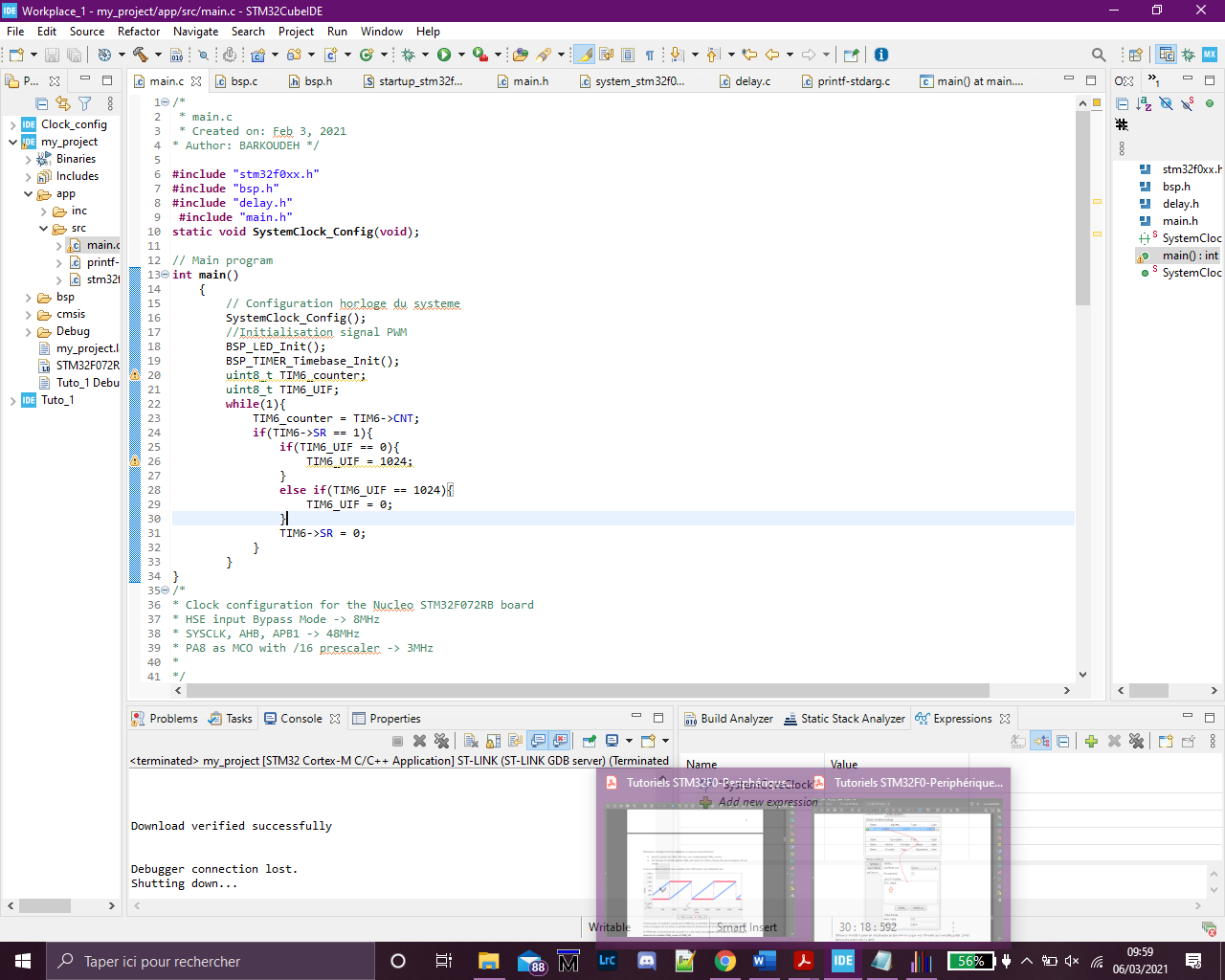
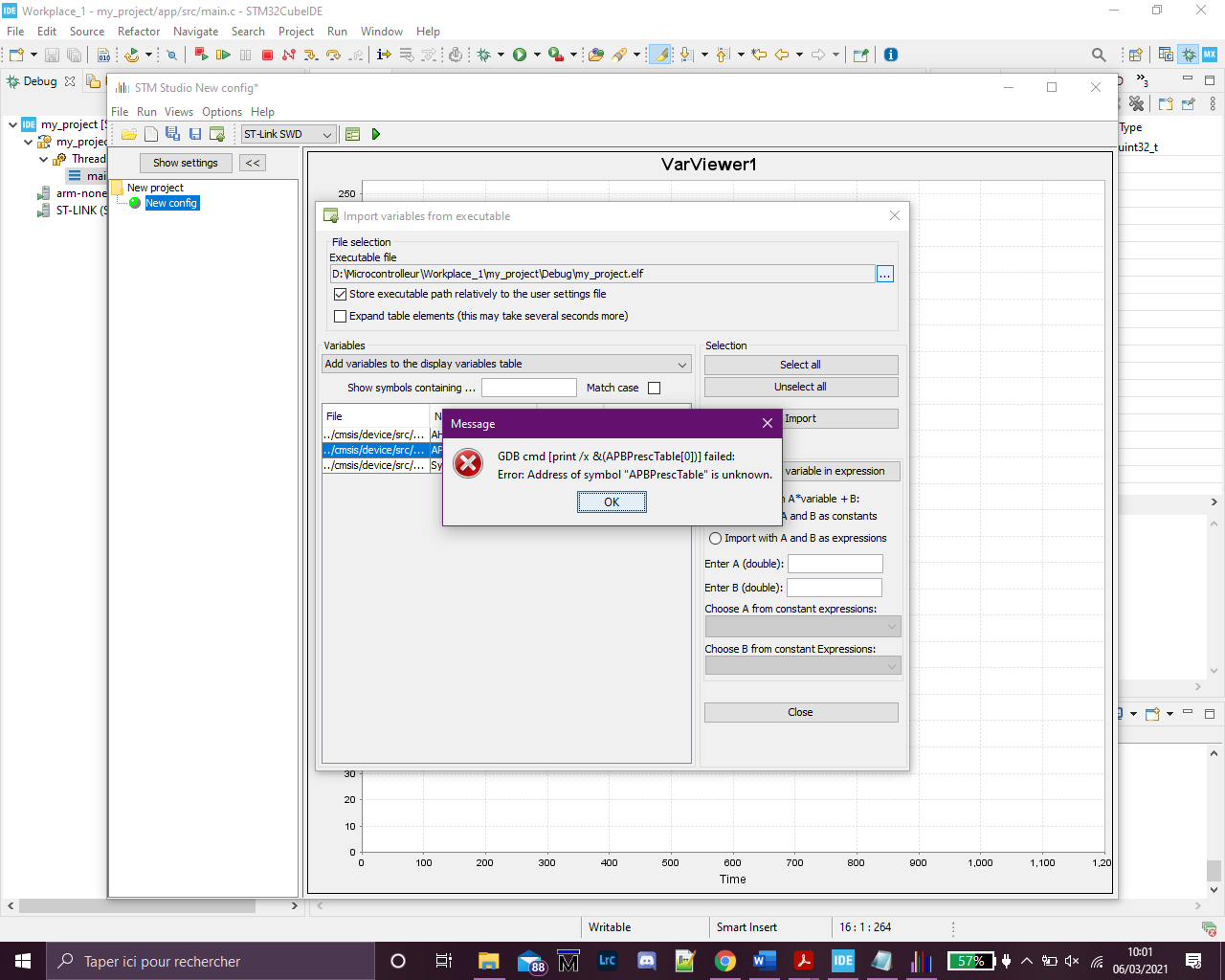
**DM tutoriel Timers**

**EI2I3-IIA Julian BARKOUDEH**

Note 1 : Tous les codes de BSP.c relatifs à ce tutoriel se trouvent dans la partie Timer, à partir de la ligne 340.

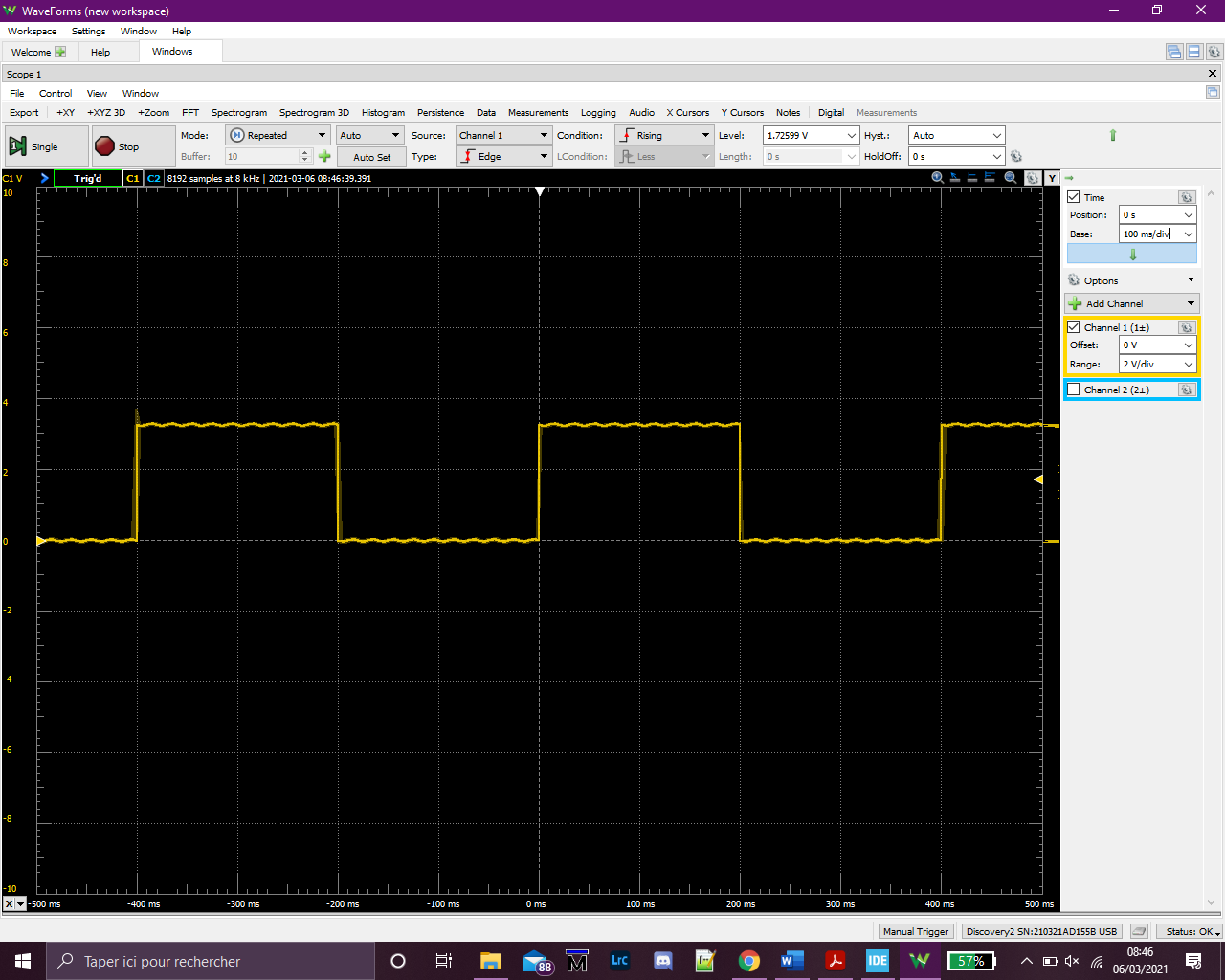
Note 2 : La fonction de printf est appelée j\_printf dans mes codes. Or dans le rendu de chaque main.c je les avait appelés mon\_printf

**1.1 Base de temps simple : Mutineries** 

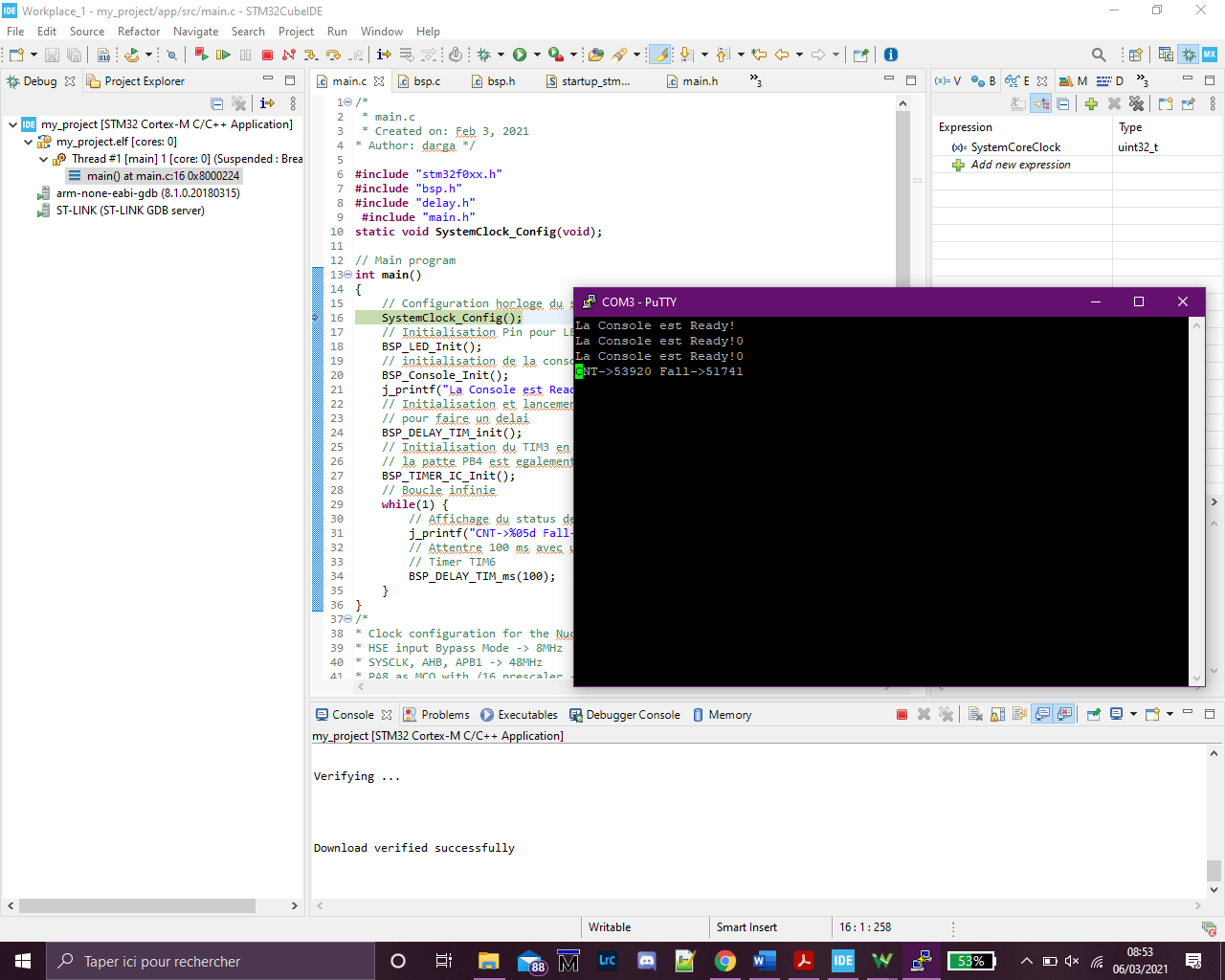


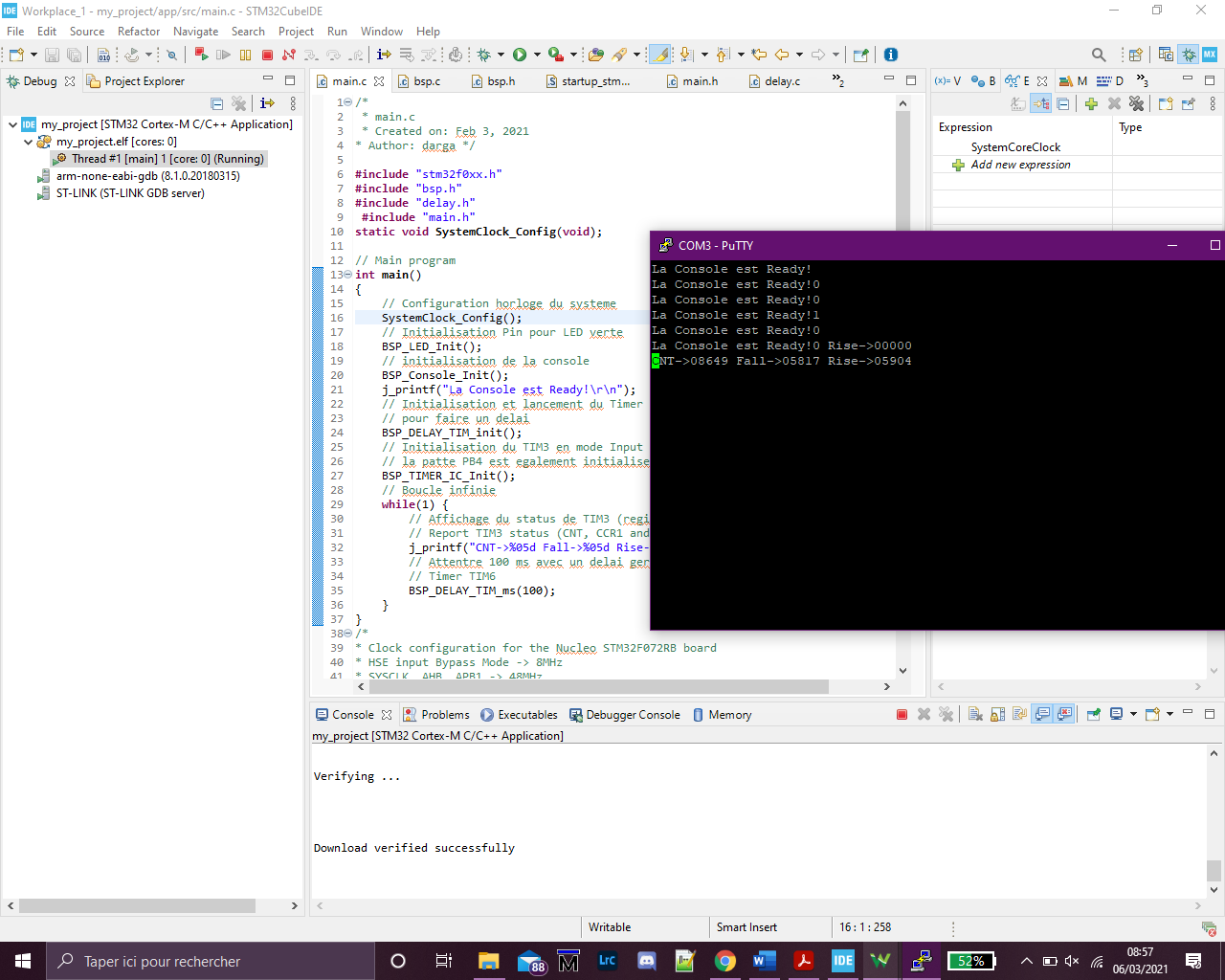
Le code ci-dessus permet de générer le signal en question. Or l’erreur suivant empêche de visualiser les variables sur STMstudio.

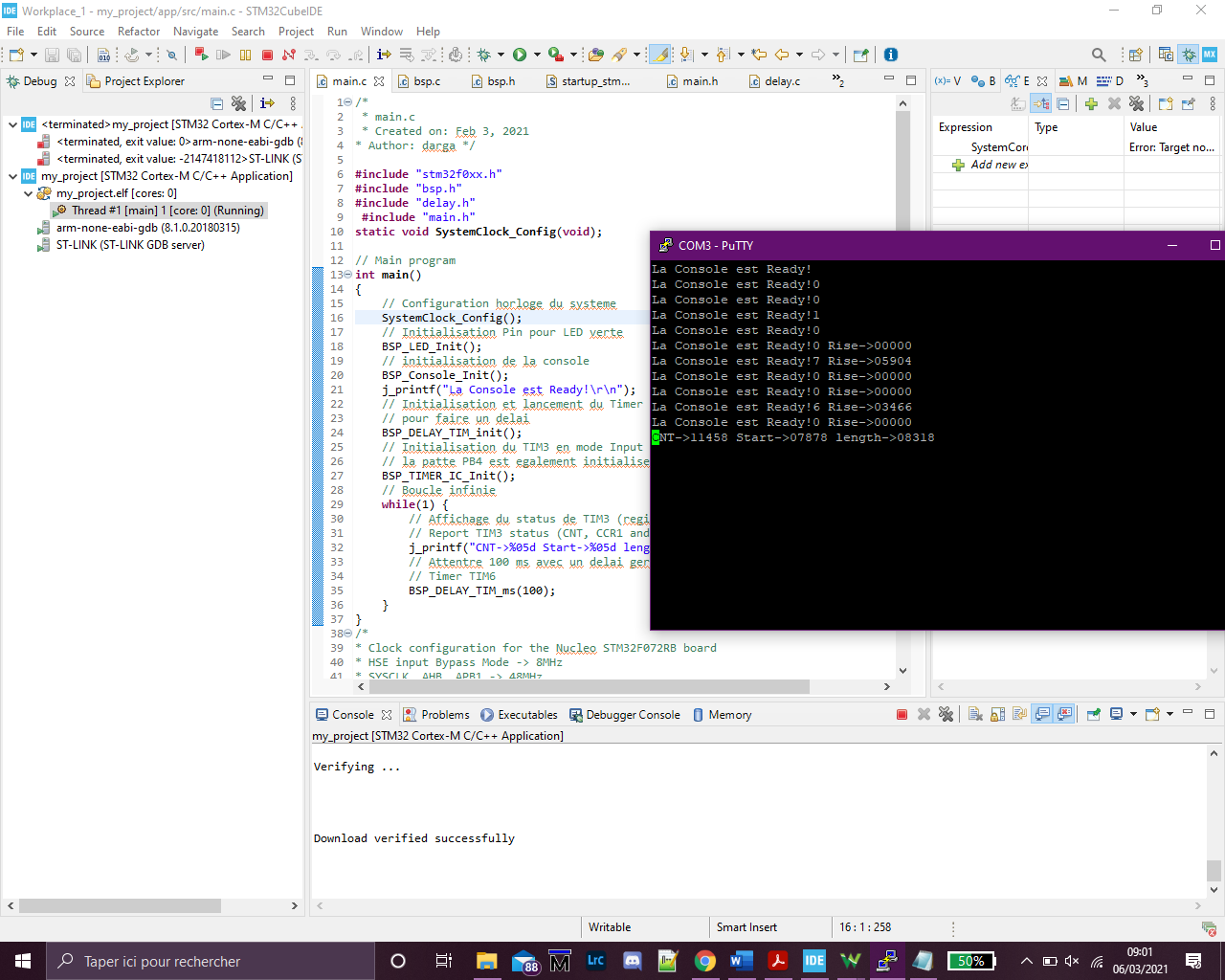
**1.2) Utilisation du Timer pour gérer une fonction de temporisation robuste**

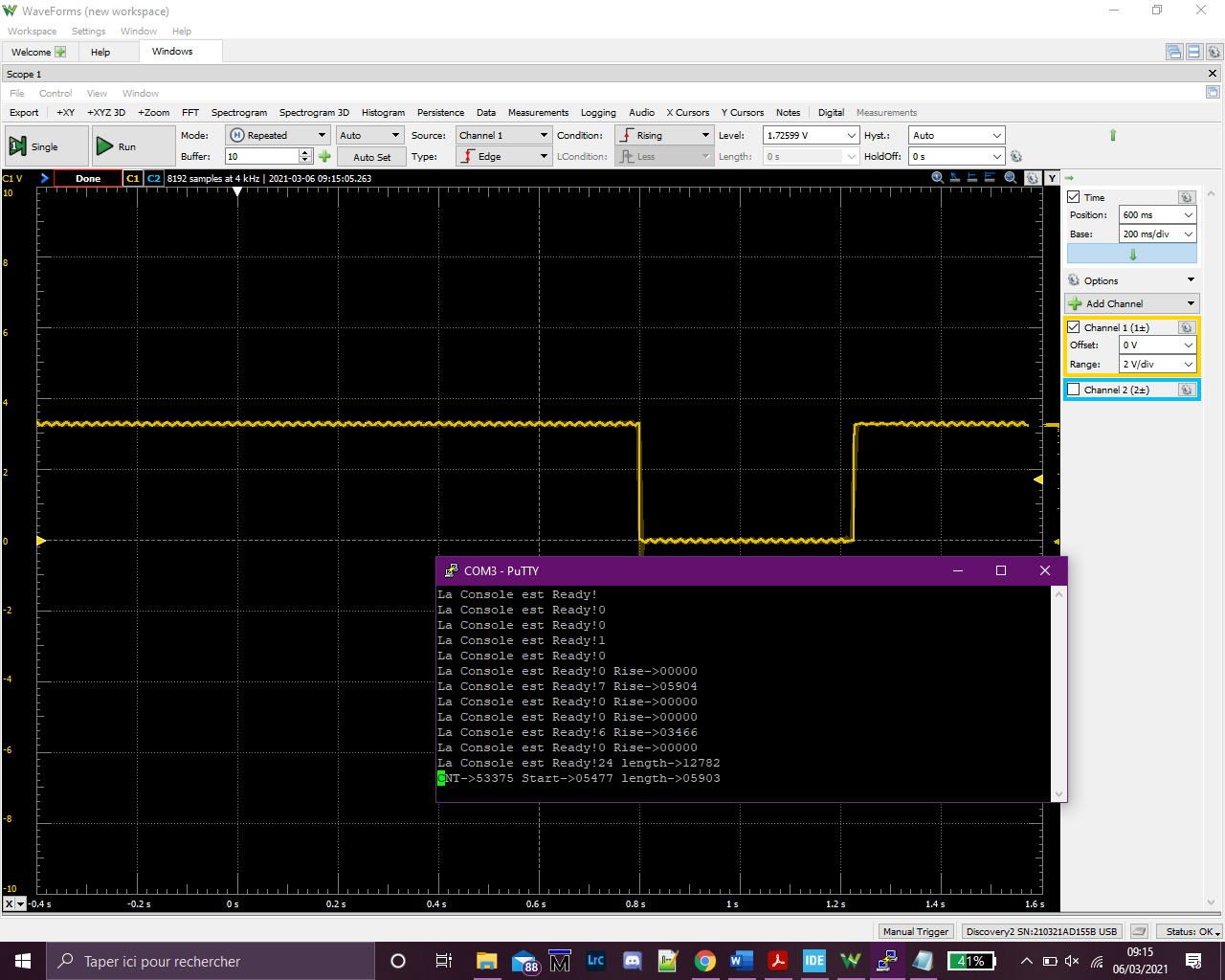


On branche l’oscilloscope sur le pin PA5 lié au LED afin de visualiser le basculement avec les fonctions delay introduites.

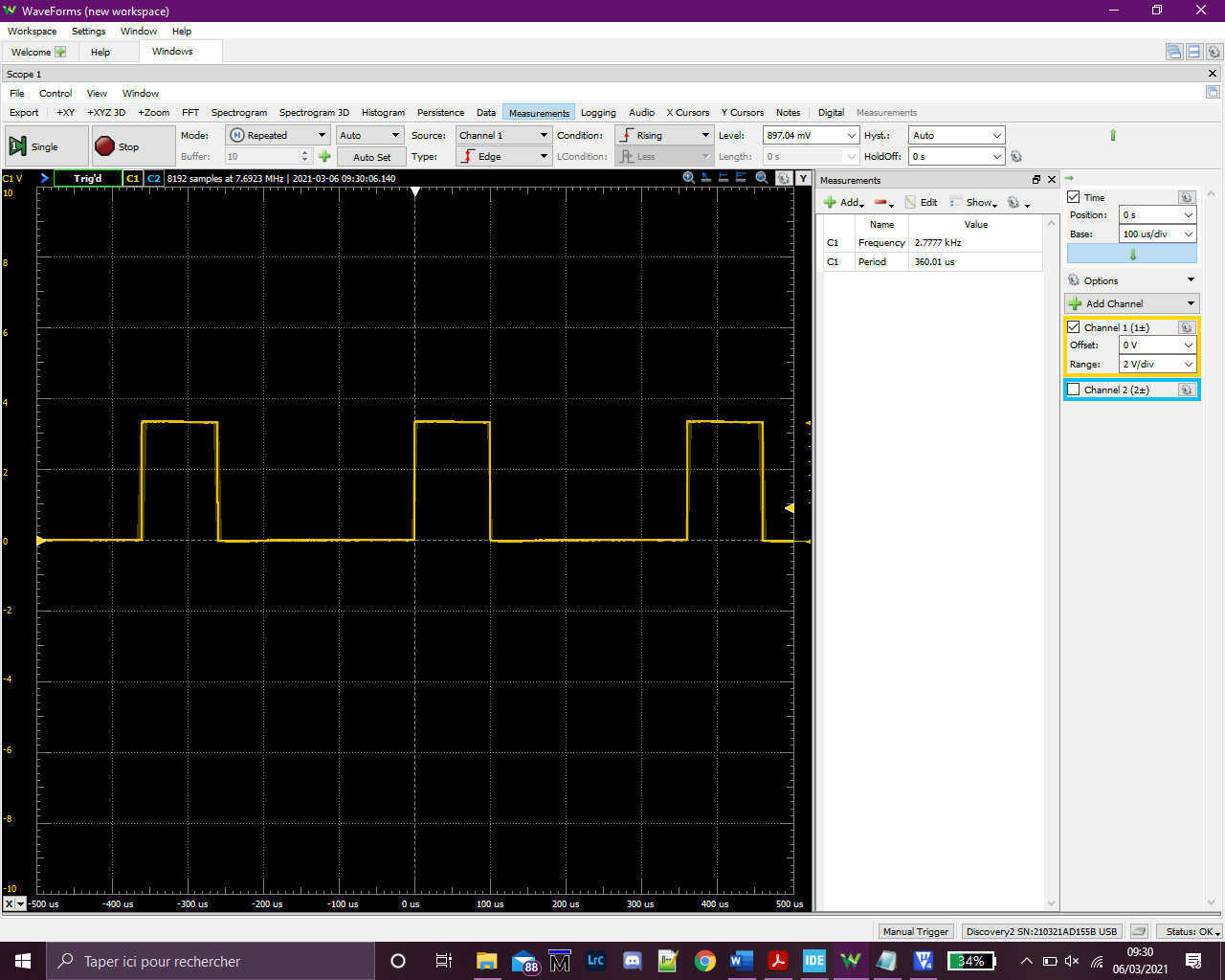
**2.1 Mode Input Capture : Single edge – détection d’un seul front** 

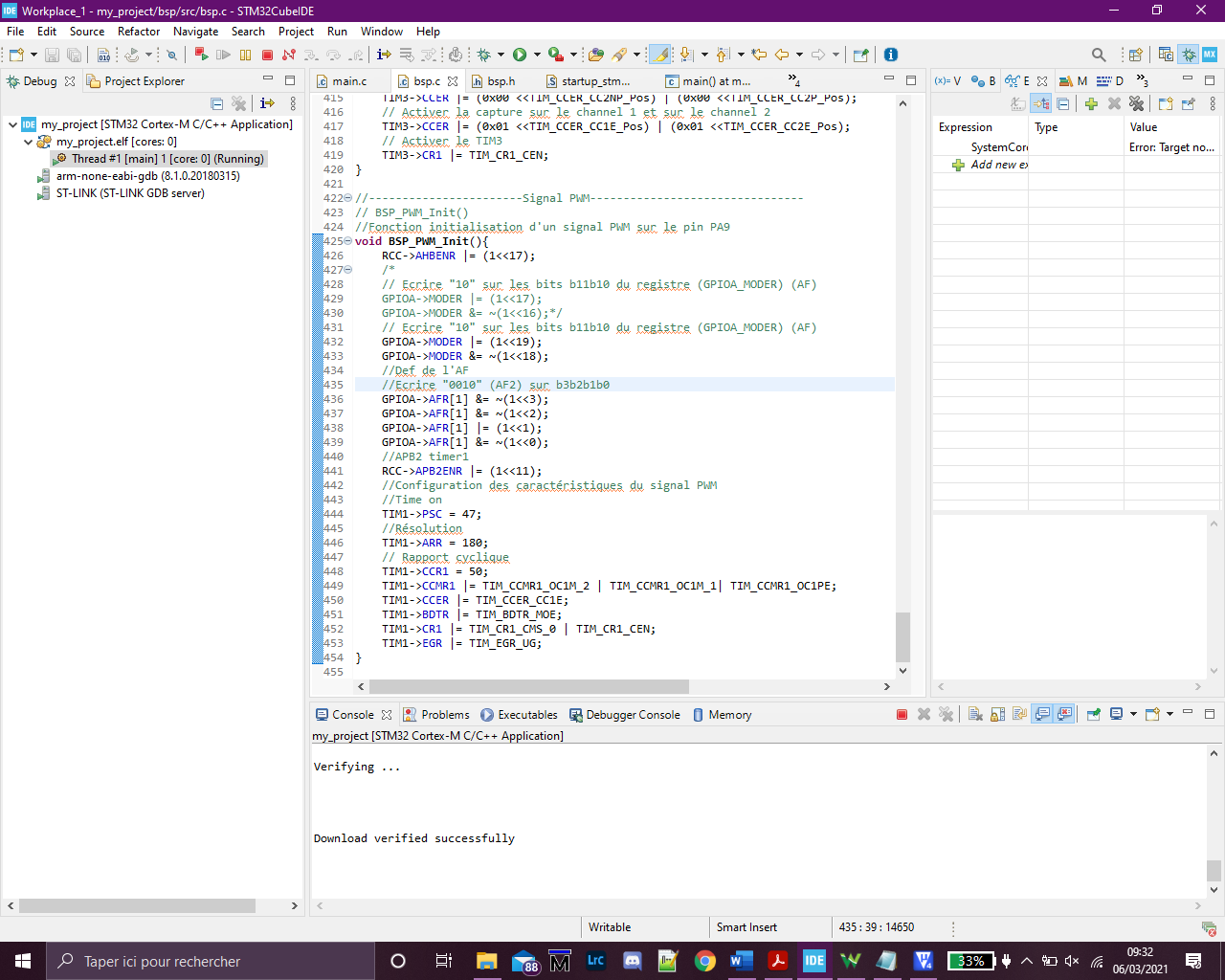
**2.2 Mode Input Capture : Dual edges captures – détection de deux fronts** 

**2.3 Mode Input Capture : Dual edges captures – détection de deux fronts avec reset automatique** 



Sur l’affichage de printf on mesure 0.59 s, ceci est cohérent avec la lecture graphique du temps du signal affiché sur l’oscilloscope. On mesure t= 1.3-0.8 = 0.5 s**.**

**3.1 Qu'est-ce qu'un signal PWM**



En se basant sur la manuelle de la carte, le code ci-dessus présent dans BSP.c permet d’initialiser un signal PWM sur la pate PA8.

Tout d’abord en met la pate PA8 en mode Highspeed, ensuite on active le AF2 sur la pate PA8. On génère un signal PWM de rapport cyclique de 50%, et d’une fréquence de 2.7 KHZ. La configuration de la fréquence est faite avec le registre PSC, on le règle à 47 us qui définit la période du signal. On définie la résolution par le registre ARR, on le règle à 180 pour avoir 180 valeurs du rapport cyclique.