GETTING STARTED WITH BLE112 DEVELOPMENT KIT

HANDS-ON LAB

Saturday, 08 June 2013

Version 1.0

TABLE OF CONTENTS

1 Introduction

The purpose of this lab is to get you familiar with the DKBLE112, the development kit for the BLE112 single-mode Bluetooth Low Energy module, and to understand how to use its various features for testing and development.

The DKBLE112 provides a comprehensive test bed for Bluetooth Low Energy projects and all features provided by the BLE112 module, including simple methods to use the following interfaces:

- USB (power and data)
- UART (DB9 RS-232
- SPI
- GPIO
- ADC
- CC debugger programming

The board additionally provides a CR2032 coin cell holder for battery power, and three current measurement test points.

Although the development kit has a large number of external components on it, nearly all of the "heavy lifting" is done by the BLE112 module itself. The demo firmware we will use runs entirely on the module itself, and only uses some of the other interfaces for various types of I/O.

2 Preparing the Development Kit

The development kit consists of a single DKBLE112 development board, programming cables, two BLE112 modules and a BLED112 USB dongle. These parts combined with freely available SDK tools provide a development environment which you can use for prototyping both ends of a wireless BLE connection.

For this lab, you will need the **DKBLE112 board**, **mini USB cable**, **CC debugger**, **BLED112 dongle**, and **SDK software** (available from our Tech Forum).

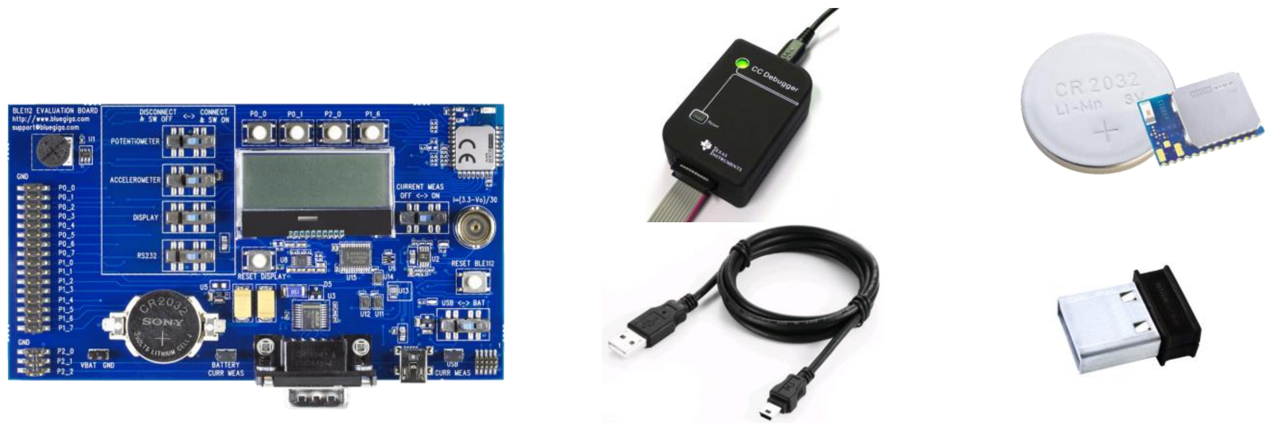


Figure 1: BLE112 Development Kit Contents

2.1 Install the SDK

Before working with the hardware, you should obtain and install the BLE SDK and programming utility that you will be using. This consists of two required main parts and two recommended downloads which provide a good text editor with syntax highlighting:

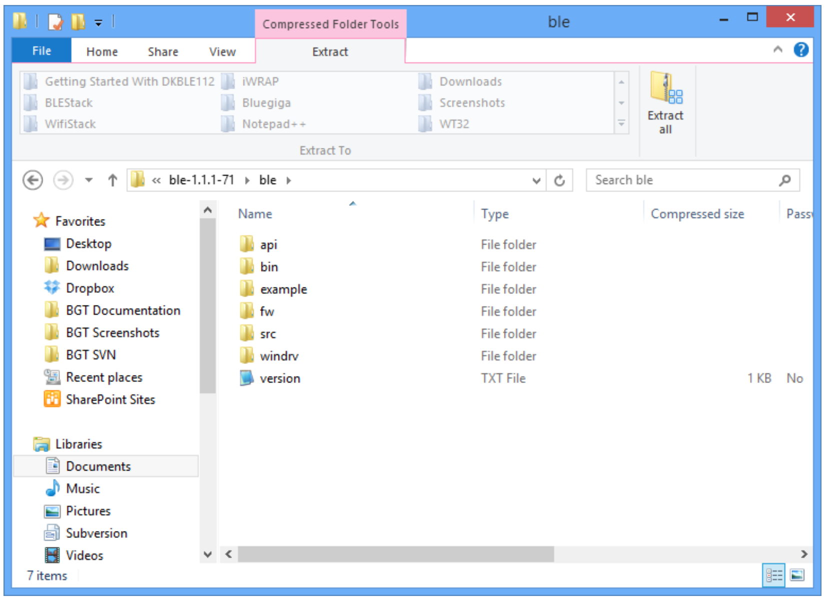
1. BLE SDK archive v1.1.1-71 or later
2. BLE Update utility v1.0.4 or later
3. Notepad++ v6.3.3 or later
4. NP++ BGScript highlighter

You can technically use any text editor you like, but the above utility is recommended because of the availability of the highlighter. The SDK archive, BLE Update utility, and highlighter file are all available from the BLE112 page of our Tech Forum. This requires a free account in order to access. The Notepad++ application is freely available from the author's website.

2.1.1 Extract the BLE SDK archive

After you have downloaded the BLE SDK archive, you should extract all of the contents to your preferred location. All build tools use relative paths, so you can put this anywhere you like. The SDK archive contains build tools in the **/bin** folder and some example projects in the **/example** folder, as well as USB CDC and USB DFU drivers in the **/windrv** folder. For the moment, we are concerned only with the following file:

- **/bin/blegui2.exe** – Windows application for testing with the BLED112 dongle

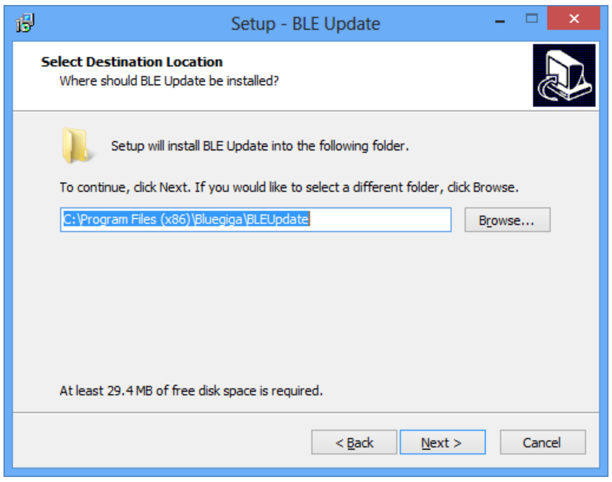


Name	Type	Compressed size	Pass
api	File folder		
bin	File folder		
example	File folder		
fw	File folder		
src	File folder		
windrv	File folder		
version	TXT File	1 KB	No

Figure 2: BLE SDK archive contents

2.1.2 Install the BLE Update utility

The **BLE Update** application provides a simple Windows GUI method to build and flash a BLE project in a single click, as well as providing a way to see your module's serial number, Bluetooth MAC address, and license key. With this utility, you will not need to use the console-based **bgbuild.exe** compiler. Install BLE Update to the desired location before continuing.



Select Destination Location
Where should BLE Update be installed?

Setup will install BLE Update into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

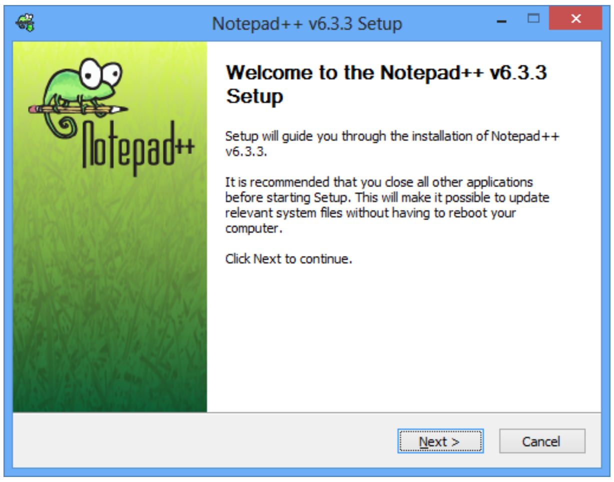
C:\Program Files (x86)\Bluegiga\BLEUpdate

At least 29.4 MB of free disk space is required.

Figure 3: BLE Update installer

2.1.3 Install the Notepad++ Text Editor

The **Notepad++** text editor provides a simple code editor with optional BGScript syntax highlighting. The SDK does not have its own IDE, so this is one of the best ways to get started writing BGScript code. It is an excellent editor for other text formats as well. Install Notepad++ to the desired location as shown below.



Welcome to the Notepad++ v6.3.3 Setup

Setup will guide you through the installation of Notepad++ v6.3.3.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 4: Notepad++ installer

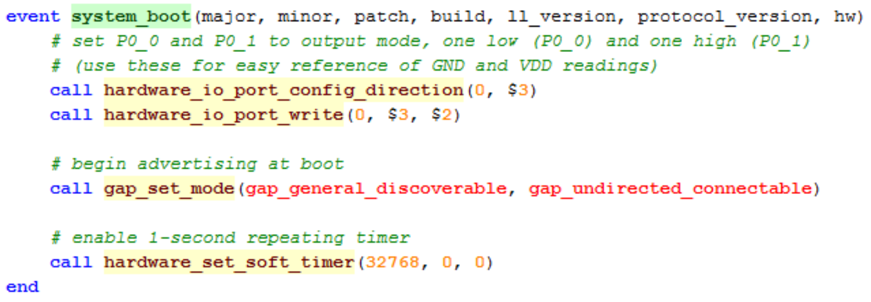
2.1.4 Install the Notepad++ BGScript highlighter

The BGScript highlighter makes it much easier to work with BGScript code within Notepad++. To install this highlighter, perform the following steps:

a. Open Notepad++
b. In the 'Language' menu, click 'Define your language...'
c. Click 'Import...' and select the 'userDefineLang_BGScript.xml' file
d. Copy 'BGScript.xml' file to '<NPP-Install-Dir>\plugins\APIs'
e. In the 'Settings' menu, click 'Preferences...', then 'Backup/Autocompletion'
f. Enable Auto-Completion options if desired

Two things to note:

a. If you already have a BGScript user-defined language in Notepad++, you MUST remove it first.
b. The 'APIs' folder is typically found at 'C:\Program Files\Notepad++\plugins\APIs'.



```
event system_boot(major, minor, patch, build, ll_version, protocol_version, hw)
    # set P0_0 and P0_1 to output mode, one low (P0_0) and one high (P0_1)
    # (use these for easy reference of GND and VDD readings)
    call hardware_io_port_config_direction(0, $3)
    call hardware_io_port_write(0, $3, $2)

    # begin advertising at boot
    call gap_set_mode(gap_general_discoverable, gap_undirected_connectable)

    # enable 1-second repeating timer
    call hardware_set_soft_timer(32768, 0, 0)
end
```

Figure 5: Notepad++ BGScript syntax highlighting sample

3 Testing the Demo Firmware

The DKBLE112 comes preflashed with a demo firmware project to show a few of the available features. This section guides you through the connection and test process, making use of the development kit as a BLE peripheral device as well as the BLED112 dongle as a central device. Note that if you have changed the firmware already, you can follow the main steps in Section 4 to reflash the default ”dkble112_lab” project.

3.1 Connect the DKBLE112 to a power source

You have two options for powering the BLE112 development kit board: USB or battery. For anything other than a quick battery-powered test, it is usually better to use USB for a consistent supply.

3.1.1 USB power

To power the dev kit using USB:

a. Plug the mini USB cable into the connector near the bottom right corner of the board

b. Move the ”USB <-> BAT” power selector switch to the **USB** position

c. Make sure there is a jumper placed on the ”**USB CURR MEAS**” header below the switch

d. Note that a small green LED near should come on when properly connected.

e. Note that Windows may report an ”Unknown device” connected when the ”dkble112_lab” firmware is used. This is expected behavior, because in this demo project, the USB port is used only for power and not for data. The message may be safely ignored in this case.

3.1.2 Battery power

To power the dev kit using a battery:

a. Place a fresh CR2032 coin cell in the battery holder towards the bottom left corner of the board

b. Move the ”USB <-> BAT” power selector switch to the **BAT** position

c. Make sure there is a jumper placed on the ”**BATTERY CURR MEAS**” header near the battery

d. Note that the green power indicator LED used for USB power **will not come on** when battery power is used, in order to reduce current consumption. The BLE112 can drop all way down to the 0.4uA range when fully optimized, and even a weak LED will draw much more current than that.

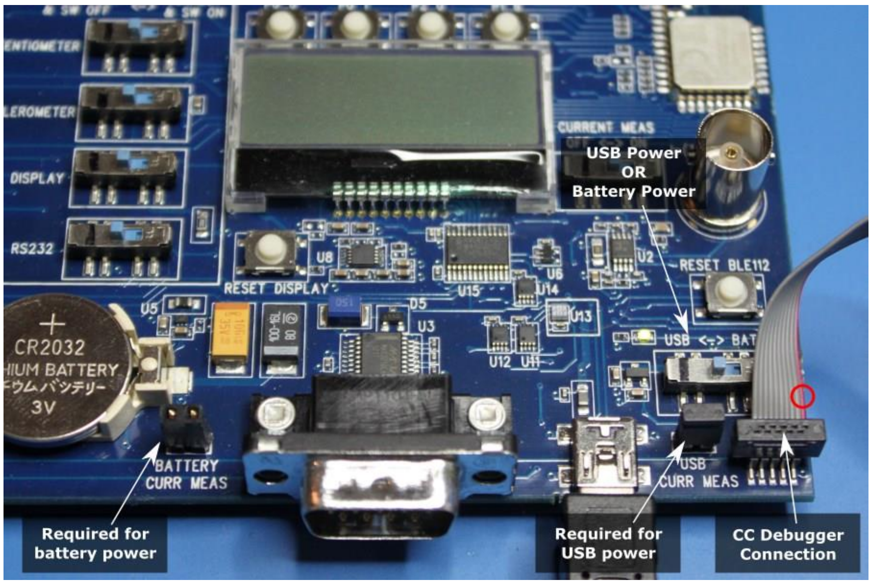


Figure 6: DKBLE112 with USB power and debugger connected

3.2 Testing the DKBLE112 board hardware

Once all of the connections are properly made, the board should be ready to test. There are a number of peripheral switches on the left side of the LCD which should be in the following positions:

- POTENTIOMETER: **ON**
- ACCELEROMETER: OFF
- DISPLAY: **ON**
- RS232: OFF

Once these switches are in the above positions, press the ”RESET DISPLAY” button just below the LCD, and then press the ”RESET BLE112” button towards the right side of the dev kit.

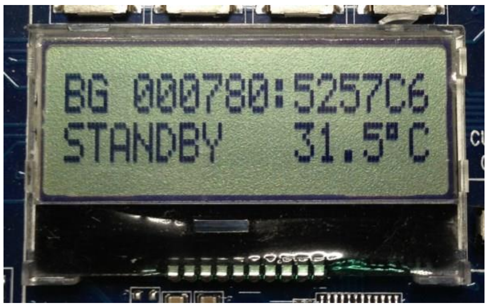


Figure 7: LCD status in standby state

If everything is working correctly, the LCD should show the Bluetooth MAC address in hexadecimal notation, such as ”00:07:80:AA:BB:CC”, along with a ”STANDBY” message which indicates that the device is in a standby state. In order to start sending advertisements (i.e. make it visible and connectable), you will need to press and hold the **P0_0** button above the display. Wait until the next step before doing this.

3.3 Testing the DKBLE112 board remotely via BLEGUI

Now that you know the development kit hardware is working correctly, you can establish a wireless BLE connection using the BLEGUI application and the BLED112 dongle from a remote PC.

3.3.1 Preparing the BLED112 USB dongle

Plug the BLED112 into an available USB port on your PC. If you have already installed the USB CDC drivers on your computer, then it should enumerate as a new COM port (make a note of which port this is) and begin working immediately. If you have not installed the USB CDC driver and you receive an ”Unknown device” error, then you will need to go to Device Manager via the Control Panel and install the driver from the **/windrv** folder found inside the SDK archive.

3.3.2 Scanning and connecting with BLEGUI

Now we are ready to scan for the DKBLE112 and connect to it. Follow these steps:

a. Start BLEGUI (find and run the **/bin/blegui2.exe** application from the extracted SDK)
b. Select the ”Bluegiga Bluetooth Low Energy (COM[x])” device in the dropdown near the top
c. Click the ”Attach” button, at which point you should see a green Connected indicator
d. Make sure the box on the left side marked ”Active Scanning” is checked
e. Make sure the ”Generic” Scan type option is selected
f. Click the ”Start” button to begin scanning for BLE devices
g. Press the P0_0 button on the DKBLE112 board to begin advertising
h. You should see your device appear in BLEGUI in the area on the right.
i. Make sure the MAC address listed matches that on the LCD, then click the ”Connect” button

If everything went smoothly, you are now connected to your BLE device!

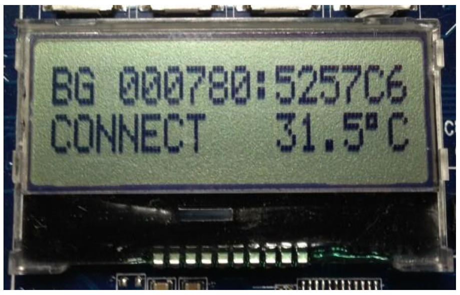


Figure 8: LCD status in connected state

3.3.3 GATT discovery and receiving temperature data

Now that we have connected, you can use BLEGUI to explore the GATT database structure and enable **indications**, or acknowledged pushed data packets, on the temperature measurement. The ”Health Thermometer” service is an official service adopted by the Bluetooth SIG, and the DKBLE112 lab demo firmware implements a simple version of it.

Assuming you are already connected, here are the steps required to enable indications on temperature data:

a. Click the ”GATT” button to the right of your device in the list in BLEGUI

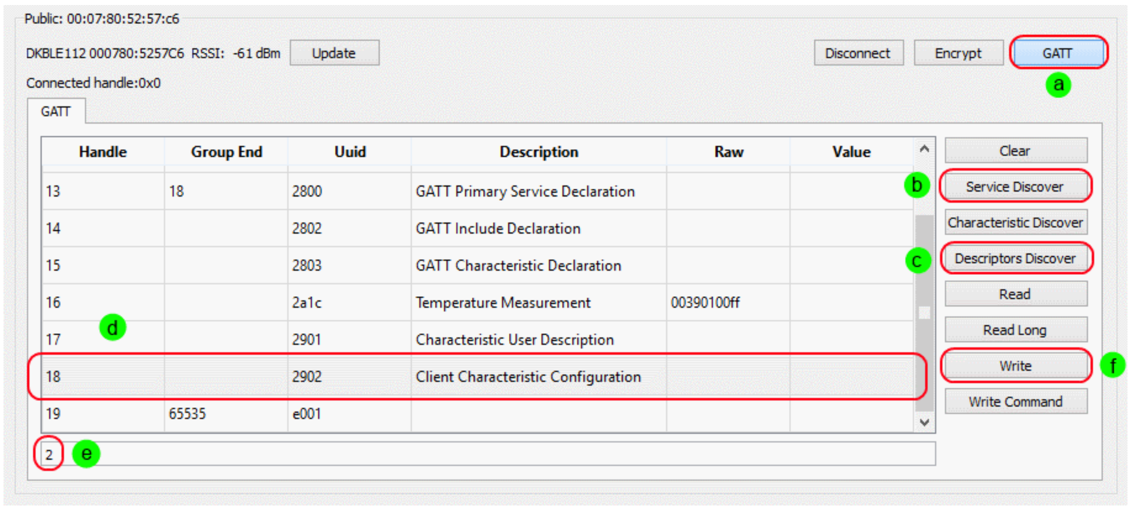
b. Click the ”Service Discover” button

c. Select the ”Health Thermometer” service with UUID=1809 and click the ”Descriptors Discover” button

d. Select the ”Client Characteristic Configuration” entry with UUID=2092 directly below the ”Temperature Measurement” entry with UUID=2A1C.

e. Enter the value ”2” in the long text field immediately below the GATT listing

f. Click the ”Write” button to write ”2” to the Client Characteristic Configuration attribute


Public: 00:07:80:52:57:c6

DKBLE112 000780:5257C6 RSSI: -61 dBm

Connected handle:0x0

Handle	Group End	Uuid	Description	Raw	Value
13	18	2800	GATT Primary Service Declaration		
14		2802	GATT Include Declaration		
15		2803	GATT Characteristic Declaration		
16		2a1c	Temperature Measurement	00390100ff	
17		2901	Characteristic User Description		
18		2902	Client Characteristic Configuration		
19	65535	e001			

2

Figure 9: Enabling indications on the Temperature Measurement attribute

If everything went smoothly, you should now be receiving temperature updates once per second. The data flowing in will be in 5-byte binary packets, which are not really human-readable. The first byte (0x00) is the **flags** byte, which says the measurement will be in Celsius. The remaining four bytes make up a little-endian 32-bit single-precision floating-point number representing the temperature itself. In the image above showing 0x390100ff, the big-endian representation is 0xFF000139, which means the following:

- Sign = 1, number is positive
- Exponent = -1
- Mantissa is 0x139 = 313

Therefore, the temperature is +313 * 10^-1, or 31.3 degrees Celsius.

The Client Characteristic Configuration attribute is a special attribute which gets built into the GATT structure automatically by our BLE stack for any corresponding attributes which support either **notifications** or **indications** (pushed data). Notifications and indications are virtually identical, except that **notifications** are not acknowledged, while **indications** are acknowledged. This concept is analogous to UDP vs. TCP. Notifications may be enabled by writing a value of ”1” instead of ”2” to the configuration attribute.

4 Flashing a Modified Project

In this section, you will make a small modification to the demo ”dkble112_lab” project and flash it onto the development kit in order to become familiar with the BLE Update utility and flashing process.

4.1 Modify the ”dkble112_lab.bgs” BGScript source

Open the **Notepad++** text editor and browse to the ”dkble112_lab” project folder in the files that should be available along with this lab. The only change we need to make is in the **dkble112_lab.bgs** file, which contains the BGScript source for the project.

Near the top of the source file will be an event handler for the ”system_boot” event. About 15 lines into this function is a section where the device name is built one byte at a time as ”DKBLE112 [BT MAC address]”. You will now change the ”DKBLE112” value to something else. This could be anything you like, but note that it must be built one ASCII byte at a time due to BGScript limitations.

For a quick example, instead of using ”DKBLE112”, you could use ”MyDevice”:

```
ascii(0:1) = $4D # 'M'
ascii(1:1) = $79 # 'y'
ascii(2:1) = $44 # 'D'
ascii(3:1) = $65 # 'e'
ascii(4:1) = $76 # 'v'
ascii(5:1) = $69 # 'i'
ascii(6:1) = $63 # 'c'
ascii(7:1) = $65 # 'e'
```

The ASCII to Hex website is handy for the byte conversion. Whatever value you choose should fill all 8 available characters. This restriction also can be changed, but it requires modifying more than just these lines (shifting other byte indexes to fit, and changing the GATT characteristic definition to have an appropriate length). Since the rest of the project relies on exactly a 22-byte name, just leave the rest as-is for now.

Save the source file when you are finished, then move on.

4.2 Connect the CC Debugger

In order to flash a new firmware image onto the device, the CC debugger must be connected. This is a pretty straightforward process:

a. Connect a mini USB cable between the CC debugger and your PC

b. Connect the small 0.1” to 0.05” adapter board to the CC debugger

c. Connect the small 10-pin 0.05” ribbon cable between the CC debugger and the DKBLE112, making sure that both rows of pins are aligned properly in the holes and that the red wire (pin 1) is on the far right edge of the header on the dev kit, not on the side closest to the USB connector.

d. With the DKBLE112 powered and the CC debugger connected, press the small black button on the CC debugger and verify that the debugger’s LED is **green**, not red. This indiates a proper connection. Note that you may need to make extra sure that the 0.1” to 0.05” adapter board is seated properly, as it has been known to make intermittent connections.

e. The CC debugger may not enumerate properly without the correct drivers. If necessary, you can download the drivers directly from TI, or download and install the TI SmartRF Flash Programmer utility which includes drivers for the CC debugger.

4.3 Compile and flash the new project with BLE Update

Now comes the moment of truth! The follow steps will build and flash the modified firmware onto the module, assuming the CC debugger is properly connected as described above:

a. Run the BLE Update utility

b. Make sure the CC debugger device is selected from the Port dropdown

c. Click the ”Info” button in BLE Update to verify that the debugger can communicate with the module

d. Click the ”Browse” button and locate the ”project112.bgproj” file in the **/dkble112_lab** project folder

e. Click the ”BGBuild” menu and verify that an SDK is selected

- If not, click the ”Select manually...” item and browse to the **/bin/bgbuild.exe** file from the extracted SDK archive

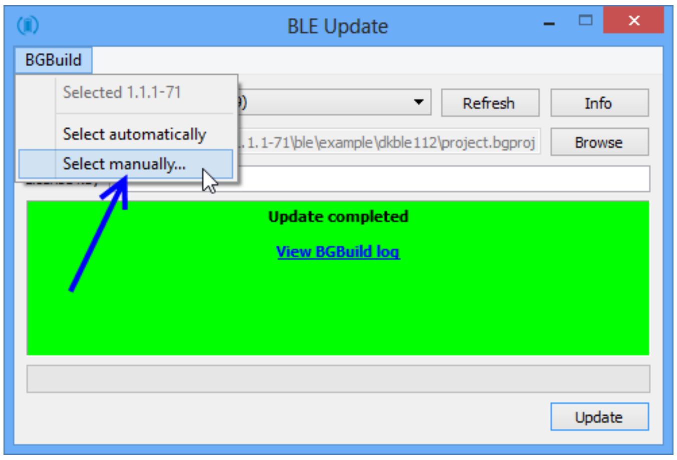


Figure 10: Manually selecting a BGBuild executable

f. Click the ”Update” button to compile and flash the project onto the module

You should now have a working, customized project running on your development kit. You can verify the updated device name value by repeating the scanning steps in BLEGUI described earlier.

Congratulations!

You have now worked through all of the steps necessary to gain a good familiarity with the typical usage of the BLE112 development kit.

5 Contact Information

Sales:	sales@bluegiga.com
Technical support:	support@bluegiga.com http://techforum.bluegiga.com
Orders:	orders@bluegiga.com
WWW:	www.bluegiga.com www.bluegiga.hk
Head Office / Finland:	
	Phone: +358-9-4355 060 Fax: +358-9-4355 0660 Sinikalliontie 5A 02630 ESPOO FINLAND
Postal address / Finland:	
	P.O. BOX 120 02631 ESPOO FINLAND
Sales Office / USA:	
	Phone: +1 770 291 2181 Fax: +1 770 291 2183 Bluegiga Technologies, Inc. 3235 Satellite Boulevard, Building 400, Suite 300 Duluth, GA, 30096, USA
Sales Office / Hong-Kong:	
	Phone: +852 3182 7321 Fax: +852 3972 5777 Bluegiga Technologies, Inc. 19/F Silver Fortune Plaza, 1 Wellington Street, Central Hong Kong