ButtonManager : Rapport

Favre Lenny
PTR
I5 - 2020

Table des matières

Introduction

Dans ce laboratoire, nous avons dû réaliser quelques classes dans le but de détecter les changements d'états des boutons de la carte d'extension du kit STM32F7 discovery de l'école. Les classes créées sont décrites dans ce court rapport.

Le STM32 a été configuré avec CubeMX pour que les boutons soient des interruptions externes à chaque rising/falling edge avec pull-up.

Diagrammes UML

Diagramme de classe

Voici le diagramme de classe simplifié et le diagramme de classe complet.

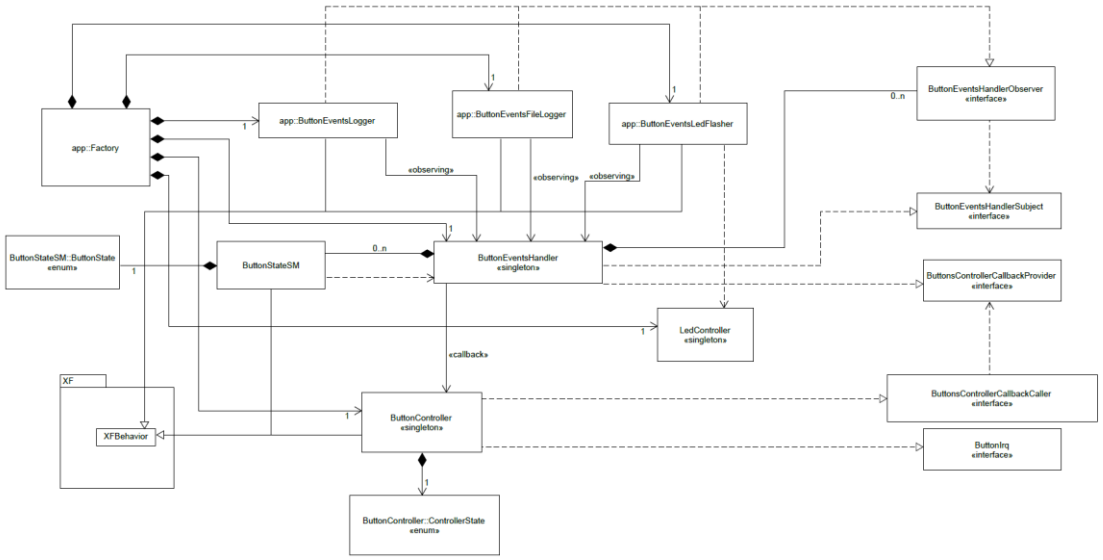

Figure 1 : Diagramme de classe simplifié

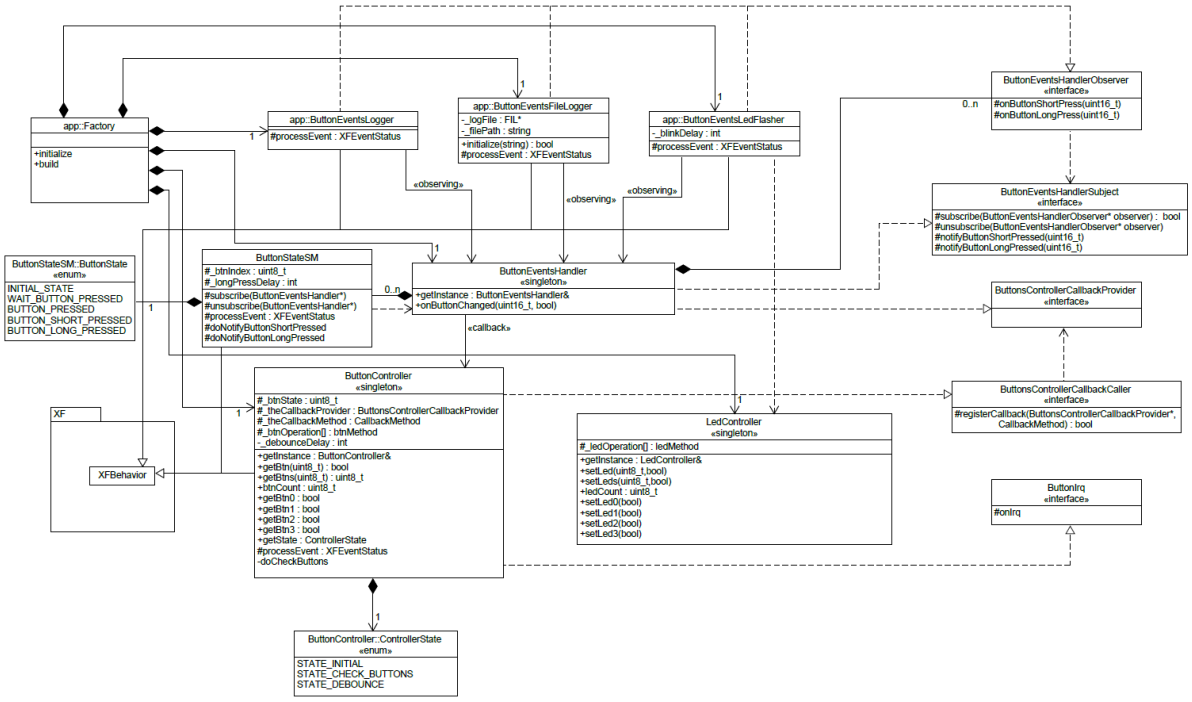

Figure 2 : Diagramme de classe complet

Diagrammes d'état

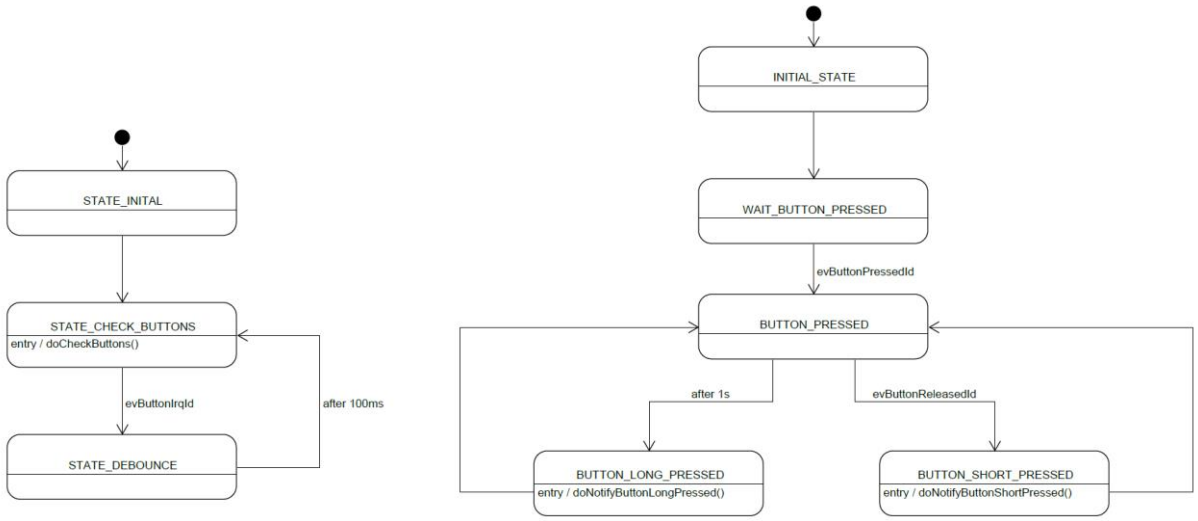

Figure 3 : Diagramme d'état du ButtonController(gauche) et de ButtonStateSM(droite)

Diagramme de séquence

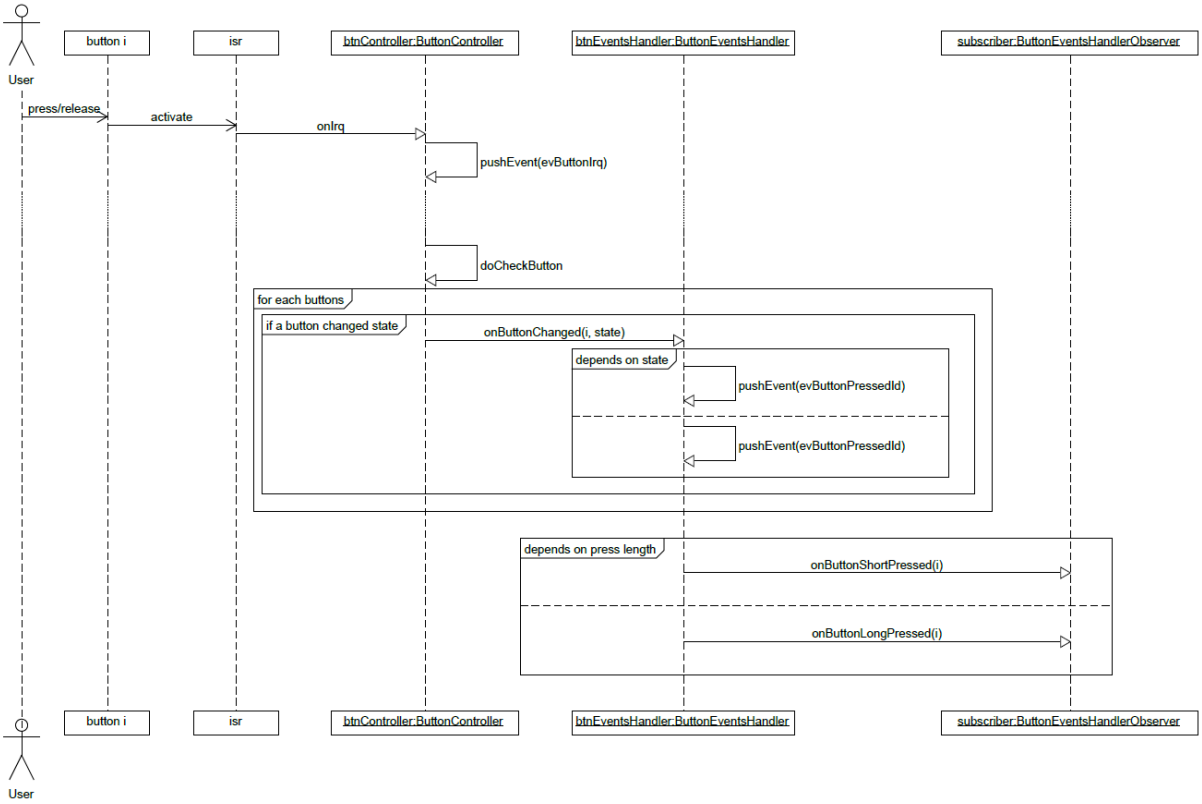

Figure 4 : Diagramme de séquence du projet complet

Explications

Les classes ButtonController et ButtonEventsHandler sont liées par le pattern callback. La classe ButtonEventsHandler est le sujet du pattern subject-observer. Ses observeurs sont les classes ButtonEventsLogger, ButtonEventsFileLogger et ButtonEventsLedFlasher. J'ai décidé de ne pas utiliser de machine d'état pour le ButtonEventsLedFlasher pour que je puisse faire flasher plusieurs LEDs en même temps sans avoir besoin de définir une machine d'état par LED. Le reste des diagrammes est assez simple et la compréhension de ceux-ci est laissée comme exercice au lecteur.

Comme remarque personnel, je ne comprends pas bien pourquoi ces diagrammes ont dû être re-fait puisqu'ils sont basiquement un copier-coller des diagrammes fournit au début du projet.

Classes créées

Pour le projet ButtonManager de base, trois classes principales ont été écrites à partir des diagrammes d'états donnés par les professeurs.

ButtonsController

Le ButtonsController est une classe singleton qui contient une méthode onIrq qui est appelé à chaque interruption sur un bouton. Cette classe contient une machine d'état qui sert à debounce les boutons. Si l'état d'un bouton est changé, une callback méthode est appelée. Celle-ci est contenue dans la classe ButtonEventsHandler. Le code de la classe ButtonsController est basé sur celui de la classe LedController déjà fournit.

ButtonEventsHandler

Le ButtonEventsHandler est une classe singleton qui gère les événements relatifs aux boutons. Elle contient 4 machines d'états, une pour chaque bouton. Grâce à un pattern subject-observer, lorsqu'un bouton est pressé rapidement ou longuement, les observers sont notifiés des événements.

ButtonEventsLogger

Le ButtonEventsLogger est un observer qui est notifié à chaque fois qu'un bouton est pressé rapidement ou longuement. Il écrit ensuite sur l'UART par USB les changements qu'il observe. On peut ensuite relever les événements grâce au programme TraceLogger sur PC.

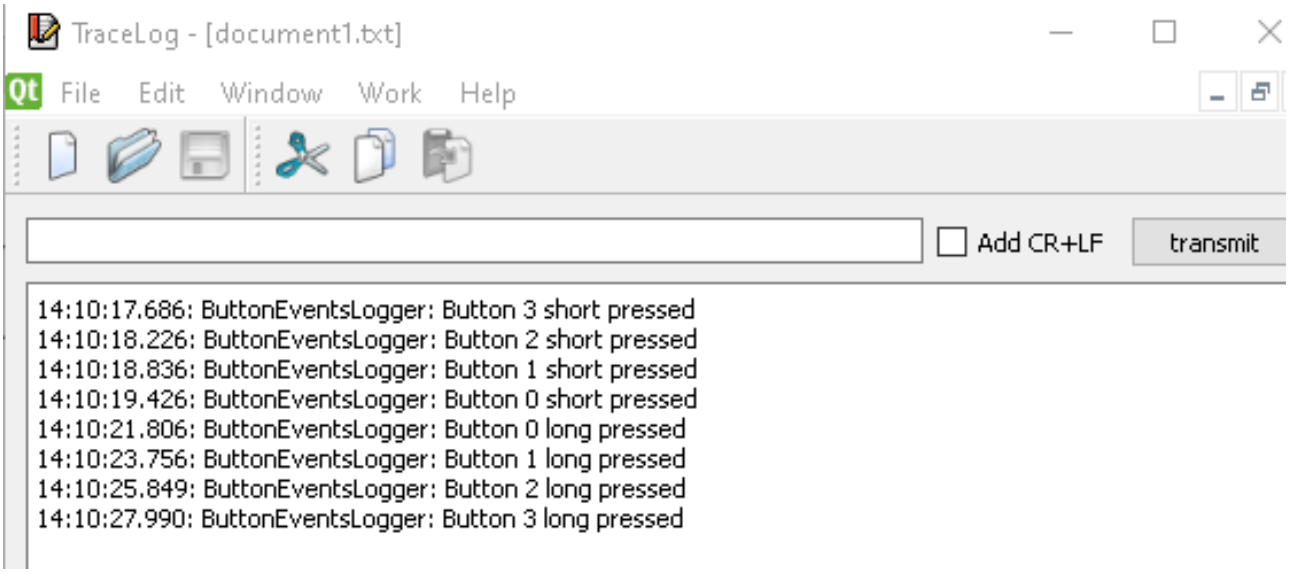

Figure 5: Logs du ButtonEventsLogger sur PC

Tâches optionnelles

Le projet ButtonManager avait aussi 2 tâches optionnelles qui sont décrites dans ce chapitre.

ButtonEventsLedFlasher

La classe ButtonEventsLedFlasher est un observer tout comme le ButtonEventsLogger. Lorsqu'il observe des événements relatifs aux boutons, il fait flasher la led correspondant au bouton pendant 200ms lors d'un appui court et la fait flasher 2 fois pendant 200ms lors d'un appui long.

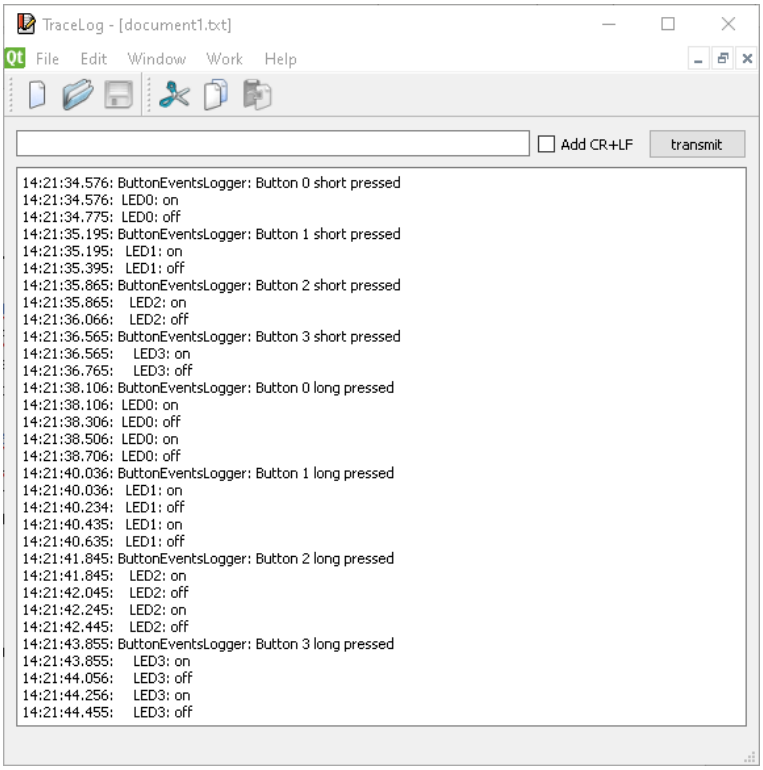

Figure 6: Logs de flash des LEDs du LedController

ButtonsEventsFileLogger

La classe ButtonEventsFileLogger est un observer tout comme le ButtonEventsLogger. Cependant lorsqu'il observe un événement, il ne l'envoie pas au PC mais l'écrit dans un fichier stocké sur une carte SD présente sur la carte STM32F discovery.

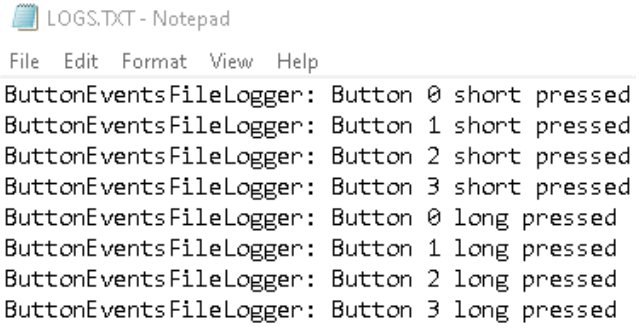

Figure 7: Logs du ButtonEventsFileLogger

Tests et résultats

Les tests ont été effectué à l'aide du debugger de System workbench ainsi qu'avec l'outil TraceLog. Lors de ce laboratoire, j'ai complété le projet ButtonManager de base comme vous pouvez le voir sur la figure 5. J'ai aussi effectué les deux tâches optionnelles du ButtonEventsLedFlasher et du ButtonEventsFileLogger comme vous pouvez le voir sur les figures 6 et 7.