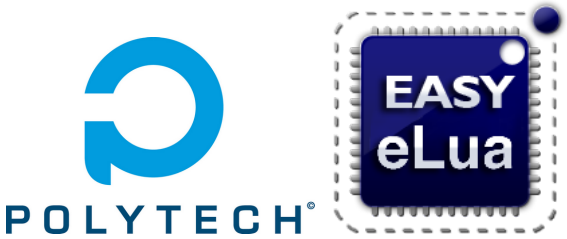

Elizabeth PAZ
Salem HARRACHE

Easy eLua

eLua et approche Arduino sur STM32F4-DISCOVERY

Projet Innovant, Avril 2012

RICM 4A

Table des matières

Contents i

List of Figures ii

List of Tables iii

Introduction 1

Approche Arduino 1

1 Approche Arduino 3

1.1 Qu'est-ce qu'Arduino ? 3

1.2 Avantages de l'utilisation de Arduino 4

1.3 Utilisation d'Arduino 4

2 elua sur STM32F4-DISCOVERY 7

2.1 Qu'est-ce que Lua ? 7

2.2 Qu'est-ce qu'eLua ? 7

2.2.1 Architecture de eLua 8

2.2.2 Code en commun 9

2.3 Avantages de eLua 9

3 Designer l'esprit Arduino en LUA 11

4 Avacement du projet 13

5 Difficultés 15

6 Demo **17**

7 Conclusion **19**

Bibliographie **20**

Appendix **21**

Table des figures

1.1 Logo Arduino 3
1.2 Carte Arduino Uno 4
1.3 Environnement de travail : choix de la carte et du port 5

2.1 Logo Lua 7
2.2 Structure logique de eLua 8

Liste des tableaux

1.1 Création d'une nouvelle fonction en Arduino 5
1.2 Exemple Blink pour Arduino 5

Introduction

Présentation de Lua

Présentation de Arduino

Présentation de carte STM32F4-DISCOVERY

Chapitre 1

Approche Arduino

1.1 Qu'est-ce qu'Arduino ?

Figure 1.1 – Logo Arduino

Le système Arduino [1] est une plateforme *open-source* d'électronique programmée basée sur une carte à microcontrôleur de la famille AVR [2], et sur un environnement de développement intégré qui permet d'écrire, compiler et transférer un programme vers la carte. Ce logiciel utilise la technique du Processing/Wiring [3].

Avec le système Arduino on peut réaliser divers tâches, par exemple le développement des objets interactifs indépendants : le prototypage rapide. Aussi, la domotique, qui grâce aux différents interrupteurs/capteurs permet de contrôler plusieurs sorties matérielles : contrôle des appareils domestiques, pilotage de robot, etc ...ou même charger des batteries par l'analyse et la production des signaux éléctriques. De plus, il peut communiquer avec des logiciels tournant sur l'ordinateur tel que Macromedia Flash, Processing, MaxMSP, etc ....

Le langage de programmation Arduino est une implémentation de Wiring, une plateforme de développement similaire, qui est basée sur l'environnement multimédia de programmation Processing.

Les cartes électroniques sont accesibles a tous, elles peuvent être achetées préassemblées ou être fabriquées manuellement, tout en ayant la totalité des informations nécessaire à l'assemblage.

1. Le projet Arduino a reçu un titre honorifique à l'Ars Electronica 2006, dans la catégorie Digital Communities
2. Cœur du processeur et famille de microcontrôleurs
3. Processing est un langage de programmation et un environnement de développement

1.2 Avantages de l'utilisation de Arduino

Le système Arduino a simplifié la façon de travailler avec les microcontrôleurs, en offrant plusieurs avantages pour les enseignants, les étudiants et les amateurs intéressés par d'autres systèmes. Ce système prend en charge des détails compliqués de la programmation des microcontrôleurs et les intègrent dans une présentation facile à utiliser. Voici, plusieurs avantages qui propose Arduino :

Peu coûteux : En comparaison a d'autres plateformes les cartes Arduino sont peu coûteuses, les moins chères sont les versions qui peuvent être assemblées à la main.

Multi-plateforme : Le logiciel de programmation des modules Arduino est une application Java, libre et qui peut être tourné dans plusieurs systèmes d'exploitation tel que Linux, Windows et Mac.

Environnement de programmation clair et simple : Le logiciel est facile à utiliser pour les débutants (nous même l'ayant utiliser suite au cours d'initiation à l'Arduino) ; de plus, il reste flexible pour des utilisateurs avancés.

Logiciel Open Source et extensible : Le logiciel et le langage Arduino sont publiés sous licence Open Source qui est donc disponible à tous, ce qui permet donc la possibilité d'être complété et amélioré par des programmateurs plus experimentés. Le langage Arduino, qui utilise le langage C++, peut être étendu grâce à l'aide des librairies du C++.

Matériel Open Source et extensible : La version sur plaque d'essai de la carte Arduino peut être acheté à très bas coût et peut être réalisé par tout utilisateur, elle a pour but comprendre comment la carte fonctionne. De plus, tous les schèmas de modules Arduino sont publiés alors les utilisateur plus experimentés en circuits peuvent apporter des améliorations à leur cartes.

1.3 Utilisation d'Arduino

Dans cette partie on va décrire certains points importants pour l'utilisation du logiciel, de la carte et du langage.

Figure 1.2 – Carte Arduino Uno

Un programme en langage Arduino doit obligatoirement être composé de ces deux fonctions :
– la fonction d'initialisation *setup*() qui est exécutée une seule fois au démarrage.
– la fonction "boucle sans fin" loop() qui est exécutée en boucle une fois que la fonction setup() a été exécutée une fois.
Puis, si besoin, autres fonctions peuvent être crées en suivant ce schèma :

```
type nom_fonction (arguments) {
   // ici le code de la fonction
}
```

Table 1.1 – Création d'une nouvelle fonction en Arduino

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, ...
      repeatedly.
*/
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);    // set the LED on
  delay(1000);               // wait for a second
  digitalWrite(13, LOW);     // set the LED off
  delay(1000);               // wait for a second
}
```

Table 1.2 – Exemple Blink pour Arduino

L'utilisation du logiciel est facile et très clair, ce qui est très important a rétenir c'est de bien vérifier avant d'envoyer le programme à la carte, si on utilise le bon port et la bonne carte.

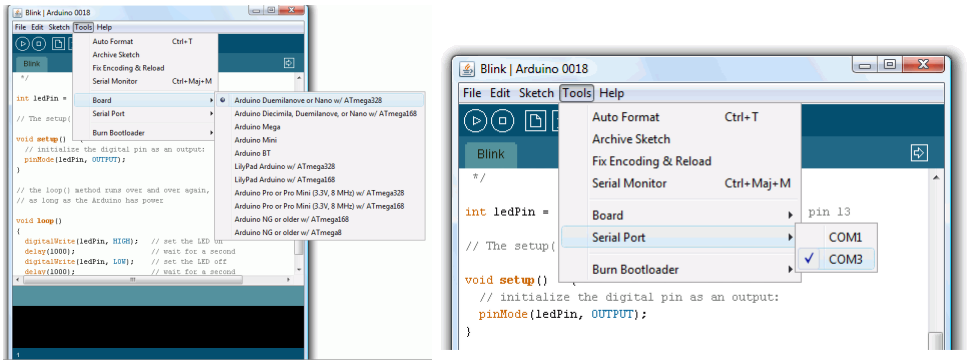

Figure 1.3 – Environnement de travail : choix de la carte et du port

Chapitre 2

elua sur STM32F4-DISCOVERY

2.1 Qu'est-ce que Lua ?

Lua est un langage de script libre, réflexif et impératif. Il a été conçu afin de pouvoir être embarqué au sein d'autres applications et les étendre Lua (qui signifie lune en portugais) a été développé par des membres du groupe de recherche TeCGraf, de l'université de Rio de Janeiro au Brésil. Il est écrit en langage C ANSI strict, et grâce a cela il est compilable sur une grande variété de systèmes ; le plus souvent est utilisé dans des systèmes embarqués, dont sa compacité est très appréciée, de plus, il possède la compatibilité du langage Cpour s'intégrer facilement dans la plupart des projets. Il profite de la compatibilité que possède le langage C avec un grand nombre de langages pour s'intégrer facilement dans la plupart des projets. Lua a déjà été utilisé pour le développement de jeux vidéo, entre eux, par exemple l'interface du jeu World of Warcraft de Blizzard Entertainment, SimCity4, entre autres.

Figure 2.1 – Logo Lua

2.2 Qu'est-ce qu'eLua ?

elua adopte le langage de programmation Lua pour faire une complète implémentation dans le monde de l'embarqué, elua ajoute des caractéristiques spécifiques pour une

efficacité, portabilité et développement de logiciels embarqués. eLua propose la totalité des caractéristiques de la version de bureau de Lua, et c'est important de remarqué que elua utilise les mécanismes de base pour pouvoir l'étendre avec des fonctionnalités de développement de l'embarqué optimisés et spécifiques.

2.2.1 Architecture de eLua

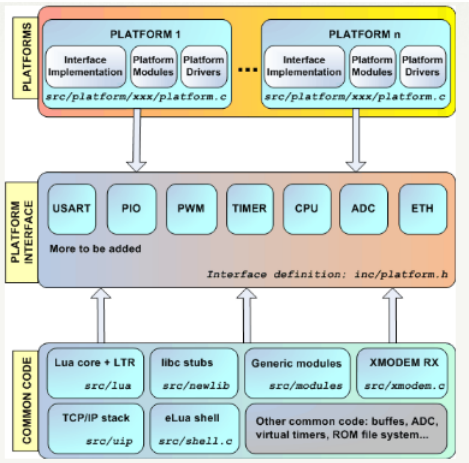

Figure 2.2 – Structure logique de eLua

Dans la documentation de eLua on utilise la notion de "plateforme" pour designer un groupe de CPU qui partagent la même structure du noyau ; pourtant ils peuvent différer dans le nombre de péripheriques integrés, la mémoire interne, et toute autres attributs. Un port eLua implemente un ou plusieurs CPU d'une même plateforme. Dans la figure **??**, on observe que eLua essaie d'être aussi portable que possible dans différents plateformes, pour cela plusieurs règles ont été mises en place, entre elles :

- Le code qui est indépendant du type de plateforme doit être écrit en ANSI C dans toutes les parties ou cela est possible, afin qu'il soit portable parmi plusieurs architectures et compilateurs, comme Lua.
- Le code qui ne peut pas être générique (par exemple le péripheriques et le code spécifique au CPU) doit être fait aussi portable que possible en utilisant une interface en commun qui doit être implementé par toutes les plateformes dans lequelles eLua peut être executé. On va l'appeler "interface de plateforme" (à complèter avec la documentation).
- Toutes les plateformes et leur péripheriques ne sont pas crées de la même façon et possèdent donc des fonctionnalités très variés. Pour accèder a une fonctionnalité spécifique à une plateforme on peut utiliser un module (voir dans la documentation). Ces modules ont été crées afin de complèter l'écart entre l'interface de la plateforme et toutes les caractéristiques proposés par la plateforme. toutes les fonct

2.2.2 Code en commun

La liste suivante montre quelques éléments qui sont classifiés comme du code en commun :

- Le code Lua
- Tous les composants eLua (par exemple le ROM file system, le shell eLua, et autres)
- Tous les modules génériques, qui sont des modules exportés de Lua
- Le code générique des péripheriques

C'est important de remarquer que les parties génériques doivent être la plupart du code. Par exemple, lorsqu'on veut ajouter un nouveau fichier du système celui-ci doit être un code générique, sinon le code aura des dependances par rapport où il reside. On peut corriger cela en utilisant des fonctions définies dans l'interface de la plateforme, mais si cela n'est pas possible il faudra séparer les fonctions spécifiques dans une interface separé qui va devoir être implementé par toutes les plateformes qui veulent utiliser ce nouveau fichier. Ceci donne un maximum de portabilité au code.

L'utilisateur ne doit jamais oublier le but principal de eLua : la flexibilité. Il doit donc être dans la capacité de savoir quelles composants font partie de son binaire eLua, de même pour les modules, il doit savoir quels modules il a besoin. L'utilisateur est demandé de faire leur code pour différents scénarios possible, afin de promouvoir la portabilité.

2.3 Avantages de eLua

Contrôle total de la plateforme : il n'existe pas un systèmee d'exploitation entre les programmes et le microcontrôleur.

Portabilité du code : Comme Lua, le programme peut être executée dans un grand groupe varié de plateformes et architectures.

Facilité de transformation : Le code et le design des produits pour eLua peuvent être conçu indépendament du matériel, ainsi toute changement ou amélioration dans le future peut être fait facilement et gagner du temps.

Développement d'objectifs : Lua est complètement fonctionnel avec la possibilité d'avoir un shell dans le microcontrôleur, il n'y a pas besoin de rien installer côté ordinateur a part la connexion du port ou ethernet. Les programmes sont utilisable directement dans les plateformes.

Flexibilité des produits : L'utilisation de ce langage script de très haut niveau dans un projet rend celui-ci très adaptable, facilement reprogrammable et reconfigurable. Les systèmes sont très efficients pour une future évolution.

Embarqué RAD : Prototype et experimenté dans un modèle "Rapid Aplication Develop". Les idées peuvent être testé directement dans les plateformes avec les kits de développement ; l'utilisateur n'a pas besoin des simulateurs ou des futures modification du code pour le rendre adaptable.

Les kits sont prêts a être utilisés : Grand nombre des logiciels open source et plateformes sont utilisables.

Long cycle de vie : Add user configuration and scripting capabilities to your projects, making them adaptable to the always changing contexts of industrial processes, evolving engineering, automation standards, field optimizations etc...

Apprendre l'embarqué : L'experimentation est très simple et interactive. L'utilisateur est invité a utiliser ses compètences en programmation pour devenir un développeur des systèmes embarqués rapidement et de façon amusante.

Open Source : elua est libre, gratuit, et open source logiciel, comme Lua, elle possède une licence MIT qui permet l'utilisation de eLua dans des codes "closed" source. Il n'y a aucune permission a demander, ils demandent juste de faire circuler l'information au monde : on utilise eLua !

Chapitre 3

Designer l'esprit Arduino en LUA

Chapitre 4

Avacement du projet

Chapitre 5

Difficultés

Chapitre 6

Demo

Chapitre 7

Conclusion

Bibliographie