Fernbedienung verbunden mit Raspberry Pi

Den Rollladen steuern mit Pi

René Jahn, SIB Visions GmbH

Die Steuerung von Rollläden kann heutzutage bequem per Fernbedienung oder durch eine zentrale Steuereinheit erfolgen. Der Einsatz von zentralen Steuereinheiten automatisiert wunderbar alltägliche Abläufe wie zeitgesteuertes Öffnen und Schließen. Doch nicht jeder Haushalt mit elektronischen Rollläden besitzt auch eine zentrale Steuereinheit, denn diese sind meist recht teuer und erfüllen äußerst selten die gewünschten Anforderungen. Wer auf keine Funktion verzichten möchte und schon immer von einer eigenen Rollladensteuerung träumte, findet hier eine Anleitung.

Wer bisher seine Rollläden mit einer Fernbedienung steuert, wird folgende Probleme nur allzu gut kennen. Man fährt in den Urlaub und lässt die Rollläden entweder unten, oben oder einen Spalt geöffnet, damit die Pflanzen auch etwas Licht bekommen. Im Idealfall kümmert sich eine Nachbarin um das Problem, doch die optimale Lösung ist das nicht. Es kann aber auch schon einmal vorkommen, dass ganz einfach vergessen wurde, die Rollläden zu schließen, weil die Zeit etwas knapp war. Man bemerkt das üblicherweise erst, wenn es zu spät ist. In diesem Fall wäre es natürlich wünschenswert, die Rollläden irgendwie runterfahren zu können. Ohne zentrale Steuereinheit lassen sich die Probleme jedoch nicht lösen. Denn nur damit kann festgelegt werden, wann die Rollläden hoch- oder runterfahren.

Es gibt auch praktische Erweiterungen, etwa für die tageslichtabhängige Steuerung oder spezielle Smartphone Apps. Doch selbst damit gibt es Einschränkungen und als Software-Entwickler möchte man ja auf keinen Komfort verzichten. Vor allem nicht, wenn man es selbst besser machen könnte. Genau hier beginnt es spannend zu werden, denn wir entwickeln einfach unsere eigene zentrale Steuereinheit, die noch dazu kostengünstig ist.

Das Ziel ist klar definiert: Wir wollen unsere Rollläden per SMS öffnen oder schließen. Um das zu realisieren, benötigen wir jedoch mehr als nur ein Stück Software. Denn irgendwie müssen wir die Rollläden auch ansteuern. Doch welche Hardware kann das? Es scheint ja eigentlich ganz trivial zu sein, denn eine Fernbedienung schickt bestimmte Signale zu den Empfängern der Rollläden. Es sollte also ausreichen, diese Signale aufzuzeichnen, um sie später wieder senden zu können. Dazu wären nur ein Sender für Funksignale und ein Oszilloskop erforderlich, um die Signale aufzuzeichnen. Natürlich muss man die richtigen Signale am Ende noch softwaretechnisch erzeugen. Doch so einfach und günstig, wie sich der Autor das vorgestellt hatte, war es nicht. Es war schon nicht möglich, einen geeigneten Sender für die benötigte Frequenz zu organisieren, ohne gleich eine komplette Schaltung zu bauen.

Weil das Ganze auch ohne notwendiges Expertenwissen nicht umsetzbar gewesen wäre, musste eine andere und vor allem einfache Lösung her. Am einfachsten erschien es, Software-gesteuert einen Tastendruck auf der Fernbedienung durchzuführen. Die

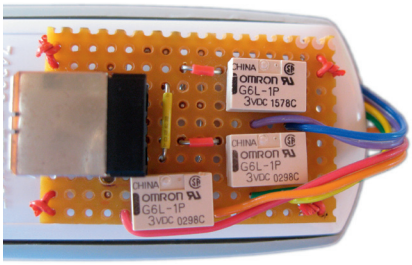

Abbildung 1: Fernbedienung mit aufgesetzter Taster-Steuerung

Fernbedienung war sowieso schon vorhanden und so musste nur überlegt werden, wie eine Taste gedrückt werden konnte und vor allem, wodurch? Dieses Problem war relativ leicht lösbar, da jede Taste auf der Fernbedienung lediglich einen Stromkreis schließt.

Um einen Stromkreis zu schließen reicht es aus, einen Schalter umzulegen, und genau das ist die Aufgabe von Relais. Somit war dieses Problem gelöst und es fehlte nur noch eine Lösung für die Ansteuerung von Relais. Eine übliche Lösung wäre der Einsatz einer seriellen Schnittstelle (RS232) oder eines „USB zu RS232"-Adapters. Für die Ansteuerung der seriellen Schnittstelle existiert mit RxTx [1] eine praxiserprobte Open-Source-Lösung. Bei seriellen Schnittstellen kommt jedoch schnell der Gedanke hoch, dass ein Notebook oder Desktop PC notwendig ist. Die Lösung sollte aber auf keinen Fall mehr Strom verbrauchen als unbedingt nötig, daher musste eine schlanke Alternative her.

Der Raspberry Pi

Spätestens seit der JavaOne 2012 sind Embedded Devices in aller Munde. Allen voran der Raspberry Pi. Ein kreditkartengroßer Computer mit ausreichend Rechenpower und allen notwendigen Schnittstellen wie USB, LAN und HDMI. Auf der Konferenz wurde jedoch vor allem die Verwendbarkeit von JavaFX auf Embedded Devices hervorgehoben. Das beeindruckte einerseits und verhalf andererseits JavaFX zu mehr Akzeptanz. Seit diesem Event haben JavaFX und der Raspberry Pi auch den Autor in ihren Bann gezogen.

Aus diesem Grund war auch für die Rollladensteuerung ein Raspberry Pi (RasPi) als Hardware gesetzt. Ein großer Vorteil war

```
public static void main(String[] pArgs) throws Exception
{
 System.out.println("== GPIO control Test ==");
 GpioController gpio = GpioFactory.getInstance();
 GpioPinDigitalOutput pinDown;
 GpioPinDigitalOutput pinUp;
 GpioPinDigitalOutput pinStop;
 pinDown = gpio.provisionDigitalOutputPin(
        RaspiPin.GPIO_01, "Down", PinState.LOW);
 pinUp = gpio.provisionDigitalOutputPin(
       RaspiPin.GPIO_02, "Up", PinState.LOW);
 pinStop = gpio.provisionDigitalOutputPin(
        RaspiPin.GPIO_03, "Stop", PinState.LOW);
 System.out.println("--> GPIO down ON");
 pinDown.high();
 Thread.sleep(1000);
 pinDown.low();
 System.out.println("--> GPIO down OFF");
 Thread.sleep(5000);
 System.out.println("--> GPIO up ON");
 pinUp.high();
 Thread.sleep(1000);
 pinUp.low();
 System.out.println("--> GPIO up OFF");
 Thread.sleep(5000);
 System.out.println("--> GPIO down ON");
 pinDown.high();
 Thread.sleep(1000);
 pinDown.low();
 System.out.println("--> GPIO down OFF");
 Thread.sleep(2000);
 System.out.println("--> GPIO stop ON");
 pinStop.high();
 Thread.sleep(1000);
 pinStop.low();
 System.out.println("--> GPIO stop OFF");
 Thread.sleep(5000);
 System.out.println("--> GPIO down ON");
 pinDown.high();
 Thread.sleep(1000);
 pinDown.low();
 System.out.println("--> GPIO down OFF");
}
```

Listing 1

```
SerialModemGateway modem;
modem = new SerialModemGateway(
   "modem","/dev/ttyUSB0", 115200,"Cinterion","EGS3");
modem.setSimPin("0000");
modem.setInbound(true);
modem.setSmscNumber("");
Service serv = Service.getInstance();
serv.addGateway(modem);
serv.startService();

//check incoming messages
List<InboundMessage> msgList = new ArrayList<InboundMessage>();
Service.getInstance().readMessages(msgList,
          InboundMessage.MessageClasses.ALL);
String sText;
for (InboundMessage msg : msgList)
{
 sText = msg.getText();

 if ("UP".equalsIgnoreCase(sText))
 {
  up();
 }
 else if ("DOWN".equalsIgnoreCase(sText))
 {
  down();
 }
}
serv.stopService();
```

Listing 2

GPIO Pin	Funktion
1	AB
2	AUF
3	STOPP

Tabelle 1

auch, dass der RasPi problemlos mit dem USB-Port eines WLAN-Routers betrieben werden kann. Der Stromverbrauch ist minimal. Zudem besitzt der RasPi mit den GPIO-Pins eine sehr gute Möglichkeit, digitale Signale zu schalten. Die serielle Schnittstelle beziehungsweise ein USB-Port musste somit nicht belegt werden, um die Relais anzusteuern. Nachdem alle Hardware-Probleme für die Rollladensteuerung gelöst waren, wurde mit etwas Geschick und einem Lötkolben die Hardware zusammengesetzt (siehe Abbildung 1).

Als Relais kamen G6L-1P (3V) von Omron zum Einsatz und für die Verbindung mit dem RasPi wurde ein USB-Port aufgelötet. Für die Steuerung von „Auf", „Ab" und „Stopp" wurde je ein Relais verbaut. Ein USB-Kabel stellte abschließend die Verbindung zum RasPi her. Das Praktische an der Lösung ist, dass die Fernbedienung auch weiterhin manuell bedient werden kann.

Die Software-Ansteuerung

Die Hardware ist eine sehr wichtige Komponente für die Steuerung. Doch ohne passende Software funktioniert auch unsere Lösung nicht. Der erste Test erfolgt mithilfe eines Java-Programms (siehe Listing 1).

Zur Ausführung am RasPi ist ein Java-Runtime-Environment für ARM-Prozessoren erforderlich. Das kann eine JavaSE-Embedded-Version oder auch die JDK8-Early-Access-Version [2] mit Hard-Float-Unterstützung sein. Zusätzlich wird die Open-Source Bibliothek „Pi4J" [3] benötigt, die

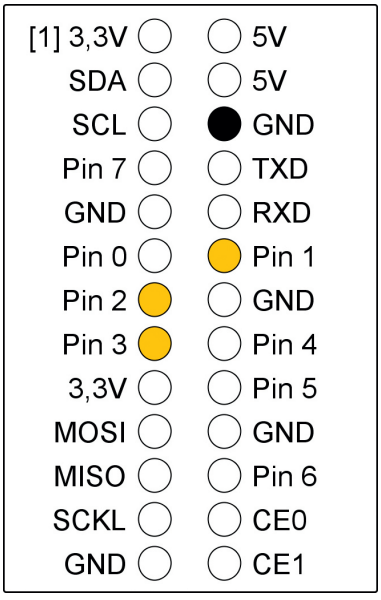

Abbildung 2: Raspberry Pi (Model B), GPIO für Pi4J

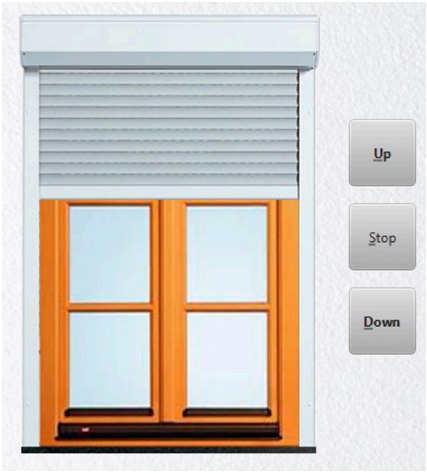

Abbildung 3: JavaFX-Applikation zur Steuerung

den Zugriff auf die GPIO-Pins mit einem einfachen API bietet. Das Testprogramm prüft, ob die Rollläden nach unten beziehungsweise oben gefahren und gestoppt werden können. Es setzt eine definierte GPIO-Pin-Belegung voraus (siehe Tabelle 1). Abbildung 2 zeigt die vollständige GPIO-Pin-Belegung, angepasst an die Verwendung mit Pi4J.

Die größten Hürden sind überwunden und die Steuerung der Rollläden kann bereits durch unsere Software erfolgen. Ein entscheidender Punkt fehlt allerdings noch: Wir wollten SMS-Nachrichten für die Bedienung einsetzen. Damit unsere Software SMS verarbeiten kann, ist weitere Hardware notwendig. Es eignen sich sowohl Mobiltelefone mit integriertem Modem als auch spezielle GSM-Module, wie sie beispielsweise Cinterion anbietet. Es sollte jedoch darauf geachtet werden, dass die eingesetzte Java-Bibliothek etwa mit „SMSLib" [4] die GSM-Hardware unterstützt.

Die GSM-Hardware wird über eine serielle Schnittstelle (RS232) oder einen „USB zu RS232"-Adapter angeschlossen. Die Ansteuerung mit Java erfolgt unter Verwendung von SMSLib (siehe Listing 2).

Je nachdem, welche GSM-Hardware zum Einsatz kommt, kann es zu Problemen bei der Kommunikation kommen. In unserem Fall mussten zusätzliche System-Properties gesetzt sein, damit die Hardware mit dem RasPi funktionierte: "-Dsmslib.serial.polling -Dsmslib.at.wait=500 -Dsmslib.nocops=1"-.

Das definierte Ziel ist damit erreicht. Auch wenn es sich noch nicht um eine vollständige Applikation handelt, sollte es nicht sehr schwierig sein, die einzelnen Teile zusammenzusetzen.

Erfahrungsbericht

Die beschriebene Lösung steuert Rollläden des Autors seit Beginn des Jahres im 7x24-Betrieb und hat bereits den ersten Urlaub erfolgreich gemeistert. Natürlich wurden zahlreiche Erweiterungen integriert, etwa das Setzen der Öffnungs- und Schließzeiten per SMS, das Ein-/Ausschalten der automatischen Steuerung, ein Wochenendmodus etc. Zusätzlich wurde eine JavaFX-Applikation zur Steuerung umgesetzt (siehe Abbildung 3).

Aufgrund der integrierten Effektmöglichkeiten von JavaFX war die Umsetzung des Bewegungsablaufes ein Leichtes und die App bewegt den Rollladen zeitgleich zur Realität.

Fazit

Die Entwicklung der Lösung bereitete großen Spaß, da gerade das Zusammenspiel von Hard- und Software ein spannendes Thema war. Vor allem die Verwendung des RasPi brachte enorme Zufriedenheit und bietet tolle Möglichkeiten zur (Heim-) Automatisierung zu einem unschlagbaren Preis.

Links

[1] RxTx: http://rxtx.qbang.org
[2] JDK8 Early Access: http://jdk8.java.net/fxarm-preview/index.html
[3] Pi4J: http://www.pi4j.com
[4] SMSLib: http://smslib.org/doc/compatibility

René Jahn
rene.jahn@sibvisions.com

René Jahn ist Mitbegründer der SIB Visions GmbH und Head of Research & Development. Er verfügt über langjährige Erfahrung im Bereich der Framework- und API-Entwicklung. Sein Interessenschwerpunkt liegt auf der Integration von State-of-the-Art-Technologien in klassische Business-Applikationen. Unter anderem betreut er die Open-Source-Sparte bei SIB Visions und veröffentlicht regelmäßig Artikel im Unternehmensblog.