**­The RTU Program Environment Setup and Installation Procedure**

**Lu Zhen**

**20/05/2015**

The RTU program is developed in Java language and so it is an OS-Cross design. At present, it has been running on Microsoft Windows (host on Industrial PC and Linux (host on Raspberry Pi B+).

To facilitate the management and maintenance, all of the RTUs in the SinBerBEST BIMG Test-Bedding system use one version of RTU program with different setting files (i.e. each RTU uses its individual SettingSaving.xml). In this way, the parameters setting of RTUs is converted to be a text file (xml file) editing work, which is readable/writeable by any text file editor.

### File Map for the Operating Environment of RTU Program

The RTU program is programmed under development tools JDK 1.8.0 (or above). Therefore, the host machine must install JDK 1.8.0 (or above) and set up the JAVA\_HOME properly.

1. The default root fodder of the RTU program is c:/LatestSetup/RPi518 in Microsoft Windows (or \home\root\NetBeansProjects\RPi518\dist in Linux of Raspberry Pi). In the root fodder, there is a batch of files/sub-fodder as follows:

|  |  |
| --- | --- |
| File/Sub-fodder Name | Description |
| MapTagPiSforAC\_MeterE.xml | The Map file of Pi-System tags for AC Meter energy kind of data saving, which can be auto-generated by RTU in first power on or by BIMG server remote-setting command. |
| MapTagPiSforAC\_MeterN.xml | The Map file of Pi-System tags for AC Meter normal kind of data saving, which can be auto-generated by RTU in first power on or by BIMG server remote-setting command. |
| MapTagPiSforDC\_MeterE.xml | The Map file of Pi-System tags for DC Meter energy kind of data saving, which can be auto-generated by RTU in first power on or by BIMG server remote-setting command. |
| MapTagPiSforDC\_MeterN.xml | The Map file of Pi-System tags for DC Meter normal kind of data saving, which can be auto-generated by RTU in first power on or by BIMG server remote-setting command. |
| MapTagPiSforRelay.xml | The Map file of Pi-System tags for DC Meter (or AC Branch Control) Relay control kind of data saving, which can be auto-generated by RTU in first power on or by BIMG server remote-setting command. |
| MapTagPiSforSwitch.xml | The Map file of Pi-System tags for DC Meter (or AC Branch Control) Switch status kind of data saving, which can be auto-generated by RTU in first power on or by BIMG server remote-setting command. |
| README.TXT | Brief setting procedure describing |
| SettingSaving.xml | The operation parameters setting file of the RTU (first thread in Microsoft Windows visual RTU of AC Power Source Emulator) |
| SettingSaving1.xml | The operation parameters setting file of the RTU (second thread in Microsoft Windows for visual RTU of PV Emulator) |
| SettingSaving2.xml | The operation parameters setting file of the RTU (third thread in Microsoft Windows for visual RTU of AC Electronic Load) |
| SettingSaving3.xml | The operation parameters setting file of the RTU (fourth thread in Microsoft Windows for visual RTU of DC Electronic Load) |
| SettingSaving4.xml | The operation parameters setting file of the RTU (fifth thread in Microsoft Windows for visual RTU of Battery Emulator) |
| RPi518.jar | The executable jar file of the RTU program |
| lib | Sub-fodder for java library (only needs in Raspberry Pi) |

1. The sub-fodder of ‘lib’ in the root fodder of the RTU program is needed only in Linux OS environment (Raspberry Pi), which includes jar library files as follows:

|  |  |
| --- | --- |
| File Name | Description |
| grizzly-framework-2.3.15-gfa.jar | The jar library file for Websocket communication in client side |
| grizzly-http-2.3.15-gfa.jar | The jar library file for Websocket communication in client side |
| grizzly-http-server-2.3.15-gfa.jar | The jar library file for Websocket communication in client side |
| java-json.jar | The jar library file for JSON |
| javax.websocket-api-1.1.jar | The jar library file for Websocket communication |
| junit.jar | The jar library file for pi4j |
| pi4j-core.jar | The jar library file for pi4j |
| pi4j-core-javadoc.jar | The jar library file for pi4j |
| pi4j-core-sources.jar | The jar library file for pi4j |
| pi4j-device.jar | The jar library file for pi4j |
| pi4j-device-javadoc.jar | The jar library file for pi4j |
| pi4j-device-sources.jar | The jar library file for pi4j |
| pi4j-example.jar | The jar library file for pi4j |
| pi4j-example-javadoc.jar | The jar library file for pi4j |
| pi4j-example-sources.jar | The jar library file for pi4j |
| pi4j-gpio-extension.jar | The jar library file for pi4j |
| pi4j-gpio-extension-javadoc.jar | The jar library file for pi4j |
| pi4j-gpio-extension-sources.jar | The jar library file for pi4j |
| pi4j-service.jar | The jar library file for pi4j |
| pi4j-service-javadoc.jar | The jar library file for pi4j |
| pi4j-service-sources.jar | The jar library file for pi4j |
| rxtx-2.1.7.jar | The jar library file for serial port communication |
| rxtx-2.2pre1.jar | The jar library file for serial port communication |
| rxtx-2.2pre2.jar | The jar library file for serial port communication |
| tyrus-client-1.9.jar | The jar library file for Websocket communication in client side |
| tyrus-container-grizzly-client-1.9.jar | The jar library file for Websocket communication in client side |
| tyrus-container-grizzly-server-1.9.jar | The jar library file for Websocket communication |
| tyrus-core-1.9.jar | The jar library file for Websocket communication |
| tyrus-server-1.9.jar | The jar library file for Websocket communication |
| tyrus-spi-1.9.jar | The jar library file for Websocket communication |
| xmlpull-1.1.3.1.jar | The jar library file for xml file converting |
| xpp3\_min-1.1.4c.jar | The jar library file for xml file converting |
| xstream-1.4.7.jar | The jar library file for xml file converting |

1. Since the RTUs hosting on Raspberry Pi are mostly working on non-user-interface mode, it is very important for the RTUs’ design to have the following features: i) power-on and play; ii) auto-recovering from any abnormal situations; iii) intelligently auto-selecting WiFi-Router path to BIMG server(s) as well as auto-switching to the BIMG server(s) in activity. To achieve these features requires the RTU program in using the Linux system files as follows:

|  |  |  |
| --- | --- | --- |
| File Name | Location | Description |
| hostname | /etc | The context of the file is the hostname of RTU which displayed on network. e.g. ‘RTU-DC\_BRANCH\_000\_001’ |
| profile | /etc | The file is for the RTU program auto-starter when the Raspberry Pi is power-on (or reboot). Normally, it just needs to add the following linux commands to the end of the file:  *cd /home/root/NetBeansProjects/RPi518/dist*  *sudo /opt/java/jdk1.8.0\_33/bin/java -jar RPi518.jar*  If the RTU needs to work in setup environment mode, the above linux commands should be masked. i.e.  *#cd /home/root/NetBeansProjects/RPi518/dist*  *#sudo /opt/java/jdk1.8.0\_33/bin/java -jar RPi518.jar* |
| wpa\_supplicant.conf | /etc/wpa\_supplicant | The context of the file is the parameter settings for accessing WiFi-Routers. |
| interfaces | /etc/network | The context of the file is the parameter for the RTU’s IP address settings. |

**Note1: the context of the files in red-colour is varied according to the individual RTU’s real environment;**

**Note2: the context of the files in red-colour can be remotely edited by BIMG servers. The RTU program has offered the remote-reading and remote-writing functions for the BIMG server side;**

### Details of Environment Setting Files

There are several files which are used for the environment setting of RTU program. These files are editable by any kind of text file editor. With modifying the context of the files, the RTU program and Linux OS determine their operating modes.

1. **SettingSaving.xml, SettingSaving1.xml, SettingSaving2.xml, SettingSaving3.xml & SettingSaving4.xml settings for example:**

*<config>*

*<properties>*

*<!-- TYPE\_RTU is defined as follow:*

*0—DC meter without Relay Control; 1—DC meter with Relay Control;*

*2—Relay Control only; 3—AC meter with Relay Control;*

*4—AC meter without Relay Control; 5—DC\_AC Converter in Serial port;*

*6—DC Programmable Load in SCPI; 7—AC Programmable Load in SCPI;*

*8—AC Programmable Source in SCPI; 9—PV Simulator in SCPI;*

*10-BE10 battery emulator -->*

*<property name="TYPE\_RTU" value= "0"/>*

*<!-- The Refresh Time Setting is millisecond -->*

*<property name="NormalData Refresh Time Setting" value= "2000"/>*

*<property name="EnergyData Refresh Time Setting" value= "60000"/>*

*<!-- When the BIMG system is working in dual server mode, setting to true*

*When the BIMG system is working in single server mode, setting to false-->*

*<property name="Double Server System" value= "true"/>*

*<!-- When the BIMG server is working in auto-switch-status monitoring mode, setting to*

*True; otherwise setting to false-->*

*<property name="Auto-Sending Switch State" value= "true"/>*

*<!--If the* Device Error RebootEnable *is setting to true, when RTU detects the device’s*

*communication having lost more than a threshold time, reboot the Raspberry Pi -->*

*<property name=*"Device Error Reboot Enable" *value= "true"/>*

*<!-- When the BIMG server is working in auto-switch-status monitoring mode, setting to*

*True; otherwise setting to false-->*

*<property name="Auto-Sending Switch State" value= "true"/>*

*<!-- The RTU can link with two servers at the same time (or link with one server in*

*single-server mode). If an operating server goes to off-line in any situation, the*

*RTU will auto-select the next server in the following list to link with-->*

*<property name="Server-Address1" value= "172.25.186.73"/>*

*<property name="Server-Address2" value= "172.25.186.74"/>*

*<property name="Server-Address3" value= "10.25.187.9"/>*

*<!-- When the RTU is working in multi-WiFi-Routers mode, setting to true -->*

*<property name="Enable\_WiFiAutoSelect" value= "true"/>*

*<!-- If an operating WiFi-Router goes to off-line in any situation, the linked RTU in*

*multi-WiFi-Routers mode will auto-select the next WiFi-Router in the following*

*list to link with-->*

*<property name="WiFiRouter1st" value= "SBB\_BIMG\_Router-4"/>*

*<property name="WiFiRouter2nd" value= "SBB\_BIMG\_Router-1"/>*

*<property name="WiFiRouter3rd" value= "SBB\_BIMG\_Router-3"/>*

*<!— When the RTU is working in debug monitoring mode, setting the relevant bit to ‘1’*

*will display the status on monitor.*

*1-main; 2-WS; 4-PiS; 8-serial; 16-device inside; 32-WiFi; 64-device outside -->*

*<property name="Enable\_Display\_WS\_Status" value= "11"/>*

*<!— When the RTU need to work on debug monitoring mode, setting the relevant bit to ‘1’*

*1-main; 2-WS; 4-PiS; 8-serial; 16-device inside; 32-WiFi; 64-device outside -->*

*<property name="Pi-System User ID and Password" value= "piadmin:p@ssw0rd"/>*

*<property name="Pi-System\_URL" value= "https://10.25.187.52/piwebapi/"/>*

*<property name="Pi-System Element Name" value= "SG.CR11.BIMG.DC\_BRANCH\_001\_001"/>*

*<!— The RTU directly sends saving-data to Pi-System by means of Web-API, which is a*

*latest Pi-System accessing technology of OSIsoft. To use Web-API, the RTU program*

*must have the Root\_Element WebID, Root\_Element Name as well as the Pi-System URL -->*

*<property name="Pi-System\_URL" value= "https://172.25.186.70/piwebapi/"/>*

*<property name="Pi-System AC Branches Root\_Element WebID" value=*

*"E03RtlcqyEokGCdohIi2-*

*\_uQ8CBvQCjR5BGCY6A2n09rSwU0JCUElBRlNWUjFcQUZEQlxCSU1HXEFDX0JSQU5DSA" />*

*<property name="Pi-System DC Branches Root\_Element WebID" value=*

*"E03RtlcqyEokGCdohIi2-*

*\_uQHQpaxCjR5BGCY6A2n09rSwU0JCUElBRlNWUjFcQUZEQlxCSU1HXERDX0JSQU5DSA"/>*

*<property name="Pi-System Switches Root\_Element WebID" value= "E03RtlcqyEokGCdohIi2-*

*\_uQ6P3LgSjR5BGCY6A2n09rSwU0JCUElBRlNWUjFcQUZEQlxCSU1HXEFDX1NXSVRDSA"/>*

*<property name="Pi-System AC Source Root\_Element WebID" value="E03RtlcqyEokGCdohIi2-*

*\_uQxL\_GqCrR5BGCY6A2n09rSwU0JCUElBRlNWUjFcQUZEQlxCSU1HXEFDX1NPVVJDRQ"/>*

*<property name="Pi-System AC Load Root\_Element WebID" value= "E03RtlcqyEokGCdohIi2-*

*\_uQyho9CSrR5BGCY6A2n09rSwU0JCUElBRlNWUjFcQUZEQlxCSU1HXEFDX0xPQUQ"/>*

*<property name="Pi-System PV AC DC PS Root\_Element WebID" value=*

*"E03RtlcqyEokGCdohIi2-*

*\_uQaXDZByvR5BGCY6A2n09rSwU0JCUElBRlNWUjFcQUZEQlxCSU1HXERDX0FDX1BTXzAwOV8wMDE"/>*

*<property name="Pi-System BE10 Root\_Element WebID" value="E03RtlcqyEokGCdohIi2-*

*\_uQHIx9c3Lj5BGCdmRRBirKHAU0JCUElBRlNWUjFcQUZEQlxCSU1HXEJBVFRFUlk"/>-->*

*<!— If the RTU links with device by Ethernet port* *RJ45, the Ethernet IP address of the*

*device must be specified to let RTU program to construct the network socket -->*

*<property name="PV\_SCPI\_Socket\_IP" value= "192.168.55.2"/>*

*<property name="BE10\_Socket\_IP" value= "192.168.55.3"/>*

*<!— If the RTU links with device by serial port, the serial port name and its baudrate*

*must be specified to let RTU program know which serial port is used to the serial*

*communication. The default serial port is setting to Raspberry Pi’s "/dev/ttyUSB0" -->*

*<property name="serial.port" value= "COM2"/>*

*<property name="serial. baudrate" value= "9600"/>*

*<!— Each RTU has a name, which may be used to GUI display in BIMG server side. -->*

*<property name="RTU\_Name" value= "DC\_BRANCH\_000\_001"/>*

*<!— Each RTU should have an Unique Identification Number, which is used for recognizing the*

*RTU identification by BIMG server. -->*

*<property name="RTU\_UID" value= "9611466950198501"/>*

*</properties>*

*<tuples/>*

*</config>*

**Note1: each RTU hosted on Raspberry Pi has a SettingSaving.xml file located in \home\root\**

**NetBeansProjects\RPi518\dist, which is editable by any text-editor and will affect the RTU’s operation in next reboot;**

**Note2: the visual RTUs hosted on Industrial PC have their SettingSaving.xml files individually. Please see the last part of this document for how to locate them and how to use them;**

1. **wpa\_supplicant.conf settings for example:**

*ctrl\_interface=DIR=/var/run/wpa\_supplicant*

*GROUP=netdev*

*update\_cogfig=1*

*network={*

*ssid="SBB\_BIMG\_Router-1"*

*psk="12345678"*

*proto=RSN*

*key\_mgmt=WPA-PSK*

*pairwise=TKIP*

*auth\_alg=OPEN*

*priority =1*

*}*

*network={*

*ssid="SBB\_BIMG\_Router-4"*

*psk="12345678"*

*proto=RSN*

*key\_mgmt=WPA-PSK*

*pairwise= TKIP*

*auth\_alg=OPEN*

*priority =1*

*}*

*network={*

*ssid="SBB\_BIMG\_Router-3"*

*psk="12345678"*

*proto=RSN*

*key\_mgmt=WPA-PSK*

*pairwise= TKIP*

*auth\_alg=OPEN*

*priority =1*

*}*

**Note1: ssid can be one of the nearby WiFi-Routers’ name and psk must be the relevant password key setting in the WiFi-Router;**

**Note2: the most favorited linking WiFi-Router put to the forefront and then the second favorited linking WiFi-Router;**

**Note3: the parameter settings must keep in the same with relevant WiFi-Router to be linked;**

**Note4: when new WiFi-Routers are needed to join the BIMG System, they must support TKIP. This is because the current version of WPA on Raspberry Pi is not full support other Wi-Fi Security function;**

1. interfaces settings for example:

Normally, the context of the interfaces file is as follows:

*auto lo*

*iface lo inet loopback*

*auto eth0*

*allow-hotplug eth0*

*iface eth0 inet dhcp*

*allow-hotplug wlan0*

*iface wlan0 inet manual*

*wpa-roam /etc/wpa\_supplicant/wpa\_supplicant.conf*

*iface default inet dhcp*

When the RJ45 port is used for linking with device, the context of the interfaces file is change to:

*auto lo*

*iface lo inet loopback*

*auto eth0*

*#allow-hotplug eth0*

*iface eth0 inet static*

*address 10.25.188.132*

*netmask 255.255.255.240*

*gateway 10.25.188.1*

*auto wlan0*

*allow-hotplug wlan0*

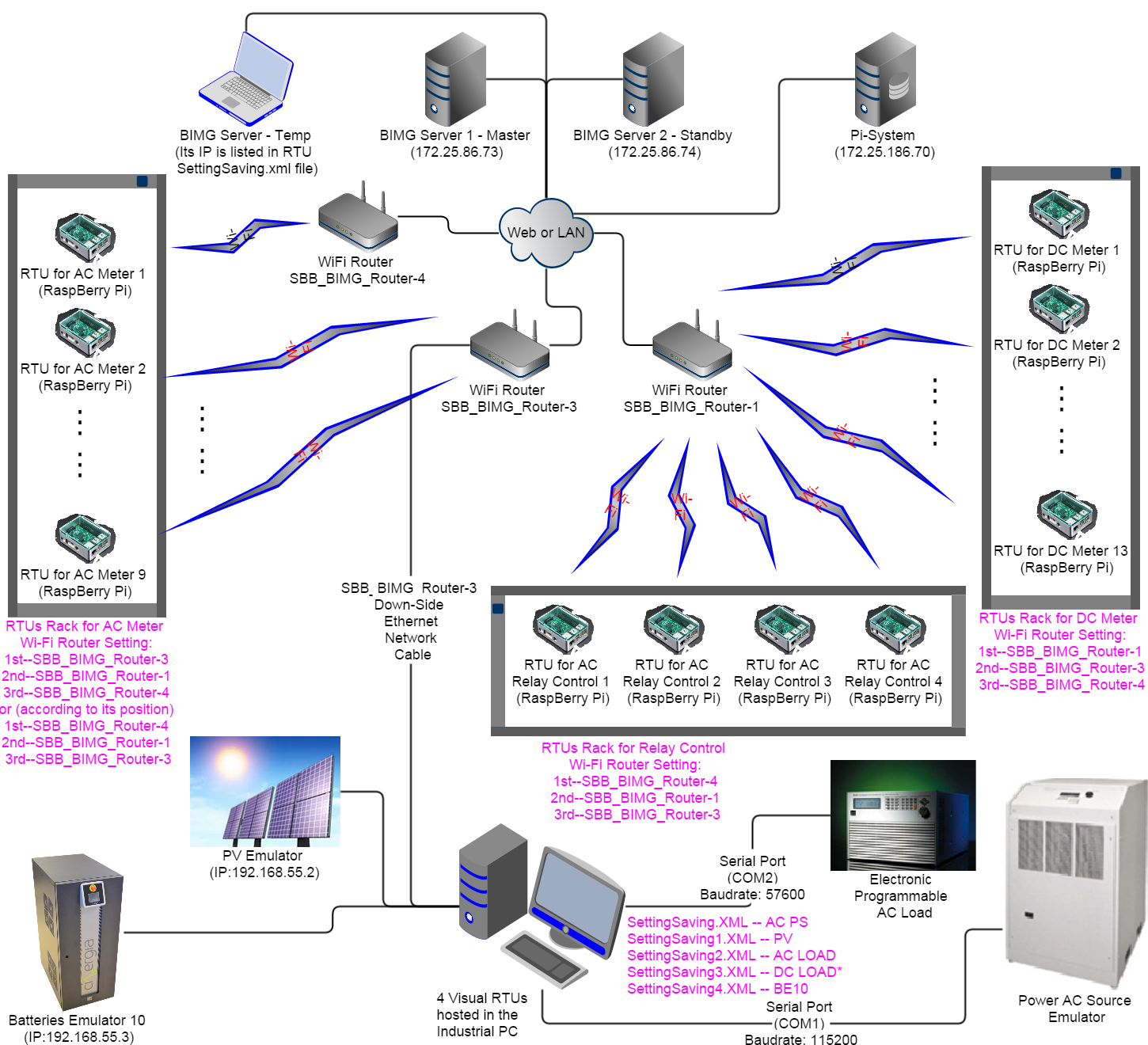
*iface wlan0 inet manual*

*wpa-roam /etc/wpa\_supplicant/wpa\_supplicant.conf*

*iface default inet dhcp*

**Note1: eth0 settings should be modified according to the device IP address settings (the above example is to link with a PAC 3200 AC Power meter with IP:** *10.25.188.131* **);**

### RTU Program Installation Procedure

To be better understanding how to setup the RTU, I have drawn a layout map for the SinBerBEST BIMG Test-Bedding Monitoring & Control System in communication network view. (please see below)

In this system layout map, the location of all of current deployed RTUs are shown. And

1. **The RTU program installation on Linux (hosted on Raspberry Pi)**

In general, the RTU program installation in Linux (hosted on Raspberry Pi) is a quite complicated process. This is because Raspberry Pi is a low-end computer and its OS (Linux) is open-sourced. Therefore, the technical support on Raspberry Pi is very weak and most of the software needs to be searched, installed and upgraded manually.

To avoid doing so much of the repeat work and facilitate the RTU program installation process, there have been produced three Disk-Images files: *2015-5-12(Relay)-wheezy-raspbian.img, 2015-5-12(DC)-wheezy-raspbian.img and 2015-5-12(AC)-wheezy-raspbian.img and put in* in fodder /win32diskimager-v0.9-binary (of the RTU program development PC: Machine: HP Pro 6300, Hostname: SBB-SGH242RHSF). These Disk-Images files are duplicated from a well environment setup Raspberry Pi B+ with the latest version of RTU program installed on.

Therefore, the RTU program installation in Linux (hosted on Raspberry Pi) is simplified as the following steps:

***Step 1:*** Make a duplication-copy on a new MicroSDHC (8GB) by the Win32DiskImager.exe (which has been installed on the RTU program development PC: SBB-SGH242RHSF). If the target RTU is for DC meter (AcuDC243), the source file using *2015-5-12(DC)-wheezy-raspbian.img;* If the target RTU is for AC meter (PAC3200), the source file using *2015-5-12(AC)-wheezy-raspbian.img* andIf the target RTU is for AC Relay Control (with Pi-face digital), the source file using *2015-5-12(Relay)-wheezy-raspbian.img.*

***Step 2:*** Put the duplication-copy (i.e. MicroSDHC) to a Raspberry Pi B+ with GUI (i.e. with Monitor, Keyboard and Mouse). Turn on the power. And then follow the spec on part: Details of Environment Setting Files, to modify the relevant context of the Environment Setting Files. Normally, the file modification is on few of items, such as RTU\_Name, RTU\_UID and WiFiRouter1st & WiFiRouter2nd etc. Other items keep no change.

***Step 2a:*** Delete all of MapTagPiSfor\*.xml files, if there is any such file in the root fodder of RTU program.

After above steps, the RTU program has successfully installed to a MicroSDHC(8GB).

***Step 3:*** Put the MicroSDHC to the RTU (Raspberry Pi B+) which is deployed in a real position of BIMG system. Then turn on the power of the RTU. When the RTU goes into its steady operating state after a while, check the Pi-System for data-saving result and BIMG server for real-time data refreshing.

1. **The RTU program installation on Microsoft Windows**

The RTU program installation depends on OS-System as well as host machine. On Microsoft Windows (hosted on Industrial PC), it just is a simple files-copy process and plus xml-setting files’ modification (SavingSetting.xml ~ SavingSetting4.xml).

***Step 1:*** As mentioned above, the RTU program is programmed under development tools JDK 1.8.0 (or above). Therefore, the pre-installation of RTU program is to install JDK 8 and setup the JAVA\_HOME on Windows System Property. For this process, please reference:

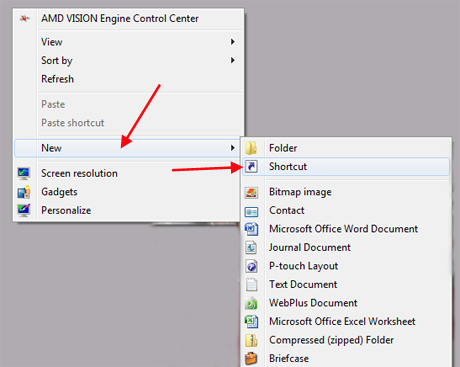
[**http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html**](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) as well as Microsoft Windows Help for system property setting.

***Step 1a:*** If the visual RTU on Microsoft Windows is in using serial port communication, it is required to install the RXTX dll. For this process, please reference the attached file “Installing RXTX.doc” in fodder /BIMG-Document (of the SD-Card for handover).

***Step 2:*** The latest version of RTU program has been produced and put in fodder /LatastSetup/Rpi518 (of the SD-Card for handover). So the easiest way for the RTU program installing is to copy the whole fodder /LatastSetup of the handover SD-Card to C driver of the target machine, i.e. C: /LatastSetup.

***Step 3:*** The RTU program can be started by Command Prompt in Windows 7 & Windows 8. However, it is more convenient to create one shortcut for each visual RTU thread. After then we start the visual RTU thread just by clicking the shortcut. To create the shortcuts, please follow the steps below:

***Step 3a:*** right-click on the *Desktop*. From the menu that appears, choose *"New -> Shortcut"*. The contents of your menu will look slightly different, depending on what you've got installed on your computer.



This will open the *Create Shortcut* wizard. It will ask you the location of the item and name you want to create a shortcut for. Click it to finish.

***Step 3b:*** Right-kick the new created shortcut and go into properties modify dialog to edit the relevant items as follows: (for examples)

AC PS Emulator RTU (SettingSaving.XML):

Name— ACPS RTU

Target— C:\Windows\System32\cmd.exe /k java -jar RPi518.jar

Start in— C:\LatastSetup\RPi518

PV Emulator RTU (SettingSaving1.XML):

Name— PV RTU

Target— C:\Windows\System32\cmd.exe /k java -jar RPi518.jar COPY\_NO=1

Start in— C:\LatastSetup\RPi518

AC LOAD RTU (SettingSaving2.XML):

Name— ACL RTU

Target— C:\Windows\System32\cmd.exe /k java -jar RPi518.jar COPY\_NO=2

Start in— C:\LatastSetup\RPi518

BE10 RTU (SettingSaving4.XML):

Name— BE10 RTU

Target— C:\Windows\System32\cmd.exe /k java -jar RPi518.jar COPY\_NO=4

Start in— C:\LatastSetup\RPi518

**Note: all of the visual RTUs have well been setup in the industrial PC outside the switch-board room. Please reference the settings in detail.**