User Guide

Overview

This system was tested on the following equipment:

- PC with ubuntu 14.04
- Border router device (Zolertia RE-MOTE)
- Texas Instruments SensorTag CC2650
- Android Phone

Prerequisite

Ericsson Forge SSH key

Some of the project's git repositories are located in Ericsson forge and therefore a ssh-key is required for downloading these git repositories. If there is not an existing ssh-key on the computer it can be created by entering the following command to gnome-terminal.

```
$ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

The following options are presented when entering the previous command. If security is not a concern then every question can be left black and proceed by only hitting enter for every question.

```
Enter a file in which to save the key (/Users/USERNAME/.ssh/id_rsa): [Press enter]
Enter passphrase (empty for no passphrase): [Press enter]
Enter same passphrase again: [Press enter]
```

Next the public key needs to be copied to the Forges key log. Open the public key file with following command and copy everything with `ctrl+a`, `ctrl+c`.

```
$sudo gedit ~/.ssh/id_rsa.pub
```

Go to Ericsson Forge and under the Username account settings there is option to add ssh key (Username -> My Account -> +Add Keys). Paste the copied content of the public key file and save the key.

GNU ARM Embedded Toolchain

Contiki and sparrow border router used in this project uses ARM compiler. GNU ARM Embedded Toolchain can be downloaded from:
https://launchpad.net/gcc-arm-embedded/+download
This project has been tested by using GNU ARM Embedded Toolchain version 5.4.1. To install ARM, all that needs to be done is to download the linux tarball and set an environment variable to the installation folder. In this example the tarball is extracted to `~/projectCS/` -folder. To setup an environment variable open the hidden userprofile file:

```
$sudo gedit ~/.profile
```

Next the path to the ARM compiler needs to be added to the environment variable. It is done by adding the following lines to the end of the profile file and saving the file. *Note:* Be careful with this step, it is possible to wipe the whole path empty and lock the user out of the account.

```
#ARM Compiler
export PATH=$PATH:"/home/USERNAME/
 projectCS/gcc-arm-none-eabi-5_4-2016q3/bin"
```

After modifying the profile file a restart is needed for the changes to take effect.

Installation

Installation includes Content centric network environment which is used instead of regular TCP/IP network, Mongo database for storing data, Border router device for making a wireless connection to nearby sensors, SensorTag -sensor for producing sensor values and Android application for displaying the data.

Download and installation

In this user guide everything is downloaded and installed in one working directory called “projectCS” under the Ubuntu 14.04 operating system home directory. Directory can be created through the gnome-terminal with command:

```
$mkdir ~/projectCS
```

CCN-Lite

This project uses heavily modified version of CCN-lite Project. Modified version adds auto connection between relays, support for database and sensor directory service and other functionalities to the CCN-lite software.

CCN-lite requires OpenSSL to run. Installing OpenSSL is done with the following command:

```
$sudo apt-get install libssl-dev
```

To start, the project git needs to be cloned to the computer. Note that in the git clone -command the “YOURUSERNAME” needs to be changed to an existing BitBucket username.

```
$cd ~/projectCS
$git clone https://YOURUSERNAME@bitbucket.org/
  MaxWijnbladh/ccn_lite_greeniot.git ccn-lite
```

Before the project can be compiled a environment variable needs to be set to the project folder. Path is added to a hidden profile file that is personal for every user on Ubuntu operating system. To open the profile file, run:

```
$sudo gedit ~/.profile
```

Next the path to the git repository needs to be added to the environment variable. It is done by adding the following lines to the end of the profile file and saving the file. Note: Be careful with this step, it is possible to wipe the whole path empty and lock the user out of the account.

```
#CCN-Lite
export CCNL_HOME="/home/USERNAME/projectCS/ccn_lite_greeniot"
export PATH=$PATH:"$CCNL_HOME/bin"
```

After modifying the profile file a restart is needed for the changes to take effect. Now only thing left to do is compile the ccn-lite software.

```
$cd ~/projectCS/ccn-lite/scr
$make clean all
```

Refer to the Usage part of this document for running a ccn-lite relay.

Texas Instruments SensorTag - CC2650

Installing Contiki - Contiki CCNLite

```
$cd ~/projectCS
$git clone ssh://gitolite@forge.ericsson.net/
 uppsalaproj2016/contiki-ccnlite.git
$cd contiki-ccnlite
$git checkout icn2016
$git submodule sync
$git submodule update --init
$cd apps/ccn-lite
$git clone ssh://gitolite@forge.ericsson.net/uppsalaproj2016/
 ccnlite-contiki.git ccn-lite
$cd ccn-lite
$git checkout icn2016
```

Updating Sensor Firmware

You need to download XDS Emulation Software Package

- go to 32-bit: http://software-dl.ti.com/dsps/forms/self_cert_export.html?prod_no=ti_emupack_setup_6.0.407.6_linux_i386.bin&ref_url=http://software-dl.ti.com/dsps/dsps_public_sw/sdo_ccstudio/emulation

-OR-

- go to 64-bit: http://software-dl.ti.com/dsps/forms/self_cert_export.html?prod_no=ti_emupack_setup_6.0.407.6_linux_x86_64.bin&ref_url=http://software-dl.ti.com/dsps/dsps_public_sw/sdo_ccstudio/emulation

Download it to `~/projectCS`

```
$cd ~/projectCS
$sudo ./ti_emupack_setup_6.0.83.0_linux_i386.bin
$cd /opt/ti/ccs_base/common/uscif/xds110/
$sudo ./xdsdfu -m
$sudo ./xdsdfu -f firmware.bin -r
$sudo ./xdsdfu -m
```

Flashing Tool For SensorTag

- Navigate to: http://software-dl.ti.com/dsps/forms/self_cert_export.html?prod_no=uniflash_3.4.1.00012_linux.tar.gz&ref_url=http://software-dl.ti.com/ccs/esd/uniflash/

- Save the file to `~/projectCS`
- Extract the uniflash_3.4.1.00012_linux.tar file at `~/projectCS`

```
$cd ~/projectCS
$sudo ./uniflash_setup_3.4.0.00003.bin
```

Compile a .elf file for TI CC2650

- Navigate to the directory of the file you want to compile

```
$cd ~/projectCS/contiki/examples/icn2016-demo/
$sudo make TARGET=srf06-cc26xx
 BOARD=sensortag/cc2650 icn2016-mote.elf
```

- This process will create an .elf file which is the format it should have to be uploaded in the sensor

Flashing the Sensor

- Run uniflash from the Desktop
- Go to File -> New Configuration
- On Connection Field choose: **Texas Instruments XDS110 USB Debug Probe**
- On Board or Device Field Choose: **CC2650F128** and click **OK**
- Click on **Erase Entire Flash**
- Then click on **Programs** from the Dropdown list on the left
- Click **Add** and navigate to the location of the file
- After you select, you will be back on the previous menu, click **Program** to flash the sensor

Sometimes the sensor needs to be unplugged and replugged during erasing and flashing.

Border router device - Zolertia RE-MOTE

Border router software were delivered for this project by SICS and the final version of this project ended up using just slightly modified version of SICSs original code.

Download/Installation

Border router software needs ARM compiler. Refer to the prerequisite section for more information.

Sparrow git with submodules are needed to run the border router.

```
$cd ~/projectCS
$git clone https://github.com/sics-iot/sparrow.git
$cd sparrow/products/sparrow-border-router
$git submodule sync
$git submodule update --init
```

If SDS from this project is going to be used then some files are needed to be changed in order for the Register Service to work. Modified files are located in a zip-file inside the ccn_lite_greeniot git used in CCN-Lite section. Refer to CCN-Lite section for more information if that part was skipped while following this user guide. Zip-file can be found from:

```
$cd ~/projectCS/ccn-lite/scr/util/SDS/RegisterService/sparrow.zip
```

Extract sparrow.zip to projectCS folder and the files should be extracted into the right places in the sparrow git folder. In unlikely case where this does not work follow the README_GREENIOT -readme file from sparrow.zip to see where the files should be extracted.

Flashing the border router device

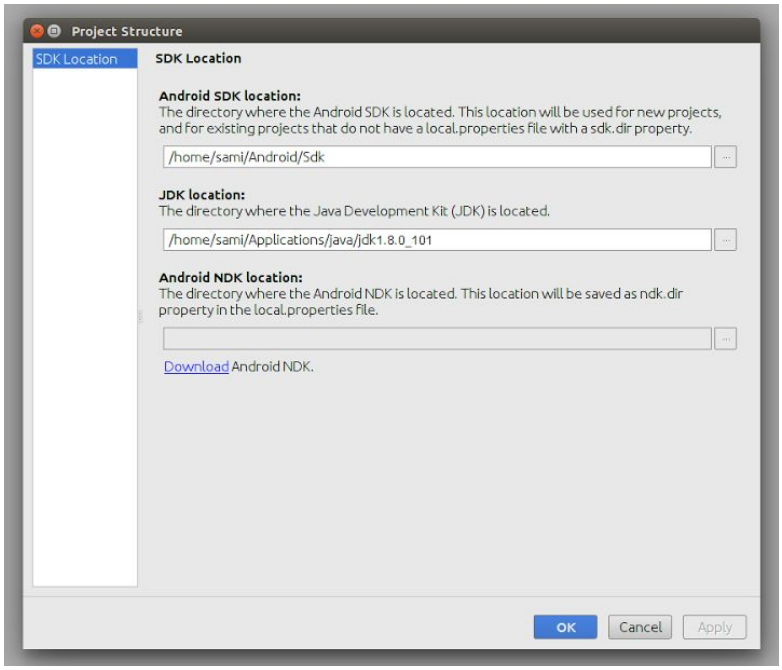
If changes are dome into the border router code it needs to be flashed. Flashing this border router is much more simpler than flashing the Sensor Tag device.

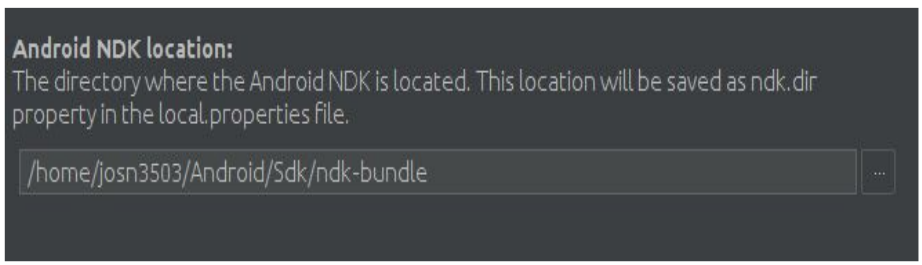
```
cd ~/projectCS/sparrow/products/sparrow-serial-radio
make TARGET=zoul-sparrow IMAGE=1
make TARGET=zoul-sparrow rescue-image
sudo make TARGET=zoul-sparrow upload-rescue-image
```

Note: This may need to be done for a clean clone of the git repository before extracting the files that are part of this project.

Android

Download and setup Android Studio for Ubuntu

1. Download Android studio here: https://developer.android.com/studio/index.html
2. You can find and install guide here: https://developer.android.com/studio/install.html
    a. First instal lJava > 1.8
    b. Go through the first two bullets
        i. Unpack the `.zip`, put the `android-studio` in the `/usr/local/` directory
        ii. To launch Android Studio cd to `/usr/local/android-studio/bin` then type in `./studio.sh`
        iii. Follow the Android Studio setup wizard and you are done with the installation
        iv. You may need additional libraries, to install then execute: `sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0 lib32stdc++6`
3. We need Android NDK to use C files in our project, here is the information to install it along with the other libraries the project needs.
        i. Go to `File-> Project Structure`
        ii. You should see this
        iii. 

        iv. Under the heading android NDK location click download and follow the instructions
        v. 

b. Download the rest of the libraries
    i. Go to `Tools->Android->SDK Manager` and click
    ii. Make sure you have the following thing installed
    iii. Under the tab `SDK tools`:
        1. `Android SDK Tools`
        2. `Android SDK Platform-tools` <version number>
        3. `Android SDK Build-tools` <version number>
    iv. Under the tab SDK platforms
        1. Android 6.0 - API 23
        2. Android 5.1 - API 22
        3. Android 5.0 - API 21
        4. Android 4.3 - API 18
        5. Android 4.0 - API 15

Setup the project

1. Pull the project from the online repository:
    a. cd in terminal to the `AndroidStudioProjects`
    b. `git clone https://Aranor@bitbucket.org/Aranor/ccn-lite-android.git`
2. Import project to Android Studio:
    a. In Android studio: go to `File->New->Import project` click
    b. Find the `ccn-lite-android` project in the `AndroidStudioProjects` folder, click ok
    c. There should now be a window with the project and you should be able to see the files in the project window
3. Use ndk-build command after any change to the C files:
    a. In the terminal (either in android studio or a normal terminal window):
        i. Cd to app/src/main
        ii. Run: ndk-build
        iii. Then you can run the project

You are now ready to develop in the Android Studio environment.

Relay service

In order for the application to work normally, you need to have the service relay running on the phone as well.

Mongo Database

Mongo Database was used for storing data from the sensors and from the GreenIoT network.

First the MongoDB public GPG Key is imported for the system:

```
$sudo apt-key adv --keyserver hkp://keyserver.ubuntu
 .com:80 --recv 7F0CEB10
```

Then a list file is created for the Ubuntu 14.04 with the following command:

```
$echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.0
 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list
```

Finally the local packages are reloaded and MongoDB packages are installed with the following commands

```
$sudo apt-get update
$sudo apt-get install -y mongodb-org
```

To open the database for all remote connections mongod.conf need to be modified. In this user guide the config file is opened by using Ubuntu default text editor, *gedit*, by the following command:

```
$sudo gedit /etc/mongod.conf
```

The following section of the config file need to be edited as shown here:

```
#network interfaces
net:
  port: 27017
  bindIp: 0.0.0.0
```

To add a MongoDB support for the python scripts that are used to update and control the database, the following commands are needed:

```
$sudo apt-get install python-pip
$sudo pip install pymongo
```

Usage

Database

To start MongoDB on the database computer write the following command into the terminal:

```
$sudo service mongod start
```

To open the mongodb interface,write the following command into the terminal:

```
$mongo
```

Two python programs (*sensordata.py* and *nfn.py*) run on the database machine. These can be found in the *py/server* folder within the *src* folder:

```
$cd $CCNL_HOME/src/py/server
```

Both files have a config file that can be found in the same folder. However, these do not need to be altered as long as the mongodb database is installed and running on the same machine as the python programs.

DS

Directory service is a python script that was downloaded with our CCN-Lite repository. In this project it is called Sensor Directory Service (SDS) and it can be found from the ccn-lite folder:

```
$cd $CCNL_HOME/src/util/SDS
```

DS need to be configured before it can be used properly. The only part for a normal user that need be changed is the database ip part. The configure file can be found under the config -folder and this user guide uses gedit to open it:

```
$gedit config/config.ini
```

Next DS is ran by python command:

```
$python SDS.py
```

Notes:
(1) SDS.py needs to be already running before starting any CCN relays for it to be able to create a map of the network.
(2) Part of the DS runs on the same machine as the border router is running. Further information see Border router section.

CCN-Lite

The relays in our network need to be started from the *src* folder.

```
$cd $CCNL_HOME/src/
```

The CCN-Lite tutorial and documentation provided by the University of Basel has a thorough explanation of all the CCN-Lite functionality [1]. The testbed used in this project can be created by simple running a script:

```
$cd $CCNL_HOME/src/util/scripts
$./start_demo
```

Border router

After connecting the border router device through an USB cable to the computer the border router software can be compiled.

```
$cd ~/projectCS/sparrow/products/sparrow-border-router
$make connect-full
```

After the border router has started up, the following commands can be used to see if there is any neighbours connected to the device and if there is any route existing to the neighbour through the border router device:

```
nbr
routes
```

Part of the DS is ran on the same computer where a border router is running. To start the config file need to be prepared. Config file includes description for every value for user to modify and it is found from register services config folder:

```
$cd $CCNL_HOME/src/util/SDS/RegisterService
$gedit config/re_config.ini
```

After defining location and the other values for the configure file, the register service is ran.

```
$python RegisteryService.py
```

Android

In order for the Android application to work, the relay service needs to be running in the background and its gateway need to be pointing an ip within the CCN network.

References

[1] CCN-lite and NFN Tutorial
https://github.com/cn-uofbasel/ccn-lite/blob/master/tutorial/tutorial.md