Controler sa maison à l'aide d'un Raspberry: Exemples de codes de controle des composants

Essig Meryll Fortin Loic Rocacher Tamara

4 février 2016

Introduction

Dans l'attente du materiel electronique, nous avons commencé à organiser le code de controle des differents composants. Pour cela nous avons définit les premiers éléments que nous voulions utiliser pour le controle de la maison. Dans l'idée d'etre proche de la réalité, nous voulons travailler sur une maquette de maison, ce qui implique des limites sur ce qui es controlable et sur la taille des composants. Ainsi, nous avons décider de commencer avec les actions suivantes :

- controler la lumières des pièces de vie,
- controler la mise en marche du chauffage principale,
- controler l'ouverture des volets de la chambre

Chaque composant pourra etre controlé sur demande de l'utilisateur, via l'interface, ou automatiquement dans different modes pré-définis :

- mode jour : lorsque le soleil se lève, les volets s'ouvrent,
- mode nuit : lorsque le soleil se couche, les volets se ferment, les lumières s'allument
- mode hiver : le chauffage se met en marche sur une periode définie par l'utilisateur

D'autres modes ou d'autres actions pourront etre ajoutés par la suite. Cependant nous sommes partis de ces trois composants pour rechercher et construire le code général de chacun.

Controle d'une LED pour la lumière

```
1
2  import com.pi4j.io.gpio.GpioController; // Permet de contrler les GPIO
3  import com.pi4j.io.gpio.GpioFactory; // Contient les mthodes statiques permettant de  crer  des instances de
       GpioController.
4  import com.pi4j.io.gpio.GpioPinDigitalOutput; // Les sorties sur le GPIO (contrle sortant des GPIO)
5  import com.pi4j.io.gpio.PinState; // Permet d'avoir des informations sur l'tat du pin
6  import com.pi4j.io.gpio.RaspiPin; // Les diffrents pins du Raspeberry
7  import java.lang.System;
8  import java.lang.Thread;
9  import java.lang.InterruptedException;
10
11 public class led {
12     private final GpioPinDigitalOutput pin; // Permet l'accs en sortie vers le pin.
13
14     public led(int pin) {
15         // Pin  fix  sur 1 dans le code. Ne tien pas compte de l'argument de fonction
16
17         this.pin = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01, "LedControl", PinState.LOW); // L'accs
               en criture au pin pour allumer la led. Par dfaut, la led est  teinte .
18     }
19  /*
20     Allume la led
21      */
22     public void on() {
23         //this.pin.setShutdownOptions(true, PinState.HIGH);
24         this.pin.high();
25     }
26
27     /*
28     Eteint la led
29      */
30     public void off() {
31         //this.pin.setShutdownOptions(true, PinState.LOW);
32         this.pin.low();
33     }
34
35
36     /*
37     Vrai si la led est  teinte , faux sinon.
38      */
39     public boolean isLow() {
40         return this.pin.isLow();
41     }
42
43     /*
44     Vrai si la led est  allume , faux sinon
45      */
46     public boolean isHigh() {
47         return this.pin.isHigh();
48     } /*
49       Rcupre  l' tat  du pin (HIGH ou LOW)
50      */
51     public boolean getState() {
52         return this.pin.getState();
53     }
```

```
    /*
    Eclairage clignotant de la led

    time int Reprsente le temps que doit rester la led alllume
     */
    public void pulse(int time) {
        // Mme chose, il faudrait vrifier que les valeurs de time soient correctes.
        this.pin.pulse(time, true);
    }
}

public class ledControl {
    /*
    Programme principal, permet de lancer (thoriquement ...) un test de led sur le raspberry.
    Note : un thread est susceptible de lancer un InterruptedException. Il est ignor ici.
     */
    public static void main(String[] args) throws InterruptedException {
        led ledtest = new led(1);

        // Donne un accs au gpio
        final GpioController gpio = GpioFactory.getInstance();

        ledtest.on();
System.out.println("LED allume.");

        Thread.sleep(5000);

        ledtest.off();

        System.out.println("LED teinte.");

        Thread.sleep(5000);

        System.out.println("LED blink (interval : 1s).");

        ledtest.pulse(1000);

        // Libre les accs GPIO
        gpio.shutdown();
    }
}
```

Controle d'un micro-moteur pour les volets électriques

Pour ce composant, on se base sur un servomoteur multi-tours, qui tourne par pas d'avancement.

On utilisera donc une classe existante de pi4j, **GpioStepperMotorComponent**, pour représenter le composant par ses états (arret : STOP, marche avant : FORWARD, marche arrière : REVERSE) ; et une nouvelle classe de controle héritée de la classe **GpioStepperMotorControl**, qui contient la méthode run, permettant de mettre en route le moteur selon son état, à laquelle on ajoute un attribut afin de retenir l'état des volets (ouverts ou fermés).

Le main() permet de verifier l'etat des volets avant de lancer l'ouverture puis la fermeture. Par la suite, l'ouverture ou la fermeture seront demandées par un événement qui viendra d'un bouton de l'interface utilisateur.

```java
import com.pi4j.component.motor.impl.GpioStepperMotorComponent;
import com.pi4j.component.motor.impl.GpioStepperMotorControl;
import com.pi4j.io.gpio.GpioController; // Permet de contrler les GPIO
import com.pi4j.io.gpio.GpioFactory; // Contient les mthodes statiques permettant de crer des instances de
    GpioController.
import com.pi4j.io.gpio.GpioPinDigitalOutput; // Les sorties sur le GPIO (contrle sortant des GPIO)
import com.pi4j.io.gpio.PinState; // Permet d'avoir des informations sur l'tat du pin
import com.pi4j.io.gpio.RaspiPin; // Les diffrents pins du Raspeberry
import java.lang.System;
import java.lang.Thread;
import java.lang.InterruptedException;


public class MotorControl extends GpioStepperMotorControl {

  private boolean open;
  private boolean closed;
  private GpioStepperMotorComponent motor;

  public MotorControl(GpioPinDigitalOutput pins[]){
    super(pins);
    open=false;
    closed=true;
    motor= new GpioStepperMotorComponent(pins);
  }

  public static void main(String[] args) throws InterruptedException {

    final GpioController gpio = GpioFactory.getInstance();
    GpioPinDigitalOutput[] pins = new GpioPinDigitalOutput[4];

    pins[0] = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01, "MotorPin1", PinState.HIGH);
    pins[1] = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_02, "MotorPin2", PinState.HIGH);
  pins[2] = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_03, "MotorPin3", PinState.HIGH);
  pins[3] = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_04, "MotorPin4", PinState.HIGH);

  GpioStepperMotorControl controller = new GpioStepperMotorControl(pins);

       System.out.println("Moteur eteint, volets ferms");
  //ouverture des volets s'ils sont ferms
  if(controller.closed)
  controller.ouvrir();
  System.out.println("Moteur allum en ouverture");
  While (motor.getSate() != STOP){
    System.out.print(". "); // affiche la progression de l'ouverture et assure d'attendre l' arrt du moteur
  }
  System.out.println("\nMoteur teint,volets ouverts");

  //fermeture des volets s'ils sont ouverts
  if(controller.open)
  controller.fermer();
  System.out.println("Moteur allum en fermeture");
  While (motor.getSate() != STOP){
    System.out.print(". "); // affiche la progression de la fermeture et assure d'attendre l' arrt du moteur
  }
  System.out.println("\nMoteur teint,volets ferms");
  }
```

```
  public void ouvrir(){
  motor.setState(FORWARD);
  controller.run();
  open=true;
  closed=false;
  }

  public boolean fermer(){
    motor.setState(REVERSE);
    controller.run();
    open=false;
    closed=true;
    }
}
```

Controle d'un luxmetre pour la detection de luminosité exterieure

Ce composant permettra d'activer les modes jour et nuit en fonction de la luminosité exterieure, que nous pourront faire varier avec une lampe pour imuler le lever et le coucher du soleil.
Ainsi, il s'agira pour le controle du luxmetre de récupérer une information analogique en entrée, ce qui permettra par la suite de déclencher un événement agissant sur d'autre classes de composants.
Pour cette catégorie d'éléments, nous sommes encore sur la documentation et les essais de code.