Université IBN TOFAIL – Kénitra
ECOLE NATIONALE DES SCIENCES APPLIQUEES

Cours Microcontrôleurs Avancés

CH4: LE GPIO en Entrée

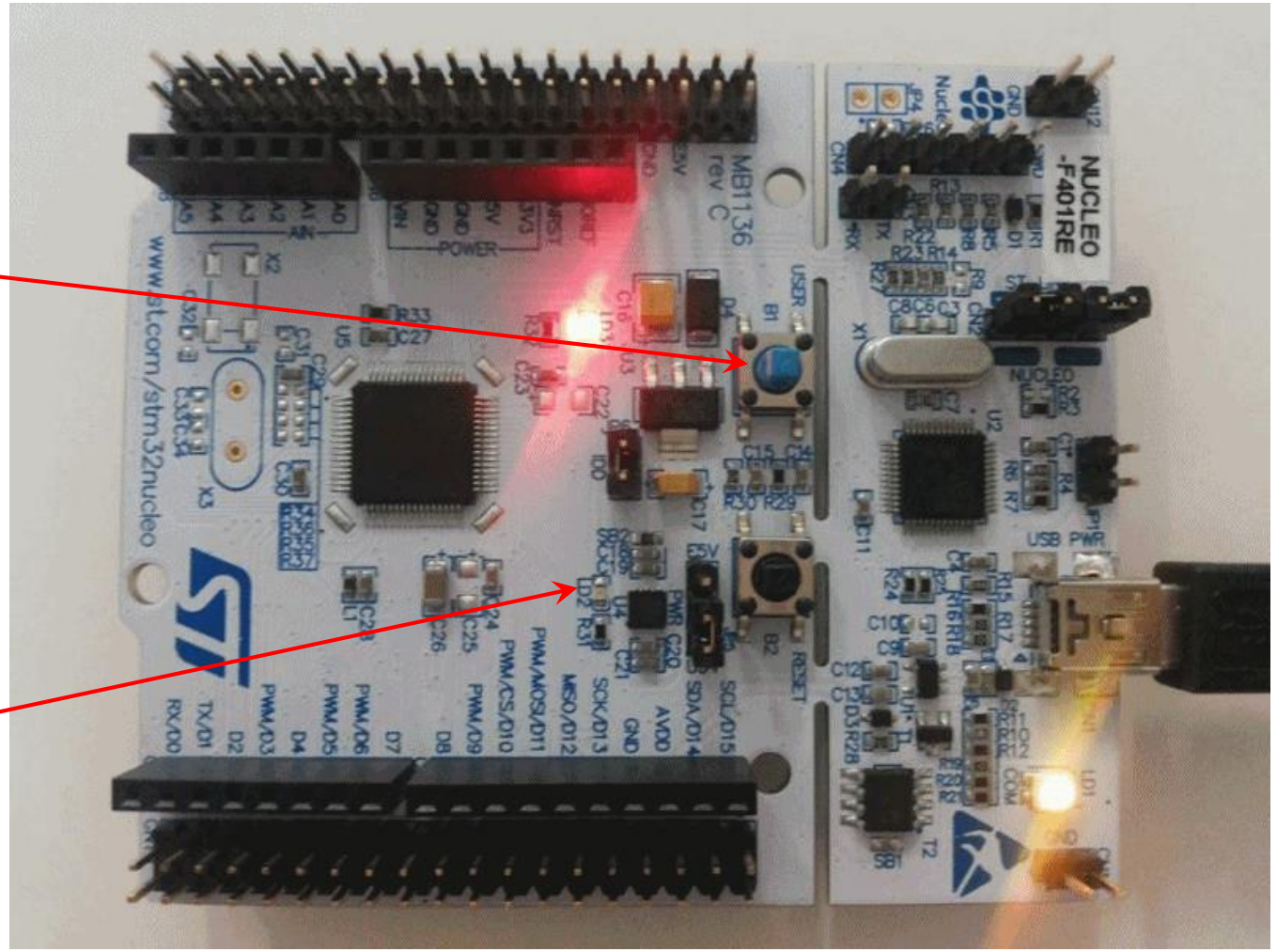
Par:

Hassan EL FADIL

elfadil.h@gmail.com

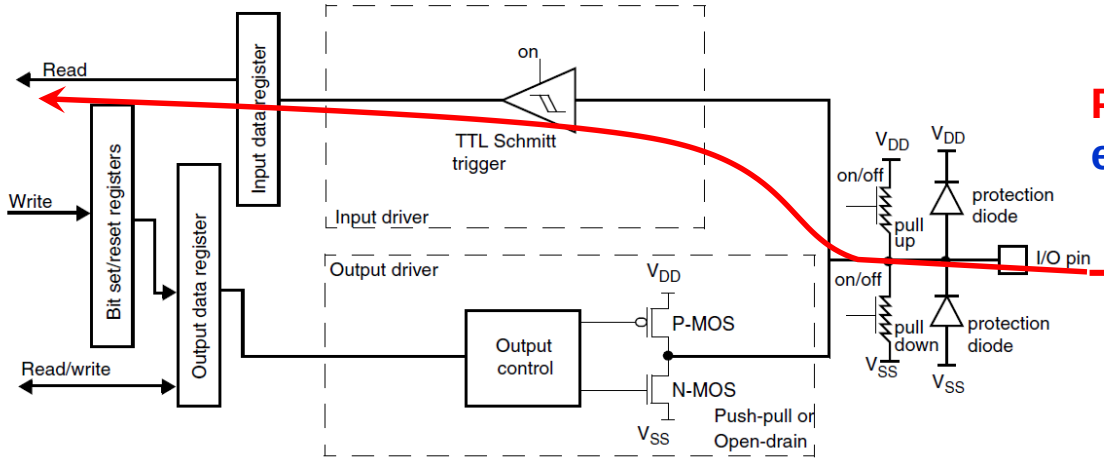
Deuxième Application

Bouton poussoir utilisateur B1 (Bleu)



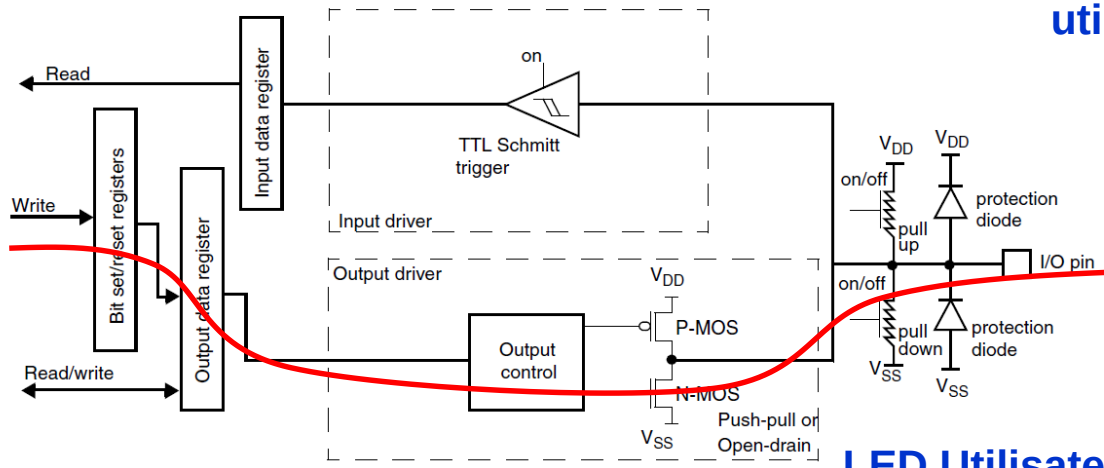
- Programme d'allumage de la LED utilisateur (LED Verte) de la carte par l'appui du bouton utilisateur B1 (Bouton Bleu)
- Appuyer sur B1 → LED Allumée
- Relâcher B1 → LED Eteinte

Ports GPIO utilisés



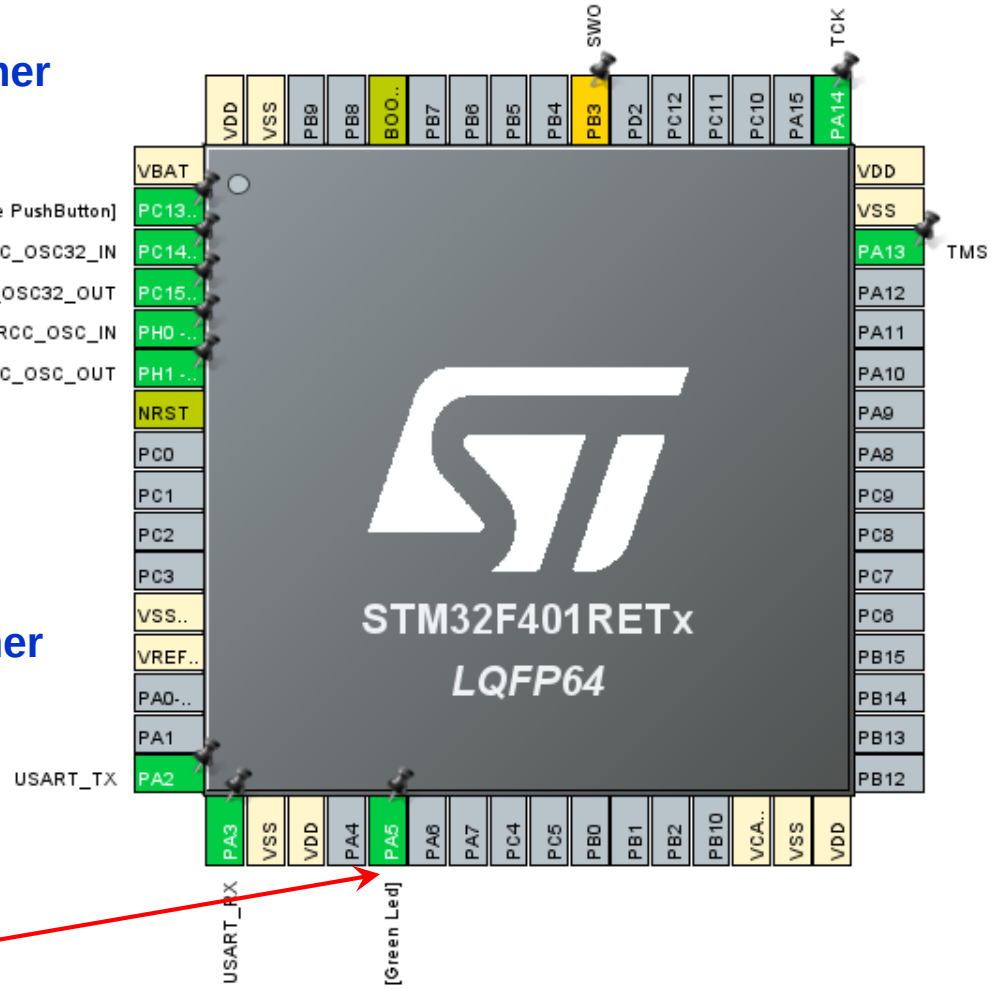
PC13: à programmer en entrée

Bouton poussoir utilisateur B1 (Bleu)

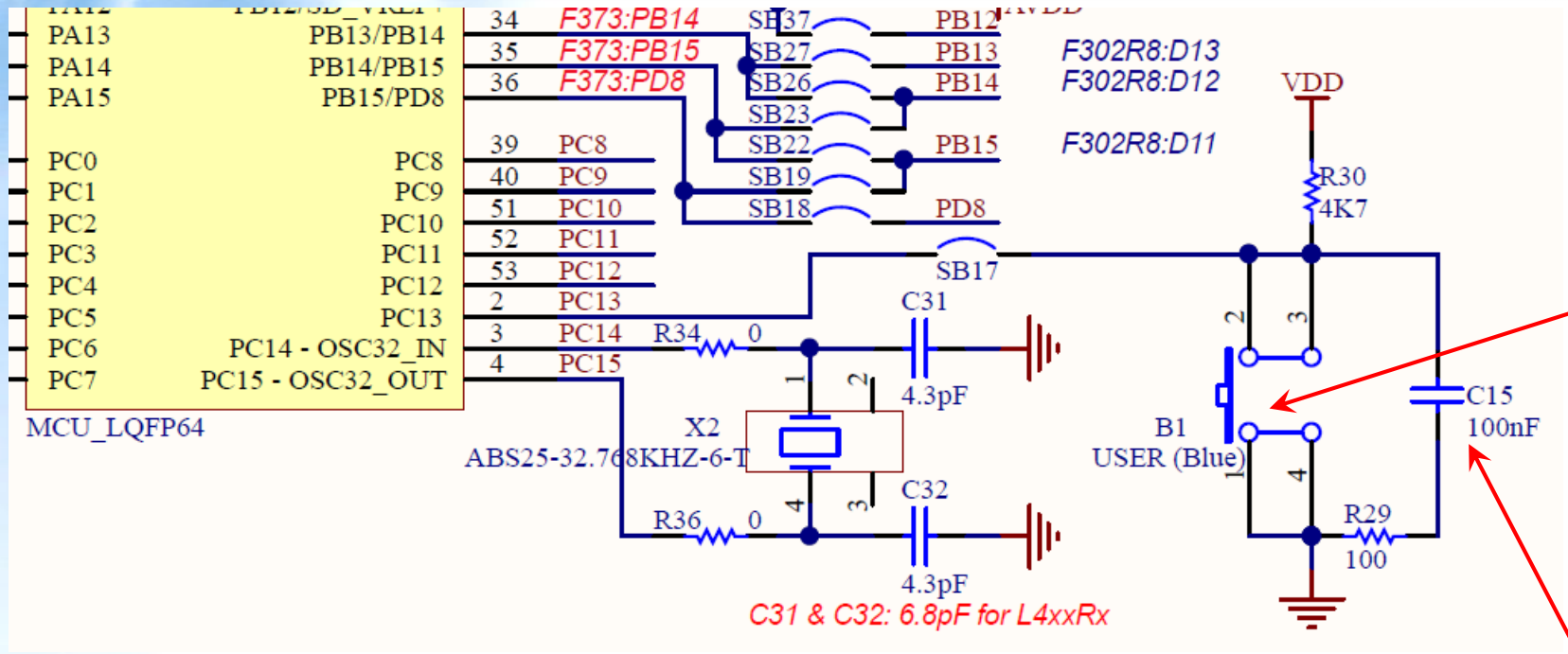


PA5: à programmer en sortie

LED Utilisateur LD2 (verte)

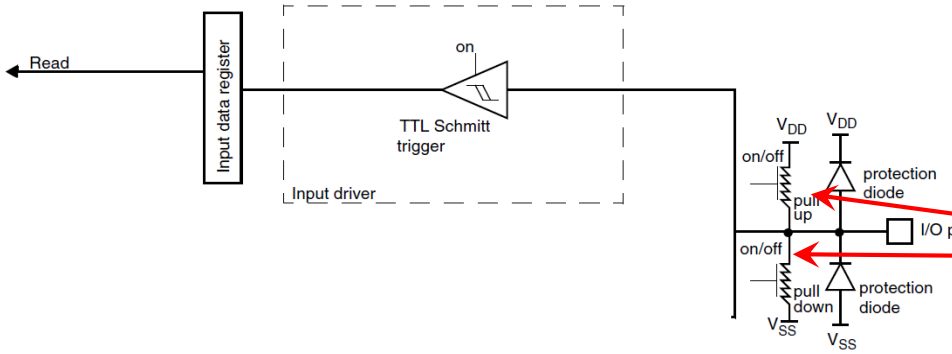


Le Bouton B1 et le microcontrôleur



Bouton poussoir utilisateur B1 (Bleu)

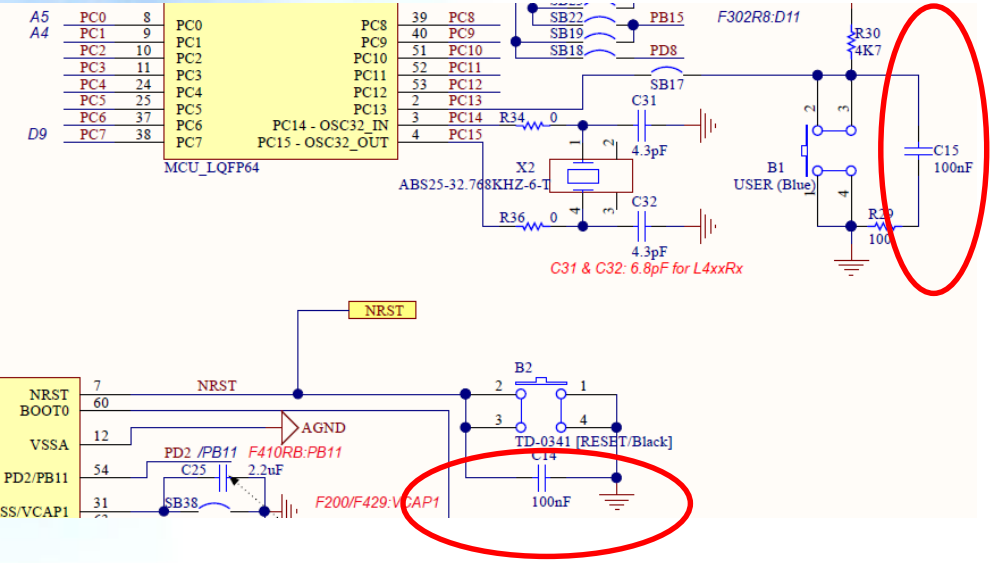
- Si le bouton n'est pas appuyé, la Pin PC13 est liée à VDD=3.3V. C'est un **« 1 »**
- Si le bouton est appuyé, la Pin PC13 est liée à la masse. C'est un **« 0»**
- Le condensateur C15 permet la mise à 1 de la Pin au relâchement du Bouton.



Pin PC13

Le port sera programmée en entrée, les résistances de pull-up seront désactivées (switch off).

Remarques sur le rôle des condensateurs en parallèle des Boutons Poussoirs



Voici quelques fonctionnalités de ces condensateurs:

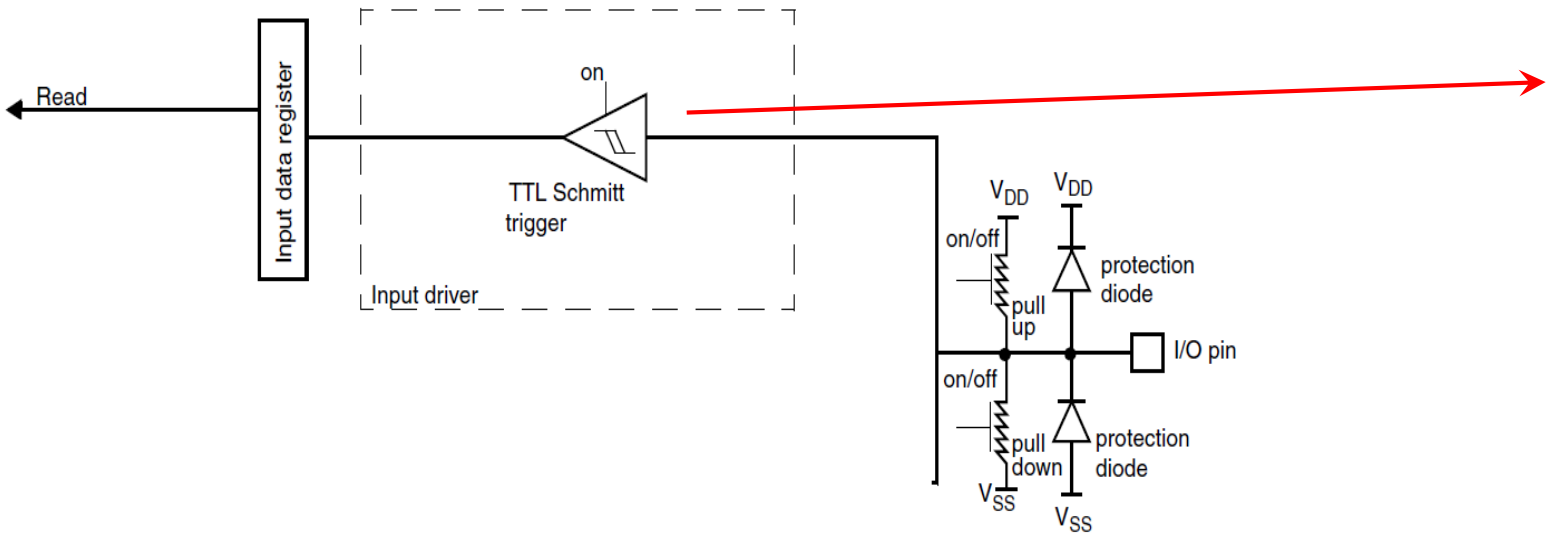
1.Filtrage du signal : Le condensateur agit comme un filtre pour le signal. Lorsque le bouton est relâché, le condensateur se charge rapidement, ce qui provoque une transition douce du niveau logique bas au niveau logique haut sur la broche. Cela empêche les rebonds de signal, qui sont des fluctuations rapides du signal qui peuvent survenir lorsqu'un bouton est enfoncé ou relâché.

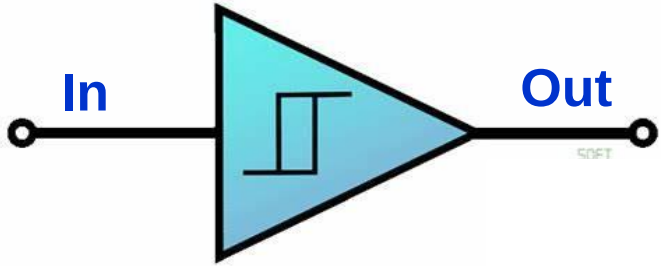
2.Stabilisation du niveau logique : Une fois le condensateur chargé, il maintient le niveau logique haut sur la broche tant que le condensateur n'est pas déchargé..

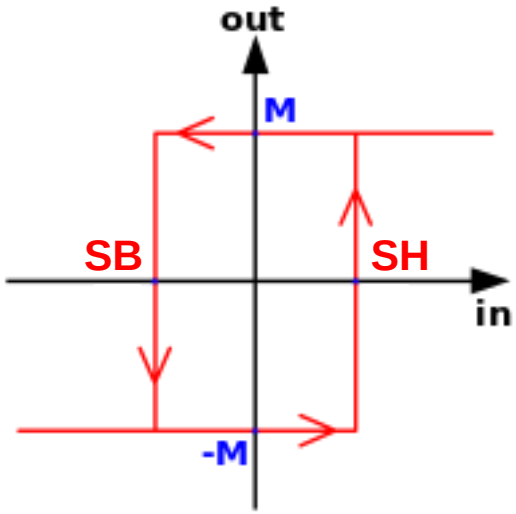
3.Prévention des réinitialisations indésirables : Le condensateur en parallèle peut également aider à prévenir les réinitialisations indésirables dues à des bruits électromagnétiques ou d'autres perturbations électriques.

4.Amélioration de la robustesse du système : En réduisant les chances de réinitialisations indésirables, le condensateur contribue à rendre le système plus robuste, en particulier dans des environnements électriquement bruyants.

Trigger de Schmitt

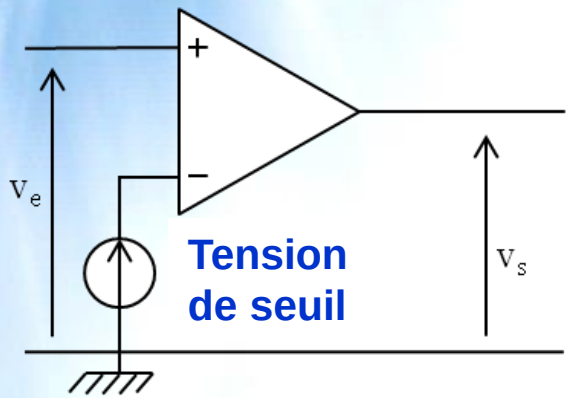




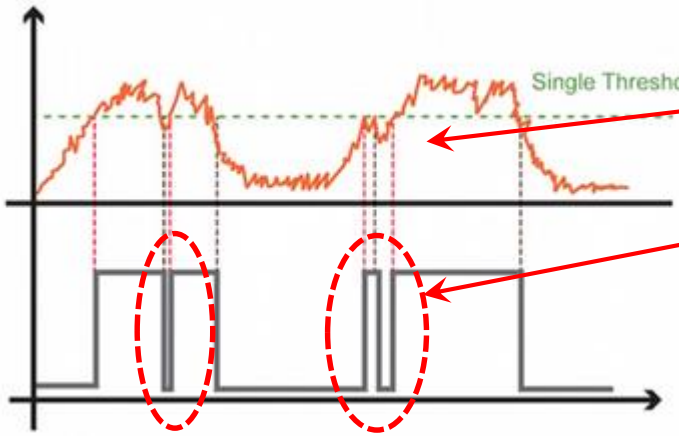


- Comparateur utilisant deux niveaux de tension de seuil différents pour les fronts montant et descendant (seuil haut SH et le seuil bas SB).
- Ceci est utile car cela peut éviter les erreurs lorsque nous avons des signaux d'entrée bruyants à partir desquels nous voulons obtenir des signaux d'onde carrée.
- Pour cela, il suffit que l'écart entre le seuil haut SH et le seuil bas SB soit supérieur à l'amplitude crête-à-crête du bruit ;

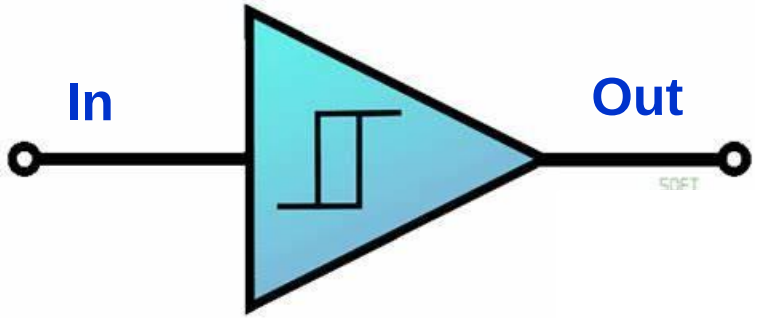
Trigger de Schmitt



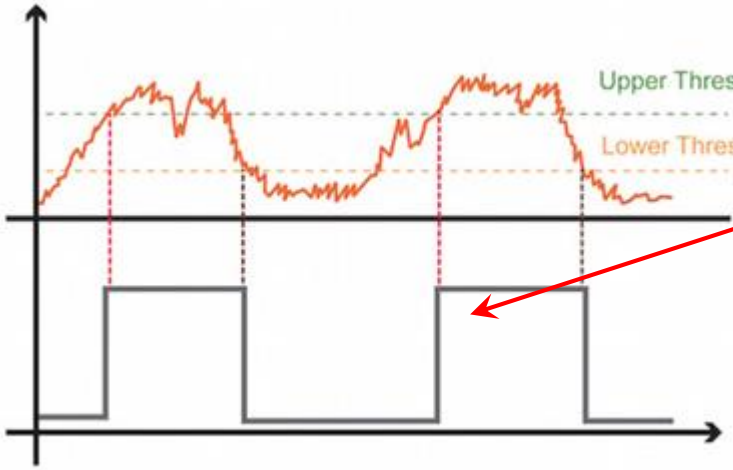
Comparateur à un seuil



En présence du bruit, il y a apparition des commutations indésirables



Comparateur à Hystérésis
(Trigger de Schmitt)



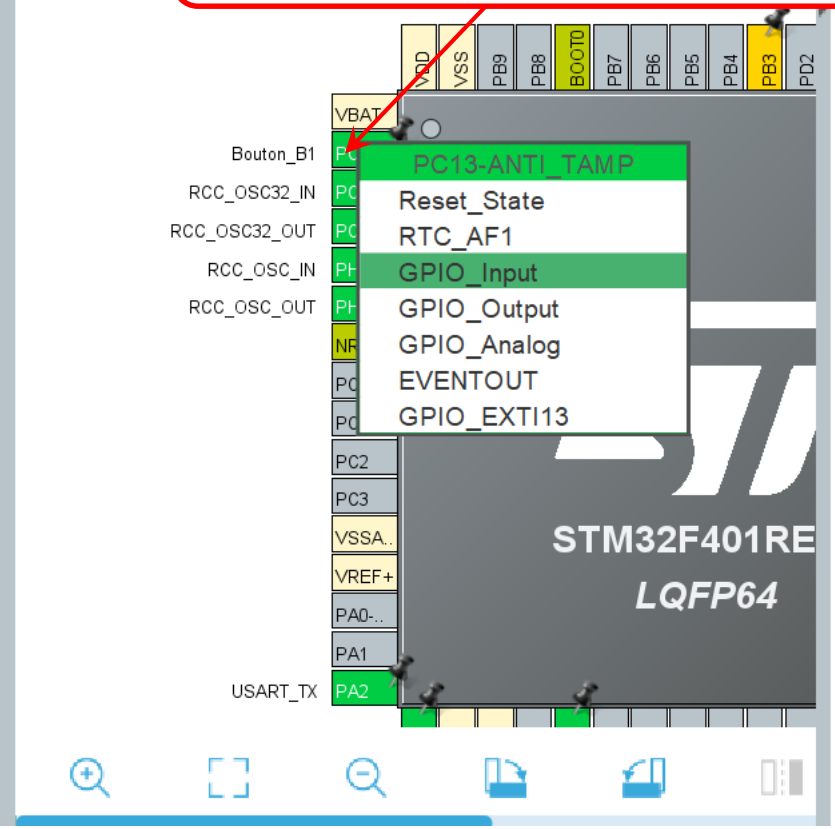
L'utilisation du Trigger de Schmitt permet d'éviter les commutations indésirables causées par la présence du bruit

Configuration du GPIO

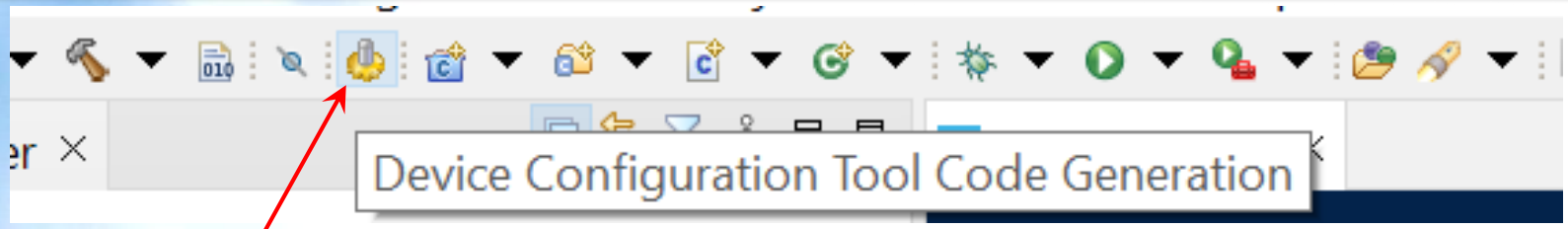
1 On commence par créer un nouveau projet STM32CubeIDE (**Voir Chapitre 3**)
Appeler votre projet: **GPIO_LED_Bouton**

3 **Ensuite, Sélectionnez la ligne PC13 et Donnez un label (Bouton_B1)**

2 **Ensuite, cliquez sur la pin PC13 et configurez le GPIO en entrée (GPIO_Input)**

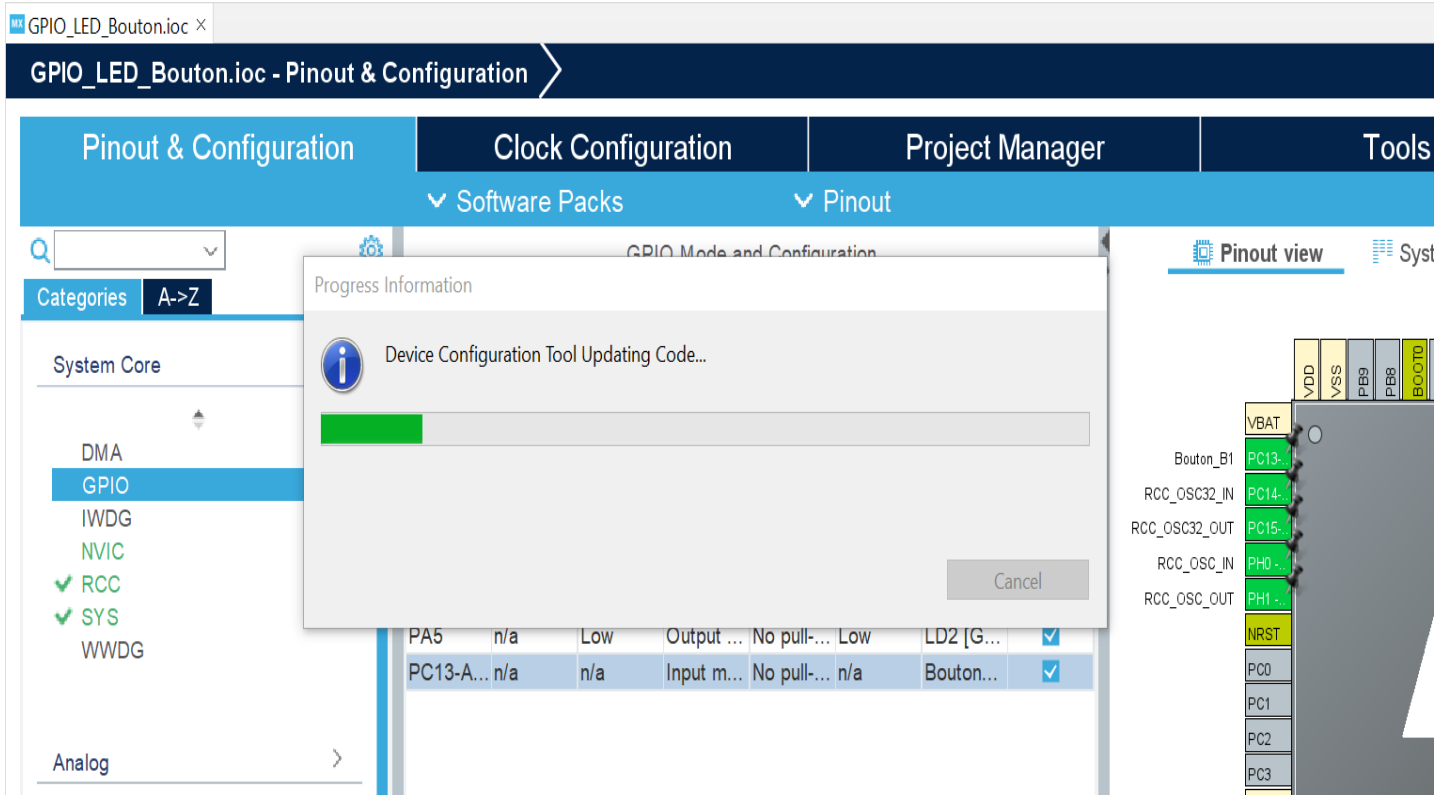


Génération du Code

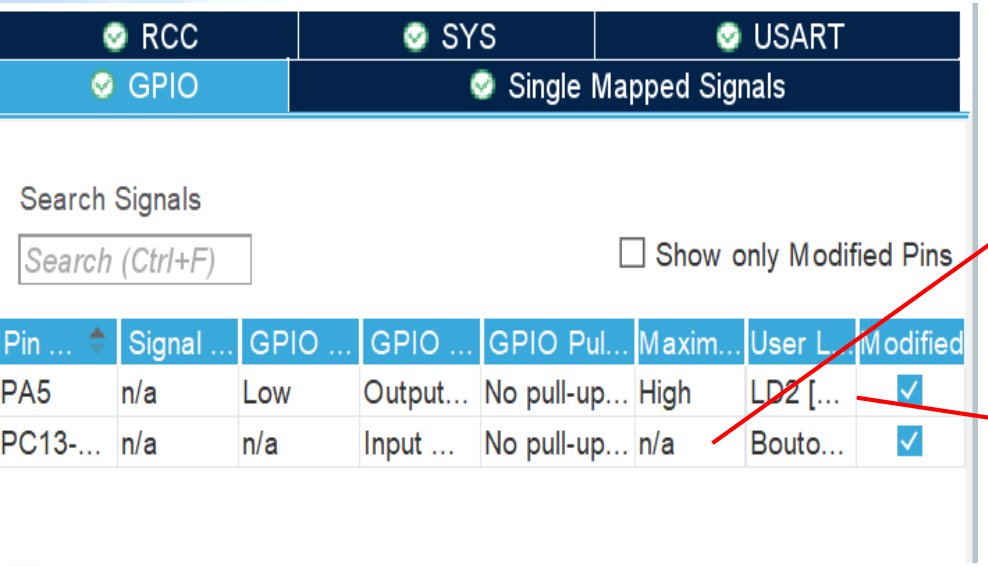


4

Cliquer ensuite sur « Device Configuration Tool Code Generation »

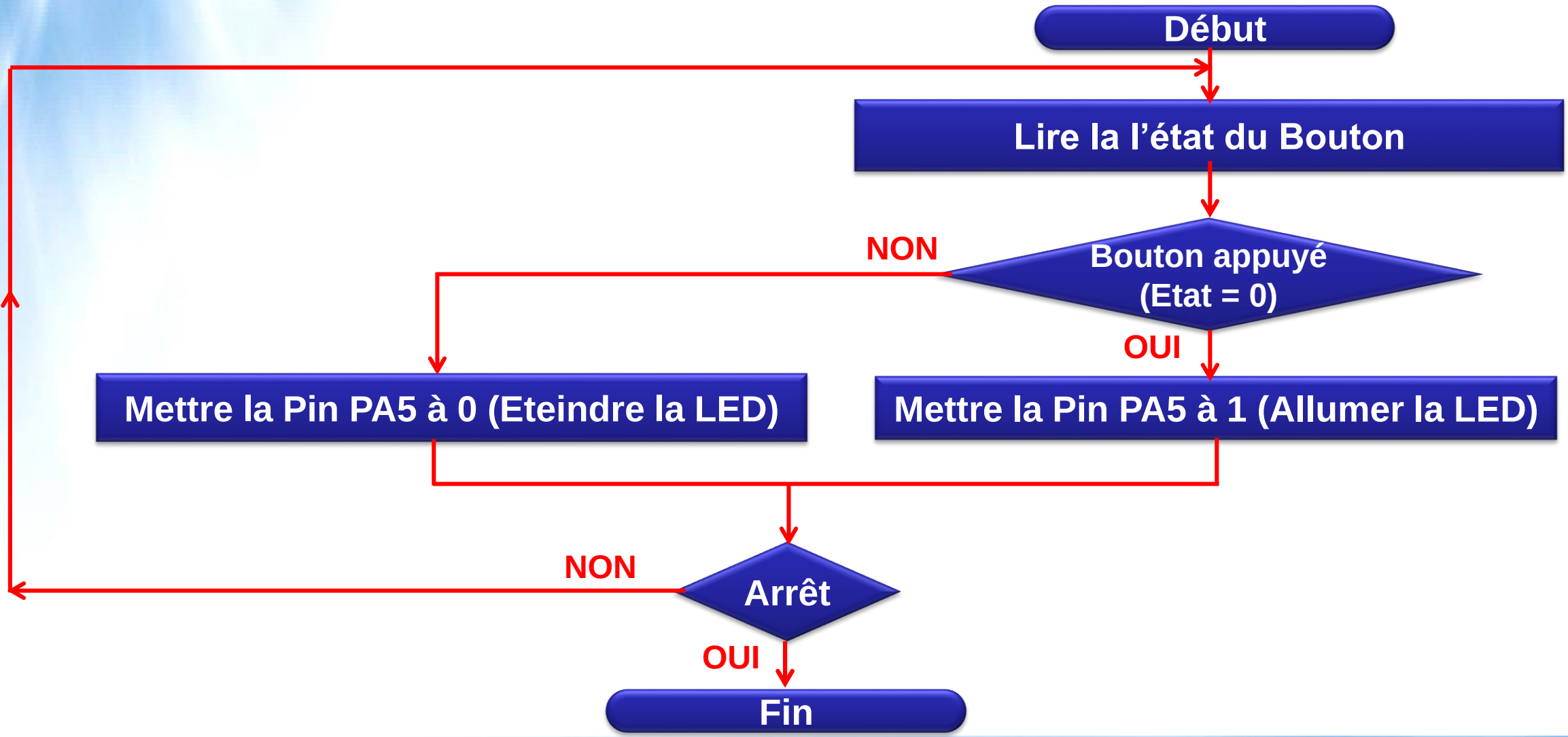


Génération du Code




```
202   /*Configure GPIO pin Output Level */
203   HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
204
205   /*Configure GPIO pin : Bouton_B1_Pin */
206   GPIO_InitStruct.Pin = Bouton_B1_Pin;
207   GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
208   GPIO_InitStruct.Pull = GPIO_NOPULL;
209   HAL_GPIO_Init(Bouton_B1_GPIO_Port, &GPIO_InitStruct);
210
211   /*Configure GPIO pin : LD2_Pin */
212   GPIO_InitStruct.Pin = LD2_Pin;
213   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
214   GPIO_InitStruct.Pull = GPIO_NOPULL;
215   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
216   HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
217
```

Organigramme du programme



Le Code

5

Dans main.c, définir une variable en entier non-signé sur 8 bits en utilisant la fonction uint8_t : **val_Bouton**

```
88  /* Initialize all configured peripherals */
89  MX_GPIO_Init();
90  MX_USART2_UART_Init();
91  /* USER CODE BEGIN 2 */
92  uint8_t val_Bouton=0;
93  /* USER CODE END 2 */
94
95  /* Infinite loop */
96  /* USER CODE BEGIN WHILE */
97  while (1)
98  {
99    /* USER CODE END WHILE */
```

6

Affecter l'état de la Pin du GPIO connecté au Bouton B1 à la variable : **val_Bouton**

```
101   /* USER CODE BEGIN 3 */
102     /* Lire l'état de la pin du GPIO connectée au Bouton B1 */
103     val_Bouton=HAL_GPIO_ReadPin(Bouton_B1_GPIO_Port, Bouton_B1_Pin);
104     /* Vérifier si cet état est HAUT ou BAS */
105     /* Si bouton appuyé, alors allumer la LED */
106     /* Sinon éteindre la LED
```

Le Code

REMARQUES:

- La fonction HAL_GPIO_ReadPin se trouve dans le fichier stm32f4xx_hal_gpio.c
- Ces argument sont définies dans le fichier main.h.
- Pour ouvrir une fonction, sélectionner cette fonction dans main.c puis appuyer sur F3. Le fichier main.h s'ouvre.

```
59 /* Private defines -----------------
60 #define Bouton_B1_Pin GPIO_PIN_13
61 #define Bouton_B1_GPIO_Port GPIOC
62 #define USART_TX_Pin GPIO_PIN_2
63 #define USART_TX_GPIO_Port GPIOA
```

```
59 /* Private defines -----------------
60 #define Bouton_B1_Pin GPIO_PIN_13
61 #define Bouton_B1_GPIO_Port GPIOC
62 #define USART_TX_Pin GPIO_PIN_2
63 #define USART_TX_GPIO_Port GPIOA
```

Le Code

7

Si la variable val_Bouton =0 (Bouton appuyé) alors allumer la LED sinon éteindre la LED

8

Lancer la compilation

```
Finished building: GPIO_LED_Bouton.list

12:42:27 Build Finished. 0 errors, 0 warnings. (took 1s.265ms)
```

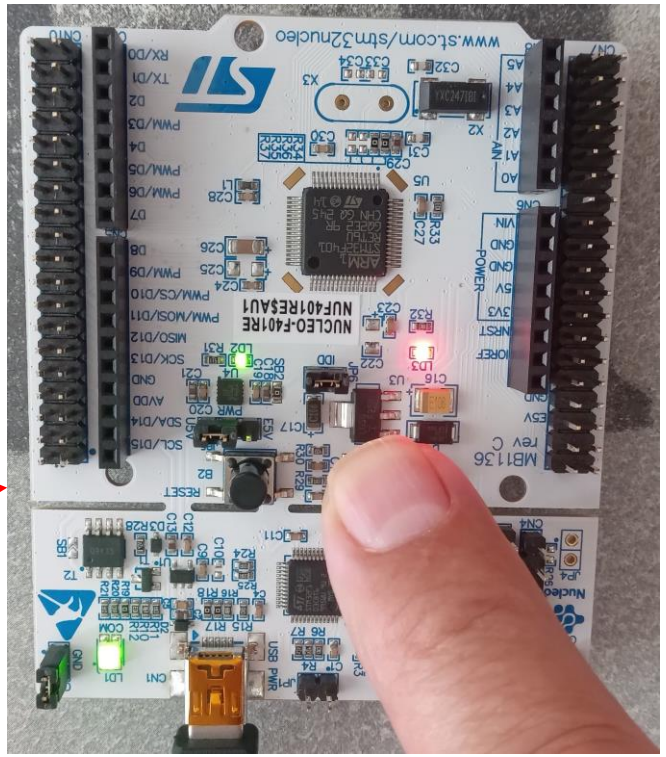
```
97  while (1)
98  {
99    /* USER CODE END WHILE */
100
101   /* USER CODE BEGIN 3 */
102     /* Lire l'état de la pin du GPIO connectée au Bouton B1 */
103     val_Bouton=HAL_GPIO_ReadPin(Bouton_B1_GPIO_Port, Bouton_B1_Pin);
104     /* Vérifier si cet état est HAUT ou BAS */
105     if (val_Bouton==0)
106     {
107         /* Si bouton appuyé, alors allumer la LED */
108         HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, SET);
109     }
110     else
111     {
112         /* Sinon éteindre la LED */
113         HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, RESET);
114     }

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

Débogage et Vérification

Lancer le Débogage

Cliquer enfin sur ce bouton RESUME en VERT
Cela permet d'exécuter la suite du programme.
Si vous appuyez sur le bouton Bleu normalement la LED Verte s'allume, si vous relâchez le bouton la LED s'éteint.



VERIFIER !!! THAT'S ALL!

Une autre façon de Programmer !

```
92 #define verifie_bouton  HAL_GPIO_ReadPin(Bouton_B1_GPIO_Port, Bouton_B1_Pin)
93 #define appuye  0
94   /* USER CODE END 2 */
95
96   /* Infinite loop */
97   /* USER CODE BEGIN WHILE */
98   while (1)
99   {
100    /* USER CODE END WHILE */
101
102    /* USER CODE BEGIN 3 */
103
104      if (verifie_bouton==appuye)
105      {
106          /* Si bouton appuyé, alors allumer la LED */
107          HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, SET);
108      }
109      else
110      {
111          /* Sinon éteindre la LED */
112          HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, RESET);
113      }
114
115
116  }
117  /* USER CODE END 3 */
118 }
119
```

Définition d'une Macro

```
#define verifie_bouton HAL_GPIO_ReadPin(Bouton_B1_GPIO_Port, Bouton_B1_Pin)
#define appuye 0
while (1)
{

if (verifie_bouton==appuye)
{
/* Si bouton appuyé, alors allumer la LED */
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, SET);
}
else
{
/* Sinon éteindre la LED */
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, RESET);
}

}
}
```

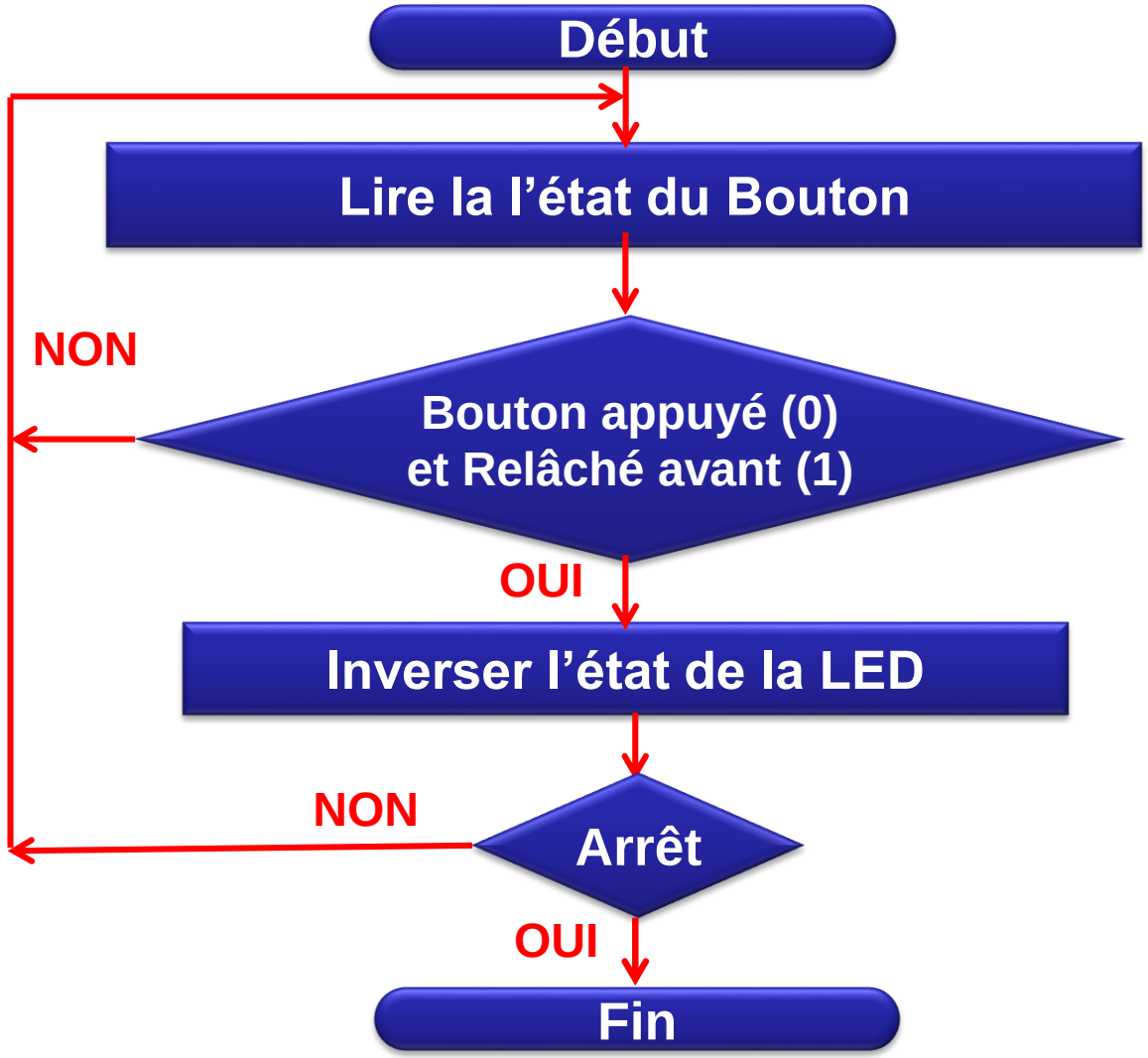
EXERCICE D'APPLICATION

- Dans l'application précédente, il s'agissait d'allumer la LED une fois le bouton appuyé et de l'éteindre une fois relâché!

- L'objectif maintenant consiste à l'allumer une fois le bouton appuyé, et reste allumée même lorsqu'on relâche le bouton. On l'éteindra lorsqu'on appuie de nouveau sur le bouton.

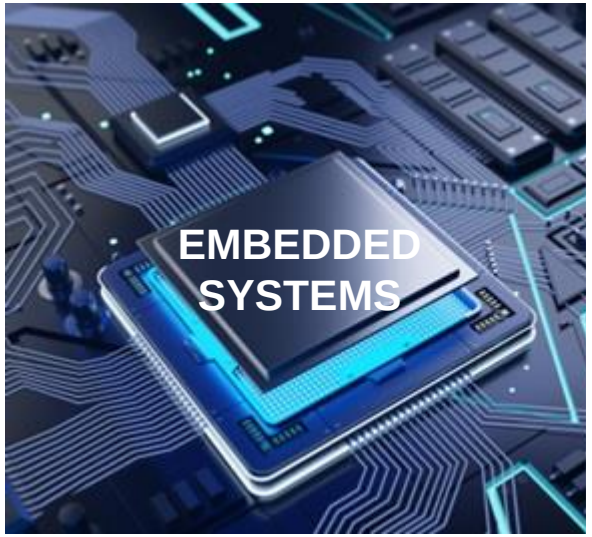
- Elaborer le code!

Solution Exercice: L'Organigramme



Solution Exercice: Le Code

```
uint8_t Etat_bouton = 1;
uint8_t Etat_precedent = 1; // Bounton n'est pas précédemment appuyé
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
// Lire l'état du bouton
Etat_bouton = HAL_GPIO_ReadPin(B1_Button_GPIO_Port, B1_Button_Pin);
// Si le bouton est appuyé et n'est pas précédemment appuyé, alors changer l'état de la LED
if(Etat_bouton == 0 && Etat_precedent == 1)
{
HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin); // Toggle LED state
}
// Sauvegarder l'état actuel du bouton pour l'itération suivante
Etat_precedent = Etat_bouton;
// Ajoutez un petit délai pour anti-rebondir le bouton (facultatif)
HAL_Delay(100);
}
/* USER CODE END 3 */
}
```



FIN !
Questions ?