Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

Bd Pérolles 80
case postale 32
CH-1705 Fribourg
t. +41 (0)26 429 66 11
f. +41 (0)26 429 66 00
www.eia-fr.ch

# Console embarquée temps réel

## Rapport d'analyse

| | |
|---|---|
| **Cours** | Systèmes embarqués 2 |
| **Professeur** | Daniel Gachet, <daniel.gachet@hefr.ch> |
| **Etudiant** | Loïc Gremaud, <loic.gremaud@edu.hefr.ch> |
| **Classe** | T-2a |
| **Date** | 23.03.2014 |
| **Groupe** | 5 |

Hes·so FRIBOURG FREIBURG
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

# Table des matières

# Rapport d'analyse

## 1. Introduction

Ce rapport a pour but d'analyser certaines fonctionnalités que l'on désir mettre en œuvre dans le projet. Les fonctionnalités qui m'ont été attribué sont les suivantes :

- Ecran tactile
- Interpréteur de commande
- Chronomètre
- Thermomètre

Le thermomètre ne fera pas partie du rapport car il a déjà été réalisé lors d'un travail pratique. Il faudra juste faire une adaptation afin qu'il puisse tourner un parallèle avec les autres applications.

Voici une liste des fonctionnalités qui sont traité par les autres membres du groupe :

| **Membre** | **Fonctionnalité** |
|---|---|
| Romain Froidevaux | - Affichage<br>- Implémentation des commandes<br>- Prototype des interfaces graphiques |
| David Rossier | - Contrôleur des applications<br>- Protocole UDP/IP<br>- Ping |

# 2. Ecran tactile

Pour interagir avec nos applications, nous allons utiliser un écran tactile. Ce chapitre traite donc les aspects importants pour pouvoir gérer efficacement et facilement la partie tactile de notre écran.

## 2.1. Librairie

Nous avons à disposition la librairie « TSC2101 - Touch Screen Controller » qui permet de contrôler l'écran tactile. Elle est plutôt simpliste car elle ne possède que deux fonction. L'une est pour initialiser le contrôleur de l'écran tactile, le second pour récupérer la position d'un seul doigt sur celui-ci.

### 2.1.1. Position

La position est composée de trois coordonnées sur 16 bits. Les coordonnées x et y correspondent à la position sur l'écran et z à la pression exercée sur l'écran.

### 2.1.2. Test sur la cible

Après quelques tests effectués sur la cible voici les informations obtenues :

- Position pour le coin supérieur gauche : x = 1000, y = 50
- Position pour le coin supérieur droite: x = 1000, y= 950
- Position pour le coin inférieur gauche : x =30, y = 50
- Position pour le coint inférieur droite : x = 30, y = 950
- Forte pression : z varie entre 500 et 600
- Moyenne pression : z varie entre 300 et 500
- Faible pression ; z varie entre 0 et 300
- Lorsque l'on ne touche plus l'écran, c'est la dernière position et pression qui est donnée

## 2.2. Rafraîchissent

Comme la librairie que l'on a disposition ne gère par les interruptions pour signaler un changement de position, il faudra utiliser une des méthodes par scrutation. De nos jours, le taux de rafraîchissement minimum recommandé pour une souris est de 125 Hz mais ce ne sera pas nécessaire pour notre cas car 10 Hz pourrait être suffisant (à tester).

### 2.2.1. Mode de scrutation

Je vais présenter deux manières de déclencher la scrutation sachant qu'on ne va pas utiliser d'interruption matérielle. La routine d'exécution se charge d'aller lire la position sur le capteur et de la passer à la fonction de traitement du module de l'écran tactile.

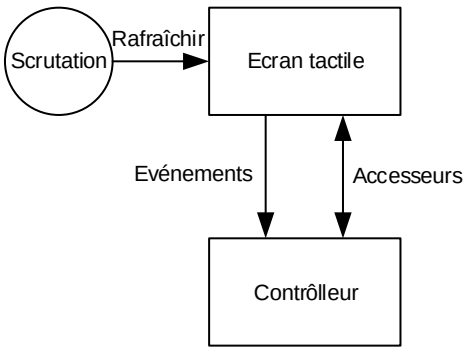


*Figure 1: Scrutation de la position sur l'écran tactile*

### Scrutation continue dans un thread

Un thread se charge de déclencher continuellement la routine de scrutation.

Avantage :

- Implémentation simple

Désavantage :

- Difficile à maîtriser le taux de rafraîchissement lorsque processeur est beaucoup utilisé
- Processeur trop sollicité

### Scrutation périodique

Un « timer » se charge de déclencher périodiquement la routine de scrutation.

Avantage :

- Maîtrise du taux de rafraîchissement
- Bonne réactivité peu importe le taux d'utilisation du processeur

Désavantage :

- Implémentation plus difficile
- Routine de scrutation et les fonctions appelées risque de bloquer trop longtemps les autres tâches

### Conclusion

La scrutation périodique semble être la meilleure des deux solutions mais il est possible de diminuer le temps d'exécution lors de l'appel de la routine de scrutation. On pourrait éviter que la fonction de traitement du module de l'écran tactile déclenche immédiatement l'événement sur le contrôleur. Pour cela, la fonction de traitement place l'événement dans une queue et un thread séparer se charge de déclencher les événements contenus dans la queue. En cas de forte utilisation du processeur, les actions de l'utilisateur ne sont pas perdues mais il risque d'avoir une latence avant d'avoir un résultat.

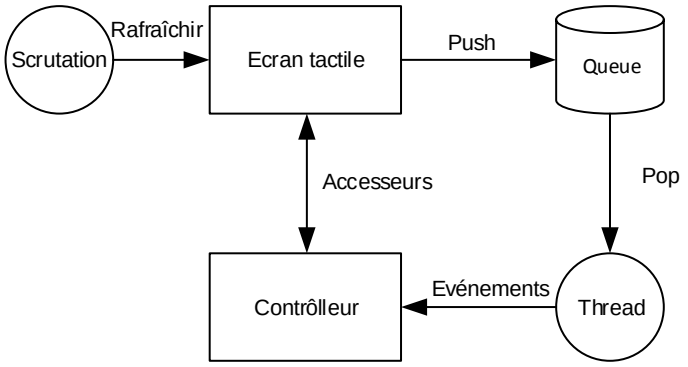


*Figure 2: Scrutation plus rapide*

## 2.3. Calibration

Les coordonnées données par la partie tactile ne correspondent pas au positionnement des pixels sur l'écran.

Pour pallier à cette différence entre l'affichage et le tactile, il faudra mettre un place un système de calibration. Le but de la calibration est de recueillir suffisamment de donnée afin de pouvoir passer d'un système de coordonnée à un autre.

### 2.3.1. Les différences

C'est l'affichage qui va servir de valeur de référence. Sur la partie tactile, on peut dire que :

- L'axe des x est inversé
- L'échelle n'est pas la même sur les axes x et y
- Pas la même hauteur, ni largeur
- Il y a un décalage sur les deux axes

Par contre, il n'y a pas de différence d'inclinaison et l'échelle est uniforme pour toute la largeur ou la hauteur ce qui implique un calcul plus simple.

| | **Affichage** | **Tactile (mesuré)** |
|---|---|---|
| **Hauteur** | 480 | ~920 |
| **Largeur** | 800 | ~950 |
| **Direction axe x** | haut vers bas | bas vers haut |
| **Direction axe y** | gauche vers droite | gauche vers droite |
| **Valeur de départ axe x** | 0 | ~30 |
| **Valeur de départ axe y** | 0 | ~50 |

*Tableau 1: Différence entre l'affichage et l'écran tactile*

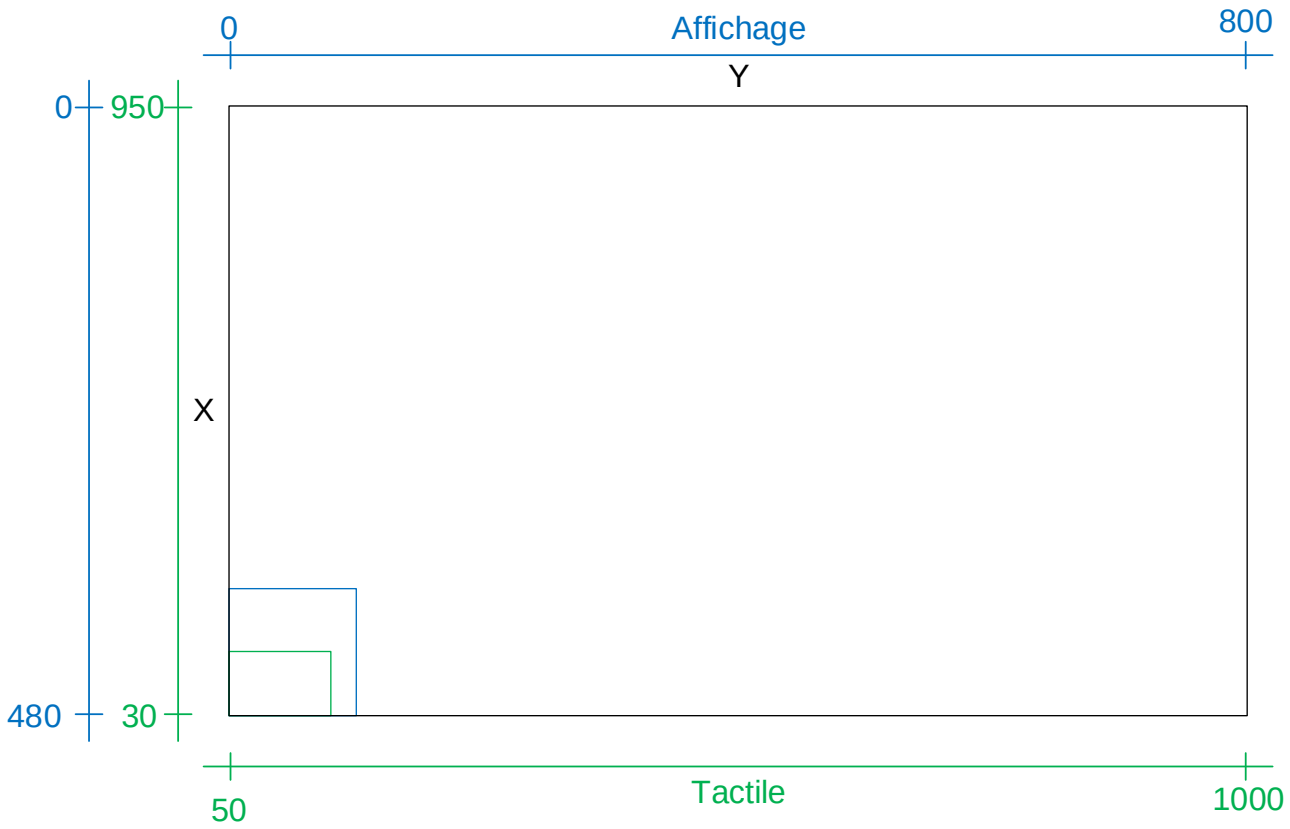


*Figure 3: Représentation des différences*

## 2.4. Système de calibration

Cette partie traite de la création d'un système de calibration. Pour notre fonction de conversion, il nous faudra au minimum connaître les valeurs x min/max et y min/max de la partie tactile, les valeurs de l'affichage étant déjà connus. La difficulté pour recueillir ces valeurs provient du fait qu'il n'est pas possible de cliquer tout au bord de l'écran, il faudra donc laisser une marge.

### 2.4.1. Etapes de calibration

Voici les étapes de calibrations, vue de l'utilisateur :

1. Annoncer à l'utilisateur qu'il doit cliquer sur les points qui vont apparaître à l'écran et qu'il est préférable de le faire avec l'aide d'un stylet pour améliorer la précision.
2. Faire apparaître le premier point et attendre.
3. Faire apparaître un second point et attendre.
4. *Faire éventuellement apparaître d'autres points pour améliorer la précision en faisant une moyenne.*
5. La calibration est terminée.

### 2.4.2. Fonction de conversion

Voici les opérations mathématiques qui vont aider à la réalisation de la fonction de conversion d'un point de l'écran tactile en pixel de l'affichage. La fonction inverse n'étant pas indispensable, ne va pas être développé.

Soit $x$ le point à convertir, $h_{affichage}$ la hauteur de l'affichage, $x_{min\ calibration}$ la valeur de départ de l'écran tactile selon la dernière calibration, $h_{calibration}$ la hauteur de l'écran tactile selon la dernière calibration.

$$x' = (h_{affichage} - 1) - \frac{(x - x_{min\ calibration}) * h_{affichage}}{h_{calibration}}$$

Soit $y$ le point à convertir, $l_{affichage}$ la largeur de l'affichage, $y_{min\ calibration}$ la valeur de départ de l'écran tactile selon la dernière calibration, $l_{calibration}$ la largeur de l'écran tactile selon la dernière calibration.

$$y' = \frac{(y - y_{min\ calibration}) * l_{affichage}}{l_{calibration}}$$

Légende :

- En vert, inversion des valeurs sur l'axe
- En bleu, correction du décalage sur l'axe
- En noir, correction du ratio

Remarque :

- Les largeurs et hauteurs ne sont pas forcément celles de l'écran mais peuvent être celles données par un rectangle sur l'écran (voir Figure ci-dessous). Le but étant qu'il représente la même grandeur dans les deux systèmes de coordonnées pour faire la correction du ratio.

### 2.4.3. Fonction de calibration

Le but de la fonction de calibration est de récupéré les valeurs que la fonction de conversion a besoin, respectivement le décalage en x et y et la taille en x et y de l'écran tactile. Il nous faut au minimum de deux points qui se trouvent à l'opposé l'un de l'autre.

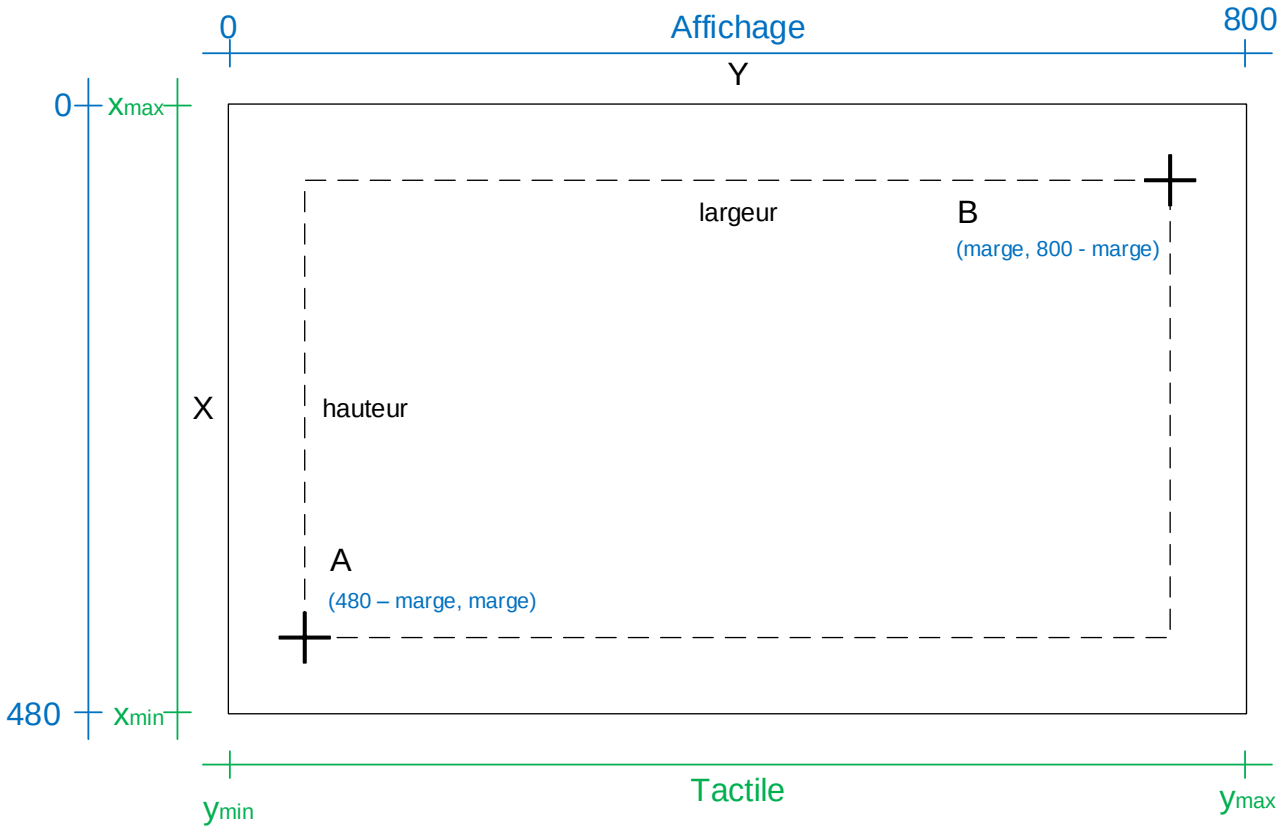


*Figure 4: Calibration*

Soit $A$ le point de calibration avec les minimas, B le point de calibration avec les maximas.

$$h_{affichage} = 480 - 2 * marge$$

$$l_{affichage} = 800 - 2 * marge$$

$$h_{calibration} = B_x - A_x$$

$$l_{calibration} = B_y - A_y$$

$$x_{min\ calibration} = A_x - \frac{marge * h_{calibration}}{h_{affichage}}$$

$$y_{min\ calibration} = A_y - \frac{marge * l_{calibration}}{l_{affichage}}$$

Légende :

- En vert, calcul de la marge sur l'écran tactile avec la correction du ratio

Remarque :

- Plus de points de calibrations peuvent être définis et on peut prendre la moyenne des valeurs calculées.

## 2.5. Gestionnaire de disposition

Pour la construction d'une interface tactile, il est important d'avoir un gestionnaire de disposition. Il va permettre de gérer de et structurer facilement les éléments cliquables. Dans ce cas le gestionnaire de disposition s'occupe uniquement de la partie tactile mais pourrait également être utilisé pour la partie affichage.

### 2.5.1. Définition des besoins

Il est essentiel de définir les besoins afin que l'on puisse définir une bonne structure ainsi tous les objets que nous allons utiliser. Ce sont les prototypes d'interface graphique de Romain Froidevaux (cf. annexe) qui va servir de référence pour trouver ce dont nous avons besoins.+69

Dans toutes nos interfaces, il y a trois zones fixes qui gardent la même dimension et position durant tout le fonctionnement. Des éléments cliquables peuvent apparaître dans toutes ces zones.

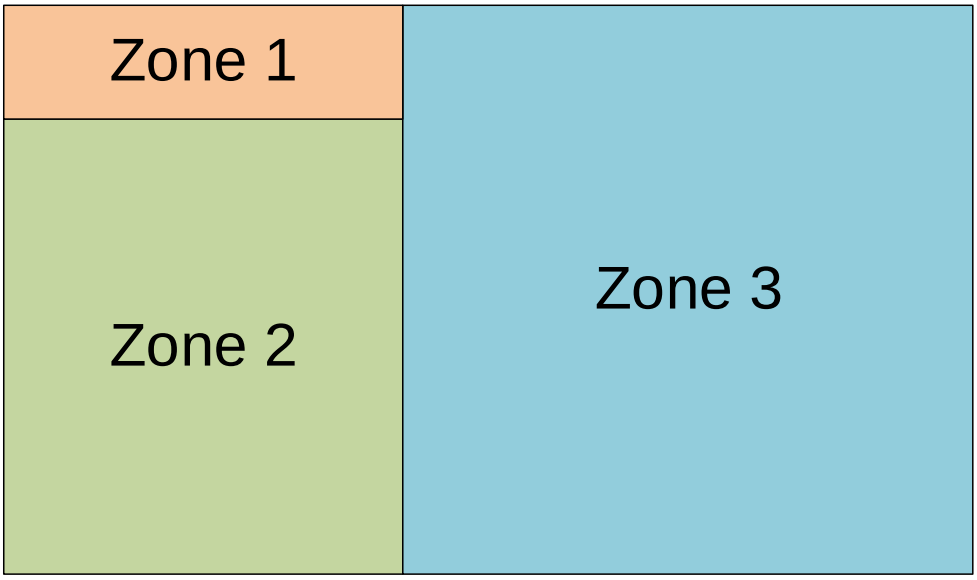


*Figure 5: Zones de l'interface graphique*

Nous avons également des boutons ainsi que des grilles dans lesquelles les cases sont cliquables.

Comme notre interface est relativement petite et que nous n'avons pas de zone mobile, nous pouvons utiliser une structure simple et qui est facilement adaptable peut être pour d'autres applications. Les composants proposés sont les suivants :

- Ecran
- Zone
- Zone avec boutons
- Zone avec grille
- Boutons

Pour nous simplifier, nous allons gérer que deux couches de composants imbriqués aux maximum et qu'une seule fonction par zone lorsqu'un bouton pressé.

### 2.5.2. Ecran

C'est la racine de tous les composants, il y en a qu'une active pour toute l'application. Un **Ecran** possède plusieurs **Zone** et permet de les gérer. Un processus de rafraîchissement sera chargé d'appeler la fonction de détection de changement périodiquement.

### Fonctions

- Initialiser :
  - Récupérer les dimensions de l'écran
- Donner la dimension de l'écran
- Donner une **Zone** selon un identifiant
- Ajouter une **Zone**
- Supprimer une **Zone** selon son identifiant
- Supprimer toutes les **Zone**
- Désactiver l'écran tactile :
  - Termine le processus de rafraîchissement
- Activer l'écran tactile
  - Démarre le processus de rafraîchissement
- Détecter un changement :
  - Vérifier si la position et/ou la pression a changé depuis la dernière fois.
  - Lorsqu'un clic est détecté, appeler la fonction **Actionner** de la **Zone** correspondante en lui passant la position du clic.

## 2.5.3. Zone

C'est une zone de l'écran qui est tactile. Elle est définit par rapport à sa taille et sa position sur l'écran. Afin de simplifier l'implémentation, le développeur devra faire en sorte qu'aucunes zones ne se chevauchent et que les identifiants uniques pour l'écran.

### Fonctions

- Constructeur :
  - Passer un identifiant
  - Passer la position sur l'écran
  - Passer la taille
- Donner sa position sur l'écran
- Donner sa taille
- Donner l'identifiant
- Fixer la référence vers la fonction qui doit être appelé par **Actionner**
- Actionner :
  - Lorsque la zone est activée, appeler la fonction en référence en passant la position du clic.
- Désactiver la zone
- Activer la zone

## 2.5.4. Zone avec boutons

Etend le composant **Zone** en offrant la possibilité d'ajouter des boutons cliquables. Une **Zone** possède plusieurs boutons. Lorsqu'un bouton a été pressé, une seule fonction pour toute la zone est appelé avec l'identifiant du bouton.

### Fonctions

- Constructeur :
  - Appeler la fonction **Fixer référence** du parent **Zone** en passant la référence vers la fonction **Actionner un bouton**
- Fixer la référence vers la fonction qui doit être appelé par **Actionner un bouton**
- Ajouter un **Bouton**
- Supprimer un **Bouton** selon son identifiant
- Supprimer tous les **Bouton**
- Actionner un bouton :
  - Appeler la fonction en référence en passant l'identifiant du bouton.

## 2.5.5. Zone avec grille

Etend le composant **Zone** en offrant une grille d'élément cliquable. Lorsqu'une case a été cliqué, une seule fonction pour toute la zone est appelé avec la coordonnées celui-ci.

### Fonctions

- Constructeur :
  - Appeler la fonction **Fixer référence** du parent **Zone** en passant la référence vers la fonction **Actionner grille**
- Donner la taille de la grille
- Fixer la taille de la grille
- Fixer la référence vers la fonction qui doit être appelé par **Actionner une case**
- Fixer la taille de la grille
- Actionner une case :
  - Appeler la fonction en référence en passant les coordonnées de la case.

## 2.5.6. Boutons

Le bouton est un composant cliquable. Il est définit par sa taille et sa positions sur la **Zone avec boutons**. Afin de simplifier l'implémentation, le développeur devra faire en sorte qu'aucuns boutons ne se chevauchent et que les identifiants soient unique pour chaque zone.

### Fonctions

- Constructeur :
  - Passer l'identifiant
  - Passer la position sur l'écran
  - Passer la taille
- Donner l'identifiant
- Donner sa position sur la zone
- Donner sa taille

# 3. Interpréteur de commandes

Un interpréteur de commande, « Shell » est nécessaire pour pouvoir effectuer certaines opérations sur la machine. Ce chapitre ne traite pas de l'implémentation des commandes mais de la manière à gérer les entrées/sorties sur la console.

## 3.1. Commandes

Dans cette partie, nous allons étudier les commandes dont nous avons besoins afin de déterminer comment construire notre interpréteur de commandes.

### 3.1.1. Configuration du réseau

Il faudra pouvoir récupérer et modifier l'adresse IP.

#### Commandes

```
ip set <ip address> <subnet mask> [<default gateway>]

ip show
```

### 3.1.2. Calibration de l'écran

Démarrer le programme de calibration de l'écran :

```
screen calibrate
```

### 3.1.3. Ping

Envoyer un ping vers une adresse IP de destination :

```
ping <ip destination>
```

### 3.1.4. Gestion des applications

Démarrer une application :

```
app start <application name>
```

Arrêter une application :

```
app stop <application name>
```

Lister toutes les applications ou celles qui sont en cours d'exécution ou arrêté :

```
app list [-running|-halted]
```

### 3.1.5. Gestion du chronomètre

Démarrer le chronomètre :

```
chrono start
```

Arrêter le chronomètre :

```
chrono stop
```

Mettre en pause le chronomètre :

```
chrono pause
```

Affiche la valeur du chronomètre à cet instant ou en l'affichant continuellement avec possibilité de le stoppé avec CTRL+C par exemple :

```
chrono show [-refresh]
```

### 3.1.6. Console

Remettre à zéro le contenu de la console :

```
clear
```

### 3.1.7. Aide

Affiche la liste des commandes disponibles :

```
help
```

Affiche l'aide d'une commande :

```
<commande> -help
```

## 3.2. Structure des commandes

Voici une convention simple proposé pour structurer nos commandes. La commande est lancée lors d'un saut de la ligne.

```
commande [action] [-option] [<paramètre 1> [<paramètre 2> [<paramètre 3> ...]]]
```

### 3.2.1. Commande

Chaque commande correspond à un module qui contient une fonction de base où l'on pourra y passer l'action, l'option et le tableau de paramètres. C'est l'interpréteur qui va se charger de vérifier si la commande est correcte et d'appeler la fonction de base du bon module.

### 3.2.2. Action

Dans une commande, on la possibilité de spécifier une seule action. La fonction de base sera chargée d'appeler la fonction correspondant à l'action. Il faudra se charger de vérifier que :

- L'action est valide
- L'action est obligatoire ou non

Une fois la vérification effectué, on peut s'occuper de l'option.

### 3.2.3. Option

Dans la fonction de l'action, il y a la possibilité de spécifier une seule option. Il faudra se charger de vérifier que :

- L'option est valide

### 3.2.4. Paramètres

Comme on peut recevoir un nombre indéterminé de paramètre, il faudra les placer dans un tableau. Il faudra se charger de vérifier que :

- Chaque paramètre soit du bon type ou qu'une conversion est possible
- Il y ait le bon nombre de paramètres

Une fois la vérification effectué, on peut appeler la fonction qui implémente ce que l'on veut faire en lui passant les bons paramètres.

## 3.3. Interface série

Nous aurons à disposition un pilote pour la 2ème interface série de l'APF27 en mode interruptif.

Voici ce que l'on devra faire sur un thread qui gérera l'interface série :

- Afficher les caractères saisis sur la console distante et sur le moniteur de l'écran
- Passer la chaîne de caractère à l'interpréteur lors de la saisie du caractère de retour à la ligne
- Gérer correctement le caractère d'effacement

# 4. Chronomètre

Pour notre projet, on devra pouvoir afficher un chronomètre pendant l'exécution d'autre programme. Le chronomètre pourra être géré avec l'aide des boutons poussoirs.

## 4.1. Gestion des boutons

Nous n'allons pas traiter de la gestion des boutons car nous l'avons déjà fait lors d'un travail pratique.

## 4.2. Module GPT

Le module GPT, « General Purpose Timer » de l'i.MX27 implémente les fonctions que l'on a besoin, c'est-à-dire :

- Démarrer un compteur
- Mettre en pause le compteur
- Arrêter le compteur

Il y a 6 compteurs de 32-bit à dispositions mais pour le chronomètre un seul est nécessaire.

Remarque : C'est aussi ce module qui va être utilisé pour faire des interruptions périodiques.

### 4.2.1. Horloge

Il y a possibilité de passer différente horloge pour l'incrémentation du compteur.

| Horloge | Valeur | Description |
|---|---|---|
| Aucune | 0x0 | Compteur désactivé |
| IPG_CLK_PERCLK | 0x1 | Horloge principal (attention prescaler doit être > 2), |
| IPG_CLK_PERCLK 2 | 0x2 | Horloge principal divisé par 4 |
| IPP_GPT_TIN | 0x3 | Horloge externe que l'on peut connecter sur la puce |
| IPG_CLK_32K | 0x4-7 | Une horloge fonctionnant à 32 kHz, elle a la particularité de ne pas s'arrêter lorsque le système est en veille |

*Tableau 2: Horloges*

### 4.2.2. Mode d'opération

Il existe deux modes d'opération que l'on peut configurer sur le registre de contrôle de GPT.

#### Restart mode

Le timer redémarre à zéro lorsqu'une valeur qui se trouve dans le registre de comparaison est atteinte.

#### Freerun mode

Le timer redémarre à zéro lorsque la valeur maximale de celui-ci est atteinte.

### 4.2.3. Registres

Pour chaque compteur GPT, il y a 6 registres de 32-bit à dispositions, toutes les valeurs commencent à partir du bit de poids faible.

| Registre | Nom | Description |
|---|---|---|
| TCTL1-6 | Control Register | Registre de contrôle |
| TPRER1-6 | Prescaler Register | Valeur pour la diviseur d'horloge (11-bit, RW) (0x00 pour divisé par 1 et 0x7FF pour divisé par 2048) |
| TCMP1-6 | Compare Register | Valeur de comparaison (32-bit, RW) pour redémarrer à zéro en mode restart et peut aussi générer une interruption |
| TC1-6 | Capture Register | Valeur de capture (32-bit, RO) pour générer une interruption |
| TCN1-6 | Counter Register | Valeur du compteur (32-bit, RO) |
| TSTAT1-6 | Status Register | Contient les flags pour savoir si le compteur a atteint son maximum (bit 0, RO) ou si une interruption a eu lieu (bit 1, RO) |

*Tableau 3: GPT registres*

Nous allons nous intéresser sur le registre de contrôle car c'est le plus complexe. Pour les autres registres la description donnée ci-dessous contient suffisamment d'information.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | CC | OM | FRR | CAP | | CAPT EN | COMP EN | CLK SOURCE | | | TEN |
| W | SWR | - | - | - | - | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Tableau 4: GPT registre de contrôle*

Et voici les fonctions de chaque champ :

| Champs | Nom | Description |
|---|---|---|
| TEN | Timer Enable | Enclenche 1 ou déclenche 0 le compteur |
| CLKSOURCE | Clock source | Sélection de l'horloge source (voir Horloges ci-dessus) |
| COMP EN | Compare Interrupt Enable | Enclenche 1 ou déclenche 0 l'interruption sur le registre Compare. |
| CAPT EN | Capture Interrupt Enable | Enclenche 1 ou déclenche 0 l'interruption sur le registre Capture |
| CAP | Capture Edge | Sélection du type de flanc pour la fonction de capture. La fonction de capture a pour but de copier la valeur du compteur dans le registre de capture lorsqu'un flanc est détecté sur le pin TIN.<br>0x0 : capture désactivé<br>0x1 : capture sur le flanc montant, génère une interruption<br>0x2 : capture sur le flanc descendant, génère une interruption<br>0x3 : capture sur les deux flancs, génère une interruption |
| FRR | Free-Run/Restart | Séléction du mode, 0 pour restart, 1 pour Free-run |
| OM | Output mode | |
| CC | Counter clear | Action lorsque on déclenche le compteur (TEN = 0)<br>0 : le compteur est stoppé<br>1 : le compteur est mis à zéro |
| SWR | Software reset | Permet de reseter complétement le compteur.<br>Il faut maintenir à 1 pendant deux coups d'horloge (IPG_CLK) et le reset prend 3 autres cycles. Il faut 5 cycles pour faire un reset. |

*Tableau 5: GPT détails registre de contrôle*

# 4.3. Rafraîchissement

Il existe plusieurs façons de rafraîchir sur l'écran la valeur du chronomètre.

Par scrutation, scrutation périodique ou par interruption. Les deux fonctions de scrutation sont un peu similaire que celui que j'ai expliqué pour l'écran tactile.

## 4.3.1. Interruption

Admettons que l'on veuille faire un chronomètre qui affiche que les secondes, il faudrait générer une interruption toutes les secondes et appeler directement la fonction qui met à jours la valeur sur l'affichage.

Avantage comparé aux méthodes de scrutation :

- L'affichage est modifié quasiment à la second près
- Simple comme on utilise déjà un module qui génère des interruptions

Désavantage :

- Les autres applications sont interrompu juste pour modifié un affichage qui n'est pas forcément le plus important

# 5. Conclusion personnelle

Cette analyse m’a permis de mieux comprendre le fonctionnement d’un écran tactile en général ainsi que des compteurs GPT. Les informations se trouvant dans ce rapport vont m’aider à la réalisation des pilotes et des programmes. Les parties qui m’ont données le plus de fils à retordre sont la calibration de l’écran ainsi que les compteurs GPT, en effet durant la rédaction de ce rapport j’ai déjà essayé de réfléchir plus loin afin d’avoir une idée pour les prochaines étapes du projet.

La Roche, le 23 mars 2014,

Loïc Gremaud

# 6. Bibliographie

[1] A. Tanenbaum, Systèmes d'exploitation, Paris: PEARSON, 2008.

[2] D. Gachet, «Systèmes Embarqués 1 & 2: Entrées - Sorties,» HES-SO / EIA-FR, Fribourg, 2014.

[3] Freescale Semiconductor, «MCIMX27 Multimedia Applications Processor Reference Manual,» Freescale Semiconductor, Chandler, Arizona, 2008.

# 7. Annexes

- Prototypes d’interface graphique de Romain Froidevaux

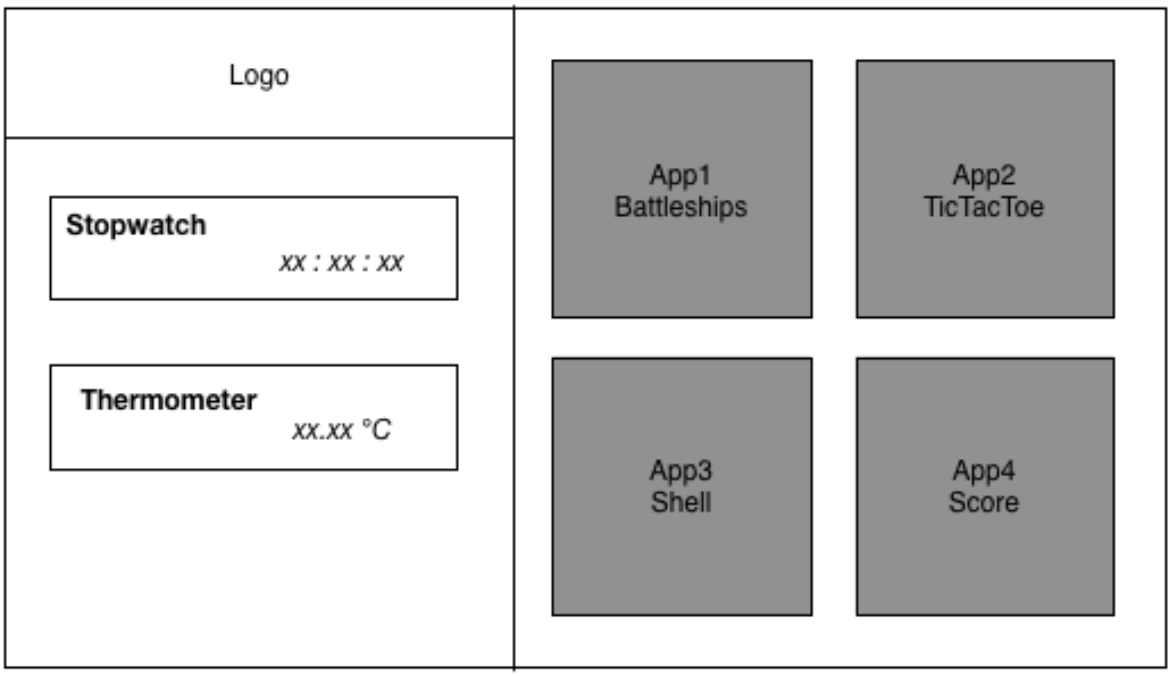


*Figure 1: Accueil*

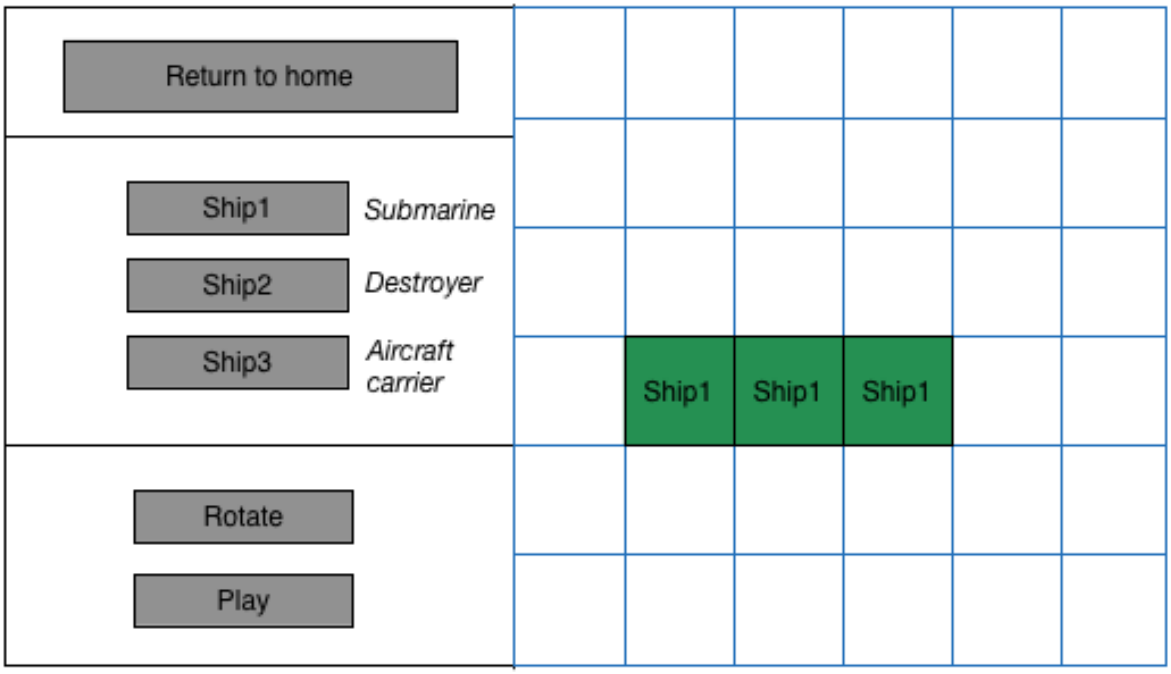


*Figure 2: Bataille navale*

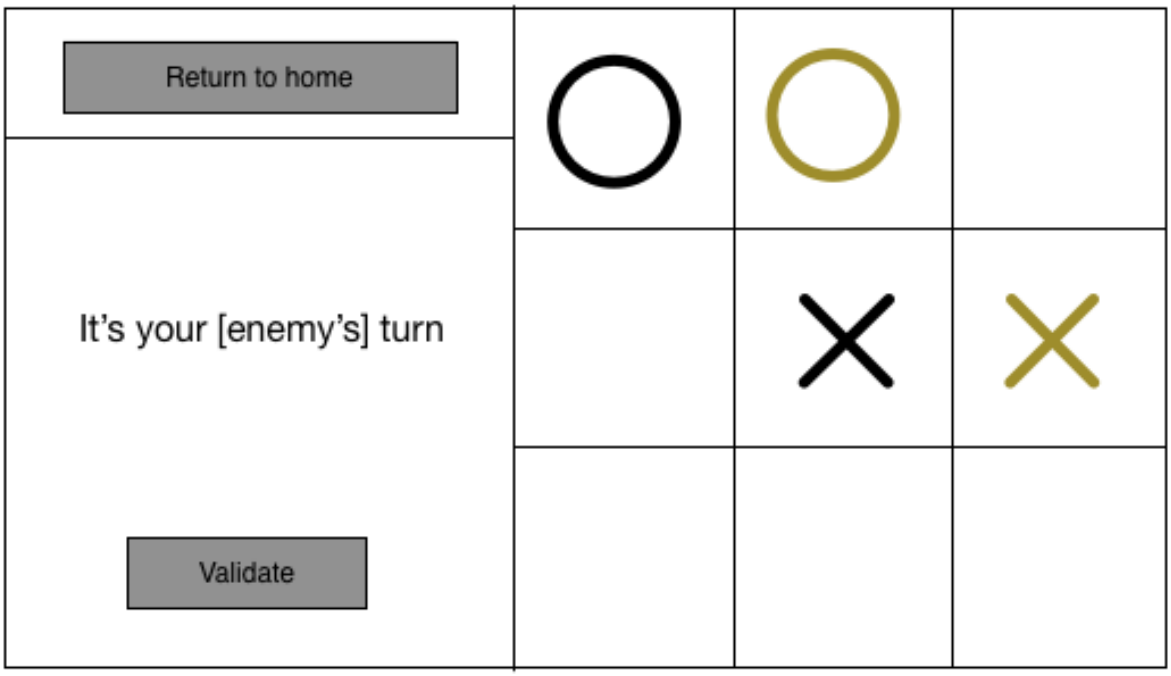


*Figure 3: Tic Tac Toe*

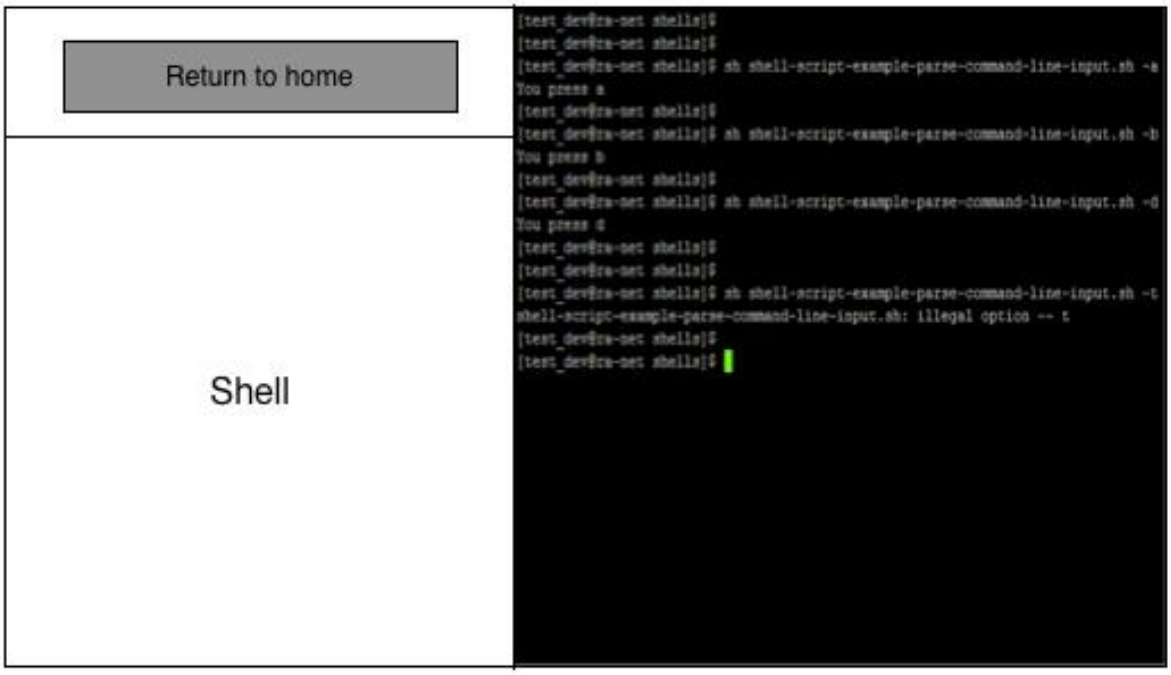


*Figure 4: Console*