Projet innovant RICM4: Easy-eLua

Elizabeth Paz Salem Harrache

Polytech'Grenoble
Olivier Richard
Didier Donsez

27 Avril 2012

POLYTECH

Sommaire

POLYTECH

Introduction : Présentation Arduino

Système Arduino: plateforme open-source de programmation embarquée, basée sur une carte à microcontrôleur de la famille AVR.

Arduino permet de réaliser: du prototypage rapide, la domotique, communication avec des logiciels, etc . . .

Principales avantages: peu coûteux, multi-plateforme, le logiciel et le matériel sont open source et extensibles, etc . . .

Utilisation: deux méthodes à implementer *obligatoirement*: loop() et setup().

POLYTECH

Introduction : Présentation eLua

Lua est un langage de script libre, réflexif et impératif. But: pouvoir être embarqué au sein d'applications et les étendre.

eLua: adopte le langage de programmation Lua et propose une implémentation complète de celui-ci pour les systèmes embarqués.

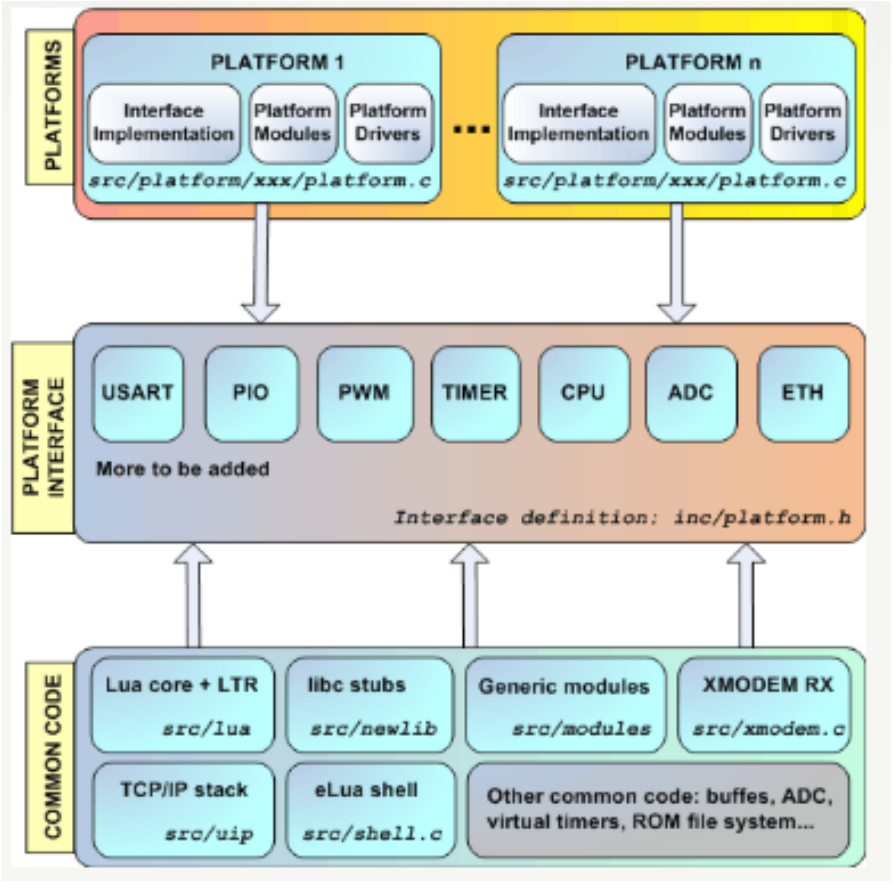
POLYTECH

Introduction : Présentation eLua

Principales avantages de l'utilisation de eLua: un contrôle absolu des plateformes, portabilité du code, développement autonome, flexibilité, open source, etc ...

Architecture de eLua: composé de façon a être le plus portable possible en suivant de près certaines de règles et ayant toujours du code qui sera commun a toutes les plateformes.

Introduction : Présentation eLua



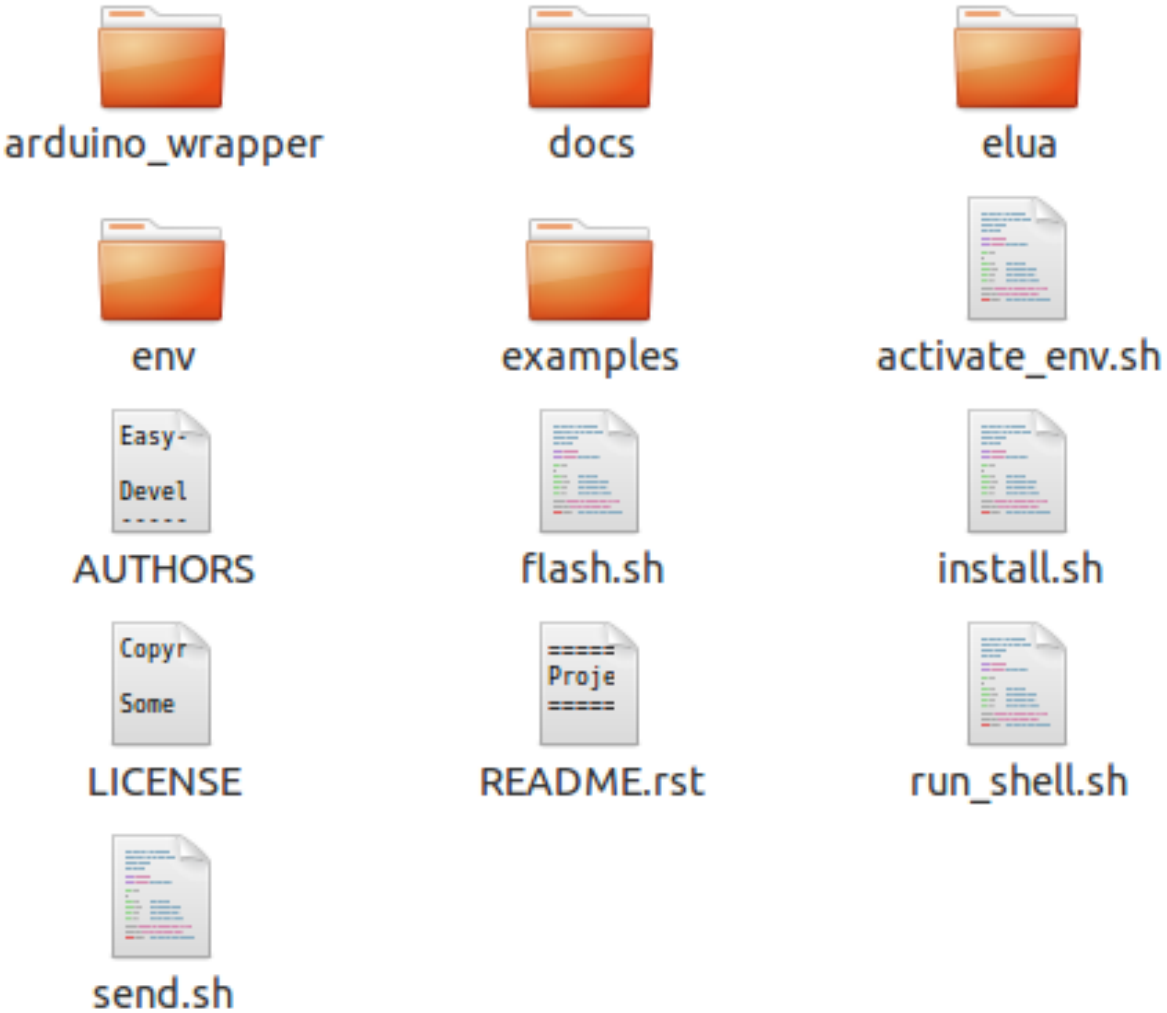
POLYTECH

Organisation du travail : Agilité et Autodidacte

1. Phase I
   - Premiers sprints assez longs et peu productifs
     - Recherche d'informations sur eLua et d'éventuels 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'architecure du projet
   - Développement d'outils de travail (installation, flash)
   - Utilisation d'un gestionnaire de version

POLYTECH

Arboresence du projet



Fonctions portées (1)

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 série
   - Serial::read()
   - Serial::write()
   - Serial::print()
3. Time
   - millis() → Durée d'exécution du programme
   - micros()
   - delay() → Attente passive
   - delayMicroseconds()

POLYTECH

Fonctions portées (2)

Arduino	**Easy-eLua**
Serial::begin() →	SerialPort:begin()
Serial::available() →	SerialPort:readWait() SerialPort:read()
Serial::read() →	SerialPort:read()
Serial::write() →	SerialPort:write()
Serial::print() →	SerialPort:print()
Serial::println() →	SerialPort:println()
Serial::end() →	
Serial::flush() →	

Nouveaux concepts

1. Programmation orienteé 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
```

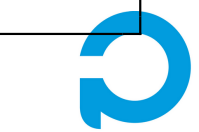
Exemple : "Blink" (Lua)

```
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
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 ?