



Cours Microcontrôleurs Avancés

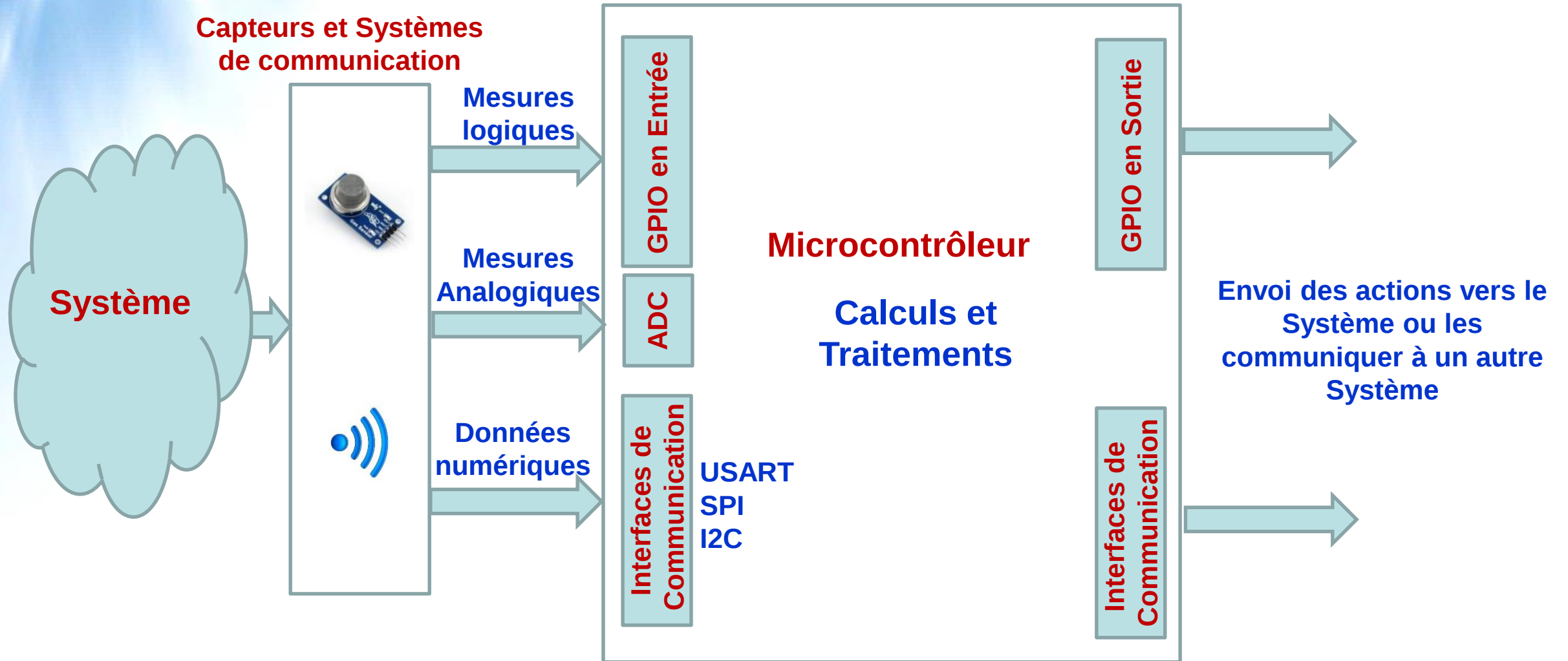
CH3: Projet STM32 CubeIDE: LE GPIO en Sortie

Par:

Hassan EL FADIL

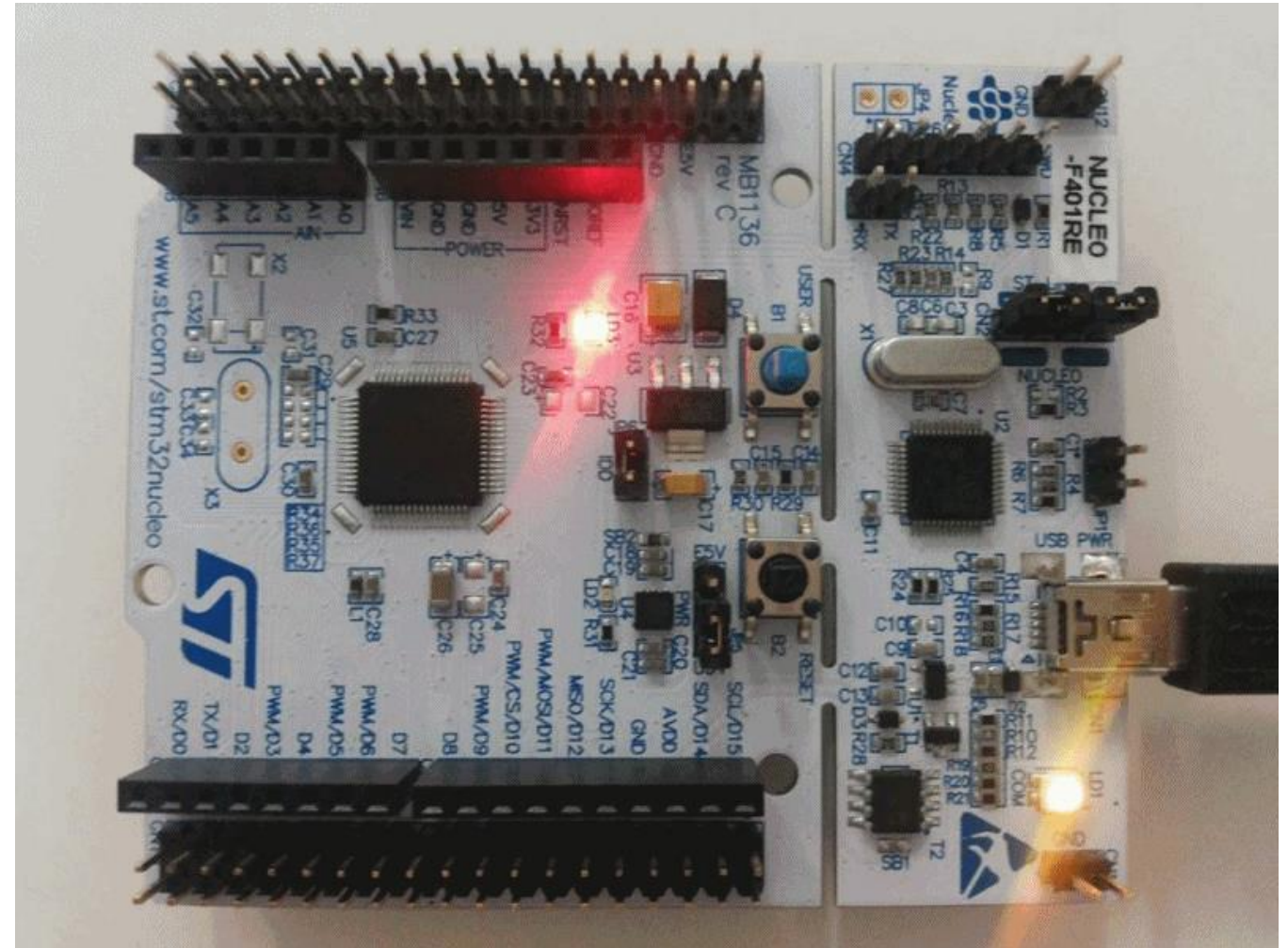
elfadil.h@gmail.com

Architecture d'une commande par microcontrôleur



Cahier des Charges:

- Programmer le clignotement de la LED utilisateur LED Verte de la carte
- La fréquence de clignotement est réglable par l'utilisateur



Chacun des GPIO dispose de **quatre** registres **32 bits** de contrôle (**GPIOx_MODER**, **GPIOx_OTYPER**, **GPIOx_OSPEEDR**, **GPIOx_PUPDR**) pour configurer jusqu'à 16 E/S.

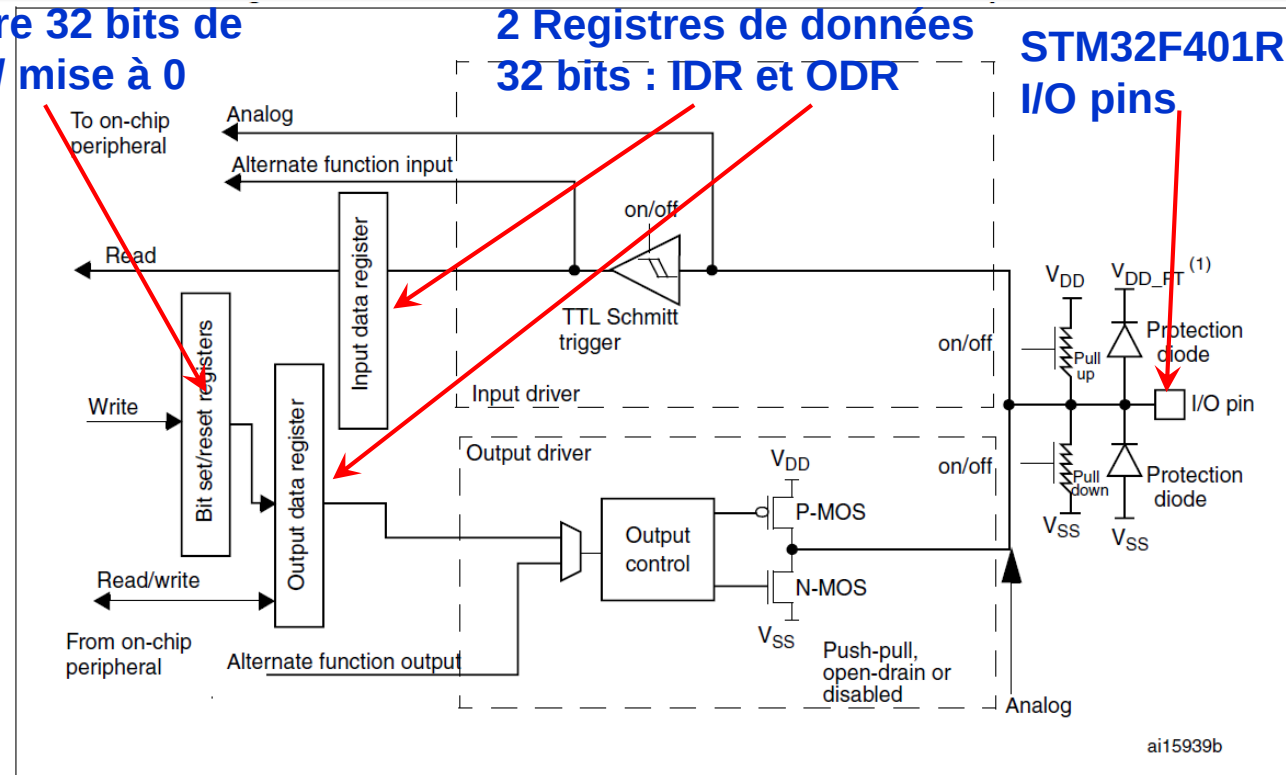
- Le registre **GPIOx_MODER** permet de sélectionner le sens des E/S (entrée, sortie, AF: alternate func, analogue).
- Les registres **GPIOx_OTYPER** et **GPIOx_OSPEEDR** permettent de sélectionner le type de sortie (push-pull ou open-drain) et la vitesse (les broches de vitesse d'E/S sont directement connectées aux bits de registre GPIOx_OSPEEDR correspondants quel que soit le sens des E/S).
- Le registre **GPIOx_PUPDR** permet de sélectionner le pull-up/pull-down quel que soit le sens des E/S.

(x = A, B, C, D and H)

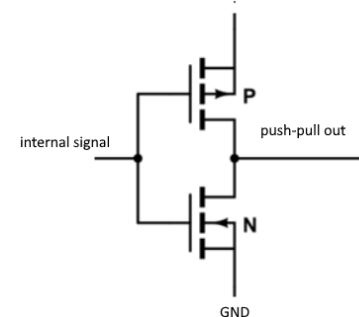
un registre 32 bits de mise à 1 / mise à 0

2 Registres de données 32 bits : IDR et ODR

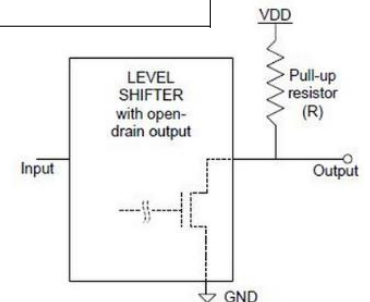
STM32F401RE: 51 GPIO I/O pins



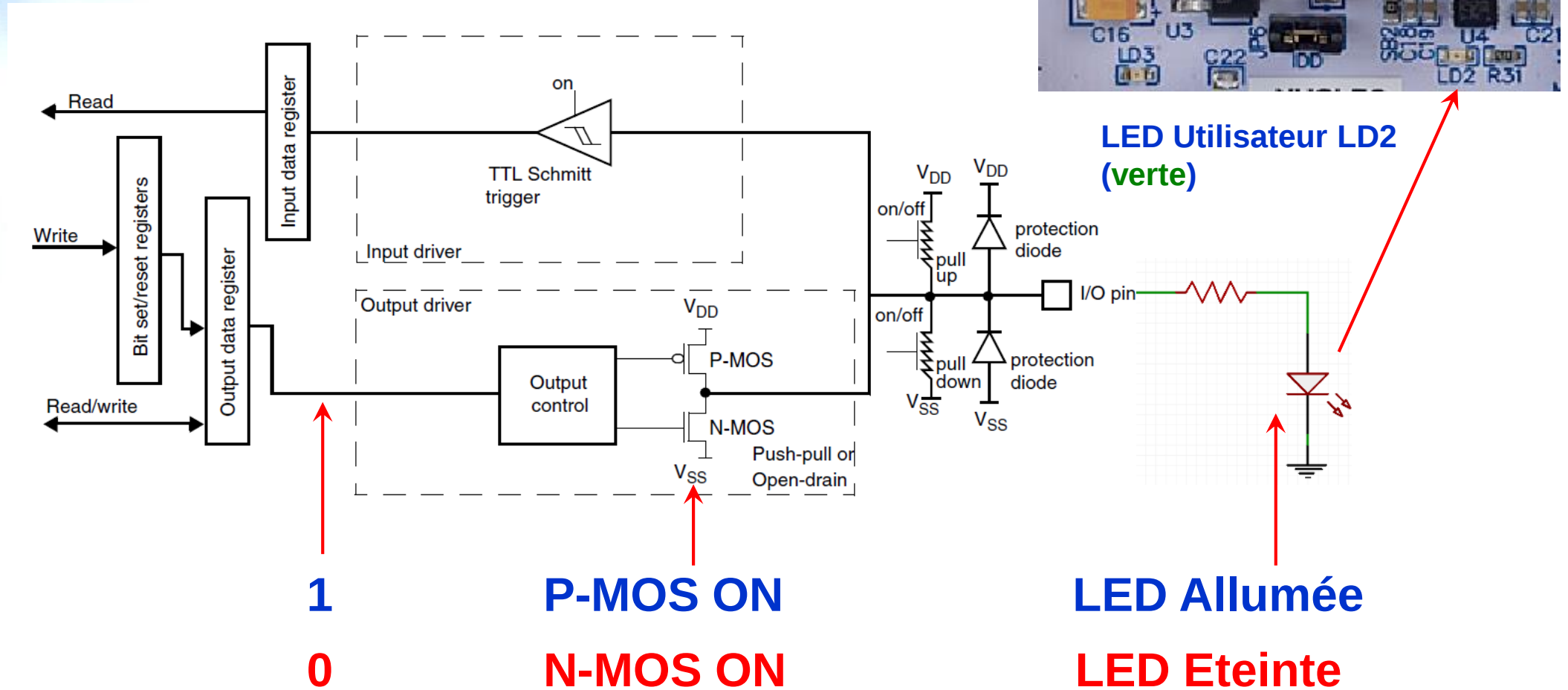
Push-Pull



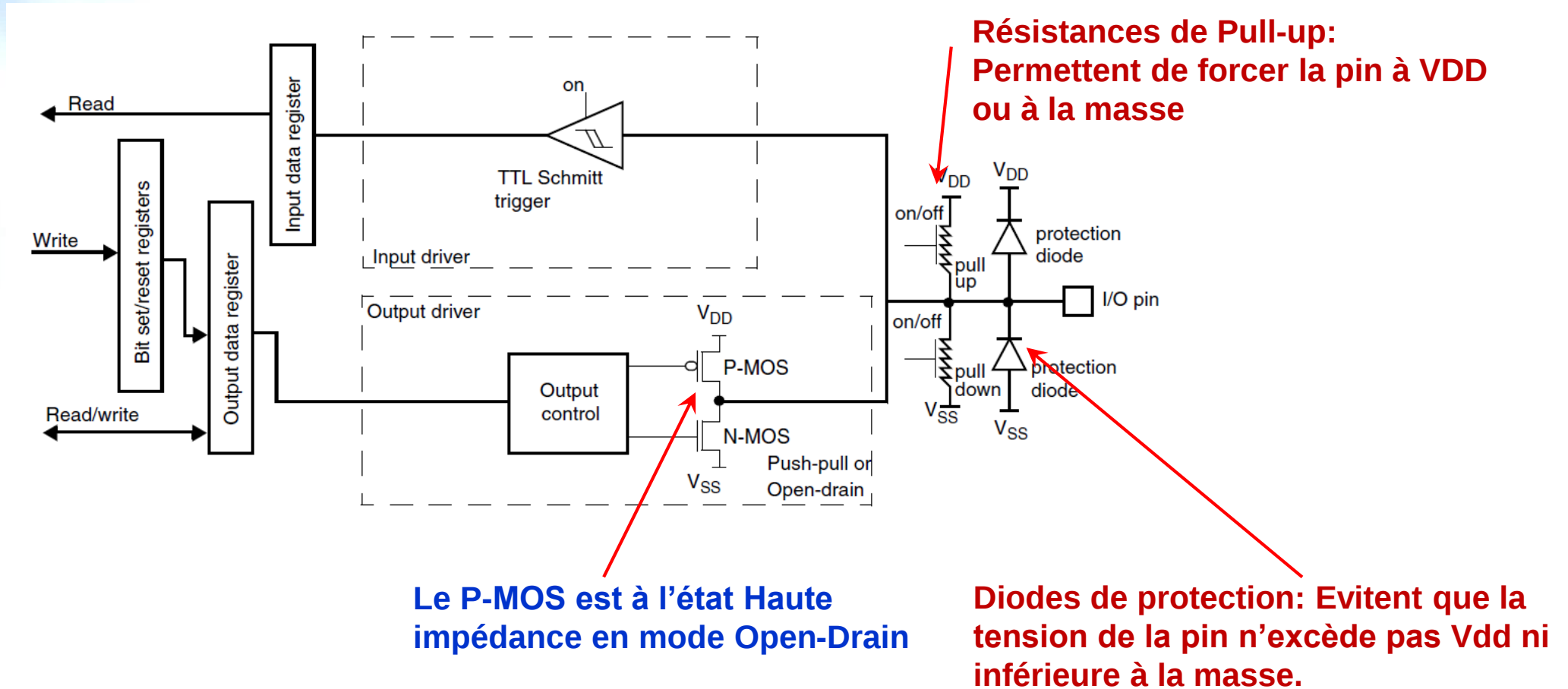
Open Drain (Drain Ouvert)



Configuration Push-Pull par exemple:



En Configuration Open-Drain le P-MOS est dans l'état Haute impédance:



- **STM32CubeIDE**, le IDE libre pour developper les programmes STM32
- Lien pour télécharger: <https://www.st.com/en/development-tools/stm32cubeide.html>

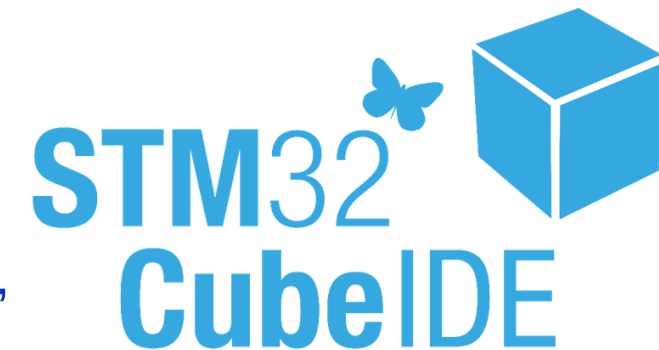
+	STM32CubeIDE-Win	STM32CubeIDE Windows Installer	1.13.2	Get latest	Select version ▼
---	------------------	--------------------------------	--------	------------	------------------

IDE (Integrated Development Environment) signifie Environnement de Développement Intégré.

Voici quelques fonctionnalités de l'IDE:

- Éditeur de code
- Compilateur/Interprète : traduit le code source écrit par le programmeur en code machine ou bytecode
- Débogueur : permet de rechercher et de corriger les erreurs dans le code
- Gestion de projet : l'IDE aide à gérer les projets en organisant les fichiers, les ressources et les dépendances
- Mise en évidence des erreurs
- ...

Vous devez vous inscrire à STM pour télécharger le logiciel, les mises à jour et l'installation de certains freeware.



File Edit Source Refactor Navigate Search Project Run Window Help Hello Hassan



“Device Configuration Tool Code Generation” : Pour générer le code

“Run”

Lance l'exécution de votre application sur le microcontrôleur STM32 connecté. Lorsque vous sélectionnez cette option, STM32CubeIDE compile votre code (si nécessaire), charge le binaire sur le microcontrôleur et commence à exécuter votre application. Cette option est utilisée lorsque vous souhaitez démarrer l'application et la laisser s'exécuter sans pause pour le débogage.

“Debug”

Ces options permettent de contrôler le processus de débogage, permettant aux développeurs d'analyser de manière interactive le comportement de leur code, d'identifier les problèmes et de vérifier l'exactitude de leurs applications sur les microcontrôleurs STM32. (START, STOP, RESUME, PAUSE, RESTART, BREAKPOINT, ...)

“Build Debug”

Compilation les fichiers de code source sont compilés en fichiers objets, Linking : les fichiers objets sont liés entre eux pour créer un fichier exécutable.

“Manage Configurations”

Cette fonctionnalité permet de créer, modifier et supprimer différentes configurations de build.

workspace_1.13.2 - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help Hello Hassan

New Alt+Shift+N >
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P

Makefile Project with Existing Code
C/C++ Project
STM32 Project
STM32 Project from an Existing STM32CubeMX Configuration File (.ioc)
STM32 CMake Project
Project...
Source Folder
Folder
Source File
Header File
File from Template
Class
Other...

Nouveau Projet STM32

Choisir « Board Selector »

MCU/MPU Selector
Board Selector
Example Selector
Cross Selector

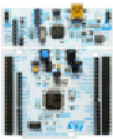
Board Filters
Commercial Part Number
NUCLEO-F401RE
NUCLEO-F401RE
PRODUCT INFO
Type
Supplier
MCU / MPU Series
Marketing Status
Price
MEMORY

Fe... Large Pi... Docs & Reso... Data...




New STM32H5 MCU series: more performance & scalable security

Boards List: 1 item
Export

	Overview	Comm...	Type	Marketi...	Unit Pri...	Mounte...
☆		NUCLE...	Nucleo-64	Active	13.0	STM32F4...

Chercher NUCLEO-F401RE

The screenshot shows the 'New Project' wizard in STM32CubeIDE. On the left, there's a sidebar with 'Commercial Part Number' set to 'NUCLEO-F401RE'. The main area displays the 'STM32F4 Series' with a detailed view of the 'NUCLEO-F401RE' board, including its description: 'STM32 Nucleo-64 development board with STM32F401RE MCU, supports Arduino and ST morpho connectivity'. Below this, a 'Boards List' table shows one item: 'NUCLEO-F401RE' (Nucleo-64 Active) with a unit price of 13.0. At the bottom, navigation buttons are visible: '< Back', 'Next >', 'Finish', and 'Cancel'. Red arrows point from French annotations to the board details, the boards list, and the 'Next >' button.

Commercial Part Number: NUCLEO-F401RE

PRODUCT INFO

- Type
- Supplier
- MCU / MPU Series
- Marketing Status
- Price

MEMORY

STM32F4 Series

NUCLEO-F401RE

STM32 Nucleo-64 development board with STM32F401RE MCU, supports Arduino and ST morpho connectivity

Boards List: 1 item

	Overview	Comm...	Type	Marketi...	Unit Pri...	Mounte...
☆		NUCLEO-F401RE	Nucleo-64 Active		13.0	STM32F4...

< Back **Next >** Finish Cancel

Vous pouvez visualiser ici les informations sur la carte

Vous pouvez allonger cette partie avec la souris pour voir plus de détails

Sélectionner la carte et cliquer sur « Next »

IDE STM32 Project

Setup STM32 project

Project

Project Name: GPIO_LED

☒ Use default location

Location: C:/Users/Dell/STM32CubeIDE/workspace_1.13.2

Options

Targeted Language

☒ C ☐ C++

Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

Buttons: ? < Back Next > Finish Cancel

Donner un nom au projet
(Exemple GPIO_LED)

Répertoire de travail où sera
enregistré votre projet (Gardez
le par défaut)

Langage de programmation
(Gardez le par défaut: C)

Garder par défaut
les autres parties

Cliquez ensuite sur « Next »

IDE STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference: NUCLEO-F401RE

Firmware Package Name and Version: STM32Cube FW_F4 V1.27.1

Firmware and Software Package Repository

Location: C:\Users\...

See 'Firm...

Code Generator Options

☐ Add necessary library files as reference in the toolchain project configuration file

☐ Copy all used libraries into the project folder

☒ Copy only the necessary library files

Buttons: ? < Back Next > Finish Cancel

Gardez cette option: « Copy
only the necessary files »

Cliquez ensuite sur « Finish »

The image shows the STM32CubeIDE interface during the initial project setup. The main window displays the 'GPIO_LED.ioc - Pinout & Configuration' tab, which includes a 'Pinout view' showing the STM32F401RE pinout. The pinout is color-coded: green for pins used (e.g., PA0, PA1, PA2, PA3, PA4, PA5, PA6, PA7, PA8, PA9, PA10, PA11, PA12, PA13, PA14, PA15, PB0, PB1, PB2, PB3, PB4, PB5, PB6, PB7, PB8, PB9, PB10, PB11, PB12, PB13, PB14, PB15, PC0, PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, PC9, PC10, PC11, PC12, PC13, PC14, PC15, PD0, PD1, PD2, PD3, PD4, PD5, PD6, PD7, PD8, PD9, PD10, PD11, PD12, PD13, PD14, PD15, PE0, PE1, PE2, PE3, PE4, PE5, PE6, PE7, PE8, PE9, PE10, PE11, PE12, PE13, PE14, PE15, PF0, PF1, PF2, PF3, PF4, PF5, PF6, PF7, PF8, PF9, PF10, PF11, PF12, PF13, PF14, PF15, PG0, PG1, PG2, PG3, PG4, PG5, PG6, PG7, PG8, PG9, PG10, PG11, PG12, PG13, PG14, PG15, PH0, PH1, PH2, PH3, PH4, PH5, PH6, PH7, PH8, PH9, PH10, PH11, PH12, PH13, PH14, PH15, PI0, PI1, PI2, PI3, PI4, PI5, PI6, PI7, PI8, PI9, PI10, PI11, PI12, PI13, PI14, PI15, PJ0, PJ1, PJ2, PJ3, PJ4, PJ5, PJ6, PJ7, PJ8, PJ9, PJ10, PJ11, PJ12, PJ13, PJ14, PJ15, PK0, PK1, PK2, PK3, PK4, PK5, PK6, PK7, PK8, PK9, PK10, PK11, PK12, PK13, PK14, PK15, PL0, PL1, PL2, PL3, PL4, PL5, PL6, PL7, PL8, PL9, PL10, PL11, PL12, PL13, PL14, PL15, PM0, PM1, PM2, PM3, PM4, PM5, PM6, PM7, PM8, PM9, PM10, PM11, PM12, PM13, PM14, PM15, PN0, PN1, PN2, PN3, PN4, PN5, PN6, PN7, PN8, PN9, PN10, PN11, PN12, PN13, PN14, PN15, PO0, PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO9, PO10, PO11, PO12, PO13, PO14, PO15). Yellow pins (e.g., VDD, VSS, VBAT, NRST, VREF, VCA, VDDA, VSSA, VDDIO, VSSIO, VDDIO1, VSSIO1, VDDIO2, VSSIO2, VDDIO3, VSSIO3, VDDIO4, VSSIO4, VDDIO5, VSSIO5, VDDIO6, VSSIO6, VDDIO7, VSSIO7, VDDIO8, VSSIO8, VDDIO9, VSSIO9, VDDIO10, VSSIO10, VDDIO11, VSSIO11, VDDIO12, VSSIO12, VDDIO13, VSSIO13, VDDIO14, VSSIO14, VDDIO15, VSSIO15) are used for power supply. Grey pins (e.g., PC13, PC14, PC15, PH0, PH1, PH2, PH3, PH4, PH5, PH6, PH7, PH8, PH9, PH10, PH11, PH12, PH13, PH14, PH15, PI0, PI1, PI2, PI3, PI4, PI5, PI6, PI7, PI8, PI9, PI10, PI11, PI12, PI13, PI14, PI15, PJ0, PJ1, PJ2, PJ3, PJ4, PJ5, PJ6, PJ7, PJ8, PJ9, PJ10, PJ11, PJ12, PJ13, PJ14, PJ15, PK0, PK1, PK2, PK3, PK4, PK5, PK6, PK7, PK8, PK9, PK10, PK11, PK12, PK13, PK14, PK15, PL0, PL1, PL2, PL3, PL4, PL5, PL6, PL7, PL8, PL9, PL10, PL11, PL12, PL13, PL14, PL15, PM0, PM1, PM2, PM3, PM4, PM5, PM6, PM6, PM7, PM8, PM9, PM10, PM11, PM12, PM13, PM14, PM15, PN0, PN1, PN2, PN3, PN4, PN5, PN6, PN7, PN8, PN9, PN10, PN11, PN12, PN13, PN14, PN15, PO0, PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO9, PO10, PO11, PO12, PO13, PO14, PO15) are not used.

Annotations in the image include:

- “Gris” Pins non utilisées**: Points to the grey pins in the pinout view.
- “Vert” Pins utilisées**: Points to the green pins in the pinout view.
- “Jaune” Pins alimentation**: Points to the yellow pins in the pinout view.
- Cliquez sur « Yes »**: Points to the 'Yes' button in the 'Board Project Options' dialog.
- Elargir à volonté avec la souris pour faire apparaître les pins du microcontrôleur**: Points to the 'Pinout view' tab.

The 'Board Project Options' dialog is open, showing the 'Initialize all peripherals with their default Mode?' checkbox. The 'Code Generator Options' section is also visible, with the 'Copy only the necessary library files' option selected. The 'Perform Project Creation. Please Wait For Completion ...' progress bar is shown at the bottom.

Sélectionner « System Core », GPIO.

Pinout & Configuration

Clock Configuration

Project Manager

Tools

Software Packs

Pinout

Search

Categories

A->Z

System Core

DMA

GPIO

IWDG

NVIC

✓ RCC

✓ SYS

WWDG

Analog

Timers

Connectivity

GPIO Mode and Configuration

Configuration

Group By Peripherals

✓ SYS

✓ USART

✓ NVIC

✓ GPIO

✓ Single Mapped Signals

✓ RCC

Search Signals

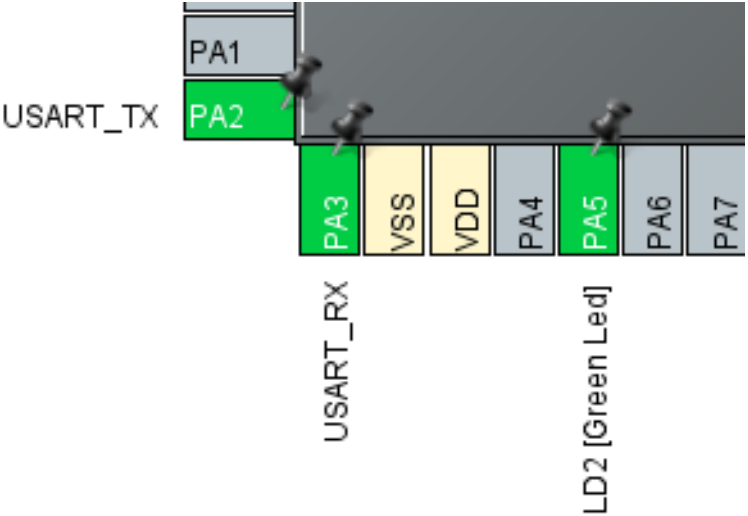
Search (Ctrl+F)

☐ Show only Modified Pins

P...	Sign...	GPI...	GPI...	GPI...	Maxi...	User...	Modi...
PA5	n/a	Low	Outp...	No p...	Low	LD2 ...	☒
PC1...	n/a	n/a	Exte...	No p...	n/a	B1 [...]	☒

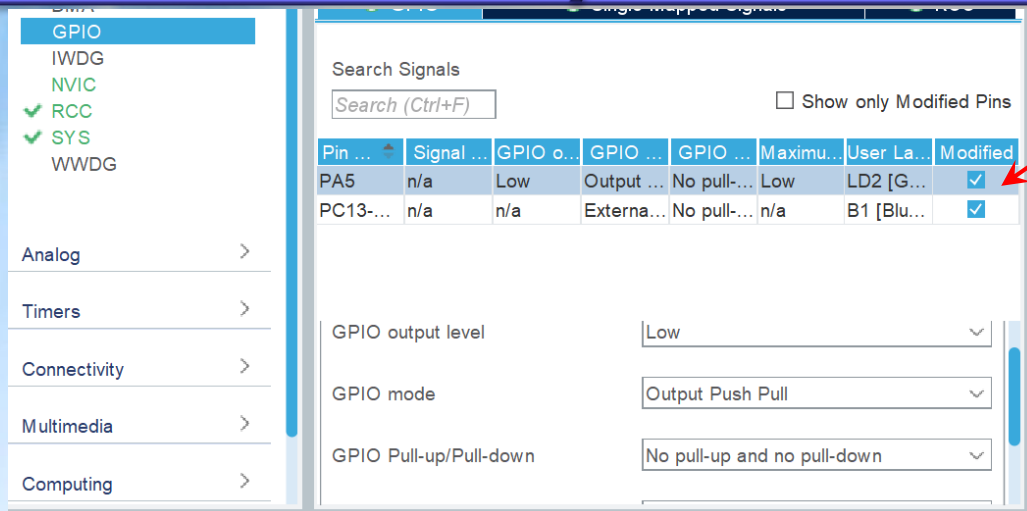
?

Select Pins from table to configure them. Multiple select

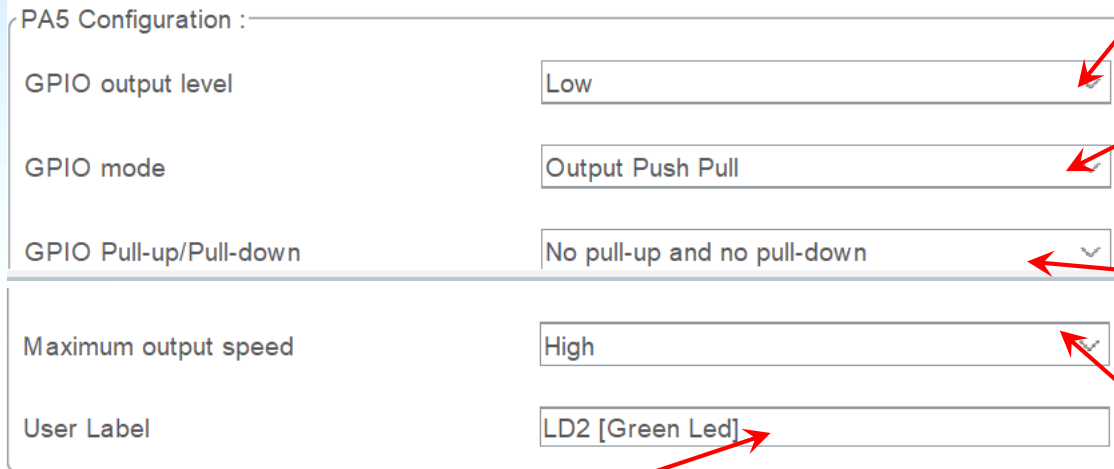
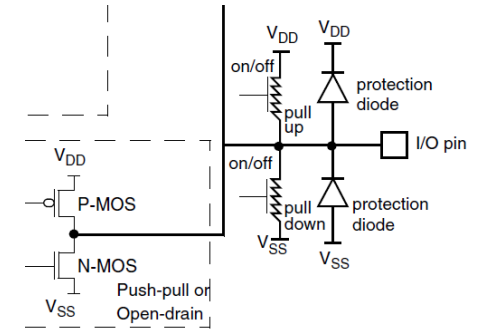


Pin PA5 déjà configurée pour la LED2.

Pin PC1 déjà configurée pour le Bouton Poussoir utilisateur B1 (Bleu).



Double-Cliquer sur la ligne PA5



Valeur initiale (Garder Low): LED éteinte au départ

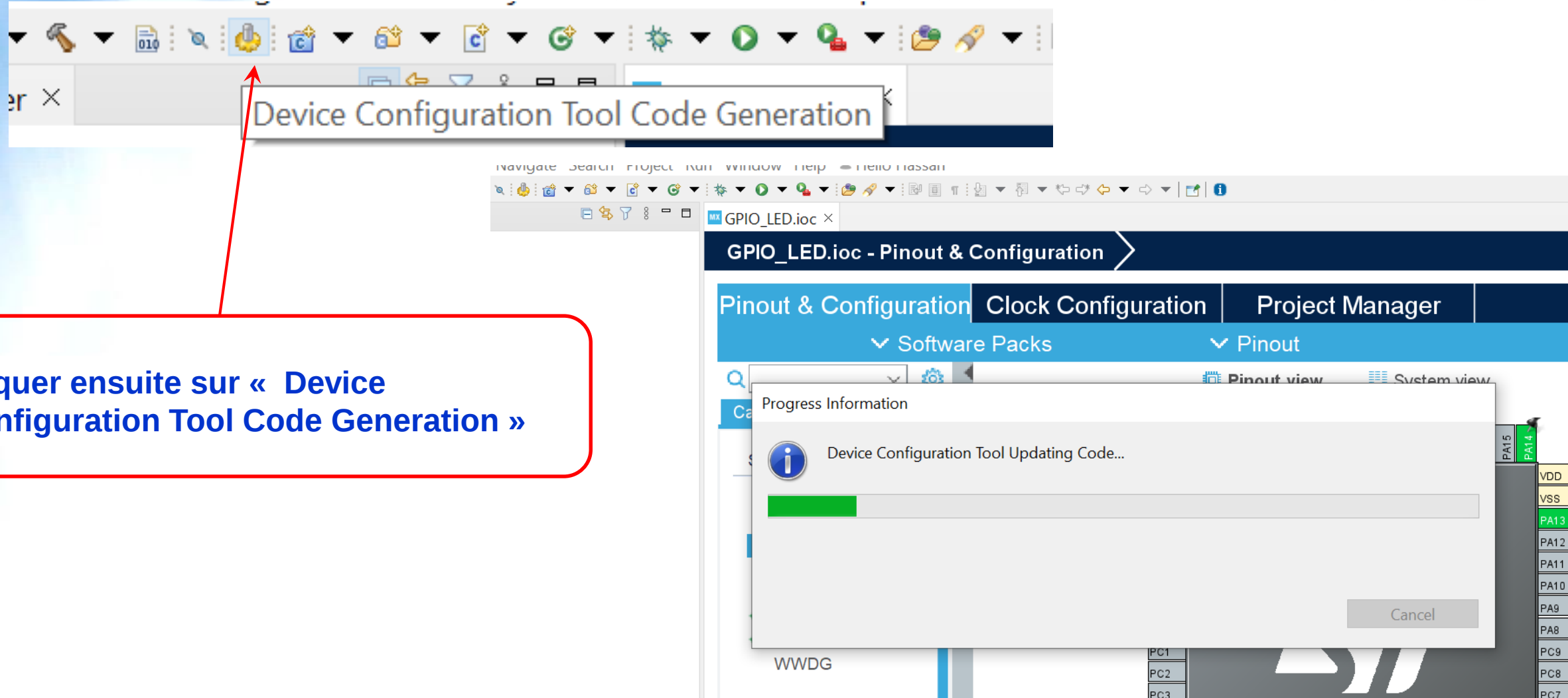
Mode: Push-Pull, Open Drain (Garder Push-Pull)

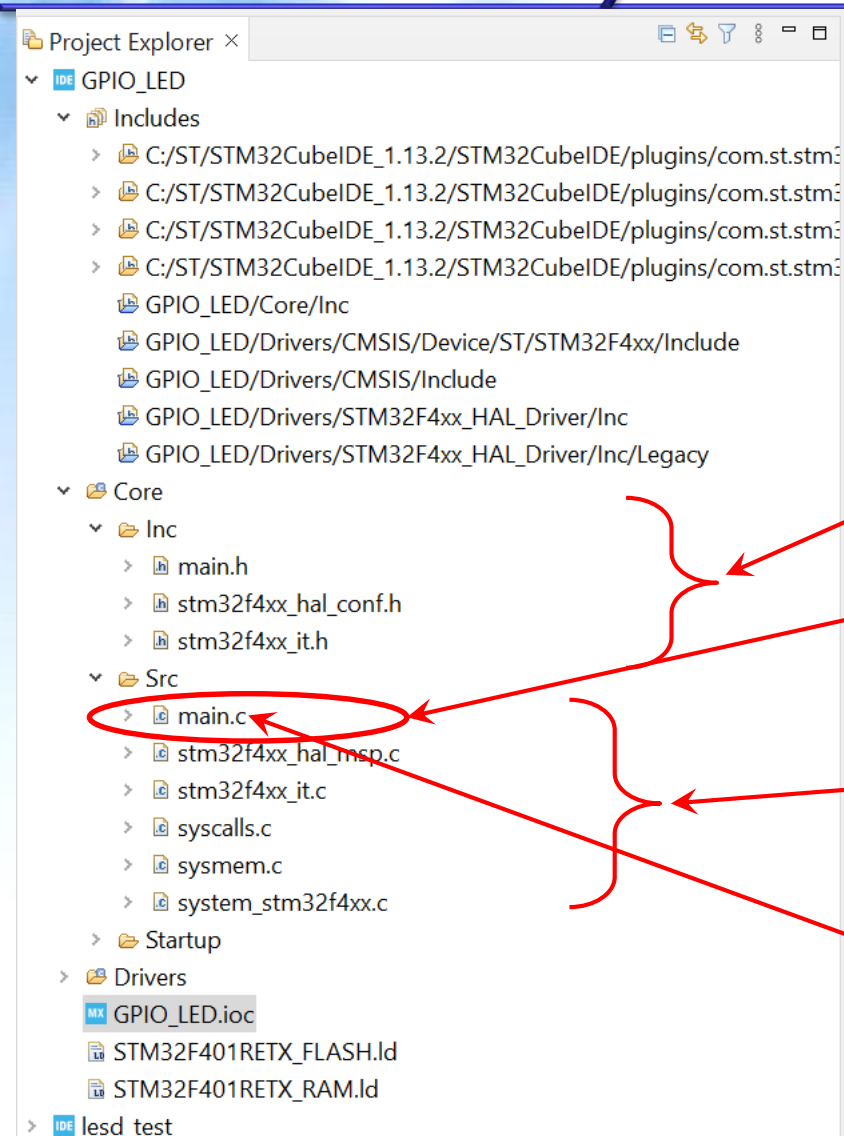
Activation des résistances de pull-up / pull-down (Garder No)

Vitesse du GPIO: Low, Medium, High, Very-High
Mettre « High »

Label utilisateur: Vous pouvez lui donner un nom ou le laisser par défaut

Cliquer ensuite sur « Device Configuration Tool Code Generation »





Toutes les Laibrairies incluses dans ce projet

Fichiers avec l'extension .h (Header Files ou Fichiers d'En-tête)

Le programme principal :
Fichier Source

Fichiers avec l'extension .c (Source Files ou Fichiers Source) :

- Les fichiers source contiennent l'implémentation réelle des fonctions, des structures, des variables globales, etc., déclarées dans les fichiers d'en-tête.
- Les fichiers source sont compilés par le compilateur C pour générer le fichier exécutable final.

Double-Cliquer ensuite sur « main.c »

main.c

```

58 /* USER CODE BEGIN 0 */
59
60 /* USER CODE END 0 */
61
62 /**
63  * @brief The application entry point.
64  * @retval int
65  */
66 int main(void)
67 {
68     /* USER CODE BEGIN 1 */
69
70     /* USER CODE END 1 */
71
72     /* MCU Configuration-----*/
73
74     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
75     HAL_Init();
76
77     /* USER CODE BEGIN Init */
78
79     /* USER CODE END Init */
80
81     /* Configure the system clock */
82     SystemClock_Config();
83
84     /* USER CODE BEGIN SysInit */
85
86     /* USER CODE END SysInit */
87
88     /* Initialize all configured peripherals */
89     MX_GPIO_Init();
90     MX_USART2_UART_Init();
91     /* USER CODE BEGIN 2 */

```

Directives de préprocesseur (Preprocessor Directives):

#include

Déclarations de fonctions (Function Declarations)

void

Fonction main ():

```

int main(void)
{
    /* USER CODE BEGIN 1 */
    .....;
}

```

Définitions de fonctions :

void SystemClock_Config(void)

```

{
    .....;
}

```

Autres fonctions.



Pour voir le contenu d'une Macro: sélectionner la macro et appuyer F3

Activation de toutes les horloges des GPIOs

Fonction permettant de mettre une valeur sur une pin d'un port

GPIO output level

Low

Configuration de de la Pin LD2_Pin

PA5 Configuration :

GPIO output level	Low
GPIO mode	Output Push Pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed	High
User Label	LD2 [Green Led]

```

189  */
190  static void MX_GPIO_Init(void)
191  {
192      GPIO_InitTypeDef GPIO_InitStructure = {0};
193      /* USER CODE BEGIN MX_GPIO_Init_1 */
194      /* USER CODE END MX_GPIO_Init_1 */
195
196      /* GPIO Ports Clock Enable */
197      __HAL_RCC_GPIOC_CLK_ENABLE();
198      __HAL_RCC_GPIOH_CLK_ENABLE();
199      __HAL_RCC_GPIOA_CLK_ENABLE();
200      __HAL_RCC_GPIOB_CLK_ENABLE();
201
202      /*Configure GPIO pin Output Level */
203      HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
204
205      /*Configure GPIO pin : B1_Pin */
206      GPIO_InitStructure.Pin = B1_Pin;
207      GPIO_InitStructure.Mode = GPIO_MODE_IT_FALLING;
208      GPIO_InitStructure.Pull = GPIO_NOPULL;
209      HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStructure);
210
211      /*Configure GPIO pin : LD2_Pin */
212      GPIO_InitStructure.Pin = LD2_Pin;
213      GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
214      GPIO_InitStructure.Pull = GPIO_NOPULL;
215      GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
216      HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStructure);
217
218      /* USER CODE BEGIN MX_GPIO_Init_2 */
219      /* USER CODE END MX_GPIO_Init_2 */
220  }
221
222  /* USER CODE BEGIN 4 */
223
224  /* USER CODE END 4 */
    
```

Port, Pin, valeur (ici 0: RESET)

Un clique sur un lien, vous ramène vers la fonction

La première fonction à être exécutée dans le programme principal est: **main** et **HAL_Init**.

HAL: **H**ardware **A**bstracti**o**n **L**ayer (Couche d'abstraction matérielle): Programmes intermédiaire entre le logiciel et le matériel. Il offre des fonctions standardisées de manipulation du matériel tout en cachant les détails techniques de la mise en œuvre.

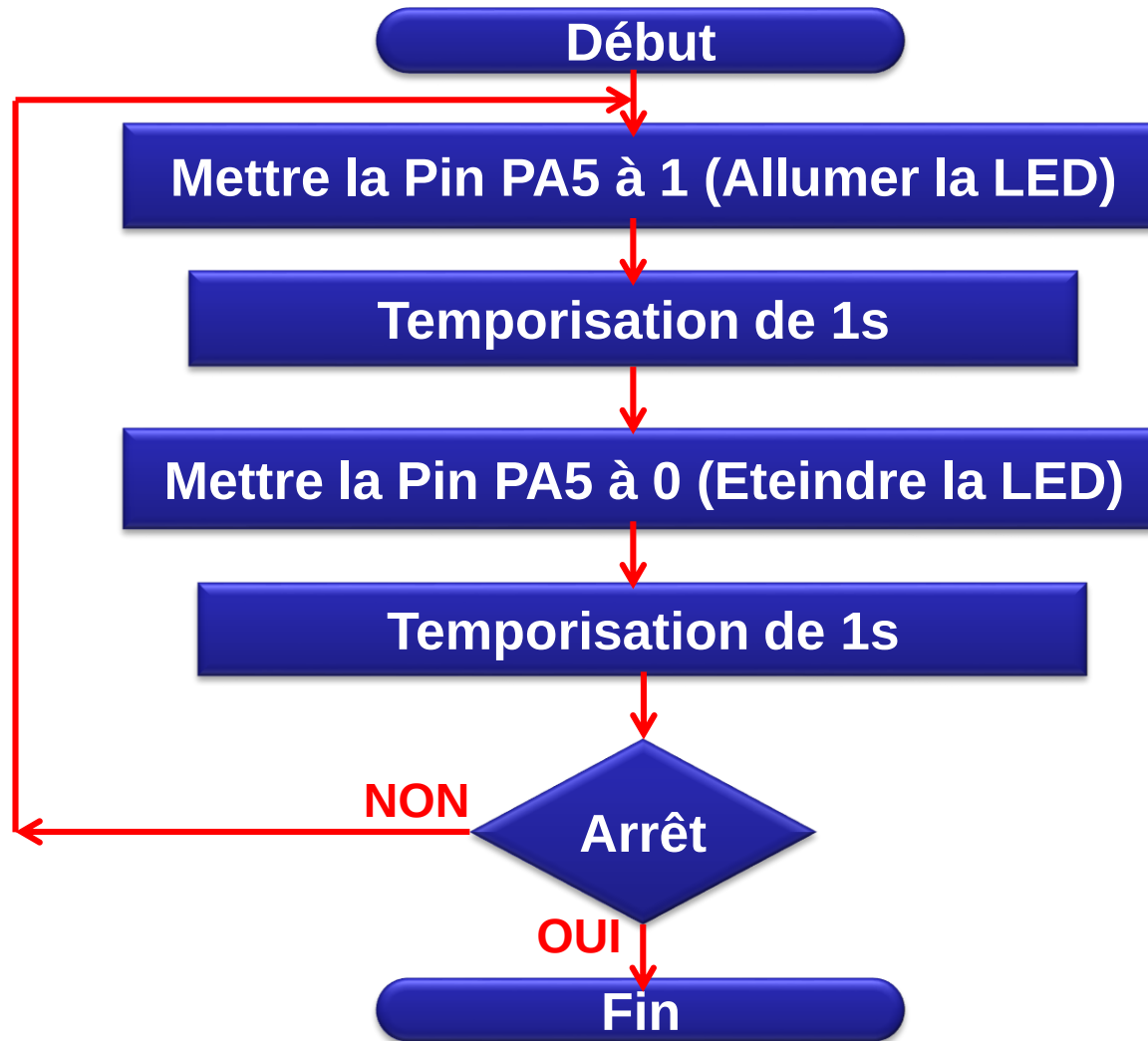
Ici: **HAL_Init** permet une réinitialisation de tous les périphériques, Initialise l'interface Flash et le SysTick

Le **SysTick** est un timer intégré. Il s'agit d'un compteur à décompte automatique de 24 bits qui peut être utilisé pour générer des retards, pour mesurer le temps ou pour déclencher des interruptions périodiques.

```
58 /* USER CODE BEGIN 0 */
59
60 /* USER CODE END 0 */
61
62 /**
63  * @brief The application entry point.
64  * @retval int
65  */
66 int main(void)
67 {
68     /* USER CODE BEGIN 1 */
69
70     /* USER CODE END 1 */
71
72     /* MCU Configuration-----*/
73
74     /* Reset of all peripherals, Initializes the Flash interface and the SysTick. */
75     HAL_Init();
76
77     /* USER CODE BEGIN Init */
78
79     /* USER CODE END Init */
80
81     /* Configure the system clock */
82     SystemClock_Config();
83
84     /* USER CODE BEGIN SysInit */
85
86     /* USER CODE END SysInit */
87
88     /* Initialize all configured peripherals */
89     MX_GPIO_Init();
90     MX_USART2_UART_Init();
91     /* USER CODE BEGIN 2 */
```

Configuration
d'horloge

Organigramme du programme



Insertion du code utilisateur dans main.c et compilation

 Il faut s'assurer d'écrire votre code dans cet emplacement
(entre `/* USER CODE BEGIN 3 */`
Et `/* USER CODE END 3 */`)
Sinon il sera supprimé lors de la generation du code
"Code Generation"

```

97  while (1)
98  {
99      /* USER CODE END WHILE */
100
101      /* USER CODE BEGIN 3 */
102      HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET); /* Mettre la pin LD2_pin à 1 */
103      HAL_Delay(1000); /* Temporistaion de 1000ms=1s */
104      HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET); /* Mettre la pin LD2_pin à 0 */
105      HAL_Delay(1000); /* Temporistaion de 1000ms=1s */
106  }
107  /* USER CODE END 3 */
108  }

```

Lancer la
compilation

```

92  /* USER CODE END 2 */
93  /* USER CODE END 2 */
94
95  /* Infinite loop */
96  /* USER CODE BEGIN WHILE */
97  while (1)
98  {
99      /* USER CODE END WHILE */
100
101      /* USER CODE BEGIN 3 */
102  }
103  /* USER CODE END 3 */
104  }

```

CDT Build Console [GPIO_LED]

arm-none-eabi-size	GPIO_LED.elf					
text	data	bss	dec	hex	filename	
7752	20	1636	9408	24c0	GPIO_LED.elf	

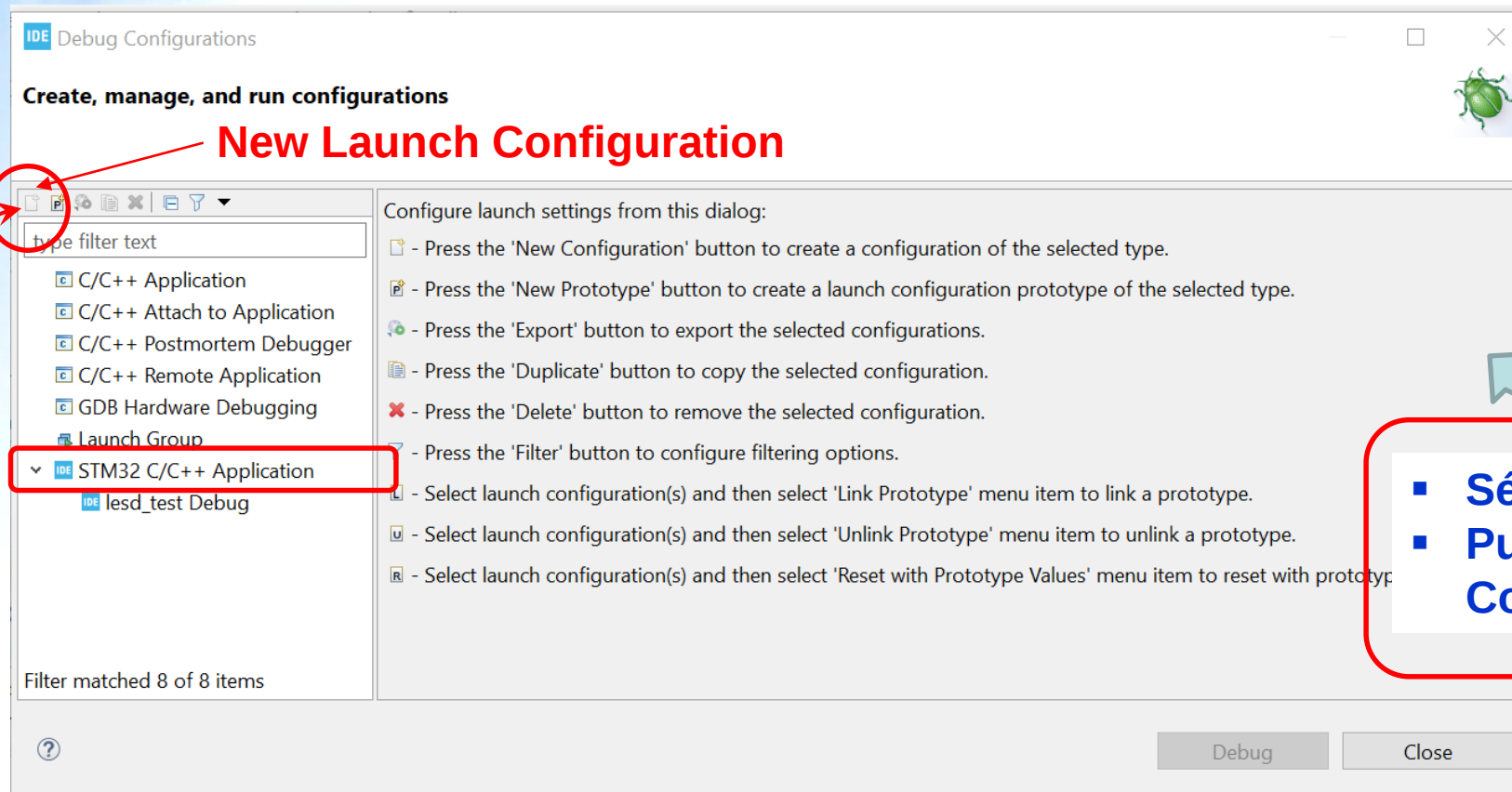
Finished building: default.size.stdout

18:11:18 Build Finished. 0 errors, 0 warnings. (took 215ms)



PAS D'ERREURS !!!

La dernière étape avant de flasher le code est de créer « Debug Configuration »



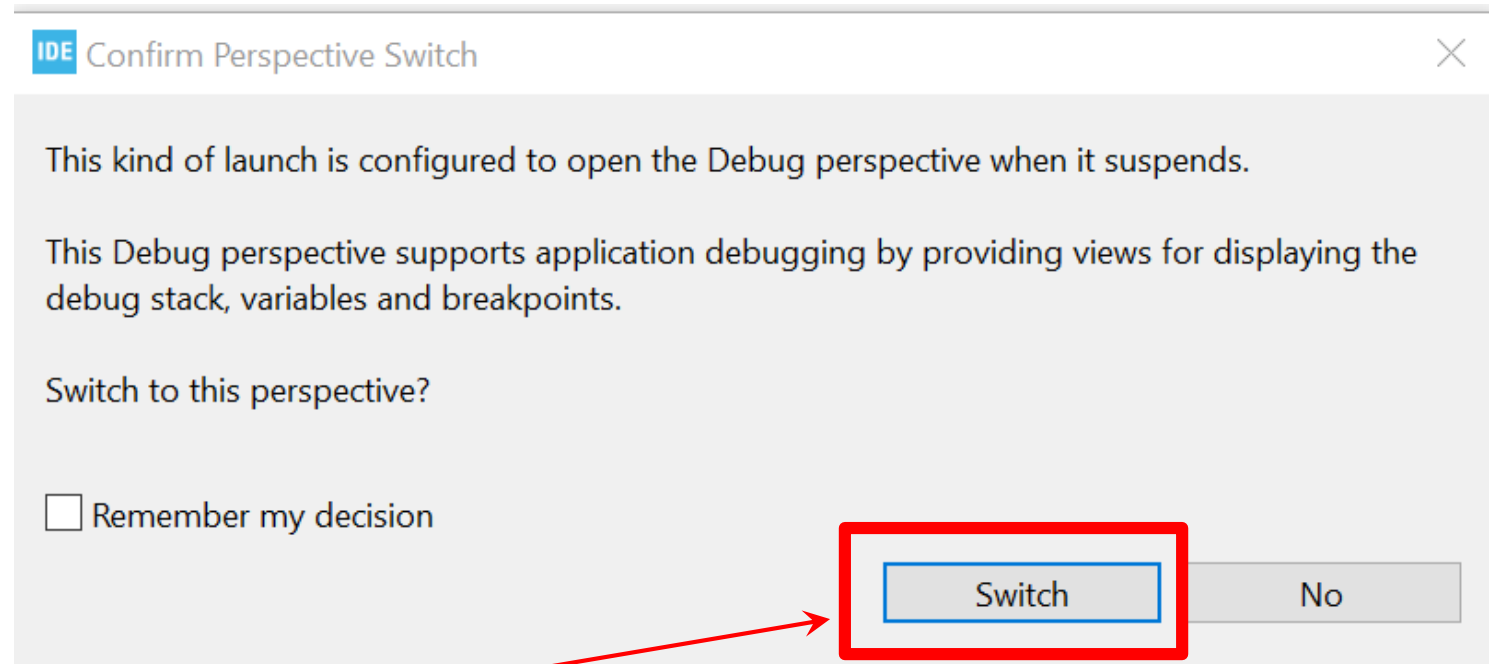
- Sélectionner STM32C/C++Application
- Puis cliquer sur « New Launch Configuration »



S'assurer que votre carte est connectée au port USB (utiliser un câble mini USB type V3)

The screenshot shows the 'Debug Configurations' window in an IDE. The left sidebar lists various debug configurations, with 'GPIO_LED Debug (1)' selected under the 'STM32 C/C++ Application' category. The main panel shows the configuration for 'GPIO_LED Debug (1)'. Annotations with red boxes and arrows point to specific elements:

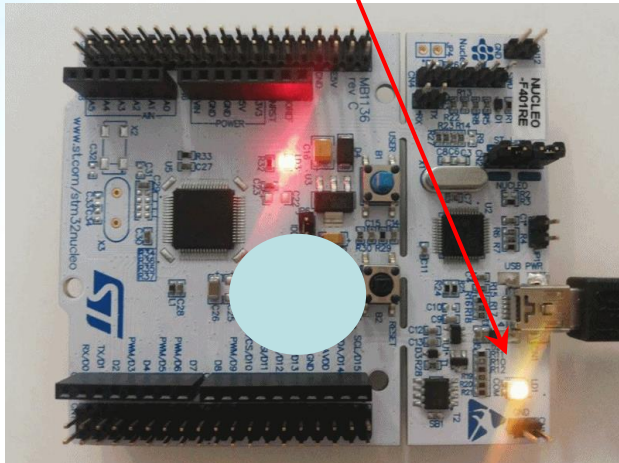
- S'assurer que le fichier .elf est généré est que la compilation est réalisée sans erreurs**: Points to the 'C/C++ Application' field, which contains 'Debug/GPIO_LED.elf'.
- Cliquer sur Debugger**: Points to the 'Debugger' tab in the configuration window.
- Activer SWD Cocher ST-LINK**: Points to the 'Interface' section where 'SWD' is selected and 'ST-LINK S/N' is checked.
- Cliquer sur « Scan » Cela permet de détecter le ST-LINK intégré à la carte**: Points to the 'Scan' button next to the 'ST-LINK S/N' field.
- N'oubliez pas d'appliquer les changements**: Points to the 'Apply' button at the bottom right.
- Le Clique sur « Debug » permet de flasher le code sur le microcontrôleur**: Points to the 'Debug' button at the bottom right.



Le Cliquer sur « Switch »

Quand l'étape de Débogage est terminée vous apercevez:

- Un message « Download verified successfully »
- Le programme s'arrête à la ligne 75 (HAL_Init en **surbrillance verte**)
- La LED de la carte ST-LINK clignote **vert/rouge**



Câble mini USB V3



main.c × GPIO_LED.ioc main.h startup_stm32f401retx.s

```

67 {
68  /* USER CODE BEGIN 1 */
69
70  /* USER CODE END 1 */
71
72  /* MCU Configuration-----
73
74  /* Reset of all peripherals, Initializes the Flash interface and the System
75  HAL_Init();
76
77  /* USER CODE BEGIN Init */
78
79  /* USER CODE END Init */
80
81  /* Configure the system clock */
82  SystemClock_Config();
83
84  /* USER CODE BEGIN SysInit */
85
86  /* USER CODE END SysInit */
87
88  /* Initialize all configured peripherals
89  MX_GPIO_Init();
90  MX_USART2_UART_Init();
91  /* USER CODE BEGIN 2 */
92
93  /* USER CODE END 2 */

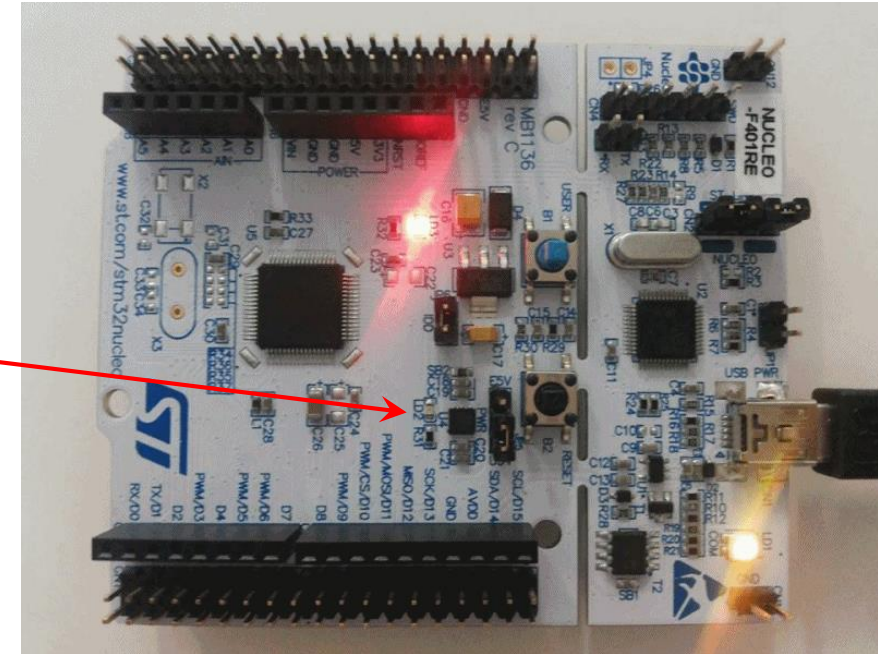
```

Download verified successfully

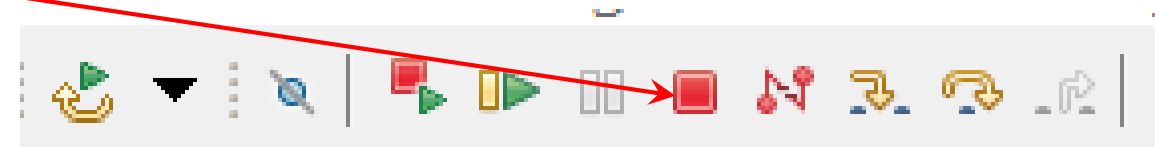
Le programme est maintenant à l'étape HAL_Init Cette fonction n'est pas encore exécutée



Cliquer enfin sur ce bouton **RESUME** en **VERT**
Cela permet d'exécuter la suite du programme,
et en particulier le programme du clignotement
de la LED **verte** à une fréquence de 0.5Hz (1sec
ON et 1s OFF).



Vous pouvez **STOPPER** le debugage en
appuyant sur le bouton **STOP**
Cela veut dire que le Débogage est terminé
mais le programme est en exécution sur le
microcontrôleur



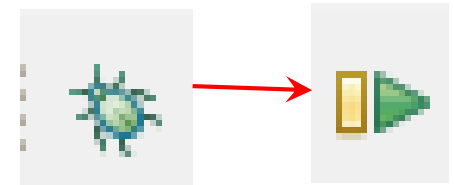
On peut ajouter des points d'arrêt dans le programme (BREAK POINTS)
On ajoute un point d'arrêt en double-cliquant sur la ligne (Exemple: ligne 104)

- "Breakpoint" est une fonctionnalité de débogage qui vous permet de suspendre l'exécution de votre programme sur une ligne de code spécifique pendant le débogage.
- Lorsque vous définissez un point d'arrêt, le programme s'exécutera jusqu'à ce qu'il atteigne cette ligne, puis il se mettra en pause, vous permettant d'inspecter l'état du programme, ses variables et d'autres informations à ce stade.
- Il s'agit d'un outil indispensable pour déboguer et analyser le comportement de votre application sur un microcontrôleur STM32.

```
93  /* USER CODE END 2 */
94
95  /* Infinite loop */
96  /* USER CODE BEGIN WHILE */
97  while (1)
98  {
99      /* USER CODE END WHILE */
100
101      /* USER CODE BEGIN 3 */
102      HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET); /* Mettre la
103      HAL_Delay(1000); /* Temporistaion de 1000ms=1s */
104      HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET); /* Mettre ]
105      HAL_Delay(1000); /* Temporistaion de 1000ms=1s */
106  }
107  /* USER CODE END 3 */
108 }
109
110 /**
```

Ce symbole indique qu'il y a un point d'arrêt à la ligne 104

Cliquer de nouveau sur **DEBUG** puis sur **RESUME**



Une autre façon de programmer le Clignotement de la LED

Utiliser TogglePin:

HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin)

Dans STM32CubeIDE, le terme « **Toggle Pin** » fait généralement référence à l'action de changer l'état d'une broche numérique sur un microcontrôleur STM32 de haut (niveau logique 1) à bas (niveau logique 0) ou vice versa (**Basculer l'état**).

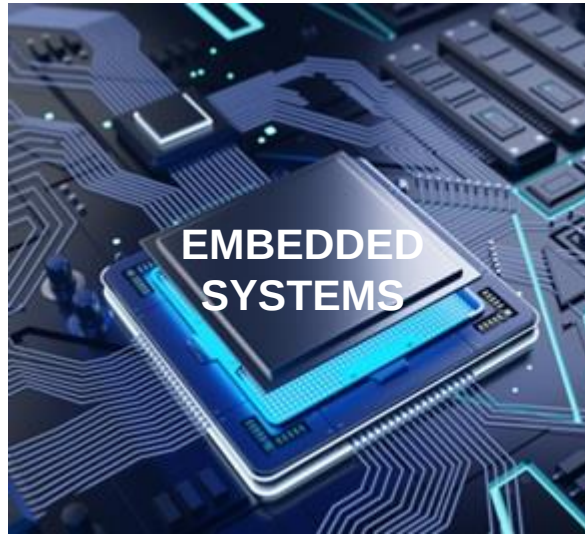
Le basculement d'une broche est une opération courante dans les systèmes embarqués où vous devez allumer ou éteindre un appareil connecté, tel qu'une LED, un moteur ou un capteur.

```
94
95  /* Infinite loop */
96  /* USER CODE BEGIN WHILE */
97  while (1)
98  {
99      /* USER CODE END WHILE */
100
101      /* USER CODE BEGIN 3 */
102
103      HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
104      HAL_Delay(1000); /* Temporistaion de 1000ms=1s */
105
106  }
107  /* USER CODE END 3 */
108 }
109
```

La Pin

Le port

Essayez de changer la fréquence de clignotement !!!
THAT'S ALL!



FIN !
Questions ?