FAKULTÄT FÜR !NFORMATIK

Scalimero

TUTORIAL

ausgeführt von

Gabriel A. Grill and Alexander C. Steiner

am:
Institut für rechnergestützte Automation

Betreuer: Ao.Univ.Prof. Dr. Wolfgang Kastner

Wien, 25. 08. 2010

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Inhaltsverzeichnis

Kapitel 1

Introduction

Because of the increasing poularity of home automation, the desire to develop applications for controlling your house became more and more important. Calimero is a collection of Java APIs that together form a foundation for building such applications in EIB/KNX installations. Detailed knowledge of the protocol is not required. On the top of that Calimero only requires J2ME enviorments, which enables use on embedded platforms. It is an open source project and was developed by the Institute of Computer Aided Automation of the Technical University of Vienna.

EIBnet/IP allows you to communicate with components of the widely spreaded EIB/KNX standard by tunnelling over IP Networks. The protocol is published in the KNX Handbook and in the European standard EN 13321-2:2006 (Open Data Communication in Building Automation, Controls and Building Management - Home and Building Electronic Systems - Part 2: KNXnet/IP Communication), available from European standardization organizations. Note: EN 13321-2 is useless without information on the KNX/EIB control network specific data structures (cEMI, DPTs); these should be defined in EN 50090 (Home and Building Electronic Systems).

Since development of Calimero many great projects are using it:

- **KNX@Home**(http://knxathome.fh-deggendorf.de/)
- **BASys**(http://sourceforge.net/projects/basys/)
- **KNXnet/IP Wireshark dissector**(http://knxnetipdissect.sourceforge.net/)
- **CONECT**(http://sourceforge.net/projects/conect/)
- **EIB Home Server**(http://eibcontrol.sourceforge.net/)
- **LEIBnix**(http://leibnix.sourceforge.net/Wikka/HomePage)
- **Sombrero**(http://grill.github.com/sombrero/)
- **Scalimero**(http://grill.github.com/SCalimero/)

Kapitel 2

Installation

First, download the calimero package from http://www.auto.tuwien.ac.at/downloads/calimero-all-2.0a4.zip and unpack it to a detination of your choice. It contains several other archives, including 4 Java Archives (.jar), the source code of all of them and documentation of tools and the library. The core library is calimero-2.0a4.jar, this is the file you want to have on the classpath of all your calimero projects. Calimero-gui and calimero-tools are graphical and command line tools to test calimero and KNX, and to gain a better understanding of calimero by looking at their source code. The calimero-rxtx package contains an optional way of serial port access, which is not needed in most circumstances.

The important thing is to put calimero-2.0a4.jar on your classpath. How to add files to your classpath depends on your development environment and should easily be found in the corresponding manual or home page. It is a good idea to unzip the calimero-2.0a3.zip to a place where you can access the documentation comfortably. It is also useful to get familiar with the command line tools and especially their source code, because it covers some important use cases in a concise way.

Kapitel 3

DSL

3.1 Examples

The scalimero DSL is an easy interface for accessing KNX devices and meant to be used in the Scala interpreter. To use it,

```
1 import tuwien.auto.scalimero.dsl._
```

Then create a network and open it using

```
1 Network("10.0.0.5") open
```

The IP address specified here is the KNX router used to access the network. Then, we need to create some devices.

```
1 val lA = Lamp("1/1/0")
2 val lB = Lamp("1/1/1")
```

Then we can turn the lamps on and off.

```
1 lA turn on
2 lA turn off
```

More general:

```
1 lB send true
2 lB send false
```

We can also read from the devices.

```
1 val b = lA.read
```

Note that b is of type `Boolean`, it gets converted from KNX data automatically.

You can also subscribe to events:

```
lA.eventSubscribe(on) {
  println("lA has been turned on")
}
```

If you want to be notified for every write on the device, subscribe a write callback:

```
lA.writeSubscribe{
  newstatus : Boolean =>
  println("lA status: " + newstatus)
}
```

Kapitel 4

Conclusion and Outlook

Bla Bla