

# 树莓派开发

## 16 在树莓派上用JAVA控制电灯

# 1、配置树莓派

## □ 1) 安装Java

- 要想控制继电器，我们必须先学会控制树莓派上的GPIO
- 控制GPIO有很多方式：python、wiringPi、pi4j。
- 这里介绍的pi4j，该项目旨在为全面进入Raspberry Pi的提供本地库和Java之间的桥梁
- 要想使用pi4j必须给树莓派安装Java，不过在最新的树莓派系统Raspbian中已经内置了Java（[2013-12-20-wheezy-raspbian.zip](http://2013-12-20-wheezy-raspbian.zip)）

# 1、配置树莓派

- ❑ 如果不知道你的树莓派中有没有Java，你可以使用javac -version 来查看是否有安装了Java。
- ❑ 如下图出现了版本信息，就表示安装了Java。如果没有就去下载最新的Raspbian系统吧。

```
pi@raspberrypi: ~  
pi@raspberrypi ~$ javac -version  
javac 1.7.0_40  
pi@raspberrypi ~$ █  
  
http://blog.csdn.net/qigenhuochai
```

# 1、配置树莓派

□ 如果已经有java了，依照惯例运行一个HelloWorld程序：

```
pi@raspberrypi: ~/code/java
pi@raspberrypi ~$ cd code/java
pi@raspberrypi ~/code/java$ javac HelloWorld.java
pi@raspberrypi ~/code/java$ java HelloWorld
Hello World!
pi@raspberrypi ~/code/java$
```

<http://blog.csdn.net/qigenhuochai>

□ 作为Java程序员，看到这个在树莓派上运行，这意味着你以前在Windows上编写的程序都可以直接放到树莓派上运行，而不需要重新编译，Java嘛，跨平台地球人都知道。是不是？

## 2、安装Pi4j

- ❑ 如果没有，则要下载安装一下。
- ❑ 打开树莓派的控制台输入下面的命令下载Pi4j文件  
`wget http://get.pi4j.com/download/pi4j-0.0.5.deb`
- ❑ 如果你已经安装了Pi4j的其他版本，请先执行如下命令将其卸载掉：`sudo dpkg -r pi4j`
- ❑ 下载完成后使用下面的命令安装：  
`sudo dpkg -i pi4j-0.0.5.deb`
- ❑ 安装Pi4j库和示例文件：  
`/opt/pi4j/lib`
- ❑ 如果想要卸载Pi4j执行如下命令：  
`sudo dpkg -r pi4j`

### 3、Pi4j程序的编译与运行示例

- ❑ 要想编译使用Pi4j的程序必须指明其包的路径
- ❑ 比如在home/pi/code/java/下编写了一个叫Test.java的程序，应该执行如下命令：
- ❑ 首先进入到程序的目录：  
`cd code/java`
- ❑ 编译：  
`javac -classpath .:classes:/opt/pi4j/lib/*' -d . Test.java`
- ❑ 运行：  
`sudo java -classpath .:classes:/opt/pi4j/lib/*' Test`

## 2、安装Pi4j

- ❑ 如果没有，则要下载安装一下。
- ❑ 打开树莓派的控制台输入下面的命令下载Pi4j文件  
`wget http://get.pi4j.com/download/pi4j-0.0.5.deb`
- ❑ 如果你已经安装了Pi4j的其他版本，请先执行如下命令将其卸载掉：`sudo dpkg -r pi4j`
- ❑ 下载完成后使用下面的命令安装：  
`sudo dpkg -i pi4j-0.0.5.deb`
- ❑ 安装Pi4j库和示例文件：  
`/opt/pi4j/lib`
- ❑ 如果想要卸载Pi4j执行如下命令：  
`sudo dpkg -r pi4j`

### 3、Pi4j程序的编译与运行示例

- ❑ 要想编译使用Pi4j的程序必须指明其包的路径
- ❑ 比如在home/pi/code/java/下编写了一个叫Test.java的程序，应该执行如下命令：
- ❑ 首先进入到程序的目录：  
`cd code/java`
- ❑ 编译：  
`javac -classpath .:classes:/opt/pi4j/lib/'*' -d . Test.java`
- ❑ 运行：  
`sudo java -classpath .:classes:/opt/pi4j/lib/'*' Test`



## 4、连接GPIO和继电器

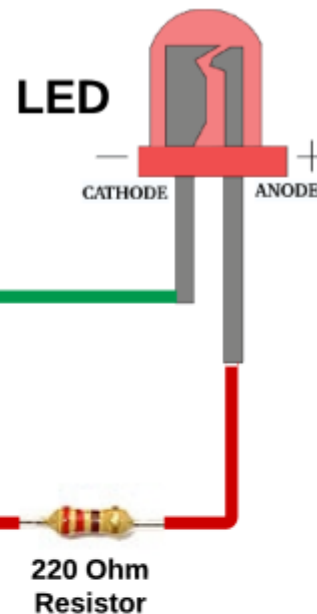


□ 这树莓派的GPIO针脚（实物）

## 4、连接GPIO和继电器

Raspberry Pi P1 Header				
PIN #	NAME		NAME	PIN #
	3.3 VDC Power	1	5.0 VDC Power	2
8	SDA0 (I2C)	3	DNC	4
9	SCL0 (I2C)	5	0V (Ground)	6
7	GPIO 7	7	TxD	15
	DNC	9	RxD	16
0	GPIO 0	11	GPIO1	1
2	GPIO2	13	DNC	14
3	GPIO3	15	GPIO4	4
	DNC	17	GPIO5	5
12	MOSI	19	DNC	20
13	MISO	21	GPIO6	6
14	SCLK	23	CE0	10
	DNC	25	CE1	11

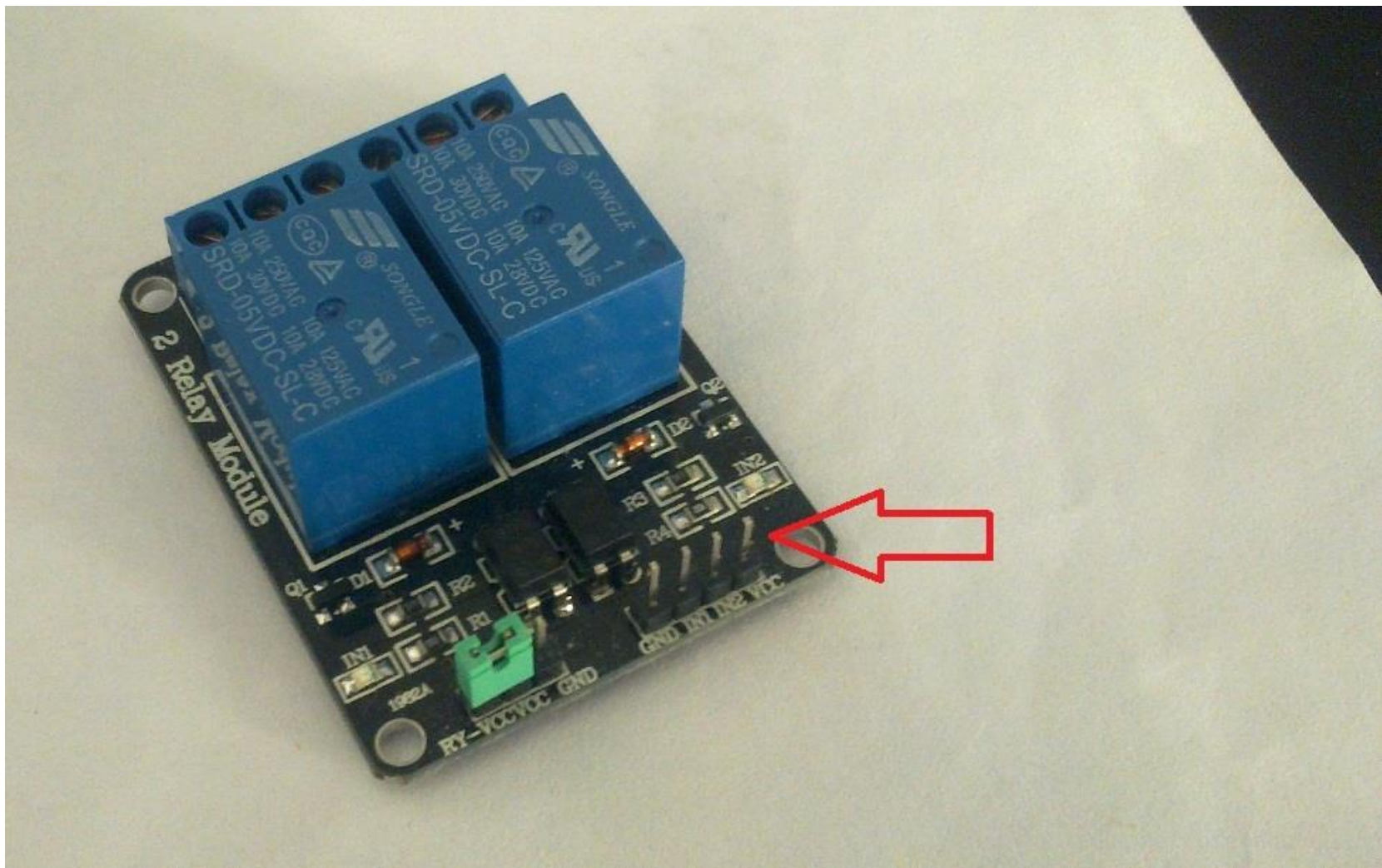
<http://www.pi4j.com>



□ 这便是树莓派的  
GPIO针脚的逻辑图

## 4、连接GPIO和继电器

- 我们希望通过GPIO对继电器进行控制，控制了继电器就等于控制了电灯。



## 4、连接GPIO和继电器

- 继电器的这个地方有四个针脚：
- 分别为：GND（接地）、IN1（控制继电器1）、IN2（控制继电器2）、VCC（电源）放大来看：

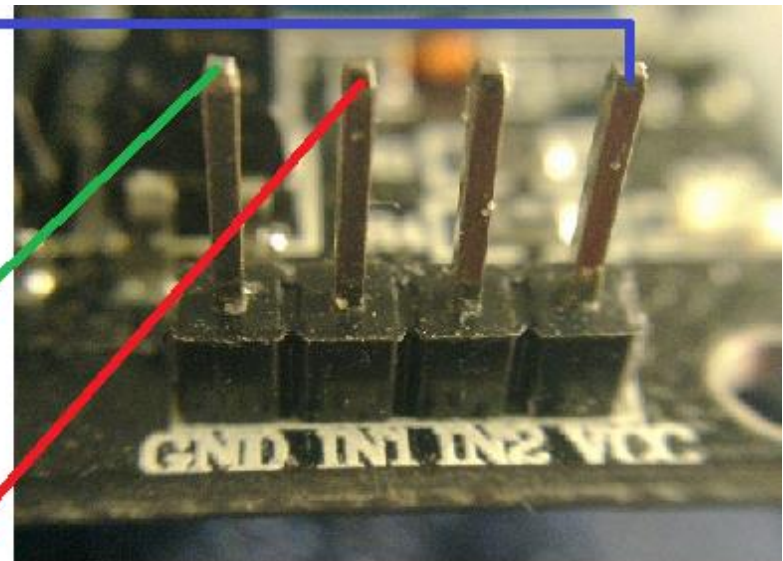




## 4、连接GPIO和继电器

Raspberry Pi P1 Header				
PIN #	NAME		NAME	PIN #
	3.3 VDC Power	1	3.3 VDC Power	
8	SDA0 (I2C)	3	DNC	4
9	SCL0 (I2C)	5	0V (Ground)	6
7	GPIO 7	7	TxD	15
	DNC	9	RxD	16
0	GPIO 0	11	GPIO1	1
2	GPIO2	13	DNC	14
3	GPIO3	15	GPIO4	4
	DNC	17	GPIO5	5
12	MOSI	19	DNC	20
13	MISO	21	GPIO6	6
14	SCLK	23	CE0	10
	DNC	25	CE1	11

<http://www.pi4j.com>



- 连接GPIO和继电器：
- GND接树莓派上的6号针脚，VCC接树莓派的2号针脚，因为继电器工作电压为5V，所以不需要电阻，IN1接在树莓派的12号针脚（GPIO1）。连接如图。

## 4、连接GPIO和继电器

- 连接好之后就是这个样子的（这个图片中IN1连接的是26号针脚）：



## 5、控制继电器

- ❑ 在home/pi/code/java/下新建java程序Test.java，功能就是让继电器每隔一秒就切换一下状态：

```
import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;
/** 控制树莓派上的GPIO实例
public class Test {
    public static void main(String[] args) throws InterruptedException {
        // 创建一个GPIO控制器
        final GpioController gpio = GpioFactory.getInstance();
        // 获取1号GPIO针脚并设置高电平状态，对应的是树莓派上的12号针脚
        // ，可以参考pi4j提供的图片。
        final GpioPinDigitalOutput pin = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01, "LED", PinState.HIGH);
```

## 5、控制继电器

```
while(true){  //设置高电平
    pin.high();
    System.out.println("打开继电器");
    //睡眠1秒
    Thread.sleep(1000);
    //设置低电平
    pin.low();
    System.out.println("关闭继电器");
    Thread.sleep(1000);
    //切换状态
    //pin.toggle();
}
}
```



## 5、控制继电器

- ❑ 进入到程序的目录：

```
cd code/java
```

- ❑ 编译：

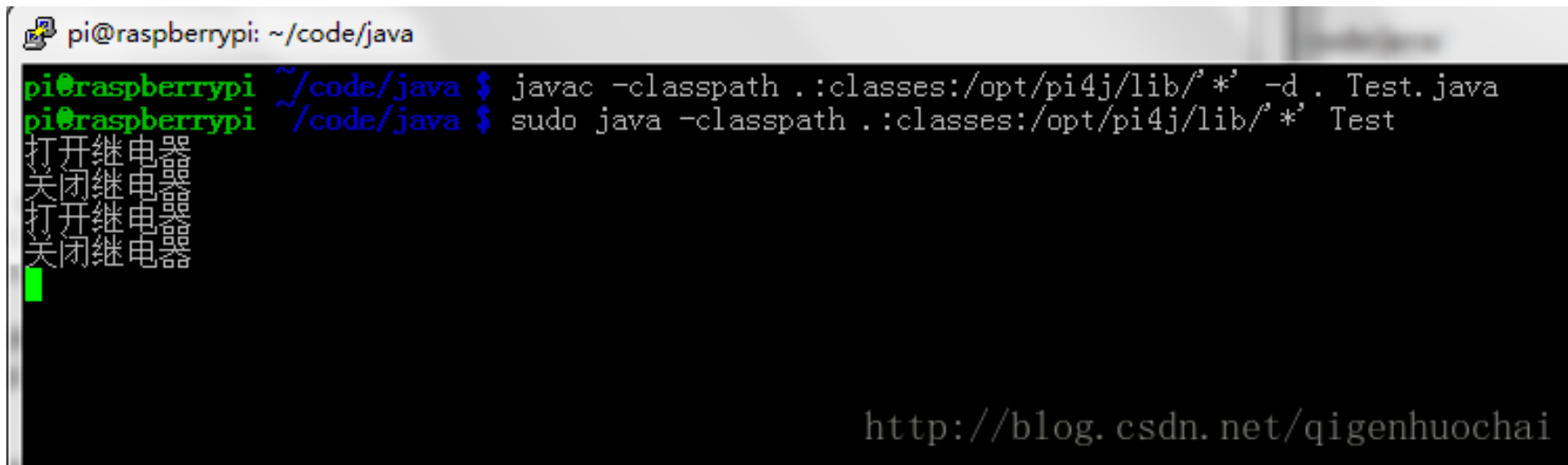
```
javac -classpath .:classes:/opt/pi4j/lib/'*' -d . Test.java
```

- ❑ 运行：

```
sudo java -classpath .:classes:/opt/pi4j/lib/'*' Test
```

## 5、控制继电器

□ 运行结果如下图：



```
pi@raspberrypi: ~/code/java
pi@raspberrypi ~/code/java $ javac -classpath .:classes:/opt/pi4j/lib/* -d . Test.java
pi@raspberrypi ~/code/java $ sudo java -classpath .:classes:/opt/pi4j/lib/* Test
打开继电器
关闭继电器
打开继电器
关闭继电器

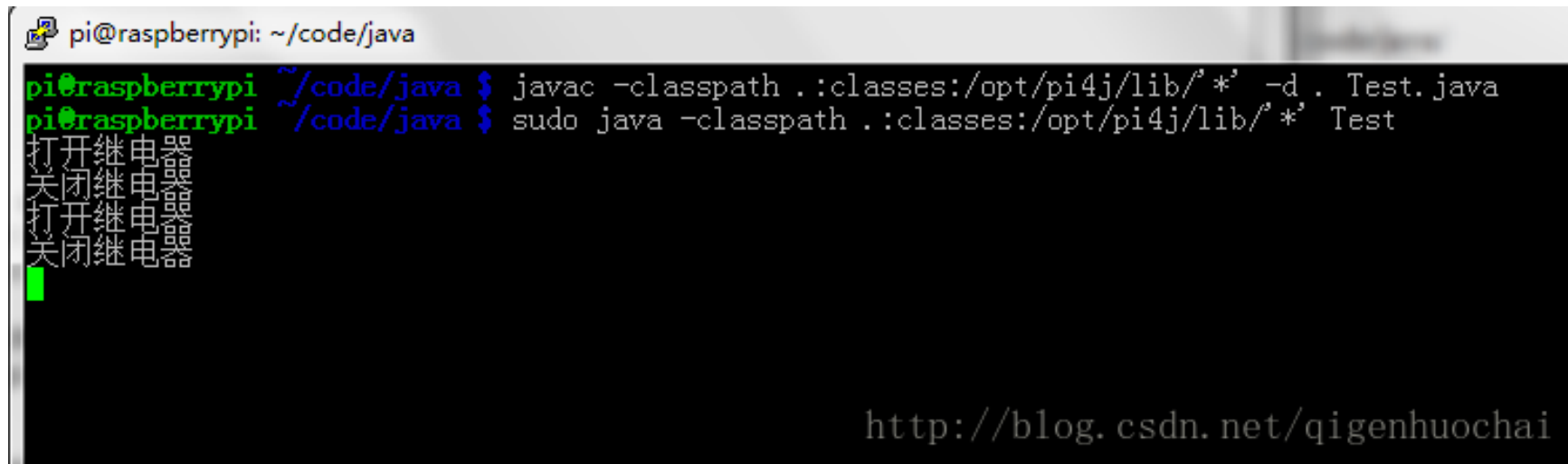
```

<http://blog.csdn.net/qigenhuochai>

- 而此时电灯是一闪一闪的。
- 只要把继电器改为别的电器，这个电器就“一闪一闪”的了。
- 把树莓派放到公网上，就可以远程（如在办公室）控制你家里的树莓派电灯了。

## 5、控制继电器

□ 运行结果如下图：



```
pi@raspberrypi: ~/code/java
pi@raspberrypi ~/code/java $ javac -classpath .:classes:/opt/pi4j/lib/* -d . Test.java
pi@raspberrypi ~/code/java $ sudo java -classpath .:classes:/opt/pi4j/lib/* Test
打开继电器
关闭继电器
打开继电器
关闭继电器

```

<http://blog.csdn.net/qigenhuochai>

□ 而此时电灯是一闪一闪的。

□ 只要把继电器改为别的电器，这个电器就“一闪一闪”的了

□ 把树莓派放到公网上（需要自己的静态IP），就可以远程（如在办公室）控制你家里的树莓派电灯了。

# 7、Yeelink平台

## □接入任何的传感器设备

□Yeelink独有设计的高并发接入服务器和云存储方案，能够同时完成海量的传感器数据接入和存储任务，确保您的数据能够安全的保存在互联网上，先进的鉴权系统和安全机制，能够确保数据只在您允许的范围内共享。

## □事件触发机制

□当您的数据达到某个设定阈值的时候，Yeelink平台会自动调用您预先设定的规则，发送短信，微博，或者是邮件，您还可以充分利用平台的计算能力，定期的将统计分析数据发送到邮箱内，这一切仅需在网页上简单的点击几个按钮，家里被非法闯入时候给您的手机发彩信？OK，没问题。

## □基于地理和时间的数据管理

□所有的数据，均可以通过地图和时间轴方式进行展现，使用iphone或android手机，可以很容易的发现身边被公开的传感器，获取诸如空气质量，PM2.5指数等数据，停车场的剩余车位数量，或是获取其他类似公交状况等城市公共数据。

## 6、Yeelink平台

### □释放您的智慧和创新能力

□在Yeelink上，极客们手中的arduino和能力将被完全释放出来，不需要编写一行代码，无需繁琐的服务器编程技术，就能够将手中的硬件和传感器数据通过网络发布出来，并能随时随地的将数据从服务器中取回，通过微博插件向您的朋友或是社会分享，让您的创意没有边界。

### □双向传输和控制功能

□Yeelink平台的最大特点，在于不仅仅能够提供数据的上行功能，还能够实现对家庭电器的控制功能，快要到家前想洗个热水澡，还是要提前把空调打开？很简单，用手机的智能App，这些就是举手之劳。

### □社交网络融合

□在Yeelink上，数据不再是孤单的节点，存储在Yeelink的数据，可以简单的被API取回，放置到您的个人博客上，或者根据规则自动转发到您指定的微博上，在这里，您将会感受到数据和人之间的全面融合。

## 7、 注册Yeelink账号

- ❑用Yeelink的手机客户端（浏览器也行）去改变服务器上特定设备的状态（0/1），然后让树莓派检测该设备上的状态，最后根据获取到的状态控制继电器。
- ❑访问Yeelink的官方网站（<http://www.yeelink.net/>），点击快速开始，按照步骤注册账号。

## 8、添加一个设备

□ 下面是Yeelink的官方向导：

□ 1 进入用户中心：



# 7、使用Yeelink平台

## □2 增加新设备



The screenshot displays the Yeelink user interface. On the left is a dark sidebar with the Yeelink logo and a navigation menu. The main content area is titled '欢迎使用 Yeelink' (Welcome to Yeelink) and includes a '设备概要' (Device Overview) table. A red circle highlights the '增加新设备' (Add New Device) option in the sidebar, and a red arrow points from this option to the table. A text overlay in the center-right of the image reads: '在“我的设备”中选择“增加新设备”' (Select 'Add New Device' in 'My Devices').

**Yeelink™**  
欢迎您 [用户名]  
您上次登录于 2013-07-25 16:56:27  
注销

用户中心 首页

- 帐户
- 我的设备
- 增加新设备**
- 管理设备
- 管理动作

API 文档

Yeelink

### 欢迎使用 Yeelink

—— 免费、开放的平台，快速开启您的物联网时代

#### 设备概要

设备类型	设备名称	状态	最新数据
自组设备	Yeelink光敏电阻		
自组设备	LED开关		
自组设备	微博控制LED		
自组设备	Yeelink温湿度		

© Copyright 2013 青岛亿联智信信息技术有限公司 | Privacy Policy | Yeelink

在“我的设备”中选择“增加新设备”

<http://blog.csdn.net/qigenhuochai>



## □2 增加新设备



### 欢迎使用 Yeelink

—— 免费、开放的平台，快速开启您的物联网时代

#### 管理设备

##### LED开关

远程控制LED的开关

如何查看设备编号和传感器编号？

在"用户中心"中选择 "我的设备" -- "管理设备" --  
在"状态URL"中即可查看

无图无真相

鼠标移上图片，为它增加描述：

传感器

状态URL: 为设备添加描述，以便在状态改变时通知您。

增加一个传感器

<http://api.yeelink.net/v1.0/device/4077/sensor/5818/datapoints>



控制开关



用于控制LED的开关：

状态URL

<http://api.yeelink.net/v1.0/device/4077/sensor/5818/datapoints>

控制操作

按动图标以改变开关的状态



设备编号

传感器编号

## □2 增加新设备



欢迎使用 Yeelink  
—— 免费、开放的平台，快速开启您的物联网时代

管理设备

LED开关

远程控制LED的开关

如何查看设备编号和传感器编号？

在"用户中心"中选择 "我的设备" -- "管理设备" --  
在"状态URL"中即可查看

无图无真相

鼠标移上图片，为它增加描述：

传感器

状态URL: 为设备添加描述，以便在状态改变时通知您。

增加一个传感器

<http://api.yeelink.net/v1.0/device/4077/sensor/5818/datapoints>



控制开关



用于控制LED的开关：

状态URL

<http://api.yeelink.net/v1.0/device/4077/sensor/5818/datapoints>

控制操作

按动图标以改变开关的状态



设备编号

传感器编号

## 欢迎使用 Yeelink

—— 免费、开放的平台，快速开启您的物联网时代

## 管理设备

## LED开关

远程控制LED的开关

设备类型

状态

最新数据

自备设备

设备图片

设备图片将在前台devices界面展示。

上传图片



无图无真相

就标移上图片，为它增加描述。

## 传感器

管理该设备的传感器并为它们添加联动动作，以便在状态改变时通知您。

+ 增加一个传感器

还没有任何传感器

点击“管理设备”即可

看到我们之前添加的设备“LED开关”

点击“增加一个传感器”为我们的设备“LED开关”增加一个传感器

## 欢迎使用 Yeelink

—— 免费、开放的平台，快速开启您的物联网时代

## 增加新传感器

传感器名

为它起一个标记的名字,以区别于其它传感器

控制开关



类型

开关



标签TAGS

标签之间请使用逗号分隔

LED,开关



描述

不超过30个字符

用于控制LED的开关



清空

保存

填写相应的传感器信息，  
注意“类型”中选择“开关”

填写完成后点击“保存”

## 欢迎使用 Yeelink

— 免费、开放的平台，快速开启您的物联网时代

## 管理设备

## LED开关

远程控制LED开关

设备

设备类型

状态

最新数据

自设备



## 设备图片

设备图片将在前台devices界面展示

上传图片



无图无真相

鼠标移上图片，为它增加描述

现在我已经完成了重要的两步：

- 1、我们新建了一个名为“LED开关”的设备
- 2、我们为这个设备中添加了一个名为“控制开关”的开关型传感器

## 传感器

管理该设备的传感器并为它们增加触发动作，以便在状态改变时通知您

增加一个传感器



控制开关



用于控制LED的开关

状态URL

<http://api.yeelink.net/v1.0/devices/4077/sensor/5818/datapoints>

控制操作

该动作用于改变开关的状态



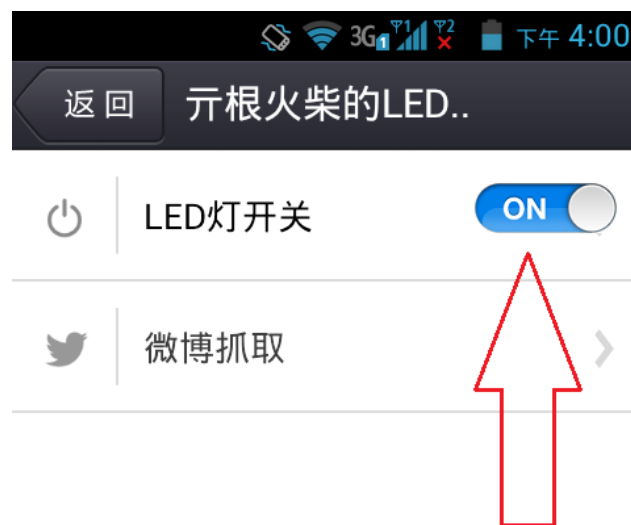
开关型传感器

### ❑3 下载Yeelink的客户端

❑访问Yeelink的下载页面下载手机客户端来控制开关:

<http://www.yeelink.net/developer/tools>

❑打开后点击“我的Yeelink”，然后点击你的设备，使用开关来控制设备的状态。



## 8、完整的Java代码

- ❑搭建好Yeelink平台后，就可以用树莓派访问设备的状态链接以获得设备的当前状态，然后根据得到的状态来控制GPIO。
- ❑编写程序每2秒获取一次状态，访问时间间隔不建议设置的太短。
- ❑在home/pi/code/java/下新建java程序ControlLed.java，代码没有什么算法，都是基本的操作。



```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;
/**
 * 远程控制树莓派上的GPIO实例
 **/
public class ControlLed {
    GpioController gpio;
    GpioPinDigitalOutput pin;
    public static void main(String s[]){
        //下面的链接就是你在Yeelink的设备的状态URL
        String urlStr = "http://api.yeelink.net/v1.0/device/ 《这里是你的设备编号》
```



```

/sensor/ 《这里是你的传感器编号》 /datapoints";
    ControlLed cl = new ControlLed();
    cl.gpio = GpioFactory.getInstance();
    cl.pin = cl.gpio.provisionDigitalOutputPin
(RaspiPin.GPIO_01, "LED", PinState.HIGH);
    boolean current = cl.getStatus(urlStr);
    cl.setLedStatus(current);
    while(true){
        try{
            if(cl.getStatus(urlStr) != current){
                current = cl.getStatus(urlStr);
                cl.setLedStatus(current);
            }
            Thread.sleep(2000);
        }catch(Exception e){e.printStackTrace();}
    }
}

public boolean getStatus(String urlStr){
    URL url;boolean on = false;
    try {
        url = new URL(urlStr);
        HttpURLConnection conn = (HttpURLConnection)

```

```

url.openConnection();
    conn.setRequestMethod("GET");
    InputStream in = conn.getInputStream();
    BufferedReader rd = new BufferedReader(new InputStreamReader(in));
    StringBuilder tempStr = new StringBuilder();
    while (rd.read() != -1) {
        tempStr.append(rd.readLine());
    }
    //System.out.print("--> 服务器上传感器的信息: ");
    String status = tempStr.substring (tempStr.lastIndexOf(":")+1, tempStr.length()-1);
    //System.out.println(status);
    on = status.equals("1")? true:false;
    //System.out.println(on);
    return on;
} catch (IOException e) {
    e.printStackTrace();
} return on;
}

```

```
public void setLedStatus(boolean sta){  
    if(sta){  
        pin.low();  
        //因为我的继电器是低电平有效  
        System.out.println("--> 更新GPIO的状态: 开");  
    }else {  
        pin.high();  
        System.out.println("--> 更新GPIO的状态: 关");  
    }  
}  
}
```

```
public void setLedStatus(boolean sta){
    if(sta){
        pin.low();
        //因为我的继电器是低电平有效
        System.out.println("--> 更新GPIO的状态: 开");
    }else {
        pin.high();
        System.out.println("--> 更新GPIO的状态: 关");
    }
}
}
```