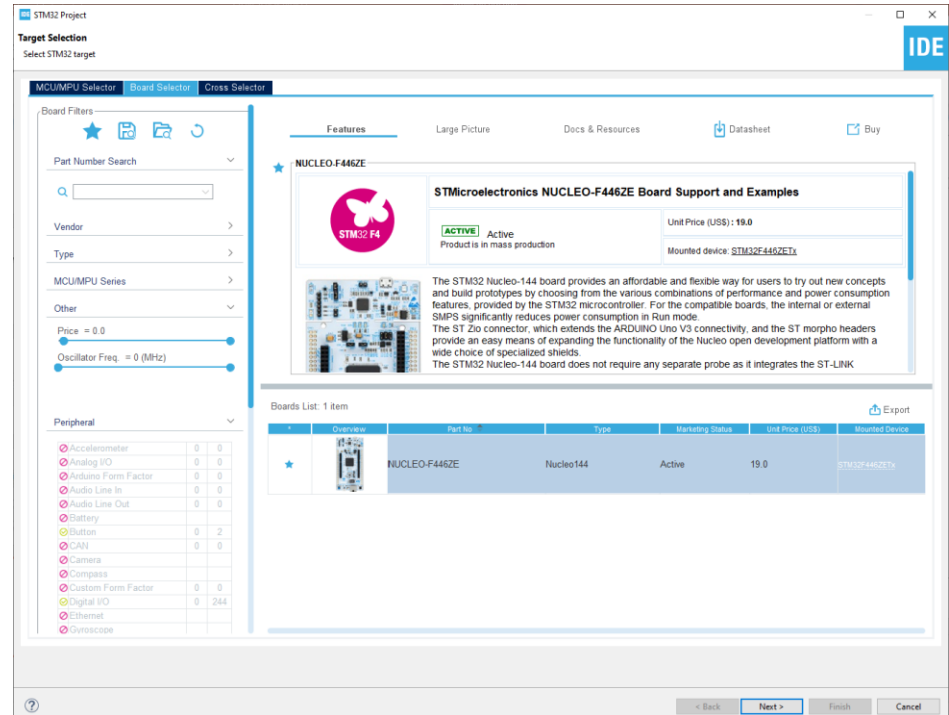
Hardware used in this example:

- **NUCLEO-F446ZE**

Start a new project

From the stm32IDE software click on
File -> New -> STM32 Project.

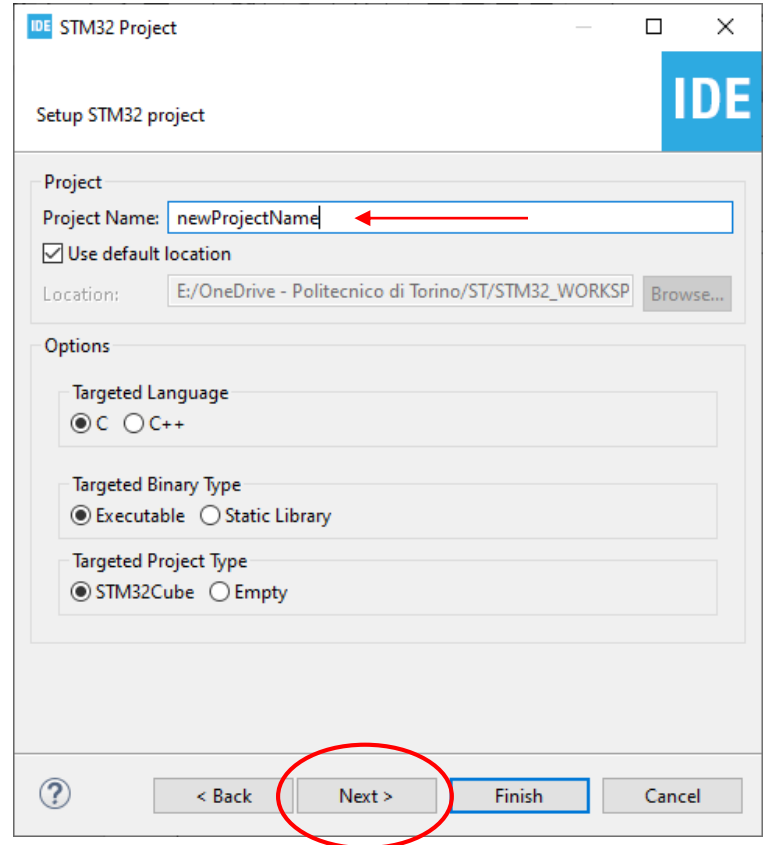
Select your board or your uC and click
next.



Start a new project

Type the name of your project and click next.

By default the project will be created in the workspace folder.



The screenshot shows the 'Setup STM32 project' dialog box in the IDE. The dialog has a title bar with 'IDE STM32 Project' and standard window controls. The main content is divided into sections: 'Project' and 'Options'. In the 'Project' section, the 'Project Name' field contains 'newProjectName' and is highlighted with a red arrow. Below it, the 'Use default location' checkbox is checked. The 'Location' field shows 'E:/OneDrive - Politecnico di Torino/ST/STM32_WORKSP' with a 'Browse...' button. The 'Options' section contains three groups of radio buttons: 'Targeted Language' with 'C' selected, 'Targeted Binary Type' with 'Executable' selected, and 'Targeted Project Type' with 'STM32Cube' selected. At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >' (circled in red), and 'Finish'. The 'Finish' button is also highlighted with a blue border.

Start a new project

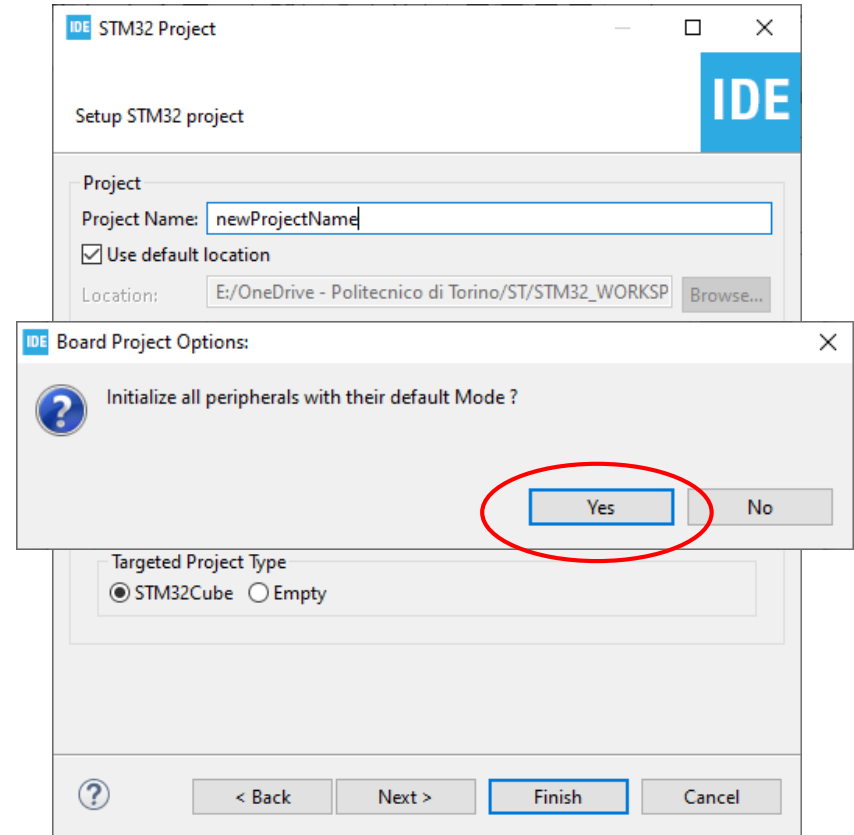
Type the name of your project and click next.

By default the project will be created in the *workspace* folder.

The *STM32IDE* has the option to initialize all the peripheral with their **default** mode:

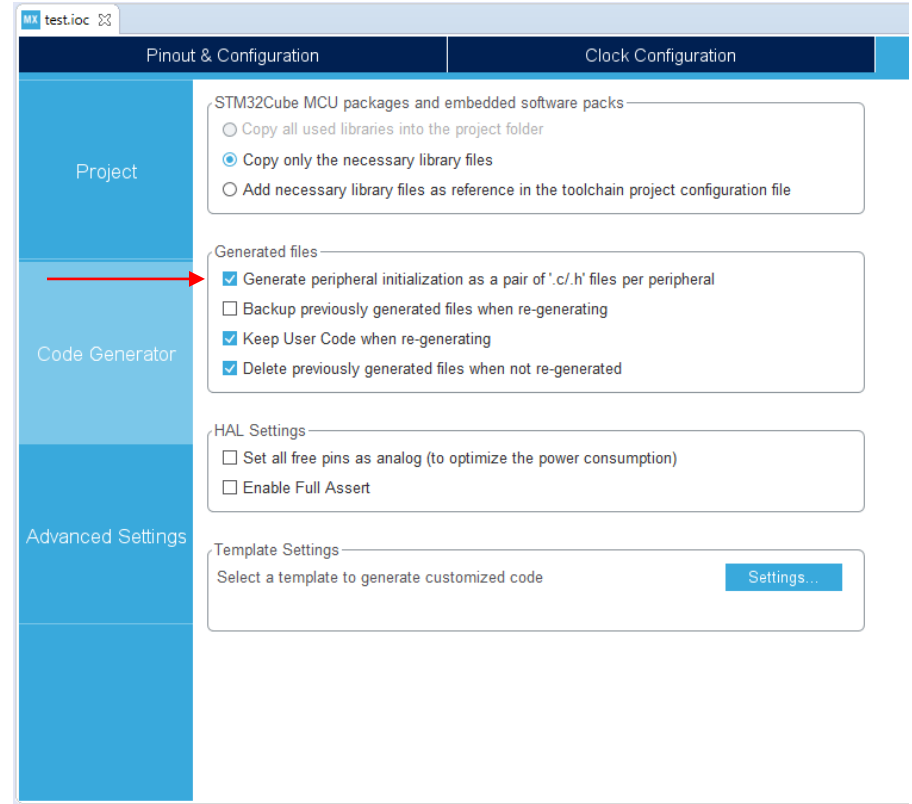
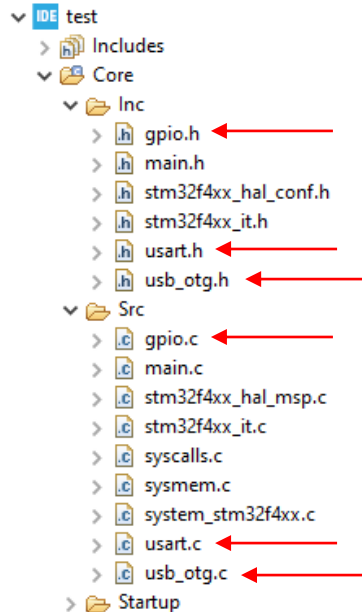
Clicking *Yes* the *USART3*, all the *LEDs* and the blue *UserButton* will be configured as default.

Click *Yes*.



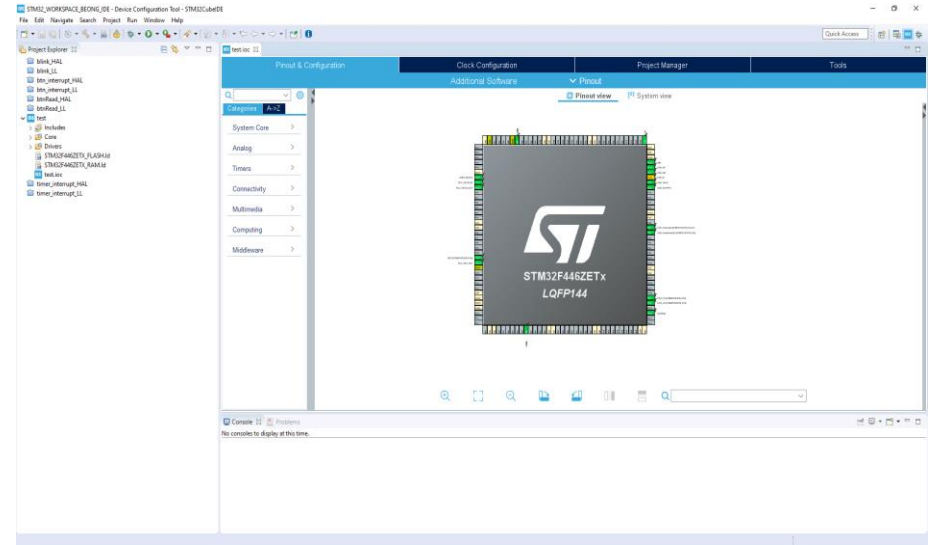
Project Manager

In the Code Generator Tab check the ***Generate peripheral initialization [...]*** box: each peripheral will have a disting *periph.c* and *periph.h* files.

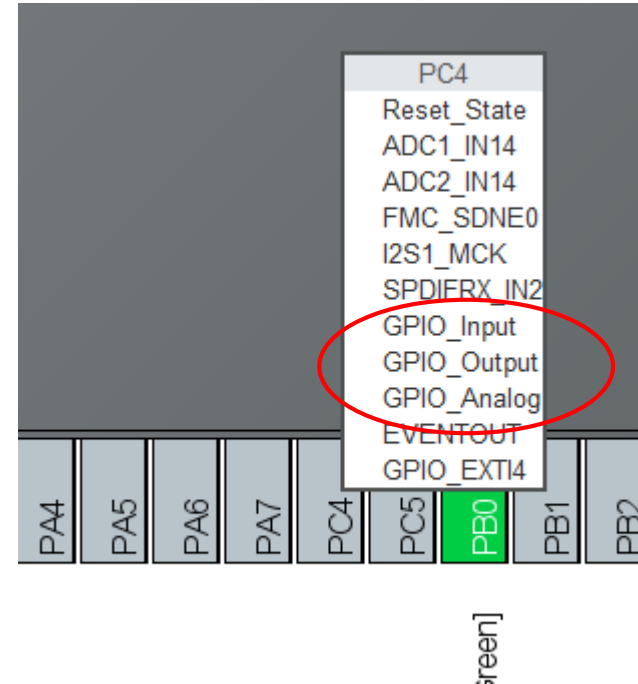


GPIO: General Purpose Input/Output

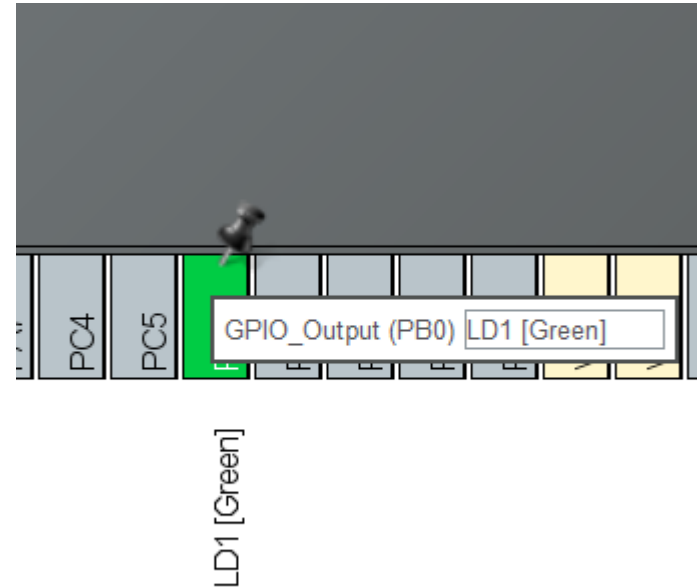
- After creating the project and initializing the functions to their default values, it is useful, before starting to scroll, to quickly scroll through the characteristics of the GPIO pins and how to be configured.
- By clicking on a generic pin with the left mouse button a list will appear with all the pin configuration options.



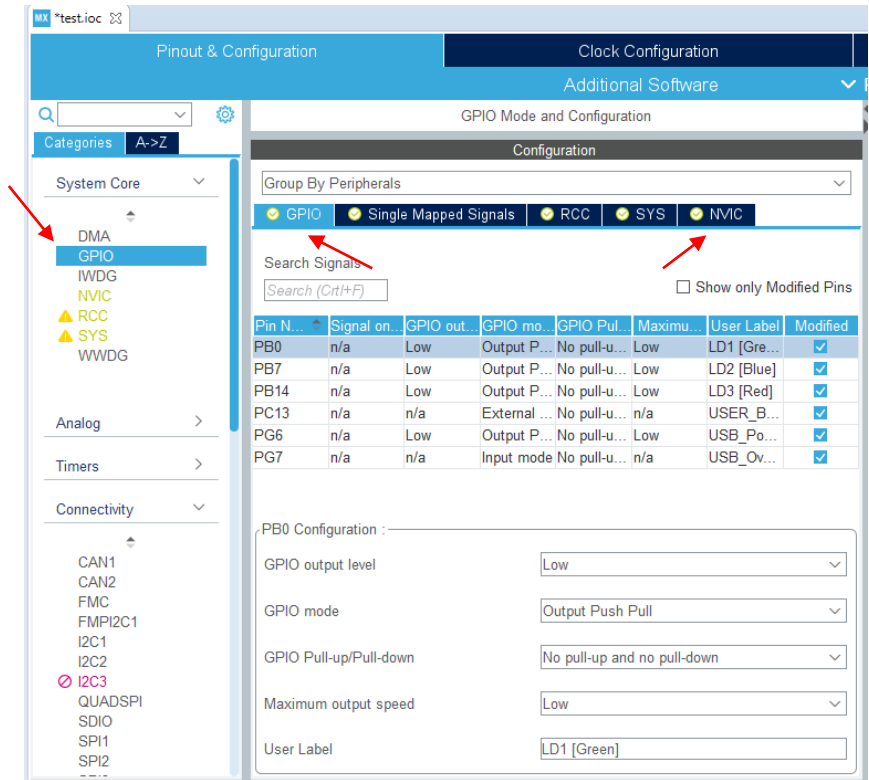
- In this example we will analyze in particular the ***GPIO_xx***:
- ***GPIO_OUTPUT***: Used to control an output device (e.g. LED, Motor, etc.)
- ***GPIO_INPUT***: It is used in case you want to read the state of an input device (eg Switch, Sensors, etc.)
- ***GPIO_ANALOG***: Similar to the input, it is used when the input device is analog (e.g. Photoresistor, Potentiometer, etc.)



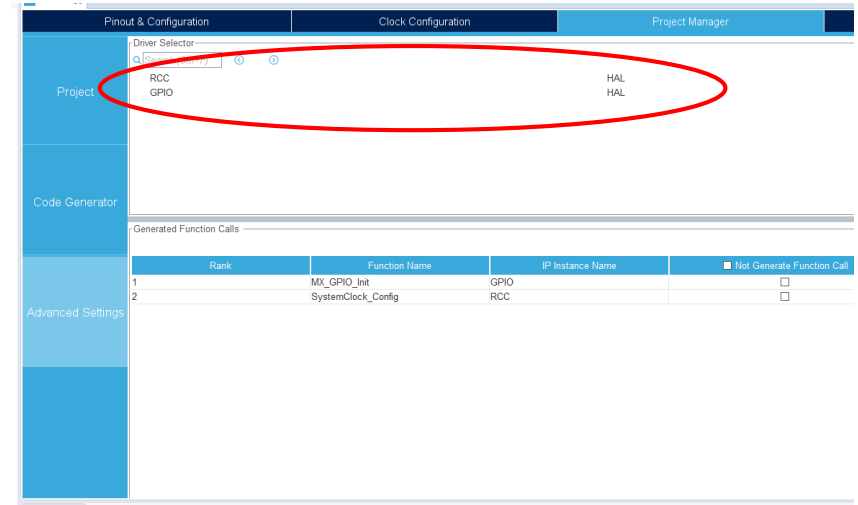
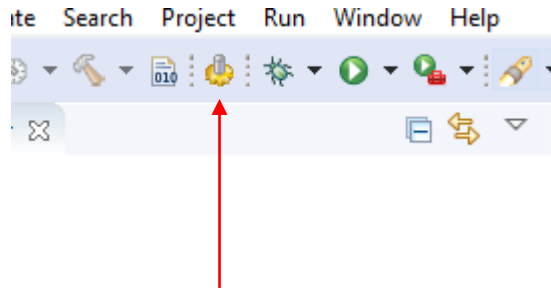
- By right-clicking on an already set pin it is instead possible to insert a **label**, very useful when writing the code, when you have numerous peripherals to manage.
- Under the heading **system core** → **GPIO** we find a more in-depth view of GPIOs



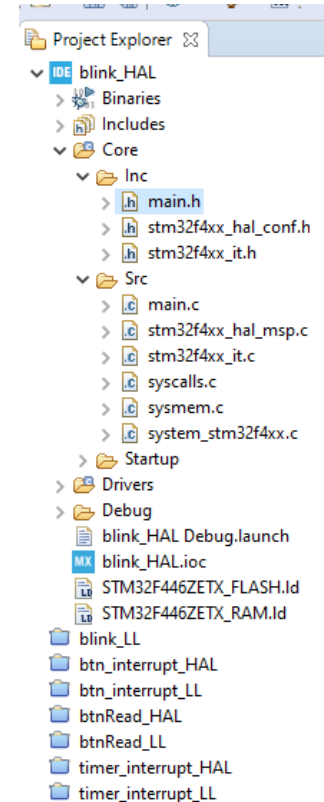
- Below the **GPIO** tab we find the list of the set pins and their characteristics specifically.
- Further down we find the *Configuration* section where you can set other parameters such as default output level, mode, etc.
- Important, among the upper Tabs, we find the **NVIC** tab, useful for setting the interrupts which we will discuss later.



- At this point, all that remains is to verify that you have chosen the **HAL** libraries for managing the GPIO and continue with the generation of the code using the appropriate icon.



- Once the code is generated, on the left we find the ***project explorer***. from here we have an overview of all the files that make up our project.
- Let's start by opening ***main.h*** and ***main.c***



User code BEGIN and END

- Inside files generated by **CubeMX** we will find spaces where the user can write some code: these spaces are marked with comments ***/* USER CODE BEGIN X */ and /* USER CODE END X */***
- These will be the only fields that will not be touched after a possible regeneration of the code by CubeMX, therefore it is **essential** not to write outside these spaces.
- In **main.c**, in section 4, you can add the functions written by the user, let's see them in more detail.

```
228
229 /* USER CODE BEGIN 4 */
230
231 void blink_toggle(int time, GPIO_TypeDef* GPIOx_PORT, uint16_t Ldx_Pin){
232
233     HAL_GPIO_TogglePin(GPIOx_PORT, Ldx_Pin); // toggle the pin each time you call the function
234
235     HAL_Delay(time); //ms delay
236 }
237
238 void blink_writePin(int time, GPIO_TypeDef* GPIOx_PORT, uint16_t Ldx_Pin){
239
240     /* Read the led Pin:
241      * led_state = SET    ( '1' )    if the led is on
242      * led_state = RESET  ( '0' )    if the led is off
243      */
244     int led_state = HAL_GPIO_ReadPin(GPIOx_PORT, Ldx_Pin);
245
246
247     if (led_state) {
248         HAL_GPIO_WritePin(GPIOx_PORT, Ldx_Pin, RESET); //turn off the led if it is on
249     }
250
251     else {
252         HAL_GPIO_WritePin(GPIOx_PORT, Ldx_Pin, SET); //turn on the led if it is off
253     }
254
255     HAL_Delay(time); //ms delay
256 }
257
258 /* USER CODE END 4 */
259
260
```

blink_writePin()

To make the led on the board flash, you can proceed in different ways:

- A first strategy is to read the current state of the pin using the ***HAL_GPIO_ReadPin()*** function and consequently use the ***HAL_GPIO_WritePin()*** function to set the output, in this case it is possible to pass the pin status through a parameter
 - ***SET*** → '1'
 - ***RESET*** → '0'

```
void blink_writePin(int time, GPIO_TypeDef* GPIOx_PORT, uint16_t LDx_Pin){  
  
    /* Read the led Pin:  
    * led_state = SET    ( '1' )    if the led is on  
    * led_state = RESET ( '0' )    if the led is off  
    */  
    int led_state = HAL_GPIO_ReadPin(GPIOx_PORT, LDx_Pin);  
  
    if (led_state) {  
        HAL_GPIO_WritePin(GPIOx_PORT, LDx_Pin, RESET); //turn off the led if it is on  
    }  
  
    else {  
        HAL_GPIO_WritePin(GPIOx_PORT, LDx_Pin, SET); //turn on the led if it is off  
    }  
  
    HAL_Delay(time); //ms delay  
}
```

blink_toggle()

- Another way to change the output of an output pin may be to use the ***HAL_GPIO_TogglePin()*** function;
- This does nothing but read the current state of the pin (LDx_pin) and invert its value:
 - If *SET* -> *RESET*
 - If *RESET* -> *SET*

```
228
229 /* USER CODE BEGIN 4 */
230
231 void blink_toggle(int time, GPIO_TypeDef* GPIOx_PORT, uint16_t LDx_Pin){
232
233     HAL_GPIO_TogglePin(GPIOx_PORT, LDx_Pin); // toggle the pin each time you call the function
234
235     HAL_Delay(time); //ms delay
236 }
237
```


Main loop!

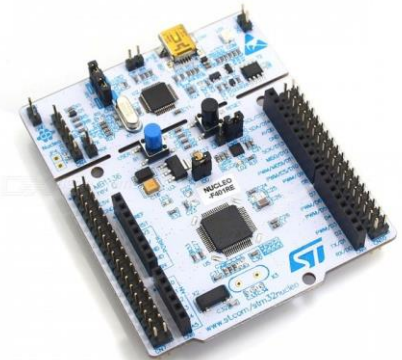
- Within the *main* function, after the various initializations that CubeMX has done for us, we find the *mainLoop*: the program will do nothing but repeat what we will write within it indefinitely.
- At this point, to make the LED flash, it is not enough to recall the functions we have just written!

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* Uncomment the function that you want to use */

    //blink_toggle(TIME_DELAY, LD2_GPIO_Port,LD2_Pin); //blink the led using the HAL_GPIO_TogglePin function
    blink_writePin(TIME_DELAY, LD3_GPIO_Port,LD3_Pin); //blink the led using the HAL_GPIO_WritePin function

    /* USER CODE END WHILE */

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



Tips & Tricks

- It is good practice to write functions that are as versatile as possible, for this we use **parameters**.
- It is also useful to use main.h to write the prototypes of the functions and also to define constants, such as the one we called **TIME_DELAY** so that we can use the same parameter several times in the code.
- It is possible to use the **CRTL + SPACE** shortcut for the autocompletion of the code, indispensable!

```
228
229 /* USER CODE BEGIN 4 */
230
231 void blink_toggle(int time, GPIO_TypeDef* GPIOx_PORT, uint16_t LDx_Pin){
232
233     HAL_GPIO_TogglePin(GPIOx_PORT, LDx_Pin); // toggle the pin each time you call the function
234
235     HAL_Delay(time); //ms delay
236 }
237
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* Uncomment the function that you want to use */

    //blink_toggle(TIME_DELAY, LD2_GPIO_Port,LD2_Pin); //blink the led using the HAL_GPIO_TogglePin function
    blink_writePin(TIME_DELAY, LD3_GPIO_Port,LD3_Pin); //blink the led using the HAL_GPIO_WritePin function

/* USER CODE END WHILE */

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

#define LD2_Pin GPIO_PIN_7
#define LD2_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
#define TIME_DELAY 1000
/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif
1
```