POLYTECH LILLE

Informatique, Microélectronique et Automatique

Rapport de projet de 4ème année

Malette Arduino éducative

YUQIAN HU

2015/05/10

Ⅰ. La première partie: l'ordonnanceur pour le TP système

1 Présentation

La première partie de ce projet consiste à concevoir l'ordonnanceur pour le Shield "TP système" conçu et réalisé dans un autre projet. Il consiste à réaliser un ordonnanceur de tâches de type Round Robin. Pour simplifier le développement et mieux appréhender le concept de tâche, cet ordonnanceur fonctionnera sur une plate-forme Arduino. Elle est architecturée autour d'un micro-contrôleur de la famille AVR d'Atmel (atmega328p).

2 Étape à réaliser dans ce projet:

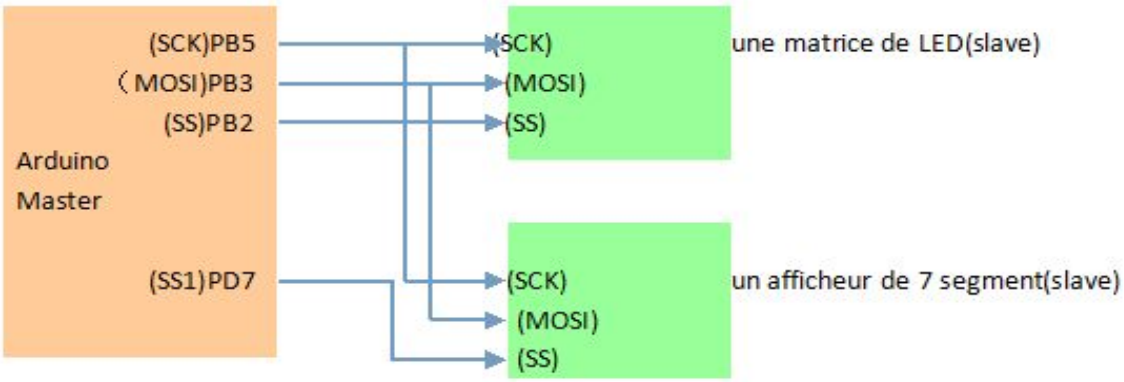
2.1 Réaliser sept tâches respectivement

- Tâche sur le bus SPI

-tâche 1: envoyer en double différents colonnes de couleurs sur une matrice de LED communiquent via le bus SPI.

-tâche 2: envoyer en boucle des chiffres sur un afficheur de 7 segments communiquent via le bus SPI.

Le figure ci-dessous montre la principe de la fonctionnement pour le bus SPI.



Pour utiliser le bus SPI, il faut utiliser les fonctions ci-dessous:

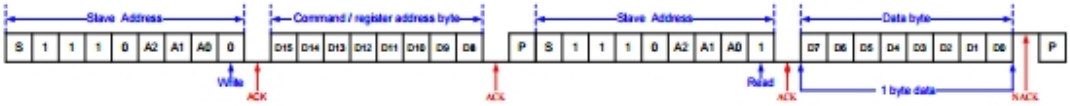
```
// Initialisation of SPI bus
void spi_init(void)
{
}

// Send byte on SPI bus
uint8_t spi_exch(uint8_t output)
{
}
```

- Tâche sur le bus I2C

-tâche 1: envoyer en bouble des dessins sur une matrice monocolore via le bus I2C.

Le paquet de la communication I2C se compose de plusieurs parties.

- le signal START
- l'adresse de paquet
- data de paquet
- le signal STOP



Reading Single Data Bytes from the HT16K33

Pour utiliser le bus i2c, il faut les fonctions suivantes:

```
// Write i2c message (with a single data byte)
uint8_t TWI_writedata(uint8_t address,uint8_t com_add,uint8_t data)
{
        TWI_start();// Send start signal
        if (TWIGetStatus() != 0x08)// Get i2c bus status
                return ERROR;
        TWI_writebyte(address<<1);// Write single byte to bus
        if (TWIGetStatus() != 0x18)// Get i2c bus status
                return ERROR;
        TWI_writebyte(com_add);// Write i2c minimal message (command)
        if (TWIGetStatus() != 0x28)// Get i2c bus status
                return ERROR;
        TWI_writebyte(data);
        if (TWIGetStatus() != 0x28)// Get i2c bus status
                return ERROR;
        TWI_stop();// Send stop signal
        return SUCCESS;
}
```

- Tâches sur le bus sérial

- tâche1: envoyer en boucle le caractère '!'
- tâche 2: envoyer en boucle le message "Bienvenue chez Polytech'Lille"

Pour utiliser le port sérial , il faut les fonctions suivantes:

```
void serial_init(int speed)
{
}

void serial_send(unsigned char c)
{
}
```

2.2 Réaliser des sémaphores

Comme les deux tâches sont sur le port série et le bus SPI, il est nécessaire de gérer l'accès concurrent au port entre deux tâches. Il faut donc mettre en place un mécanisme de sémaphores permettant d'assurer l'accès exculsif au port série et le bus SPI.

```
void P(int s)
{
}

void V(int s)
{
}
```

La fonction P() permet de vérifier si le port série ou le bus SPI est disponible, si oui le prend, si non la tâche l'exécutant est suspendue. La fonction V() sert à rendre le port série ou le bus SPI et, si besoin, rend active la tâche suspendue.

2.3 Créer des bibliothèque

Afin de rendre le programme plus propre, j'ai créé une bibliothèque pour le timer, une pour le bus SPI, une pour le bus I2C, une pour le port série et une pour les sémaphores.

2.4 Concevoir l'ordonnanceur

J'ai réalisé un ordonnanceur à partir des ressources fournies par le professeur. Cette ordonnanceur utilisait l'algorithme de Round Robin avec une intervalle de 20ms. Il fallait que l'Arduino ordonnance six tâches.

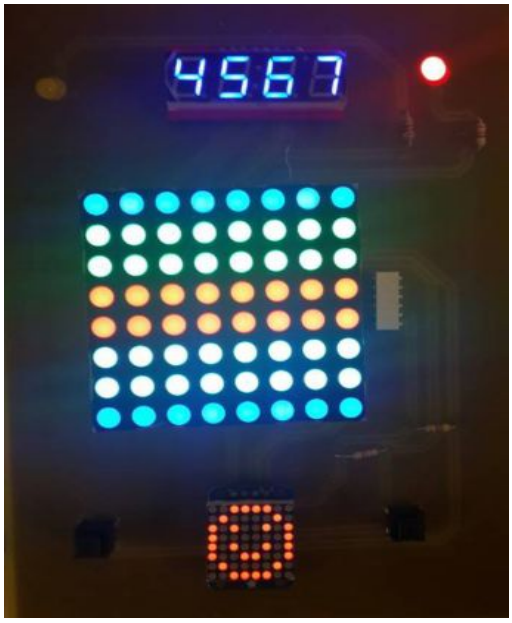
Ici, il a été nécessaire d'utiliser un système d'exploitation temps réel "avr(RTOS)" afin d' exécuter les tâches simultanément. Comme un ordinateur ne peut exécuter qu'une tâche à la fois, pour que les tâches soit exécutées de manière simultanée, il faut le noyau commute rapidement entre le contexte d'exécution des tâches. Et ceci, de manière à ce qu'une seule tâche soit exécutée à la fois mais que globalement toutes les tâches soient exécutées. Il faut donc exécuter les étapes suivantes listées ci-dessous.

- Commutation de contexte et élection de la nouvelle tâche
- Chargement par le noyau du contexte de la tâche .
- Exécution des instructions de la tâche pendant 20 millisecondes
- Sauvegarde du contexte de la tâche

Les deux fonctions SAVE_CONTEXT() et RESTORE_CONTEXT() permettent de réaliser restaurer et sauvegarder les contexte. Pour la gestion de la mémoire des différents processus, nous allons découper l'espace mémoire. Voici les adresses pour les sept tâches:

task_led_red,0x0500
task_serial_plil,0x0550
task_serial_bang,0x0600
task_spi_matrix,0x0700
task_spi_display,0x0750
task_i2c_matrix,0x0800

Voici la carte qui ordonnance tous les tâches:



Ⅱ.La deuxième partie :Carte : PROGRAM AND PLAY

1 Présentation

La seconde partie est à destination d'enfants pour l'initiation à la programmation sur Arduino.

1.Présentation du language Scratch

Scratch est un nouveau langage de programmation qui facilite la création de simulations interactives, de dessins animés, de jeux, de compositions musicales, de simulations numériques ainsi que leurs partages sur le Web. Il est conçu par le MIT afin de permettre aux enfants de faire l'apprentissage de la programmation facilement.

Il a 3 caractéristiques:

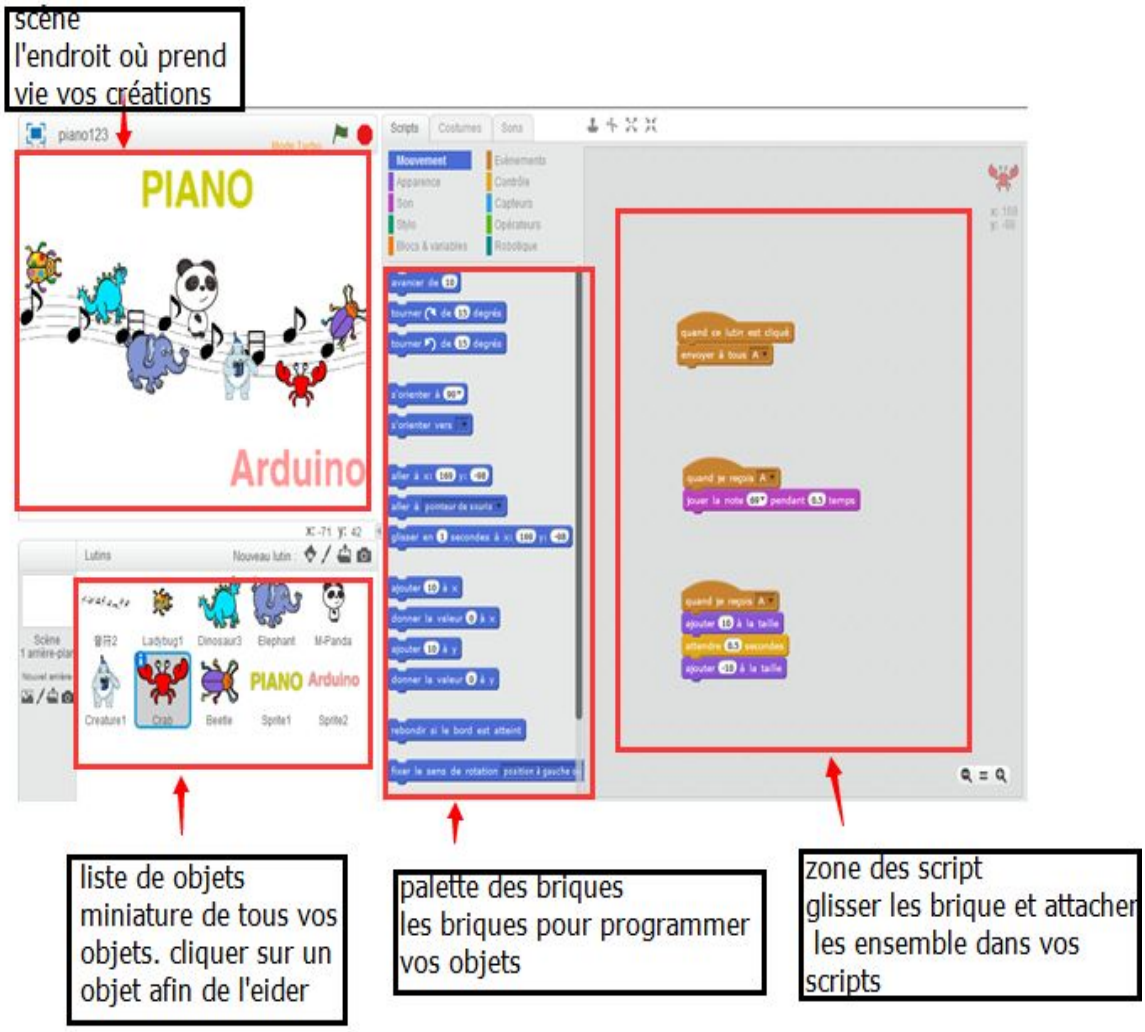
- dynamique : il permet de modifier le code du programme en cours d'exécution.
- visuel, tout le code est directement inscrit sous forme de briques en couleurs.
- partagé, Scratch permet de rendre utilisable en ligne les projets réalisés, sur la site de scratch, il y a déjà plus de 8 millions de projets.

2.Présentation de mblock

Pour ce projet, j'ai utilisé mblock qui est une extension de Scratch qui peut etre utiliser pour arduino. Il ajoute des module électronique dans le Scratch original, avec ces modules , les utilisateurs peuvent lire des capteurs, commander des moteurs et même un robot.

le lien pour la téléchargement: http://mblock.cc/download/

voici un résumé de l'interface de scratch:



2 Travail à effectuer

Pour cette partie, dans un premier temps, j'ai réalisé quelques programmes et petits jeux qui sont planifient par Mehmet sur Arduino. Et après j'ai les transformé sur mblock. Il y a deux cartes dans cette partie:

- Carte : INITIATION ARDUINO

Cette carte est constituée d'un thermocouple, d'un phototransistor, de LEDs, de boutons et d'un afficheur.

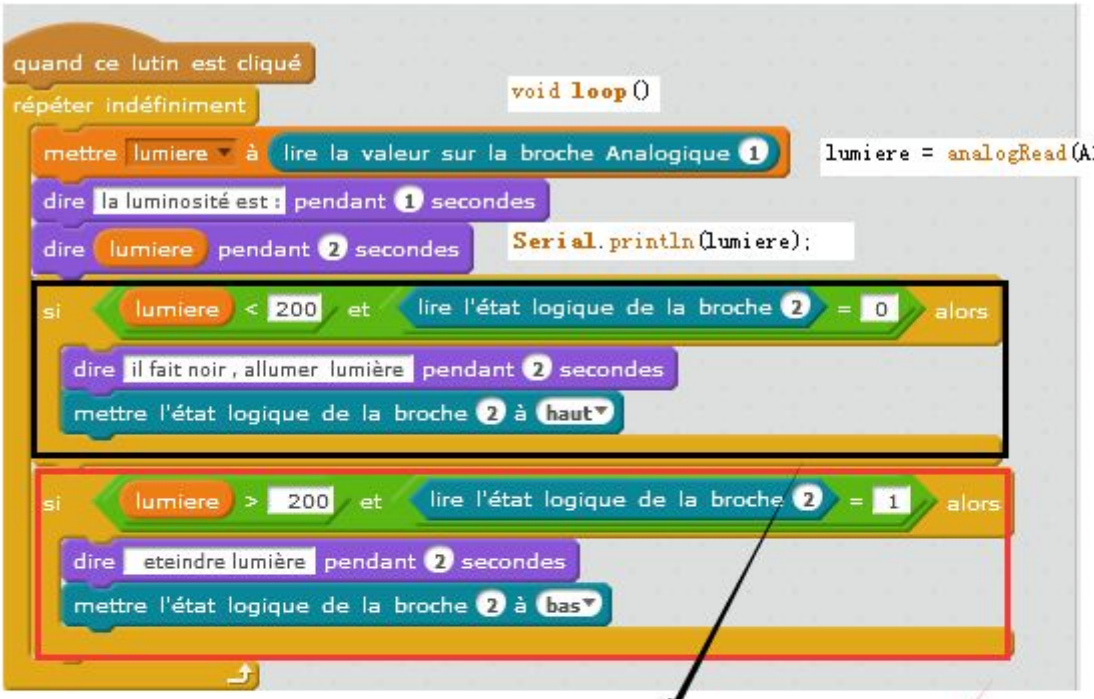
Il faudra coder chaque composant individuellement puis faire interagir les composants :
-led qui s'allume quand on appuie
-led allumé ou éteinte en fonction de la luminosité
-affichage de la température, luminosité

- Carte : PROGRAMM AND PLAY

-le piano : La programmation de ce dernier sera relié à chaque bouton et un son (fréquence, amplitude) sera joué par le haut-parleur.
-Le tape-taupe : une des six LED s'allumera pendant un certain temps pendant lequel il faudra appuyer sur le bouton correspondant. Un signal sonore indiquera si « le point a été marqué »
-répète après-moi : Il s'agira de programmer une séquence d'allumage de LED de plus en plus difficile à suivre. Le joueur essaiera de refaire la séquence. Deux bips sonores différents permettront de savoir si c'est un échec ou une réussite.

Quand j'ai programmé les jeux de tape-taupe et répéte après moi, j'ai trouvé qu'ils étaient compliqués pour les enfant, peut-être ils seraient plus utiles pour les lycéens.

Voici un des programme pour faire led allumée ou éteinte en fonction de la luminosité. Dans un premier temps, j'ai programmé sur scratch(le détail des opérations est dans la documentation). Après, j'ai écrit le programme correspondant à scratch sur arduino pour faire apprendre la programmation de Arduino aux enfants.




```
if (lumiere < 200 && ledState == 0) {
  Serial.print("il fait noir , allumer  lumière ");
  digitalWrite(ledPin, HIGH); // allumer le LED

}
```

```
if (lumiere > 200 && ledState == 1) {
   Serial.print("eteindre  lumière ");
  digitalWrite(ledPin, LOW); // éteindre le LED
}
```

III.Conclusion:

Pendant ce projet, j'ai rencontré beaucoup de difficultés, j'ai fini le projet grâce à l'aide de M.Redon. Pour la première partie, c'était difficile de comprendre le système RTOS, en même temps, c'était la premiere fois que j'écris un projet en C, c'est vraiment significatif pour moi. Pour la deuxième partie, au début, ce n'était pas très clair ce que je devais faire. j'ai seulement utilisé beaucoup de temps pour la programmation et la débogage des jeux. à la fin, j'ai compris qu'il faut apprendre aux enfants la programmation, c'est un peu dommage que je n'ai pas eu assez de temps pour transformer tous les jeux sur scratch.

Mais pour moi, cette expérience m'a beaucoup apporté. j'ai appris un nouveau language Scratch, la programmation sur arduino, j'ai bien compris l'ordonnanceur et un système d'exploitation temps réel . De plus, pendant le projet, j'ai connu la 'maker culture' comme j'ai du communiquer souvent avec les amateurs de Arduino sur Internet. Comme ça m'interesse beaucoup cette culture, je pense que je vais devenir une Maker et continuer à faire des projets aussi intéressants.

Référence :

TP système http://vantroys.polytech-lille.net/enseignement/IMA4_OS/tp_scheduler/
I2c http://www.embedds.com/programming-avr-i2c-interface/
Scratch http://fr.wikipedia.org/wiki/Scratch_%28langage%29
Ordonnancementdans les systèmes d'exploitation
http://fr.wikipedia.org/wiki/Ordonnancement_dans_les_syst%C3%A8mes_d%27exploitation
Guide de référence pour scratch http://scratchfr.free.fr/k1n8g7/ScratchRefGuidefrv14A4.pdf