Projet innovant RICM4: Easy-eLua

Elizabeth Paz Salem Harrache

Polytech'Grenoble
Olivier Richard
Didier Donsez

27 Avril 2012

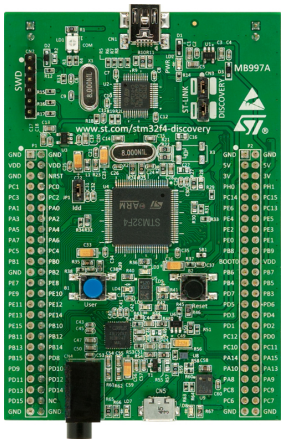
Sommaire

POLYTECH

Introduction : Présentation carte STM32F4-DISCOVERY

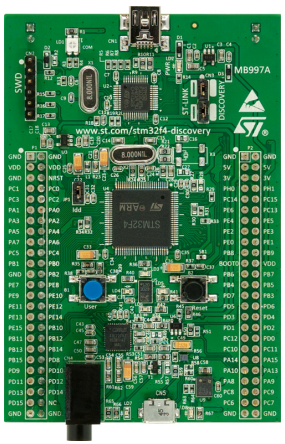
Partie Elizabeth

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



Introduction : Présentation Arduino

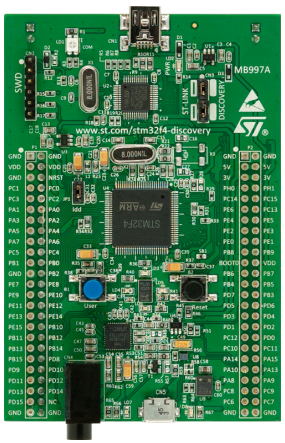
Partie Elizabeth Note salem : Le site est bien fait, puis y a des presentation sur internet de la carte donc easy. Dire qu'il y a deux methode a implementer (loop et setup) et puis un point sur l'open source. Meme les carte en elle meme sont open source, ce qui est vraiment appreciable pour tout ingenieurs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



POLYTECH

Introduction : Présentation eLua

partie Elizabeth
`http://www.eluaproject.net/overview` + architecture : `http://www.eluaproject.net/doc/v0.8/en_arch_overview.html`
la utilise le schéma et dis que notre mission c'etait d'evaluer le portage, et dire qu'on a contacté ton James (xD) et qu'on a reussis a le faire travailler pour nous xD
C'est important je pense de le mentionner :)



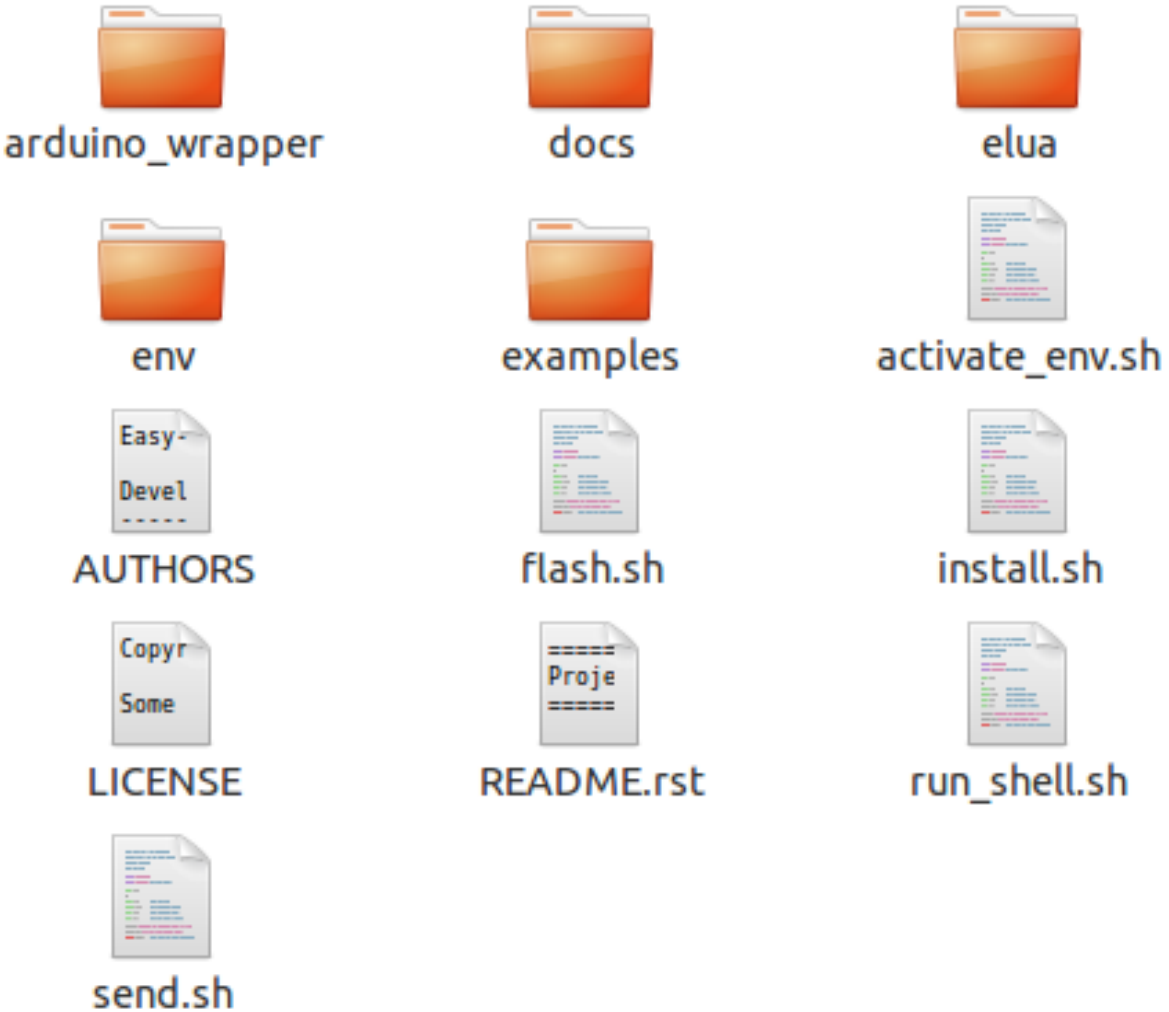
POLYTECH

Organisation du travail : Agilité et Autodidacte

1. Phase I
   - Premiers sprints assez long et peu productif
     - Recherche d'information sur eLua et d'eventuelles portages pour STM32
     - Evaluation de la faisabilité
     - Initiation à la prgramation avec la lib stlink sur linux
   - Concertation hebdomdaire sur l'avancement
   - Discussions avec notre tuteur de stage
2. Phase II
   - Sprints très courts
   - Conception de l'architecure du projet
   - Developpement d'outils de travail (instalation, flash)
   - Utilisation d'un gestionnaire de version

POLYTECH

Arboresence du projet



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. Commincation 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()

POLYTECH

Nouveaux concepts

1. Programmation orienteé objects
2. Lua et la métaprogrammation → Redéfinition du type “Class“
3. Introduction de l’objet App qui s’execute 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
```

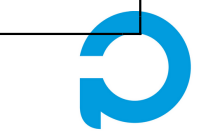
Exemple : "Blink" (Lua)

```
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
}
```

POLYTECH

Demonstration : “Hello Word!”

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

Demonstration : “Blink with button” avec flash

```
require("arduino_wraper")

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 --  !  ASCII character
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 :

- 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.
- Flexible products: Add modern high level script-language capabilities to your projects, resulting in highly adaptable, field-programable and reconfigurable designs. Efficient (and cheap!) future evolution to your systems.

POLYTECH

Conclusion

Des questions ?