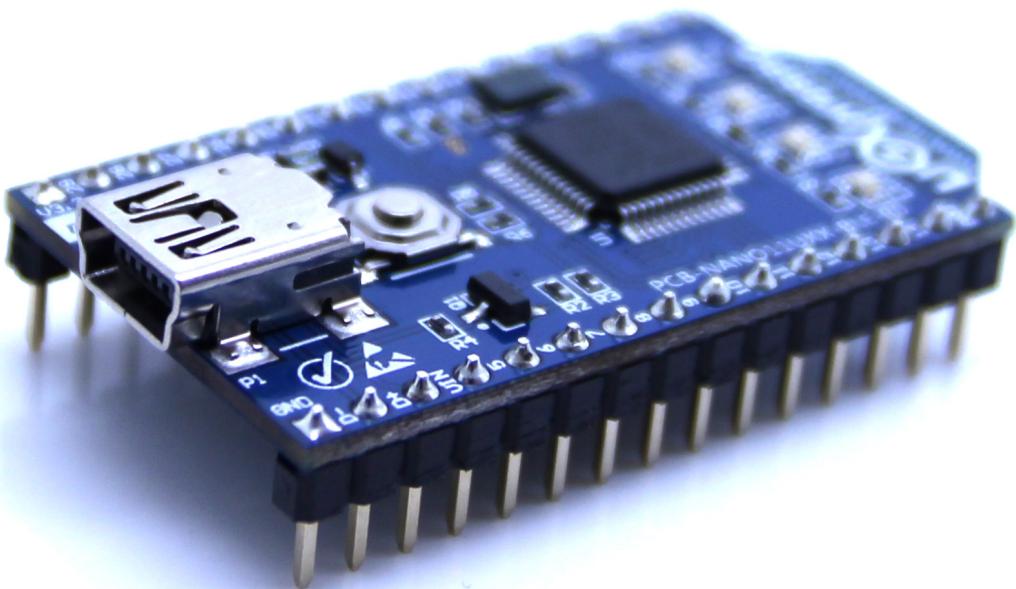


Getting Started with uCXpresso.BLE



2014/4/1

v1.0.2

History:

2014/3/16 Add Overview

2014/3/12 Add “Create A New Project”, “Serial Terminal & Debug”

2014/3/11 First Edition

Overview

First of all, an espresso is necessary,



The uCXpresso.BLE framework have to work on the LPCXpresso IDE, you need to install the LPCXpresso IDE in your system first. The LPCXpresso IDE are available for Windows, Linux and Mac OS/X, also you can download the LPCXpresso IDE from www.lpcware.com with a free license after register. About LPCXpresso IDE, please refer to
<http://www.lpcware.com/lpcxpresso/download>

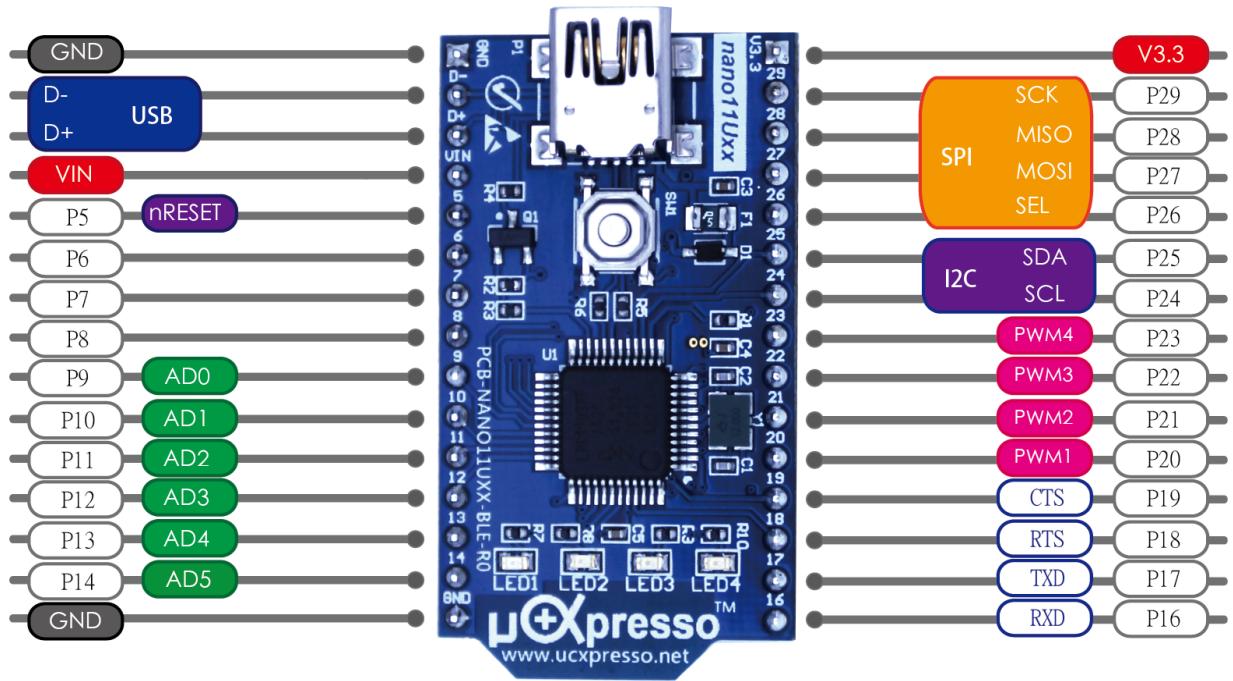
The uCXpresso.BLE is a RTOS C++ framework that port for the ARM Cortex-M0 MCU and BLE (Bluetooth Low Energy), and provides many features such as Peripherals (ADC, PWM, I2C, SPI, PIO, Interrupt, Timer....), BLE Services (Serial, Proximity, Heart Rate Meter...) and Multi-Thread (RTOS Thread, Semaphore, Mutex, MailBox...).

Easy to use with C++ framework:

Ex. A LED on the pin 19 of nano11U37

```
CPin led(P19);           // connect led object to P19 pin
led.output();             // set P19 as an output pin
led = HIGH;               // set P19 to HIGH (turn on LED)
```

Multiple Pin Definition:



Ex. Set P20 as an input pin

```
CPin key(P20);           // connect key object to P20
key.input();              // set key as an input pin with internal Pull-Up.
if ( key==LOW ) {        // check key PIN LEVEL
    ...
}
```

Ex. Set P20 as a PWM output

```
CPwm::period(0.02);      // set global PWM period time to 20ms
CPwm servo(PWM1);        // connect servo object to PWM1 (P20)
servo.enable();            // enable the servo PWM to output
servo.dutyCycle(0.8);     // set servo duty-cycle to 80%
servo = 0.005;            // set servo PWM period to 5ms
```

Multi-Thread

The uCXpresso.BLE framework is base on an advanced Real-Time operating system (RTOS) that called the FreeRTOS, and using the multi thread and class library will simplify your design, coding and maintenance.

Final,

Let uCXpresso C++ Framework to shorten your developed time, improves the qualities and enjoy in working.

1. Upgrade Your LPCXpresso IDE

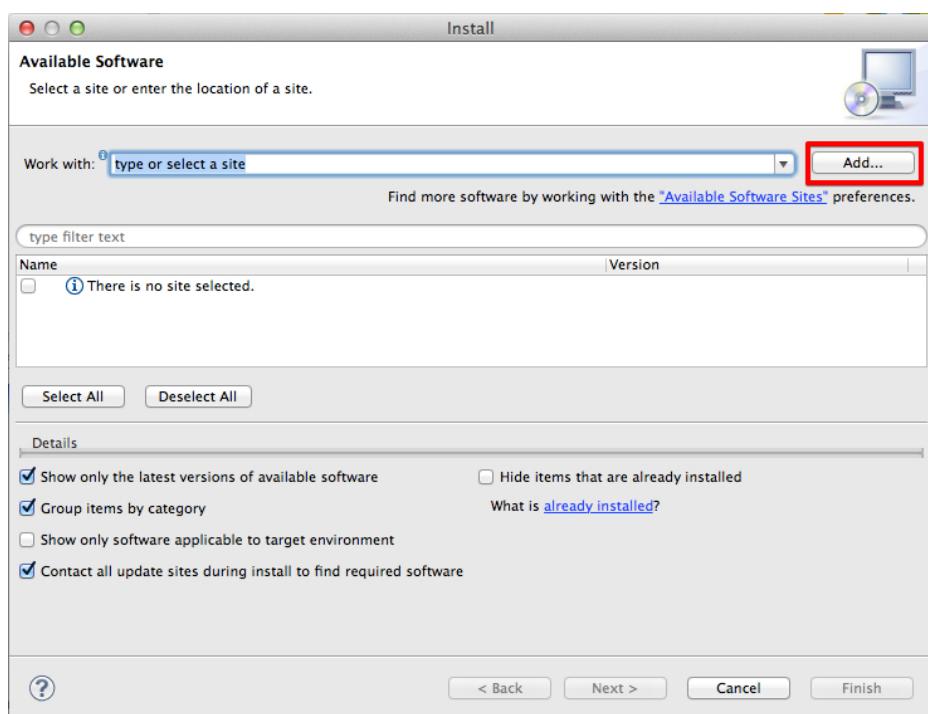
1.1. Increase VM memory for Windows System

- a. Close LPCXpresso if executing.
- b. Open c:/nxp/LPCXpresso_x.x/lpcxpresso/lpcxpresso.ini
- c. Modify MaxPermSize to 512

1.2 Install new software (Eclipse Plug-in)

LPCXpresso Main Menu > Help > Install New Software...

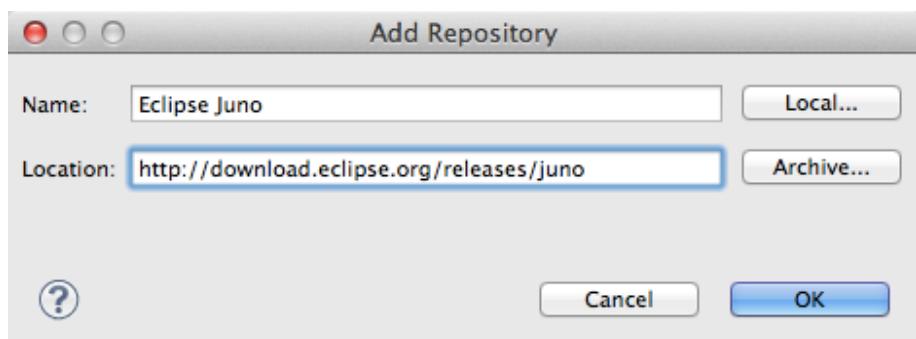
Click [Add]



1.3 Install Eclipse Plug-In

Name: Eclipse Juno

Location: <http://download.eclipse.org/releases/juno>



In General Purpose Tools section, select “Marketplace Client”

Name	Version
ACTF Visualization Extension for PDT Feature	1.0.2.R201302130546
ACTF Visualization Extension for WST Feature	1.0.2.R201302130546
ACTF Visualization Feature	1.0.2.R201302130546
ACTF Visualization SDK Feature	1.0.2.R201302130546
ChangeLog Management Tools	2.8.0.201302051708
ChangeLog Management Tools for C/C++	2.8.0.201302051708
ChangeLog Management Tools for Java	2.8.0.201302051708
Dynamic Languages Toolkit - Core Frameworks	4.0.0.201206120848
Dynamic Languages Toolkit - Remote Development Support	4.0.0.201206120848
Eclipse Plug-in Development Environment	3.8.2.v20130116-091538-7c7wfj0FFt6Z...
Local Terminal (Incubation)	0.2.200.201301070737
m2e - Maven Integration for Eclipse	1.3.0.20130129-0926
m2e - slf4j over logback logging (Optional)	1.3.0.20130129-0926
Marketplace Client	1.1.1.I20110907-0947
Memory Analyzer	1.2.1.201211051250

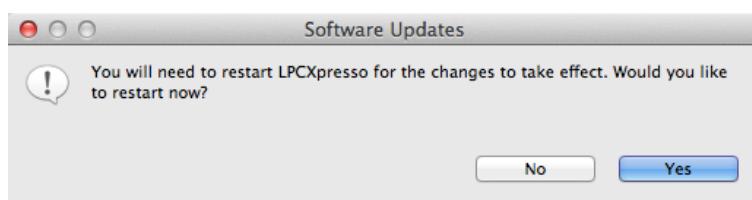
2 items selected

In Mobile and Device Development section, select “Target Management Terminal”

Name	Version
Mobile and Device Development	
C/C++ GCC Cross Compiler Support	1.1.0.201302132326
C/C++ GDB Hardware Debugging	7.0.0.201302132326
C/C++ Memory View Enhancements	2.2.0.201302132326
C/C++ Remote Launch	6.0.0.201302132326
Remote System Explorer End-User Runtime	3.4.1.201302122026
Remote System Explorer User Actions	1.1.400.201301240456
 Target Management Terminal	3.3.2.201301080822
TCF C/C++ Debugger	1.0.1.201212201401
TCF Remote System Explorer add-in	1.0.0.201212201401
TCF Target Explorer	1.0.0.201212201401
Modeling	
Programming Languages	
SOA Development	
Testing	
Web XML, Java EE and OSGi Enterprise Development	

2 items selected

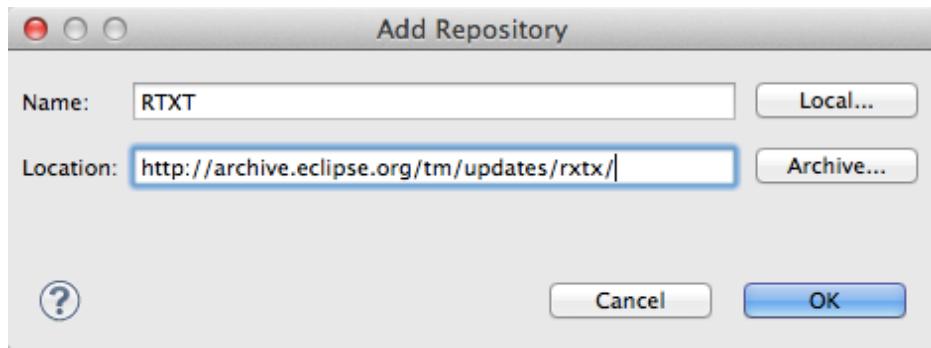
Click [Next] to start the installation. In progress, you need to [Confirm >] the Selected Features, and [accept] the licenses of software, and click [OK] to confirm any security-warning message, click [Yes] to restart and finish the installation if need.



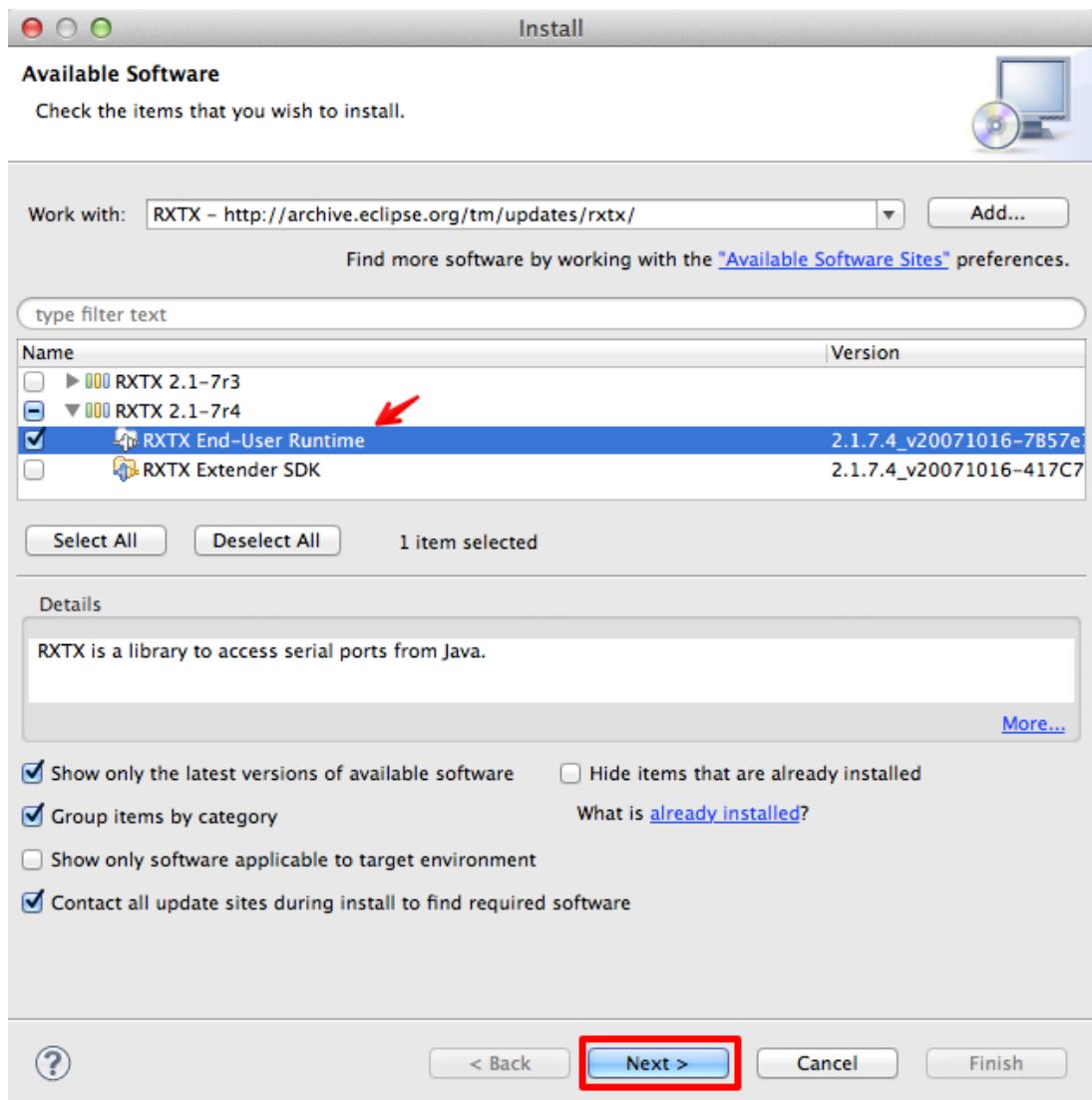
1.4. Install RXTX Plug-in

Name: RXTX

Location: <http://archive.eclipse.org/tm/updates/rxtx/>



Select latest version “RXTX End-User Runtime”



Click [Next] to start the installation.

Remark:

For Mac OS/X user, you have to follow below steps to enable the USB CDC virtual COM. Port:

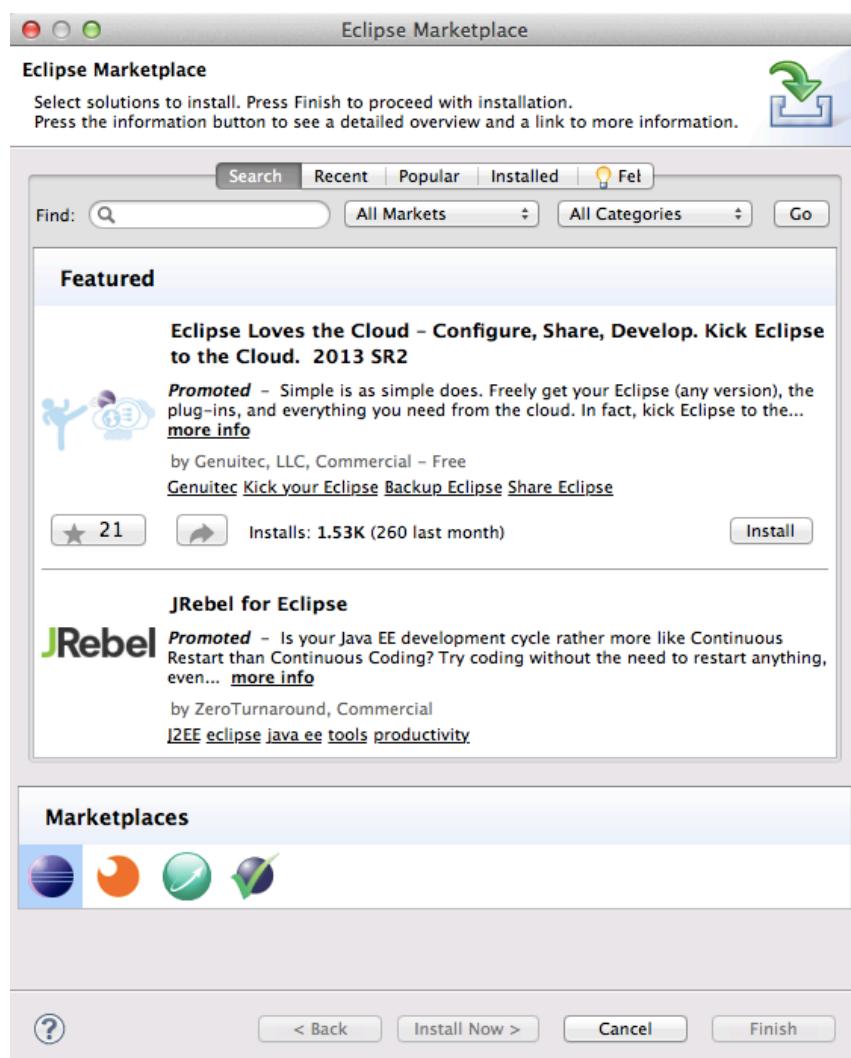
1. Open Terminal App of OS/X
2. Type :
sudo mkdir /var/lock
sudo chmod a+rwx /var/lock

In Windows System, The USB CDC driver can be download from below link:

for NANO11Uxx : http://www.embeda.com.tw/tw/wp-content/uploads/2014/01/nano11Uxx_usb_driver.zip
(unzip and copy INF file to desktop, and USBCOM port driver direct to the INF file.)

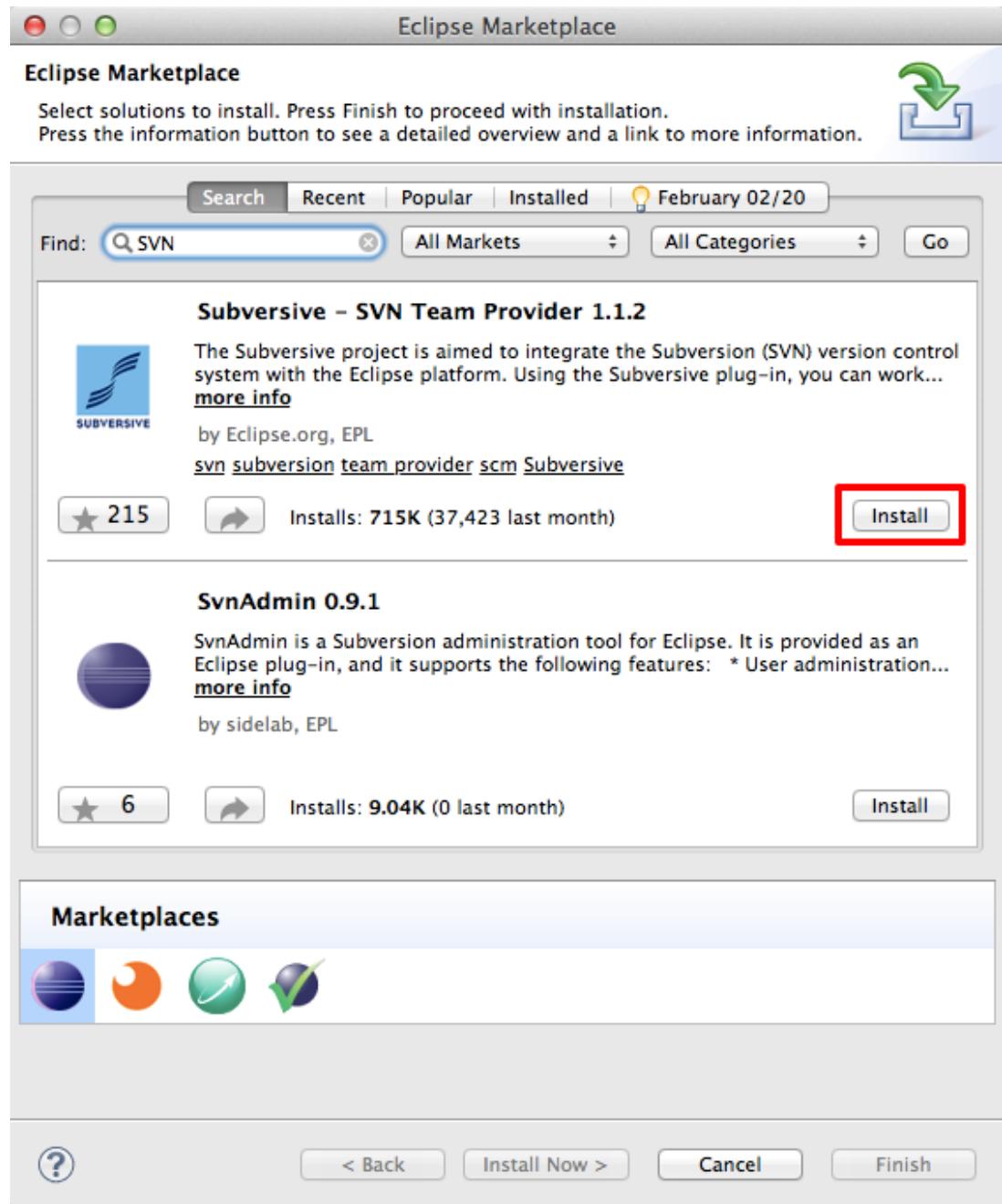
1.5. Install new software from Eclipse Marketplace

Click LPCXpresso Main Menu > Help > Marketplace...

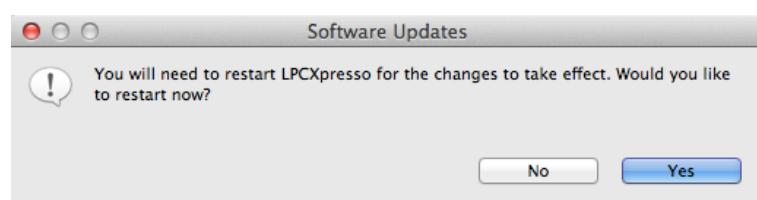


1.6 Install Subversion – SVN Team Provider (Plug-in)

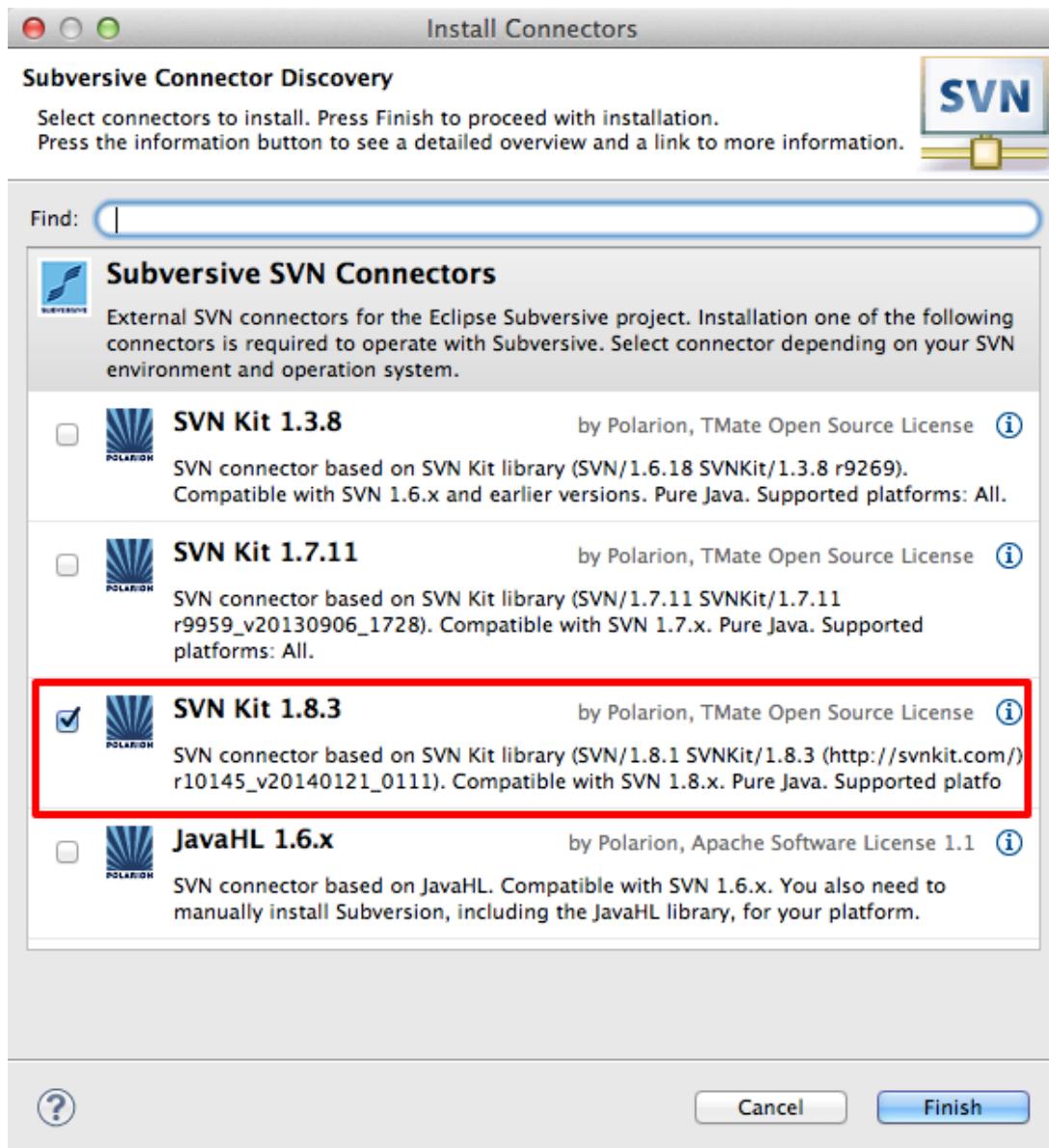
In Find, input “SVN” and click [Go]



Click [Install] to start the installation. In progress, you need to [Confirm >] the Selected Features, and [accept] the licenses of software, and click [OK] to confirm any security-warning message. Click [Yes] to restart and finish the installation if need.



After restart, you may need to install the SVN connector kit:

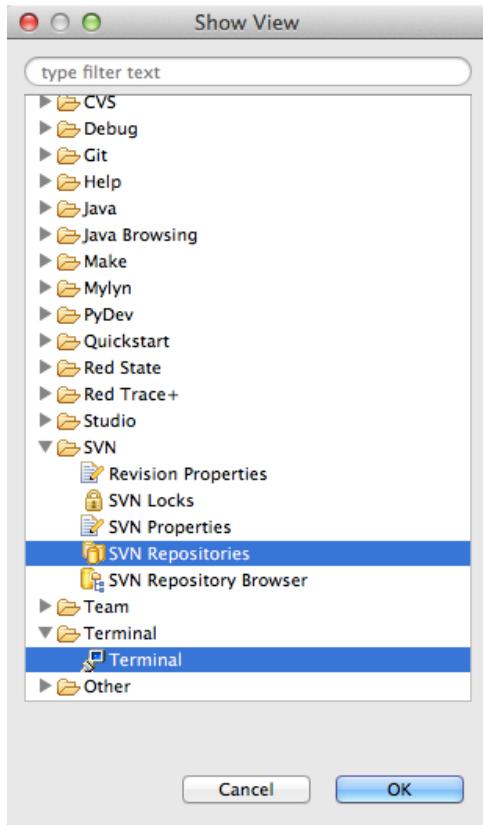


Select the latest version “SVN Kit x.x.x”, and click [Finish] to start the installation. Confirm and click [Next], and [accept] the licenses of software, and click [OK] to confirm any security-warning message. Click [Yes] to restart and finish the installation if need.



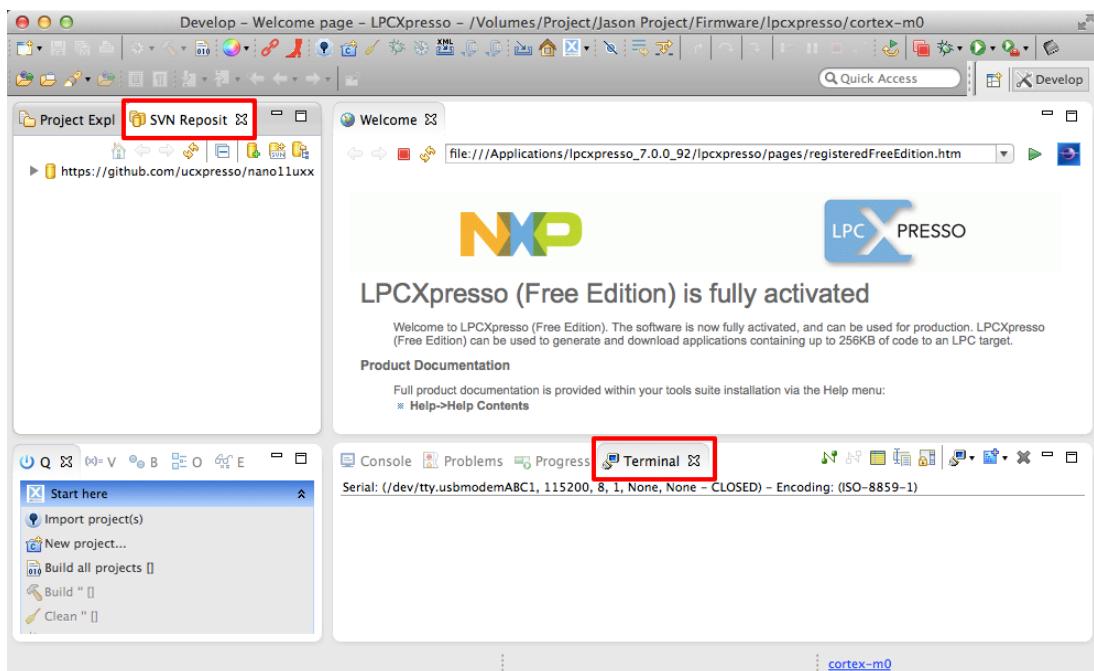
1-7. Show the SVN and Terminal Views

Click LPCXpresso Main Menu > Window > Show View > Other...



Select the “SVN Repositories” and “Terminal”, click [OK]

Move the “SVN Repositories” Tab to “Project Explorer” right side (optional)

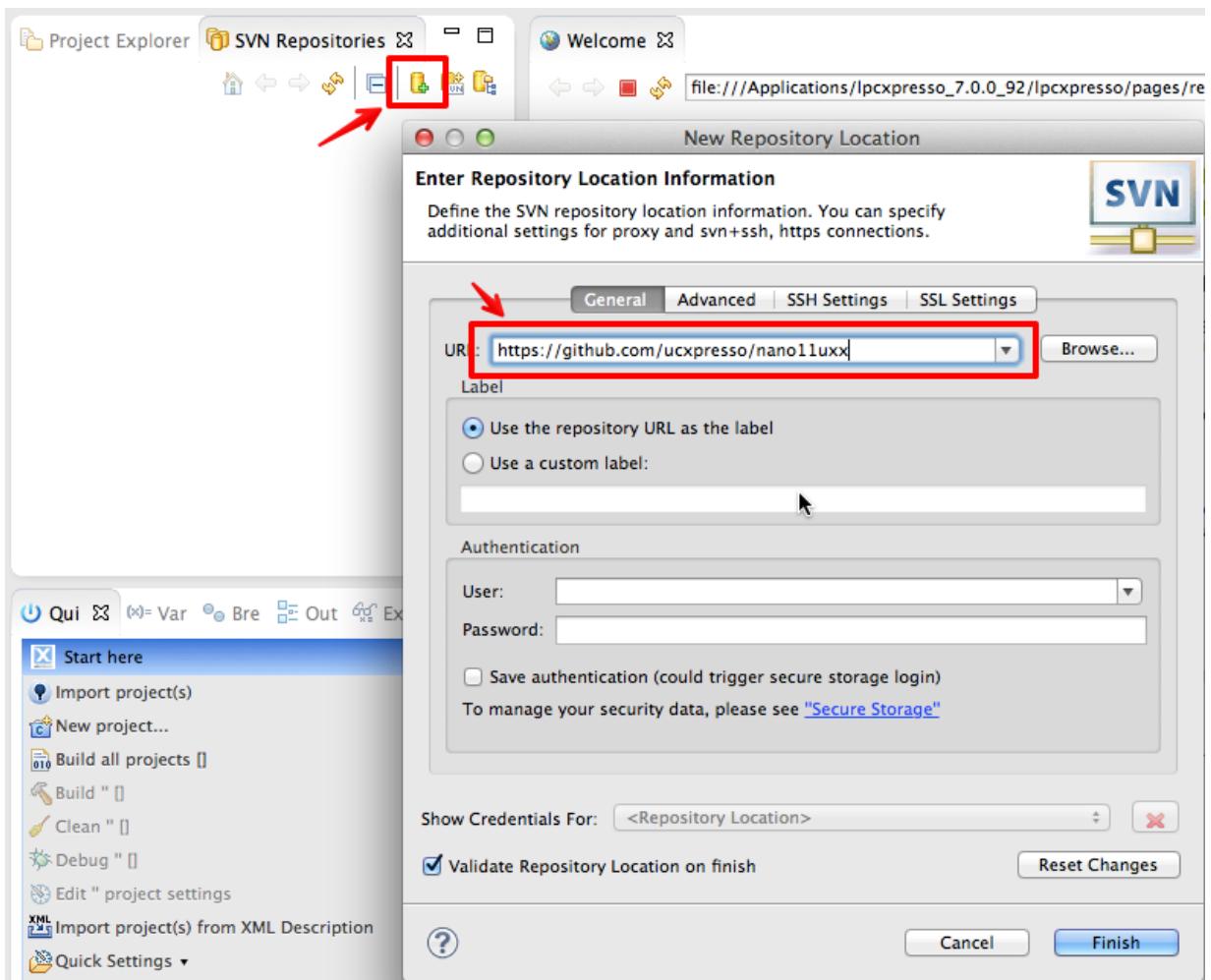


2. Install uCXpresso.BLE Framework

2.1 Using GitHub Repositories



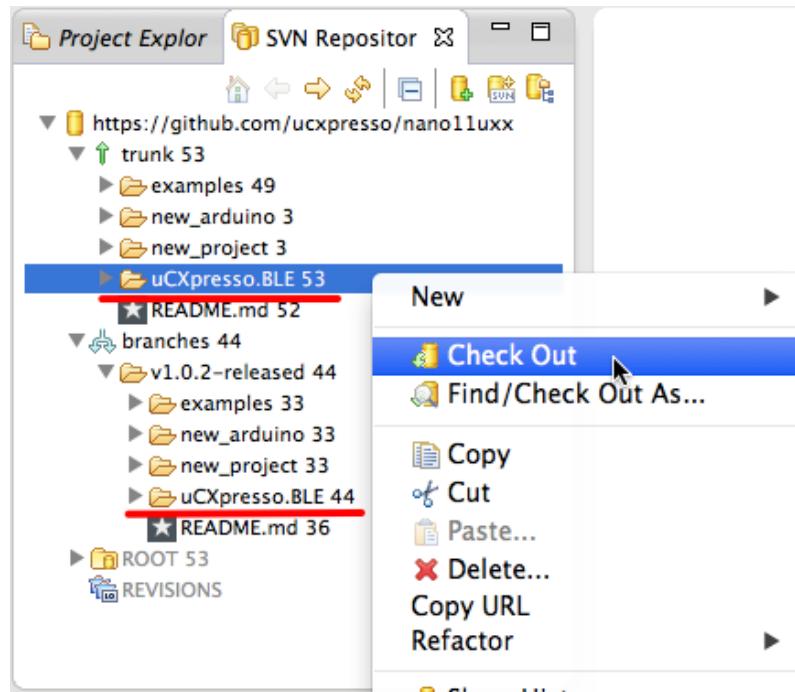
In “SVN Repositories” Tab, Click “New Repository Location”,



and URL Input: `https://github.com/ucxpresso/nano11uxx`

2.2 Check Out uCXpresso.BLE framework

Expand the uCXpresso.BLE SVN repository

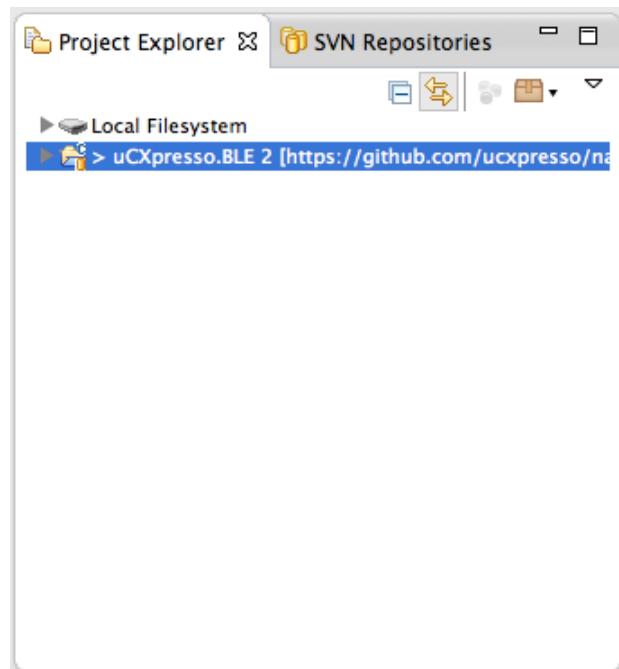


trunk: uCXpresso.BLE RC (Release Candidate) version

branches: All released version

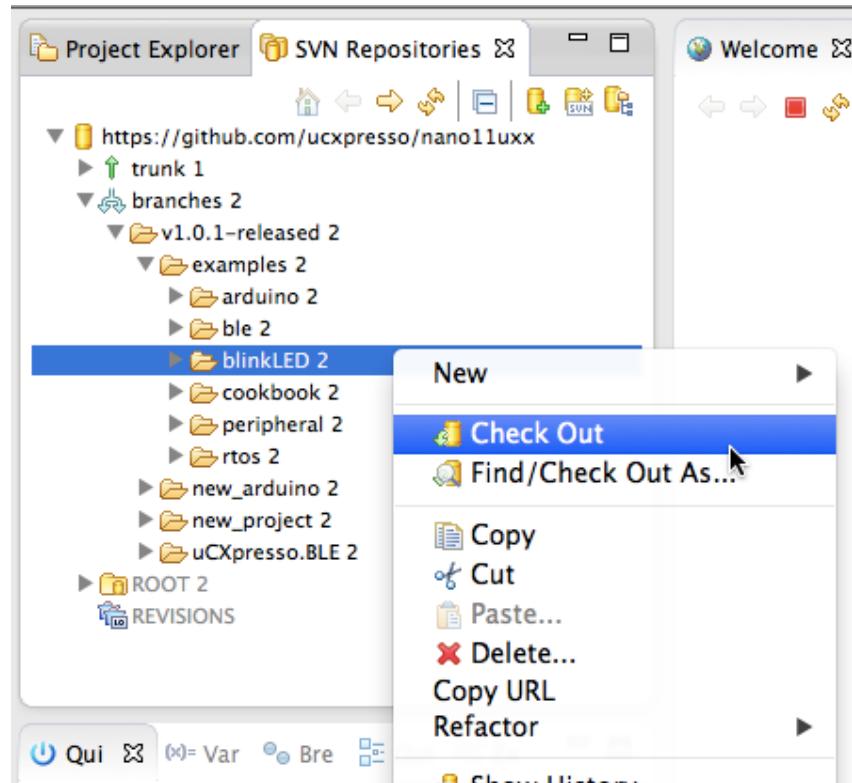
Select “uCXpresso.BLE” folder, and click right button of mouse, and click “Check Out” in drop down menu.

The uCXpresso.BLE framework show in your workspace after check out.



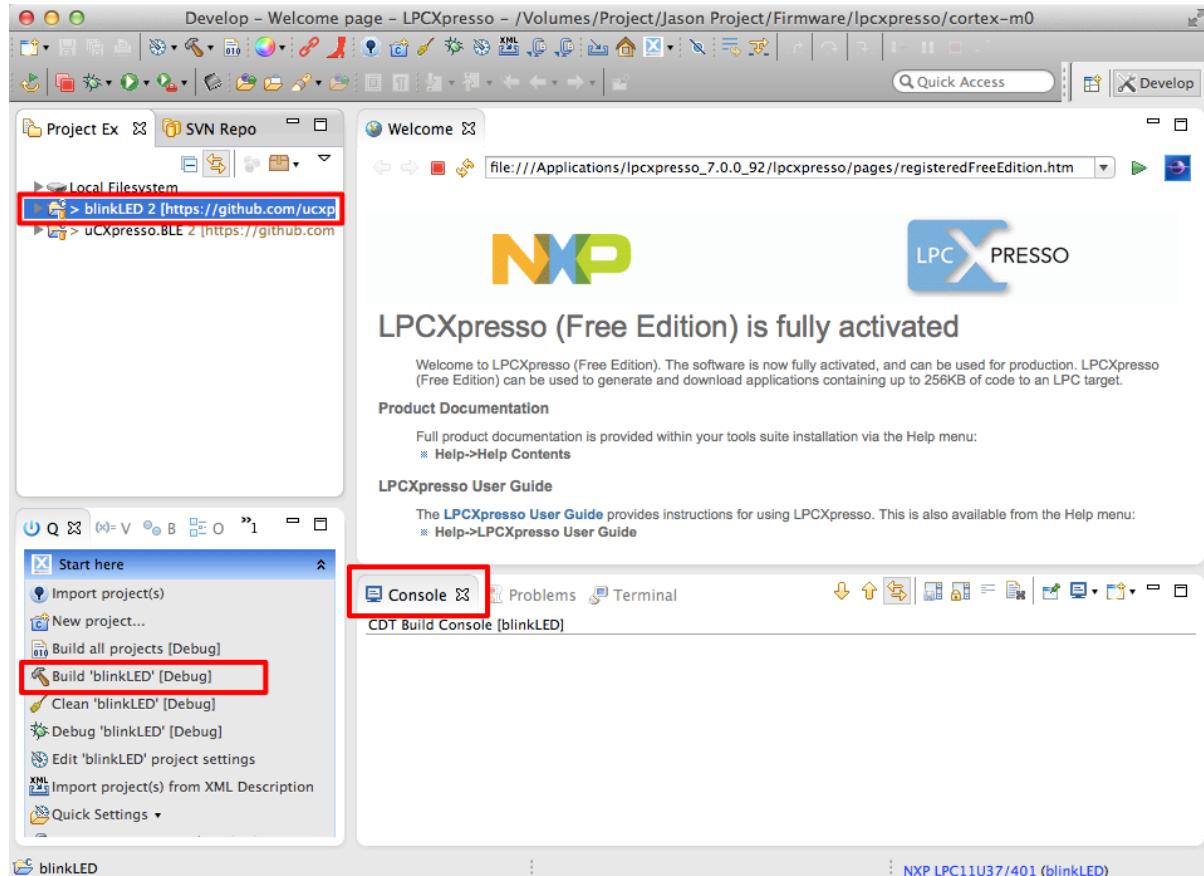
2.3 Check Out example “blinkLED”

In SVN Repositories Tab, select the “blinkLED” folder in “/examples”, then Check Out it.



2.4 Build the “blinkLED” example project

1. Click “blinkLED” Project on Project Explorer View.
2. Click “Console” View
3. Click “Build ‘blinkLED’ [Debug] on Quick Start View.



In Console View, the compiler will create a “blinkLED.bin” image file.

```
make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size blinkLED.axf; arm-none-eabi-objcopy -O binary blinkLED.axf blinkLED.bin ;
    text      data      bss      dec      hex filename
    25272      20     7268   32560    7F30 blinkLED.axf
Created checksum 0xeffffd15 at offset 0x1c in file blinkLED.bin
12:00:54 Build Finished (took 363ms)
```

Note:

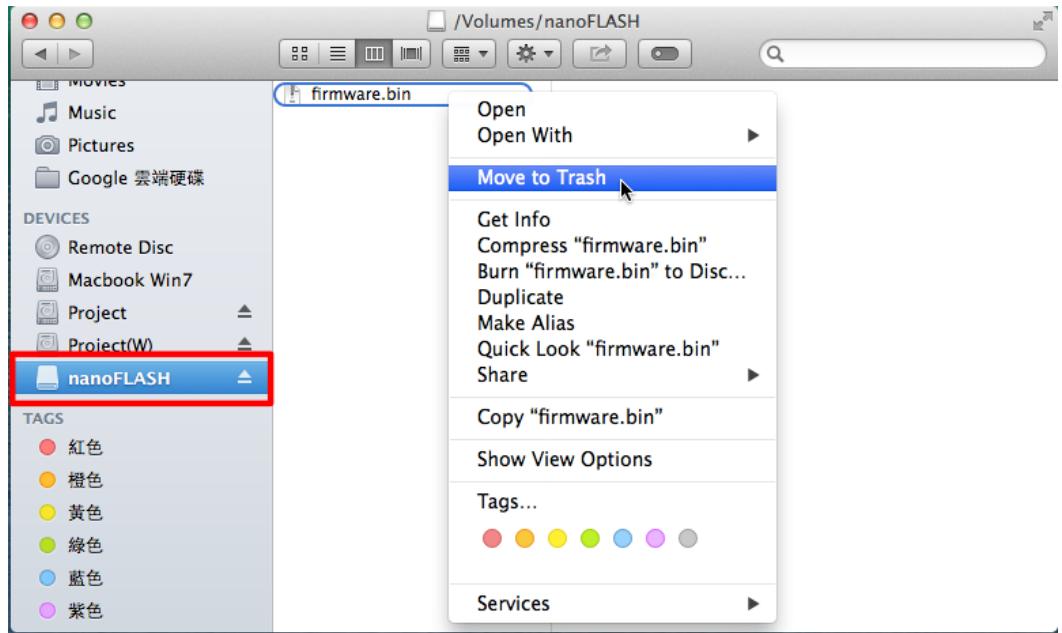
code size = text + data = 25292 bytes

ram size = data + bss = 7288 bytes

3. Download Image File

3.1 Erase Flash Code Memory

First, you need to erase the flash memory for new binary image. Click the “ISP” button on the nano11U37-BLE. In Finder (or File Manager) will creates an USB Disk and called “nanoFLASH”



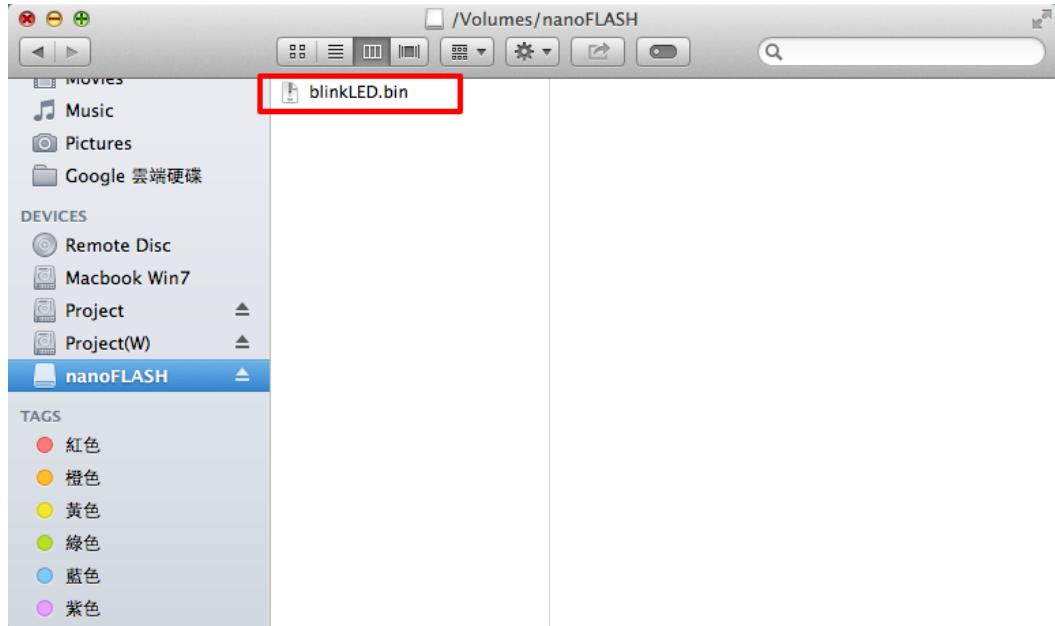
Remove the “firmware.bin” and move to trash.

Note:

In ISP (bootloader) mode: The LED1, LED2, LED3 and LED4 are blink together.

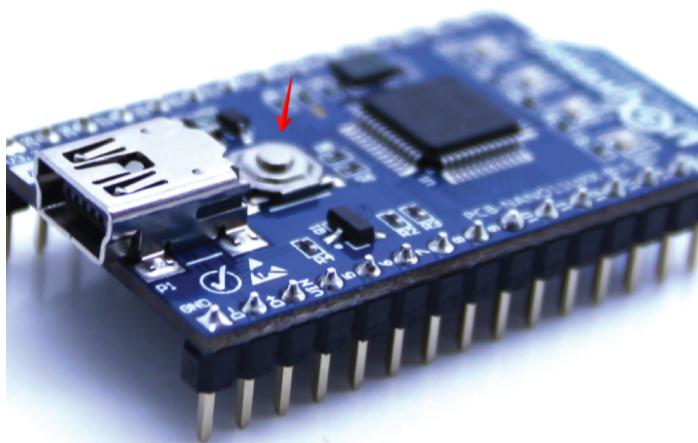
3.2 Program Flash Code Memory

Copy the “blinkLED.bin” from “blinkLED\Debug” folder to nanoFLASH



Eject the “nanoFLASH” disk and click the “ISP” button of NANO11U37-BLE again to execute your binary code. (In App Mode)

Note:

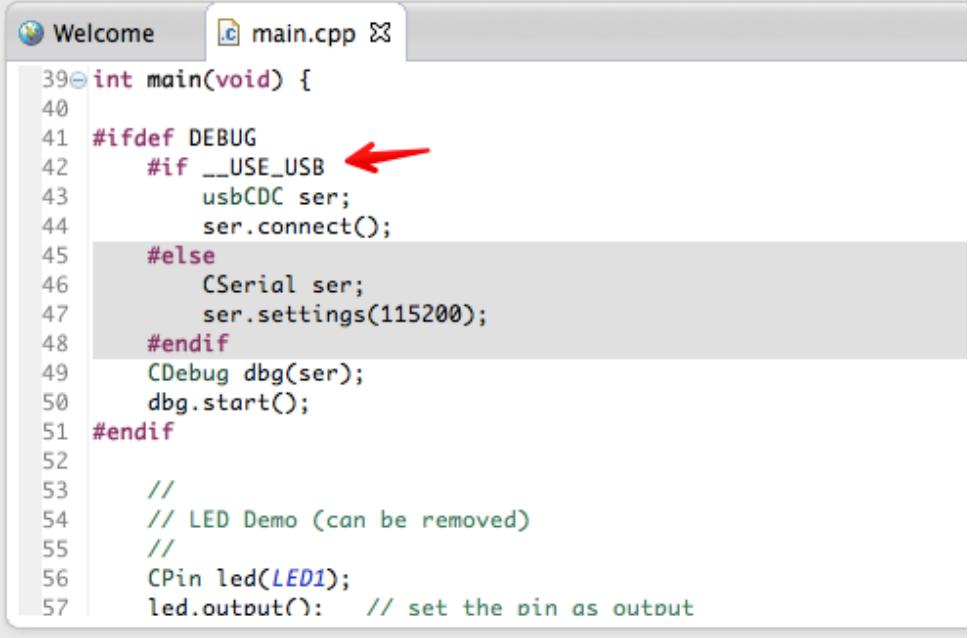


The “ISP” button.

4. Serial Terminal & Debug

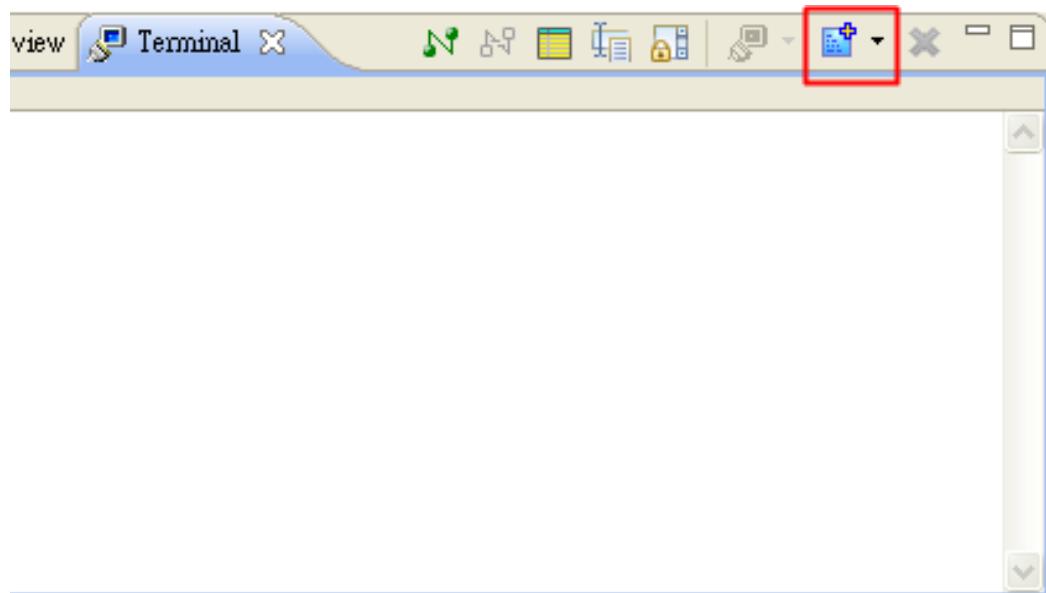
4.1 Serial Terminal (Use USB CDC Virtual COM. Port)

In main.cpp of blinkLED, the usbCDC will be enabled.



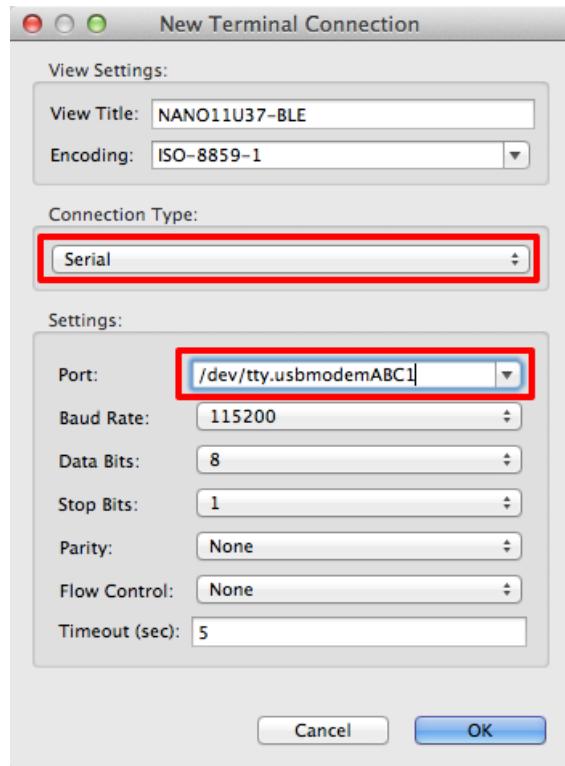
```
39 int main(void) {
40
41 #ifdef DEBUG
42 #if __USE_USB ←
43     usbCDC ser;
44     ser.connect();
45 #else
46     CSerial ser;
47     ser.settings(115200);
48 #endif
49     CDebug dbg(ser);
50     dbg.start();
51 #endif
52
53 // ...
54 // LED Demo (can be removed)
55 // ...
56 CPin led(LED1);
57 led.setOutput(); // set the pin as output
```

In Terminal View, click “New Terminal Connection” to add a new connection.



In Terminal Settings:

1. View Title: NANO11U37-BLE
2. Connection Type: Serial
3. Port : /dev/tty.usbmodemABC1 (for Mac OS/X)



The blinkLED serial shell will run in the Terminal View

```
Serial: (/dev/tty.usbmodemABC1, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
*****
*      Welcome to uCXpresso.BLE      *
*      http://www.ucpresso.net        *
* (C)2012-2014 Embeda International Inc.* 
*****
ver      : get kernel version
task     : get tasks list
heap     : get heap available size
clear    : clear terminal screen
debug    : enter to debug mode
help or ? to show the command list.

uCpresso:/>
```

```

Serial: (/dev/tty.usbmodemABC1, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
*****
*      Welcome to uCpresso.BLE      *
*      http://www.ucpresso.net       *
* (C)2012-2014 Embeda International Inc.*
*****
ver   : get kernel version
task  : get tasks list
heap   : get heap available size
clear  : clear terminal screen
debug  : enter to debug mode
help or ? to show the command list.

uCpresso:/>ver
uCpresso.BLE V1.0.1 (released)
FreeRTOS: V8.0.0
USBoot : 0.0.2

uCpresso:/>heap
Heap available size: 3544 bytes

uCpresso:/>task
Name      Stat.  Prio.  Remain  Num.
debug     R      0      32      4
IDLE      R      0      25      2
main      B      0      126     1
usbCDC   B      4      22      3

uCpresso:/>

```

Command:

ver, to show the kernel & module version.

heap, to check the available size of heap memory.

task, to check all of tasks in the system.

debug, to enter to debug mode to view the DBG(...) message from user's program.

4.2 Debug

Add 'key' object in main.cpp of blinkLED project.

```

55     //  

56     CPin led(LED1);  

57     led.output(); // set the pin as output  

58  

59     //  

60     // your setup code here  

61     //  

62     CPin key(P18);  

63     key.input();  

64  

65     while(1) {  

66         //  

67         // LED Demo (can be removed)  

68         //  

69         led = !led;  

70         sleep(500);  

71  

72         //  

73         // your loop code here  

74         //  

75         if (key==LOW) {  

76             dbg.printf("Key down!!\n");  

77         }  

78     }  

79     return 0 ;  

80 }
81

```

And add the key check in while loop.

Build Debug and download blinkLED.bin to nanoFLASH, then execute it.

In Terminal View, input 'debug' command to enter the debug mode:

```

Serial: (/dev/tty.usbmodemABC1, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)
*****
*      Welcome to uCxpresso.BLE      *
*      http://www.ucxpresso.net       *
* (C)2012-2014 Embeda International Inc.*  

*****  

ver   : get kernel version  

task  : get tasks list  

heap  : get heap available size  

clear : clear terminal screen  

debug : enter to debug mode  

help or ? to show the command list.  

uCxpresso:/>debug  

press [ESC] to exit the debug mode  

Key down!! ←
Key down!!

```

Try to push the key (pin 18) to LOW, and you can see the debug message to show in the terminal view.

4.3 Made an invalidly debug message code.

Made an invalidly debug message code in [Release Build]

```

3 Name :..
17
18 #include "uCpresso.h"
19 #include "class/serial.h"
20 #include "class/usb_cdc.h"
21 #include "debug.h"
22
23 #ifdef DEBUG
24 #define DBG dbg_printf
25 #else
26 #define DBG(...)
27 #endif
28
29
30 /**
31 // TODO: insert other include files here
32 //
33
34 /**
35 // TODO: insert other definitions and declarations here
36 //
37
38 /**
39 // LED Demo (can be removed)
40 //
41 #include "class/pin.h"
42

```

Using the `#ifdef DEBUG` to block and define the `DBG` to `dbg_printf`, and `#else` to block and define the `DBG(...)` to nothing.

Rename `dbg.printf(...)` to `DBG(...)` in your program

```

62   //
63   CPin led(LED1);
64   led.output(); // set the pin as output
65
66   //
67   // your setup code here
68   //
69   CPin key(P18);
70   key.input();
71
72   while(1) {
73     //
74     // LED Demo (can be removed)
75     //
76     led = !led;
77     sleep(500);
78
79     //
80     // your loop code here
81     //
82     if (key==LOW) {
83       DBG("Key down!!\n");
84     }
85   }
86   return 0 ;
87
88

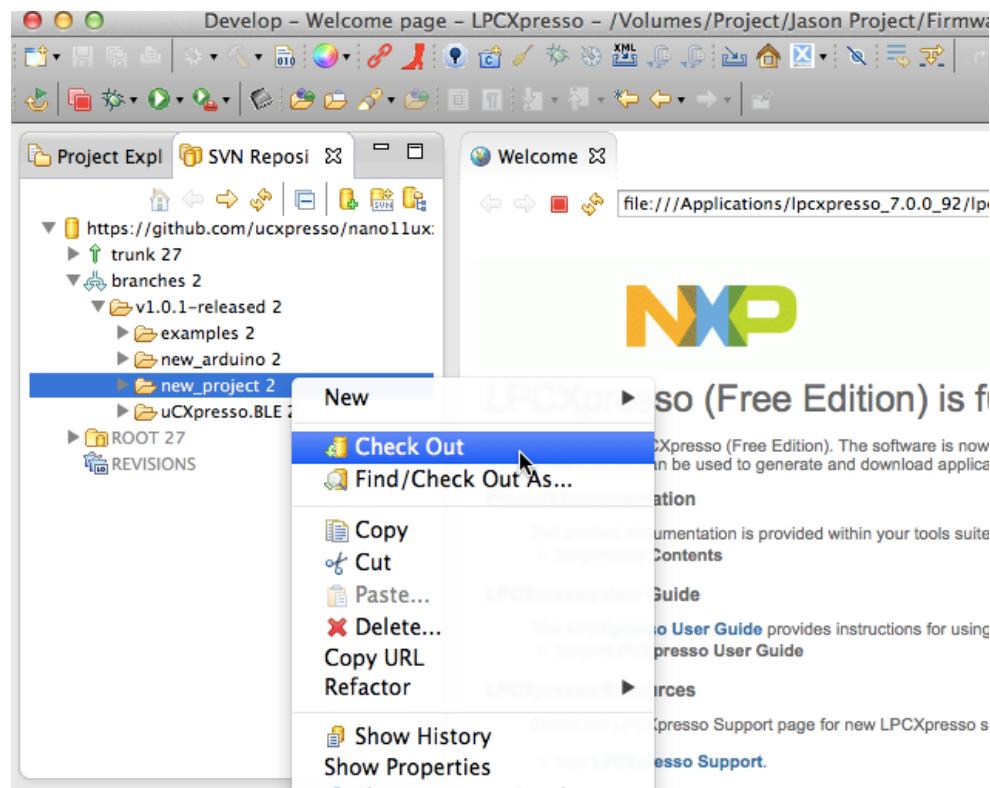
```

Now, The `DBG(...)` message code will be create in the [Debug Build] only, not in the [Release Build].

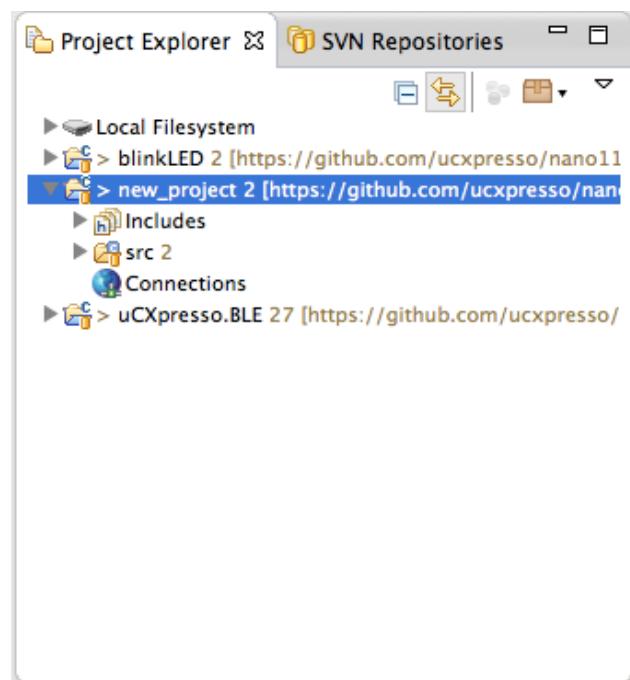
5. Create A New Project

5.1 Check Out the “new_project” template

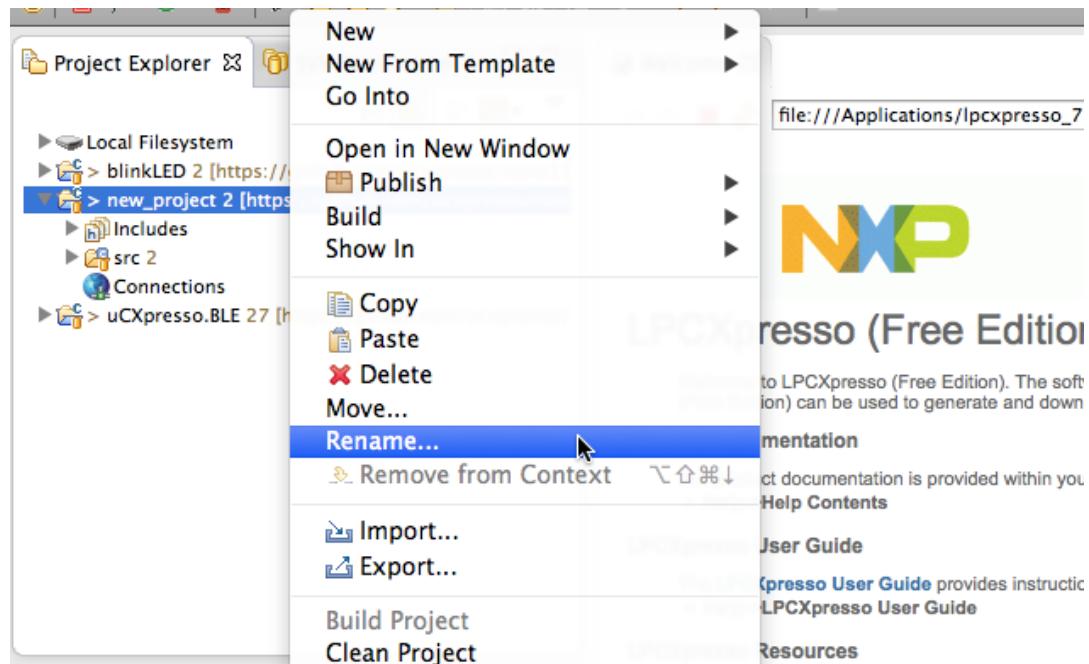
In SVN Repositories View, expand the three and “Check Out” the “new_project” to workspace.



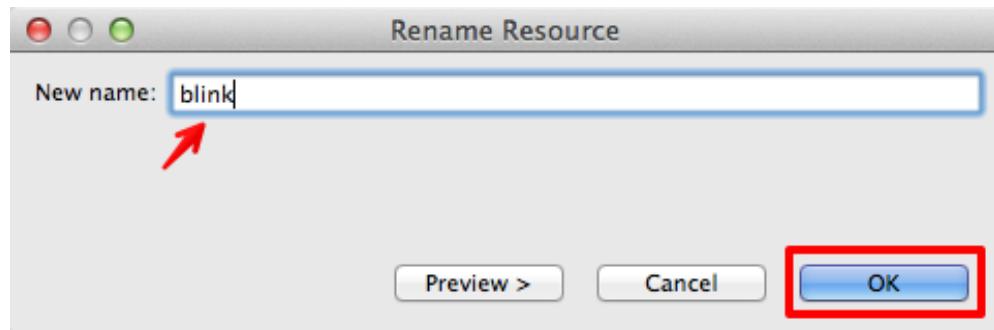
The “new_project” will show in workspace after Check Out.



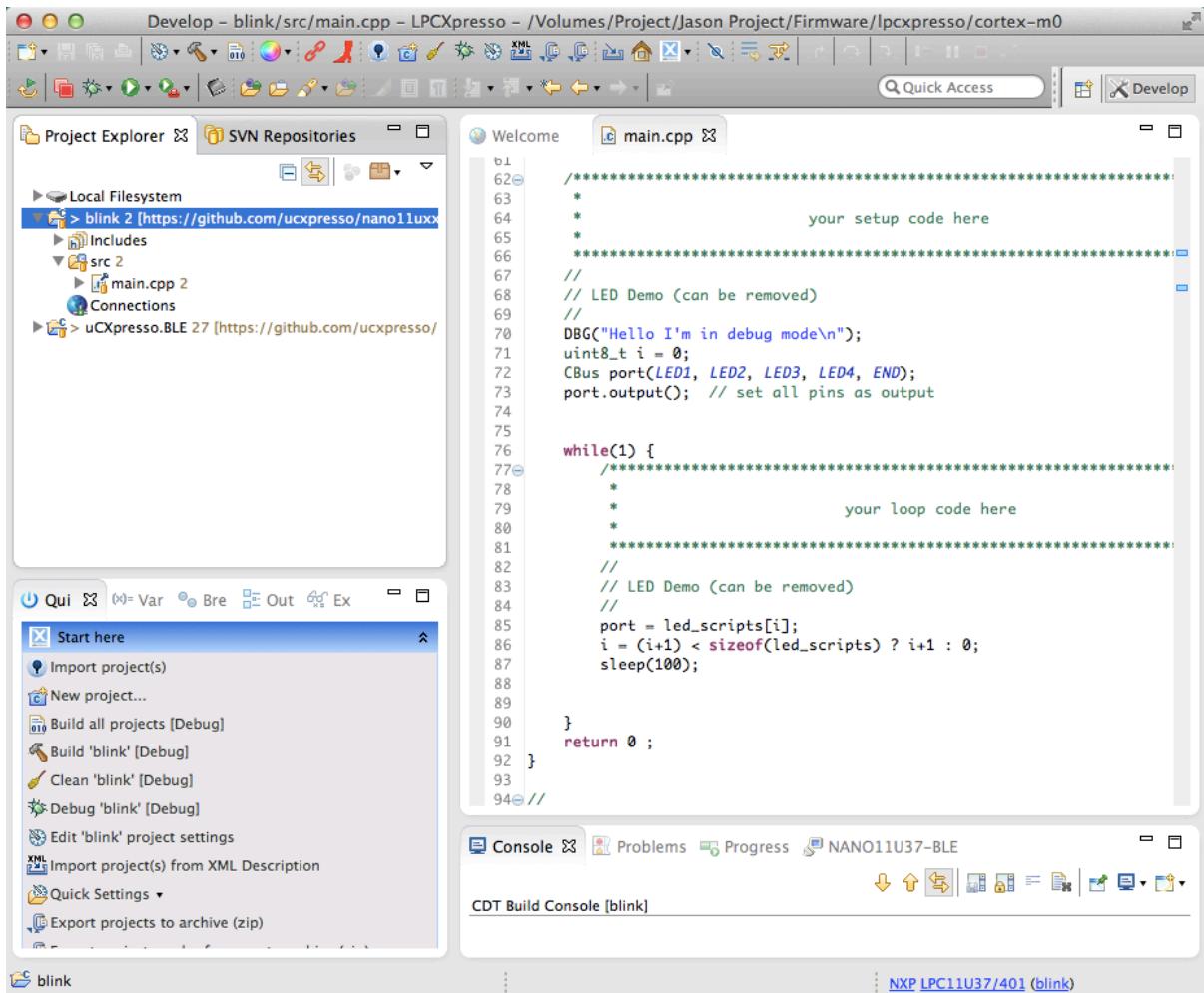
Rename the “new_project” to new project name what you want.



For example to rename to “blink” as below,



An empty project “blink” is in the workspace.



Expand the ‘src’ folder of new ‘blink’ project, and start to program your main.cpp.

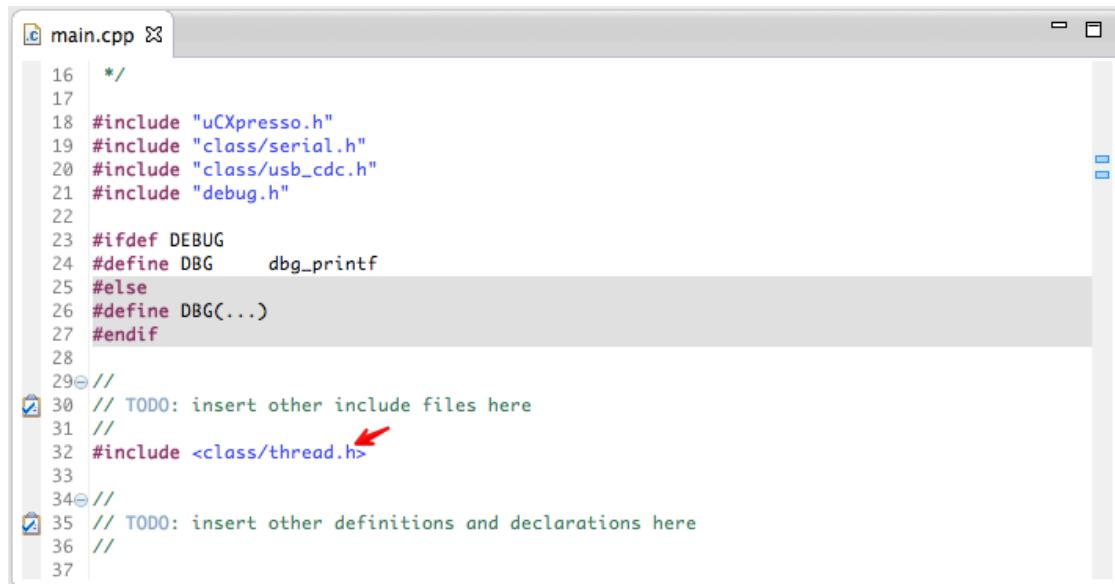
6. RTOS Thread

6.1 Overview

This is an example of how to make four threads to control the LED1, LED2, LED3 and LED4 respectively.

6.2 Make your thread class

Includes the CThread class to provide the thread features.

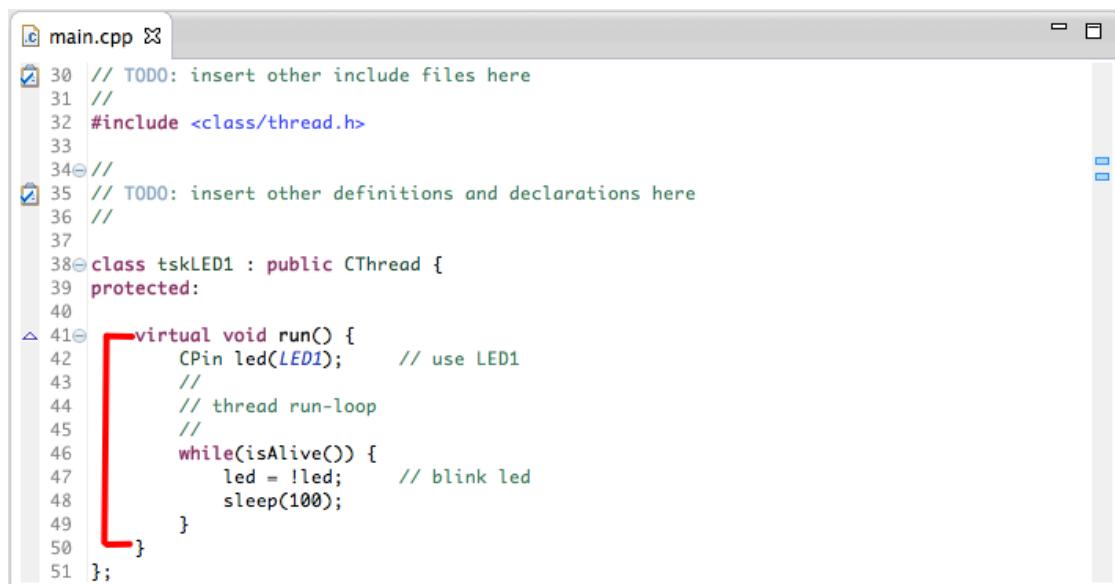


```

16 */
17
18 #include "uCpresso.h"
19 #include "class/serial.h"
20 #include "class/usb_cdc.h"
21 #include "debug.h"
22
23 #ifdef DEBUG
24 #define DBG      dbg_printf
25 #else
26 #define DBG(...)
27 #endif
28
29 //
30 // TODO: insert other include files here
31 //
32 #include <class/thread.h> ←
33
34 //
35 // TODO: insert other definitions and declarations here
36 //
37

```

Creates a “tskLED1” thread and inherits from CThread class.



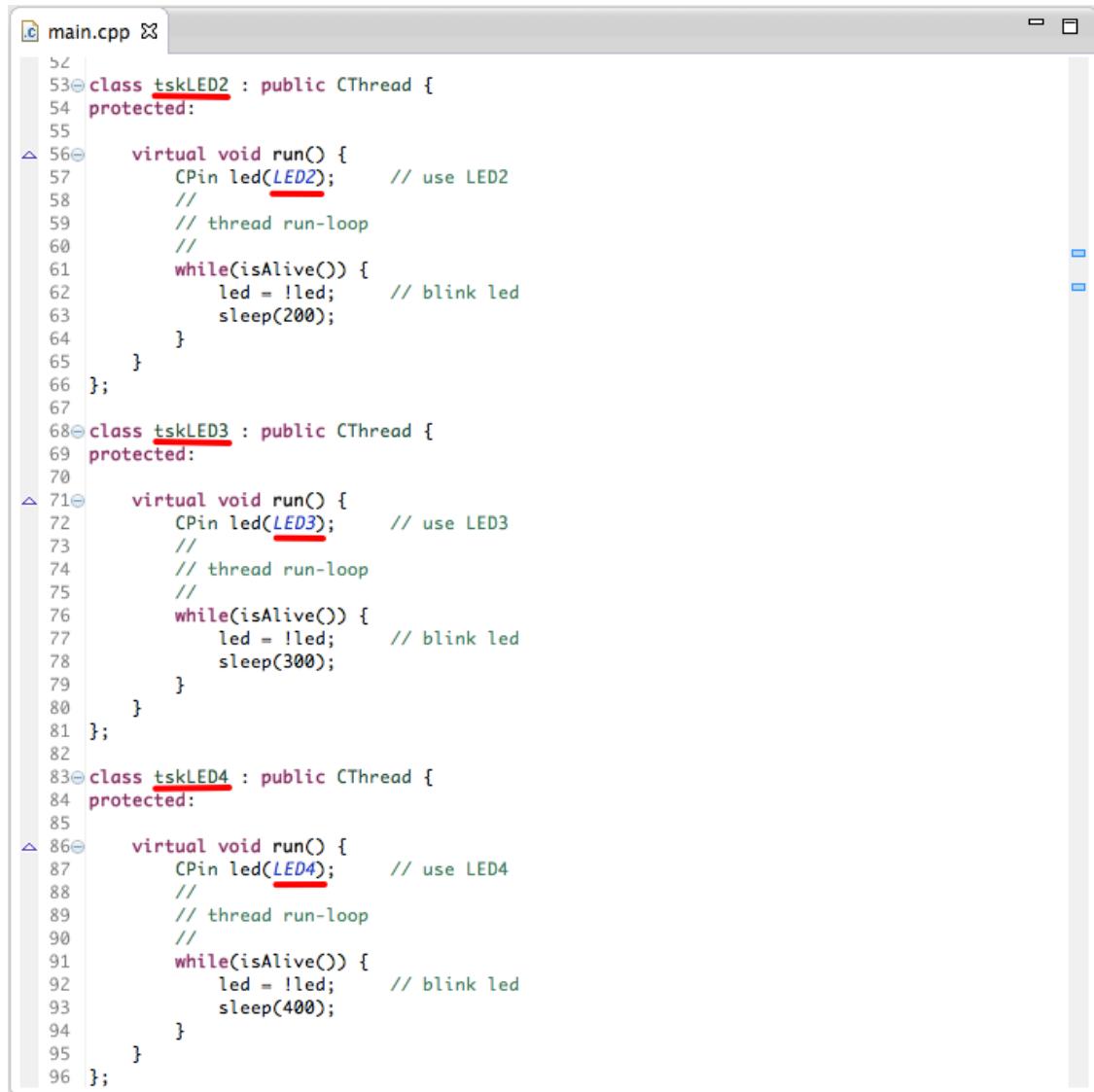
```

30 // TODO: insert other include files here
31 //
32 #include <class/thread.h>
33
34 //
35 // TODO: insert other definitions and declarations here
36 //
37
38 class tskLED1 : public CThread {
39 protected:
40
41     virtual void run() {
42         CPin led(LED1);      // use LED1
43         //
44         // thread run-loop
45         //
46         while(isAlive()) {
47             led = !led;      // blink led
48             sleep(100);
49         }
50     }
51 };

```

Implements the “virtual void run()” member function. and have to understand that “run()” member function is the life-cycle of thread. Put the isAlive() in the while-loop to check whether thread is actively or not.

In the same way, to make other tskLED2, tskLED3 and tskLED4 threads to control the different LEDs and different sleep-time.

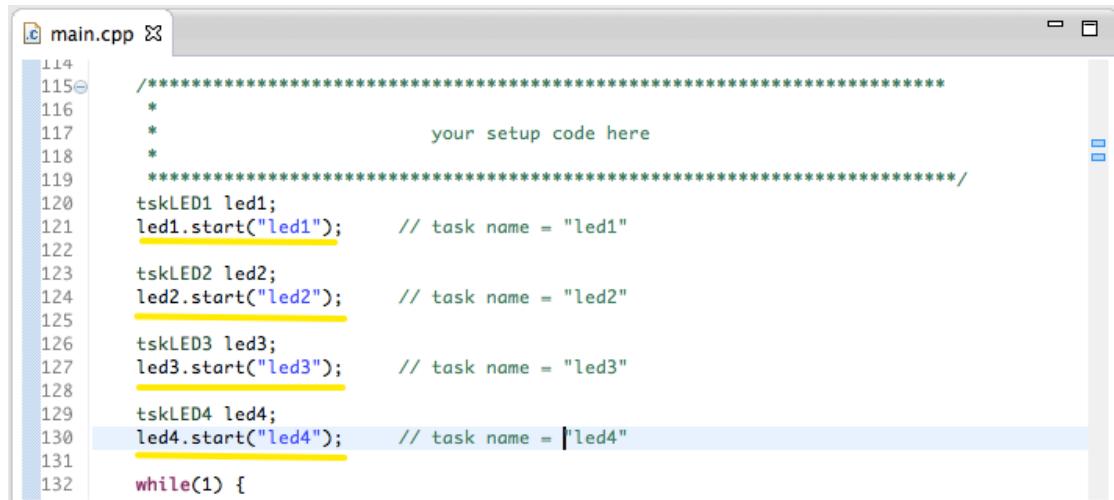


```

52
53 class tskLED2 : public CThread {
54 protected:
55
56 virtual void run() {
57     CPin led(LED2);      // use LED2
58     //
59     // thread run-loop
60     //
61     while(isAlive()) {
62         led = !led;      // blink led
63         sleep(200);
64     }
65 }
66 };
67
68 class tskLED3 : public CThread {
69 protected:
70
71 virtual void run() {
72     CPin led(LED3);      // use LED3
73     //
74     // thread run-loop
75     //
76     while(isAlive()) {
77         led = !led;      // blink led
78         sleep(300);
79     }
80 }
81 };
82
83 class tskLED4 : public CThread {
84 protected:
85
86 virtual void run() {
87     CPin led(LED4);      // use LED4
88     //
89     // thread run-loop
90     //
91     while(isAlive()) {
92         led = !led;      // blink led
93         sleep(400);
94     }
95 }
96 };

```

Start the threads



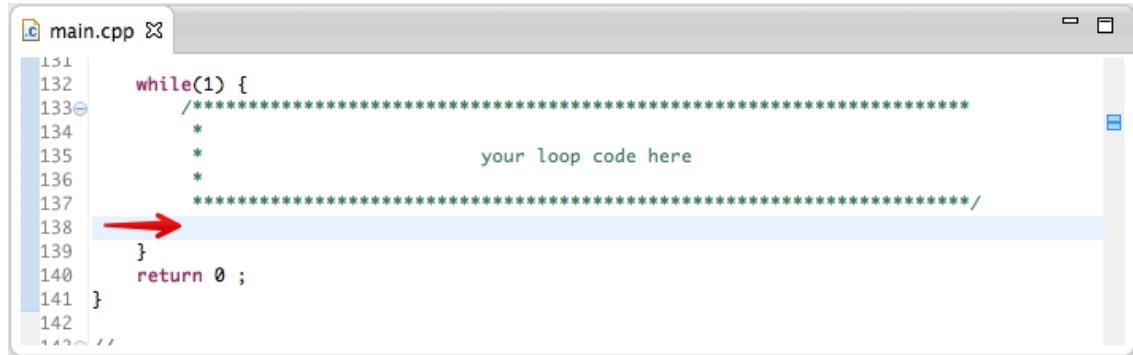
```

114 ****
115 *
116 *          your setup code here
117 *
118 ****
119
120 tskLED1 led1;
121 led1.start("led1");      // task name = "led1"
122
123 tskLED2 led2;
124 led2.start("led2");      // task name = "led2"
125
126 tskLED3 led3;
127 led3.start("led3");      // task name = "led3"
128
129 tskLED4 led4;
130 led4.start("led4");      // task name = "led4"
131
132 while(1) {

```

Remark:

The example uses the LED1~LED4, so you need to remove the all of the original LED demo code from main.cpp as below:



```
131
132     while(1) {
133         /*
134         *          your loop code here
135         *
136         */
137
138     }
139 }
```

To build the example and download the binary file to nano11U37. You will see that LED1, LED2, LED3 and LED4 are blinking standalone.

The final example “blink_thread” project on the GitHub:

/nano11uxx/examples/rtos :

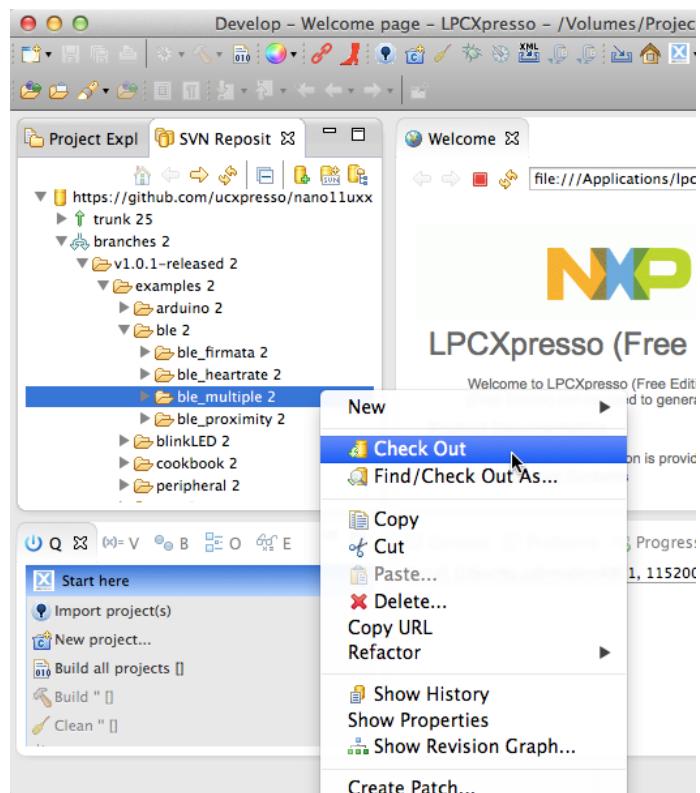
https://github.com/ucpresso/nano11uxx/tree/master/examples/rtos/blink_thread

Appendix A. Examples

A.1 ble_multiple project

The ble_multiple project included:

1. Tickless Power Save Feature
2. BLE Arduino Firmata Test
3. BLE Battery Level Service
4. BLE Health Thermometer Service
5. BLE Proximity Service



iOS App : BLE Arduino

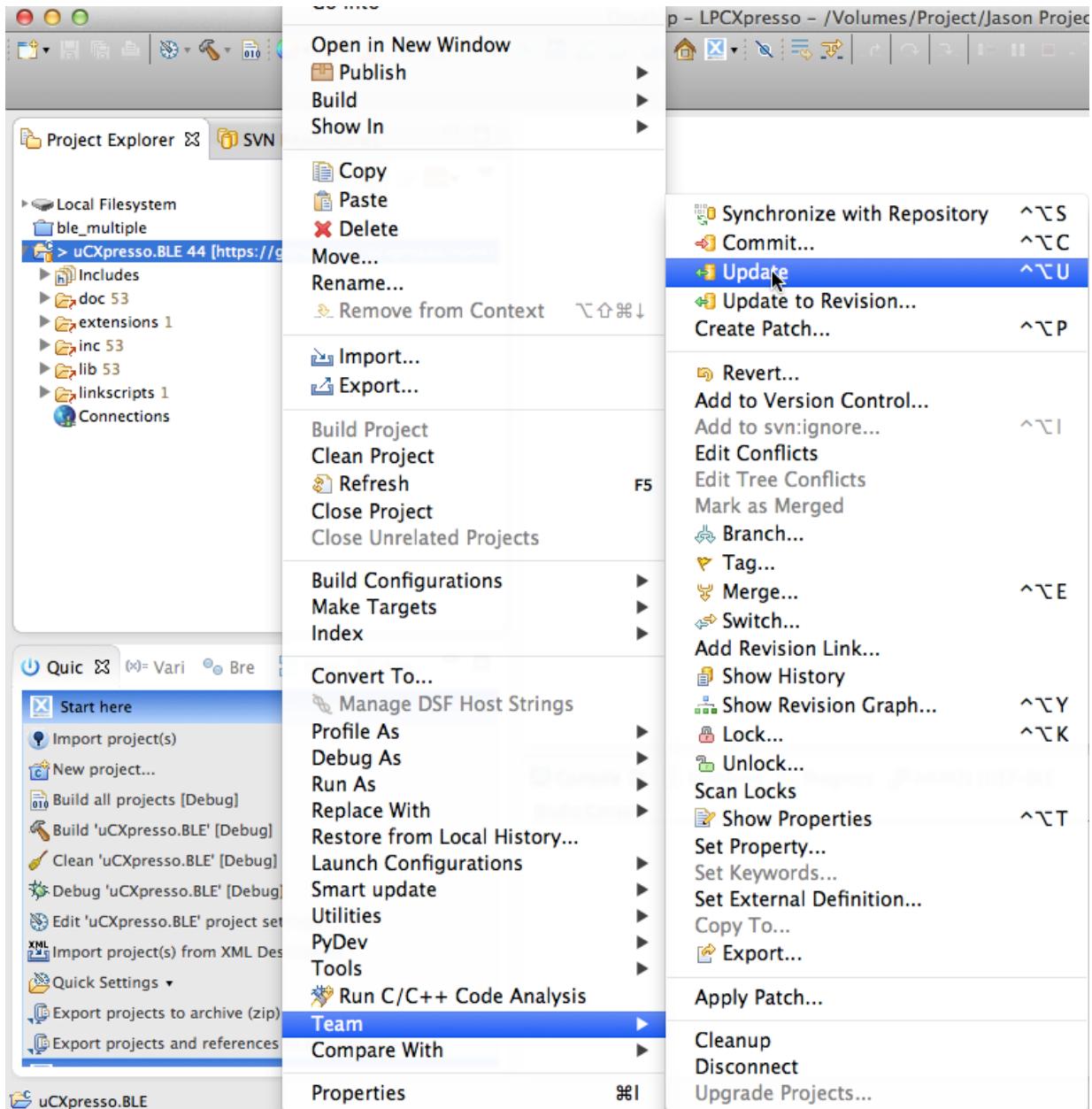
<https://itunes.apple.com/tw/app/ble-arduino/id547628998>



Youtube Firmata Demo: http://youtu.be/7v7_mqfynRA

Appendix B. Update uCXpresso.BLE framework

When your uCXpresso.BLE is check-out from the [trunk](#) of SVN repository, you may use this Appendix A to update your uCXpresso.BLE framework. Click right button of mouse on the “uCXpresso.BLE”, in drop down menu , select Team > Update

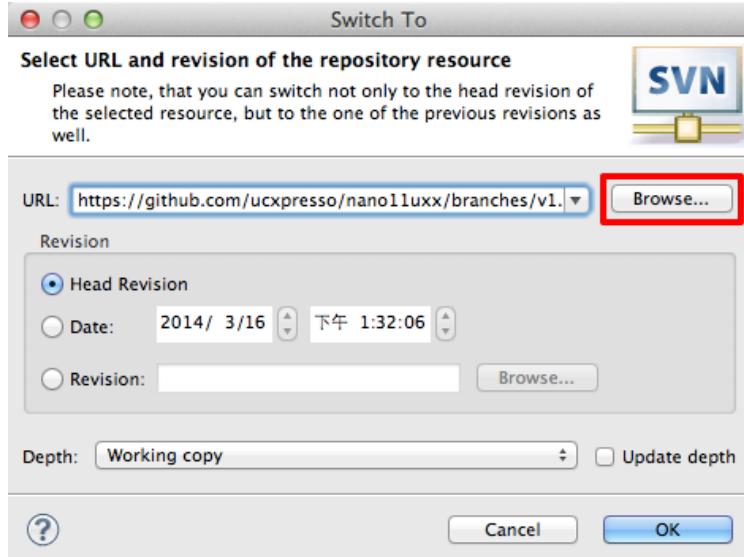


Appendix C. Switch uCXpresso.BLE version Branch

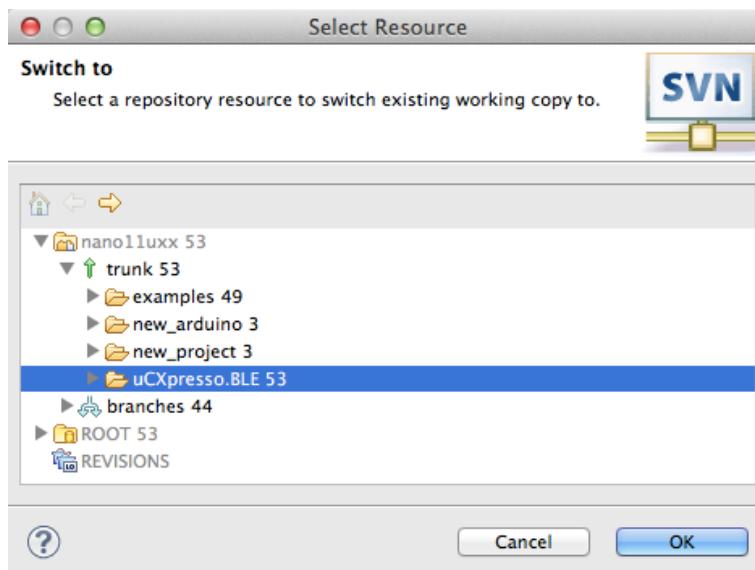
C.1 Switch SVN Branch

To switch the uCXpresso.BLE version branch, click the right button of mouse on the “uCXpresso.BLE” in Project Explorer View.

Select drop down menu “Team” > “Switch...” :



Click [Browse...] select resource (ex. select to trunk/uCXpresso.BLE)



Click [OK] on “Select Resource” > Click [OK] on “Switch To”

Final, see the Appendix A to update the uCXpresso.BLE framework.

Appendix D. Class Manual

D.1 Online Class Manual

<https://rawgithub.com/ucpresso/nano11uxx/master/uCXpresso.BLE/doc/doxygen/html/index.html>

The screenshot shows a web browser window with the title "uCXpresso.BLE: bleSerial C". The address bar contains the URL <https://rawgithub.com/ucpresso/nano11uxx/master/uCXpresso.BLE/doc/doxygen/html/index.html>. The main content area is titled "uCXpresso.BLE v1.0.2" and "RTOS C++ Framework for Bluetooth Low Energy". A navigation bar at the top includes "Main Page", "Modules", and "Classes" (which is selected). Below this are tabs for "Class List", "Class Index", "Class Hierarchy", and "Class Members".

The left sidebar shows a tree structure of the class hierarchy under "uCXpresso.BLE":

- Modules
 - BLE
 - bleBatteryLevel
 - bleHeartRate
 - bleHealthThermometer
 - bleProximity
 - bleSerial
 - Enumerations
 - Peripherals
 - RTOS
- Classes

The right panel is titled "bleSerial Class Reference" and contains the following information:

- BLE**
- bleSerial class** is a ble core, and inherits from **CStream** class to provide the stream virtual functions for serial input and output. the **bleSerial** class also inherits from the **CThread** class and can be work in background.
[More...](#)
- #include "class/ble_serial.h"**
- Inheritance diagram for bleSerial:**

The inheritance diagram shows the following hierarchy:

```
graph TD; CObject --> CStream; CObject --> CThread; CStream --> bleSerial; CThread --> bleSerial;
```

At the bottom of the right panel, it says "Generated on Tue Mar 11 2014 10:13:40 for uCXpresso.BLE by [doxygen](#) 1.8.6".