

# Un vrai exemple d'un système temps réel

Medard Rieder

HES-SO Valais, 2022

# Indexe

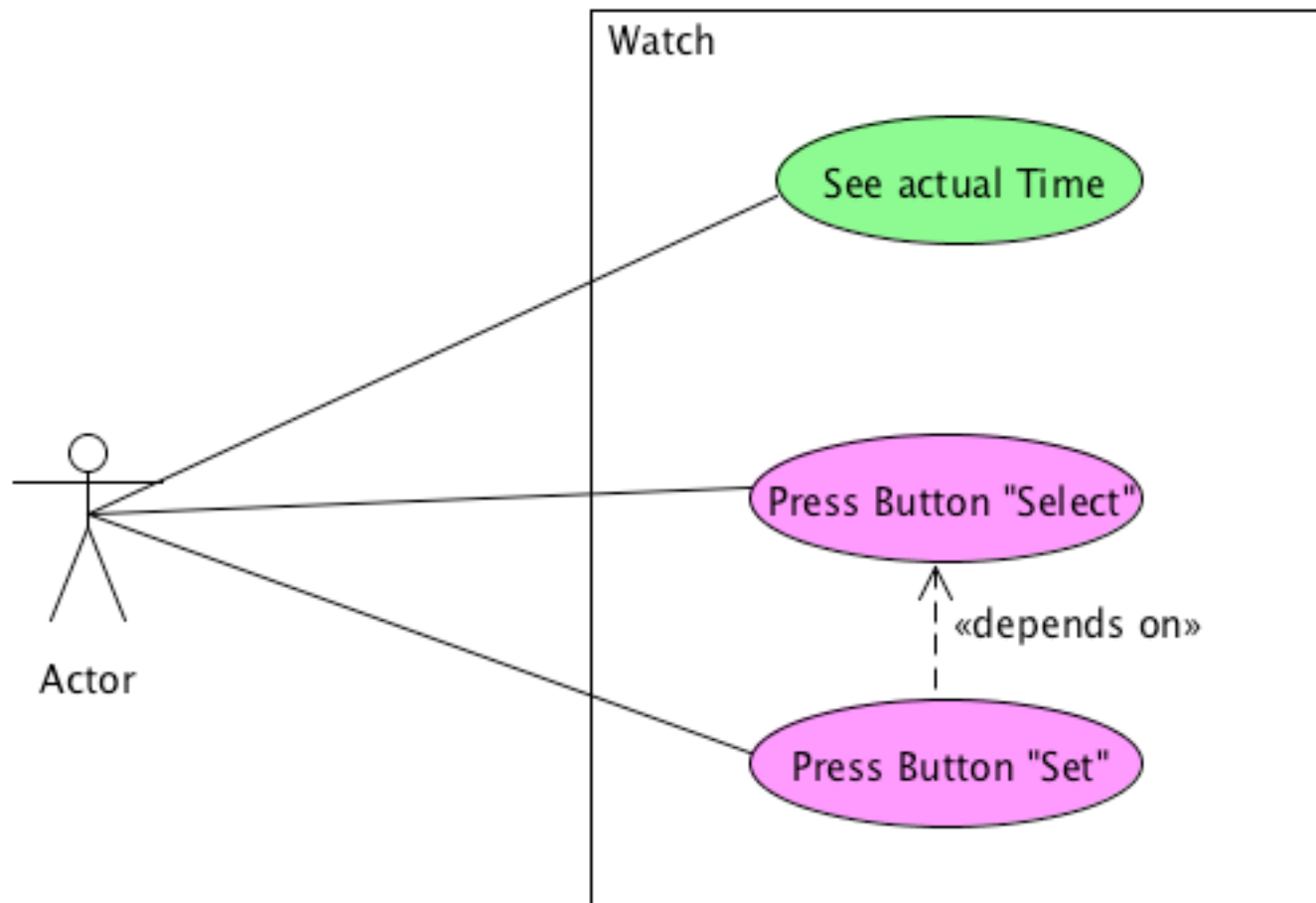
- Le problème
- L'analyse
- Transition analyse -> conception
- La conception
- Mise en oeuvre
- Vérification

## Une montre simple ...

Il faut réaliser une montre qui affiche l'heure et la minute et la seconde dans un écran LCD. Internement, les dixièmes des secondes sont prises en compte. Le battement doit être synchronisé avec les 50 Hz du secteur. Il y a deux boutons pour régler le temps: un bouton qui laisse choisir l'utilisateur parmi l'heure, la minute et la seconde et un deuxième qui sert à ajuster la valeur en question dans des incréments de un. Pour entrer en mode édition, il faut appuyer pendant 2 secondes le premier bouton, après l'édition des secondes on retourne en mode affichage.

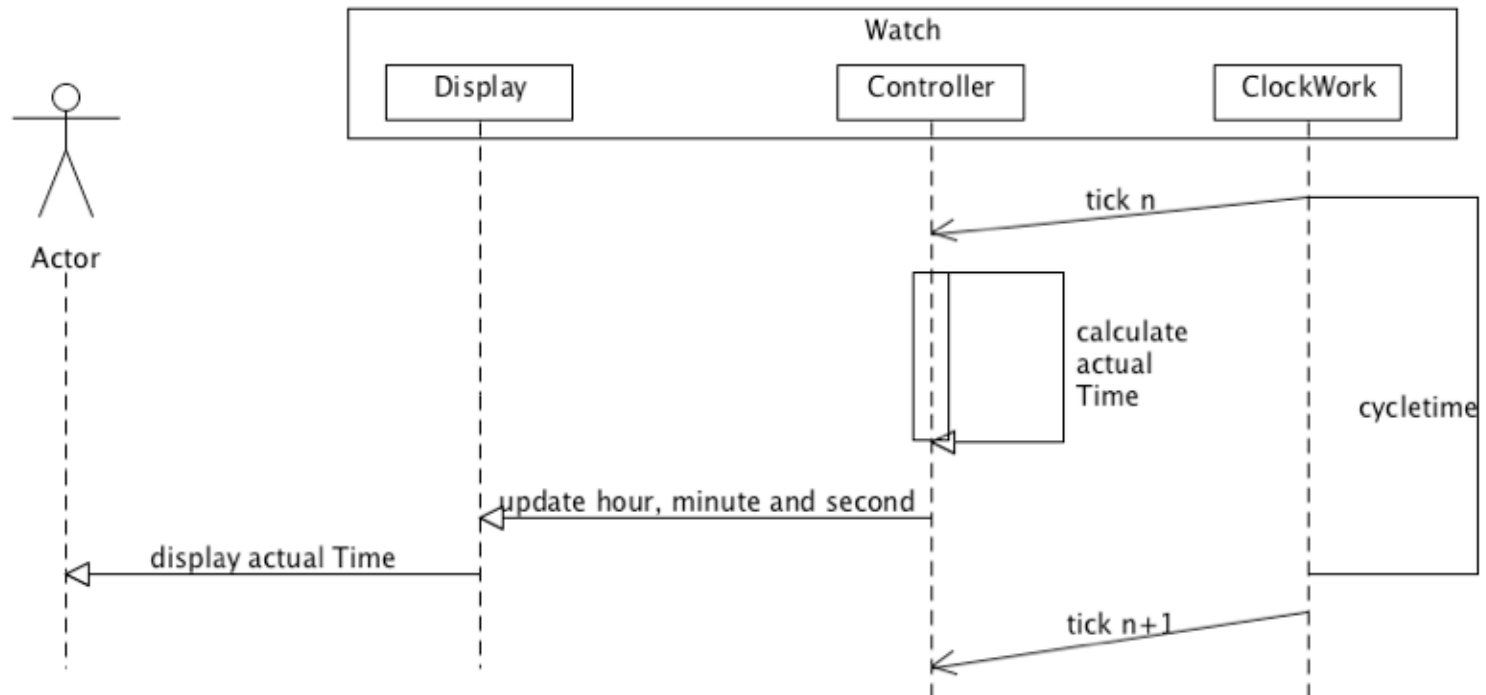
# Analyse

# Analyse Cas d'utilisation



# Analyse

## Scénario 1 de “See actual Time”



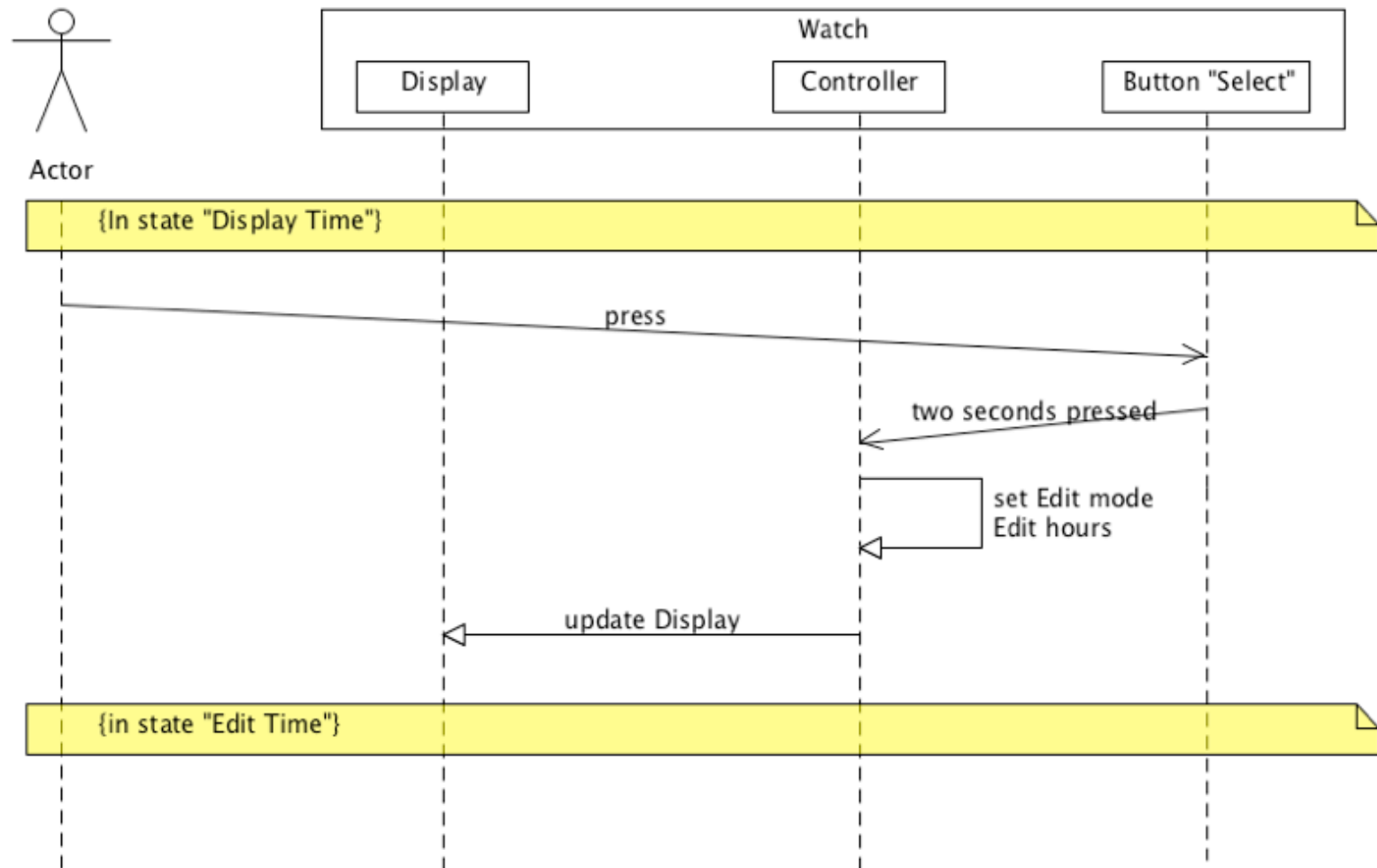
Repeat all this as long as connected to power

{the cycletime must NOT exceed the time between two ticks}

```
{the tick interval lasts 20 ms}
```

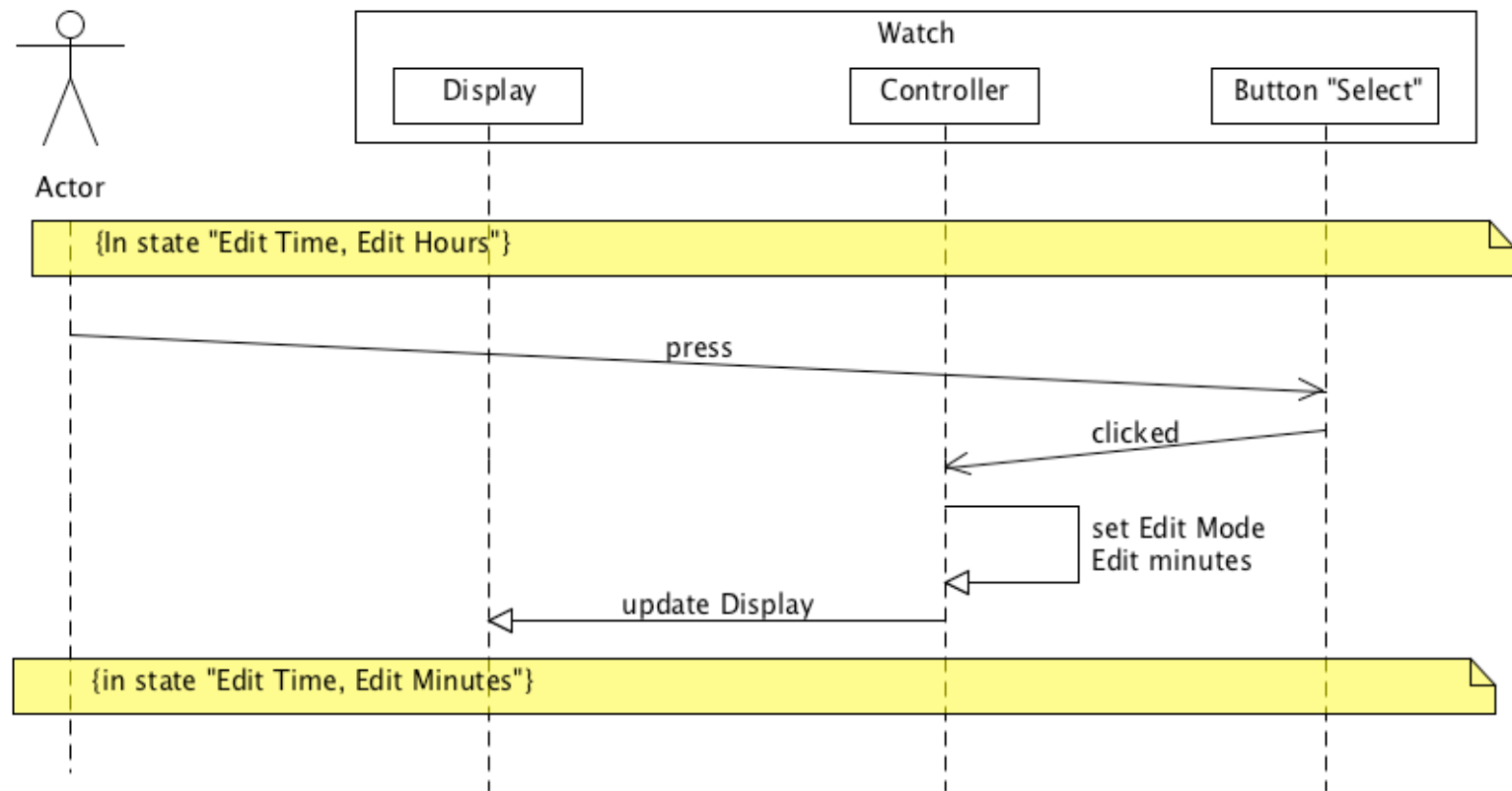
# Analyse

## Scénario 1 de "Press Button Select"



# Analyse

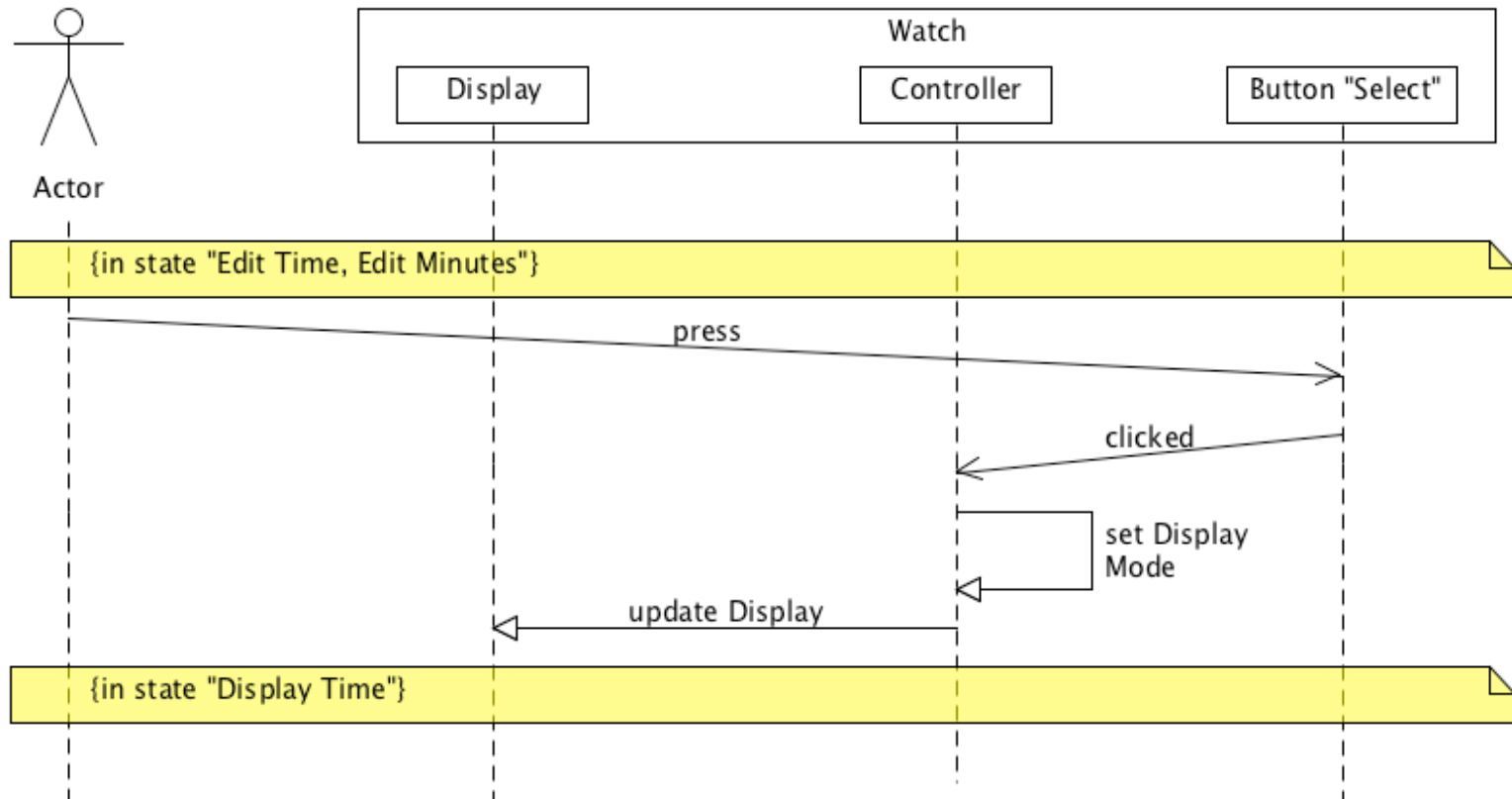
## Scénario 2 de "Press Button Select"





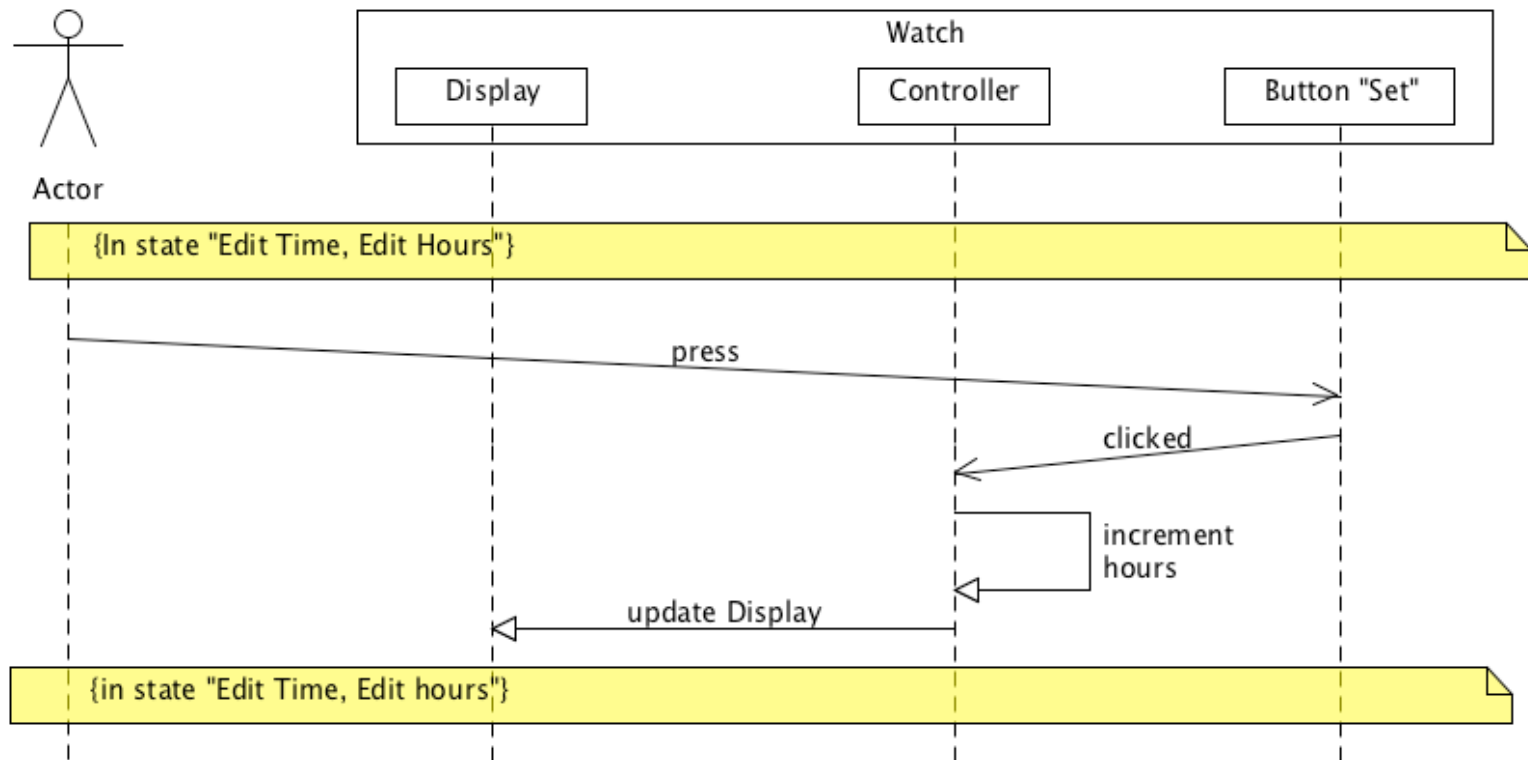
# Analyse

## Scénario 3 de "Press Button Select"



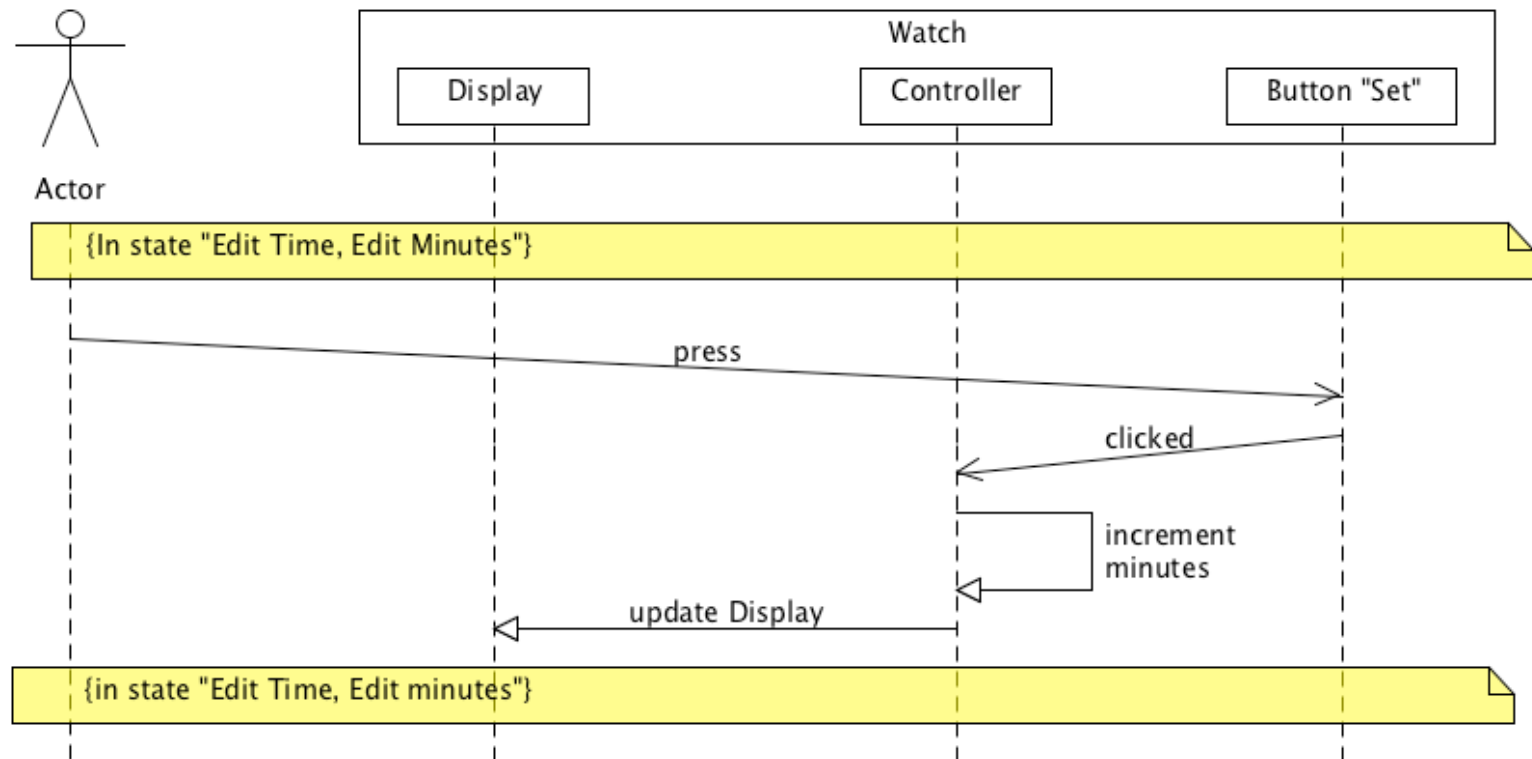
# Analyse

## Scénario 1 de "Press Button Set"



# Analyse

## Scénario 2 de "Press Button Set"



# Transition analyse -> conception

# Le différents “processus”

- De l’analyse, on peut conclure que ce système demande trois processus avec des contraintes de vitesse d’exécution différentes:
  - Un processus qui se synchronise avec les 50 Hz du secteur et qui génère les ticks
  - Un processus qui s’occupe du calcul et de l’affichage du temps
  - Un processus qui s’occupe des boutons représentant l’interface utilisateur
- Les différentes contraintes et les propriétés les plus importantes de ces processus ont besoin d’être étudié plus en détail

# Le propriétés des trois processus

## “Power Line Cycle Synchronizer”

|                       |  |
|-----------------------|--|
| <i>Process Name</i>   | Powerline Cycle Synchronizer                         |
| <i>Type</i>           | Interrupt Service Routine                            |
| <i>Priority</i>       | normal   |
| <i>Cycle Time</i>     | 5 [ms]   |
| <i>Driven by</i>      | Second Priority Interruption                         |
| <i>Implementation</i> | Sequential code, synchronous                         |
| <i>Realized by</i>    | Clockwork  |
| <i>Remarks</i>        | Simulation of PLL in Matlab, Translation into C-Code |

# Le propriétés des trois processus

## “Watch”

|                       |  |
|-----------------------|--|
| <i>Process Name</i>   | Watch                                  |
| <i>Type</i>           | reactive                               |
| <i>Priority</i>       | normal                                 |
| <i>Cycle Time</i>     | 100 [ms]                               |
| <i>Driven by</i>      | XF                                     |
| <i>Implementation</i> | Statemachine, asynchronous             |
| <i>Realized by</i>    | Clockwork, Controller, Display         |
| <i>Remarks</i>        | Responsible for time related functions |

# Le propriétés des trois processus

## “User Interface”

|                       |                                  |
|-----------------------|----------------------------------|
| <i>Process Name</i>   | User Interface                   |
| <i>Type</i>           | Reactive                         |
| <i>Priority</i>       | low                              |
| <i>Cycle Time</i>     | 20 [ms]                          |
| <i>Driven by</i>      | XF                               |
| <i>Implementation</i> | Statemachine, asynchronous       |
| <i>Realized by</i>    | Button, Controller, Display      |
| <i>Remarks</i>        | Responsible for user interaction |



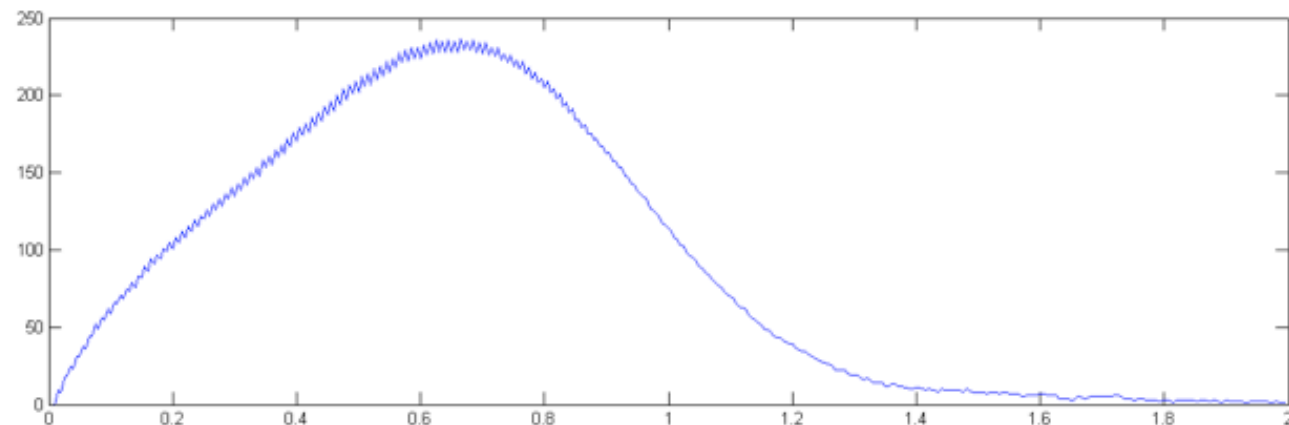
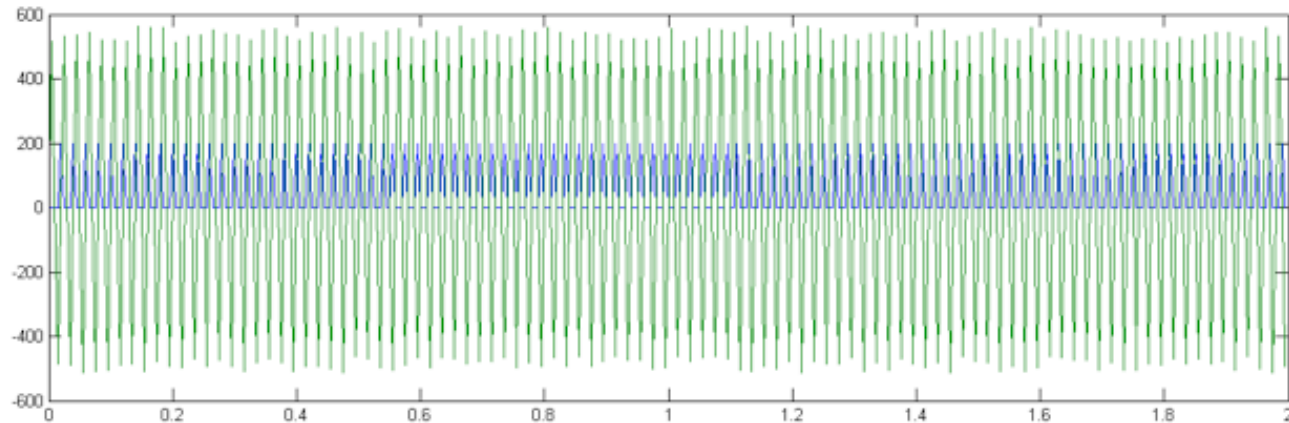
# Conception

# Conception Synchronizer

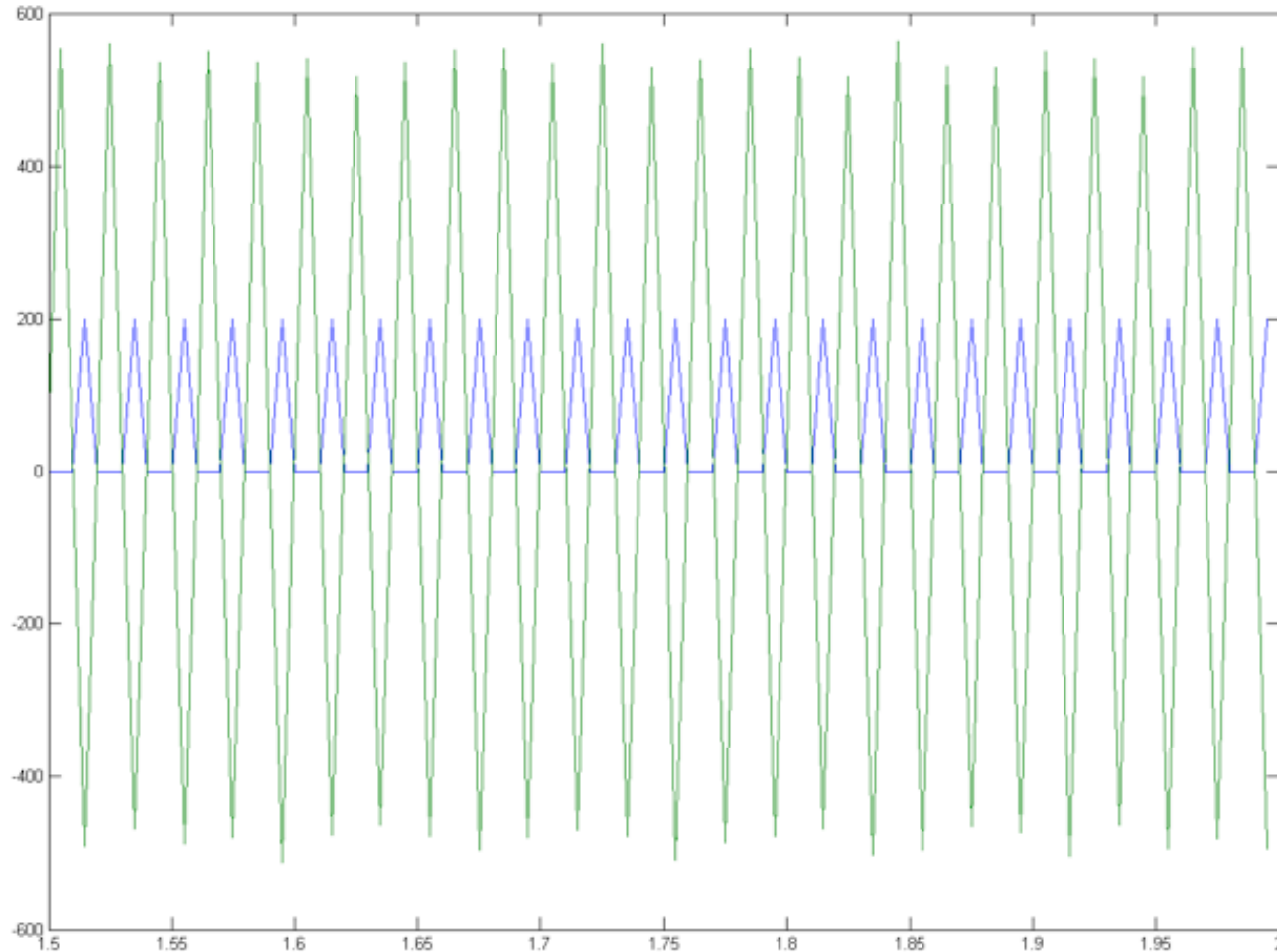
- 50 Hz représentent 20 ms. Pour pouvoir correctement détecter la fréquence, il faut faire tourner le PLL si souvent que possible.
- 5 ms semblent être correctes.
- Le code du PLL doit alors être optimisé pour ne pas dépasser les 2 ms. Comme ça il restent 3 ms pour l'exécution de tout les autres processus.
- Le synchroniseur est alimenté par une interruption d'un timer matériel de seconde priorité.

# Conception

## PLL Simulation de l'erreur



# Conception PLL, verouillage



# Conception

## XF

- Un XF est nécessaire pour faire fonctionner les processus réactifs (Watch, User Interface).
- “User Interface” demande une résolution de temps plus basse que “Watch”. La résolution maximale de temps (tic) du XF sera donc de 20 ms.
- Le tic du XF est alimenté par une interruption de première priorité d’un timer matériel.

# Conception

## Watch

- La montre doit internement gérer le dixièmes de seconde ( $100\text{ ms} = 5 \times 20\text{ ms}$ )
- Le processus sera base sur le tick 20 ms du XF.
- Le synchroniseur enregistre le nombre des cycles du secteur à un endroit accessible pour le processus “Watch”. Ceci peut être réalisé par une fonction “callback” soit par une zone de mémoire partagé entre le synchroniseur et “Watch” soit par un événement
- “Watch” se base sur ces ticks du synchroniseur pour afficher le temps précis.

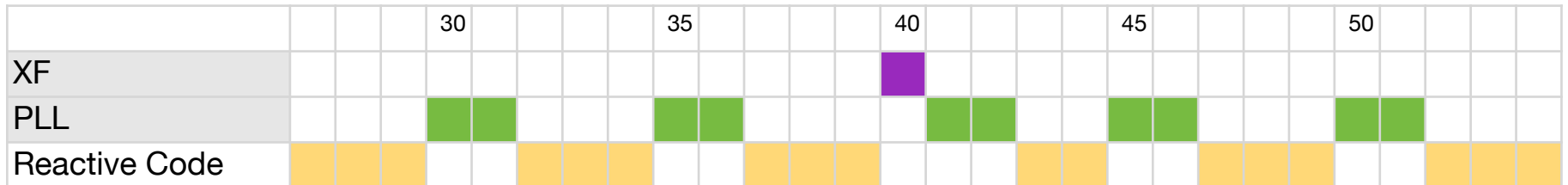
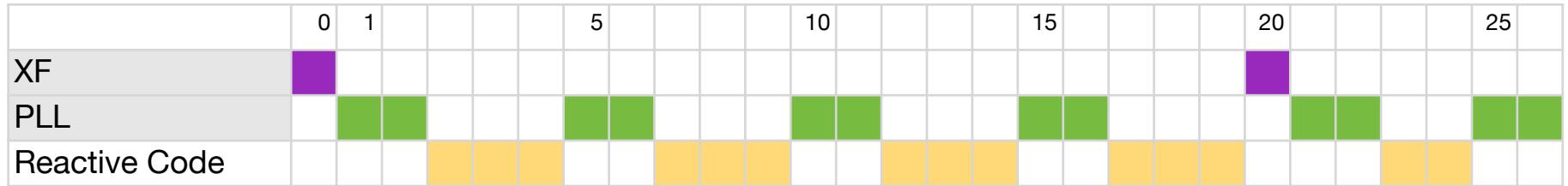
# Conception

## User Interface

- Le processus sera basé sur le tick 20 ms du XF.
- Les 20 ms garantissent un fonctionnement suffisamment réactif pour que l'utilisateur n'aie pas la sensation d'une réaction ralentie quand il appuie des boutons.

# Conception

## Diagramme des timings

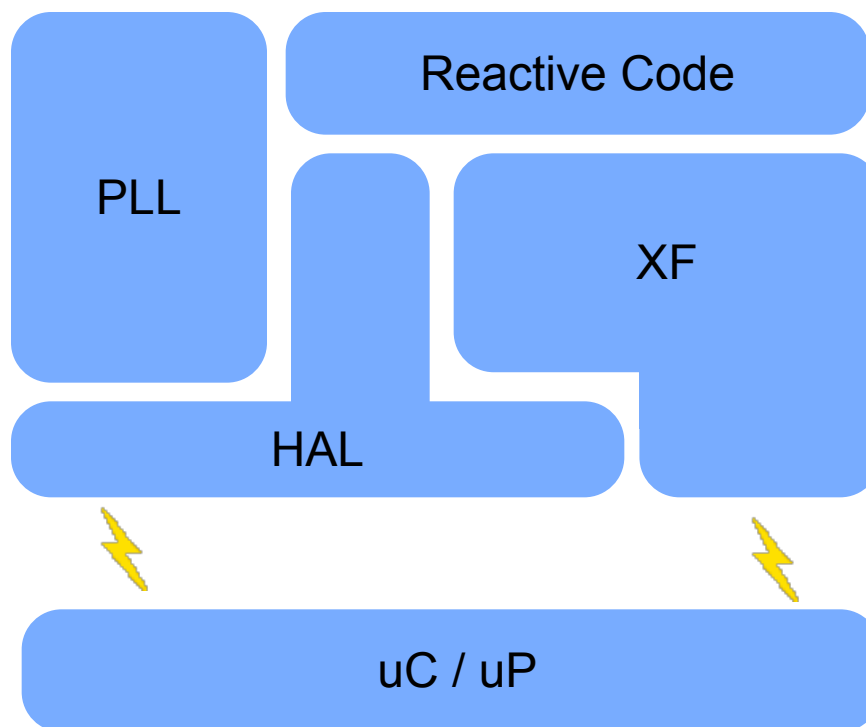


Reactive Code: Processes “Watch” and “User Interface”



# Conception

## Architecture finale



# Mise en oeuvre

# Mise en oeuvre

## PLL

- PLL classique
- Simulation en Matlab
- Mathématique Fixed Point, attention au nombre des bits
- Optimisation au niveau du langage C
- Mesure de la consommation de temps des différents éléments pour optimisation
- Durée totale de 2 ms, ce qui correspond à la conception prévue
- Optimisation possible jusqu'à 300 us!

# Mise en oeuvre

## HAL, LAL

- Button HAL
  - Modèle de conception Observateur
  - Deux événements si inscrit: Clicked, LongPressed
- LCD HAL
  - Librairie avec des fonctions pour l'initialisation, effacer l'écran et l'affichage de texte. Cet librairie contient aussi un espèce de simple tableau de caractères.
- Trace HAL
  - Fonctions pour l'initialisation et pour sortir des caractères vers le RS232

# Mise en oeuvre

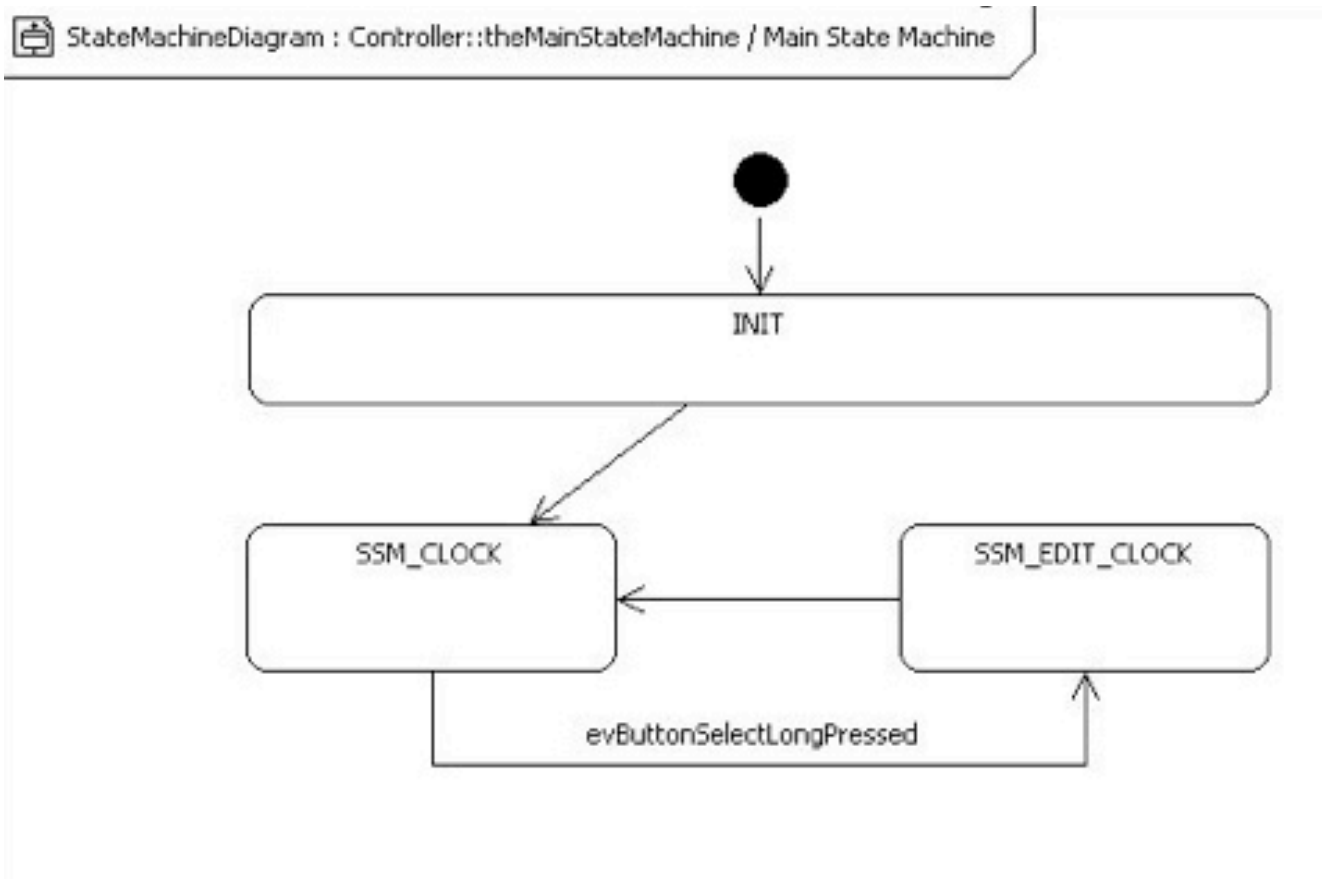
## HAL, LAL

- LCD LAL
  - Fonction pour l'initialisation
  - Fonction pour effacer l'écran
  - Fonction pour affichage de texte
- Button LAL
  - Fonction pour l'initialisation
  - Fonction pour interroger l'état actuel

# Mise en oeuvre XF

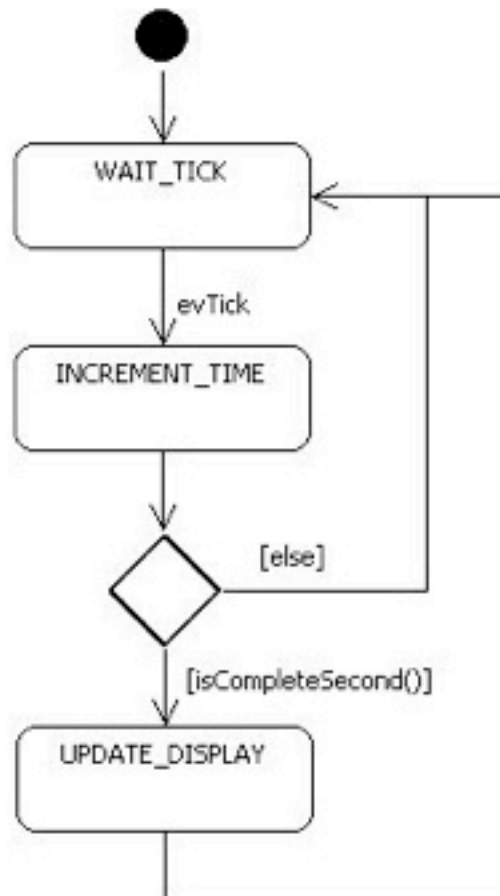
- Fille d'attente pour événements
  - 10 événements au maximum
- Gestionnaire pour timers
  - 5 événements au maximum
- Distributeur des événements
  - multiples machines d'état transitions
- Boucle principale
  - Modèle de conception Factory
- Sections critiques
  - Filles d'attente pour événements et timers
  - Verrouillage des ISR avec priorité différente

# Mise en oeuvre Controller Main



# Mise en oeuvre Controller Clock

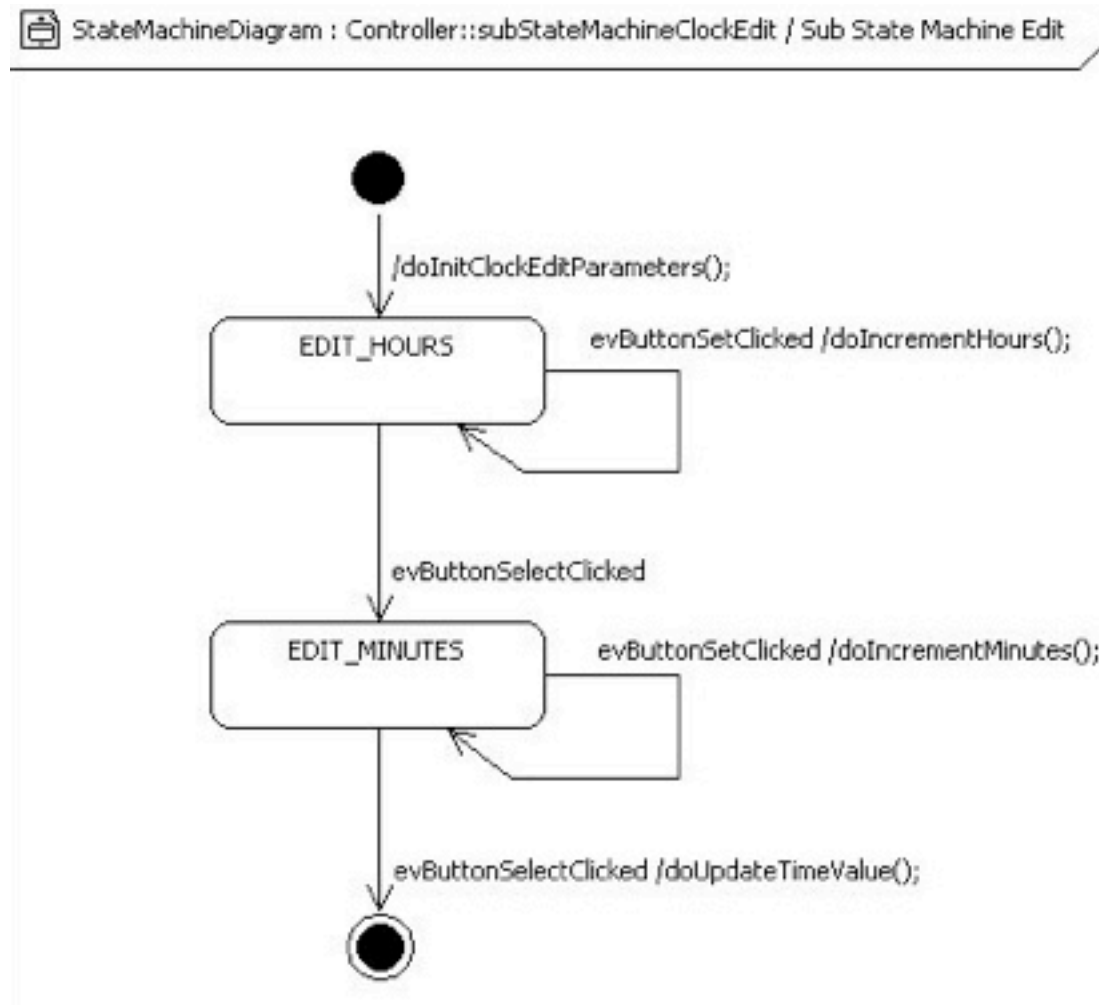
StateMachineDiagram : Controller::subStateMachineClock / Sub State Machine Clock





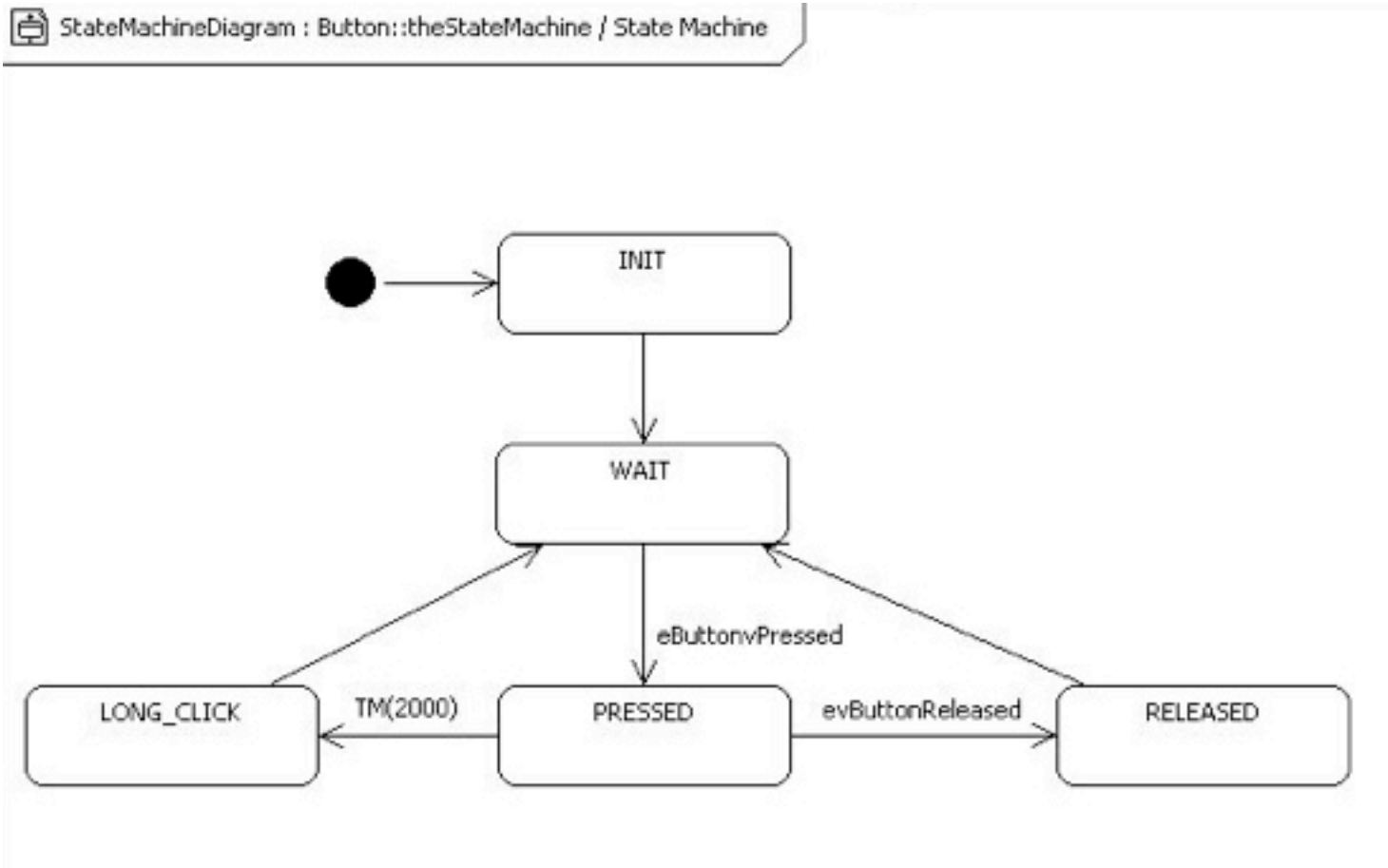
# Mise en oeuvre

## Controller Edit

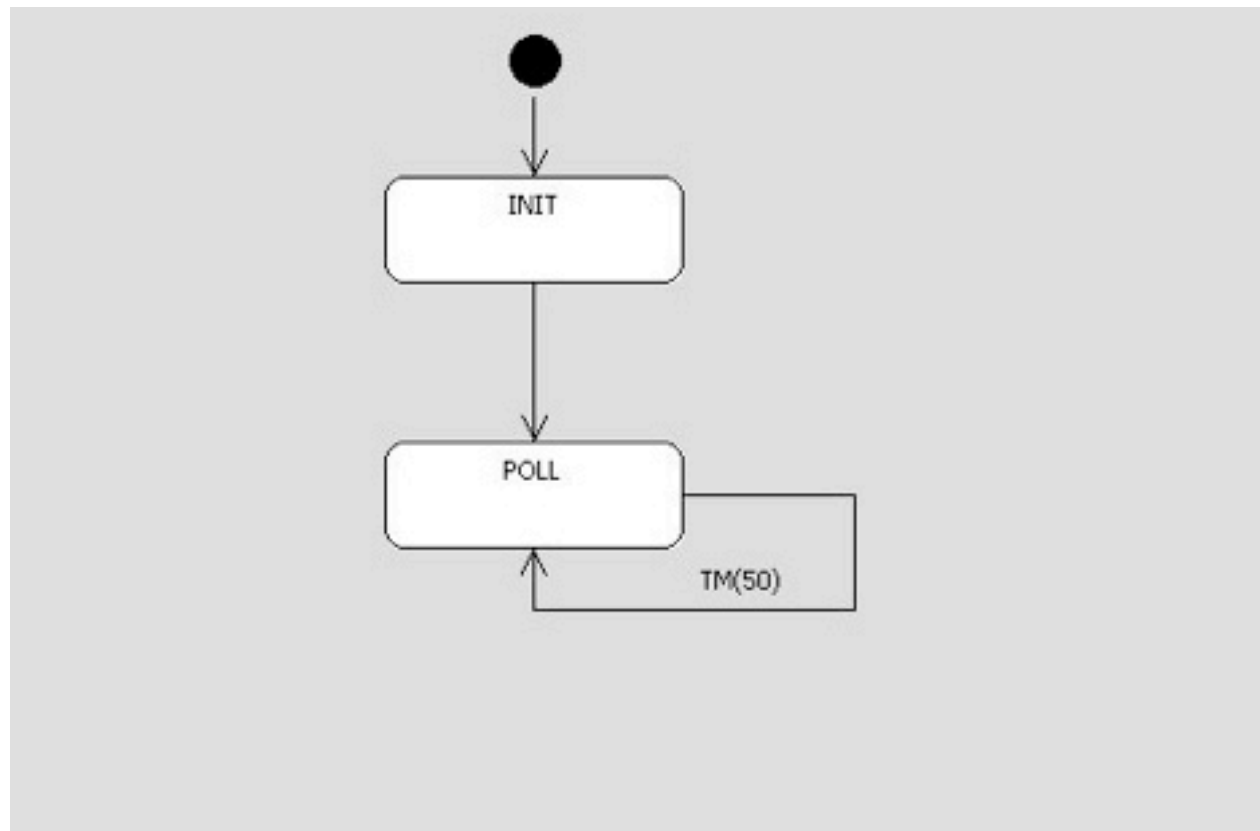


# Mise en oeuvre

## Button Events

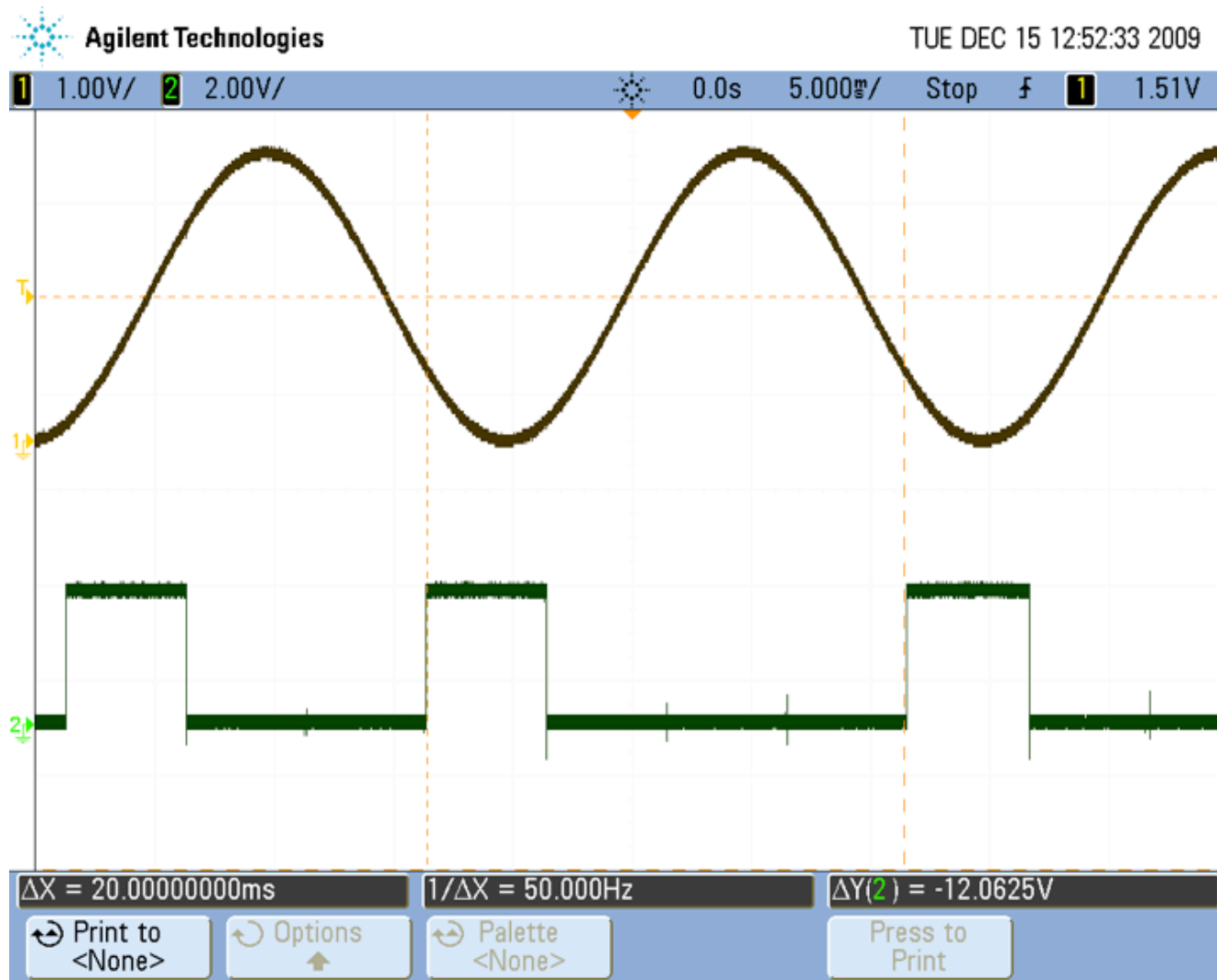


# Mise en oeuvre Button Poll



# Verification

# Vérification PLL, Fréquence



# Vérification Comportement

- Les comportements suivants ont été testés:
  - Mode montre
    - Affichage
    - Précision
    - Fonctionnement longue durée
  - Mode édition
    - heures
    - minutes

# Vérification Montre





# Vérification

## Edition, Heure





# Vérification Edition, Heure

