DIY Zoning: Data Acquisition

Table of contents

1. What exactly is covered here

The new code base, DZ3. Older releases are too old to be easily supported on modern distributions, there's been a lot of code rot since they were released.

2. Prerequisites

No matter which way you go (easy or hard), you need these two:

- RxTx for 1-Wire devices and serial servo controllers
- javax.usb for USB servo controllers

Both of these may or may not be available as packages on your platform. On Ubuntu 9.10, where DZ is currently being developed, RxTx installs with simple `sudo apt-get install librxtx-java`, but `javax.usb` requires tinkering.

3. Installation: Easy Way

At this moment, no platform specific packages are available - the code base is moving too fast. As soon as it is stabilized, RPM and Deb packages will be available. For now, go for downloads and get `dz-3.0.INSTRUMENTATION.tar.gz` - it contains everything you need to run instrumentation right out of the box. Connect your 1-Wire network, run `dz-runner`, and RRD databases (located in `./rrd`) will start filling in. Tinker with configuration (read on) to make graphs more civilized - by default, DZ dumps all data channels into one graph.

4. Installation: Hard Way (compiling from source)

4.1. Prerequisites

- Maven
- Subversion

4.2. Fedora 10+

Try to copy this into a shell script and run it:

```
        #! /bin/sh

        yum install maven2
        yum install svn
        cd ${your_development_directory}
        svn co
https://jukebox4.svn.sourceforge.net/svnroot/jukebox4/trunk/jukebox-master
jukebox-master
        (cd jukebox-master && mvn install -Dmaven.test.skip)
        svn co
https://diy-zoning.svn.sourceforge.net/svnroot/diy-zoning/trunk/dz3-master
dz3-master
        (cd dz3-master && mvn install -Dmaven.test.skip)
```

You're done. DZ3 codebase is installed on your box.

4.3. Ubuntu 9.04+

The only difference with the above is that instead of executing

```
        yum install maven2
        yum install svn
```

you execute

```
        sudo apt-get install maven2
        sudo apt-get install subversion
```

5. Minimal Configuration

DZ is dead without configuration. Fortunately, to boot it a very simple configuration file is required. Here it is:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

        <!-- 1-Wire Adapter Driver -->
        <!-- This is a data source for *all* sensor devices on your 1-Wire
bus -->
         <bean id="device_factory"
class="net.sf.dz3.device.sensor.impl.onewire.DeviceFactory"
init-method="start">
                <!-- REPLACE THIS WITH YOUR PORT -->
                <constructor-arg index="0" value="/dev/ttyUSB0"/>
```

```
                <constructor-arg index="1" value="regular"/>
        </bean>

        <!-- Loggers -->
        <bean id="rrdtool" class="java.io.File">
                <constructor-arg type="java.lang.String"
value="/usr/bin/rrdtool"/>
        </bean>
        <bean id="rrdbase_onewire" class="java.io.File">
                <constructor-arg type="java.lang.String"
value="./rrd-onewire"/>
        </bean>
        <bean id="rrdlogger_onewire"
class="net.sf.jukebox.datastream.logger.impl.rrd.RrdLogger"
init-method="start">
                <constructor-arg index="0" type="java.util.Set">
                 <set>
                       <!-- Even though it's just one entry, logger will
create separate channels for each sensor -->
                          <ref bean="device_factory"/>
                 </set>
                </constructor-arg>
                <constructor-arg index="1" type="java.io.File"
ref="rrdbase_onewire"/>
                <constructor-arg index="2" type="java.io.File"
ref="rrdtool"/>
        </bean>

        <!-- JMX configuration -->
        <bean id="jmx-wrapper" class="net.sf.jukebox.jmx.JmxWrapper">
                <constructor-arg index="0" type="java.util.Set">
                        <set>
                                <ref bean="device_factory"/>
                        </set>
                </constructor-arg>
        </bean>
</beans>
```

6. Advanced Configuration

Looking at the example above, you may have already figured that it is based on Spring Framework. Being familiar with Spring and having a brief glance at DZ code base will give you a complete idea of what to do next.

Nevertheless, more configuration examples will be coming up in further sections.

7. Next Step

Now that your computer is busy collecting sensor data, you must be definitely concerned with visualization.