Raspberry Pi + Pebble + Android

构建属于你自己的智能家居系统

Phodal Huang

October 24, 2017

目录

原文链接：https://www.wandianshenme.com/play/diy-smart-home-by-raspberry-pi-pebble-smartw

市面上，已经有几种产品使您的公寓更智能，但大多数方案都是专有解决方案。但是为什么需要互联网连接，才能用智能手机切换灯光？这也是我建立自己的智能家居解决方案的一个原因。

我编写了一个在 Raspberry Pi 上，运行的服务器应用程序。这是一个基于 Java 的开源项目，它允许您配置您的平板，并连接多个客户端和“可控制单元”。我展示了一个解决方案，可以处理遥控（rc）电源开关，在 Raspberry Pi 上播放音乐和视频，显示 Smart Mirror （智能镜子）的状态，可以由 Android 应用程序和两个 Pebble 应用程序控制。源码托管在我的 GitHub 上：http://www.instructables.com/id/Smart-Home-by-Raspberry-Pi/

步骤 1：材料清单

要设置智能家居，您需要以下“成分”（材料）：

- Raspberry Pi 2 B 以上的型号
- 433 MHz 无线发射器
- 连接 Raspberry Pi 和发射方的 3 条跳线
- 一些 433 MHz 的无线电控制插座
- Android 智能手机，用于运行客户端应用程序

此外，您可以使用更多可选的客户端和单位来扩展这个项目：

- Pebble 智能手表
- Smart Mirror（智能镜子）（见：Smart Mirror by Raspberry Pi）
- 433 MHz 控制 LED 灯条（见：RC Controlled Rgb Led Strip）

步骤 2：为 Raspberry Pi 准备 433 MHz

在以下步骤里，您需要访问 Raspberry Pi 上的命令行。

将 433 MHz 发射器与 Raspberry Pi 连接：

- GND (433Mhz) <-> 6 GND (Raspberry Pi)
- VCC (433Mhz) <-> 2 +5V (Raspberry Pi)
- DATA (433Mhz) <-> 11 GPIO 17 (Raspberry Pi)

请将 17cm 的天线连接到 ANT（发送器）引脚。它增加了信号的强度。

由于我们需要一些其他 git 项目库，所以我们必须安装 git：

```
sudo apt-get install git-core -y
```

为了在 Raspberry Pi 上建立 433 MHz 通信，我们需要 wiring Pi 库，以便更好地处理 GPIO。

```
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build
```

然后我们需要一个实现通用的 rc 协议的库。

```
git clone git://github.com/r10r/rcswitch-pi
cd rcswitch-pi
make
cp send /usr/bin/
```

“send” 可执行文件，将允许您发送代码来切换大部分可用的电源。

在我的智能家居设置中，我还有一个 LED 灯条，见：http://www.instructables.com/id/RC-controlled-LED-strip/ 要设置此 LED 灯条的颜色，您需要另一个发送可执行文件，它将允许您发送任何整数值（编码颜色）。

```
#include "RCSwitch.h"

// TODO INCLUDES MISSING
int main(int argc, char *argv[]) {
    int PIN = 0;
    int message  = atoi(argv[1]);
    if (wiringPiSetup () == 1) return 1;
    printf("sending message[%d]\n", message);
    RCSwitch mySwitch = RCSwitch();
    mySwitch.enableTransmit(PIN);
    mySwitch.send(message, 32);
}
```

因此，编译以下程序并将其移动到 /usr/bin/sendInt：

```
sudo g++ $file -o /usr/bin/sendInt /home/pi/rcswitch-pi/RCSwitch.o
    -I/home/pi/rcswitch-pi -lwiringPi
```

现在，你已经有两个可用的执行文件来发送 rc 命令：**/usr/bin/send** 和 **/usr/bin/sendInt**。

步骤 3：搭建智能家居服务器

首先你需要安装几个软件包。

- 智能家居应用程序是基于 Java 的，在 OpenJDK-7 上运行良好。我不知道其他 Java 运行时环境是否正常，但是 Oracle Java 8 也应该可以工作。
- mplayer 是一个极简主义的命令行音乐播放器。
- omxplayer 使用 Raspberry Pi 图形进行视频编码，因此应该用于视频。
- 此程序需要 ant 来构建 Java 应用程序。

执行以下命令完成安装：

```
1 sudo apt-get install mplayer omxplayer openjdk-7-jdk ant -y
```

为 jar 文件和日志创建目录：

```
1 sudo mkdir /opt/neo
2 sudo chown pi:pi /opt/neo
3 mkdir /home/pi/Logs
```

然后配置启动脚本，用于在启动时，自动启动应用程序。为此，需要将附加的 smart-home 脚本复制到目录 /etc/init.d/ 中。我还在 /usr/bin/ 中创建了一个脚本，它将命令管道附加到脚本中，所以我刚刚进入 smart-home 到控制台执行命令。

```
1 sudo cp smart-home /etc/init.d/smart-home
2 sudo chmod +x /etc/init.d/smart-home
3 sudo sh -c "echo '#! /bin/bash' > /usr/bin/smart-home"
4 sudo sh -c "echo '/etc/init.d/smart-home \$1' >> /usr/bin/smart-home"
5 sudo chmod +x /usr/bin/smart-home
6 sudo update-rc.d smart-home defaults
```

现在是时候检查代码库，并构建应用程序了。如果您不想自己编译，您可以下载附加的 smarthome.jar 并将其移动到 /opt/neo/ ：

```
1 git clone git@github.com:dabastynator/SmartHome.git
2 ant -f SmartHome/de.neo.smarthome.build/build.ant build_remote
3 cp SmartHome/de.neo.smarthome.build/build/jar/* /opt/neo/
```

尝试启动 smart-home 并检查日志文件。为了访问 GPIO，该应用程序必须由 sudo 启动。

```
sudo smart-home start
cat Logs/smarthome.log
```

您应该看到错误消息配置文件不存在，它指向我们移到下一步。代码库中包含一个说明配置文件的自述文件（README）。在 GitHub 上，你可以看得更清楚：https://github.com/dabastynator/SmartHome

将此 xml 文件复制到 `/home/pi/controlcenter.xml`，然后设置媒体服务器的位置，并根据需要更改内容。完成配置，并重新启动智能家居（`sudo smart-home restart`）后，您应该在 `smarthome.log` 中看到以下内容：

```
24.05-08:26 REMOTE INFORMATION by
    de.neo.smarthome.cronjob.CronJob@15aeb7ab: Schedule cron job
24.05-08:26 REMOTE INFORMATION by [trigger.light]: Wait 79391760 ms for
    execution
24.05-08:26 RMI INFORMATION by Add web-handler (5061/ledstrip)
24.05-08:26 RMI INFORMATION by Add web-handler (5061/action)
24.05-08:26 RMI INFORMATION by Add web-handler (5061/mediaserver)
24.05-08:26 RMI INFORMATION by Add web-handler (5061/switch)
24.05-08:26 RMI INFORMATION by Add web-handler (5061/controlcenter)
24.05-08:26 RMI INFORMATION by Start webserver with 5 handler
    (localhost:5061)
24.05-08:26 REMOTE INFORMATION by Controlcenter: Add 1. control unit:
    MyUnit (xyz)
...
```

这表明 Web 服务器正在运行。

smart-home.zip

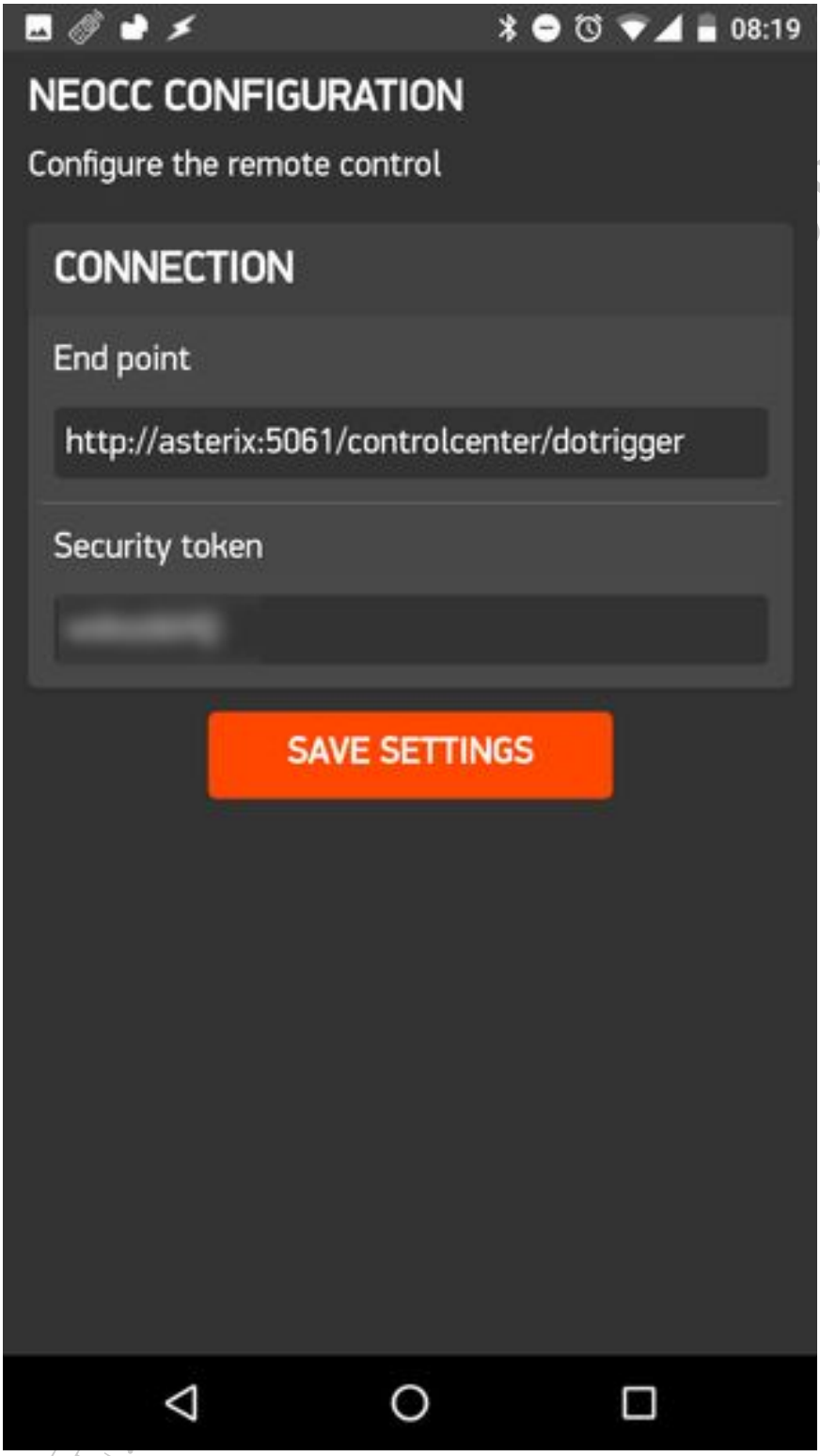
smarthome.jar

步骤 4：设置客户端

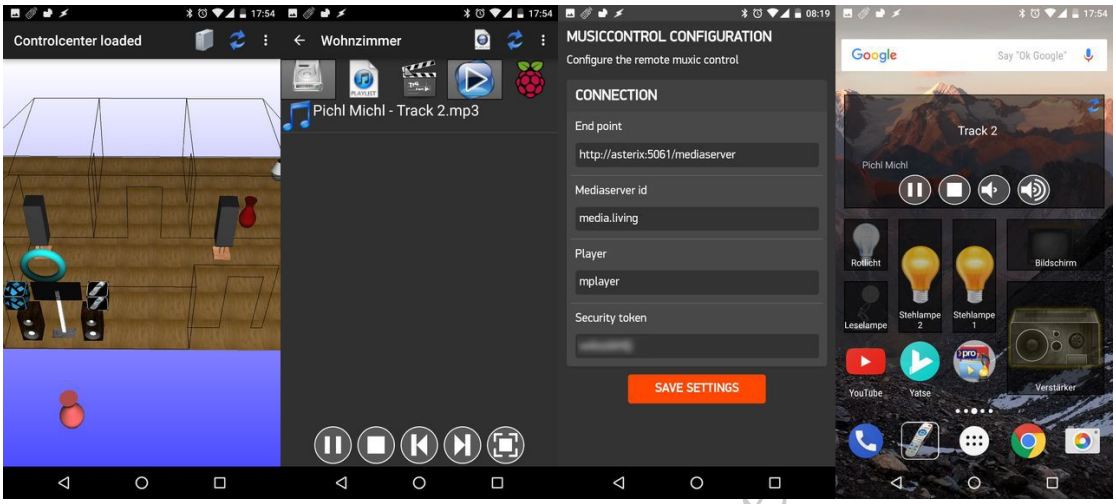
智能手机 Android 客户端

智能家居应用程序的 git 代码库，还包含了 Android 客户端的源代码，因此您可以自己编译它。你也可以直接下载 APK，地址是：de.neo.smarthome.mobile.apk。第一

次启动应用程序时，它会询问您是否有服务器，如下图所示。输入服务器的 URL 和（Security Token）安全令牌：



现在，您应该可以访问服务器，并控制您的平板（flat）、播放音乐、Raspberry Pi 上远程观看视频。请注意，您可以向主屏幕添加小部件，从而使开关和音乐控制更易于访问。



智能手表 **Pebble** 客户端

两个 Pebble 的客户端源码均托管在 GitHub 上。

一个应用程序，显示当前播放的音乐文件：https://github.com/dabastynator/PebbleRemoteMusic 。它可以允许您暂停/播放和控制音量。

第二个应用程序，触发三个操作：https://github.com/dabastynator/PebbleControl 触发器名称是：`mobile.come_home`、`mobile.leaving` 和 `mobile.go_to_bed`。如果您在 xml 配置文件中，为此触发器定义事件规则，则可以通过手表触发它们。

这都是开源的，同时您不需要自己编译，我也附上了 Pebble 应用程序。使用智能手机下载 PBW，并通过你的手机将其安装在手表上。Pebble 应用程序需要配置与服务器通信。在上一张截图中，你可以看到是如何配置的。

应用下载地址：MusicControl.pbw NEOCC.pbw

智能镜子客户端

我之前也创建过一个智能镜子（Smart Mirror）的玩法，详见见：https://www.instructables.com/id/Smart-Mirror-by-Raspberry-Pi/。其代码同样也托管在 GitHub 上：https://github.com/dabastynator/SmartMirror。Smart Mirror 软件从 smart_config.js 文件中读取配置，其不包含在 git 库中。配置文件的内容应如下列表所示：

```
1 var mOpenWeatherKey = 'your-open-wheather-key';
2 var mSecurity = 'your-security-token';
```

您还必须调整文件 smart_mirror.js 的前两行，以指定智能家居服务器的 IP 地址和位置，以获得正确的天气。

更多客户端

我们的服务器应用是一个简单的 Web 服务器。这使您能够通过简单的网络电话，从任何您想要的客户端触发操作。在演示视频中，我将结合 AutoVoice 显示 Android 应用程序任务。因此我能够通过简单的语音命令触发事件。例如 “ok google, time to sleep” 可以触发 `mobile.go_to_bed`。但是，您也可以从 IFTTT 进行网络呼叫。比如，如何将黄色闪烁的 LED 灯条用于发送电子邮件通知?

您可以向服务器询问可能的网络呼叫，如以下链接（通过您的配置替换 IP、端口和令牌）

http://localhost:5061/controlcenter/api?token=security-token

http://localhost:5061/action/api?token=security-token

http://localhost:5061/mediaserver/api?token=security-token

http://localhost:5061/switch/api?token=security-token

http://localhost:5061/ledstrip/api?token=security-token

步骤 **5**：结论

除此，还有一些功能要实现：由于服务器只提供一个简单的 web-api，客户端需要进行大量的轮询。为了减少轮询，我想要集成 MQTT，以获得更好的通知。另外使用 WiFi 应该比 RC 更加可靠，因为 RC 只是单向通信。

为这个项目开发很有趣。即使互联网连接发生故障，也可以通过多个设备来控制公寓。

原文链接: http://www.instructables.com/id/Smart-Home-by-Raspberry-Pi/

原文链接：https://www.wandianshenme.com/play/diy-smart-home-by-raspberry-pi-pebble-smartw