



## Programmation de cartes « maison » (cas particulier de l'ObcP CROC)

### Programmation via le DFU : STM32CubeProgrammer

Cette partie est dédiée à la programmation de votre objet connecté personnalisé. La programmation passe par le DFU :

[https://www.st.com/resource/en/application\\_note/cd00264379-usb-dfu-protocol-used-in-the-stm32-bootloader-stmicroelectronics.pdf](https://www.st.com/resource/en/application_note/cd00264379-usb-dfu-protocol-used-in-the-stm32-bootloader-stmicroelectronics.pdf)

Un logiciel fourni par ST Microelectronics est disponible pour la programmation des microcontrôleurs de la famille des STM. Ce programmeur peut effectuer le téléversement du programme aussi bien par debugger ou par DFU. Il est disponible dans le répertoire électronique de l'image ENSMM, dans la partie microcontrôleurs de moodle ou encore téléchargeable à cette adresse :

<https://www.st.com/en/development-tools/stm32cubeprog.html>

### Protocole de programmation de l'ObcP

**Étape 1 :** Depuis CUBEIDE, lancer la compilation et récupérer le fichier .bin (par exemple mon\_programme.bin dans le répertoire Debug ou Release du projet)

**Étape 2 :** Lancer le programme STM32CubeProgrammer (voir Figure 183)

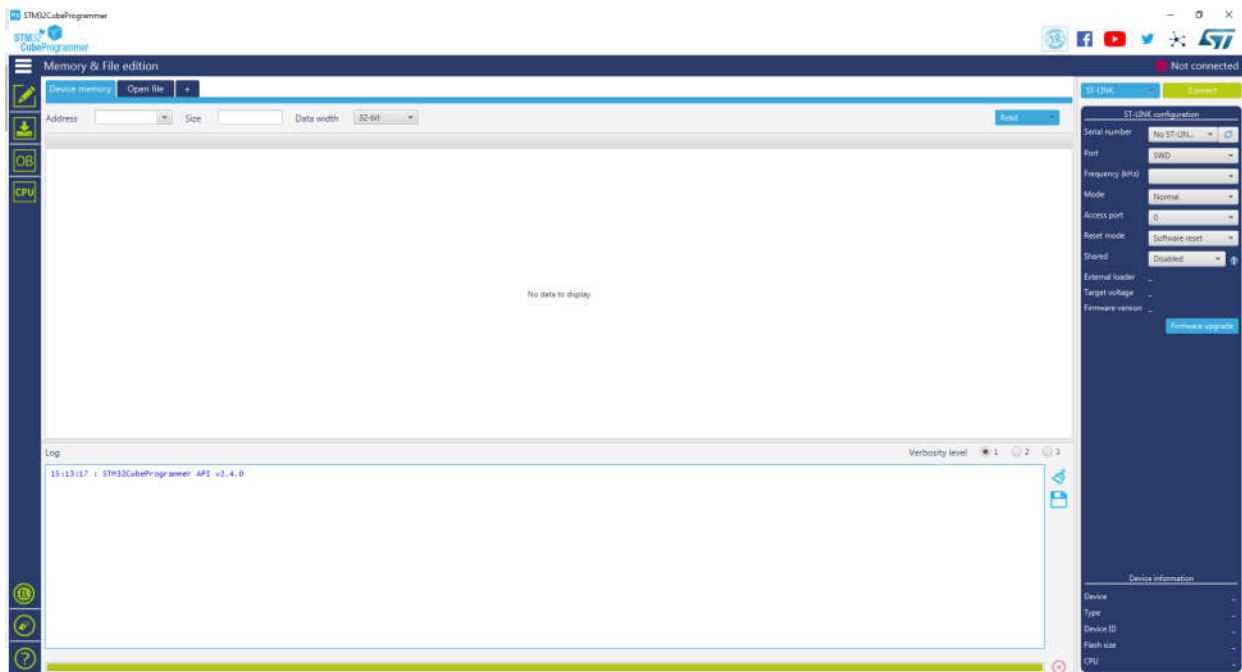


Figure 183 : Environnement STM32CubeProgrammer.



Dans la partie en haut à droite choisir USB, Figure 184 :



Figure 184 : Sélection de programmation via USB dans STM32CubeProgrammer.

**Étape 3 :** Connecter l'ObcP via son câble USB. Mettre l'ObcP en mode **Prog** (interrupteur SW4) mettre la carte sur **On** (interrupteur SW2) et appuyer sur le bouton poussoir **SW3 reset** (Figure 185).



Figure 185 : Configuration de l'ObcP pour la programmation avec STM32CubeProgrammer.

Cliquer sur le bouton refresh, Figure 186 :

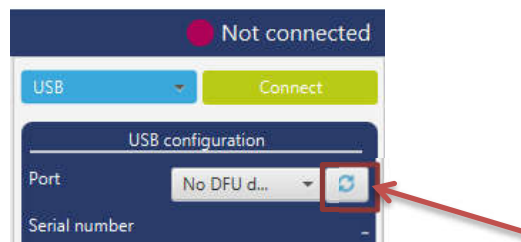


Figure 186 : Mise à jour de la liste des ports USB disponibles dans STM32CubeProgrammer.

Un fois la carte détectée en mode DFU, elle apparaît dans **Port et Serial Number**, cliquer alors sur **Connect**, Figure 187 :

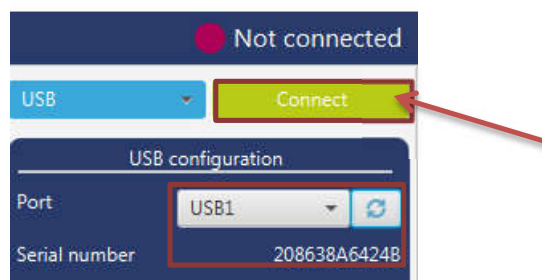


Figure 187 : Connexion au port USB dans STM32CubeProgrammer.



Aller ensuite dans le menu **erasing & programming**, Figure 188 :

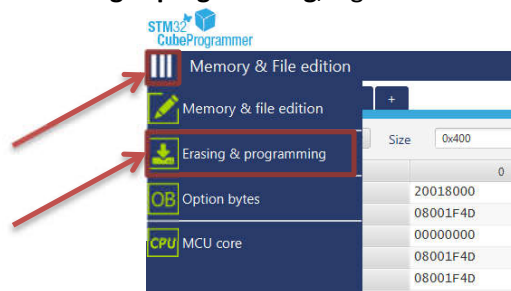


Figure 188 : Menu Erasin & Prprogramming dans STM32CubeProgrammer.

Aller ensuite dans **Browse** pour choisir le fichier **Bin** à programmer, puis cliquer sur **Start Programming** pour lancer le téléversement du programme en mémoire flash, Figure 189 :

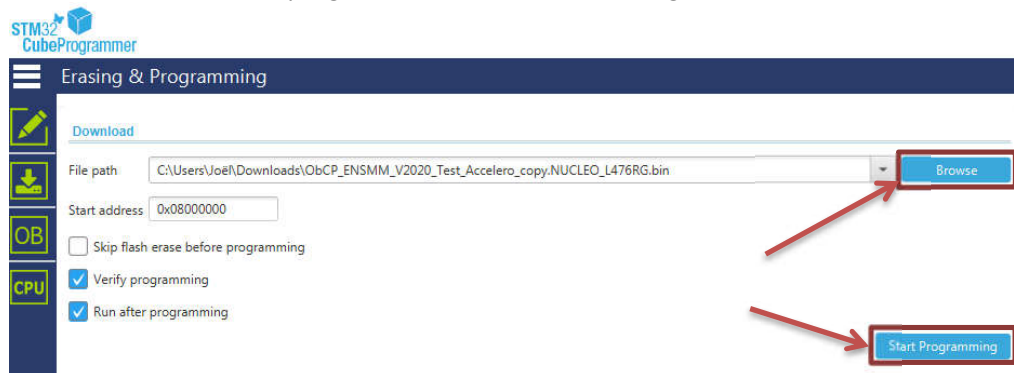


Figure 189 : Sélection du fichier et téléversement dans STM32CubeProgrammer.

**Étape 4 :** Mettre l'ObcP en mode **Run** (SW4) et appuyer sur le bouton-poussoir **SW3 reset**, Figure 190. Le programme doit se lancer... le port USB peut être utilisé pour une communication (**Virtual Com port** par exemple).

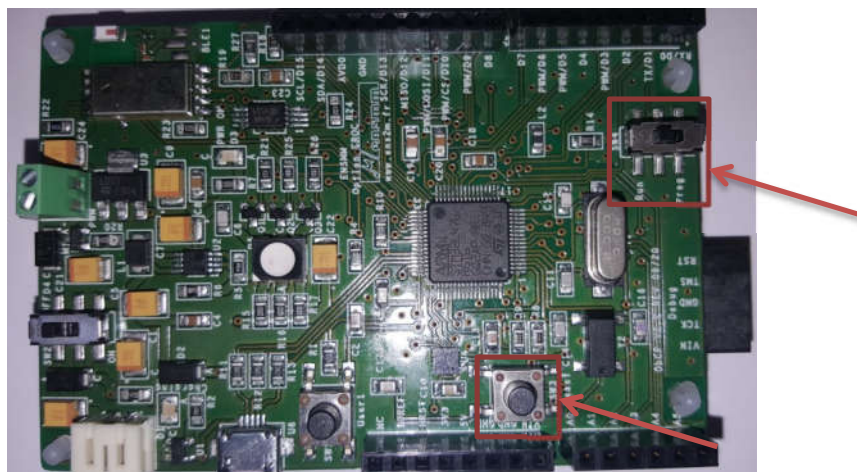


Figure 190 : Configuration de l'ObcP pour lancer le programme téléversé avec STM32CubeProgrammer.



## Configuration de CubeIDE pour utiliser CubeProgrammer

Il est possible de configurer STM32CubeIDE pour téléverser directement le programme en mode DFU dans la cible sans passer par le programmeur debugger ou CubeProgrammer comme présenté sur l'AN1801<sup>50</sup>. Pour cela, il faut lancer le téléversement par ligne de commande avec CLI, les commandes sont décrites dans le user manual UM2237<sup>51</sup>, la Figure 191 montre comment paramétrer CubeIDE.

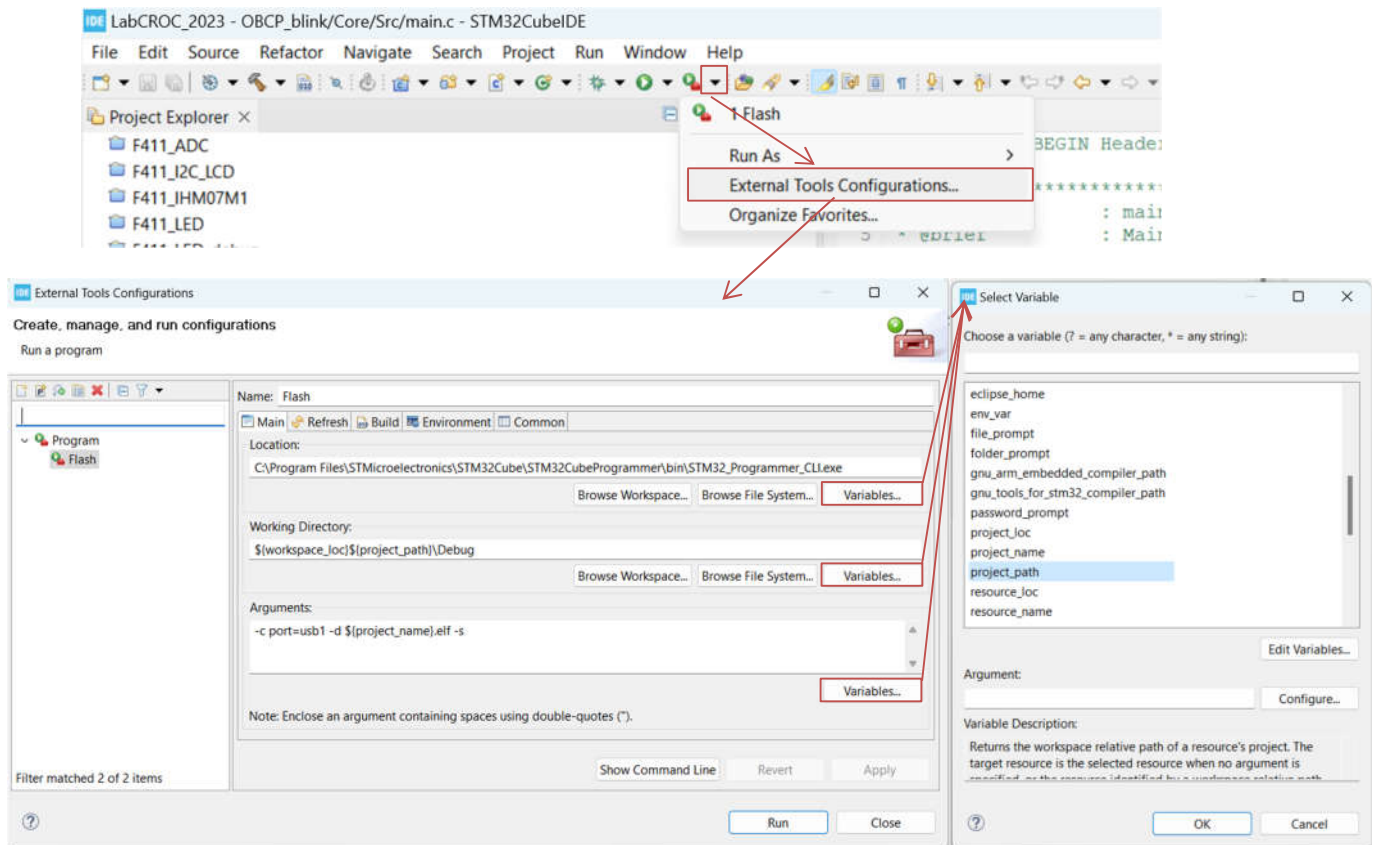


Figure 191 : Configuration de STM32CubeIDE pour la programmation direct de l'ObCP avec STM32CubeProgrammerCLI.

**Location** pointe l'emplacement de l'exécutable CLI (dépend de l'emplacement de l'installation de CubeProgrammer) :

**C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32\_Programmer\_CLI.exe**

**Working directory** définit l'emplacement du fichier de programmation dans le projet (\Debug ou \Release) :

**\${workspace\_loc}/\${project\_path}\Debug**


<sup>50</sup> [https://www.st.com/resource/en/application\\_note/an1801\\_stm32cubeprogrammer\\_in\\_trustudio-installing-stm32cubeprogrammer-in-trustudio-stmicroelectronics.pdf](https://www.st.com/resource/en/application_note/an1801_stm32cubeprogrammer_in_trustudio-installing-stm32cubeprogrammer-in-trustudio-stmicroelectronics.pdf)

<sup>51</sup> [https://www.st.com/resource/en/user\\_manual/um2237-stm32cubeprogrammer-software-description-stmicroelectronics.pdf](https://www.st.com/resource/en/user_manual/um2237-stm32cubeprogrammer-software-description-stmicroelectronics.pdf)



**Arguments** comme décrit dans le UM2237, `usbx` dépend de votre installation, à tester de 0 à n :

**- c port=usb1 -d \${project\_name}.elf -s**

Pour lancer la programmation, positionner l'interrupteur en mode « **Prog** », appuyer sur reset puis cliquer sur  pour lancer le téléversement. Une fois le téléversement terminé, positionner l'interrupteur sur « **Run** » puis appuyer sur reset pour lancer le programme.





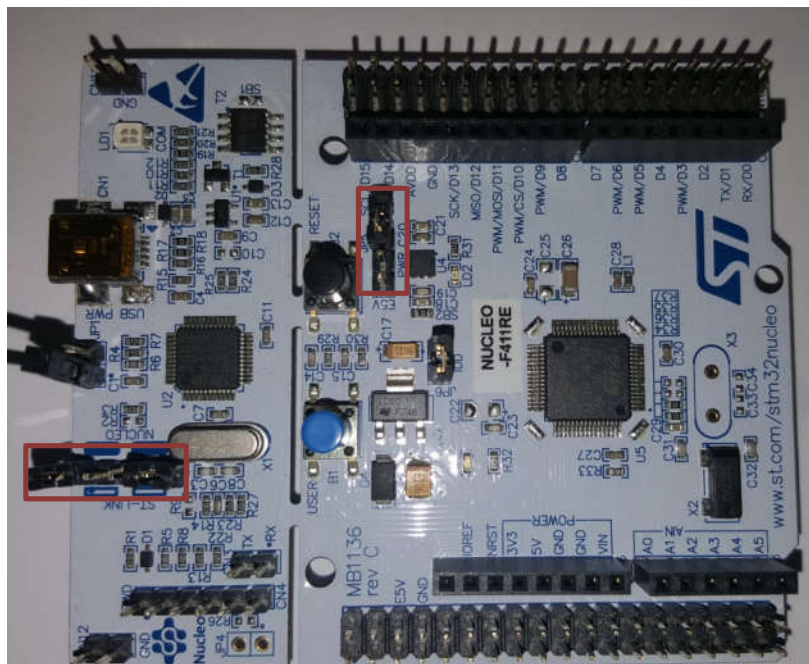
## Programmation via un debugger

Plusieurs solutions existent pour programmer une carte « maison », le programmeur/debugger existe sous différentes formes commerciales, officielles ou non sous la dénomination ST-LINK :



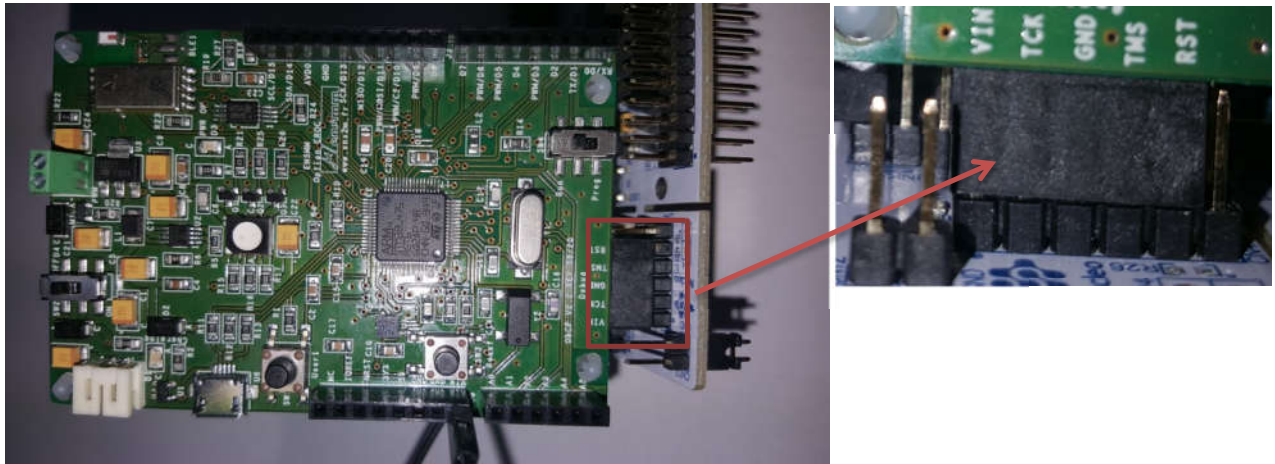
Il est tout à fait possible d'utiliser le debugger (ST-LINK) d'une carte Nucleo pour programmer l'ObcP, la programmation se fait alors par un simple « glisser-déplacer ».

**Étape 1 :** Configurer la carte Nucleo ; les cavaliers ST-LINK et JP5 doivent être désactivés :

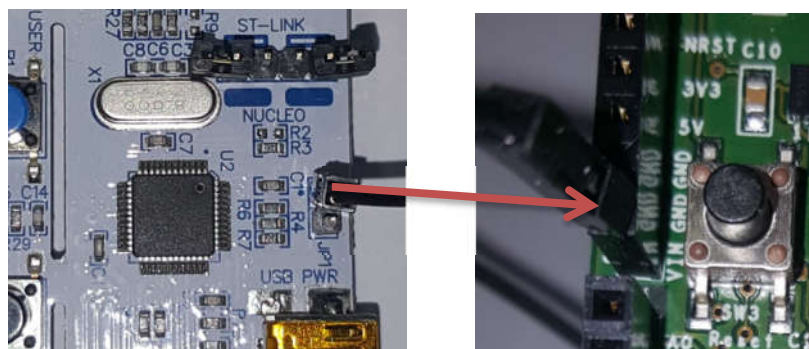




**Étape 2 :** Connecter l'ObCP sur le connecteur SWD de la carte Nucleo via son port de debug :



**Étape 3 :** Connecter l'alimentation de la carte Nucleo (Pin 1 de JP1) sur l'entrée Vin de l'ObCP :

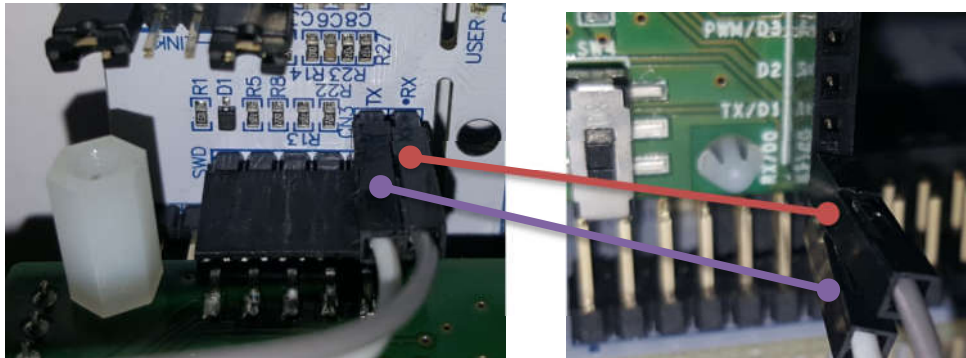


**Étape 4 :** Mettre à jour le firmware du debugger avec STM32 Cube programmer par exemple :





La carte ObCP apparaît maintenant comme un lecteur USB et peut désormais être programmée par « glisser-déplacer ». Petit plus, il est possible de connecter le TX RX pour se servir du terminal comme sur la carte Nucleo en connectant RX de l'ObCP sur TX de la carte Nucleo et TX de l'ObCP sur RX de la carte Nucleo :







## Brochage et définition de l'ObCP

Les ports d'entrées/sorties sont définis comme sur la carte Nucléo en ce qui concerne le port compatible Arduino. Les autres ports sont définis comme indiqué dans le tableau suivant :

Fonctions	Brochage
<b>Entrées / Sorties</b>	
LED verte	PC_8
LED rouge	PC_6
LED bleue	PC_9
Bornier sortie PWM	PB_1
Bouton-poussoir	PC_13
<b>Accéléromètre</b>	
MOSI	PC_12
MISO	PC_11
CS	PC_5
SCLK	PC_10
<b>Bluetooth low energy</b>	
MOSI	PB_15
MISO	PB_14
nCS	PB_2
RESET	PC_2
IRQ	PB_12
SCK	PB_13



## Programmation avec l'IDE ADUINO

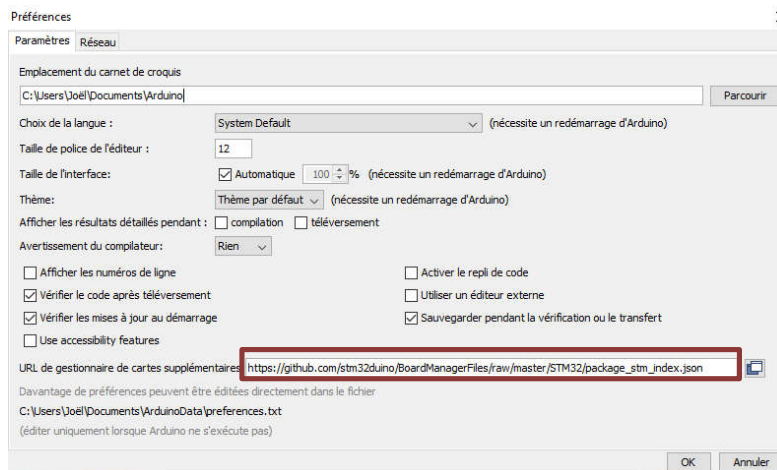
Il est tout à fait possible de programmer l'ObCP ENSMM avec l'environnement Arduino en suivant la procédure décrite ci-après. (<https://github.com/stm32duino/wiki/wiki/Getting-Started>)

Étape 1 : Installer STM32 Cube Programmer (voir chapitres précédents)

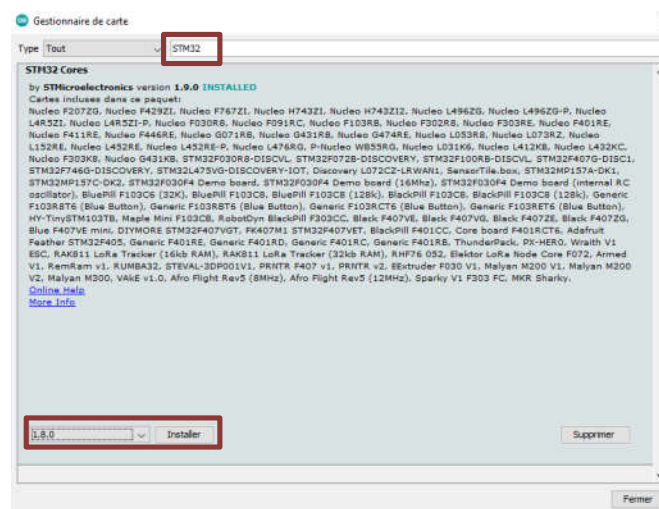
Étape 2 : Installer l'IDE ARDUINO : [www.arduino.cc](http://www.arduino.cc)

Étape 3 : Lancer l'IDE Arduino, aller dans fichier, préférences, puis rentrer dans URL de gestionnaire de carte supplémentaire l'adresse suivante :

[https://github.com/stm32duino/BoardManagerFiles/raw/master/STM32/package\\_stm\\_index.json](https://github.com/stm32duino/BoardManagerFiles/raw/master/STM32/package_stm_index.json)

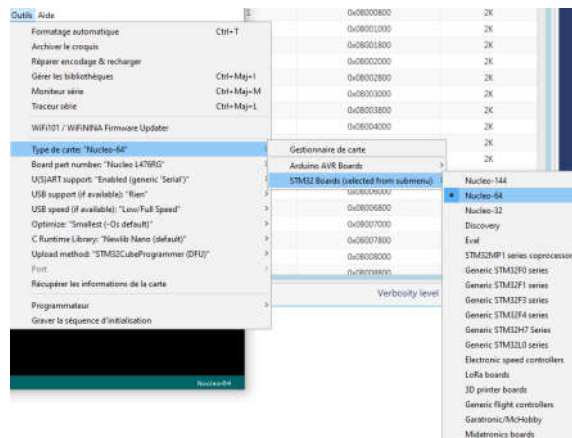


Étape 4 : Aller dans le menu Outils → Type de carte → Gestionnaire de carte puis rechercher STM32 et installer le package correspondant :

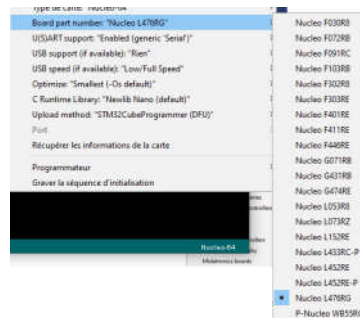




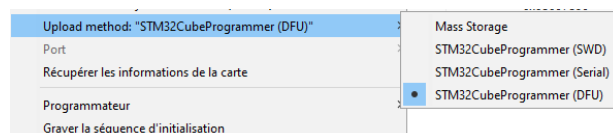
Étape 5 : Aller dans le menu Outils → Type de carte → STM32 Boards → Nucleo 64



Étape 6 : Aller dans le menu Outils → Board part number → Nucleo L476RG



Étape 7 : Aller dans le menu Outils → Upload Method → STM32CubeProgrammer(DFU)



Étape 8 : Connecter l'ObcP via son câble USB. Mettre l'ObcP en mode **Prog** (interrupteur SW4) mettre la carte sur **On** (interrupteur SW2) et appuyer sur le bouton poussoir **SW3 reset**.





Étape 9 : L'ObCP est prêt à être programmé avec l'IDE Arduino, voici un petit programme test pour faire clignoter la LED verte :

Un exemple de sketch arduino :

```
void setup() {  
    // initialize digital pin LED_BUILTIN as an output.  
    pinMode(PC8, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
    digitalWrite(PC8, HIGH);    // turn the LED on (HIGH is the voltage level)  
    delay(1000);                // wait for a second  
    digitalWrite(PC8, LOW);     // turn the LED off by making the voltage LOW  
    delay(1000);                // wait for a second  
}
```