AN997: ETHERNET TO WI-FI BRIDGE MODE

WF121 WI-FI MODULE APPLICATION NOTE

Tuesday, 28 January 2014

Version 1.2

VERSION HISTORY

Version	Comment
1.0	First version
1.1	Minor improvements
1.2	Compatible with SDK version 1.2.2 onwards

TABLE OF CONTENTS

1 Introduction

This application note describes the configuration and usage of Ethernet connectivity in the Bluegiga WF121 Wi-Fi module. The Ethernet, in the WF121, can be used in two different modes either as a Wi-Fi to Ethernet Bridge or alternatively it can be used to access the IP network instead of Wi-Fi. This application note will focus on how to configure and use the Ethernet to Wi-Fi Bridge mode.

This document follows the WF121 Software Development Kit's (SDK) Ethernet-example.

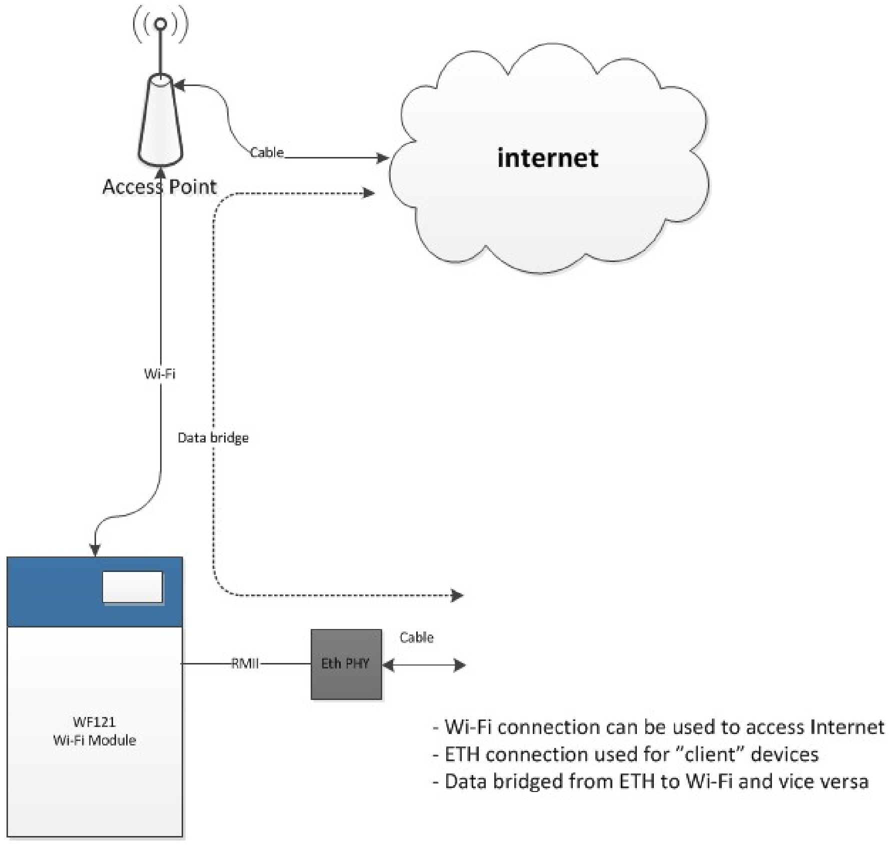

internet
Access Point
Cable
Wi-Fi
Data bridge
WF121
Wi-Fi Module
RMII
Eth PHY
Cable
- Wi-Fi connection can be used to access Internet
- ETH connection used for "client" devices
- Data bridged from ETH to Wi-Fi and vice versa

Figure 1: Illustration of Wi-Fi to Ethernet Bridge

The Ethernet to Wi-Fi Bridge mode can be used to bring wireless connectivity to devices where there is only Ethernet interface available. The Bridge will appear as transparent data pipe, streaming data from Wi-Fi network to the Ethernet device and vice versa. In this example, the setup is made easy with running an HTTP-server on the module, which can be accessed with the Ethernet connection and a web browser. After the Wi-Fi network has been configured the WF121 will act as a Bridge and the Wi-Fi router only sees the end device (i.e., the device connected to WF121 over Ethernet) and assigns an IP address to the end device.

This document will describe the following components:

- Short description of the Bluegiga Wi-Fi Software architecture
- The example's configurations for both Software and Hardware
- A walkthrough of the BGScript implementing example's functionality
- Compiling the Firmware
- Installing the Firmware
- Step by step instructions of example's usage

2 Introduction to the Bluegiga Wi-Fi Software

The Bluegiga Wi-Fi Software enables developers to quickly and easily develop Wi-Fi applications without in-depth knowledge of the Wi-Fi technology. The Wi-Fi Software consists of two parts:

- The Bluegiga Wi-Fi Software
- The Bluegiga Wi-Fi Software Development Kit (SDK)

2.1 The Bluegiga Wi-Fi Stack

The Bluegiga Wi-Fi Stack is an embedded 802.11 MAC and IPv4 stack targeted for Bluegiga's WF121 Wi-Fi module. The Wi-Fi software implements full 802.11 functionality, Station and Access Point modes, WPA2, WPA and WEP and WPS security and various IP based protocols such as TCP, UDP, DHCP, DNS, ICMP and HTTP server.

The Bluegiga Wi-Fi Stack provides powerful, low overhead and easy-to-use Bluegiga BGAPI^{TM} binary API that can be used over UART, USB or SPI interfaces and it provides functions such as such as Access Point discovery, Access Point association, encryption, and connection establishment. The Wi-Fi Stack also supports the 802.11 access point mode and implements an embedded HTTP server for the easy configuration and offers direct connections to phones, tablets and PC's. To simplify the software development the Bluegiga Wi-Fi software also includes a Bluegiga BGLib^{TM} C-library that implements the BGAPI protocol parser for various embedded systems.

For standalone applications the Bluegiga Wi-Fi Software also provides a simple BGScript^{TM} scripting language and VM environment, which enables the users to write simple applications for the WF121 Wi-Fi module. This enables lower cost and smaller designs to be made as there is no need to use an external MCU.

2.2 The Bluegiga Wi-Fi SDK

The Bluegiga Wi-Fi SDK is a software development kit, which enables the device and software vendors to develop products on top of the Bluegiga's Wi-Fi hardware and software.

The Wi-Fi SDK supports multiple development models and the software developers can decide whether the application software runs on a separate host (a low power MCU) or whether they want to make fully standalone devices and execute their code on the MCU embedded in the Bluegiga Wi-Fi modules. The SDK also contains documentation, tools for compiling the firmware, installing it into the hardware and lot of example applications speeding up the development process.

Fully standalone applications can be developed using a simple scripting language called BGScript^{TM}. Several examples are also offered as a part of the Wi-Fi SDK in order to easily develop Wi-Fi compatible end products. These examples contain for example embedded HTTP server, WPS and TCP to Serial applications.

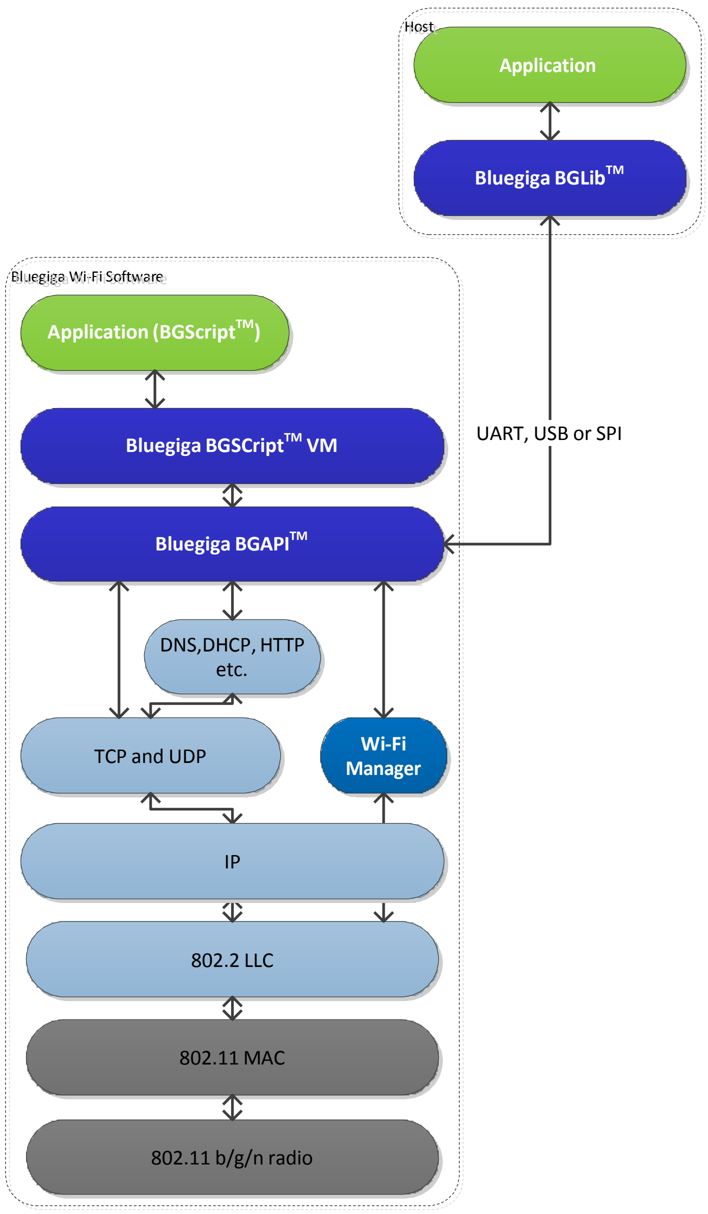


Figure 2: The Bluegiga Wi-Fi Software Architecture

The Bluegiga Wi-Fi Software architecture is illustrated above and it consists of the following components

- The Bluegiga Wi-Fi stack implementing the 802.11, 802.2, IPv4 and higher level protocols
- **BGAPI^{TM}** APIs that enable the software developers to interface to the Wi-Fi Stack
- **BGLib^{TM}** lightweight host library which implements the BGAPI binary protocol and parser and is target for applications where separate host processor is used to interface to the Wi-Fi modules over UART, USB or SPI.
- **BGScript^{TM}** Virtual Machine (VM) and scripting language which enable application code to be developed and executed directly on the Bluegiga Wi-Fi hardware

3 Implementing Ethernet to Wi-Fi Bridge

3.1 Creating a project

The implementation is started by first creating a WF121 project file (**project.xml**), which defines the resources use by the project and the firmware output file.

```
<?xml version="1.0" encoding="UTF-8" ?>
<project>

    <hardware in="hardware.xml" />

    <scripting>
        <script in="wifi.bgs"/>
    </scripting>

    <software>
        <binary in="fw/wifi_http_serv.juo" />
    </software>

    <image out="WF121_ETHBridge.dfu" out_hex="WF121_ETHBridge.hex" />

    <files>
        <file path="404.html"/>
        <file path="main.html"/>
        <file path="client.html"/>
        <file path="index.html"/>
        <file path="logo.gif"/>
        <file path="style.css"/>
    </files>

</project>
```

Figure 3: WF121 Project file

• **<hardware>**	Defines the XML-file or XML tags containing the hardware configuration.
• **<script>**	Defines the BGScript-file which contains the BGScript application code.
• **<binary>**	Defines the WF121 firmware image used in the project. With the Ethernet Bridge example a software profile called **wifi_http_serv** must be used since the example also implements HTTP server functionality which is used for configuration For details of different options see the software profiles chapter in the ***WF121 configuration guide***.
• **<image>**	Defines the DFU and HEX firmware output files
• **<files>**	Defines the HTML, CSS and GIF files used by the embedded HTTP server

3.2 Hardware configuration

The **project.xml** file contains the hardware configuration for WF121 Wi-Fi Module. It describes which interfaces and functions are in used and their properties.

```
<hardware>
    <uart channel="1" baud="5000000" api="true" handshake="True" />
    <ethernet enable="1"/>
    <sleep interrupts="1"/>
</hardware>
```

Figure 4: Hardware configuration for WF121 Wi-Fi Module

• **<uart>**	This setting configures the UART interface. The first UART tag enables the UART2 interface with 115200bps baud rate and hardware flow control enabled. The UART provides access to the BGAPI protocol.
• **<ethernet>**	This setting enables the Ethernet (RMII) interface on the WF121 module.
• **<sleep>**	This setting will enable INT0 (pin 38) wake-up interrupt

The illustration below shows the hardware block diagram on the WF121 development kit and how the development kit should be connected to a PC in order to try out this application.

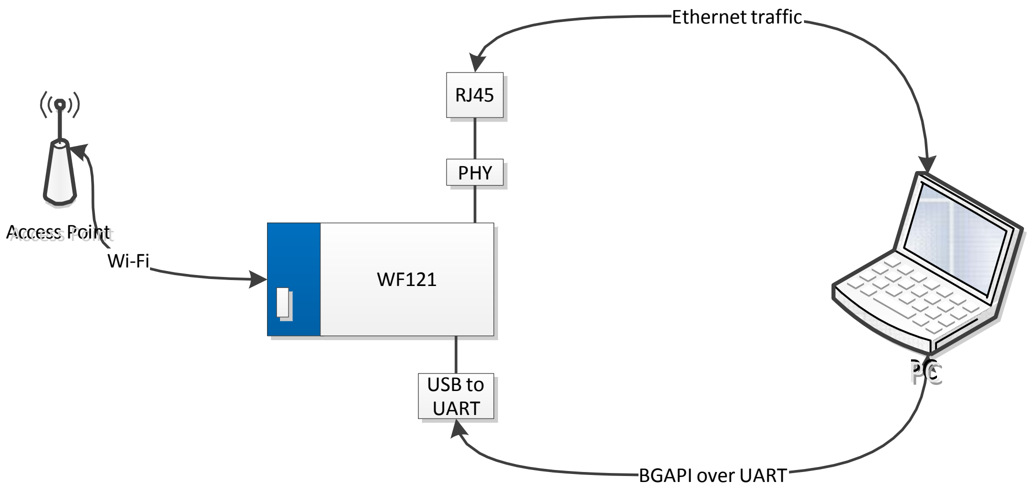


Figure 5: Block diagram of the hardware setup

On the WF121 development kit the following configurations are also needed:

- Switch the Ethernet switch (labelled SW1) to **ON** position

Note:

When Ethernet port is used with the WF121 Wi-Fi Module - and this applies to all Ethernet applications - the following pins / ports of WF121 cannot be used for any other purpose.

- o RD1, RD7, RD8, RD9, RB15, RD[port] and RF0 and RF1.

3.3 The BGScript Code

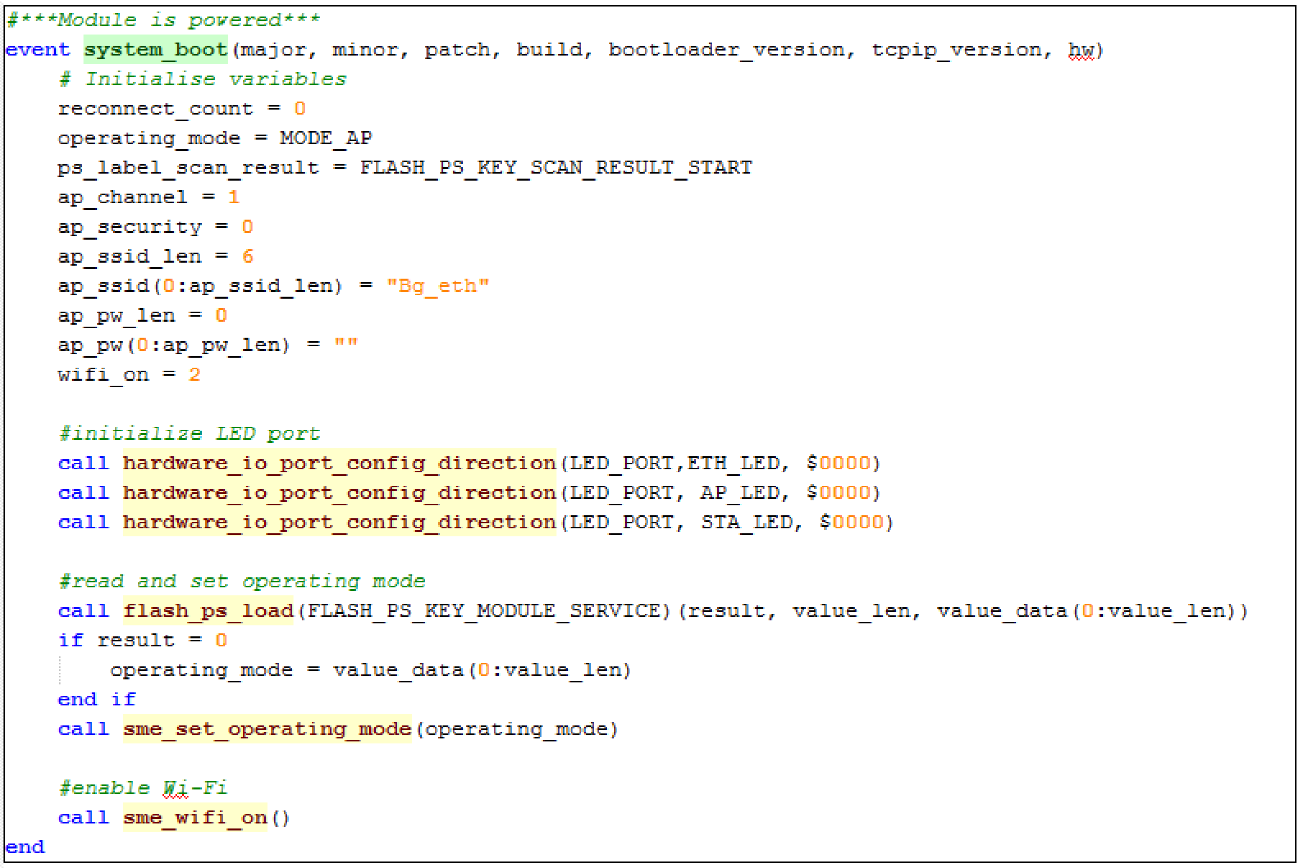
The BGScript application is used to implement the actual application logic controlling the Wi-Fi to Ethernet bridge application. This section of the document briefly walks you through the most important sections of the BGScript application and explains what the script code does.

3.3.1 Catching System Boot

The system boot event is the starting point for all BGScript applications and the event is generated always when the WF121 Wi-Fi module is power on or reset.

In this application the system boot code will perform the following steps:

1. Necessary variables are initialized
2. The last known operating mode is read from the PS keys (parameters stored of flash), in case the application has been used before and the operating mode has been stored to the flash.
    a. The operating mode function is called to set the correct operating mode
3. The Wi-Fi radio is turned on



```
#***Module is powered***
event system_boot(major, minor, patch, build, bootloader_version, tcpip_version, hw)
    # Initialise variables
    reconnect_count = 0
    operating_mode = MODE_AP
    ps_label_scan_result = FLASH_PS_KEY_SCAN_RESULT_START
    ap_channel = 1
    ap_security = 0
    ap_ssid_len = 6
    ap_ssid(0:ap_ssid_len) = "Bg_eth"
    ap_pw_len = 0
    ap_pw(0:ap_pw_len) = ""
    wifi_on = 2

    #initialize LED port
    call hardware_io_port_config_direction(LED_PORT,ETH_LED, $0000)
    call hardware_io_port_config_direction(LED_PORT, AP_LED, $0000)
    call hardware_io_port_config_direction(LED_PORT, STA_LED, $0000)

    #read and set operating mode
    call flash_ps_load(FLASH_PS_KEY_MODULE_SERVICE)(result, value_len, value_data(0:value_len))
    if result = 0
        operating_mode = value_data(0:value_len)
    end if
    call sme_set_operating_mode(operating_mode)

    #enable Wi-Fi
    call sme_wifi_on()
end
```

Figure 6 : System boot event listener

3.3.2 Scanning for available networks and starting the Wi-Fi AP mode

When the application is started the very first time (or after factory reset) it BGScript code will initialize a Wi-Fi Access Point scan in order to discovery nearby networks, store the results into PS keys and finally turn the WF121 Wi-Fi Module into an Access Point mode to initialize the stack.

- **sme_wifi_is_on**() event listener initiates the Access Point scan

- When the scan is ready the event **sme_scanned()** in generated, which on the other hand initiates sorting of the scan results based on RSSI
- Scan results are then stored into PS-keys for future usage by **sme_scan_sort_result** event listener
- Finally when sorting has been finished the WF121 is turned into Wi-Fi mode by the **sme_scan_sort_finished** event listener.

The application can be configured either over a wireless link or over the Ethernet connection. The end user can connect the WF121 Wi-Fi Module Access Point, if Ethernet cable is not connected, for example from a smart phone or a PC and access the built in HTTP server and configure the Wi-Fi network WF121 is supposed to connect to by selecting the SSID and security configuration of the network. When the Ethernet cable is connected, the HTTP server has to be accessed over that connection.

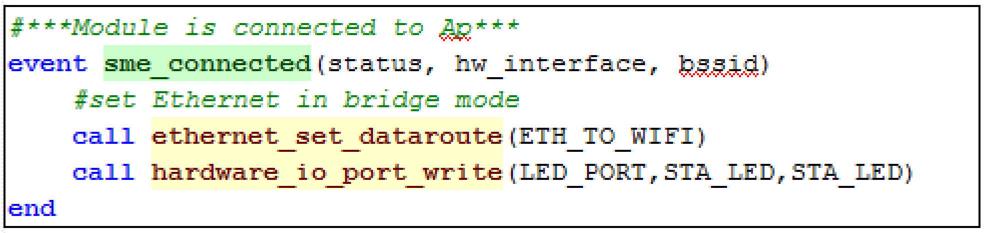
- **start_ap_mode** procedure starts the Wi-Fi AP mode and loads the previous (or factory default) settings from the PS-keys such as the SSID, password DHCP and DNS server configurations. Also the Ethernet bridging is disabled at this state of the application.
- When the AP mode has been started an **sme_ap_mode_started event** is generated, which finally enables the HTTP and DHCP servers.

After these steps have been completed by the BGScript application the device should be visible as a Wi-Fi Access Point, if the Ethernet cable is not connected, and can be connected from a smart phone, table or PC in order to configure the device via the HTTP server.

3.3.3 Connecting to a Wi-Fi Access Point and starting the Ethernet bridge mode

Once the end used has configured the WF121 Wi-Fi module via the HTTP server and selected the Wi-Fi network it needs to connect to the BGScript application performs the following steps:

- After reset and when Wi-Fi is turned on in the **sme_wifi_is_on** event handler the **start_sta_mode procedure** is called in order to start the Wi-Fi client (STA) mode and connect to a network
- In the **start_sta_mode** procedure the DHCP client is enabled and the HTTP, DHCP and DNS servers are disabled. Also the settings of the Access Point that the WF121 is supposed to connect to are read from the PS keys.
- Finally **sme_connect_ssid** is called to begin the connection establishment, authentication and encryption procedures.
- If the connection is successful **sme_connected** event is generated and the Ethernet bridge mode is also started with command **ethernet_set_dataroute(ETH_TO_WIFI).**



```
#***Module is connected to Ap***
event sme_connected(status, hw_interface, bssid)
    #set Ethernet in bridge mode
    call ethernet_set_dataroute(ETH_TO_WIFI)
    call hardware_io_port_write(LED_PORT,STA_LED,STA_LED)
end
```

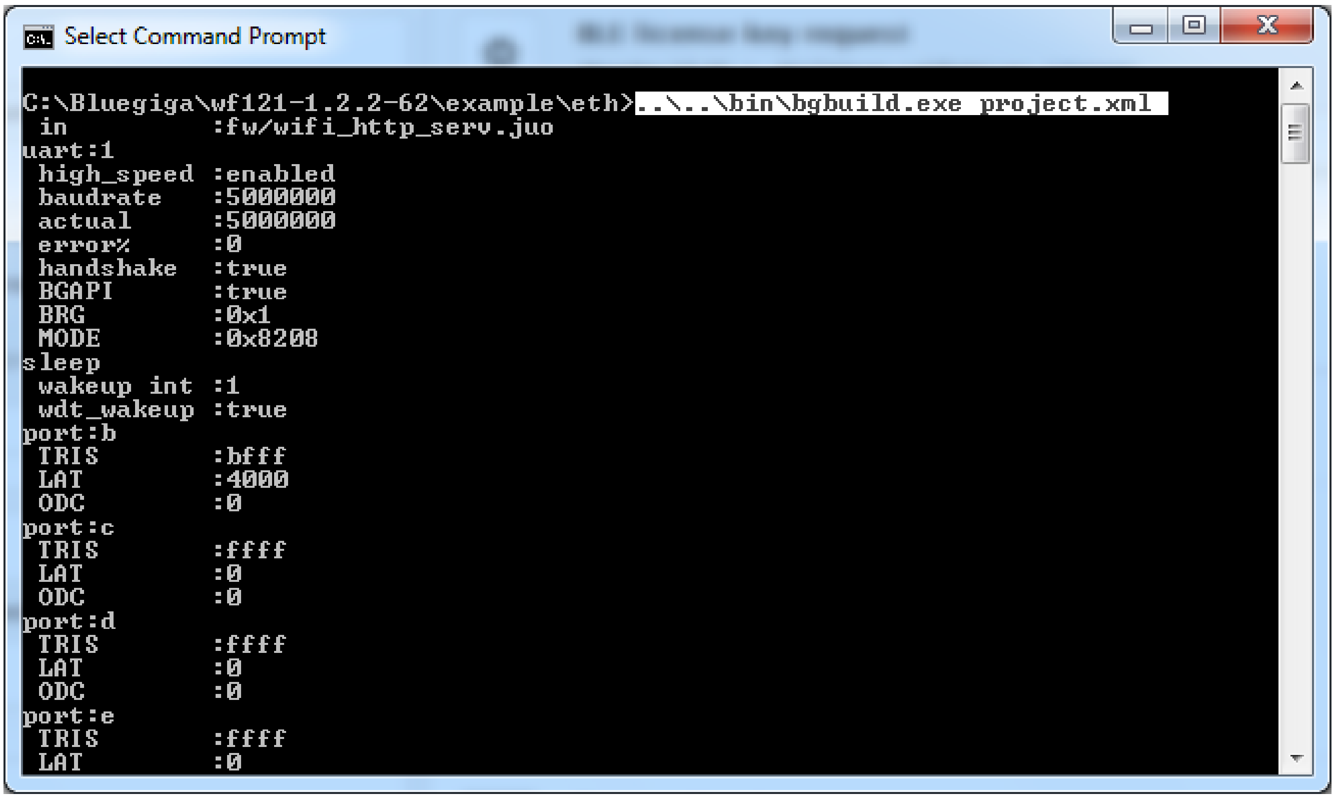
Figure 7: Starting Ethernet Bridge

Note:

The status LEDs D4 and D5 state is changed to always ON when the bridging connection is successful. For other led states and functionality, please refer to the actual BGScript code.

4 Compiling the Firmware

The firmware is compiled with either the **bgbuild.exe**, this can for example be done from the Windows Command Line Prompt (**cmd.exe)**, or the WF121 GUI's built-in compiler tool, please refer to Figure 10: Updating firmware via DFUFigure 10. The figure below shows how to compile the Ethernet example project with the BGBuild compiler.



```
C:\Bluegiga\wf121-1.2.2-62\example\eth>..\..\bin\bgbuild.exe project.xml
 in           :fw/wifi_http_serv.juo
uart:1
 high_speed  :enabled
 baudrate    :5000000
 actual      :5000000
 error%      :0
 handshake   :true
 BGAPI       :true
 BRG         :0x1
 MODE        :0x8208
sleep
 wakeup int :1
 wdt_wakeup :true
port:b
 TRIS        :bfff
 LAT         :4000
 ODC         :0
port:c
 TRIS        :ffff
 LAT         :0
 ODC         :0
port:d
 TRIS        :ffff
 LAT         :0
 ODC         :0
port:e
 TRIS        :ffff
 LAT         :0
```

Figure 8: Screen capture of compiling the project with bgbuild.exe

Compiler output and explanations with the default configurations:

Feature	Output	Explanation
uart:1	high_speed :enabled baudrate :5000000 actual :5000000 error% :0 handshake :true BGAPI :true BRG :0x1 MODE :0x8208	Shows that UART2 is enabled at 5Mbps baud rate. RTS/CTS handshaking is enabled. BGAPI protocol is also ENABLED for UART2.
sleep	wakeup int: 1 wdt_wakeup: true	Enabled sleep settings
port: N	TRIS :ffff LAT 0 ODC 0	Shows the default configuration for Port N (B to G) and the setting for tri-state configuration bit mask, open drain configuration, and latched value for the port.
script	compiler :C:/Bluegiga/wf121-1.2.x-YY/bin/script_compiler.exe script :wifi.bgs api :../../api/wifiapi.xml stack :512 variables :271	Shows the directory where the bgbuild compiler is located. Shows the BGScript source code file.
HTTP server	FILE :404.html size:202 FILE :main.html size:280 FILE :client.html size:2744 FILE :index.html size:1173 FILE :logo.gif size:1011 FILE :style.css size:1255	Shows the HTML pages and their sized that are compiled into the firmware.
Stack size	Memory Used : SW :402104 HW :122 USB :0 Script:1035 Files :6760 Free :101979/512000(20%)	SW shows the flash usage (in B) of the firmware image. HW shows the size of the hardware configuration. USB shows the size of the USB interface descriptor. Script shows the size of BGScript. Files shows the size if the HTTP server files Free shows the total size of the software and how much space is left in the device.

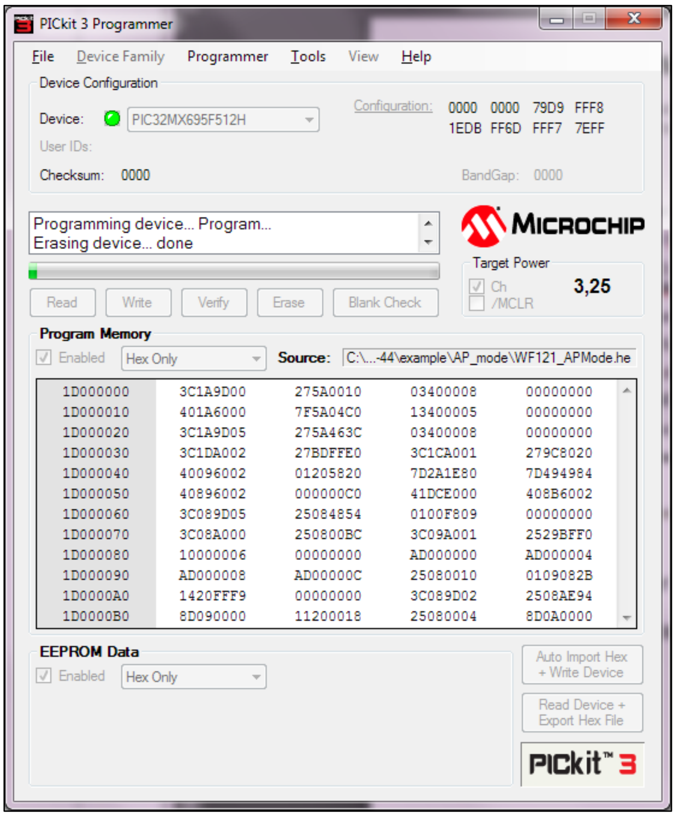
Table 1: Compiler output print explained

5 Installing the Firmware

The firmware can be installed either using the DFU protocol over UART or USB or via the debug interface using the Microchip PICkit3 tool and software.

5.1 Using PICkit 3

- As **PICkit 3** will erase the full flash, please write down the MAC (IEEE) address of your device
- Download and install **PICkit 3** software from Bluegiga web site
- Connect the **PICkit 3** to the debug interface of your WF121 (header J11 on WF121 development kit) and connect the **PICkit 3** to your PC via USB interface.
- Start **PICkit 3** software
    - From **Device Family** select **PIC32**
    - From **Device** drop down list select model : **PIC32MX695F512H**
    - Verify the **STATUS** led on the PICkit 3 device turns **green**
    - From **File** select **Import Hex**
    - Choose the **.hex** file output by the **BGBuild** compiler
    - Press **Write**
- Wait for the programming to be successfully finalized



Device: ● PIC32MX695F512H

Configuration: 0000 0000 79D9 FFF8 1EDB FF6D FFF7 7EFF

User IDs:

Checksum: 0000

BandGap: 0000

```
Programming device... Program...
Erasing device... done
```

✓ Ch 3,25
/MCLR

✓ Enabled Hex Only Source: C:\...-44\example\AP_mode\WF121_APMode.he

```
1D000000   3C1A9D00   275A0010   03400008   00000000
1D000010   401A6000   7F5A04C0   13400005   00000000
1D000020   3C1A9D05   275A463C   03400008   00000000
1D000030   3C1DA002   27BDFFE0   3C1CA001   279C8020
1D000040   40096002   01205820   7D2A1E80   7D494984
1D000050   40896002   000000C0   41DCE000   408B6002
1D000060   3C089D05   25084854   0100F809   00000000
1D000070   3C08A000   250800BC   3C09A001   2529BFF0
1D000080   10000006   00000000   AD000000   AD000004
1D000090   AD000008   AD00000C   25080010   0109082B
1D0000A0   1420FFF9   00000000   3C089D02   2508AE94
1D0000B0   8D090000   11200018   25080004   8D0A0000
```

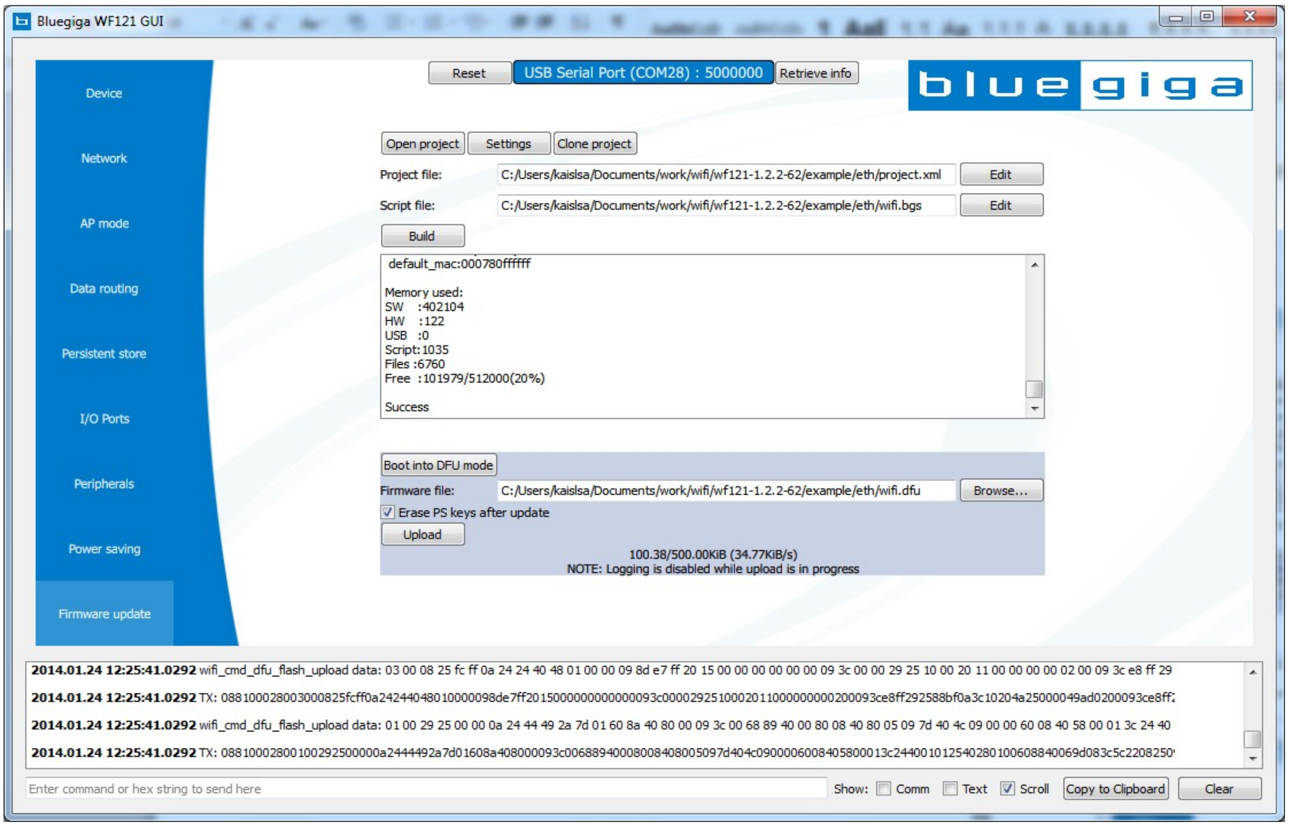
✓ Enabled Hex Only

Figure 9: Programming firmware using PICkit 3

5.2 Using DFU over UART or USB

In order to install the firmware using DFU protocol, please do the following steps:

- Connect the WF121 Wi-Fi Module to PC via UART or USB, whichever is enabled on the firmware.
- Start **WF121 GUI** software
- Select the correct **COM** port and baud rate
    - Factory default settings: UART2 and 115200 baud rate
    - Verify the communication works for example by pressing **Retrieve info** button
- Go to **Firmware update** subpage
    - Press **Boot in DFU mode** button
        - A successful DFU mode is indicated with the event : **wifi_evt_dfu_boot version: 4**
    - Select the correct .DFU file using the **Browse…** button
    - Press **Upload**
- Make sure the firmware is uploaded correctly and the device boots normally
    - A successful DFU upload is indicated with event: **wifi_rsp_dfu_flash_upload_finish result: 0**
    - A successful boot is indicated with event: **wifi_evt_system_boot**



USB Serial Port (COM28) : 5000000

Project file: C:/Users/kaislsa/Documents/work/wifi/wf121-1.2.2-62/example/eth/project.xml

Script file: C:/Users/kaislsa/Documents/work/wifi/wf121-1.2.2-62/example/eth/wifi.bgs

```
 default_mac:000780ffffff

Memory used:
SW    :402104
HW    :122
USB   :0
Script:1035
Files :6760
Free  :101979/512000(20%)

Success
```

Firmware file: C:/Users/kaislsa/Documents/work/wifi/wf121-1.2.2-62/example/eth/wifi.dfu

☑ Erase PS keys after update

100.38/500.00KiB (34.77KiB/s)
NOTE: Logging is disabled while upload is in progress

```
2014.01.24 12:25:41.0292 wifi_cmd_dfu_flash_upload data: 03 00 08 25 fc ff 0a 24 24 40 48 01 00 00 09 8d e7 ff 20 15 00 00 00 00 00 00 09 3c 00 00 29 25 10 00 20 11 00 00 00 00 02 00 09 3c e8 ff 29
2014.01.24 12:25:41.0292 TX: 088100028003000825fcff0a242440480100000098de7ff20150000000000000093c000029251000201100000000200093ce8ff292588bf0a3c10204a25000049ad0200093ce8ff
2014.01.24 12:25:41.0292 wifi_cmd_dfu_flash_upload data: 01 00 29 25 00 00 0a 24 44 49 2a 7d 01 60 8a 40 80 00 09 3c 00 68 89 40 00 80 08 40 80 05 09 7d 40 4c 09 00 00 60 08 40 58 00 01 3c 24 40
2014.01.24 12:25:41.0292 TX: 0881000280010029250000a2444492a7d01608a408000093c006889400080084080050 97d404c09000060084058000 13c24400101254028010060884 0069d083c5c2208250
```

Enter command or hex string to send here

Figure 10: Updating firmware via DFU

Note:

DFU updates allow the change of hardware settings (in project.xml), so make sure you do not make unwanted changes to your hardware configuration when performing DFU updates.

6 Testing the Ethernet Bridge

This chapter briefly describes how to configure, using the HTTP server, the WF121 Wi-Fi Module to join an existing Wi-Fi network and start the Ethernet Bridge mode.

6.1 Test Setup

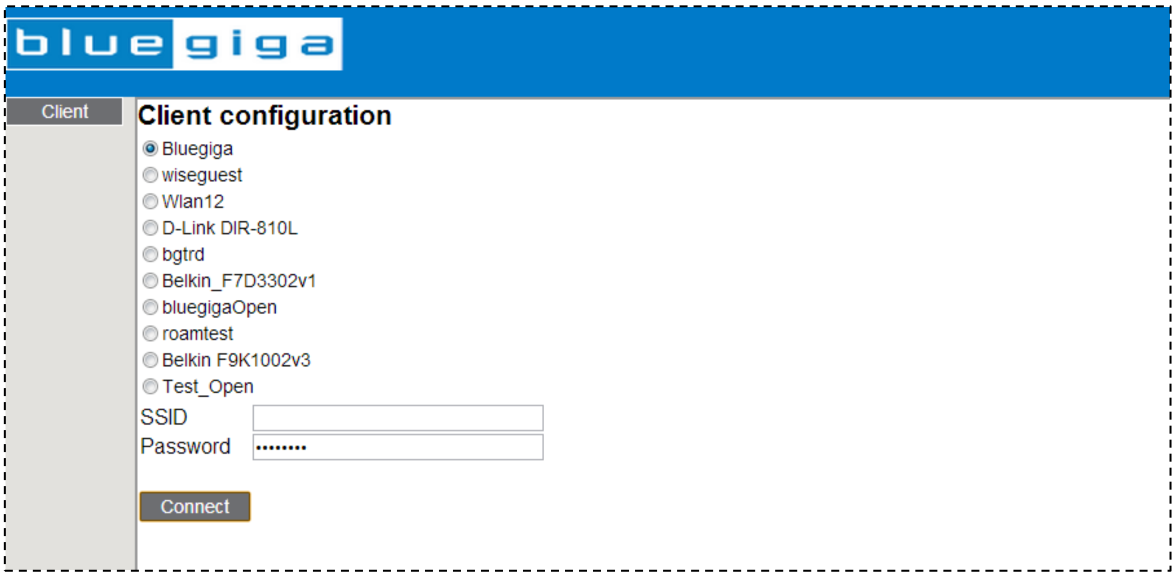
- DKWF121 development kit
    - Ethernet Bridge example firmware installed and not configured before
    - An Ethernet crossover cable available, but not connected
    - USB (USB-to-UART converter) connected to a PC to access the module with the WF121 GUI
- A Windows PC with a web browser
- A wireless network as a gateway to the Internet

6.2 Enabling the Ethernet bridge mode

1. Connect a USB cable between the PC and the WF121 development kit's USB-to-UART Converter.
2. Windows enumerates the two virtual COM ports, when the WF121 development kit is powered ON. You can verify this from the Windows' device manager.
3. Start the WIFIGUI software delivered with the SDK and **Attach** up the higher of the two COM ports and using the 115200bps baud rate.
    a. Check the BGAPI communication works, for example by pressing **Retrieve Info** button and checking from the communication log that a proper response is received.
4. The WF121 module searches for wireless networks in the area. When the scan is finished and the WF121 module is in the Access Point mode, the RD6 led remain constantly on.
5. Connect the WF121 Access Point for example using your PC and navigate to the IP address http://192.168.1.1 with your web browser. Make sure the PC is not connected to any other networks when you do this.
6. In the HTTP server the **Client**-page displays the discovered wireless networks from the area. Select a network you want to join to, or set the SSID of a known network to the corresponding text box. Next type in the password the network uses and press **Connect**. The WF121 Wi-Fi module will then connect the selected Access Point as a Wi-Fi client. Connect the crossover cable between the PC and the devkit. If the setup is successful, Ethernet bridging will be enabled and the RD4 and RD5 LEDs will turn ON.

    Note:

    The web browser might prompts that the connection to the HTTP server at 192.168.1.1 was lost and won't be accessible any more. This is normal as the WF121 module is now in the Ethernet bridge mode.
7. To access the HTTP server again you can use the WIFIGUI application and reset the PS keys. You need to write the FLASH_PS_KEY_MODULE_SERVICE value to 2 and reset the WF121 Wi-Fi module and it will start again in the Wi-Fi Access Point mode. You can use WF121 GUI Firmware Update to erase the PS Keys as well.


Client

Client configuration

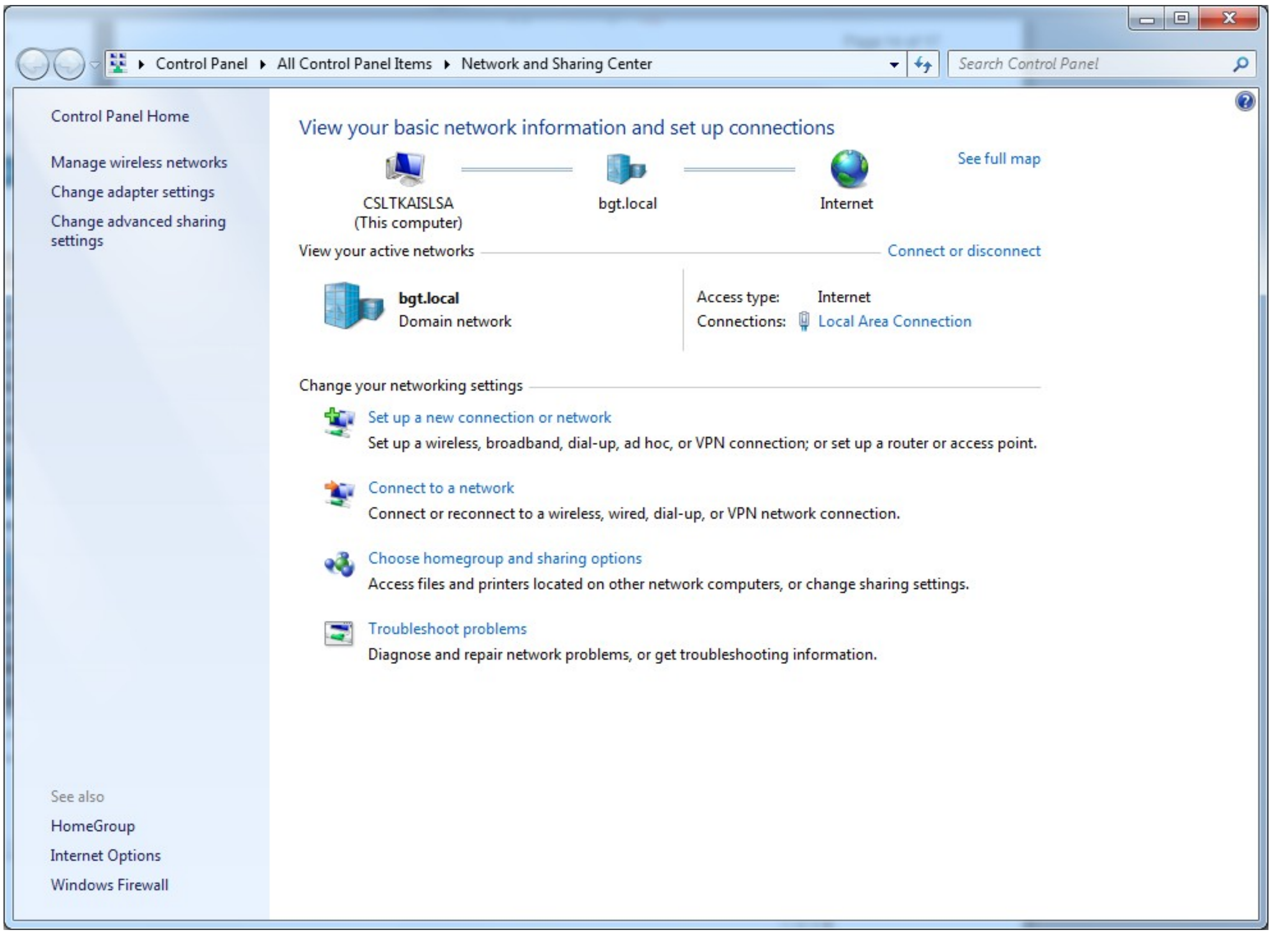
- (●) Bluegiga
- ( ) wiseguest
- ( ) Wlan12
- ( ) D-Link DIR-810L
- ( ) bgtrd
- ( ) Belkin_F7D3302v1
- ( ) bluegigaOpen
- ( ) roamtest
- ( ) Belkin F9K1002v3
- ( ) Test_Open

SSID	
Password	••••••••

Connect

Figure 11: The Client mode configuration page

Once the bridge mode has been enabled your PC should be connected to the same network as WF121 Wi-Fi module through the Wi-Fi to Ethernet Bridge. You can verify this for example from the Windows' Network and Sharing Center.


Control Panel ▸ All Control Panel Items ▸ Network and Sharing Center

Search Control Panel

Control Panel Home

Manage wireless networks

Change adapter settings

Change advanced sharing settings

View your basic network information and set up connections

CSLTKAISLSA (This computer) — bgt.local — Internet

See full map

View your active networks — Connect or disconnect

bgt.local
Domain network

Access type:	Internet
Connections:	Local Area Connection

Change your networking settings

Set up a new connection or network
Set up a wireless, broadband, dial-up, ad hoc, or VPN connection; or set up a router or access point.

Connect to a network
Connect or reconnect to a wireless, wired, dial-up, or VPN network connection.

Choose homegroup and sharing options
Access files and printers located on other network computers, or change sharing settings.

Troubleshoot problems
Diagnose and repair network problems, or get troubleshooting information.

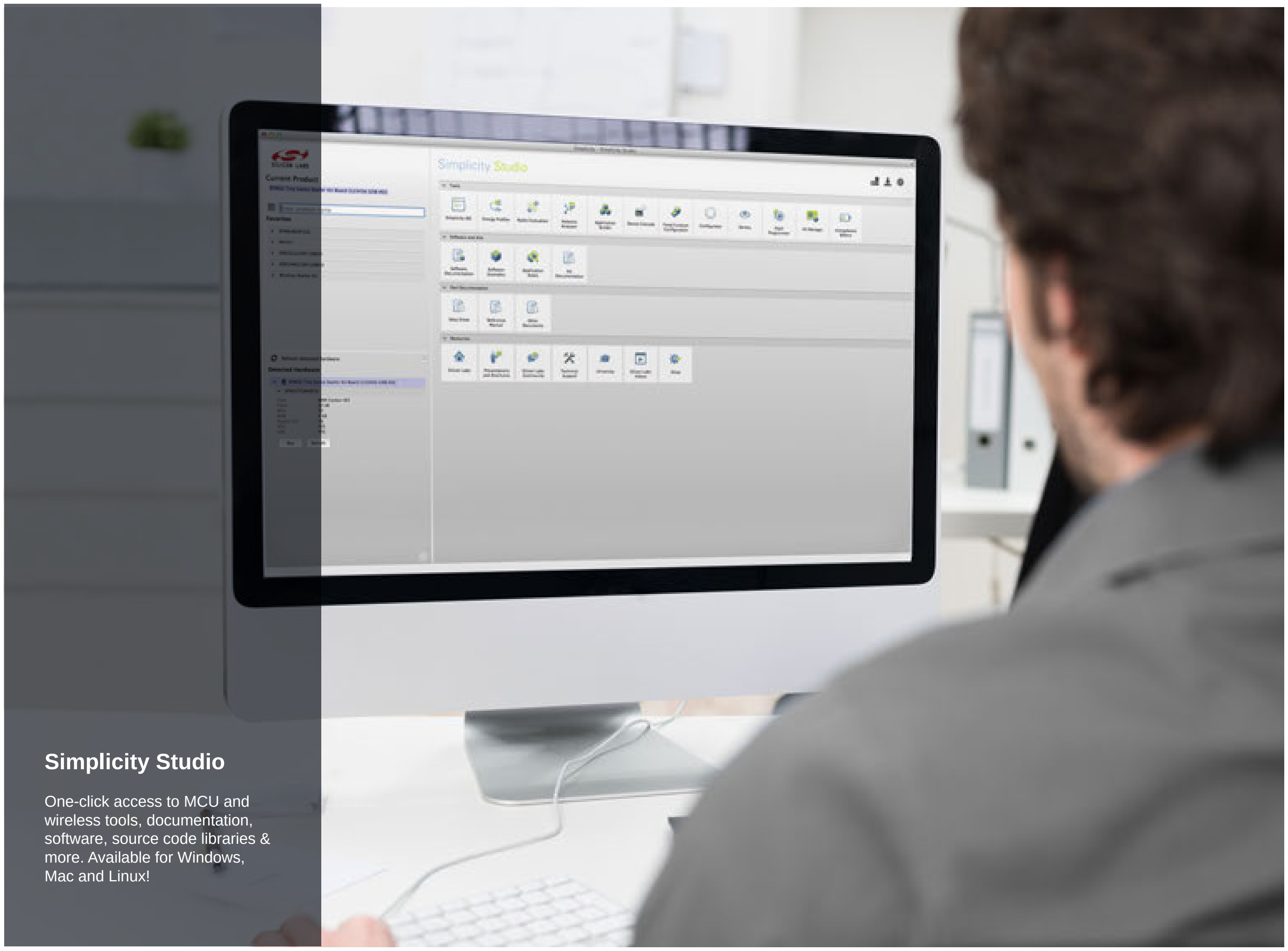
See also

HomeGroup

Internet Options

Windows Firewall

Figure 12: Windows' Network and Sharing Center displaying a Local Area Connection



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio *www.silabs.com/IoT*	**SW/HW** *www.silabs.com/simplicity*	**Quality** *www.silabs.com/quality*	**Support and Community** *community.silabs.com*

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com