Vicki

01/14/2013

Niels: Updated for Darjeeling

29/3/2013

Penn: Updated for the new wukong-vagrant install

7/9/2013

Hans: Updated for the Mac installation

21/8/2013

Penn: Added several sections, changed title

9/9/2013

KJ: Move source readme to another document

11/1/2013

**WuKong Release 0.2 Installation**

This document is a copy of Vicki’s installation guide for nanokong, and has been modified for Darjeeling. Vicki’s original guide:

https://docs.google.com/document/d/1WTHzUgvvnz6oH2alJtphxkQBpcs4H44HUhc2cXatq-0/edit

# Usage

**If you don’t want to setup your development in your laptop, please visit the Vagrant documentation** [**here**](https://docs.google.com/document/d/1JbPzLLc5KkbjmcLNTPBX443LvUs6-2YERcPsr_r-nf0/edit?usp=sharing)**.**

**0. Install the Toolchain**

1. Overview:

A. USBTiny or USBasp

The USBTiny or USBasp is a USB dongle which can program the Flash at Arduino device by using the SPI interface. It can be attached to the ICSP header and program the internal Flash in the ATmega chip. This is required if we want to install the NanoKong bootloader.

B. **avrdude, avr-gcc, avr-libc, binutils-avr**

Avrdude is the tool that will be used to upload code to the Arduino board. It supports both

programming through the Arduino bootloader, and using external programmers such as

USBTinyISP and USBasp.

C. Others tools

**git, ant, java**

D. Python libraries for WuKong Master

**tornado, jinja2, gevent, configobj, pyserial, lxml, simplejson, pip**

E. FTDI Driver (if you are running Mac)

FTDI usb to serial driver downloaded from <http://www.ftdichip.com/FTDrivers.htm>

F. Arduino IDE

Download the IDE from <http://arduino.cc/en/Main/Software>. Linux user can use apt-get.

G. Arduino code needed before running the gateway. (You can get it from people in lab.)

**Zwave\_bridge.pde**

2. For Linux users: (Tested on debian distribution like Ubuntu 12.04)

sudo apt-get install git-core

sudo apt-get install ant

sudo apt-get install avrdude gcc-avr avr-libc binutils-avr

sudo apt-get install openjdk-6-jdk

sudo apt-get install arduino

sudo apt-get install libevent-dev

sudo apt-get install python-dev // for compiling “pyzwave”

sudo apt-get install python-setuptools // for “pip”

sudo apt-get install libxml2-dev libxslt1-dev zlib1g-dev // for “lxml”

sudo pip install --upgrade pip

sudo pip install configobj simplejson gevent greenlet tornado jinja2 pyserial lxml

Note I: run sudo apt-get update If you encounter any problems regarding version conflict.

Note II: If your Linux distribution cannot detect your devices including Arduino or WuDevice, you may need to add rules to Udev which is the device manager of Linux kernel to create nodes in /dev. The Udev rules themselves are located in /etc/udev/rules.d To support the WuDevice which may use USBtinyISP or USBasp programmer, add one new file 99-avr.rules with following lines.

# Comment: USBtinyISP Programmer rules

SUBSYSTEMS=="usb", ATTRS{idVendor}=="1781", ATTRS{idProduct}=="0c9f", MODE="0666"

SUBSYSTEMS=="usb", ATTRS{idVendor}=="16c0", ATTRS{idProduct}=="0479", MODE="0666"

# Comment: USBasp Programmer rules

SUBSYSTEMS=="usb", ATTRS{idVendor}=="16c0", ATTRS{idProduct}=="05dc", MODE="0666"

and then restart udev:

sudo udevadm control --reload-rules

For those devices with unknown idVendor and idProduct, use the following command to get the IDs.

lsusb

The sample output would be like this.

Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub

The first ID 8087 is idVendor while the second ID 0024 is idProduct.

3. For Mac users:

A. ant, git, java

<http://ant.apache.org/manual/install.html>

<https://help.github.com/articles/set-up-git>

<http://www.java.com/en/download/faq/java_6.xml>

B. avrdude

Step 1: Download CrossPack 2012-11-28, including avr-gcc 4.6.2, avrdude 5.11.1, avr-libc 1.8.0...:g

<http://www.obdev.at/products/crosspack/download.html>

CrossPack doesn’t come with avrdude that supports usb and handles usbasp programmer, please go to step 2 to install the latest avrdude with usb support. After installing CrossPack, please restart your terminal.

Step 2: Download from homebrew. Install homebrew with the following command:

ruby -e "$(curl -fsSL https://raw.github.com/mxcl/homebrew/go)"

( Learn how to use homebrew here: <http://mxcl.github.com/homebrew/> )

We can install avrdude with this command:

brew tap homebrew/dupes

brew install avrdude --with-usb

~~And installing the rest with the following commands~~

~~brew tap larsimmisch/avr~~

~~brew install avr-libc~~

C. Python Libraries

sudo easy\_install configobj

sudo easy\_install simplejson

sudo easy\_install lxml

sudo easy\_install jinja2

sudo easy\_install gevent or (pip install gevent)

sudo easy\_install pyserial

sudo easy\_install tornado

D. Note: If easy\_install cannot make a complete installation, please use pip install instead.

E. FTDI Driver (in order to show USB adapter port name under /dev)

FTDI usb to serial driver downloaded from <http://www.ftdichip.com/FTDrivers.htm>

which one? use VCP

Note:

We work in the **develop branch**. We follow the git-flow workflow. The latest ‘stable’ release is release 0.2.

*Once you have everything ready, you can start the following steps ...*

**1. Download the source code**

Clone the source code and download to your preferred working directory:

git clone https://github.com/wukong-m2m/wukong-darjeeling

**2. Build and upload the bootloader**

**0. Make sure you have avrdude installed before this step.**

**1. Go to the bootloader directory**

cd <dj root>/wukong/tools/arduino-stk500v2-bootloader/

**2. Modify the Makefile before uploading the new bootloader.**

You should modify the makefile to specify which programmer you are using (either USBtiny or USBasp).

vi Makefile

In the upload\_bootloader part at the very bottom of the file, change the parameter after “-c” to either usbtiny or usbasp according to your programmer. You should change four of them under the upload\_bootloader.

**3. Connect your USBTinyISP or USBasp programmer to ICSP port at the corner of WuDevice (2x3 pins)**

The pin 2 of ICSP should be connected to VCC of the programmer.

**4. Build and upload the bootloader.**

make upload\_bootloader

For more detail about what make upload\_bootloader is doing, take a look at the following notes. Otherwise, you can skip the note part.

**Note:**

USBTiny or USBasp-based ISP programmer is necessary to install the WuKong bootloader into the WuDevice. If you want to install WuKong into a new device, “Build and download bootloader” is necessary. However, it is not necessary if you get a unit of WuDevice because the bootloader has been installed already. The WuKong bootloader is required for the over-the-air updating. It is compatible with the arduino bootloader.

The following steps are the details of make upload\_bootloader. (In /vm/build/avr\_mega2560/Makefile)

(1) Firstly, we need to enable the write permission of the bootloader sector. It is disable by default for the safety issue.

# avrdude -p atmega2560 -P usb -c usbtiny -e -U lock:w:0x3F:m -U efuse:w:0xFD:m -U hfuse:w:0xD8:m -U lfuse:w:0xFF:m

(2) Secondly, we can burn the NanoKong bootloader.

# avrdude -p atmega2560 -P usb -c usbtiny -D -V -U flash:w:../../../tools/arduino-stk500v2-bootloader/stk500boot\_v2\_mega2560.hex:i

(3) Third, we need to install the Flash programmer which can be executed in the runtime to change the Flash content. The Atmega divide the Flash into two parts so that we can program the Flash while we read instructions from it. The Flash programmer is located in the NRWW section of the Flash which is readable when we program the Flash.

# avrdude -p atmega2560 -P usb -c usbtiny -D -V -U flash:w:NanoVM\_NRWWsection.hex:i

(4) Finally, we should disable the write permission.

# avrdude -p atmega2560 -P usb -c usbtiny -U lock:w:0x0F:m -U hfuse:w:0xD0:m

**3. Build Darjeeling VM for the WuDevice**

**1. Modify the ant build.xml file for the wunode config.**

Go to <dj root>/src/, copy settings.xml.dist to settings.xml, and modify the configuration file.

cd <dj root>/src/

cp settings.xml.dist settings.xml

vi settings.xml

~~Go to <dj root>/src/config/wunode, copy build.xml.dist to build.xml, and modify the configuration file.~~

~~cd <dj root>/src/config/wunode~~

~~cp build.xml.dist build.xml~~

~~vi build.xml~~

In the avrdude target, change the usb port (after the -P parameter) to the usb port connected to the device. For linux, it should be something like /dev/ttyUSB0. For Mac, it should be something like /dev/cu.usbserial-A98FRT1X.

**2. Build the Infuser:**

Compiling the VM is done in a few phases. First, the Java program is compiled using the normal Java compiler. The infuser, a tool written in Java that comes with Darjeeling, will then process this classfile and convert it into a binary image called a infusion (<app>.di) in the application’s build directory (<dj root>/src/app/<app>/build). This is then converted into an archive file in the config directory <dj root>/src/config/wunode/app\_infusion.dja.

What happens next will depend on the platform. If we run Darjeeling on a PC, it will just read this file, but since we’re running it on the WuDevice it will first get exported as a c file called temp\_di\_app\_infusion\_archive.c in the same directory. Finally the actual VM sources are compiled, including temp\_di\_app\_infusion\_archive.c to produce a single image which includes the Java program and can be uploaded to the nodes.

To compile the VM, first we need to build the Infuser:

cd <dj root>/src/infuser

ant

**3. Compile and build Darjeeling**

In order to build Darjeeling for the WuDevice

cd <dj root>/src/config/wunode

ant

You can find the darjeeling.ihex, and darjeeling\_eeprom.ihex. The first one is the VM itself, the second one is data that will be placed in EEPROM.

**4. Upload/Install the Darjeeling**

In the same directory, the Darjeeling VM can be downloaded into the board by the arduino bootloader. Avrdude can be used to do this job. Connect GND of the program downloader to the leftmost pin of JP2 male header (based on the direction of label “JP2”). Note that JP15 should have a jumper on it to enable programming.

ant avrdude

**5. See the program’s output**

To check if avrdude is uploaded successfully, you can see the output the Arduino produces by connecting to the usb port with screen:

screen /dev/tty.usbmodem411 115200

115200 is the baud(If permission denied, please add sudo). You can replace the tty.usbmodem411 part with whatever is your usb port is (like /dev/ttyACM0 or /dev/ttyUSB0 for Linux). While screen is connected, you can’t upload new code with avrdude, so you need to exit screen first (Ctrl-A k y). If your terminal cannot find the command screen, please try this (brew install screen for Mac and sudo apt-get install screen for Ubuntu) before your execution of screen.

If we compiled the helloworld application (the application is selected in

<dj root>/src/config/wunode/build.xml) it should print “Hello world.”

**4. Run the WuKong Master**

**1. Compile z-wave for python**

cd <dj root>/wukong/tools/python/pyzwave

sudo python setup.py install

If you want to uninstall, you need to remove all files manually and also undo any other stuff that installation did manually. To know the list of all installed files, you can reinstall the package with --record option and delete it:2

sudo python setup.py install --record files.txt

cat files.txt | xargs rm -rf

**2. Make the current WuDevice a gateway/base station**

If you are using usb dongle as a gateway, please go to step 3 directly.

Upload the arduino code "/tools/zwave/arduino/Zwave\_bridge/src/Zwave\_bridge/Zwave\_bridge.ino" to the device through Arduino IDE. Before uploading, be sure to select “Arduino Mega2560” under Tools/board and also your serial port under Tools/Serial Port.(If you don’t have Arduino IDE, please go to <http://arduino.cc/en/Main/Software> )

**3. Modify the configuration file and set the gateway usb port.**

Go to <dj root>/wukong/config. If there is no master.cfg under the directory, you can create a copy of master.cfg as a basis.

cd <dj root>/wukong/config/

optionally(suggested): cp master.cfg.dist master.cfg

vi master.cfg

In the config file, change the zwave\_gateway\_ip to specify the WuDevice that serves as the gateway/base station of your sensor network. You can find your usb port by ls /dev.

ZWAVE\_GATEWAY\_IP = /dev/cu.usbmodemfd131

note: don’t put any comments in the same line after ZWAVE\_GATEWAY\_IP

**4. Run the master**

cd <dj root>/wukong/master/

python master\_server.py

**5. Start building your applications and managing your node sensors in you sensor network.**

Open the browser and go to the following link: <http://127.0.0.1:5000/>.

You can copy the folders inside <dj root>/wukong/apps\_example to <dj root>/wukong/apps for some initial examples of FBP program.

**5. Testrtt Mode (Alternative method to manage sensor network)**

Besides adding or deleting node device through the WuKong Master website, we can also manage the sensor network by using testrtt command.

**1. Compile testrtt.c codes**

cd <dj root>/wukong/tools/testrtt/

gcc testrtt.c -o testrtt

**2. Run testrtt code and specify the usb port of the gateway device**

./testrtt -d /dev/cu.usbserial-A98FRT1X

**3. Supposed you are in the testrtt mode, you can start managing your sensor network.**

Here are some easy steps and commands that you can use to add or delete nodes under the testrtt mode. For more information, look at <dj root>/tools/README

Step 1: Show all nodes in the network.

**controller initdata**

// If there’s no node device, it will show one X, which represents the gateway WuDevice itself.

Step 2: Turn on the learning mode of the gateway to add the node.

**network add**

// Gateway will start searching WuDevice with ground interrupt thru Zwave communication.

Step 3: Ground interrupt.

Connect a wire to the ground pin while the other side touches the interrupt pin of the

WuDevice, which you want to add to the network. The testrtt environment will print out some

lines that represent this device has been added as a node successfully.

Step 4: Turn off the learning mode of the gateway to delete the node.

**network stop**

Step 5: Show all nodes in the network to check whether the node is added.

**controller initdata**

// One X should be added to the network.

Step 6: Turn on the learning mode of the gateway to delete the node.

**network delete**

// Gateway will start searching WuDevice with ground interrupt thru Zwave communication.

Step 7: Ground interrupt.

Connect a wire to the ground pin while the other side touches the interrupt pin of the

WuDevice, which you want to delete from the network. The testrtt environment will print out

some lines that represent this device has been deleted successfully.

Step 8: Turn off the learning mode of the gateway to delete the node.

**network stop**

Step 9: Show all nodes in the network to check whether the node is deleted.

**controller initdata**

// One X should be delete from the network.

**4. Quit testrtt mode by Ctrl-C.**

**6. Run Darjeeling on the WuDevice**

We can run Darjeeling on an WuDevice, or on a normal PC. To run Darjeeling on a PC:

**1. Prepare another WuDevice as gateway/base station**

Follow the same steps described in section 4.2. Darjeeling will use this WuDevice to access the Zwave network, just like the master does.

**2. Build the native config:**

In Darjeeling a “config” defines how the VM is compiled by selecting an architecture, platform, application, and the libraries that will be included. The “native” config is set up to run Darjeeling on a PC.

cd <dj root>/src/config/native

ant

In the wunode config, two files were created: darjeeling.ihex, and darjeeling\_eeprom.ihex that contained the binary image that we upload to the node with avrdude.

For the native config a normal executable file is created: darjeeling.elf.

**3. Run the VM:**

Since the PC doesn’t have a Zwave radio, we will use the node prepared in step 1. When we start the VM, we need to tell it where it can find the Zwave radio by passing a command line option:

./darjeeling.elf -u 2=/dev/cu.usbserial-A9CNZHXH

You will have to replace the cu.usbserial-A9CNZHXH with the name of the WuDevice. Only the cu. version will work, the tty. will not.

If <dj root>/src/config/native/build.xml is set up to use the helloworld app, the VM should now print “Hello world.”

After this it will not return, but keeps listening to the Zwave radio to receive commands from the master. You can run a master and VM on your PC at the same time using two different WuDevices. They cannot share a single radio.

Press Ctrl-C to quit.

## How WuKong Deploys an Application

To answer this question, we first have to understand some basic terminology which will be used throughout this article.

When we talk about java application, we mean the WKDeploy.java and WKDeploy.xml generated by xml2java/generator.py.

When we talk about application specific components, we mean the WKDeployCustomComponents.xml in the WuKong Master apps’ directory.

When we talk about app deploy directory, we mean “{darjeeling-root}/src/app/wkdeploy/java”

So when you first saved your FBP and hit deploy after discovery, a number of things will follow in the Master. First it will parse the application xml into format that the Master understands, cleans up the “{darjeeling-root}/src/app/wkdeploy/java” directory, generates the corresponding java application to the directory, call codegen on application specific components to generate the parent abstract java classes for virtual wuclasses written in java prefix with “GENERATED”.

By this point of time, only WKDeploy.java, WKDeploy.xml, WKDeployCustomComponents.xml should be inside “{darjeeling-root}/src/app/wkdeploy/java”, all global wuclasses java implementations and parent classes are inside “{darjeeling-root}/wukong/javax/wukong/virtualwuclasses”, app specific wuclasses java implementations are in respective “{darjeeling-root}/wukong/apps/[app\_name]” app directory.

However, contrast to how global java wuclasses are structured, app specific java wuclasses parent classes are generated to “{darjeeling-root}/src/app/wkdeploy/java” instead, so this way app specific wuclasses will not leak into other applications.

Master will then run the “ant” command in the app deploy directory to start compiling the java application including everything mentioned above into a lower bytecode representation with extension ‘dja’ which will be uploaded by Master to all nodes mapped during the process.

The app is then deployed. Hurray~!

# Contributors

\*This comment will be deleted after everyone has written his/her name below. Please write your name below if you don’t see your name.

\*e.g. Penn Su

\*e.g. Niels Reijers

Penn Su

Sen Zhou

Niels Reijers

# Contributing

1. Paste an issue to our issue tracker and discuss with any of the contributors above first
2. Fork the repo
3. Create a new feature branch (if you expect to change a large portion of the code and abstractions)
4. Commit your changes
5. Issue a pull request

We don’t have any strict guidelines on the code style, but we absolutely don’t tolerate unreadable, obscure, misleading code.