AN1024: ETHERNET EXAMPLES

APPLICATION NOTE

Tuesday, 22 September 2015

Version 1.3

Contents

1 Examples Overview

1.1 Introduction

This Application Note covers the configuration and usage of the ethernet interface included with the WF121 module for software versions 1.3 and later. The SDK includes two separate ethernet examples, "eth" and "eth_device". The first example, "eth", demonstrates how the WF121 can be configured to operate as an Ethernet-to-WiFi bridge. The second example, "eth_device", demonstrates how the WF121 can be configured to operate as an Ethernet Device. This application note breaks down each of the two examples into their respective components so that each component can be described in a functional sense individually.

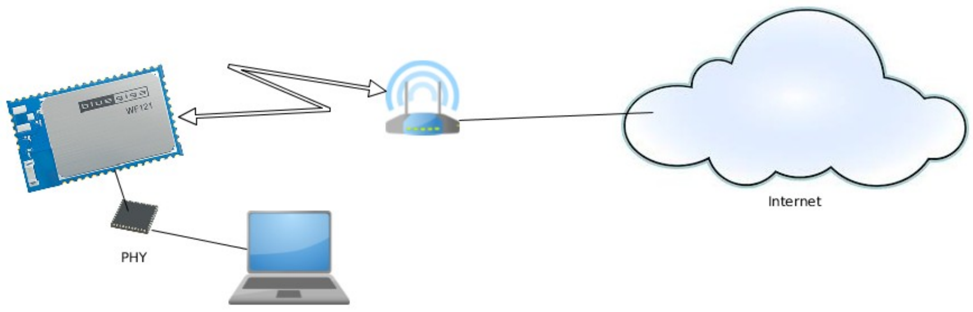
Each of the two projects includes 2 BGScript source files, an HTML source file and companion CSS style sheet, a project configuration file, and an image used by the HTML source file. The BGScript source in each project are responsible for the application logic while the HTML source files provide a user facing medium (Web Page) for user defined parameters. This application note is comprised of three sections: the first sections provides an overview of the possible Ethernet connectivity modes, the second covers the "eth" example, and the last section covers the "eth_device" example.

1.2 Ethernet connect modes

The WF121 is capable of operating in the one of the three following Ethernet "modes":

- Ethernet-to-Wi-Fi bridge
- Ethernet Server
- Ethernet Device

1.2.1 Bridge

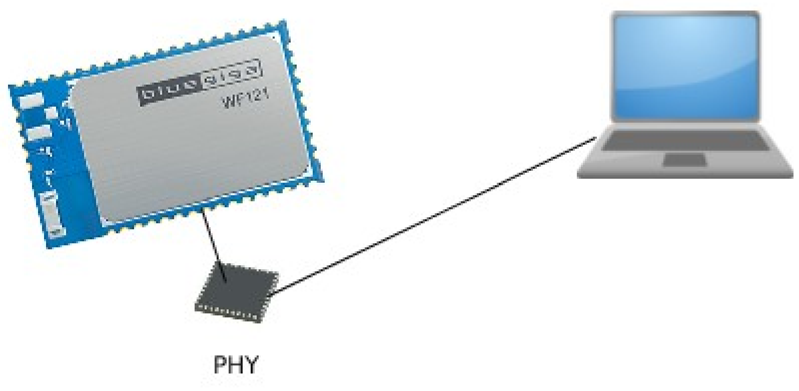


Bridge mode

The Ethernet-to-Wi-Fi Bridge mode can be used to provide Wi-Fi connectivity for devices that only support Ethernet connectivity. The "Bridge" forms a transparent data pipe between the Client ethernet interface and the Wi-Fi network connected to by the WF121, allowing for a bi-directional data stream.

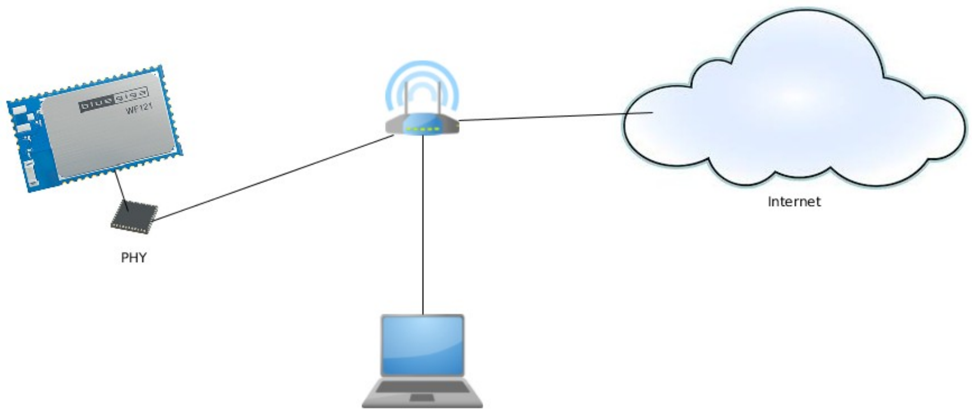
In this example, the setup is made easy by running an HTTP-server on the module, accessible via the Ethernet connection and a web browser. After the Wi-Fi network has been configured the WF121 will act as a Bridge and the Wi-Fi router only sees the end device (i.e., the device connected to WF121 over Ethernet) and will assign an IP address for the end device.

1.2.2 Ethernet Server



Ethernet Server
The Ethernet server mode is designed to enable the configuration of the WF121 Wi-Fi module using internal web pages of the module. Using this setup the module may be setup to act as a Wi-Fi client in bridge mode. Before setting the mode on, the module must first be initialized like a Wi-Fi access point with or without DHCP server usage.

1.2.3 Ethernet Device



Ethernet Device
The Ethernet device mode can be used to connect the WF121 to a network with an Ethernet cable.

2 Implementing Wi-Fi Bridge ("eth" SDK example)

The "eth" example included in the SDK demonstrates how the WF121 module may be configured to act as an Ethernet-to-Wi-Fi bridge between a Wi-Fi network and client device that only supports Ethernet connectivity. This section provides an overview of the "eth" example, breaking down the project into its' individual components.

The Ethernet cable used between WF121 and PC must be of the cross-over type.

2.1 Project Configuration

2.2 Module firmware compiling and installation is covered in WF121 Configuration Guide.

The project configuration is contained within the *project.xml* file. The project configuration for the "eth" example is defined by four tags: <hardware>,<script>, <binary>, and <files>.

```
<?xml version="1.0" encoding="UTF-8" ?>
<project>
    <scripting>
        <script in="wifi.bgs"/>
    </scripting>
    <software>
        <binary in="fw/wifi.juo"/>
    </software>
    <hardware>
        <uart channel="1" baud="5000000" api="true" handshake="true"/>
        <ethernet enable="1"/>
        <sleep interrupts="1"/>
    </hardware>
    <files>
        <file path="index.html"/>
        <file path="style.css"/>
        <file path="bluegiga.png"/>
    </files>
</project>
```

- **<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.
- **<files>** Defines the HTML, CSS and image files used by the embedded HTTP server.

2.2.1 Hardware Configuration

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

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

- **<uart>** This setting configures the UART interface. The UART tag sets the UART2 Interface to use 5000000 bps baud rate and enables hardware flow control. The UART provides access to the BGAPI protocol.
- **<ethernet>** This setting enables the Ethernet (RMII) interface on the WF121 module.
- **<sleep>** This setting enables **INT0** (pin 38) wake-up interrupt.

2.2.2 WF121 Development Kit Physical Hardware Configuration

The following switch positions should be used in regards to the WF121 Development Kit

- Set the Ethernet switch (labeled SW1) to **ON** position
- Set the UART1 switch (labeled SW4) to **OFF** position
- Set the UART2 switch(labeled SW5) to **ON** position
- Set board power switch (labeled SW14) to **ON** position

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

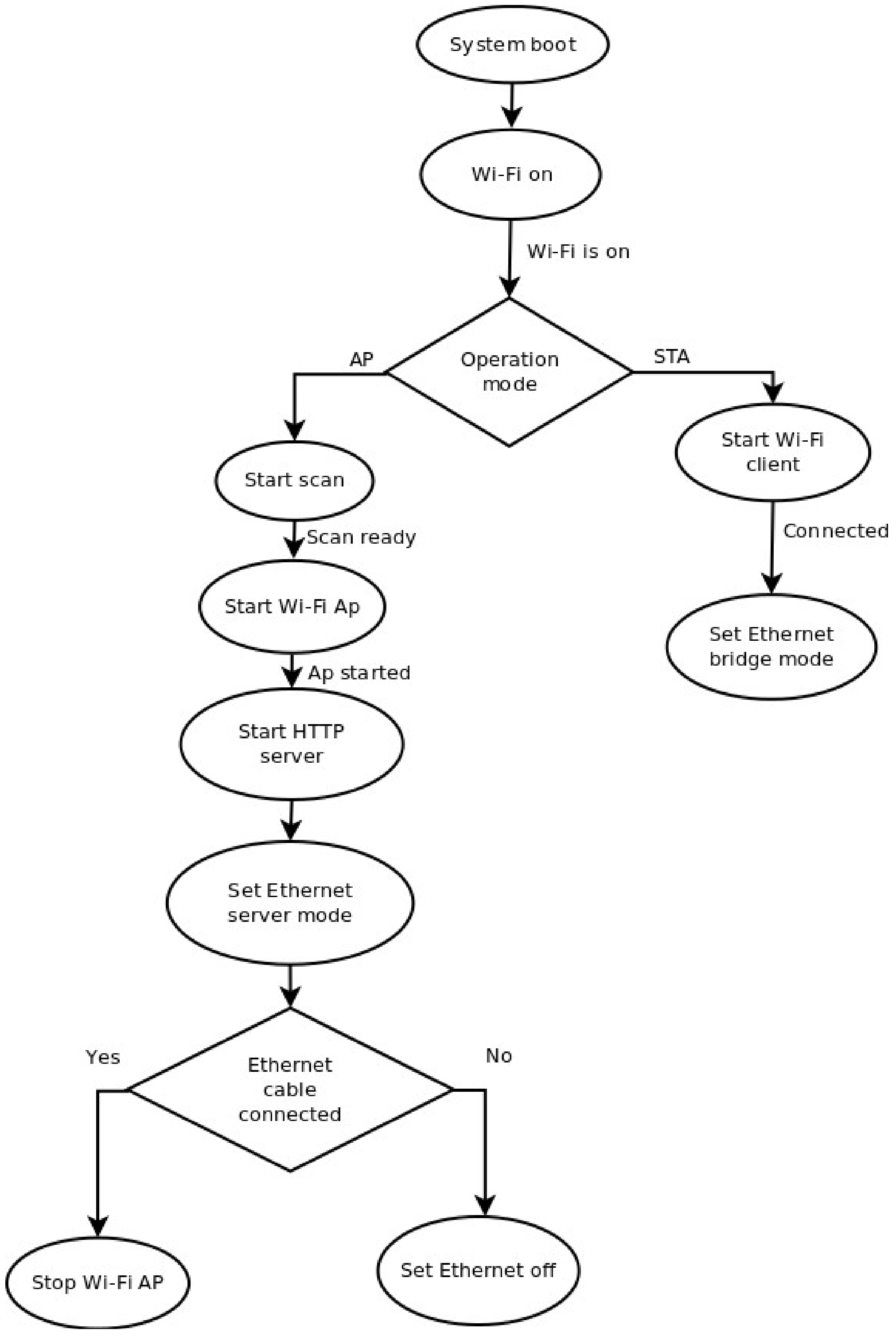
- RD1, RD7, RD8, RD9, RB15, RD[port] and RF0 and RF1.

2.3 BGScript

The BGScript source file, *wifi.bgs,* included with the "eth" example is responsible for implementing the application logic. The BGScript source file, wifi.bgs, includes multiples events and procedures, which together, comprise the BGScript application functionality. The events most closely related to the overall application functionality are covered here in more detail.

Functionality of embedded HTTP server is specified in HTTP Server Example v1.3 document.

Flowchart illustrates the logical flow for a successful operation of the Ethernet-to-Wi-Fi Bridge.



2.3.1 system_boot Event

The *system_boot* event when the module is powered up:

```
#***Module is powered***
event system_boot(major, minor, patch, build, bootloader_version, tcpip_version, hw)
    # Initialise variables
    reconnect_count = 0
    operating_mode = MODE_AP
    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)(cmd_result, cmd_value_len, cmd_value(0:cmd_value_len))
    if cmd_result = 0
        operating_mode = cmd_value(0:cmd_value_len)
    end if
    call sme_set_operating_mode(operating_mode)

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

2.3.2 sme_wifi_is_on Event

The *sme_wifi_is_on* event is received when Wi-Fi hardware is initialized:

```
#**Wi-Fi is on***
event sme_wifi_is_on(state)
    if operating_mode = MODE_STA
        # Start station mode.
        call start_sta_mode()
    else
        #start scan to update scan list shown in HTTP page
        call sme_start_scan(HANDLE_WIFI, 0, 0)
    end if
end
```

2.3.3 sme_scanned Event

The *sme_scanned* event is received when Wi-Fi access point scanning is done:

```
#***Scan compelete***
event sme_scanned(status)
    if operating_mode = MODE_AP
        #ready to start Ap
        call start_ap_mode()
    end if
end
```

2.3.4 sme_ap_mode_started Event

The *sme_ap_mode_started* event is recieved when Wi-Fi access point is initialized:

```
#***Ap started***
event sme_ap_mode_started(hw_interface)
    #configure HTTP server paths
    call https_add_path(PATH_DEVICE_API, PATH_API_LEN, PATH_API(:))
    call https_add_path(PATH_DEVICE_FLASH, PATH_ROOT_LEN, PATH_ROOT(:))

    #enable HTTP and DHCP servers
    call https_enable(1,1,0)
    #enable Ethernet
    call ethernet_set_dataroute(ETH_TO_LOCAL)

    #Test Ethernet cable connected
    call ethernet_connected()(cmd_result)
    if(cmd_result = 0)
        #Ethernet not connected, use Wi-Fi Ap
        call ethernet_set_dataroute(0)
        wifi_on = 1
    else
        #Ethernet connected, run Wi-Fi Ap down
        call sme_stop_ap_mode()
        wifi_on = 0
    end if
    call hardware_io_port_write(LED_PORT, AP_LED, AP_LED)
end
```

2.3.5 sme_connected Event

The sme_connected event is received when module connection to defined Wi-Fi access point is ready:

```
#***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
```

3 Implementing Ethernet device ("eth_device" SDK example)

The goal of this example is to connect to a network via Ethernet cable or via Wi-Fi network. Wired network is the primary connection and the Wi-Fi connections acts as a backup. Before an automatic connection switch is possible, Wi-Fi access point information must be set by using the embedded HTTP server.

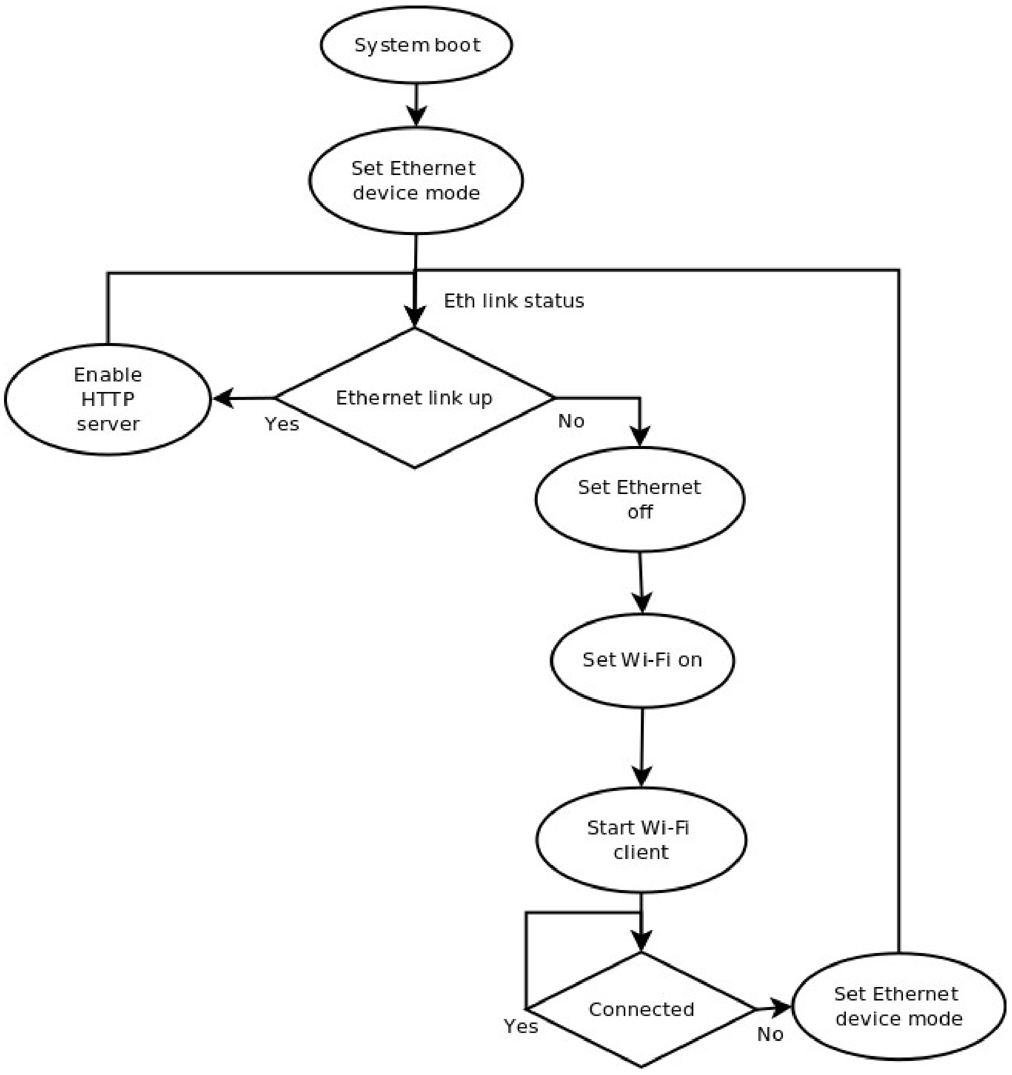
3.1 Creating a project

The process of creating a project and configuring the hardware is similar as in the Bridge mode example.

3.2 BgScript code

Flow chart illustrates the logical flow for linking the WF121 Ethernet device to a network with the Wi-Fi used as the backup connection.

Functionality of the embedded HTTP server is specified in document *HTTP Server Example v1.3* document.



Ethernet Device flow

3.2.1 Catching System boot

System boot event is sent when the module is powered up:

```
#***Module is powered***
event system_boot(major, minor, patch, build, bootloader_version, tcpip_version, hw)
    # Initialise variables
    eth_connected = 0

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

    #try to use Ethernet
    call ethernet_set_dataroute(ETH_DEVICE)
end
```

3.2.2 Catching Ethernet link status change

Ethernet link status event is sent when the link is changed from up to down or vice versa:

```
#***Link status changed ***
event ethernet_link_status(status)
     if status = 1
        eth_connected = 1

        #configure HTTP server paths
        call https_add_path(PATH_DEVICE_API, PATH_API_LEN, PATH_API(:))
        call https_add_path(PATH_DEVICE_FLASH, PATH_ROOT_LEN, PATH_ROOT(:))

        call https_enable(1,0,0)
        call hardware_io_port_write(LED_PORT,ETH_LED,ETH_LED)
    else
        call hardware_io_port_write(LED_PORT,ETH_LED,0)
        if eth_connected = 1
            call ethernet_set_dataroute(ETH_DISABLED)
            call sme_wifi_on()
        end if
        eth_connected = 0
     end if
end
```

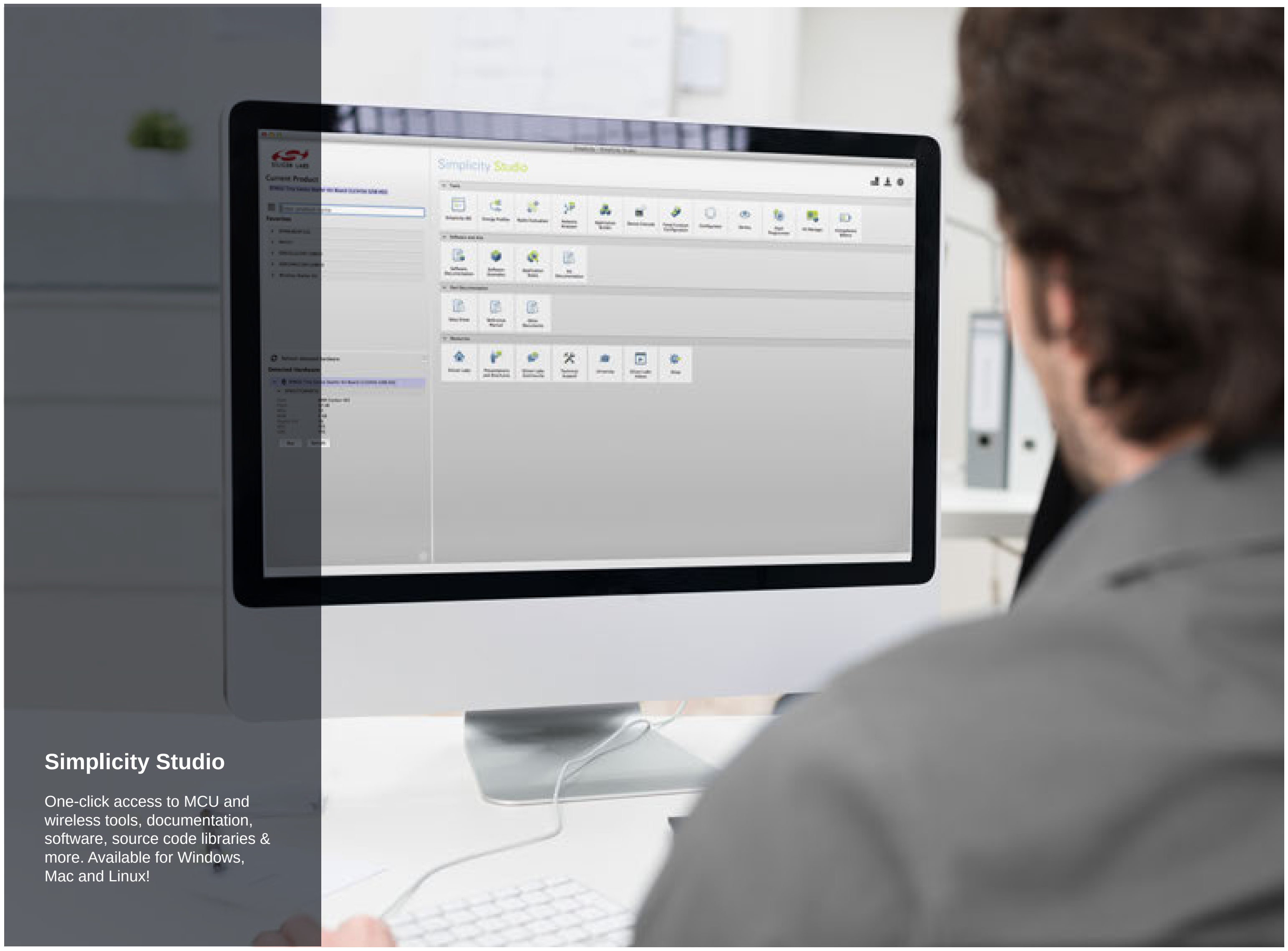
3.2.3 Catching Wi-Fi disconnected and Wi-Fi off

Disconnected event is sent when the Wi-Fi connection is lost:

```
#***Wi-Fi disconnected***
event sme_disconnected(reason, hw_interface)
  if eth_connected = 1
    call sme_wifi_off()
  end if
end
```

Wi-Fi is off event is sent when the Wi-Fi state is changed from initialized to uninitialized:

```
#***Nj.,-Fi is off***
 vent sme_wifi_is_off(state)
    call  hardware_io_port_write(LED_PORT,STA_LED,0)
    call ethernet_set_dataroute(ETH_DEVICE)
 nd
```



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 LABS

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

http://www.silabs.com