

SDK Manual for 17Cy

Contents

I. 17Cy SDK Installation

II. QtCreator Configuration

III. 17Cy Qml Project

IV. Debugging

V. QtDesigner

VI. App Performance Analyzing

VII. SimuTool

VIII. Update Record

Attention!!

A. First installation.

If this is your first time to use 17Cy SDK, read the document **from part I to part VII** please.

B. SDK update.

Read the document at **part VIII** of update record to know about the details of sdk update further.

I. 17Cy SDK Installation

To develop one 17Cy application using 17Cy SDK, you need the followings:

1. Computer for running the 17Cy SDK, with the following system requirements:

- (1) Ubuntu® 12.04 or 14.04 64bit version.
- (2) At least 4 GB of RAM memory.
- (3) At least 10 GB of free disk space.

2. Library Dependencies

On Ubuntu®, use the following commands to install library:

```
$ sudo apt-get install libegl1-mesa libgles2-mesa libtiff4-dev gdb-multiarch python2.7  
python-m2crypto python-lxml ia32-libs android-tools-adb
```

3. Download sdk

Copy the sdk from the following path to your native PC:

```
//ibigbear/17Cy/Suntec_Release/Master/SDK/  
//ibigbear/17Cy/Suntec_Release/SDK_Manual/gdb  
//ibigbear/17Cy/Suntec_Release/SDK_Manual/qt-opensource-linux-x64-5.3.1.run
```

4. Install GDB for ARM

(1) Open the terminal to execute the following commands:

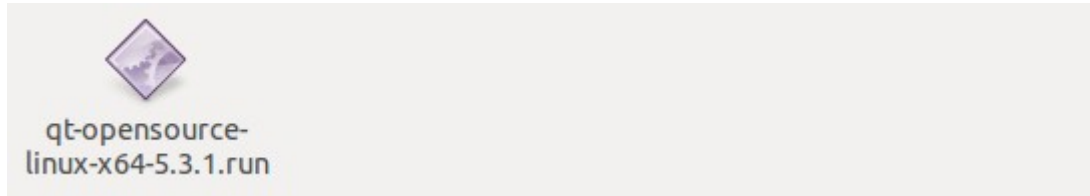
```
$ sudo apt-get install texinfo  
$ sudo apt-get install python-dev
```

(2) Enter into your GDB source code repository and execute the following commands:

```
$ ./configure --target=arm-linux --with-python --prefix=/usr/local/arm-gdb  
$ make  
$ sudo make install
```

5. Install Qt5.3.1 run

(1) Open the folder of your “qt-opensource-linux-x64-5.3.1.run” .

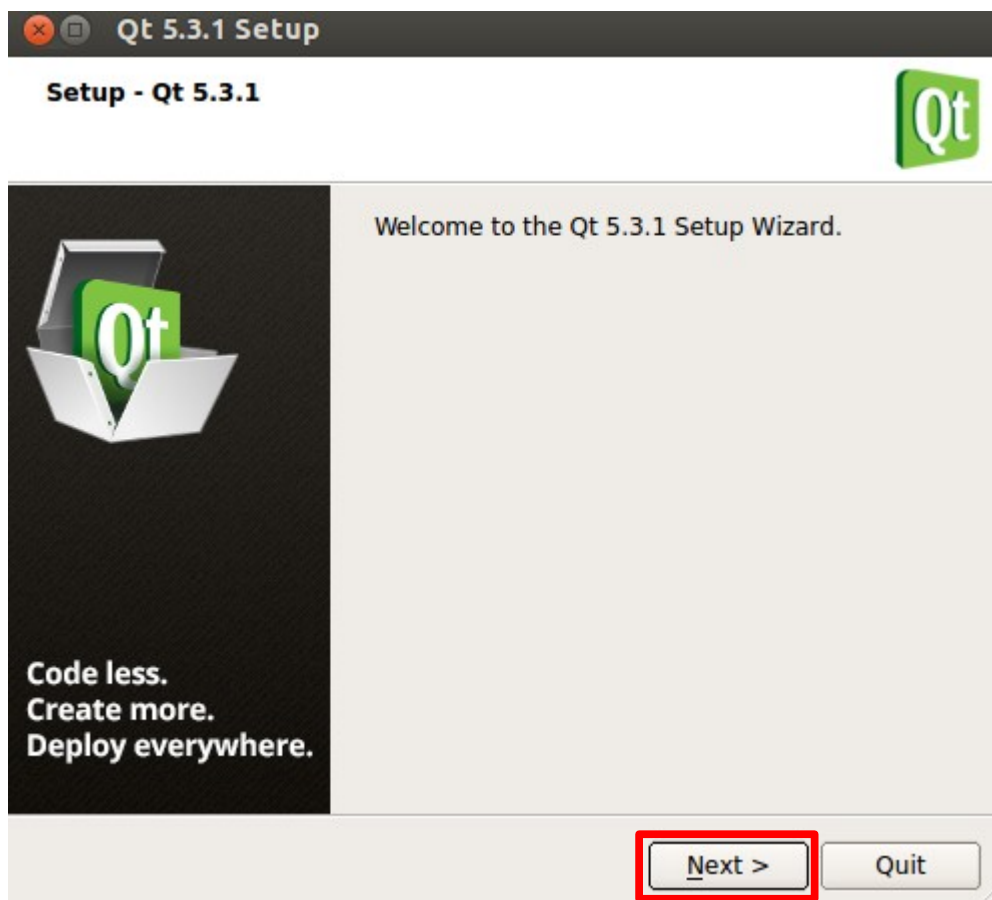


(2) Run the following command:

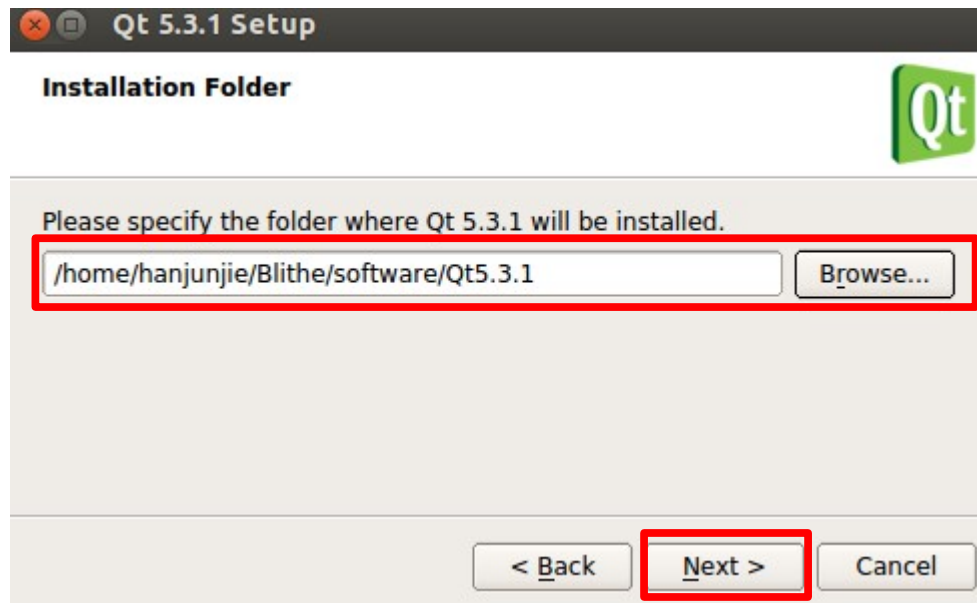
```
$ chmod +x qt-opensource-linux-x64-5.3.1.run
```

Double click the “qt-opensource-linux-x64-5.3.1.run” to start up the install wizard. Then, do according to the *Setup Wizard* as follows:

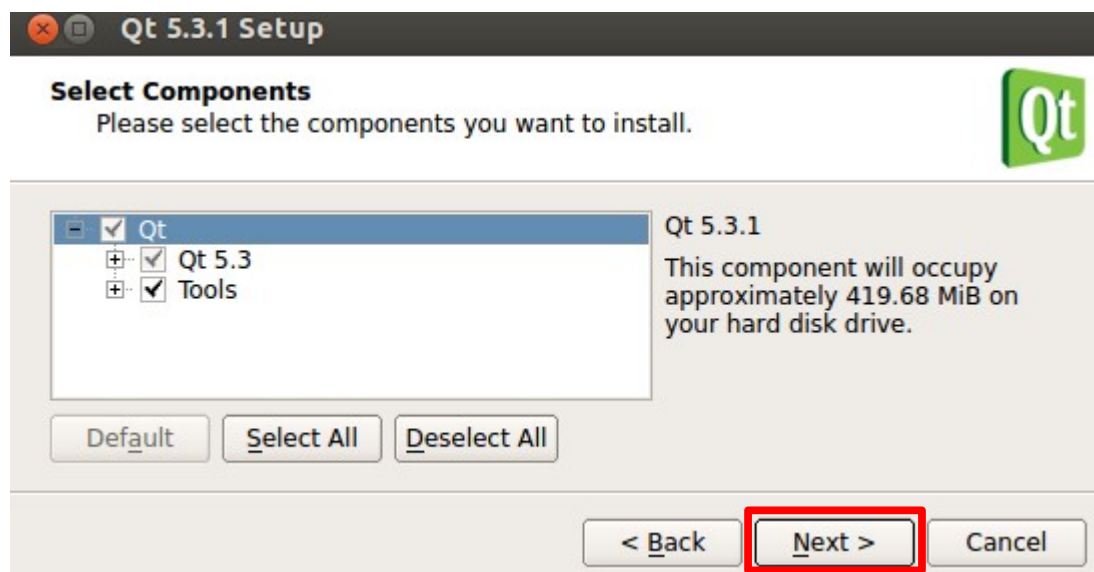
Click “Next” >



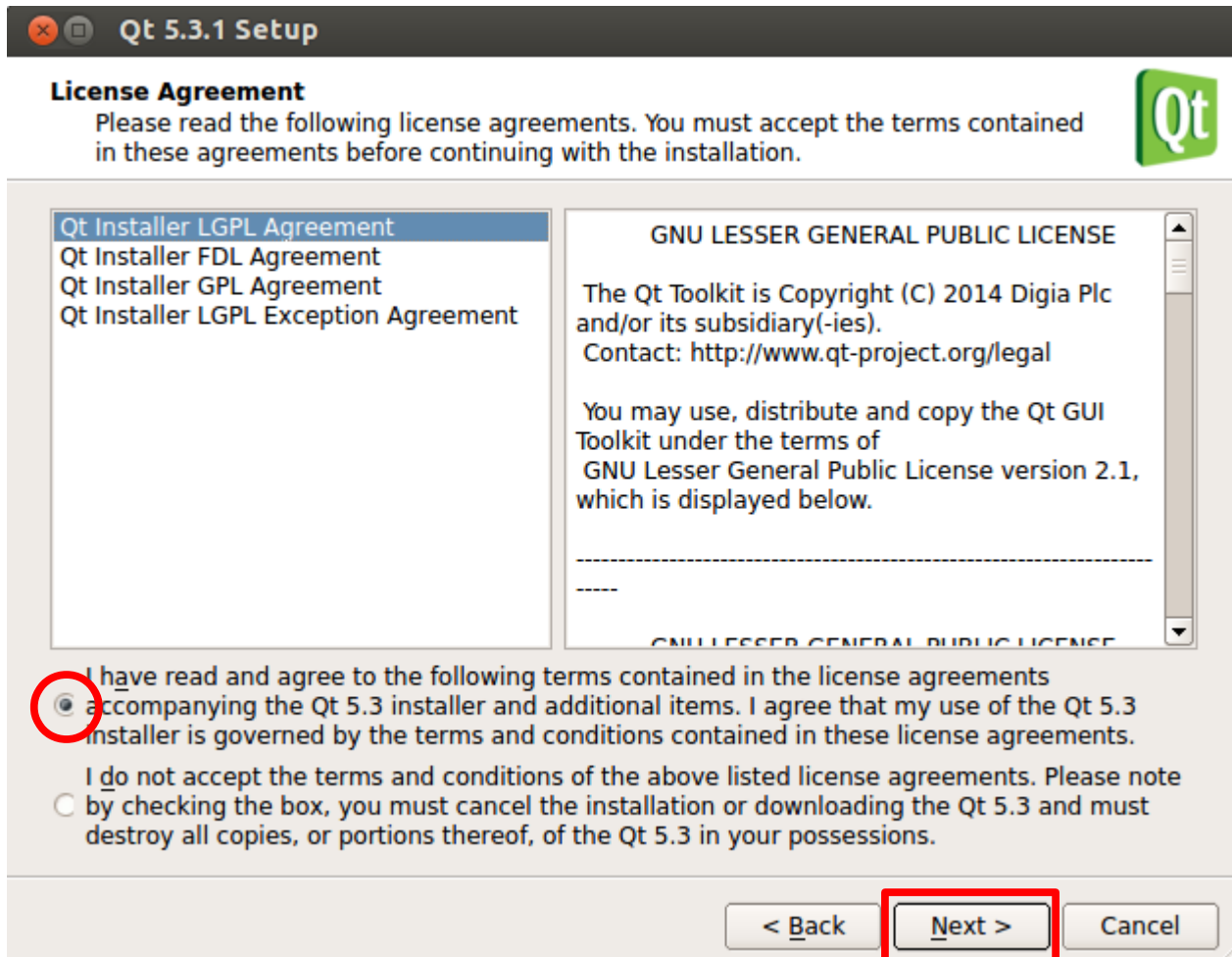
Click **“Browse”** to specify your installation folder and click **“Next”** >



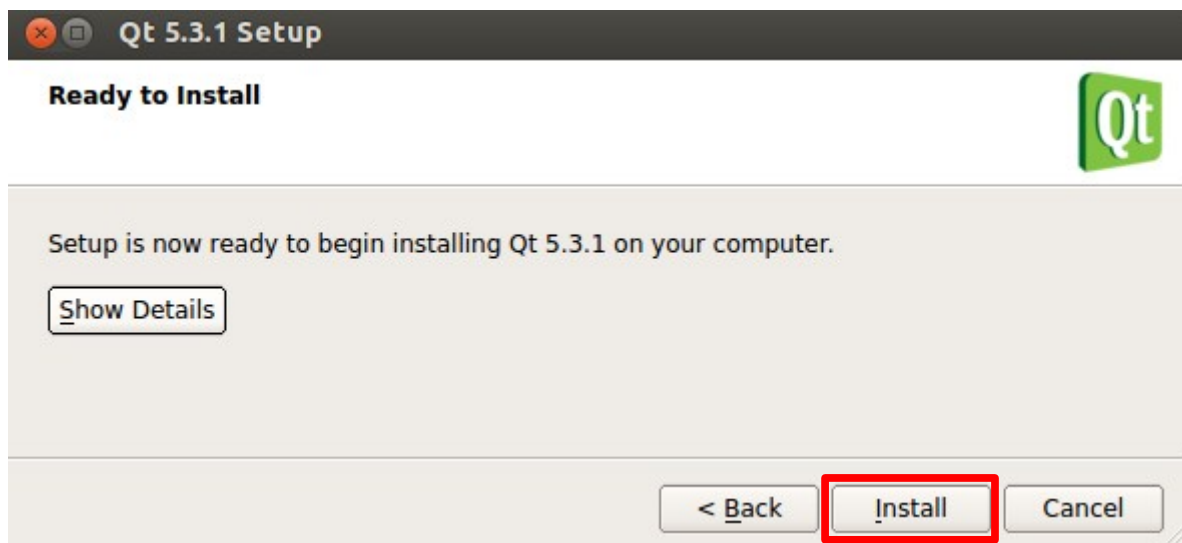
Click **“Next”** to select the default components >



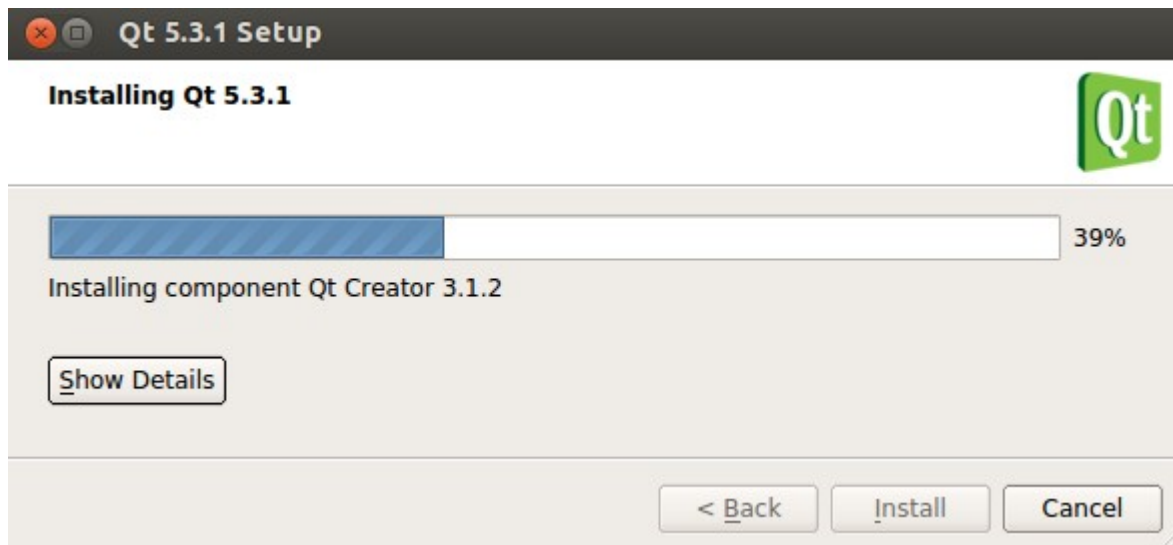
Accept the license agreement and click “Next” >



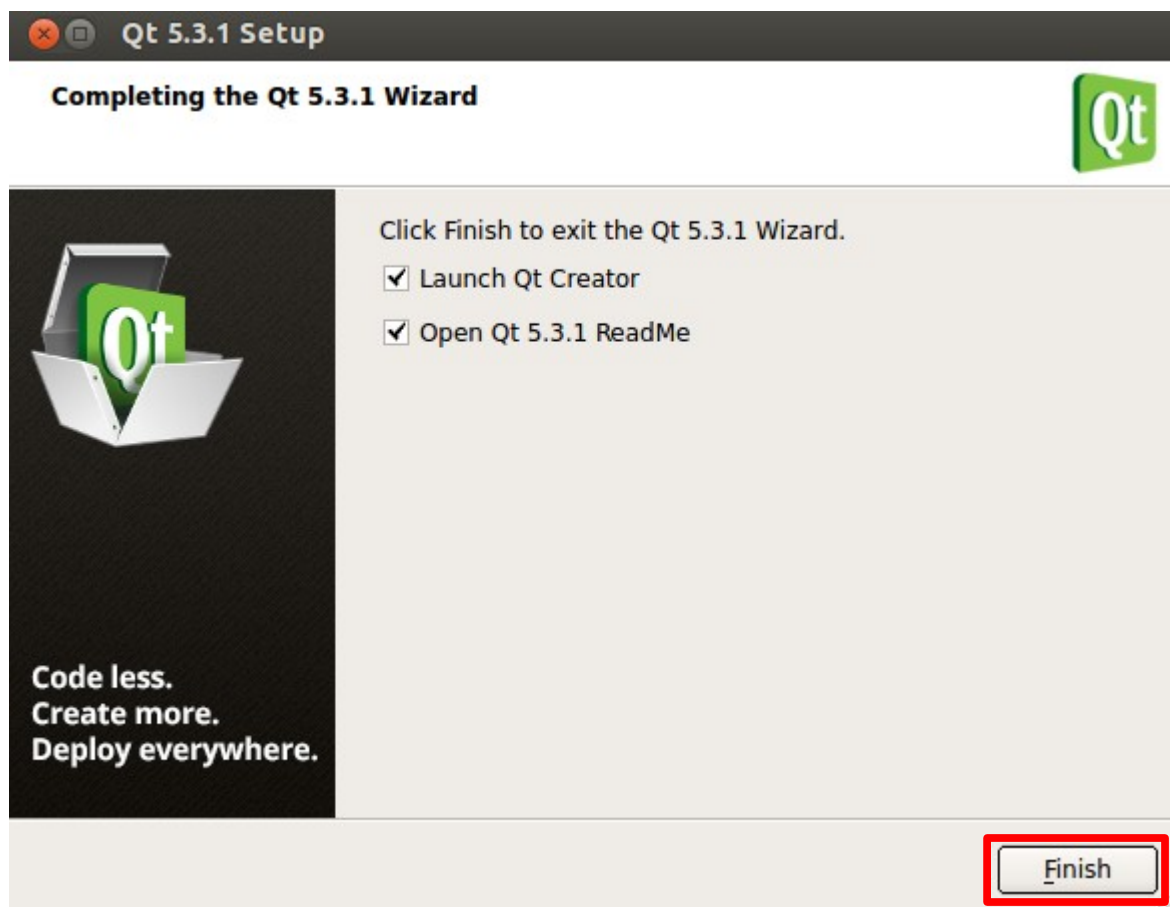
Click “Install” >



Installing for a few seconds >

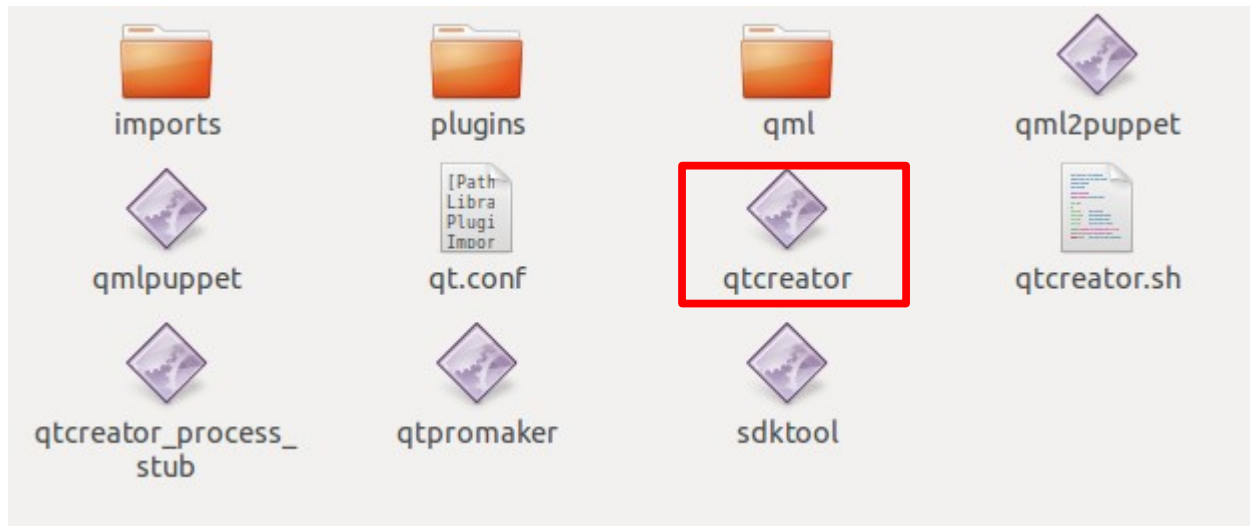


Click "Finish" >



Run "qtcreeator" to launch Qt Creator:

```
/home/hanjunjie/Blithe/software/Qt5.3.1/Tools/QtCreator/bin/qtcreeator
```



6. Install 17Cy SDK

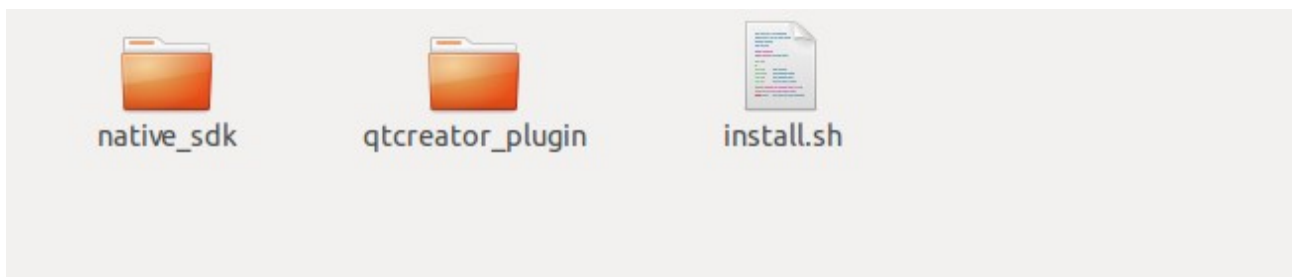
(1) Open your 17Cy SDK package. There are several different kinds of sdk. The sdk marked with red color(dcue2/dcun2n/dcun2w) are used often:

xxx_genericdcue2

xxx_genericdcun2n

xxx_genericdcun2w

“xxx” is the version number of the sdk. You should install the three different kinds of sdk one after another if necessary. For example, choose genericdcue2 and open the folder, you will see:



(2) Then run the following commands to install 17Cy SDK:

```
$ chmod +x install.sh
```

```
$ ./install.sh path
```

path: “Qt5.3.1 run” installation folder, such as
/home/hanjunjie/Blithe/software/Qt5.3.1

II. QtCreator Configuration

1. Configuration for Build & Run

(1) Click “Tools -> Options -> Build & Run” .

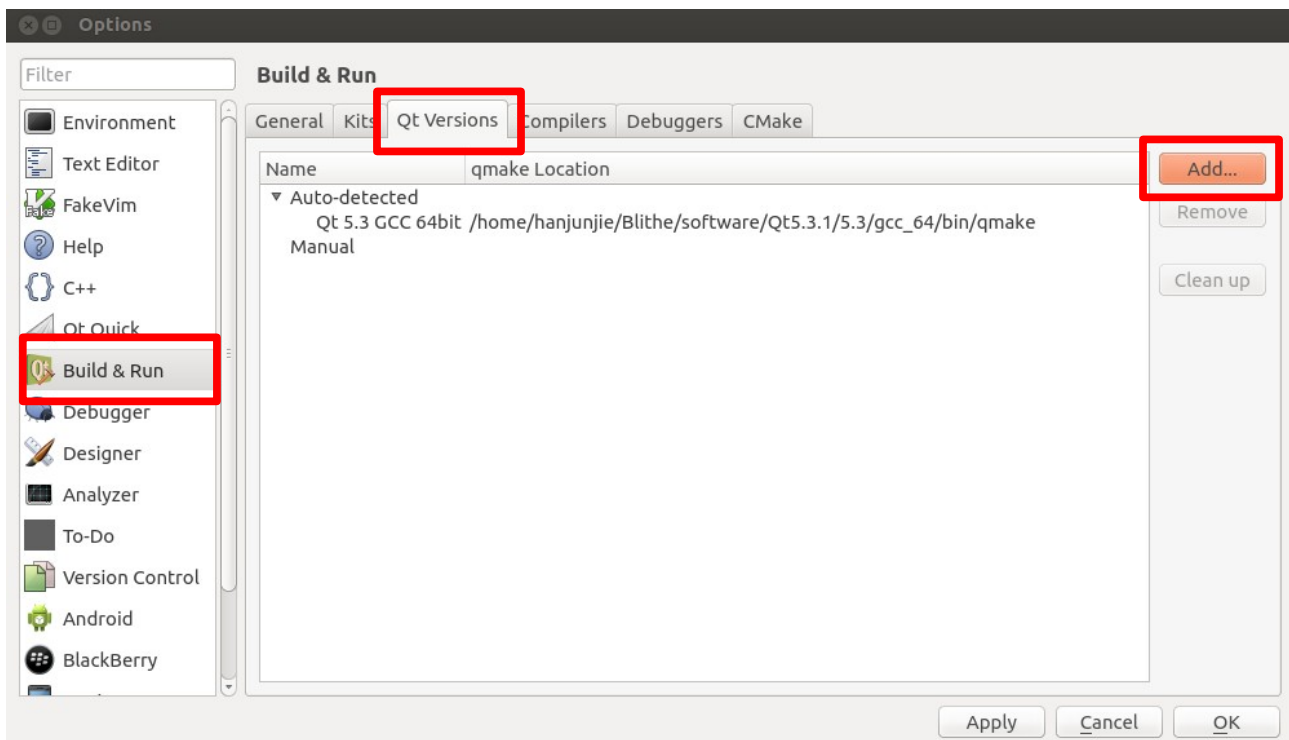
(2) Select “Qt Versions -> Add” to add three different kinds of manual qmake path one by one:

/xxx/Qt5.3.1/5.3/iautosdk/**dcue2**/native_sdk/platform/emulator/bin/qmake

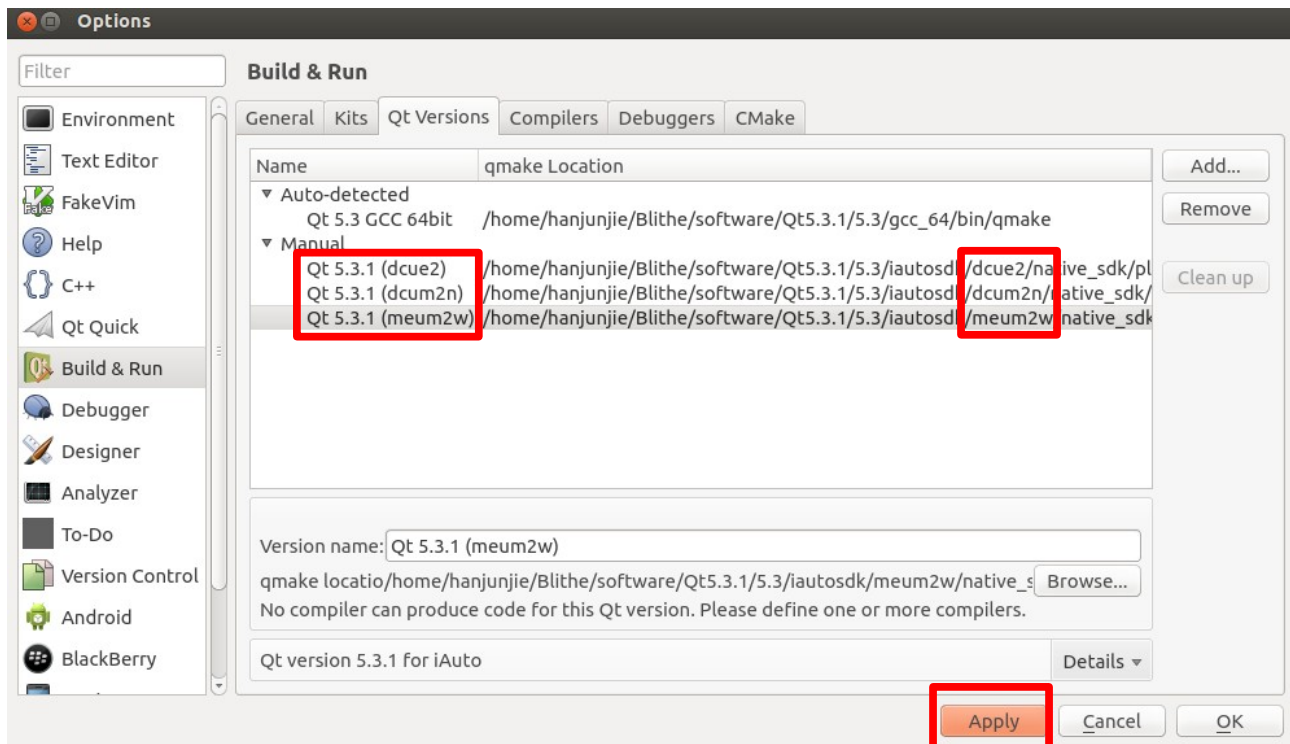
/xxx/Qt5.3.1/5.3/iautosdk/**dcum2n**/native_sdk/platform/emulator/bin/qmake

/xxx/Qt5.3.1/5.3/iautosdk/**dcum2w**/native_sdk/platform/emulator/bin/qmake

“xxx” is the path of your QtCreator.



Be sure to set meaningful version name respectively for every sdk version as below:



Click “**Apply**” to finish.

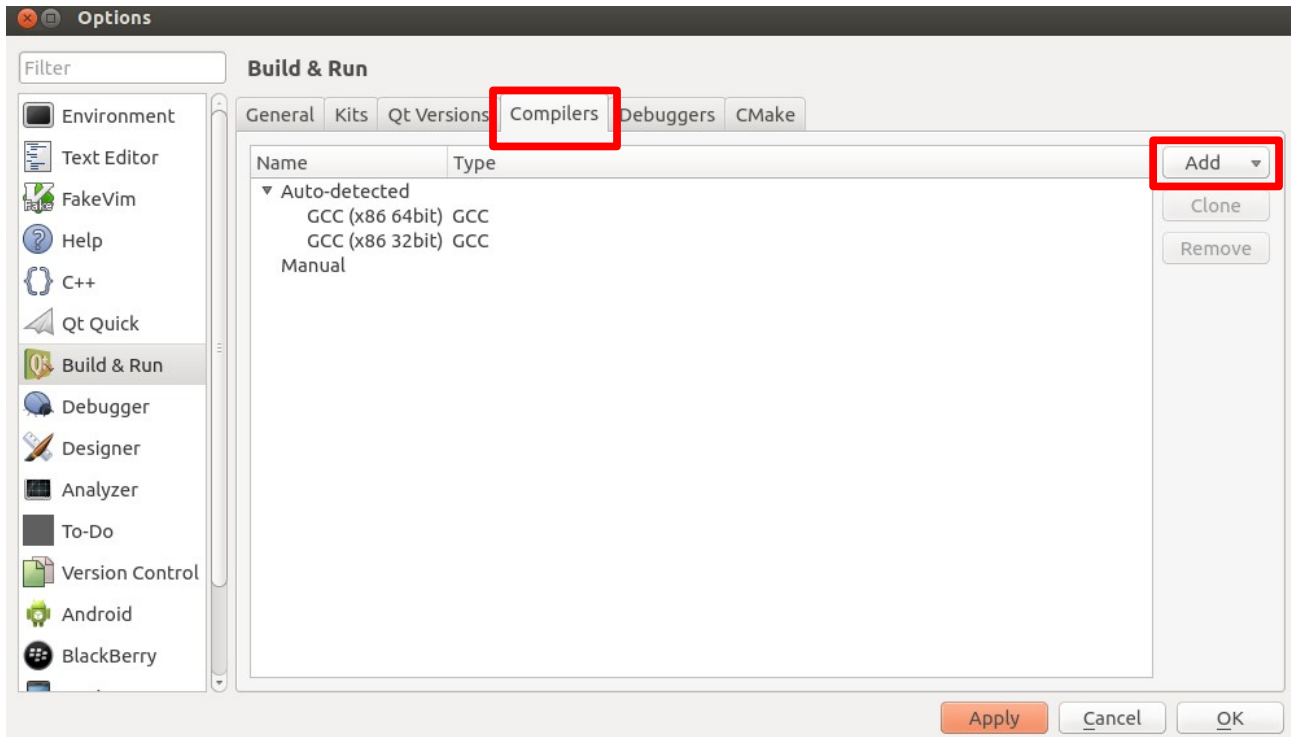
(3) Select “**Compilers -> Add -> GCC**” to add three different kinds of manual compiler path on by one:

```
/xxx/Qt5.3.1/5.3/iautosdk/dcue2/native_sdk/platform/toolchains/  
arm-poky-linux-gnueabi/4.8.3/bin/arm-poky-linux-gnueabi-g++
```

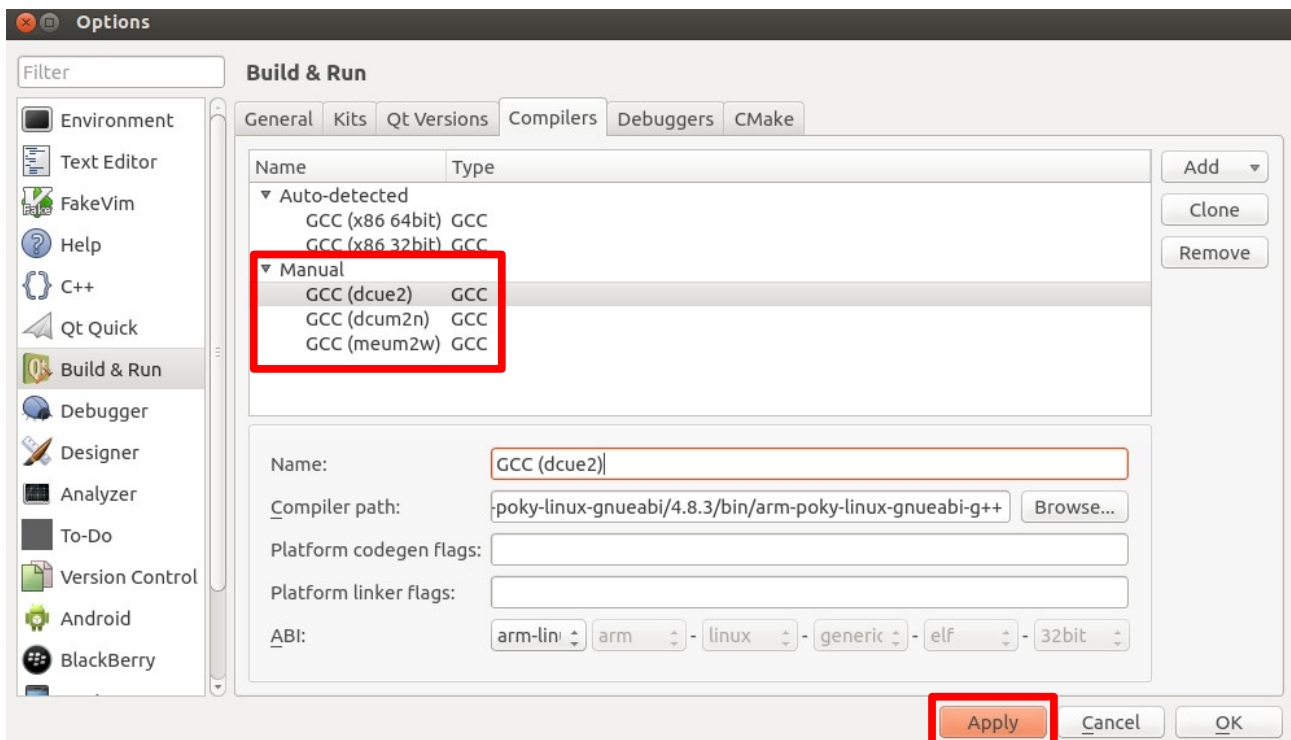
```
/xxx/Qt5.3.1/5.3/iautosdk/dcum2n/native_sdk/platform/toolchains/  
arm-poky-linux-gnueabi/4.8.3/bin/arm-poky-linux-gnueabi-g++
```

```
/xxx/Qt5.3.1/5.3/iautosdk/dcum2w/native_sdk/platform/toolchains/  
arm-poky-linux-gnueabi/4.8.3/bin/arm-poky-linux-gnueabi-g++
```

“xxx” is the path of your QtCreator.

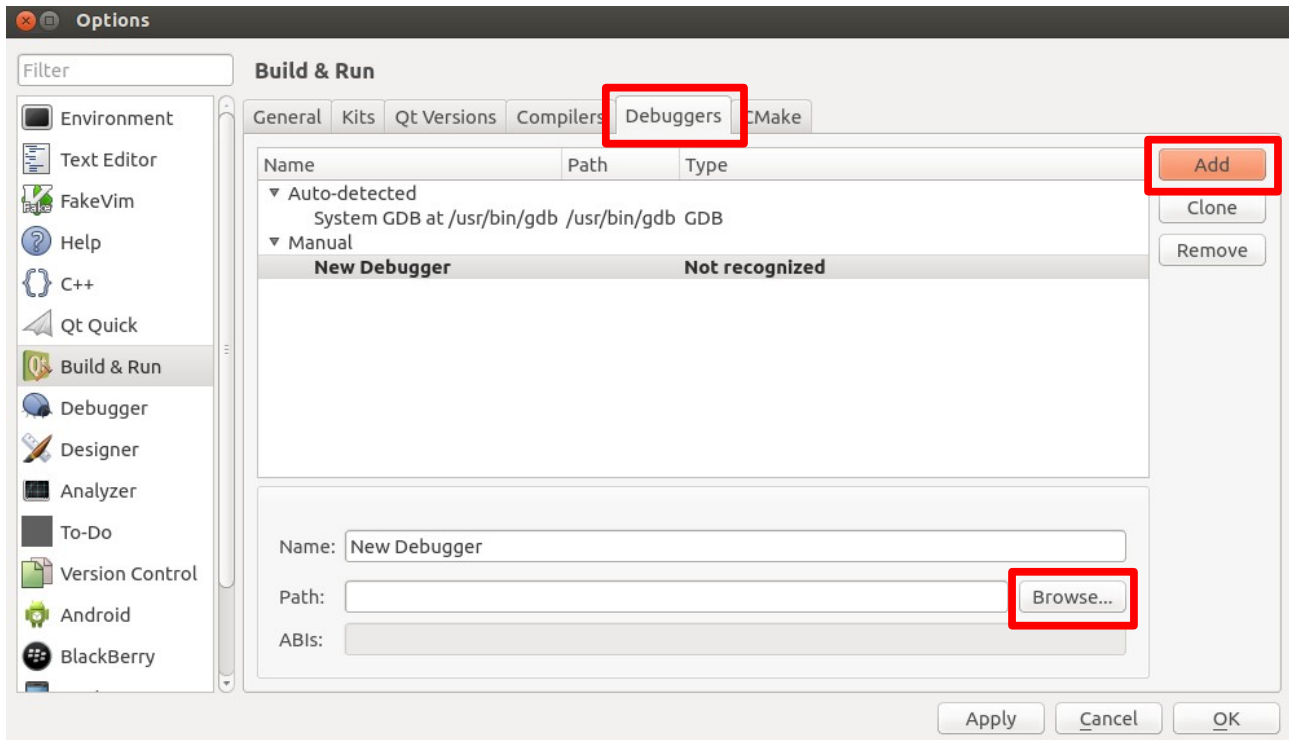


Be sure to set meaningful name respectively for every sdk compiler as below:



Click “Apply” to finish.

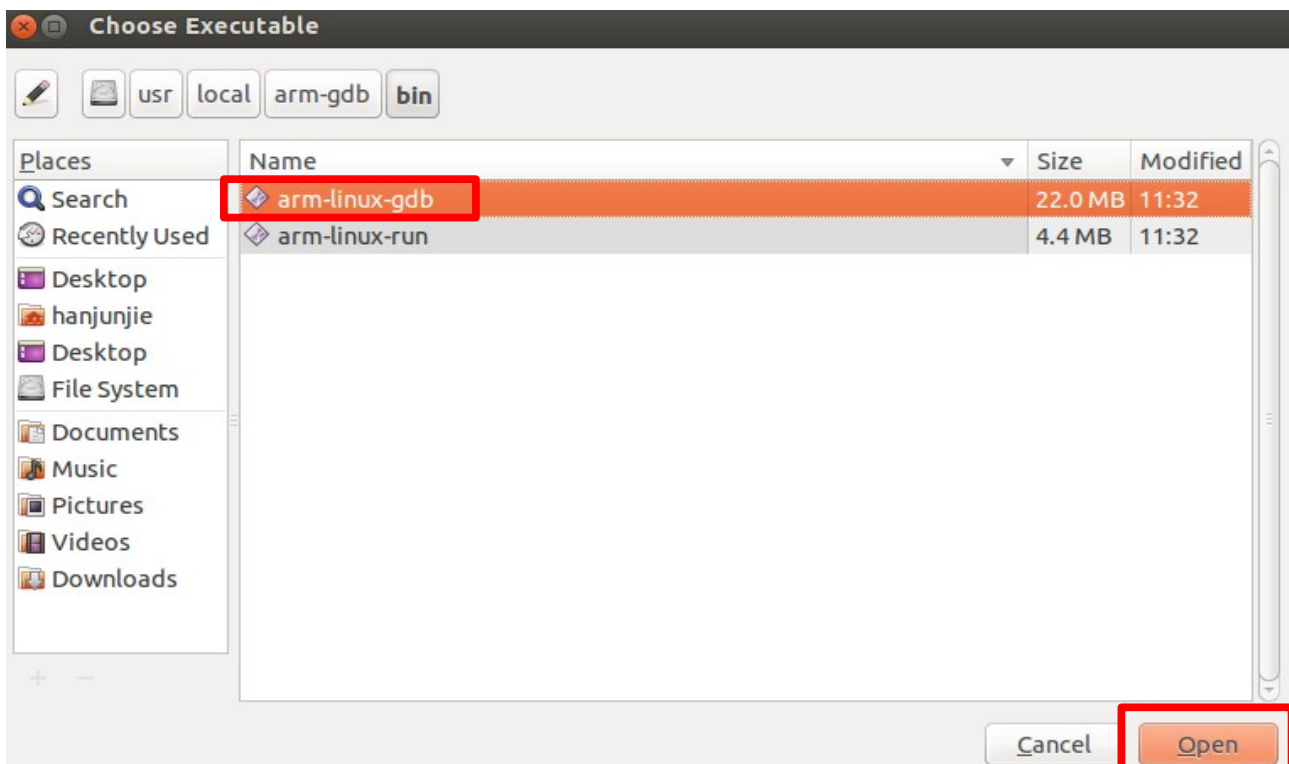
(4) Select “**Debuggers - > Add**” to add manual debugger(gdb for ARM).



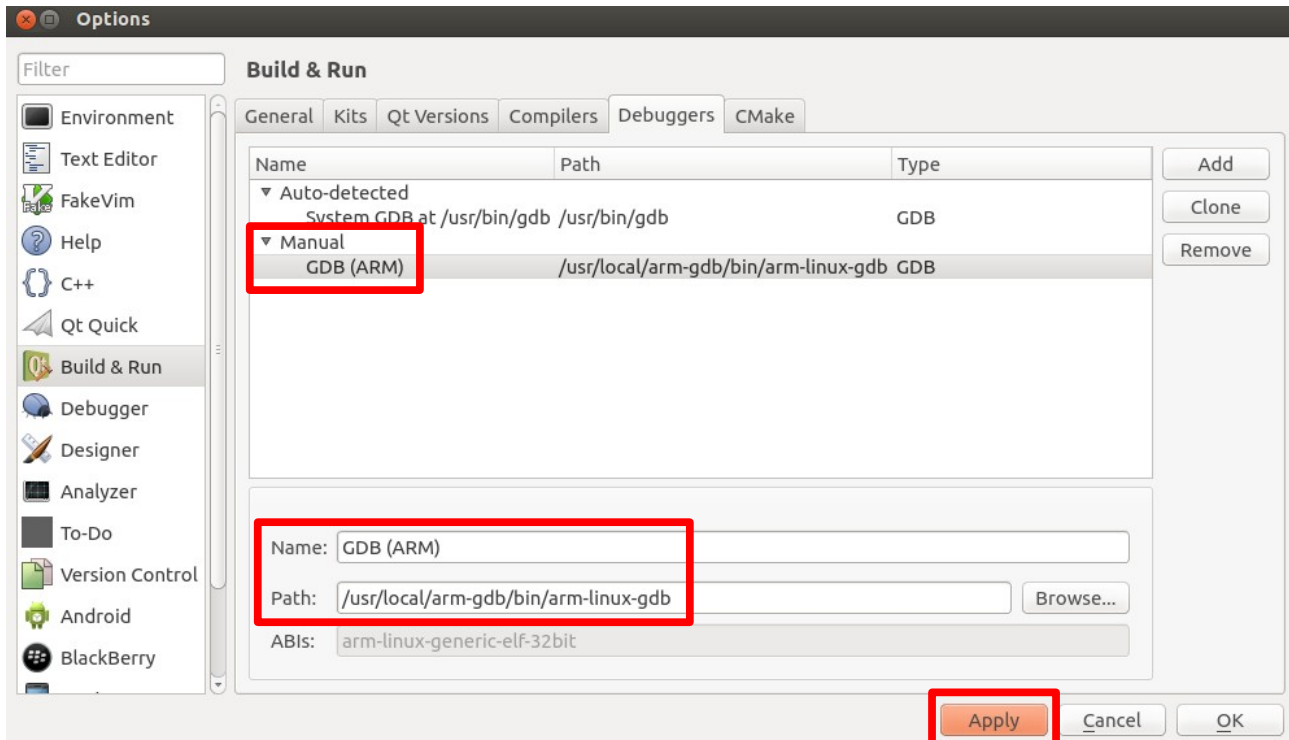
Name: change it to what you want, such as “GDB (ARM)” .

Path: click “**Browse**” to choose the right path:

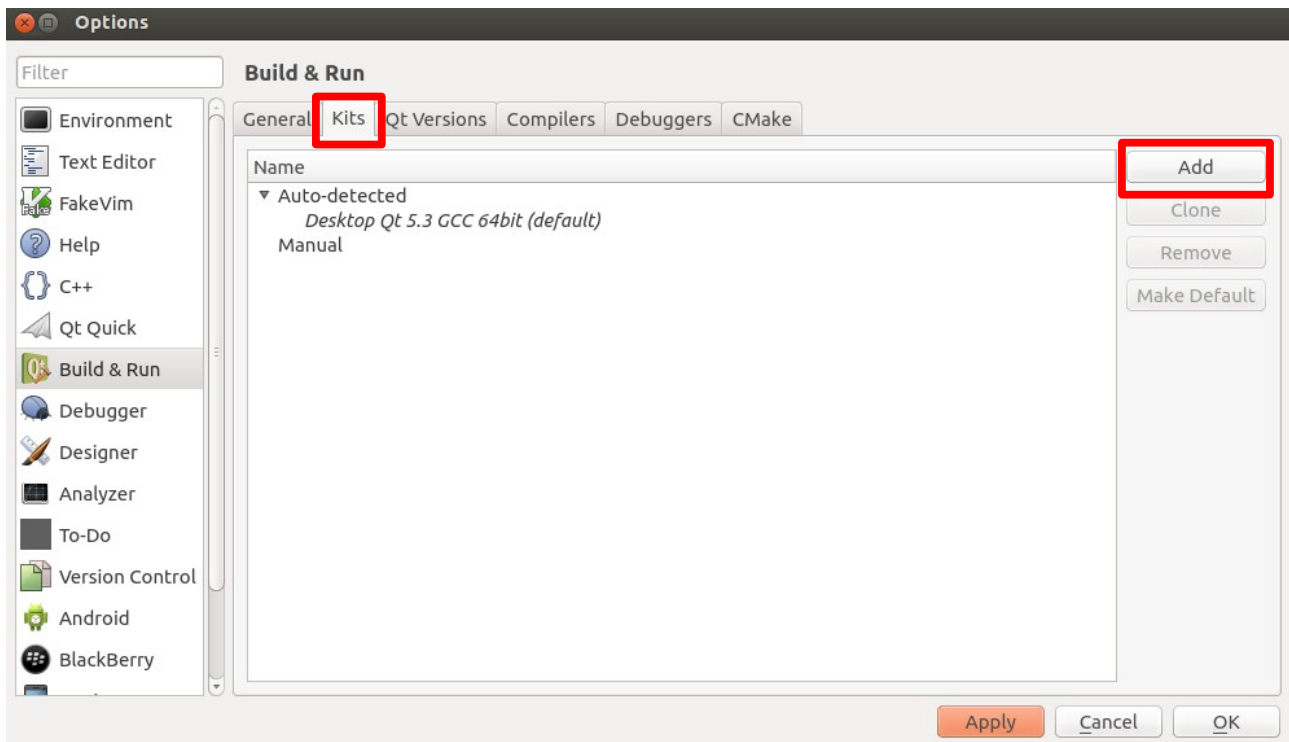
/usr/local/arm-gdb/bin/arm-linux-gdb



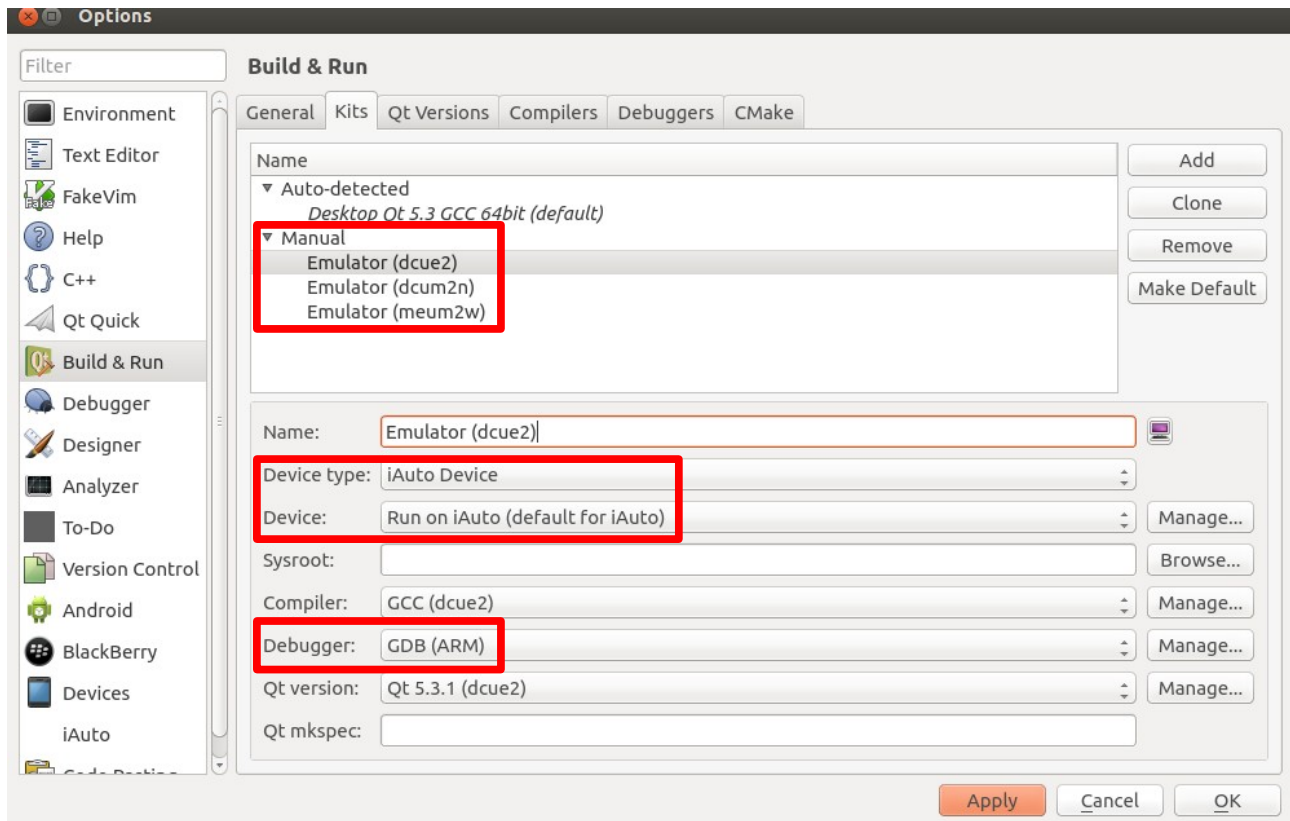
Click “Open” and “Apply” to finish.



(5) Select “Kits -> add” to add three different kinds of manual kits.



Be sure to set meaningful name respectively for every sdk kit as below:



Device Type: Select **iAuto Device** for every sdk kit.

Debugger: Select **GDB(ARM)** for every sdk kit.

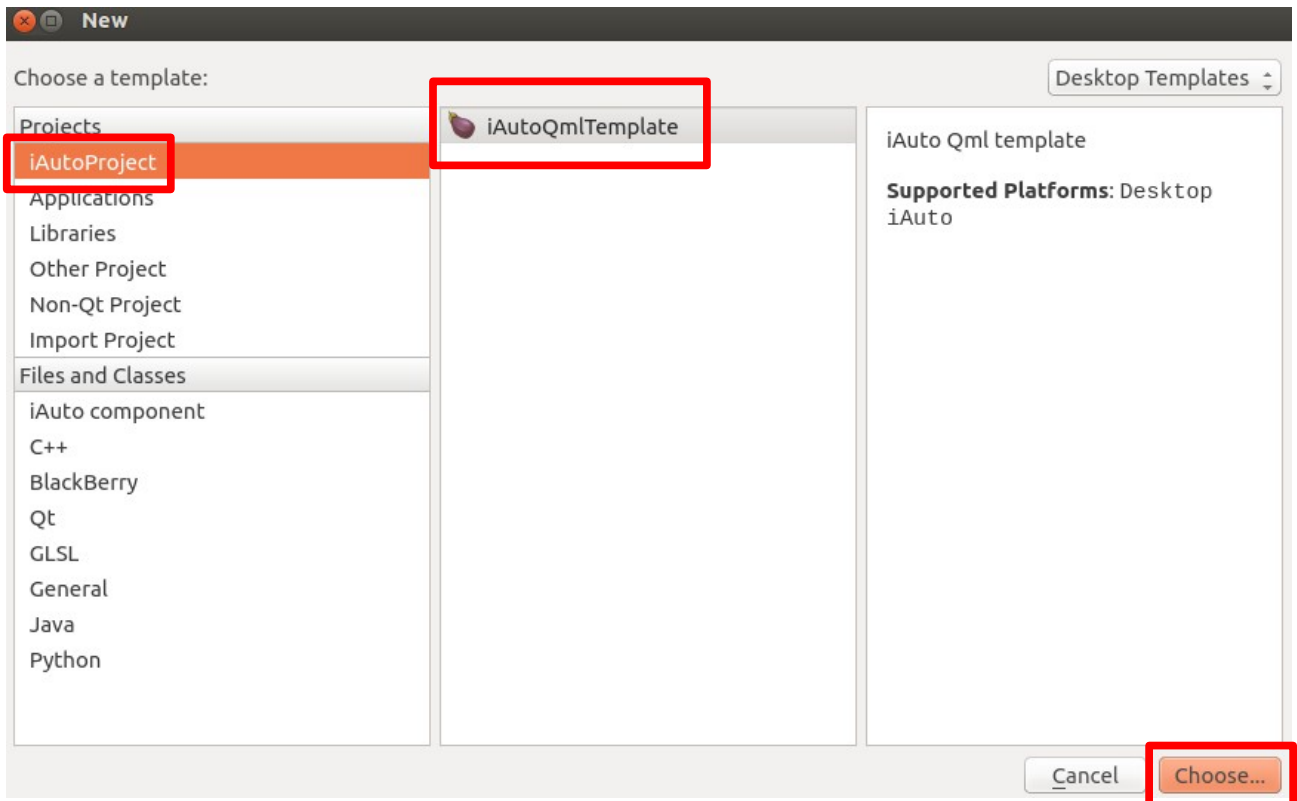
Compiler: They are not the same at all. Select respective compiler as dcue2 shows above.

Qt version: They are not the same at all. Select respective version as dcue2 shows above.

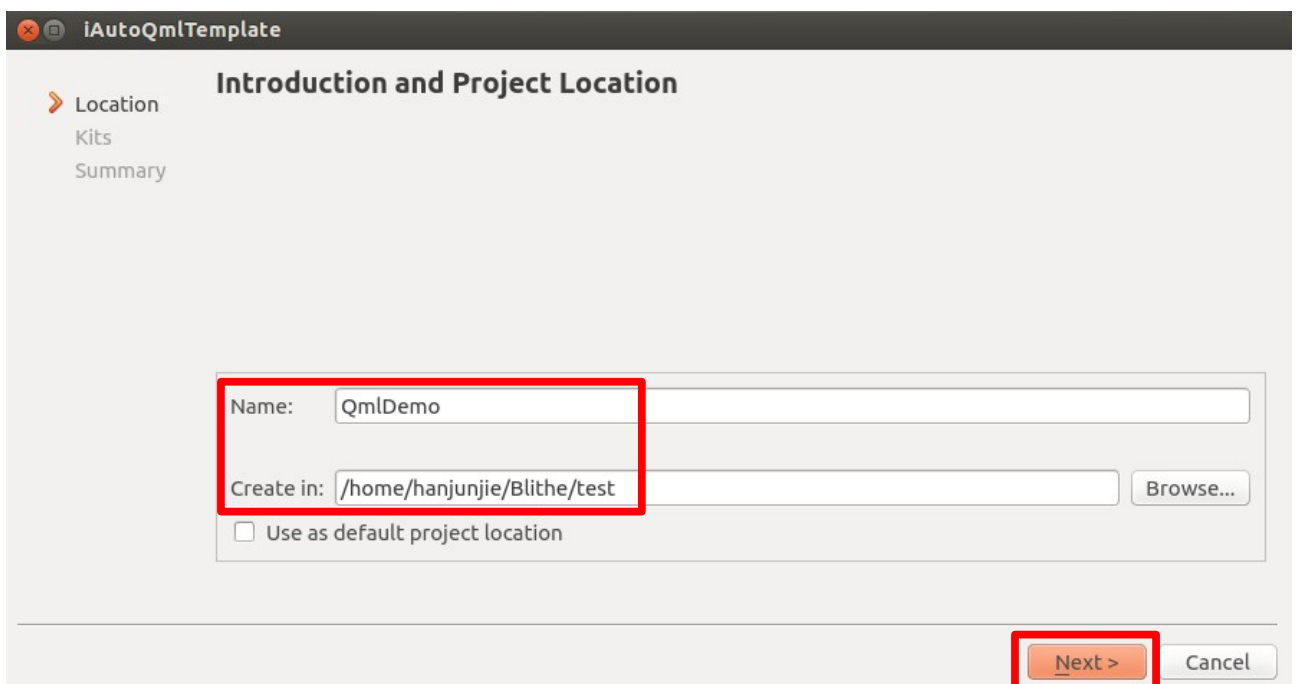
III. 17Cy Qml Project

1. Create 17Cy Qml Project

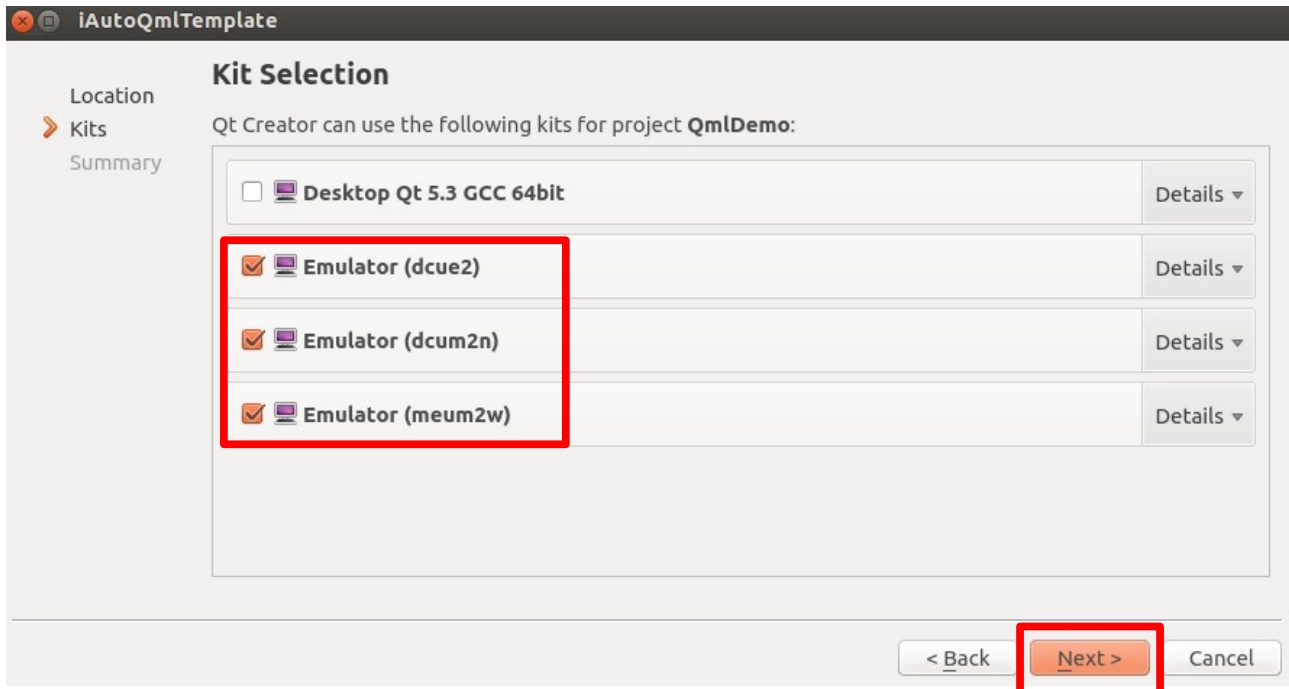
(1) Select “File -> New File or Project -> iAutoProject -> iAutoQmlTemplate -> Choose” .



(2) Write your project **name** and **location**. Then click “Next” .

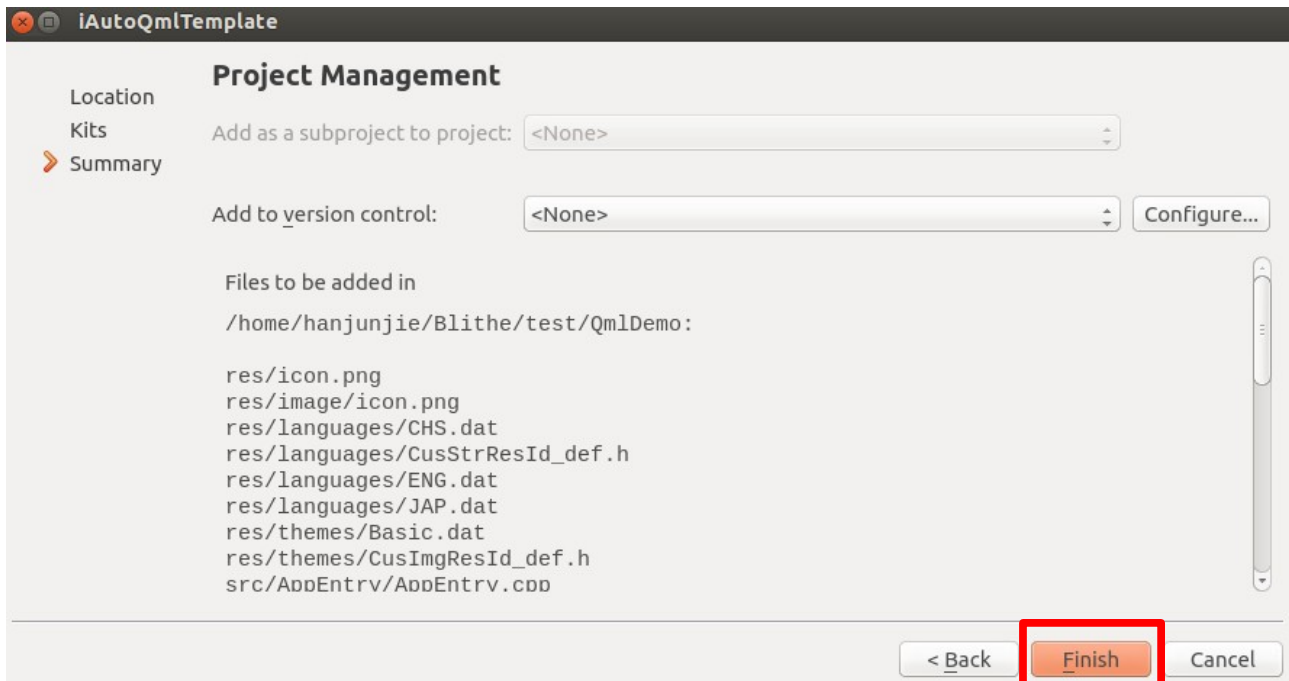


(3) Select your project kits. Then click “Next” .



(4) Click “Finish” .

Using **iAutoQmlTemplate** to create project, you may add or change some codes in the files.



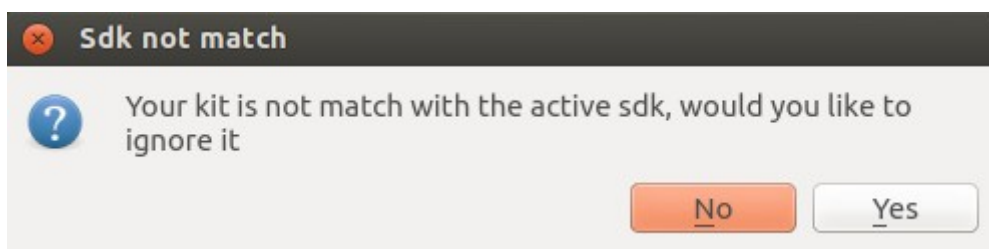
*The step (3) and (4) may execute twice for a normal project and a relative test project.

2. Run on Emulator

Important Notice ——

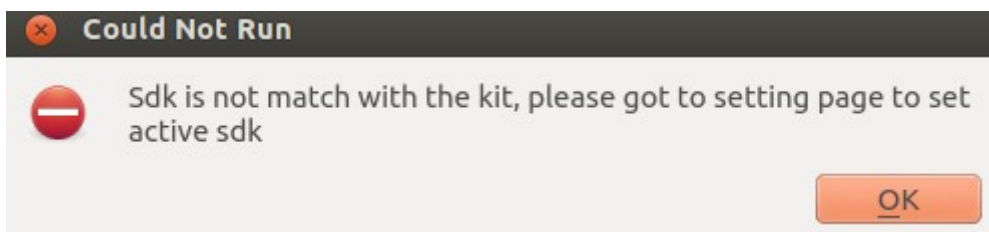
Three different kinds of sdk have been installed before, but only one is active when used according to the selected kit.

Note: Normally, the sdk related to selected kit will be set active **automatically**. If selected kit does not match the current active sdk, there will be a **notice table** as below.



“Yes” : Click “Yes” to ignore it.

“No” : Click “No” to fix it, and there will be a **notice table** as below.

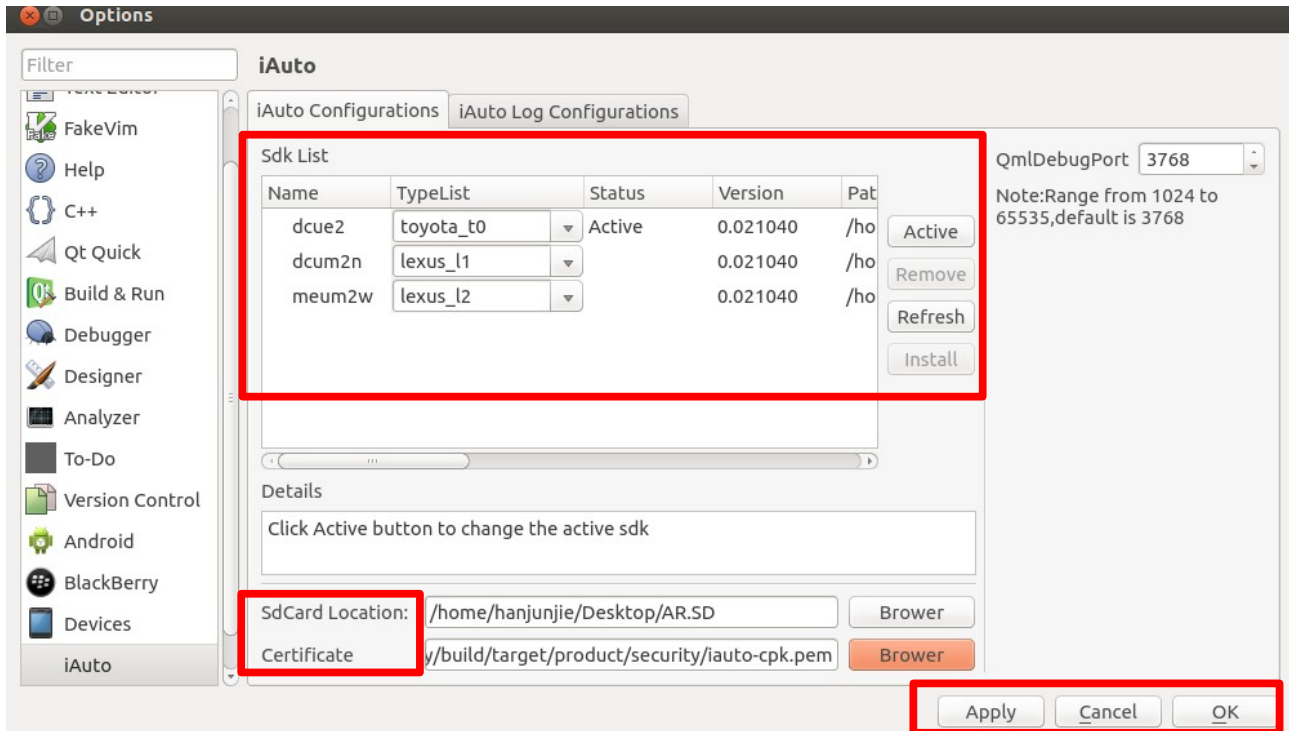


“OK” : Click “OK” to continue and **re-select one kit or reset sdk** matched the current kit active manually.

Set SDK active manually ——

Click **“Tools - > Options - > iAuto - > iAuto Configurations”** .

It shows the sdk list information as below.



“Name” : There are three kinds of sdk, dcue2, dcum2n, and meum2w.

“TypeList” : Every sdk has a few different types.

dcue2: toyota_t0, toyota_t1, toyota_t2.

dcum2n: lexus_l1, lexus_l2, toyota_temv.

meum2w: toyota_t2, lexus_l2, toyota_temv.

“Status” : Shows the current active sdk, and the active sdk is unique.

“Version” : The sdk version number.

“Path” : The sdk installation path.

“Active” : Click to set one sdk active. Select the sdk and its type firstly.

“Remove” : Click to remove the selected sdk.

“Refresh” : Click to Refresh the sdk status.

“Install” : Be reserved.

“SdCard Location” : Click “Browser” to install sd card of your local path.

“Certificate” : Click “Browser” to install key for cpk file. There are two different kinds of keys for system app and non-system app.

System key: 17Cy repository/build/target/product/security/iauto-cpk.pem

Non-system key: Using “cpktool -r” command. Please refer to wiki:

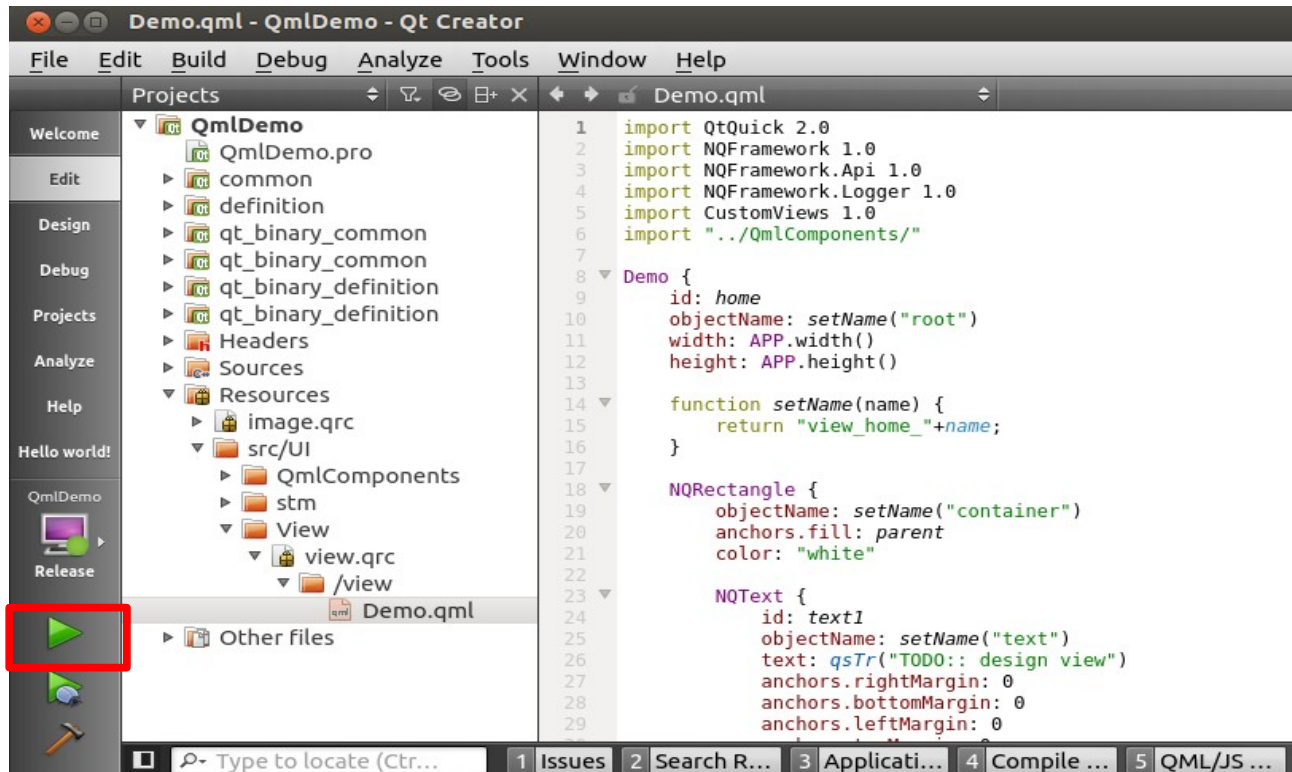
<http://review.iauto.net/develophelp/iAuto-App-Package>

For example, select **dcue2** sdk and type of **toyota_t0**, click **“Active”** .

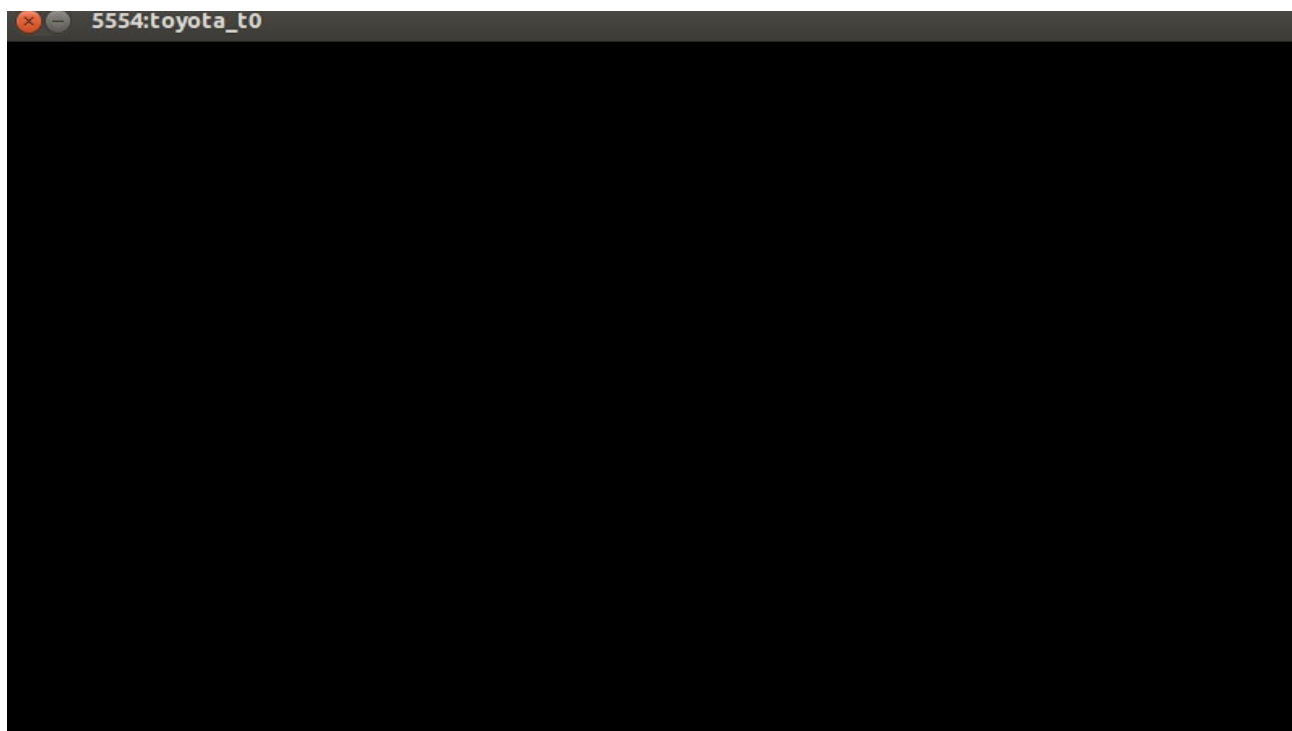
Then click **“AppLy -> OK”** to finish.

Run —

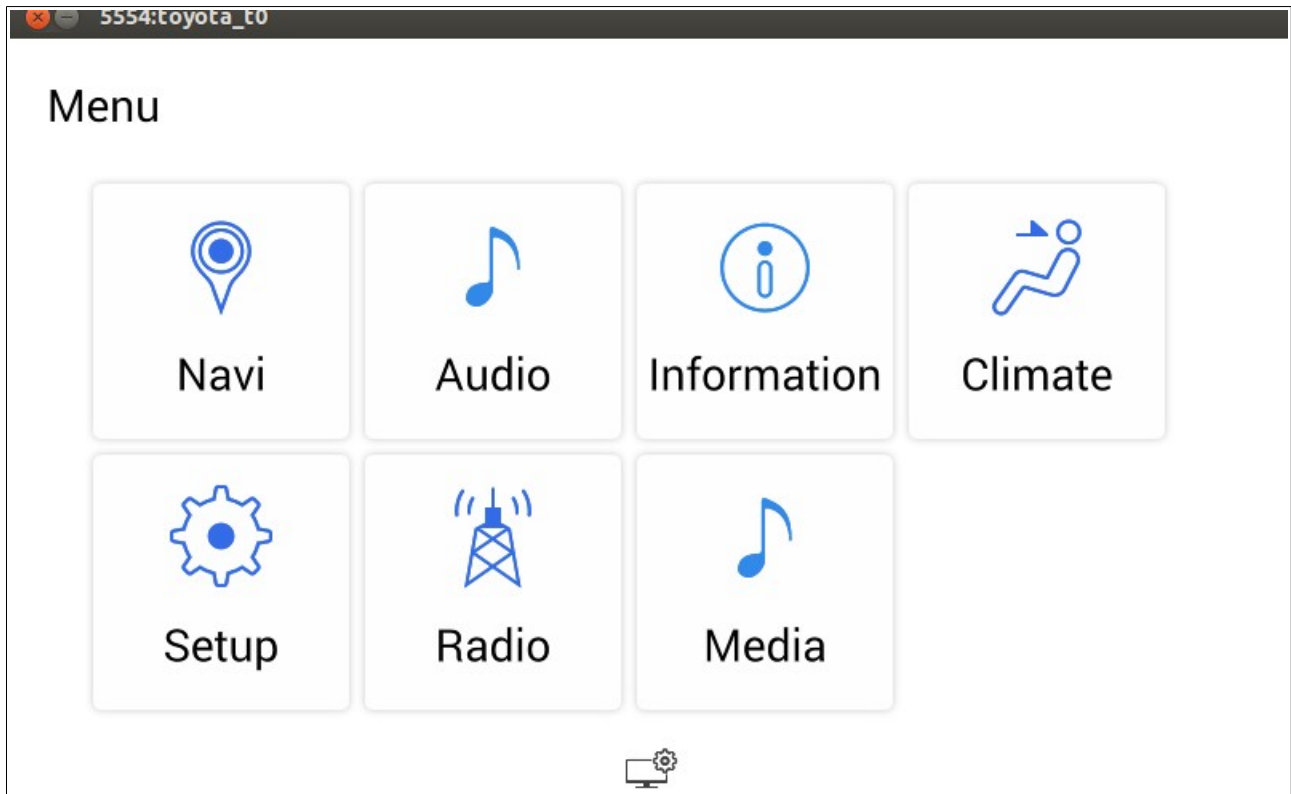
Click “Run” button or press “Ctrl+F5” .



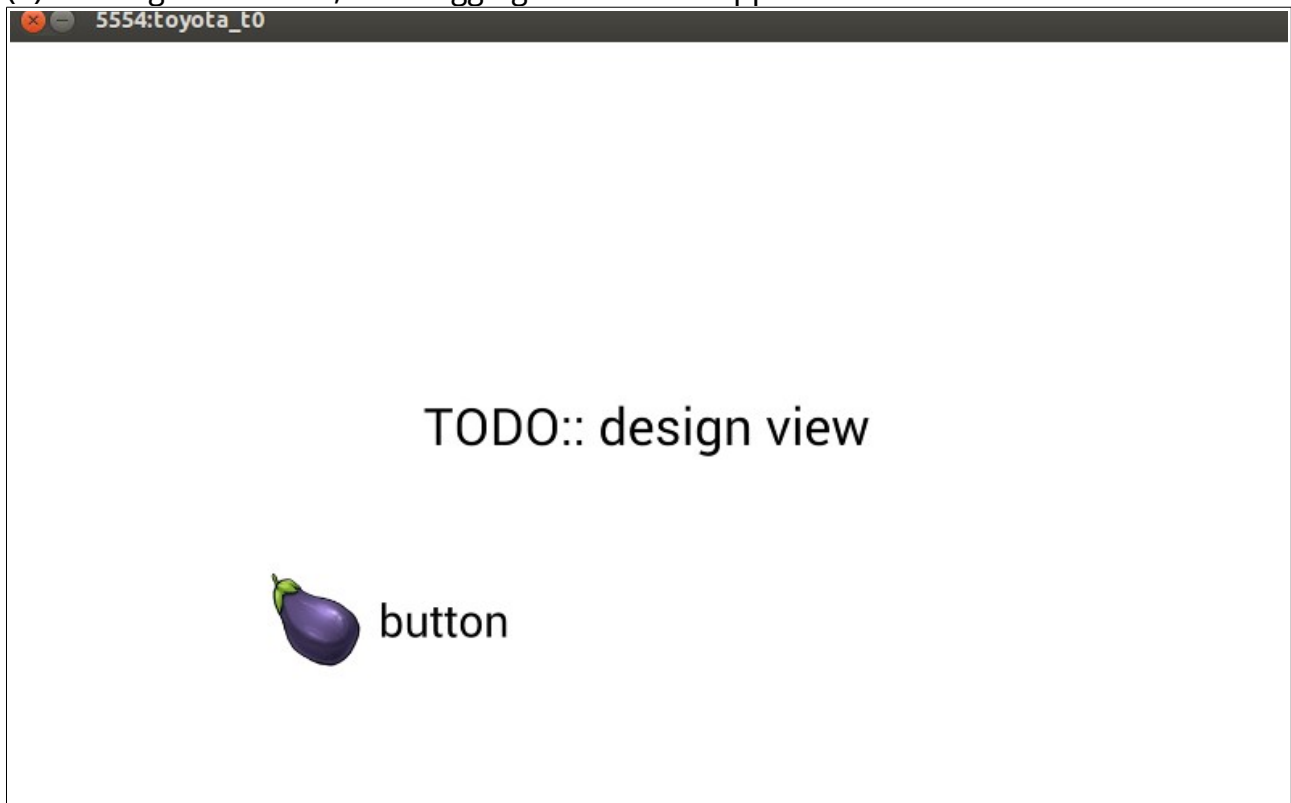
(1) Emulator begins with a black screen:



(2) Entering into home screen after a few seconds.



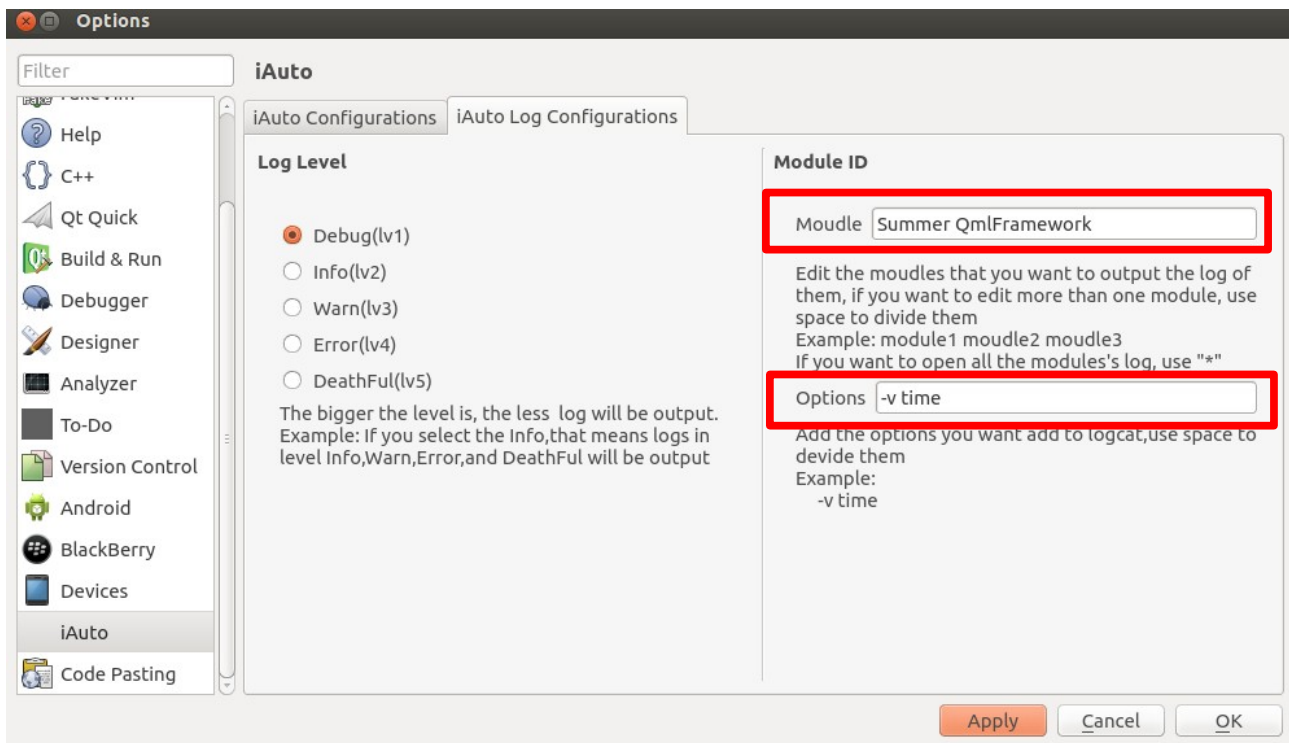
(3) Loading for a while , then logging into custom application.



IV. Debugging

1. Log Output

(1) Click “Tools -> Options -> iAuto -> iAuto Log Configurations” .



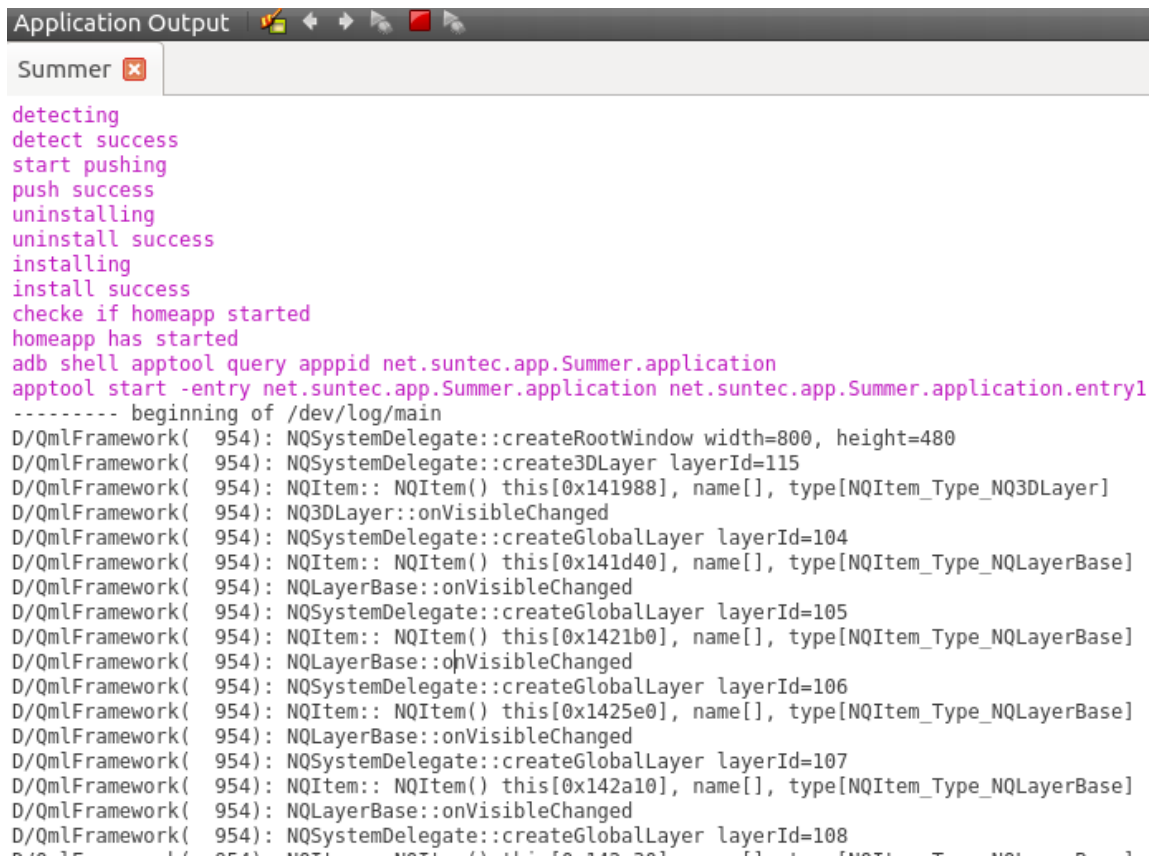
(2) Select one “Log Level” , and add necessary “Module” , and optional “Options” , then click “Apply -> OK” to finish.

“Log Level” : There are five different log level. For example, select the debug level here. Debug level here is related to `NGLogDebug()` in C++ codes and `NQLogger.debug()` in qml codes.

“Module” : Add your log tag to output the relative log, for example, add `QmlFramework` and `Summer` module here.

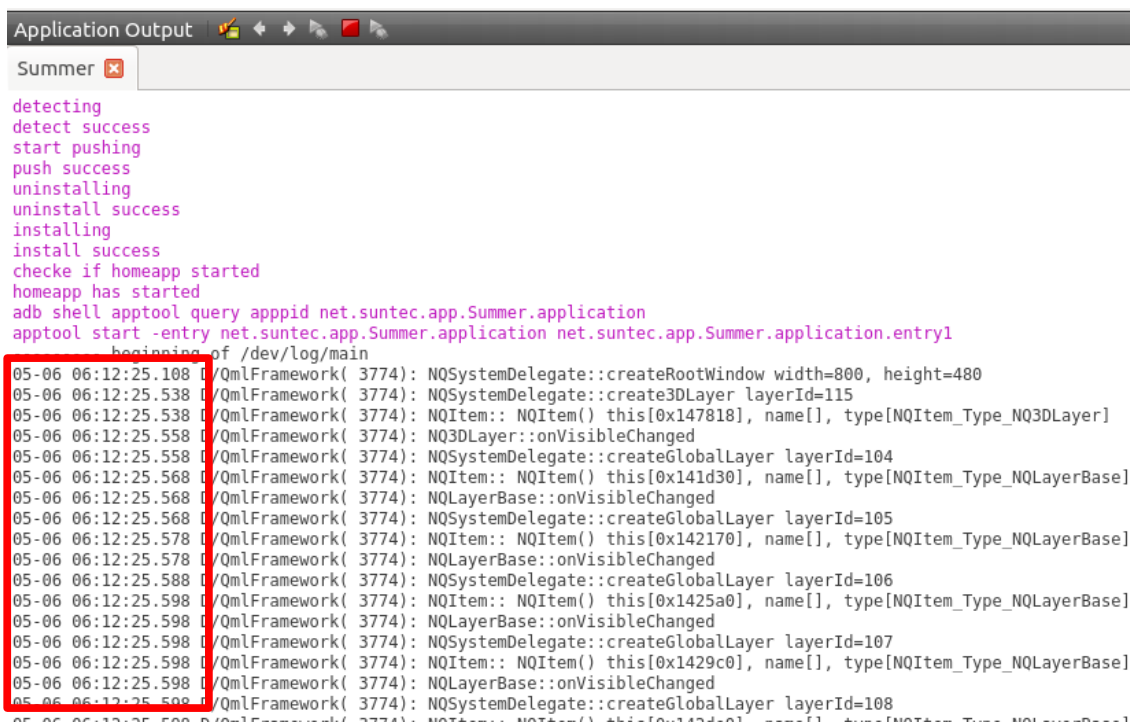
“Options” : Add options to logcat, for example, add “-v time” will output date and time information in every log.

(3) QtCreator Application Output shows the logs(Options without params):



```
Application Output
Summer
detecting
detect success
start pushing
push success
uninstalling
uninstall success
installing
install success
checke if homeapp started
homeapp has started
adb shell apptool query apppid net.suntec.app.Summer.application
apptool start -entry net.suntec.app.Summer.application net.suntec.app.Summer.application.entry1
----- beginning of /dev/log/main
D/QmlFramework( 954): NQSystemDelegate::createRootWindow width=800, height=480
D/QmlFramework( 954): NQSystemDelegate::create3DLayer layerId=115
D/QmlFramework( 954): NQItem:: NQItem() this[0x141988], name[], type[NQItem_Type_NQ3DLayer]
D/QmlFramework( 954): NQ3DLayer::onVisibleChanged
D/QmlFramework( 954): NQSystemDelegate::createGlobalLayer layerId=104
D/QmlFramework( 954): NQItem:: NQItem() this[0x141d40], name[], type[NQItem_Type_NQLayerBase]
D/QmlFramework( 954): NQLayerBase::onVisibleChanged
D/QmlFramework( 954): NQSystemDelegate::createGlobalLayer layerId=105
D/QmlFramework( 954): NQItem:: NQItem() this[0x1421b0], name[], type[NQItem_Type_NQLayerBase]
D/QmlFramework( 954): NQLayerBase::onVisibleChanged
D/QmlFramework( 954): NQSystemDelegate::createGlobalLayer layerId=106
D/QmlFramework( 954): NQItem:: NQItem() this[0x1425e0], name[], type[NQItem_Type_NQLayerBase]
D/QmlFramework( 954): NQLayerBase::onVisibleChanged
D/QmlFramework( 954): NQSystemDelegate::createGlobalLayer layerId=107
D/QmlFramework( 954): NQItem:: NQItem() this[0x142a10], name[], type[NQItem_Type_NQLayerBase]
D/QmlFramework( 954): NQLayerBase::onVisibleChanged
D/QmlFramework( 954): NQSystemDelegate::createGlobalLayer layerId=108
```

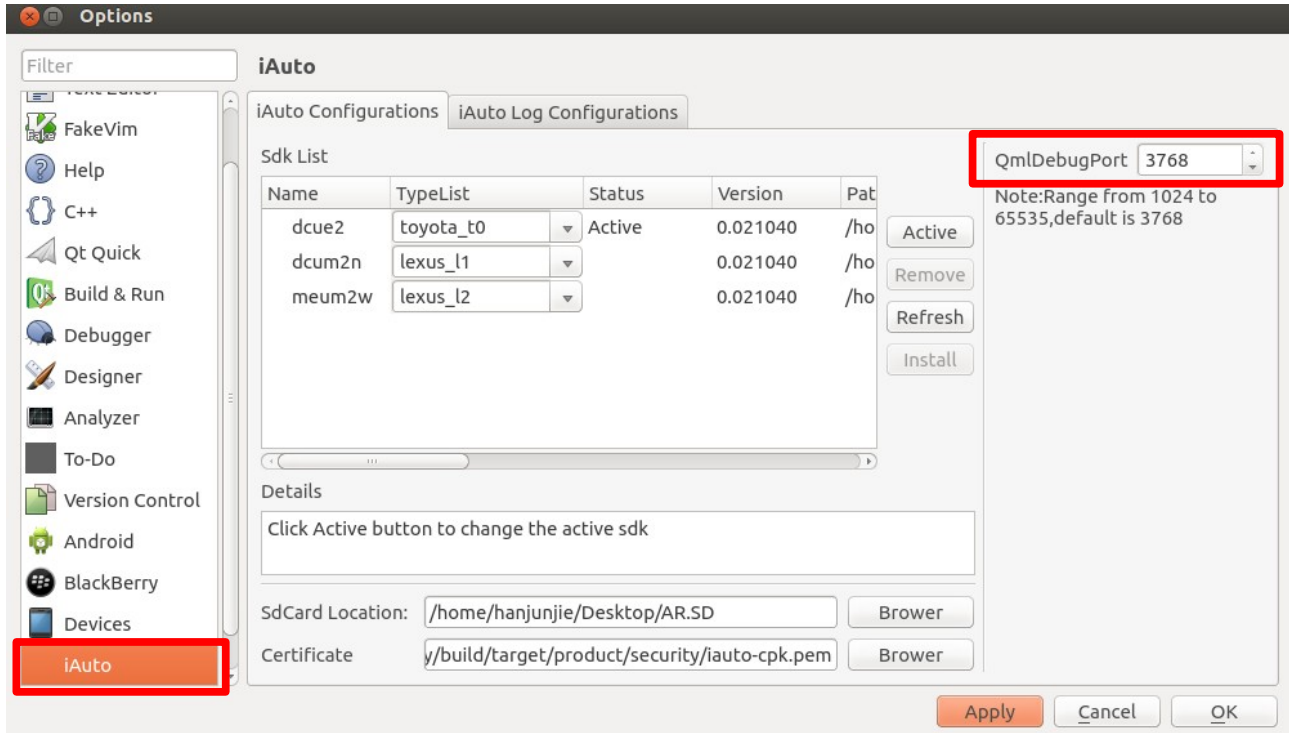
QtCreator Application Output shows the logs(Options with '-v time'):



```
Application Output
Summer
detecting
detect success
start pushing
push success
uninstalling
uninstall success
installing
install success
checke if homeapp started
homeapp has started
adb shell apptool query apppid net.suntec.app.Summer.application
apptool start -entry net.suntec.app.Summer.application net.suntec.app.Summer.application.entry1
----- beginning of /dev/log/main
05-06 06:12:25.108 D/QmlFramework( 3774): NQSystemDelegate::createRootWindow width=800, height=480
05-06 06:12:25.538 D/QmlFramework( 3774): NQSystemDelegate::create3DLayer layerId=115
05-06 06:12:25.538 D/QmlFramework( 3774): NQItem:: NQItem() this[0x147818], name[], type[NQItem_Type_NQ3DLayer]
05-06 06:12:25.558 D/QmlFramework( 3774): NQ3DLayer::onVisibleChanged
05-06 06:12:25.558 D/QmlFramework( 3774): NQSystemDelegate::createGlobalLayer layerId=104
05-06 06:12:25.568 D/QmlFramework( 3774): NQItem:: NQItem() this[0x141d30], name[], type[NQItem_Type_NQLayerBase]
05-06 06:12:25.568 D/QmlFramework( 3774): NQLayerBase::onVisibleChanged
05-06 06:12:25.568 D/QmlFramework( 3774): NQSystemDelegate::createGlobalLayer layerId=105
05-06 06:12:25.578 D/QmlFramework( 3774): NQItem:: NQItem() this[0x142170], name[], type[NQItem_Type_NQLayerBase]
05-06 06:12:25.578 D/QmlFramework( 3774): NQLayerBase::onVisibleChanged
05-06 06:12:25.588 D/QmlFramework( 3774): NQSystemDelegate::createGlobalLayer layerId=106
05-06 06:12:25.598 D/QmlFramework( 3774): NQItem:: NQItem() this[0x1425a0], name[], type[NQItem_Type_NQLayerBase]
05-06 06:12:25.598 D/QmlFramework( 3774): NQLayerBase::onVisibleChanged
05-06 06:12:25.598 D/QmlFramework( 3774): NQSystemDelegate::createGlobalLayer layerId=107
05-06 06:12:25.598 D/QmlFramework( 3774): NQItem:: NQItem() this[0x1429c0], name[], type[NQItem_Type_NQLayerBase]
05-06 06:12:25.598 D/QmlFramework( 3774): NQLayerBase::onVisibleChanged
05-06 06:12:25.598 D/QmlFramework( 3774): NQSystemDelegate::createGlobalLayer layerId=108
```

2. Debug with Breakpoints

(1) Click “Tools -> Options -> iAuto” to set “QmlDebugPort” on the right panel.



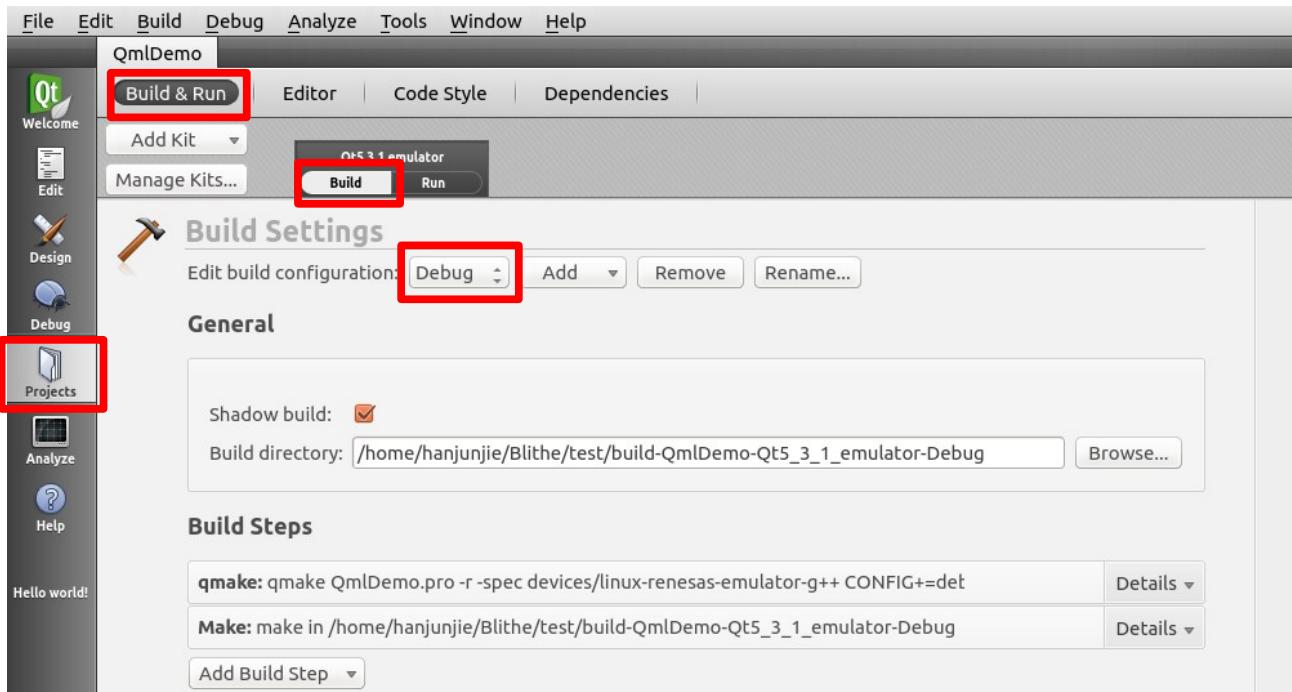
(2) Add breakpoints on C++ codes.

C++ breakpoints:

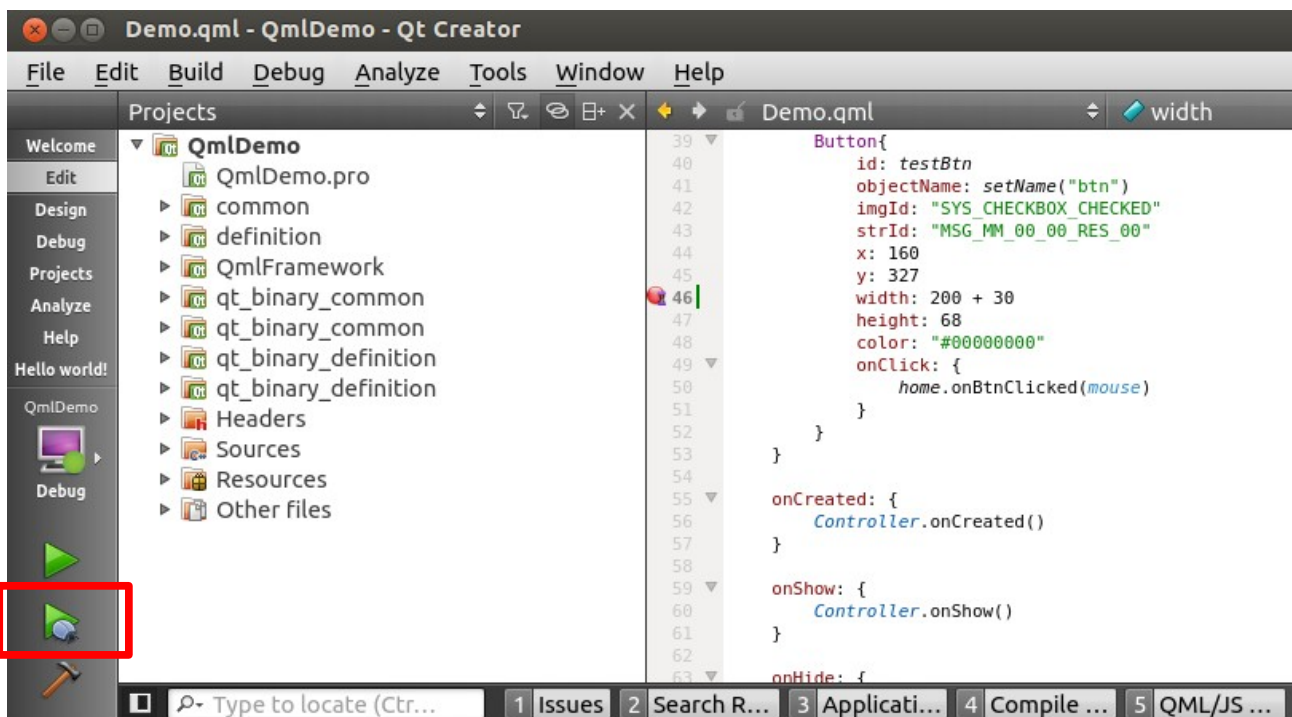
The screenshot shows a snippet of C++ code in a Qt IDE. A red circle highlights a breakpoint icon on line 26. The code is as follows:

```
26 void Demo::onCreated(){
27     this->setClip(true);
28     m_container = this->findChild<NRectangle*>("view_home_container");
29     m_btn = this->findChild<NRectangle*>("view_home_btn");
30     m_text = this->findChild<NText*>("view_home_text");
31 }
```

(3) Click “Projects -> Build & Run -> Build -> Debug” to select debug mode.

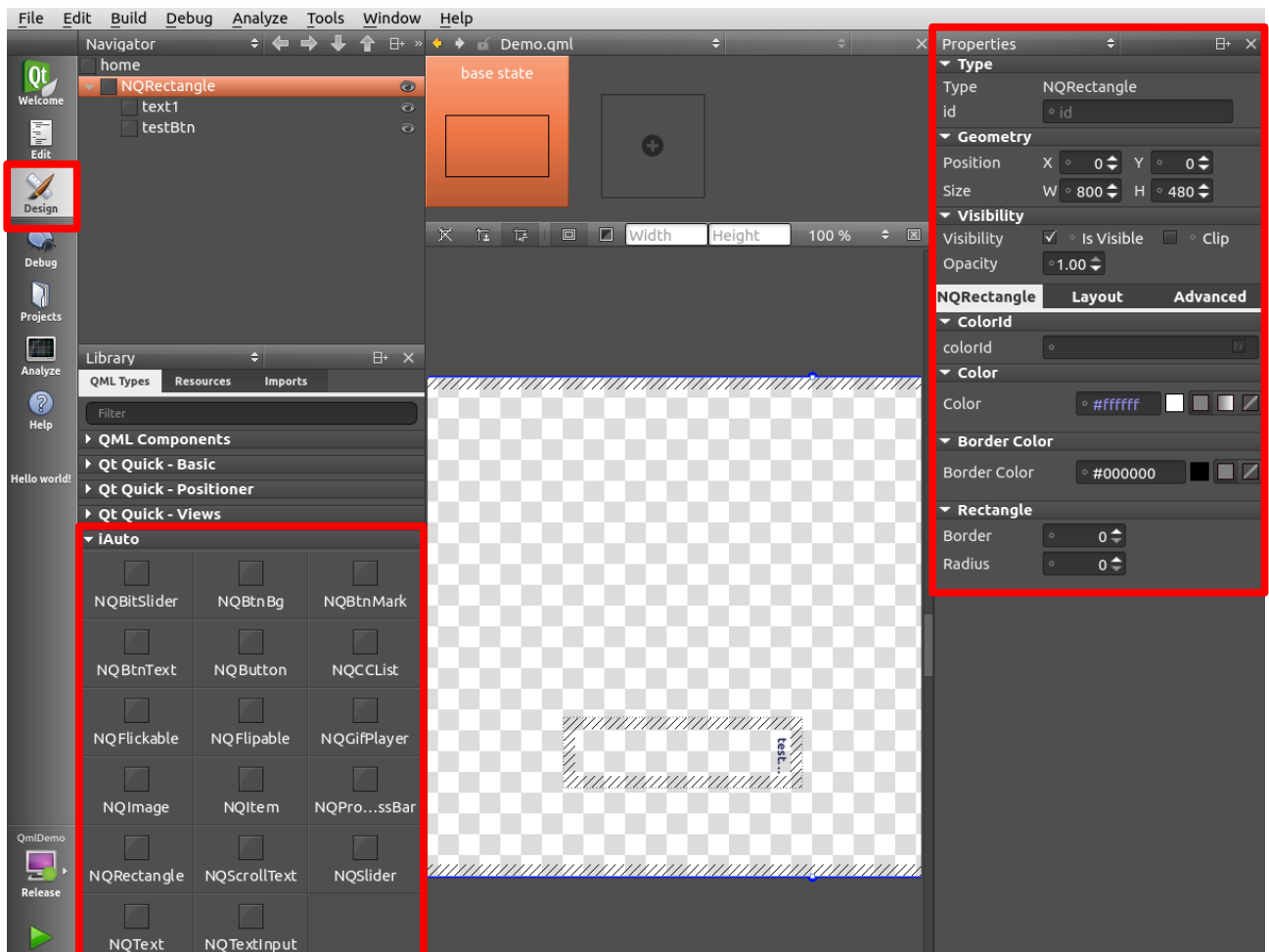


(4) Click “Start Debugging” button or press “F5” .



V. QtDesigner

On our 17Cy Qml Project, we can use “**Designer**” tool. There are a lot of **iAuto QML Types** (NQXXX) available to be used for UI.



We can drag **iAuto QML components** to our workspace and change the **properties** on the right panel.

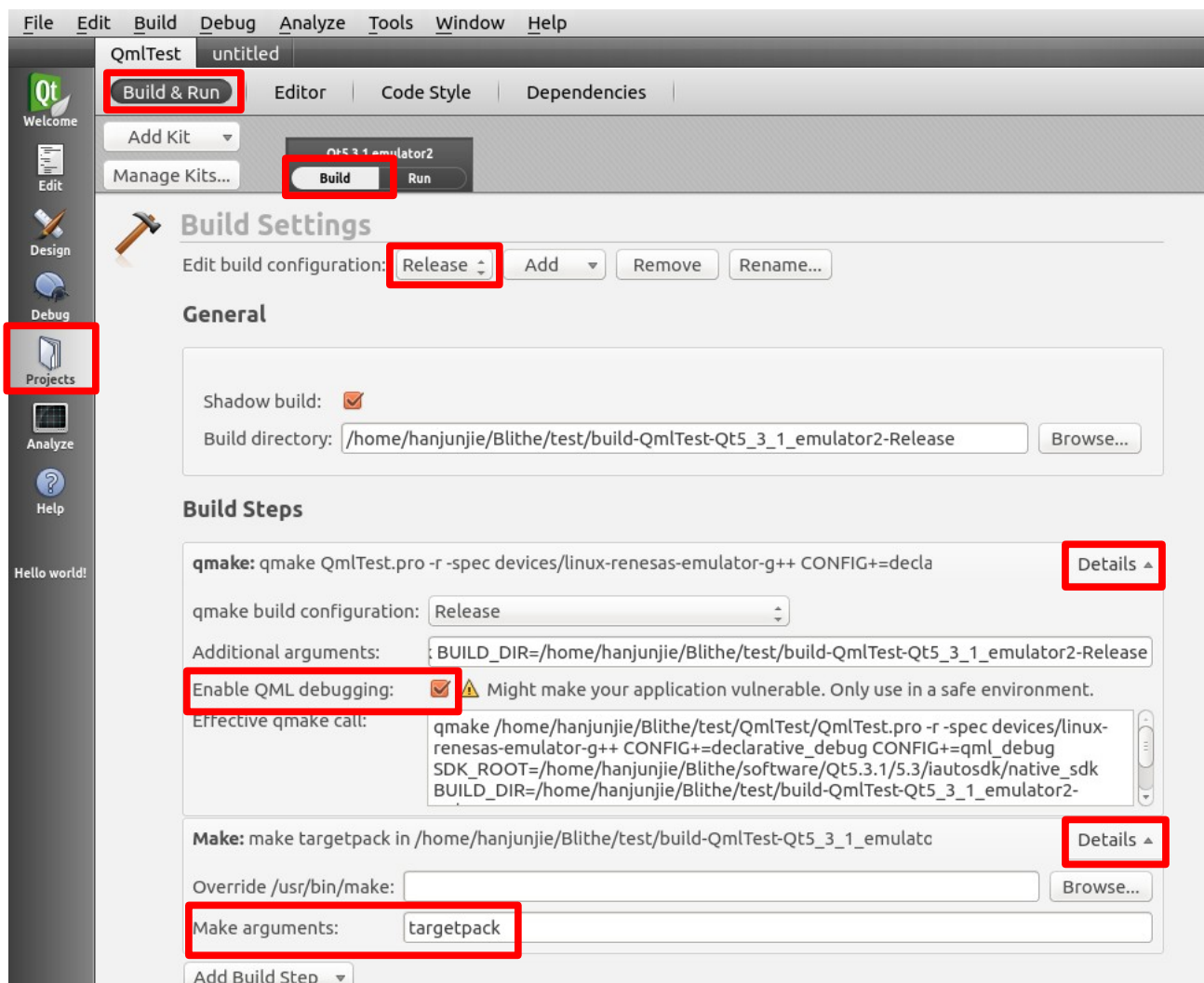
VI. App Performance Analyzing

1、Configuration

Click “**Projects - > Build & Run - > Build - > Release**” .

(1) Click “**Details**” of “qmake” , then select “**Enable QML debugging**” .

(2) Click “**Details**” of “Make” , then add “**targetpack**” to “**Make arguments**” .

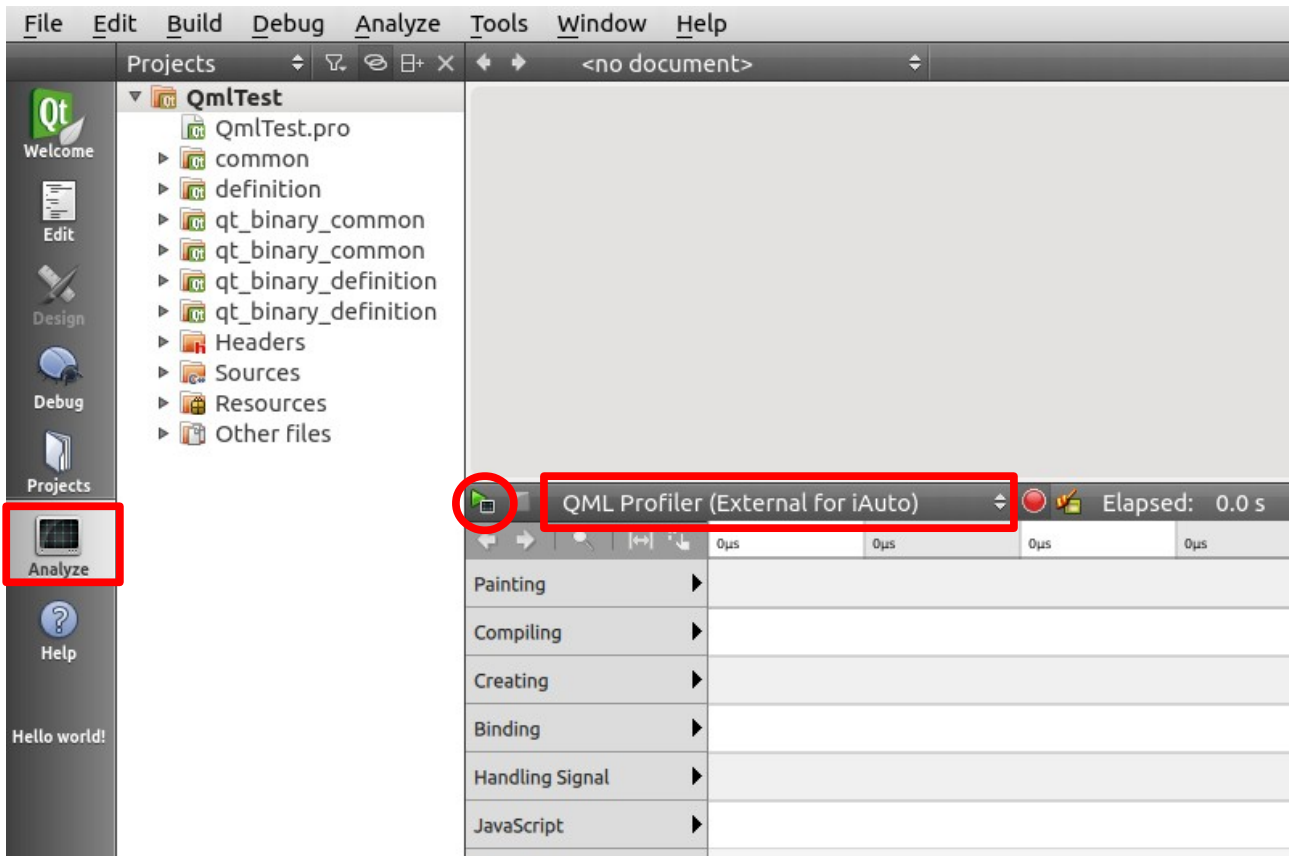


2、Using QML Profiler

Note: Select “**Release**” of the kit to start. The **emulator** can be launched firstly or not, but remember to stop the app before using qml profiler.

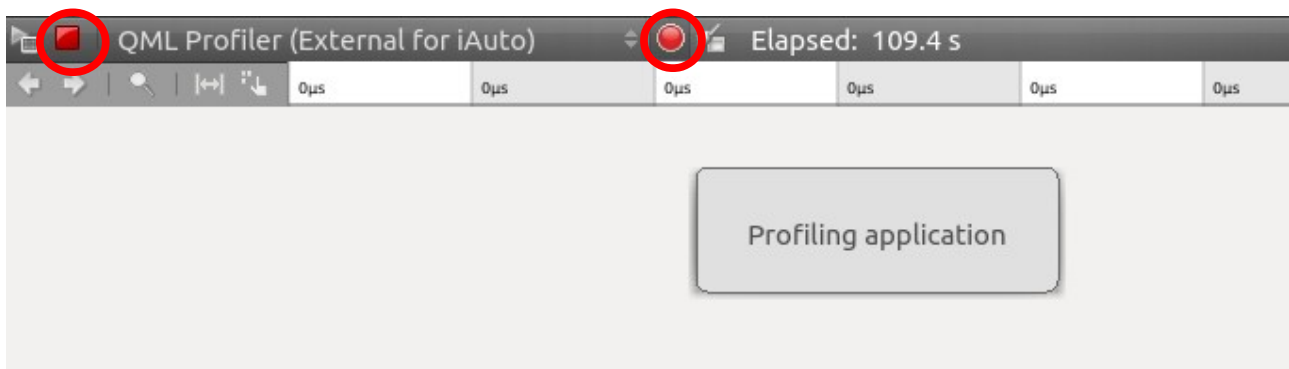
(1) Click “**Analyze**” and select “**QML Profiler (External for iAuto)**” .

Then click “**Start**” button (red circle marked) to begin.



(2) Profiling process.

Click “**Stop**” button (left red circle) to end qml profiler or click “**Disable Profiling**” button (right red circle) to stop collecting data.

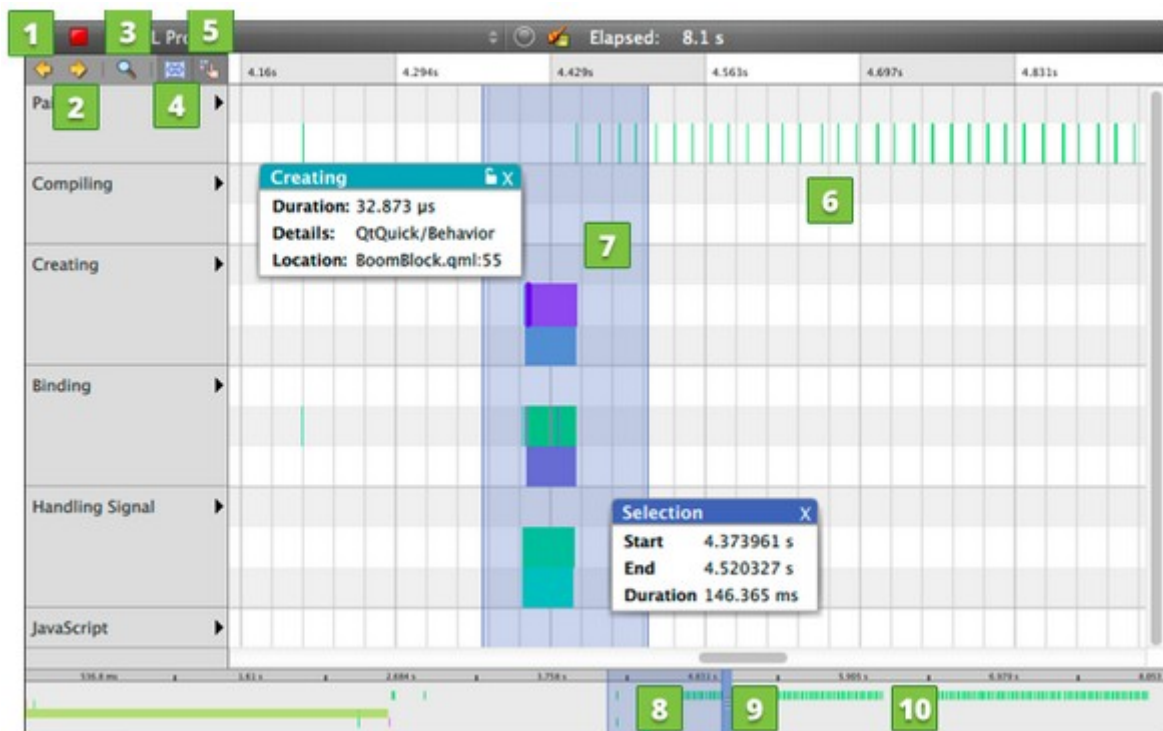
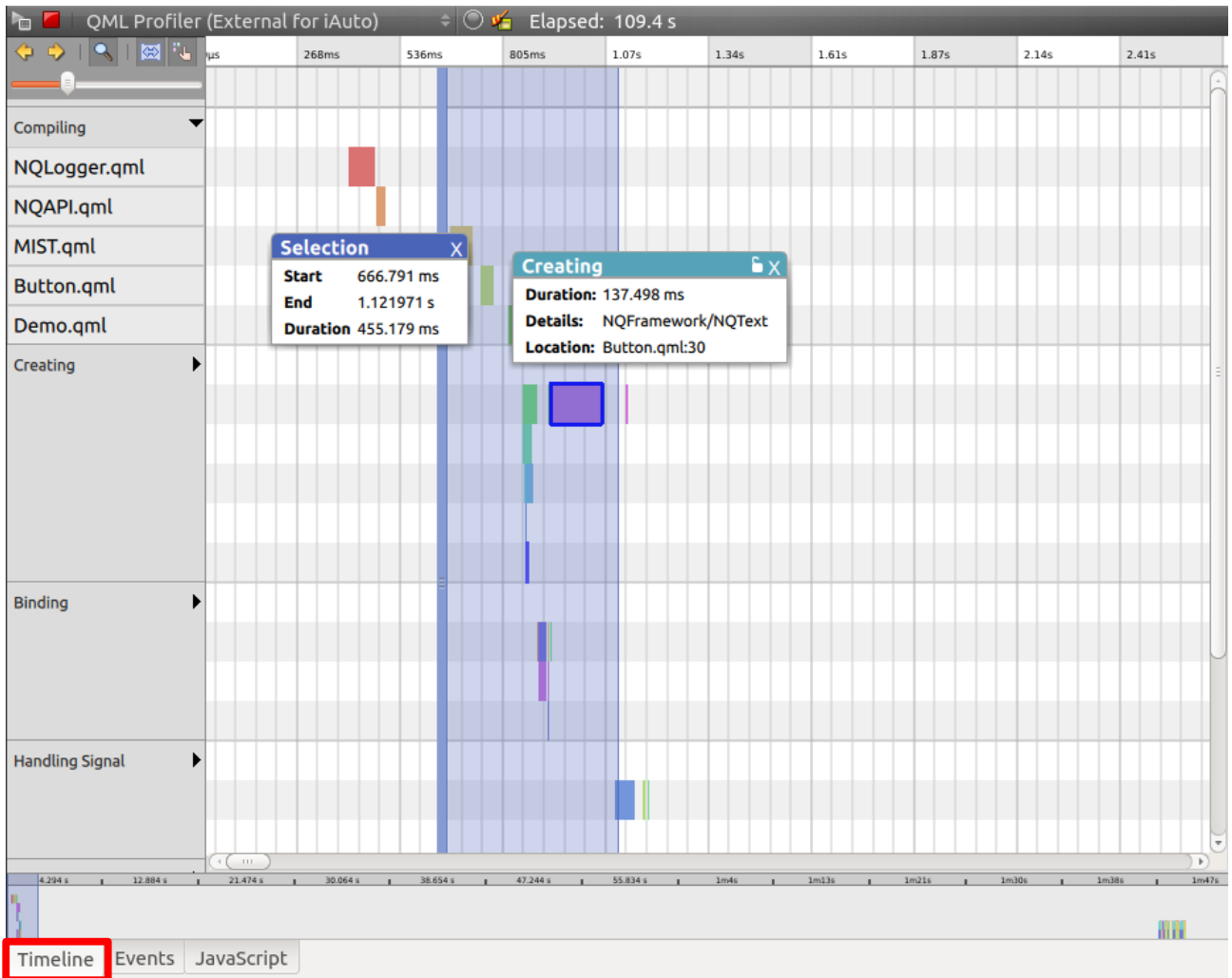


(3) Profiling results.

To save all the collected data (named xxx.qtd), select **Save QML Trace** in the context menu by right mouse clicked. And to view the saved data, select **Load QML Trace**.

Click “**Timeline**” —

The **Timeline** view displays graphical representations of QML and Javascript execution and a condensed view of all recorded events.



- 1) Click **this button** to jump to previous event.
- 2) Click **this button** to jump to next event.
- 3) Click **this button** to show or hide zoom slider that you can use to set the zoom level.
- 4) Click **this button** to active the selection tool. Then click in the timeline to specify the beginning of the event range. Drag the selection handle to define the end of the range. The length of the range indicates the frame rate of the event.
- 5) Click **this button** to enable or disable the view event information on mouseover.
- 6) Each row **here** describes a type of QML events that were recorded. Move the cursor on an event on a row to see how long it takes and where in the source it is being called.
- 7) Select an event range **here** to view the frame rate of events and to compare it with the frame rate of similar events.
- 8) Drag **here** or click the outline to move on the outline.
- 9) Drag **here** to set the zoom level.
- 10) **This outline** summarizes the period for which data was collected.

Click “Events” —

The **Events** view displays the number of times each binding, create, compile, JavaScript, or signal event is triggered and the average time it takes.

QML Profiler (External for iAuto) Elapsed: 109.4 s

Location	Type	Time in Perce	Total Time	Calls	Mean Time	Details
<program>	Binding	100.00 %	541.218 ms	1	541.218 ms	Main Program
MIST.qml:1	Compile	10.94 %	59.231 ms	1	59.231 ms	MIST.qml
NQAPI.qml:1	Compile	4.93 %	26.659 ms	1	26.659 ms	NQAPI.qml
NQAPI.qml:11	Create	0.51 %	2.785 ms	2	1.392 ms	QtQuick/Item
NQAPI.qml:80	JavaScript	0.20 %	1.084 ms	1	1.084 ms	width
NQAPI.qml:91	JavaScript	0.04 %	196.644 µs	1	196.644 µs	height
NQLogger.qml:1	Compile	12.96 %	70.133 ms	1	70.133 ms	NQLogger.qml
NQLogger.qml:11	Create	0.33 %	1.771 ms	2	885.548 µs	QtQuick/Item
NQLogger.qml:79	JavaScript	5.82 %	31.478 ms	3	10.493 ms	debug
sprintf.js:2	JavaScript	0.30 %	1.610 ms	1	1.610 ms	anonymous function
sprintf.js:15	JavaScript	4.21 %	22.766 ms	3	7.589 ms	sprintf
sprintf.js:23	JavaScript	0.15 %	825.803 µs	3	275.267 µs	anonymous function
sprintf.js:107	JavaScript	3.98 %	21.530 ms	3	7.177 ms	anonymous function
sprintf.js:164	JavaScript	0.08 %	435.909 µs	3	145.303 µs	get_type
Button.js:1	JavaScript	3.74 %	20.225 ms	15	1.348 ms	onClicked
Button.js:4	JavaScript	0.36 %	1.948 ms	15	129.835 µs	onReleased
Button.js:7	JavaScript	0.40 %	2.177 ms	15	145.117 µs	onPressed
Button.qml:1	Compile	6.40 %	34.615 ms	1	34.615 ms	Button.qml
Button.qml:5	Create	0.67 %	3.648 ms	2	1.824 ms	NQFramework/NQRectangle

Caller	Type	Total Time ^	Calls	Caller De	Callee	Type	Total Time ^	Calls	Callee C
Demo.qml:37	Create	3.637 ms	1	Butto...	Button.qml:44	Create	818.474 µs	1	NQFr...
<program>	Binding	11.071 µs	1	Main ...	Button.qml:23	Create	715.145 µs	1	NQFr...
					Button.qml:30	Create	623.946 µs	1	NQFr...

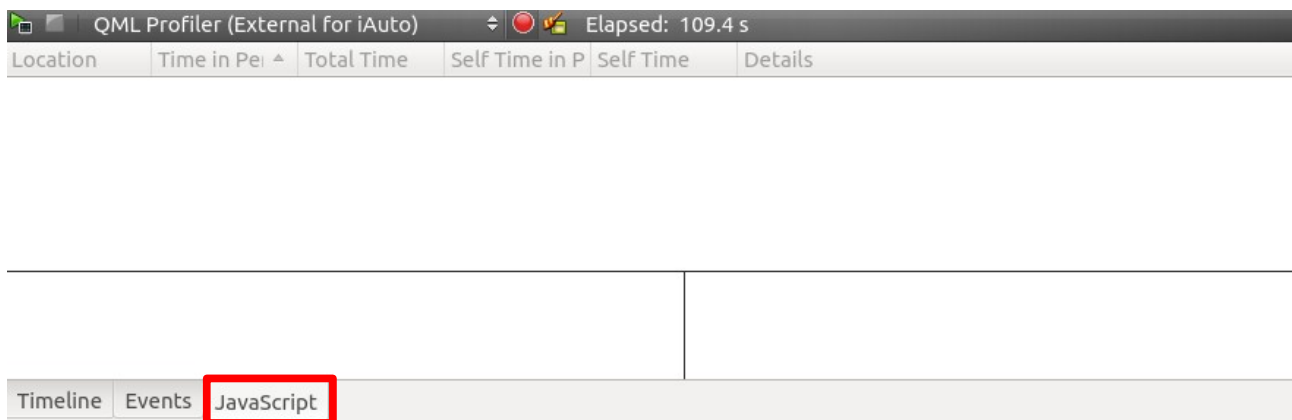
Timeline **Events** JavaScript

Note:

- a. Click on an event will move to the source code in the code editor.
- b. To view the median, longest, and shortest time for the occurrences, select **Extended Event Statistics** in the context menu by right mouse clicked.
- c. To show or hide javascript events and qml events, select **Show JavaScript Events** and **Show QML Events** in the context menu by right mouse clicked.
- d. The **Callers** and **Callees** panes show dependencies between events.
- e. To view an event range in the **Events** view, select **Limit Events Pane to Current Range** in the context menu by right mouse clicked in the Timeline view.
- f. To copy the contents of one view or row to the clipboard, select **Copy Table** or **Copy Row** in the context menu by right mouse clicked.

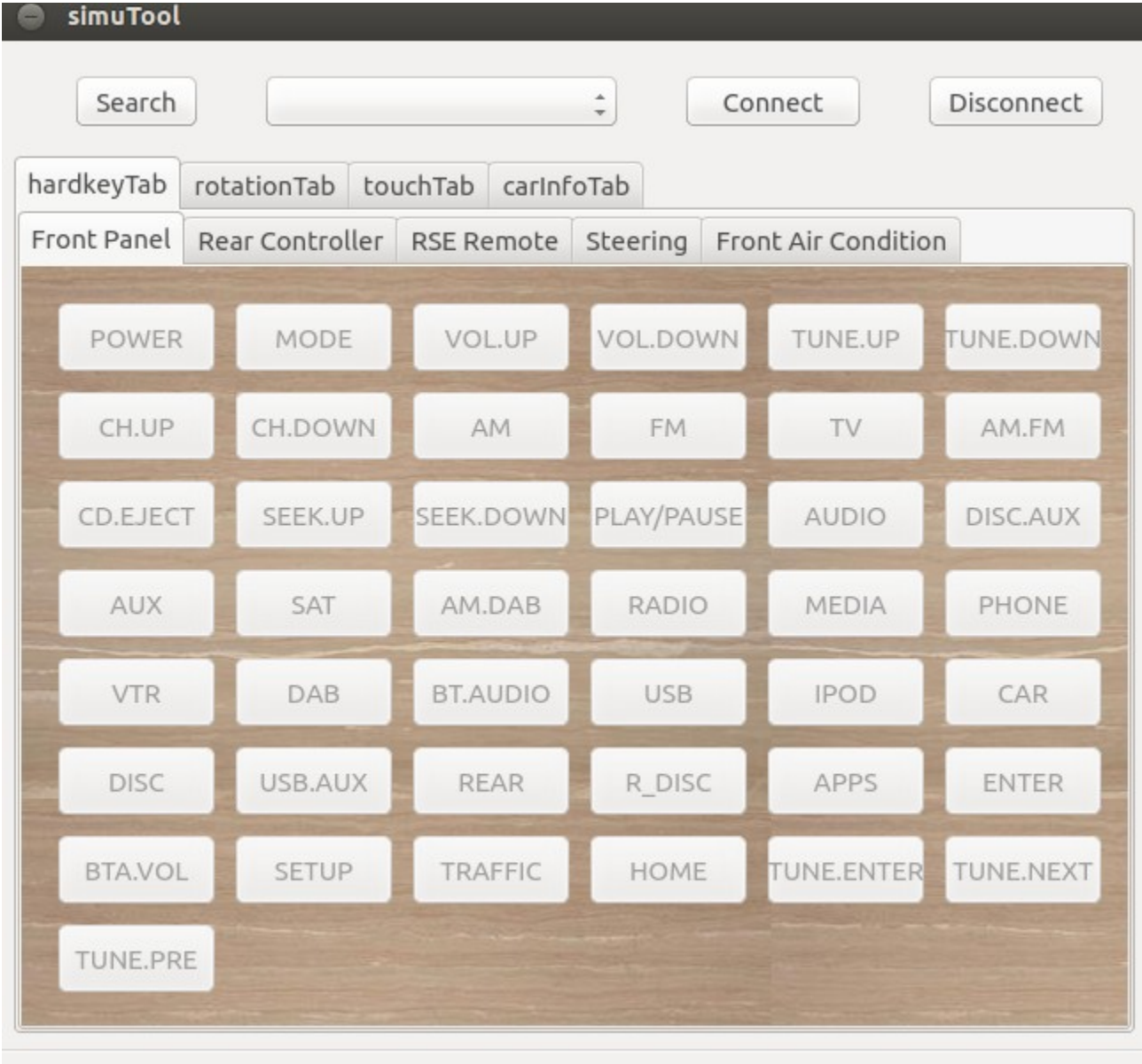
Click “JavaScript” —

JavaScript events are shown in the **Events** view only for applications that use Qt Quick 2 and are compiled with Qt 5.3 or later. For applications that use Qt Quick 2 and are built with Qt 5.0 or 5.1, you can view information about JavaScript events in the separate **JavaScript** view.

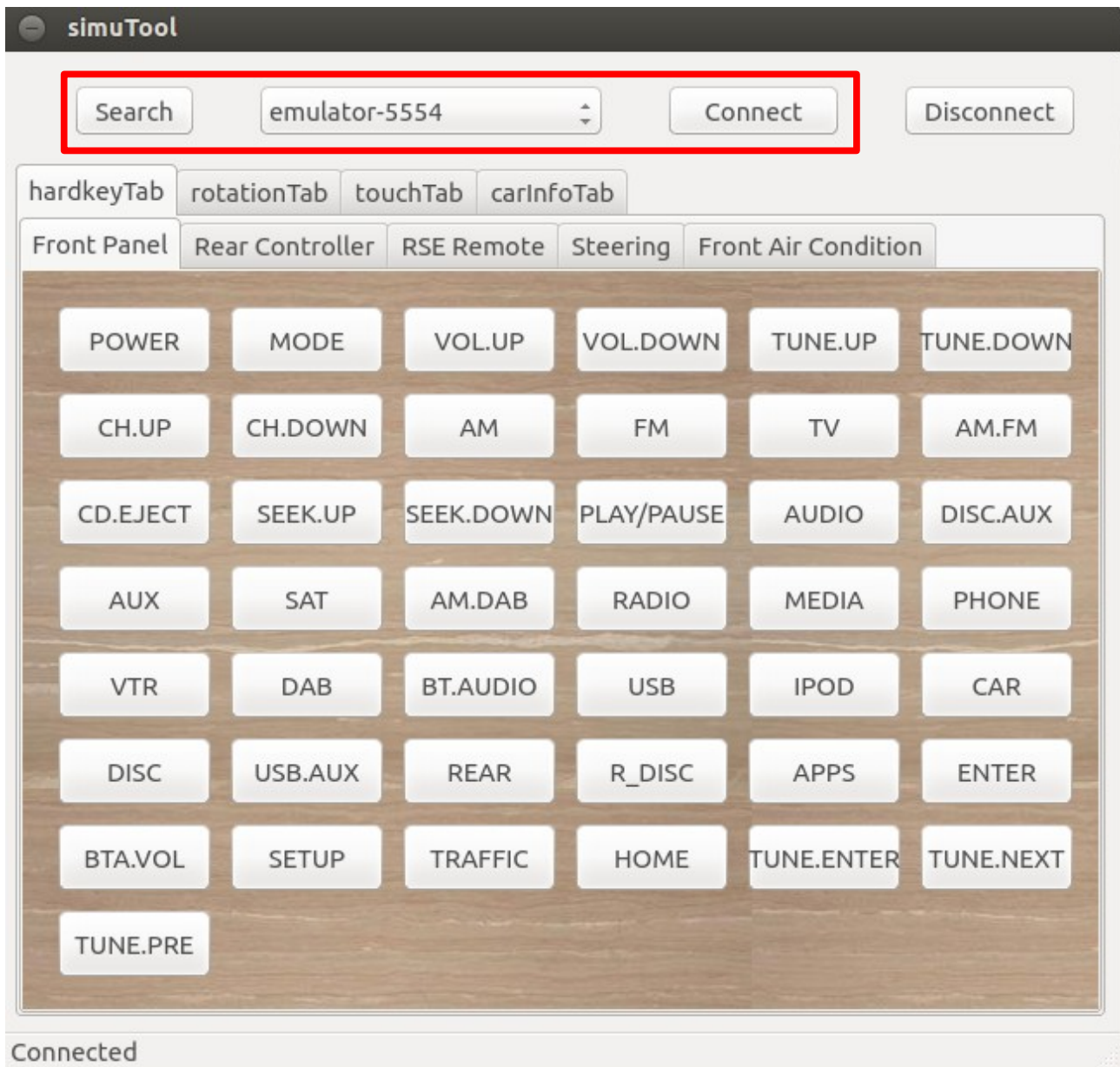


VII. SimuTool

1、SimuTool panel



2、 Search available device and **connect**.



3、 Switch tabs and click buttons.

4、 Reference:

<http://iwiki.storm/Project/17Model/iAutoPlatform/17CY/developguide/17CY-simutool-manual>

VIII. Update Record

Date	Details
...	...
2015-04-01	available
2015-05-06	Part III: Add certificate config for cpk.
	Part IV: Add logcat options.