Systèmes Embarqués 2

Journal du TP.10 : Mini-Projet

Temps effectué hors des heures de classe

Nous avons effectué 70h de plus en dehors des périodes mises à disposition pour ce projet.

Fonctionnement

Ce mini-projet a pour but de synthétiser l'ensemble des connaissances que nous avons appris durant le cours de système embarqué 1 et 2. En plus des fonctionnalités de bases imposées, nous avons choisi d'implémenter en supplément une horloge analogique.

Synthèse des acquis

Acquis :

- Interfaçage C-Assembleur.
- Implémentation d'un kernel.
- Utilisation de thread.
- Création d'un terminal.

Acquis, mais à exercer encore :

- Utilisation avancée d'Éclipse.
- Makefile.

Fonctionnalités

Kernel

Le kernel a été créé selon le l'implémentation vue en classe. Nous avons utilisé l'exemple des corrigés des exercices pour bien en comprendre le fonctionnement. Par la suite, nous avons implémenté en plus une méthode pour mettre un thread en pause durant un certain temps.

Remarque : La mise en vieille de thread au moyen de la méthode *kernel_thread_sleep* est implémentée, mais nous ne l'utilisons pas, car notre système est largement assez rapide pour le nombre de threads actifs simultanément.

Horloge Analogique

L'horloge analogie est la fonctionnalité supplémentaire de notre projet. Elle est appelée via le menu de l'OS.

```
void analogClock_appli() {
    uint32_t timeToDisplay;
    uint32_t currentSysTime = 0;

    int hourToSub = 0;
    double hour = 0;
    double min = 0;
    double sec = 0;
    double angleHour;
    double angleMin;
    double angleSec;
    int clockSize = 47;

    while (1) {
        if ((getSysTime() / 1000) != currentSysTime) {
            drawHand(CENTER, CENTER, (int) (CENTER + SecHandLenght * cos(angleSec)),(int) (CENTER
    + SecHandLenght * sin(angleSec)), BLACK);
            drawHand(CENTER, CENTER, (int) (CENTER + MinHandLenght * cos(angleMin)),(int) (CENTER
    + MinHandLenght * sin(angleMin)), BLACK);
            drawHand(CENTER, CENTER, (int) (CENTER + HourHandLength * cos(angleHour)),(int) (
    CENTER + HourHandLength * sin(angleHour)), BLACK);

            currentSysTime = getSysTime() / 1000;
            timeToDisplay = currentSysTime + startedHour * 3600 + startedMinute * 60 +
    startedSecond;

            hourToSub = timeToDisplay / 3600;
            hour = ((timeToDisplay / 3600 % 24) % 12) + 3;
            angleHour = (PI / 6) * (3 - hour);
            drawHand(CENTER, CENTER, (int) (CENTER + HourHandLength * cos(angleHour)),(int) (
    CENTER + HourHandLength * sin(angleHour)), WHITE);

            min = ((timeToDisplay/60) %60)+30;
            angleMin = (PI / 30) * (30-min);
            drawHand(CENTER, CENTER, (int) (CENTER + MinHandLenght * cos(angleMin)),(int) (CENTER
    + MinHandLenght * sin(angleMin)), WHITE);

            sec = ((timeToDisplay - hourToSub * 3600 - min * 60)) / 5 + 3;
            angleSec = (PI / 6) * (3 - sec);
            drawHand(CENTER, CENTER, (int) (CENTER + SecHandLenght * cos(angleSec)),(int) (CENTER
    + SecHandLenght * sin(angleSec)), WHITE);
        }

        for (double var = 0; var < PI / 2; var += 0.01) {     // Display circle.
            int x = CENTER + (int) (clockSize * cos(var));
            int y = CENTER + (int) (clockSize * sin(var));
            // print
            setPixelOn(x, y, WHITE);
            x = CENTER - (int) (clockSize * cos(var));
            // print
            setPixelOn(x, y, WHITE);
            y = CENTER - (int) (clockSize * sin(var));
            x = CENTER - (int) (clockSize * cos(var));
            //print
            setPixelOn(x, y, WHITE);
            x = CENTER + (int) (clockSize * sin(var));
            y = CENTER - (int) (clockSize * cos(var));
            // print
            setPixelOn(x, y, WHITE);
        }

        if (quitAppli) {
            kernel_thread_exit();
            newApp = true;
            quitAppli = false;
        }
        kernel_thread_yield();
    }
}
```

Listing 1 – Affiche d'une horloge analogique

Durant sa réalisation, nous avons passé pas mal de temps à réfléchir comment transformer un nombre d'heures ou minutes en un angle afin d'afficher une ligne. En dehors de ceci, l'implémentation de cette fonction s'est bien passée même si elle nous a pris beaucoup de temps.

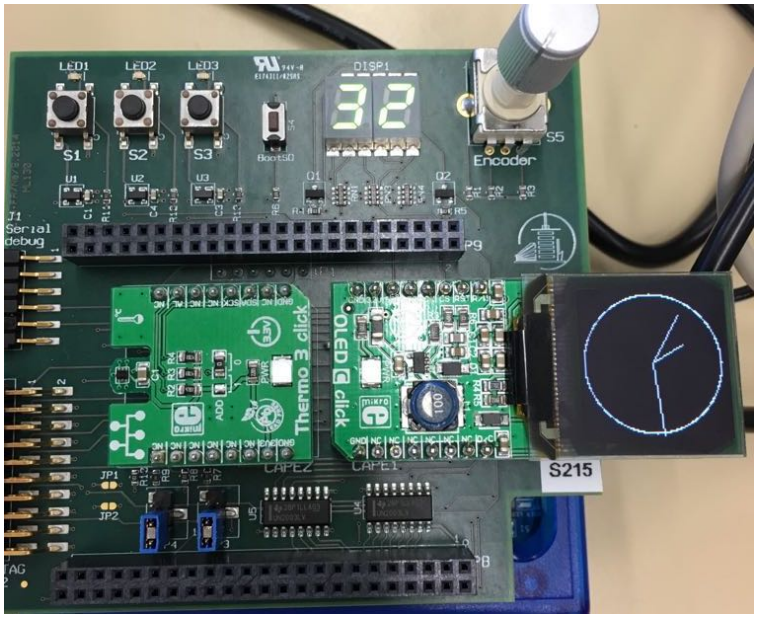

FIGURE 1 – Horloge analogique

Chronomètre

En reprenant ce qui a été fait pour la gestion de l'heure, nous avons facilement pu réaliser cette fonction du système d'exploitation. Les timers et fonctions d'affichage étaient en effet déjà tous présents. De ce fait, nous n'avons pas rencontré de problème dans cette partie.

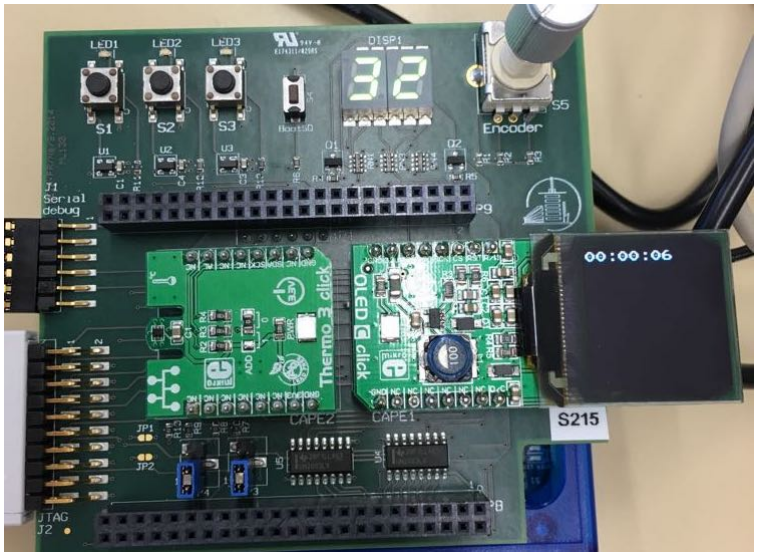

FIGURE 2 – Chronomètre

S1 permet de démarrer le chronomètre, S2 de le stopper et S3 de le reset.

Timer

Le timer fait l'inverse du chronomètre. Il permet de régler un temps, puis de décompter jusqu'à 0. Lorsque le temps arrive à 0, le timer s'arrête. L'encodeur rotatif permet de régler le temps, les boutons S1 et S2 de changer entre le réglage des heures, minutes et secondes et le bouton S3 de lancer le timer. Lorsqu'il est lancé, S2 stop le timer, S1 le relance et S3 et le reset.

Shell

Le shell est la dernière fonctionnalité que nous avons implémentée. De ce fait, elle fut relativement simple, car nous avions déjà codé pratiquement tous ce que nous avions besoin, à savoir :

— Récupérer du text en console
— Les applications/fonctions appelées par le shell sont déjà fonctionnelles.

Ainsi, nous avons lancé pour chaque fonctionnalité la même fonction que si elle était démarrée au moyen des boutons. Le shell est initialisé et disponible dès que le système est initialisé (date et heure comprises).

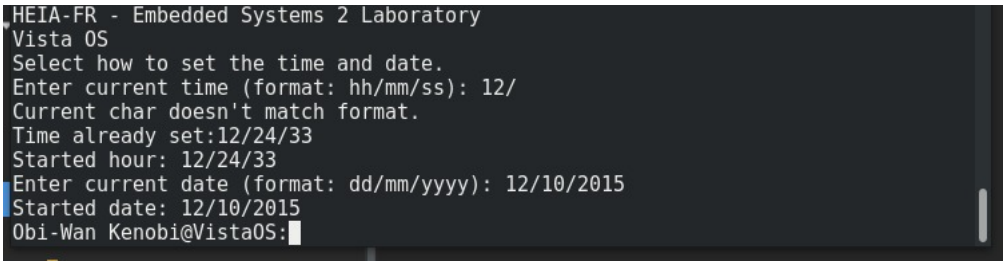

FIGURE 3 – Shell

La commande help permet de montrer toutes les commandes disponibles :

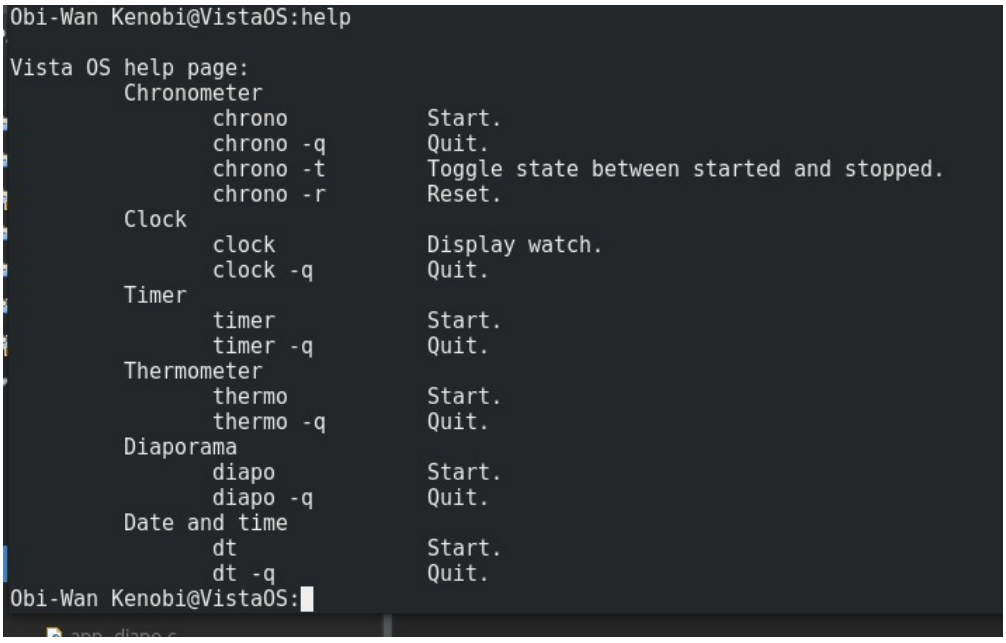

FIGURE 4 – Command Help

Chacune de ces commandes peut ensuite être lancée à n'importe quel moment.

Diaporama

Une fois la fonction permettant d'afficher des images implémentée, cette tâche ne fut pas très compliquée. Nous avons simplement trouvé des images, les avons convertis en 96x96, soit la résolution de l'écran puis avons transformé l'image en un fichier C grâce au logiciel Gimp.

Thermomètre

Le thermomètre avait déjà été implémenté dans un ancien TP. Nous avons de ce fait simplement récupéré nos anciens codes et les avons adaptés pour qu'il fonctionne dans un thread. De plus, nous avons aussi inclus la possibilité d'afficher la température sur l'écran.

Feedback

L'implémentation d'un mini OS fut très intéressante. D'une part, il s'agit d'un grand projet avec de nombreuses fonctionnalités et non d'une petite application qui n'utilise que quelques ressources du Beagle Bone. De plus, faire son propre système d'exploitation en quelque sorte est un projet qui utilise presque tout ce que l'on a appris dans le cours de Système Embarqué. Cela nous a permis de consolider nos connaissances et de les mettre en application au travers d'un cas concret.
Néanmoins, il s'est avéré que l'implémentation était plus longue et complexe que ce que l'on avait pensé dans un premier temps. La fin d'année étant très chargée, avec deux autres projets en plus des tests, nous n'avons pas eu le temps de faire les fonctionnalités optionnelles. Il serait d'ailleurs bien à ce sujet de mieux répartir les projets de semestres sur l'année entre les cours de Réseau IP, Système Embarqué et Système Numérique. Ils sont en effet tous planifiés pour le deuxième semestre alors que nous n'en avons aucun au premier.
En fin de compte, ce projet était, de notre point de vue, autant intéressant que concret.

Fribourg, le 11 juin 2017

Samuel Riedo

Pascal Roulin