

# Porting Smart-M3 Platform to MeeGo Operation System

Kirill Yudenok

St. Petersburg Electrotechnical University

St. Petersburg, Russia

kirill.yudenok@gmail.com

## Abstract

Smart-M3 is a platform for development and deployment of smart spaces application. The main goal of this project is to investigate the programming interfaces and architecture of the Smart-M3 platform and adapting this platform to launch on the MeeGo operating system.

**Index Terms:** Smart-M3, MeeGo, Smart Space.

## I. INTRODUCTION

In the past few years has substantially increased to the scientific community to the technology provided by the so-called stack technology Web 2.0, which focused on user involvement in the generation of content distributed service architecture building applications, context-sensitive information processing and human-machine interface. First of all, this is due to the proliferation of mobile devices and wireless communication capabilities that allow users resides in the "online" and be integrated with the global information resources and services. One of the key research directions in this area is the concept of intelligent semantic spaces (smart spaces).

Platforms for development and deployment of smart spaces, there are not many, and among them the most popular gathering open platform Smart-M3.

At the moment did not exist any means for automatic installation of the platform Smart-M3 on the MeeGo operating system, except hand-built platform from source and install additional components of the platform. The aim of this project is, just as in the preparation of installation packages for the new operating system installation MeeGo and the fact that the platform works correctly with the MeeGo.

Adaptation platform launch Smart-M3 on the MeeGo operating system will allow developers to design their applications using the platform and transfer existing applications to new operating system. Creating components of smart spaces Smart-M3 platform for the MeeGo operating system adds a new opportunity both to develop new services for the system and study area for intelligent spaces in general.

Solution for investigation of the platform Smart-M3 was the need for a toolkit for developing a new operating system MeeGo from developers who use the platform in the industry, as well as new, looking to master a new platform for their problems. With a new niche for the development of smart space applications, the platform is Smart-M3 broaden the scope of application and use of devices in different formats.

## II. WORK PROBLEMS

There are several challenges in creating software components of the platform for other operating systems, namely the dependence on other system components that ensure the correct operation of the platform. Typically, these components are systemic and are included in other systems, but there are additional components that must rebuild and adjust depending on the system. For architectures such as ARM, shows the same problem.

In systems with open source, this problem can be solved quite simply, the creation or transfer of these components on another system by assembling and setting its source. In our case, the Smart-M3 platform distributed under an open source license and its code is available for modification and analysis.

In my opinion, the absence of some system components depends on the version of the system for which the product is developed, and some additional components that are independent of the system, installed by hand. MeeGo operating system is precisely such a case, as it is in the final stage of testing.

Smart-M3 platform is distributed under an open source license as source code, the approaches to the use and dissemination platform, there are only two, namely:

1. Assembly and installation of the Smart-M3 on any operating system and under any architecture from source code. But since it may depend on many other components of the system, they in turn will also need to install manually, or collect, if they are absent.
2. Creating a platform installation packages for different operating systems and architectures. This approach automates the process of assembling and installing the components, since packets are based platforms. But this approach does not eliminate the problem of dependence on other system components.

## III. RELATED WORK

To run the application on the MeeGo operating system, which are written using a platform Smart-M3, needed platform itself Smart-M3 and additional libraries. Currently, the MeeGo operating system no installation packages for the platform Smart-M3, and the goal was to examine existing packages, Smart-M3 platform and adapted to run on MeeGo. This section will be fully described in the course of events on the transfer of existing components of the client platform Smart-M3 for the operating platform MeeGo.

### A. Smart-M3 platform modules.

Platform Smart-M3 consists of modules, shown in Fig. 1. The figure shows all the modules and their interactions, modules are marked in grey on a portable operating system MeeGo.

1. *Modules of client-side: libwhiteboard* module provides helper functions for other modules and, therefore, is required for all other modules, but Python KP. Module whiteboard-daemon should be running in the background, so it requires all modules, if you are running any KP. Currently, there are three modules of the transport level, namely sib-access, sib-access-nota and sib-access-plain-nota. The system designer must select at least one of the transport modules, depending on the choice of transport.

Additionally the user can choose to install the Smart-M3 Qt API. Module Smart-M3 Qt Core provides the Qt API-interfaces for the detection of SmartSpace and its basic operation. Module Python KP has no requirements for other modules.

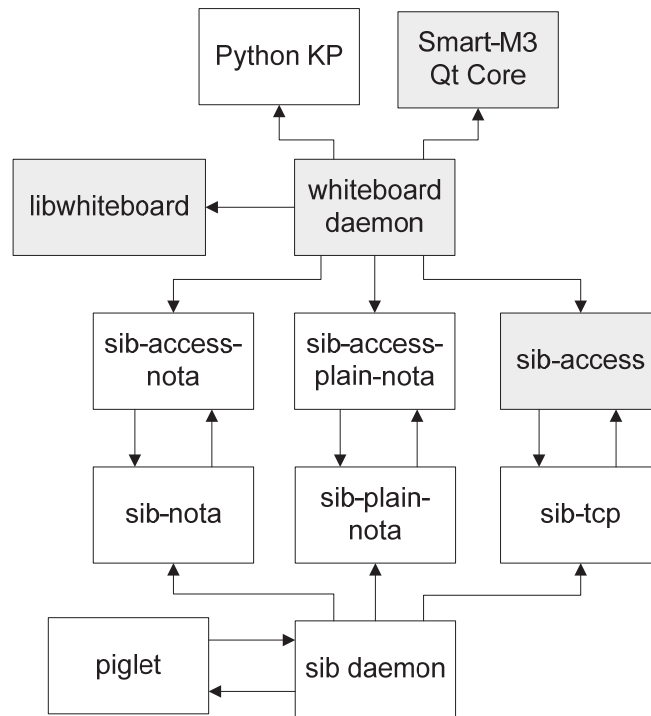


Fig. 1. Software components of the Smart-M3 platform

2. *Modules of SIB-side:* If the system designer wants to run the SIB, then he would need sib-daemon module and one or more modules of transport SIB. Currently, following transport modules are available sib-tcp, sib-nota and sib-plain-nota.

*B. The process of creating components*

As described above, the following client-side components Smart-M3 platform: libwhiteboard, whiteboard, whiteboard-sib-access, whiteboard-sib-nota, whiteboard-sib-plain-nota and Smart-M3 Qt Core.

These components exist in the form of source code and DEB packages for i386 and ARM. The software for the operating system MeeGo distributed as an RPM package that can be collected using the source code. There are also programs that will convert the existing packages in RPM DEM and back, for example, the program alien. But we will consider the reassembly of the program source code.

To begin to understand what exactly is an RPM package and what it is that will allow us to start the process of building the package.

1. *RPM package:* RPM (RPM Package Manager - RPM - a package manager, previously disclosed as a Red Hat Package Manager - Package Manager Red Hat) indicates two things: the format of the software packages and software designed to manage those packages.

Normally, each packet is an application and a number of associated files. One of the advantages of the RPM - it's what every rpm-file (which is a file from the perspective of the operating system) contains the entire set of application files.

2. *Preparing the environment for building RPM:* In any system there is a standard infrastructure for building RPM. By default it is located in /usr/src/redhat/. If there is no such directory, install the rpm-build. Not on Red Hat systems to install this package will automatically create the directory ~/rpmbuild in the user directory. Its presence enables us to collect packets, with administrative privileges, that is not always safe.

To build you need to copy the source tarball in the directory ~/rpmbuild/SOURCES.

3. *Create a SPEC-file:* Spec-file, which stands for "specification file", defines all of the rpmbuild tools, which should be taken when building an application, as well as all the steps required to install / uninstall. Each package src.rpm-incorporates the spec-file for rebuilding the package.

Spec-file - a text file. Naming convention offers the spec-file name as follows: package\_name.spec.

The text inside the spec-file has a special syntax. Syntax definitions are as determining the order of assembly, version number, information about dependencies, and in general all the information about the package, which can subsequently be requested from the RPM database.

For more information on the preparation of spec-file and the section you can refer to the relevant documentation. [5]

3.1. *Introduction section:* General information section contains information about the package, which after installation can be requested by the command rpm -qi package\_name.

3.2. *PREP section:* Preparing section is responsible for the commands needed to start building. For example, if you put SOURCES tarball of the project, it is necessary to unpack.

Section starts with the %prep. This example uses the macro% setup, which can unpack compressed archives. Typically, this is the only line in this section.

3.3. *BUILD section:* Section contains the commands to build a software. Usually in this section involves two parameters of the script *configure* (compiler optimization flags, and name of the temporary build directory) and the command *make* (without a parameter, that is, for the purpose of all). Section begins with the line %build.

3.4. *INSTALL section:* This section contains commands to install the package files into the system. At this stage, clear the build directory and copy the package files into the directory specified by the option - prefix. If you do not clean the build directory and files from previous builds may violate the purity of the installation. Section begins with the line% install.

3.5. *CLEAN section:* The commands in this section purged files created on other stages. Session begins with the line %clean

3.6. *FILES section:* Finally, the commands in the %files section lists files and directories with the appropriate attributes to be copied from the build tree to the rpm-package and then be copied to the target system when you install this package.

Section begins with the line %files. %Doc macro notes documentation files. This allows you to create documentation from the appropriate project files. After you

finish editing the spec-file is left to put it in the directory `~/rpmbuild/SPECS`, and the source tarball in `~/rpmbuild/SOURCES`.

3.8. *CHANGELOG section*: This section is intended to describe the chronology of changes that have occurred since the first version of the package assembly. Each new record is written in the following sequence: day, month, day, year, name of the developer, his email address.

### C. Compiling a package

Compiling a package by a single command: `$ rpmbuild-ba [path to the SPEC-file]`

Then you can explore the directory `~/rpmbuild/RPMS` and `~/rpmbuild/SRPMS`, find it pre-built packages and examine their commands `rpm-q [package.rpm]`, install and remove, respectively.

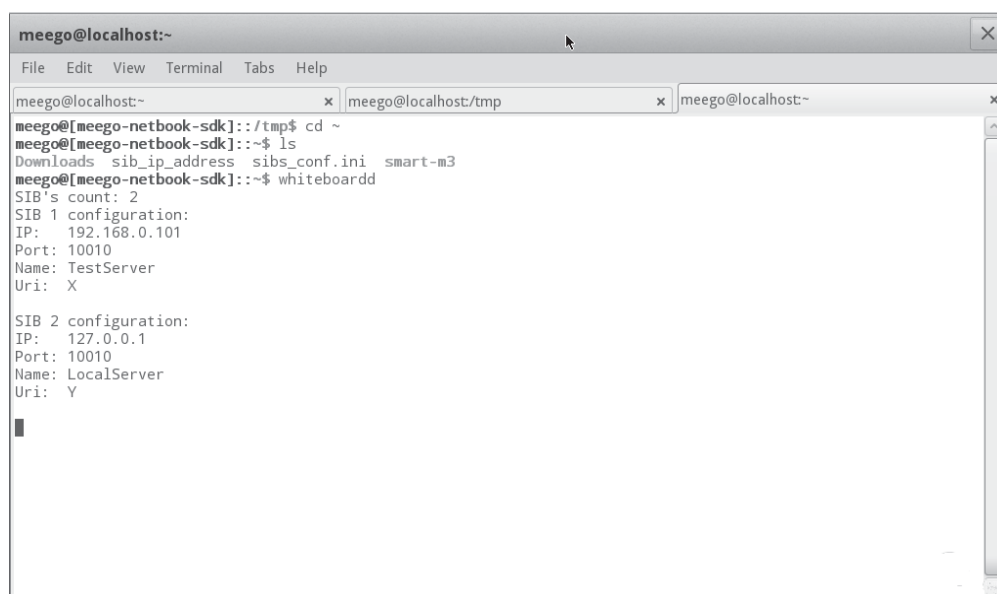
### D. Install created packages to MeeGo

After assembling the packages, make sure they are correctly installed and running on the system.

We describe a step by step how to install and run the components on the platform Smart-M3:

1. Download the RPM packages for each component. [9]
2. Install them using the `rpm` command as follows:
  - `rpm -Uhv libwhiteboard-2.0-beta1.i386.rpm`
  - `rpm -Uhv libwhiteboard-dev-2.0-beta1.i386.rpm`
  - `rpm -Uhv whiteboardd-2.0-beta1.i386.rpm`
  - `rpm-Uhv whiteboard-sib-access-2.0-beta1.i386.rpm`
  - `rpm-Uhv libwhiteboard_qt4-0.9.2-1.i386.rpm`
3. Whiteboardd run in the console.
4. After starting `whiteboardd` create all connections and will hang in the background.

An example of whiteboard daemon on MeeGo, shown in Fig.2.



```

meego@localhost:~
File Edit View Terminal Tabs Help
meego@localhost:~ x meego@localhost/tmp x meego@localhost:~ x
meego@[meego-netbook-sdk]::~/tmp$ cd ~
meego@[meego-netbook-sdk]::~-$ ls
Downloads sib_ip_address sibs_conf.ini smart-m3
meego@[meego-netbook-sdk]::~-$ whiteboardd
SIB's count: 2
SIB 1 configuration:
IP: 192.168.0.101
Port: 10010
Name: TestServer
Uri: X
SIB 2 configuration:
IP: 127.0.0.1
Port: 10010
Name: LocalServer
Uri: Y

```

Fig. 2. Whiteboard daemon launch on MeeGo

#### IV. CONCLUSION

The result of this work were:

1. investigated APIs and Smart-M3 platform architecture;
2. the mechanisms for MeeGo package management;

Create an installation package of client platform components Smart-M3 for the operating system MeeGo. Installation packages are available for free download, written documentation of the installation of the operating system MeeGo. [7, 8]

It was also developed demonstration application using the Smart-M3 platform and it has been tested by the components for the MeeGo operating system. This application was demonstrated at the 9th International Conference FRUCT, which was held in Petrozavodsk, from 25 to 29 April and at the exhibition in honor of the 125th anniversary of the ETU.

Development and deployment platform of smart spaces, Smart-M3 is ready to use on the MeeGo operating system.

#### ACKNOWLEDGMENT

Author would like to thank Finnish-Russian University Cooperation in Telecommunications (FRUCT) program for the provided support and R&D infrastructure. I would also like to thank Kirill Krinkin for providing feedback and guidance.

#### REFERENCES

- [1] Smart-M3 release. <http://sourceforge.net/projects/smart-m3/>. Referenced May 20th 2011.
- [2] Smart-M3, Wikipedia: <http://en.wikipedia.org/wiki/Smart-M3>. Referenced June 10th 2011.
- [3] Compiling RPM packages - <http://www.lexpr.ru/node/39>. Referenced May 20th 2011.
- [4] MADDE - <http://wiki.maemo.org/MADDE>. Referenced June 11th 2011.
- [5] Project bugtracker: <http://osll.spb.ru/projects/msc-yudenok>. Referenced September 28th 2011.
- [6] Project wiki: <http://osll.spb.ru/projects/msc-yudenok/wiki>. Referenced September 28th 2011.
- [7] Project files: <http://osll.spb.ru/projects/msc-yudenok/files>. Referenced September 28th 2011.