



Projet innovant RICM4: Easy-eLua

Elizabeth Paz Salem Harrache

Polytech'Grenoble
Olivier RICHARD
Didier DONSEZ

27 Avril 2012

Sommaire

1 Introduction

- Présentation carte STM32F4-DISCOVERY
- Présentation Arduino
- Présentation eLua

2 Travail réalisé

- Organisation du travail
- Arborescence du projet
- Fonctions portées
- Nouveaux concepts

3 Demonstration

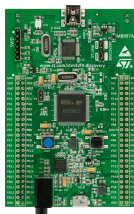
- "Hello Word!"
- "Blink with button"
- "Ascii table"

4 Conclusion

Introduction : Présentation carte STM32F4-DISCOVERY

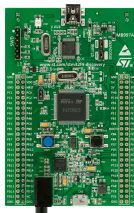
Partie Elizabeth

Note salem : présenter brièvement les caracteristiques et puis finir sur le fait que c'est compliqué de programmer dessus pour un novice



Introduction : Présentation Arduino

Partie Elizabeth Note salem : Le site est bien fait, puis y a des presentation sur internet de la carte donc easy. Dire qu'il y a deux methode a implementer (loop et setup) et puis un point sur l'open source. Meme les carte en elle meme sont open source, ce qui est vraiment appreciable pour tout ingenieurs qui veut se lancer dans un nouveau projet etc...



Introduction : Présentation eLua

partie Elizabeth

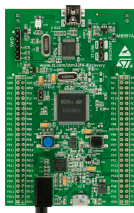
<http://www.eluaproject.net/overview> + architecture :

[http:](http://www.eluaproject.net/doc/v0.8/en_arch_overview.html)

[//www.eluaproject.net/doc/v0.8/en_arch_overview.html](http://www.eluaproject.net/doc/v0.8/en_arch_overview.html)

la utilise le schéma et dis que notre mission c'était d'évaluer le portage, et dire qu'on a contacté ton James (xD) et qu'on a réussi à le faire travailler pour nous xD

C'est important je pense de le mentionner :)



Organisation du travail : Agilité et Autodidacte

① Phase I

- Premiers sprints assez long et peu productif
 - Recherche d'information sur eLua et d'éventuelles portages pour STM32
 - Evaluation de la faisabilité
 - Initiation à la programmation avec la lib stlink sur linux
- Concertation hebdomadaire sur l'avancement
- Discussions avec notre tuteur de stage

② Phase II

- Sprints très courts
- Conception de l'architecture du projet
- Développement d'outils de travail (installation, flash)
- Utilisation d'un gestionnaire de version

Arborescence du projet



arduino_wrapper



docs



elua



env



examples



activate_env.sh



AUTHORS



flash.sh



install.sh



LICENSE



README.rst



run_shell.sh



send.sh

Fonctions portées

① Entrées/Sorties numériques

- `pinMode()` → déclarer les broches en entrée ou en sortie
- `digitalWrite()` → écriture d'une valeur HIGH/LOW (1/0)
- `digitalRead()` → lecture

② Communication Série

- `Serial::begin()` → initialiser la connexion serie
- `Serial::read()`
- `Serial::write()`
- `Serial::print()`

③ Time

- `millis()` → Durée d'execution du programme
- `micros()`
- `delay()` → Attente passive
- `delayMicroseconds()`

Nouveaux concepts

- 1 Programmation orientée objects
- 2 Lua et la métaprogrammation → Redéfinition du type "Class"
- 3 Introduction de l'objet App qui s'exécute avec un contexte

```
App = Class:new()
function App:setup()
    -- The setup function will only run once after each
    -- powerup or reset of the board
end
function App:loop()
    -- loops consecutively
end

function App:run()
    self:setup()
    while condition do
        self:loop()
    end
end
end
```

Exemple : "Blink" (Lua)

```
require("arduino_wrapper")

function App:setup()
    self.ledpin = ORANGE_LED
    pinMode(self.ledpin, OUTPUT)
end

function App:loop()
    digitalWrite(self.ledpin, HIGH)
    delay(1000)
    digitalWrite(self.ledpin, LOW)
    delay(1000)
end

app = App:new("Blink led")
app:run()
```



Exemple : "Blink" (Arduino)

```
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  
}
```

Demonstration : "Hello Word!"

```
require("arduino_wrapper")  
app = App:new(" Hello Word!")  
app:run()
```

Demonstration : “Blink with button” avec flash

```
require("arduino_wrapper")

function App:setup()
    self.ledpin = RED_LED
    pinMode(self.ledpin, OUTPUT)

    self.blink = false
    self:blink_toggle()
end

function App:loop()
    if self:btn_pressed() then
        self.blink = not self.blink
        self:blink_toggle()
    end
    delay(10)
end

function App:blink_toggle()
    [...]
end
```

Demonstration : Lancement de script sans flash

```
require("arduino_wrapper")

function App:setup()
    self.byte = 33
end

function App:loop()
    self:println()
    self:write(self.byte)
    self:print(" , dec: " .. self.byte)
    self:print(" , hex: "), self:print(self.byte, HEX)
    self:print(" , oct: "), self:print(self.byte, OCT)
    self:print(" , bin: "), self:print(self.byte, BIN)

    self.byte = self.byte + 1
    delay(1000)
end

app = App:new("ASCII Table ~ Character Map")
app:run()
```



Conclusion

Couplé à la puissance d'eLua, Easy-eLua permet :

- Débuter dans la programmation pour l'embarqué.
- Portabilité : Le code Lua produit est compatible avec différentes architectures supportant elua.
- Le RAD pour l'embarqué: Prototyper et expérimenter des applications rapidement. Testez vos idées directement sans besoin de simulations ou de futures modifications.
- Flexibilité : Lua, langage de programmation de haut niveau, permet toute sorte d'utilisation.

Conclusion

Des questions ?