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 de conception

Cours	Systèmes embarqués 2
Professeur	Daniel Gachet, <daniel.gachet@hefr.ch>
Etudiant	Loïc Gremaud, <loic.gremaud@edu.hefr.ch>
Classe	T-2a
Date	27.04.2014
Groupe	5

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

Table des matières

Rapport de conception

1. Introduction

Le but de ce rapport est définir la façon dont vont être implémentées les différentes fonctionnalités qui m'ont été attribuées pour ce projet.

Voici les points qui vont être traités dans ce rapport :

- Définition des interfaces (API)
- Design des différents modules
- Définitions de la stratégie de validation/vérification
- Définition des principes d'implémentation des différents modules et composantes
- Description des détails importants du logiciel

Les fonctionnalités qui m'ont été attribué sont les suivantes :

- Ecran tactile
- Interpréteur de commande
- Chronomètre
- Thermomètre

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

Membre	**Fonctionnalité**
Romain Froidevaux	- Affichage - Implémentation des commandes - Interfaces graphiques
David Rossier	- Contrôleur des applications - Protocole UDP/IP - Ping

1.1. Contexte

Nous devons développer une application en C pour une cible APF27.

Les fonctionnalités qui doivent obligatoirement être implémentées sont les suivantes :

- Affichage d'information sur l'écran LCD
- Lancement des applications via l'écran tactile
- Affichage en temps réel de la température sur les 7-segments côté FPGA
- Mis à disposition d'un chronomètre qui sera commandé avec les boutons poussoirs en mode interruption
- Mise à disposition de la commande « ping »
- Création d'un jeu multi-joueurs au travers du réseau
- Mis à disposition d'une shell pour lancer/stopper/lister les applications
- Mis en œuvre d'un noyau temps réel coopératif

Le jeu multijoueur qui va être développé est la bataille navale.

Nous pouvons également implémenter quelques fonctionnalités optionnelles, ceux-ci ne seront développés uniquement si l'avancement du projet le permettra. Voici ce que nous avons sélectionné :

- Comme deuxième jeu multi-joueurs, le Tictactoe
- Application permettant la gestion des scores
- Adaptation du noyau de coopératif vers préemptif

2. Ecran tactile

L'écran tactile est une partie important de notre logiciel car elle permet d'interagir avec les différents programmes et jeux.

2.1. Modules

Ci-dessous, un arbre représentant les dépendances entre les différents modules que nous utilisons pour la gestion de l'écran tactile.

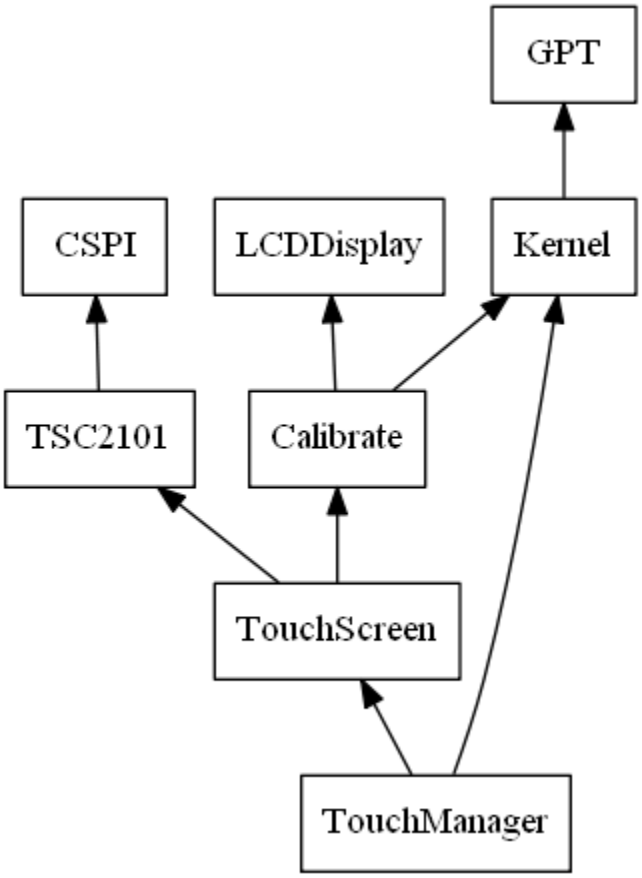

Figure 1: Dépendances des modules de l'écran tactile

2.1.1. Description des modules

Nom	Fichier	Description
TouchManager	touchmanager.h/c	API et application gérant les actions sur l'écran tactile
TouchScreen	touchscreen.h/c	Driver haut niveau gérant les actions sur l'écran tactile
TSC2101	tsc2101.h/c	Driver bas niveau pour relever la position sur l'écran tactile
Calibrate	calibrate.h/c	Programme et API permettant de calibrer l'écran tactile et d'effectuer les converions
LCDDisplay	lcddisplay.h/c	API gérant l'affichage sur l'écran LCD
Kernel	kernel.h/c	Noyau préemptif permettant l'utilisation de thread et de sémaphore
GPT	gpt.h/c	Driver permettant l'utilisation des timers

2.1.2. TouchManager

Dans ce module, il y aura un thread qui s'occupe de faire de la scrutation continue afin de détecter un événement sur l'écran tactile par le biais du driver haut niveau. Afin d'éviter des scrutations trop fréquentes, on temporisera grâce à une fonction du kernel à la fin de la fonction (environs 100 ms à tester).

Lorsqu'une pression a été détectée voici ce qui va se passer :

- Trouver la zone correspondante à la position en parcourant la liste de zones.
- Si une zone correspond, on appel la méthode qui se trouve dans la structure de la zone « event_handler » en lui passant la zone et la position (cette méthode correspond déjà au bon type de zone, bouton ou grille).
- La méthode se charge à son tour soit de trouver le boutons correspondant ou bien la case correspondante et appelle la routine de callback et en passant la case ou le bouton qui a été pressé avec le paramètre de callback.

2.1.3. TouchScreen

Ce module fournit des méthodes permettant de détecter si une pression a été exercée et de retourner une position déjà converti grâce au module Calibrate. Les différents problèmes de position seront traités par ce module. Voici les problèmes qui ont été rencontrés lors de la lecture de la position sur le driver bas niveau :

- Certaines positions apparaissent en dehors des limites de l'écran
- Certaines valeurs de pressions dépassent les limites du possible

2.1.4. TSC2101

Ce module est un driver bas niveau qui permet de récupérer la position sur l'écran tactile. Ce driver ne va normalement par être modifié et est déjà implémenté.

2.1.5. Calibrate

Ce module contient un petit programme de calibration permettant de recueillir des données pour pouvoir effectuer des conversions d'un point sur l'écran vers la position en pixel sur l'écran. Pour l'implémentation de ce module, nous allons nous inspirer des librairies libres tslib et/ou ginput de uGFX.

2.1.6. Kernel

Dans ce module en plus des fonctions de bases, il faudra une fonction permettant de temporiser la réactivation d'un thread. C'est-à-dire qu'un thread puisse rester dans l'état blocked pendant un certain temps et passé ce délai, repasse dans l'état waiting. L'utilisation du module GPT semble être une bonne solution. Une fonction sera charger de lancer le timer et une autre de réactiver le thread.

2.1.7. GPT

Se référer au chapitre traitant du chronomètre.

2.2. Définition des interfaces

Les définitions des interfaces sont disponibles en annexe dans la documentation doxygen.

2.3. Validation et vérification

Le fonctionnement de l'écran tactile paraît simple à tester, on pourrait dire que si les boutons pressés réagissent correctement l'écran tactile est fonctionnel. Mais il y a beaucoup de détails à prendre en compte, voici les quelques plus importants :

- Tous les boutons fonctionnent correctement lorsqu'ils sont pressés à différents endroit
- Il n'y a d'événements qui doivent se produire si aucune action n'est effectuée
- Toute la surface de l'écran doit être exploitable
- Il faut une bonne réactivité lorsqu'un bouton est pressé
- Le programme de calibration fonctionne correctement
- La fonction de conversion fonctionne correctement

Il est clair qu'il ne sera pas possible d'effectuer directement les premiers tests sur les applications. Il faudra y aller méthodiquement et tester chaque méthode avant d'aller plus loin.

3. Interpréteur de commandes

L'interpréteur de commandes permettra de lancer des commandes depuis un ordinateur par le biais d'une connexion serial. L'affichage s'effectuera à la fois sur la console de l'ordinateur mais également sur une console qui peut être affiché sur l'écran de la cible.

3.1. Modules

Ci-dessous, un arbre représentant les dépendances entre les différents modules que nous utilisons pour l'interpréteur de commandes.

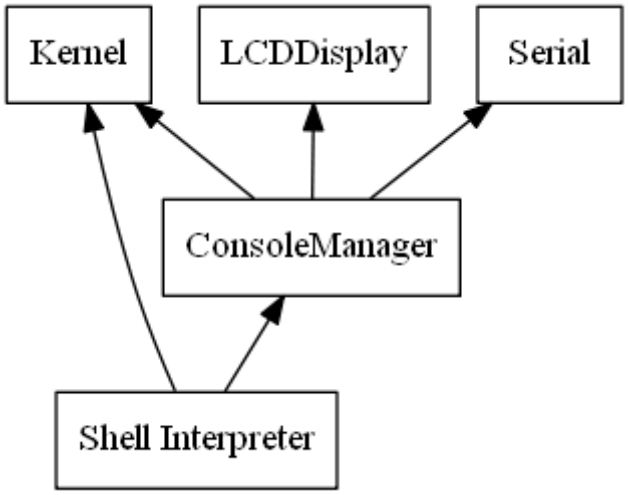

Figure 2: Dépendances des modules de l'interpréteur de commandes

3.1.1. Description des modules

Nom	Fichier	Description
Shell Interpreter	shell_interpreter.h/c	Interpréteur de commandes
ConsoleManager	consolemanager.h/c	API permettant de gérer la console
LCDDisplay	lcddisplay.h/c	API gérant l'affichage sur l'écran LCD
Serial	serial.h/c	Driver pour l'interface serial
Kernel	kernel.h/c	Noyau préemptif permettant l'utilisation de thread et de sémaphore

3.1.2. Shell Interpreter

Ce module contient un programme permettant de lire les entrées sur la console et permet de lier des programmes Shell. Un thread va être utilisé pour récupérer les commandes qui ont été entrée dans la console.

3.1.3. ConsoleManager

Ce module permet d'afficher les caractères entrés à la fois sur la console via l'interface serial et la console sur l'écran de la cible. Il permet également de récupérer une ligne entière qui a été entrée dans la console.

3.1.4. LCDDisplay

Ce module est implémenté par Romain Froidevaux. Il fournit une API permettant l'affichage de la console sur l'écran LCD de la cible.

3.1.5. Serial

Driver permettant de récupérer et de pousser des caractères sur l'interface série. Ce driver est déjà implémenté et pourrait être adapté afin d'avoir une meilleure gestion des threads grâce aux sémaphores (principe du producteur-consommateur).

3.1.6. Kernel

Ce module nous permettra l'utilisation de sémaphores.

3.2. Définition des interfaces

Les définitions des interfaces sont disponibles en annexe dans la documentation doxygen.

3.3. Validation et vérification

Il y aura beaucoup de chose à tester pour l'interpréteur de commande sans compter qu'il interagie avec des programmes Shell. Les points importants qu'il faudra vérifier sont les suivants :

- Les deux console (écran LCD + serial) contiennent les mêmes données
- Les caractères entrés sont reproduits correctement dans la console
- L'interprétation d'une ligne de commande s'effectue correctement
- Toutes les commandes fonctionnent correctement

4. Chronomètre

Un chronomètre sera disponible sur l'écran d'accueil et pourra continuer à fonctionner même si celui-ci n'est pas affiché.

4.1. Modules

Ci-dessous, un arbre représentant les dépendances entre les différents modules que nous utilisons pour le chronomètre.

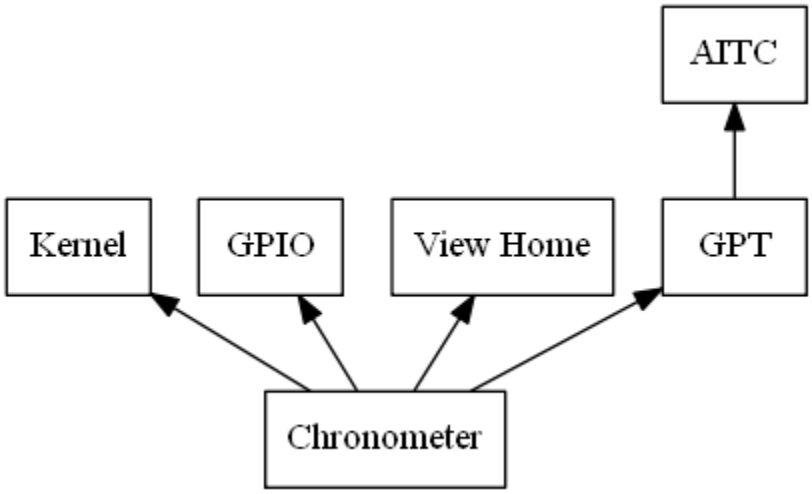

Figure 3: Dépendances des modules du chronomètre

4.1.1. Description des modules

Nom	Fichier	Description
Chronometer	chronometer.h/c	Application gérant le chronomètre
GPIO	gpio.h/c	Driver permettant l'utilisation des boutons en mode interruptif
View Home	view_home.h/c	API permettant l'affichage du chronomètre sur l'écran d'accueil
GPT	gpt.h/c	Driver permettant l'utilisation des timers
Kernel	kernel.h/c	Noyau préemptif permettant l'utilisation de thread et de sémaphore

4.1.2. Chronometer

Ci-dessous, un diagramme avec les différents états possible du chronomètre.

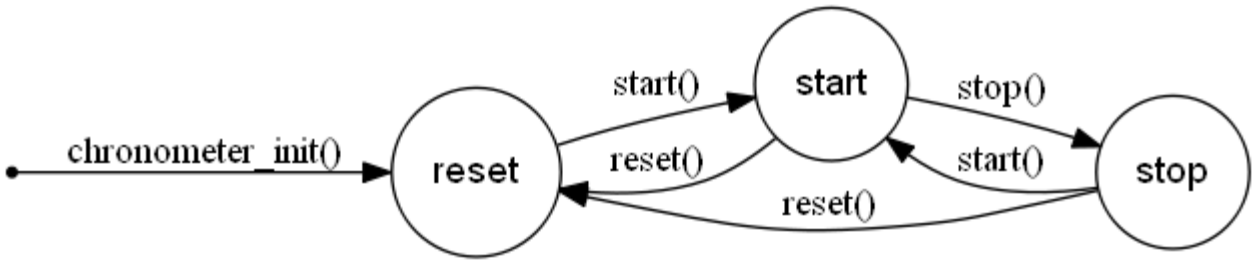

Figure 4: Diagramme d'état du chronomètre

Les fonctions start, stop et reset sont appelées lorsque les boutons correspondants sont pressés par le biais d'une interruption. Lors du passage à l'état reset, le chronomètre est arrêté et remis à zéro. Lors du passage à l'état start, le chronomètre est démarré à partir de sa valeur actuelle. Finalement lors du passage à l'état stop, le chronomètre est arrêté en gardant sa valeur actuelle.

Un thread se charge d'aller rafraîchir la valeur sur l'écran. Un timer GPT se chargera d'incrémenter la valeur du chronomètre.

4.1.3. GPIO

Ce module est déjà implémenté et ne nécessitera sans doute aucune modification.

4.1.4. View Home

Ce module est implémenté par Romain Froidevaux. Il fournit les méthodes permettant d'actualiser la valeur du chronomètre sur l'écran d'accueil.

4.1.5. GPT

Ce module permet de créer et d'utiliser les timers. Les timers sont configuré lors de leur création. Et il existe deux manières de les démarrer :

1. Continuous : le timer va générer des interruptions de manière périodique et ne s'arrêtera seulement lorsqu'on le stop. C'est cette méthode qui est utilisé par le chronomètre.
2. OneShot : le timer va générer qu'une seul interruption et va s'arrêter immédiatement.

Afin de facilité le partage des différents timers entre les applications qui en ont besoin, le module s'occupe de distribuer les timers libres à ces dernières.

4.1.6. Kernel

Ici, le module kernel nous permettra d'utiliser les sémaphores de manière à ne pas rafraîchir inutilement l'écran d'accueil.

4.2. Définition des interfaces

Les définitions des interfaces sont disponibles en annexe dans la documentation doxygen.

4.3. Validation et vérification

Le fonctionnement général du chronomètre est relativement simple à vérifier par contre il est plus difficile de vérifier si il restera précis dans le temps. Voici les points importants qu'il faudra valider :

- Les interactions avec les boutons fonctionnent correctement.
- L'affichage est correctement mis à jours.
- Le chronomètre fonctionnement toujours en arrière-plan.
- Les passages entre les unités de temps (s, min, h) s'effectuent correctement.

Remarque :

Comme l'affichage est rafraîchi séparément par un thread, il est possible que la mise à jour de l'affichage subisse de légers retards, surtout si les autres threads utilisent beaucoup de temps de calcul.

5. Thermomètre

La température sera affichée sur l'affichage 7-segments de la FPGA et pourra également affiché en même temps sur l'écran d'accueil.

5.1. Modules

Ci-dessous, un arbre représentant les dépendances entre les différents modules que nous utilisons pour le thermomètre.

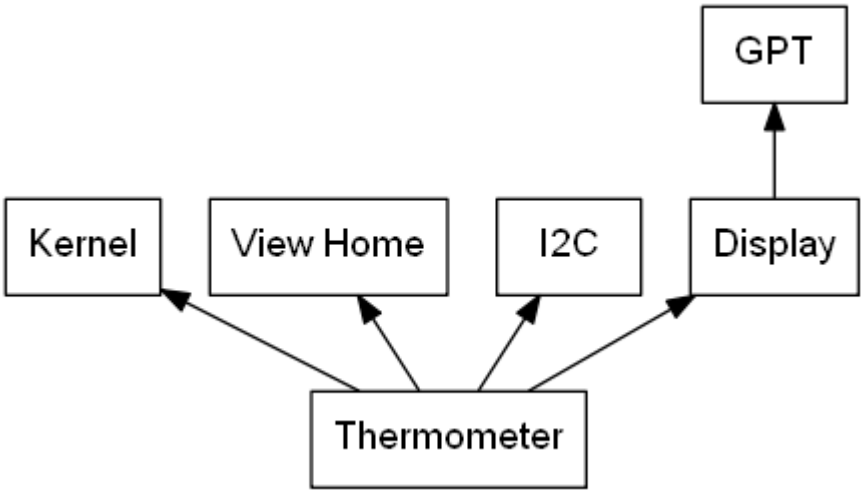

Figure 5: Dépendances des modules du thermomètre

5.1.1. Description des modules

Nom	Fichier	Description
Thermometer	thermometer.h/c	Application gérant le thermomètre
I2C	i2c.h/c	Driver permettant d'aller récupérer la valeur du thermomètre LM75
View Home	view_home.h/c	API permettant l'affichage du thermomètre sur l'écran d'accueil
Display	display.h/c	Programme permettant l'affichage sur le 7-segments du la FPGA
Kernel	kernel.h/c	Noyau préemptif permettant l'utilisation de thread et de sémaphore
GPT	gpt.h/c	Driver permettant l'utilisation des timers

5.1.2. Thermometer

Ce module, possède une fonction permettant de récupérer la température ainsi qu'un thread. Le thread se charge de récupérer la température et si celle-ci a changée, de mettre à jours l'affichage 7-segment et de l'écran d'accueil. Il faudra également temporiser l'exécution de ce thread afin de ne pas mettre à jours trop souvent les valeurs.

5.1.3. I2C

C'est ce module qui permet de communiquer avec thermomètre LM75 au travers d'un bus I2C. Ce driver est déjà implémenté et ne nécessitera pas de modification.

5.1.4. View Home

Ce module est implémenté par Romain Froidevaux. Il fournit les méthodes permettant d'actualiser la valeur du thermomètre sur l'écran d'accueil.

5.1.5. Display

Ce module est à la fois un driver et un programme qui permet d'afficher une valeur sur l'affichage 7-segments de la FPGA. Une routine sera chargée de mettre à jour l'affichage. Cette routine sera lancée périodiquement grâce à une interruption GPT et celle-ci mettra à jour une fois sur deux le digit de gauche et celui-de droite afin d'avoir la même intensité lumineuse (fréquence à tester).

5.1.6. Kernel

Ce module, nous permettra la création du thread et nous fournira la méthode pour le temporiser.

5.2. Définition des interfaces

Les définitions des interfaces sont disponibles en annexe dans la documentation doxygen.

5.3. Validation et vérification

Le thermomètre est relativement simple à vérifier. Voici les points importants qu'il faudra valider :

- La valeur est mise à jours correctement sur les deux affichages.
- La température varie en fonction de l'environnement.
- L'affichage 7-segment ne scintille pas et garde une intensité homogène.

6. Conclusion personnelle

C'est à cette étape que l'on remarque, qu'il difficile de définir à l'avance toutes les fonctions que l'on aura besoin pour nos différents modules. Il est aussi vrai que notre distribution des fonctions du logiciel au sein des membres du groupe n'a pas forcément été judicieuse. En effet, la partie affichage est étroitement lié avec la partie tactile de l'écran, de même pour la partie Shell Interpreter et Shell Program. Nous aurions pu également éviter les noms à rallonge pour les modules. La partie qui va être la plus dur à réaliser est celle de l'écran tactile.

Fribourg, le 27 avril 2014,

Loïc Gremaud

7. Annexes

- Documentation Doxygen des interfaces.