PERSEPEEUR
RÉUNION 7
PROJECT PROGRESS OF CUPCARBON

DARWISH MOLHAM

02-03/05/2017
1

**CupCarbon U-One 3.1**
What’s new?

- New **ergonomic and professional design and graphical user interface** based on the new framework **JavaFX**

- The principal **actions** can be cancelled and reconsidered (**Undo/Redo**)

- **Studied and fine drawing quality** with pleasant visual objects for an easy use of the **environment**

- Very easy **script language** based on the **SenScript**, which includes intuitive commands

- Possibility to take into account the **topology** of a city (**buildings**) as well as the **radio propagation** and **visibility** in this environment
What’s new?

- The interferences of signals (alpha-stable distribution) at PHY layer for the Acknowledgment messages.

- Possibility to consider many radio modules and standards (802.15.4, Wifi and LoRa) in the same sensor node.

- Intelligent mobility which can be based on the sensed information

- Possibility to consider the clock drift

- The development of CupCarbon Script which is used to perform the main actions of CupCarbon in a Client /Server architecture by a set of command lines
2

TASKS PERFORMED
Tasks performed

The Interference and ACK has been improved
Tasks performed

- The Interference and ACK has been improved
- Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node
Tasks performed

The Interference and ACK has been improved

Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node

Radio modules can be changed either in the environment or during simulation (in SenScript)
Tasks performed

The Interference and ACK has been improved

Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node

Radio modules can be changed either in the environment or during simulation (in SenScript)
Tasks performed

- The Interference and ACK has been improved
- Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node
- Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1
CupCarbon 3.1
Tasks performed

- The Interference and ACK has been improved
- Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node
- Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1 using JavaFX
- Develop CupCarbon Script
Tasks performed

The Interference and ACK has been improved

Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node

Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1 using JavaFX

Develop CupCarbon Script

This script used to perform the main actions in a Client / Server model.
Tasks performed

The Interference and ACK has been improved

Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node

Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1 using JavaFX

Develop CupCarbon Script

This script used to perform the main actions in a Client / Server model.

These actions are using set of commands, which are executed by a console on the client side.
Tasks performed

- The Interference and ACK has been improved

- Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node

- Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1 using JavaFX

- Develop CupCarbon Script

  - This script used to perform the main actions in a Client / Server model.

  - These actions are using set of commands, which are executed by a console on the client side.

  - The server side is responsible for running the Graphical User Interface GUI, the CupCarbon platform.
Tasks performed

- The Interference and ACK has been improved
- Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node
- Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1 using JavaFX
- Develop CupCarbon Script

This script used to perform the main actions in a Client / Server model.

These actions are using set of commands, which are executed by a console on the client side.

The server side is responsible for running the Graphical User Interface GUI, the CupCarbon platform.

It is responsible to execute the commands and visualize the executed commands on the GUI
This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

This is free software, and you are welcome to redistribute it under certain conditions; see <http://www.gnu.org/licenses/>.

This project is supported by the French Agence Nationale de la Recherche ANR
REF : ANR-14-CE24-0017-01

CupCarbon >> Waiting
CupCarbon >> Connection
Molham.darwish@univ-brest.fr

Last login: Thu Apr 20 09:09:50 on console
[CupCarbon:~ ahcenebourneur$ cd Desktop/client/
[CupCarbon:client ahcenebourneur$ java CupClient
sCupCarbon >> project new testfor1 /Users/ahcenebourneur/Desktop/
> project new testfor1 /Users/ahcenebourneur/Desktop/
sCupCarbon >> add stdsensor -4.486016035079956 48.39052932411496 0.0
> add stdsensor -4.486016035079956 48.39052932411496 0.0
sCupCarbon >>
```
Last login: Thu Apr 20 09:09:50 on console
CupCarbon:~ ahcenebounceur$ cd Desktop/client/
CupCarbon:client ahcenebounceur$ java CupClient
scupCarbon >> project new testfor1 /Users/ahcenebounceur/Desktop/
> project new testfor1 /Users/ahcenebounceur/Desktop/
scupCarbon >> add stdsensor -4.486016035079956 48.39052932411496 0.0
> add stdsensor -4.486016035079956 48.39052932411496 0.0
scupCarbon >>
```
Molham.darwish@univ-brest.fr

CupCarbon >> Waiting
CupCarbon >> Connection
CupCarbon >> project new testfor1 /Users/ahcenebounceur/Desktop/
000 New project is created
CupCarbon >> add stdsensor -4.486016035079956 48.39052932411496 0.0
000 Addina Sensor: Id=1. name=S1
Tasks performed

The Interference and ACK has been improved

Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node

Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1 using JavaFX

Develop CupCarbon Script

Applying the Undo / Redo on the performed actions

CupCarbon Documentation
Tasks performed

The Interference and ACK has been improved

Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node

Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1 using JavaFX

Develop CupCarbon Script

Applying the Undo / Redo on the performed actions

CupCarbon Documentation

CupCarbon User Guide (Online)
Tasks performed

The Interference and ACK has been improved

Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node

Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1 using JavaFX

Develop CupCarbon Script

Applying the Undo / Redo on the performed actions

CupCarbon Documentation

CupCarbon User Guide (Online)

SenScript Reference
Tasks performed

- The Interference and ACK has been improved

- Apply many radio modules and standards (802.15.4, Wi-Fi and LoRa) in the same sensor node

- Develop and improve the version 2.9.1 of the platform (software & GUI) to the version 3.1 using JavaFX

- Develop CupCarbon Script

- Applying the Undo / Redo on the performed actions

- CupCarbon Documentation
  - CupCarbon User Guide (Online)
  - SenScript Reference
  - CupCarbon script reference
3

Task To Be Performed
Code Generation
3.1

**THE PROBLEM**
The need of the code generation

- *SenScript* allows to program and to configure each sensor node individually

- In CupCarbon, this step is not enough to ensure that we reach the main simulator objectives represented by:
  - The design, visualization, debugging and validation of distributed algorithms for monitoring, environmental data collection;
  - The creation of environmental scenarios such as fires, gas, mobiles…

- It is important to perform the code generation towards hardware platforms
  - Which allow to configure the hardware platforms
  - In order to validate and execute the applied algorithms in a real manner
SenScript → Arduino

Sensor node Script

Machine
3.2 SenScRipt
SenScript

```plaintext
1. loop
2. dreadsensor x
3. println $x
4. if($x==1)
5. send 1 1
6. delay 1000
7. send 0 1
8. delay 1000
9. end
10. delay 500
```
SenScript

Low level language

```plaintext
1 loop
dreadsensor x
println $x
if($x==1)
  send 1 1
delay 1000
send 0 1
delay 1000
end
delay 500
```
SenScript

Low level language

Command 1
Command 2
...
Command m

Similar to Assembly language

```
loop
dreadsensor x
println $x
if($x==1)
  send 1 1
  delay 1000
  send 0 1
  delay 1000
end
delay 500
```
3.3

ARDUINO
Arduino

```c
#include <XBee.h>
#include <LiquidCrystal.h>

XBee xbee = XBee();
Rx64Response rx = Rx64Response();
Tx64Request tx;
XBeeAddress64 addr;
uint8_t sdata[30];
uint8_t* rdata;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
    pinMode(13, OUTPUT);

    delay(100);
    Serial.begin(38400);
    xbee.setSerial(Serial);
    delay(100);
    lcd.begin(16, 2);
    lcd.setCursor(0, 0);

    first = -2;
    step = 1;
    // ----
    marked = 1;
    digitalWrite(13, 1-(1-'1'));
    for(int i=0; i<30; i++) {
```
Arduino

C code language

Control the microcontroller of an arduino module

```c
#include <XBee.h>
#include <LiquidCrystal.h>

XBee xbee = XBee();
Rx64Response rx = Rx64Response();
Tx64Request tx;
XBeeAddress64 addr;
uint8_t sdata[30];
uint8_t* rdata;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
    pinMode(13, OUTPUT);

    delay(100);
    Serial.begin(38400);
    xbee.setSerial(Serial);
    delay(100);
    lcd.begin(16, 2);
    lcd.setCursor(0, 0);

    first = -2;
    step = 1;
    // ----
    marked = 1;
    digitalWrite(13, 1-(1-'1'));
    for(int i=0; i<30; i++) {
```
Arduino

```c
#include <XBee.h>
#include <LiquidCrystal.h>

XBee xbee = XBee();
Rx64Response rx = Rx64Response();
Tx64Request tx;
XBeeAddress64 addr;
uint8_t sdata[30];
uint8_t* rdata;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
  pinMode(13, OUTPUT);

  delay(100);
  Serial.begin(38400);
  xbee.setSerial(Serial);
  delay(100);
  lcd.begin(16, 2);
  lcd.setCursor(0,0);

  first = -2;
  step = 1;
  // ----
  marked = 1;
  digitalWrite(13, 1-(1-'1'-'1'));
  for(int i=0; i<30; i++) {
```
Arduino

Blink

```c
// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
    delay(1000);  // wait for a second
    digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
    delay(1000);  // wait for a second
}
```

**The structure**

- **Fairly simple**
- **Runs in at least two parts**
The structure

- Fairly simple
- Runs in at least two parts
The structure
- Fairly simple
- Runs in at least two parts

The preparation

```c
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}
```

The Execution

```c
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
    delay(1000);  // wait for a second
    digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
    delay(1000);  // wait for a second
}
```
// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
    delay(1000);                        // wait for a second
    digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);                        // wait for a second
}

Arduino

Manipulates the parameters of module

Send Receive (read)
void loop() {
  xbee.readPacket();
  if (xbee.getResponse().isAvailable()) {
    if (xbee.getResponse().getApiId() == RX_64_RESPONSE || xbee.getResponse().getApiId() == RX_16_RESPONSE) {
      if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
        xbee.getResponse().getRx16Response(rx16);
        rdata = rx16.getData();
      }
      if (xbee.getResponse().getApiId() == RX_64_RESPONSE) {
        xbee.getResponse().getRx64Response(rx64);
        rdata = rx64.getData();
      }
      String x = "";
      for (int i=0; i<8; i++) {
        x += (char) rdatal[i];
      }
      delay(1000);
      digitalWrite(13, 1-(i=='1' ? x.charAt(0))); //
      for (int i=0; i<8; i++) {
        sdata[i] = x.charAt(i);
      }
      addr = XBeeAddress64(0x13A280, 0x40B58284);
      tx = Tx64Request(addr, sdata, sizeof(sdata));
      xbee.send(tx);
    }
  }
}
void loop() {
    xbee.readPacket();
    if (xbee.getResponse().isAvailable()) {
        if (xbee.getResponse().getApiId() == RX_64_RESPONSE || xbee.getResponse().getApiId() == RX_16_RESPONSE) {
            if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
                xbee.getResponse().getRx16Response(rx16);
                rdata = rx16.getData();
            } else if (xbee.getResponse().getApiId() == RX_64_RESPONSE) {
                xbee.getResponse().getRx64Response(rx64);
                rdata = rx64.getData();
            }
        } else if (xbee.getResponse().getApiId() == RX_NOT.getResponse()) {
            String x = "";
            for (int i = 0; i < 80; i++) {
                x = x + (char) rdata[i];
            }
        }
    }
    delay(100);
    digitalWrite(13, !"0" + x.charAt(0));
    for (int i = 0; i < 80; i++) {
        sData[i] = x.charAt(i);
    }
    addr = XbeeAddress64(0X13A000, 0X4B58284);
    tx = Tx64Request(addr, sData, sizeof(sData));
    xbee.send(tx);
}
Drawback of SenScript

Low level language

→ It is necessary to use a higher level language

Similar to Assembly language
Drawback of SenScript

C code:

```
int x = 3;
x = 5*(x*6-3)+1;
```
Drawback of SenScript

C code:

```c
int x = 3;
x = 5*(x*6-3)+1;
```

SenScript:

```
int x 3
mult x $x 6
minus x $x 3
mult x $x 5
plus x $x 1
```
SenScript → Arduino
SenScript → Arduino

Simplify (reduce coding)

Middleware Language
SenScript → Arduino

- Simplify (reduce coding)
- Close to Arduino structure
Merci pour votre attention