**ROAMS Developer Quick Start Guide**

**Preface:**

This document is intended to give prospective developers of the ROAMS project a basic understanding of the systems implemented in ROAMS and the best practices for development. It will cover three platforms, the Intel Edison (used in the round), the Raspberry Pi (used in the server), and the android tablet (used as a tablet GUI). Along with these three platforms, this guide will cover the basic process of building and implementing a change to the software systems that run on each platform.

**Passwords:**

As a general rule, if a password is encountered while using the ROAMS System, the password will be “USMAROAMS”, exactly as found between the quotation marks.

**Tablet:**

The tablet used with this project was a Galaxy Note II Android tablet, running the standard Android distribution. The Tablet itself requires no setup in addition to the out-of-box setup. However, in order to easily install the android app, the developer will require an Android IDE, and a USB cable for interfacing with the tablet from their computer. For development of the Android app, the 2015 ROAMS team used the integrated android-Eclipse IDE. Installing the app on the tablet is as simple as using the built-in features to compile the .apk file and send it to the tablet connected by USB.

**Intel Edison:**

The intel Edison is a micro-computer developed by Intel which runs a miniaturized version of Linux, called Yocto. The Edison requires some initial setup to work as intended. Before changes can be made to the Edison, the developer must establish a connection to the local shell.

Establishing a connection to the Shell: There are two ways to establish a connection to the intel Edison, both requiring a separate computer. Initially, the user must connect VIA serial protocol to communicate with the Edison. The 2015 ROAMS team determined that using PUTTY was the easiest way of accomplishing this. By plugging a USB cable into the edison’s console board, and connecting it to another computer, the developer can connect via Putty’s serial option. The two parameters required are a Com channel, and the baud rate. The developer must determine the COM channel being used (On windows, this would be done through Device Manager), and use the baud rate 115200 for the intel Edison. Once a serial connection is established, the developer will be able to log in and interact with the command line like usual. ***\*\*NOTE: The Intel Edison puts the shell into a sleep state after a few seconds of not receiving any input. Often, keystrokes will not be registered until the second press. Beware of this effect while logging in or typing commands quickly\*\****

By default, the Intel edisons will come configured to connect to the ROAMSNET\_PI network on startup. Assuming the raspberry pi (RPi) is running, the network is up, and the Edison booted properly, the developer will be able to connect via the SSH protocol included with PuTTy. To accomplish this, ensure that the Edison has had time to boot fully, and then simply type the ip address allocated by the RPi’s DHCP server into putty’s SSH option, with port 22, and press connect. Putty should open an SSH shell into the Edison. Note that this will require you to first get the Edison’s IP address by logging in over serial. Use of the command “ifconfig” will print the Edison’s network interface information. The developer should use the wlan0 interface for connecting wirelessly.

Making changes to the Round Script: The mortar rounds run a python script on startup through use of bash startup scripts and systemd. This next section will document where to find these files on the Edison and how to make changes to them.

**File Locations in the Edison:**

1. lightOff.py -> /home/root/

2. startUp.sh -> /etc/init.d/

3. newFuzeClient.py -> /home/root/ (startUp.sh will need to reference this file's location)

**File Details:**

1. lightOff.py: A program for testing purpose. Once we execute newFuzeClient.py, the light will not turn off even after we shut down the client program and we do not want to unplug and plug it in the computer every time we start testing. Therefore, execute this program to turn off all lights except the power (whatever light that's connected to GPIO pin 15).
2. newFuzeClient.py: The Client program. This is the main client code for the Intel Edison to establish a connection with the server, receive/send data and change the fuze (change the LED lights on the physical round).
3. startUp.sh: The start-up shell script. This script is used to "open" all GPIO pins, turn on the power light (GPIO pin 15), connect to the wifi automatically, and run the newFuzeClient.py.

**How to connect to wifi and the configuration for auto wifi:**

1. Execute "configure\_edison -w" and connect to the desired network (provided by magServer)

2. Once connected, it will write a record for this connection on the "/etc/wpa\_supplicant/wpa\_supplicant.conf" file. Make sure there is only 1 record left on the configuration file to make the auto wifi work (not sure if we can only have 1 record there, though).

**Steps to make startUp.sh a default service (i.e. run on boot up):**

1. Create a directory called "init.d" in /etc/ and put this script file there

2. Execute "chmod +x startUp.sh" to ensure Edison can use this file (permission issue)

3. Execute "update-rc.d startUp.sh defaults" to make it run on start-up

4. You can then choose to reboot or re-plug in the Edison to see the effects

**Raspberry Pi Server:**

The project should come with a pre-configured system image for the Raspberry Pi Server. Installing the image and booting the RPi should start a DHCP server and run the ROAMS Data Server. Note that booting the RPi should be the first step in setting up the server. There is a runnable .jar file located in /home/runner which is the compiled java code included with the project. This code is run through a runner’s bashrc on login.

**Implementing changes to the Data Server:**

1. Make changes to java code in IDE
2. Compile runnable jar file including all libraries in the package.
3. Transfer runnable jar file (name must be magServer.jar) to the RPi from development environment. We found that again, using PSCP (Putty Secure Transfer Protocol) was the most efficient way of making this transfer.
4. Reboot RPi or restart java server from command line to cause the changes to take effect.

**GIT:**

Version control for the ROAMS project is managed through git. There is a github organization named “USMA-ROAMS” that contains a “ROAMS” repository with all of the Server and Round code.