

## Chapter 1: Introduction

After completing this chapter, you will understand what this class is, what topics are covered, the overall class objectives, and you will understand what is contained in the ModusToolbox™ for Connectivity and ModusToolbox™ for Wi-Fi run-time software.

### Table of contents

|             |  |           |
|-------------|--|-----------|
| <b>1.1</b>  | <b>What is this class?</b>   | <b>2</b>  |
| <b>1.2</b>  | <b>Prerequisites</b>   | <b>2</b>  |
| <b>1.3</b>  | <b>Required software</b>   | <b>2</b>  |
| <b>1.4</b>  | <b>Development kits</b>  | <b>3</b>  |
| 1.4.1       | PSoC™ 6 kits   | 4         |
| 1.4.2       | Arduino compatible shields   | 6         |
| <b>1.5</b>  | <b>Introduction to ModusToolbox™ for Connectivity and ModusToolbox™ for Wi-Fi run-time software</b>      | <b>7</b>  |
| <b>1.6</b>  | <b>ModusToolbox™ for Connectivity and ModusToolbox™ for Wi-Fi run-time software library descriptions</b> | <b>8</b>  |
| 1.6.1       | Wi-Fi Core FreeRTOS lwIP MbedTLS   | 8         |
| 1.6.2       | HTTP   | 9         |
| 1.6.3       | MQTT   | 9         |
| 1.6.4       | Over-The-Air (OTA) Bootloading   | 10        |
| 1.6.5       | Bluetooth®/Bluetooth® LE Hosted Stack (BTSTACK and FreeRTOS Wrapper)                                     | 10        |
| 1.6.6       | Low Power Assistant (LPA)  | 10        |
| <b>1.7</b>  | <b>Task/Thread Priorities</b>  | <b>10</b> |
| <b>1.8</b>  | <b>Path Lengths</b>  | <b>11</b> |
| <b>1.9</b>  | <b>Documentation</b>   | <b>11</b> |
| <b>1.10</b> | <b>Code examples</b>   | <b>13</b> |
| <b>1.11</b> | <b>Exercises</b>   | <b>15</b> |
|             | Exercise 1: Download Class Material  | 15        |

### Document conventions

| Convention        | Usage  | Example  |
|-------------------|--|--|
| Courier New       | Displays code and text commands  | CY_ISR_PROTO(MyISR);<br>make build                           |
| <i>Italics</i>    | Displays file names and paths  | sourcefile.hex   |
| [bracketed, bold] | Displays keyboard commands in procedures                                       | [Enter] or [Ctrl] [C]  |
| Menu > Selection  | Represents menu paths  | File > New Project > Clone                                   |
| <b>Bold</b>       | Displays GUI commands, menu paths and selections, and icon names in procedures | Click the <b>Debugger</b> icon, and then click <b>Next</b> . |

## 1.1 What is this class?

This is a class to teach how to use Wi-Fi in ModusToolbox™ applications. The descriptions and exercises use a PSoC™ 6 MCU as a host to a CYW43012 device.

After completing this class, you should be able to create and debug Wi-Fi applications using ModusToolbox™ tools. This class will introduce you to several concepts including: the network stack, sockets, TLS, HTTP, MQTT, and interacting with Amazon Web Services (AWS).

## 1.2 Prerequisites

This class assumes that you know the basics of using the ModusToolbox™ ecosystem, how to interact with PSoC™ 6 MCUs including using peripherals, and how to program a PSoC™ 6 device. If you are unfamiliar with these topics, the following classes should be reviewed first:

- ModusToolbox™ Software Training Level 1 – Getting Started
- ModusToolbox™ Software Training Level 2 – PSoC™ MCUs

## 1.3 Required software

- ModusToolbox™ tools

You should already have ModusToolbox™ tools installed on your system. If not, refer to the ModusToolbox™ Software Training Level 1 Getting Started class or visit the [ModusToolbox™ Software website](#) for instructions.

## 1.4 Development kits

For this class there are a few options for which development kits to use. If you want to do all of the exercises except for the final project in chapter 6, you only need one of the following PSoC™ kits:

- CY8CKIT-062S2-43012
- CY8CPROTO-062-4343W
- CY8CPROTO-062S2-43439

If you want to do all of the exercises including the final project in chapter 6, you will need the CY8CKIT-062S2-43012 PSoC™ kit and one of the following shields:

- CY8CKIT-028-SENSE
- CY8CKIT-028-TFT

The CY8CKIT-028-SENSE shield has a small monochrome OLED display, but it can measure the ambient temperature.

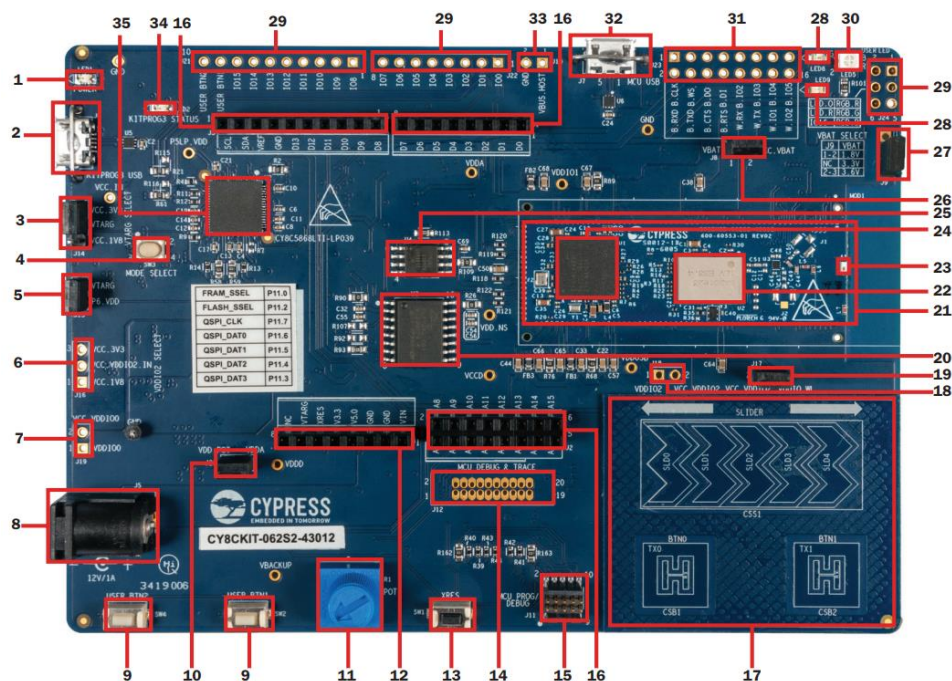
The CY8CKIT-028-TFT shield has a larger color TFT display, but it doesn't have a temperature sensor so the ambient temperature has to be simulated.

*Note: The final project is to create an IOT thermostat that connects to Amazon Web Services.*

## 1.4.1 PSoC™ 6 kits

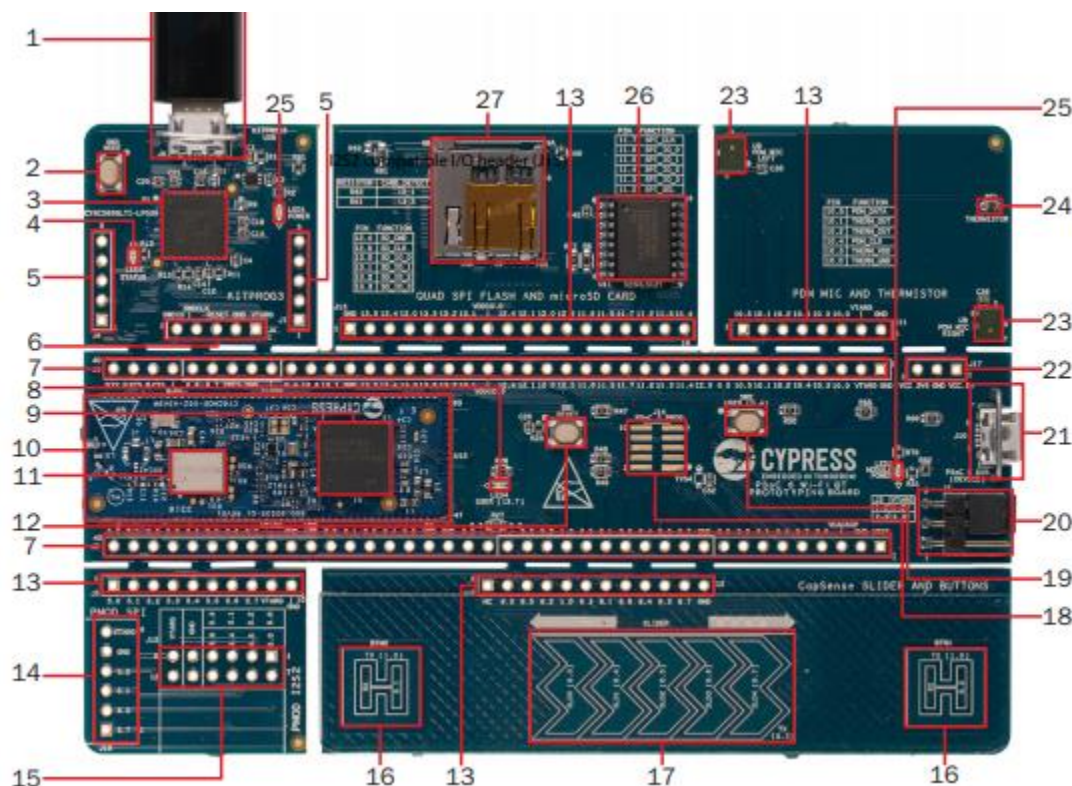
**Note:** If you are taking this course on your own, you will need two PSoC™ 6 kits to complete some of the exercises in later chapters. If you are taking this course in a classroom setting, you will only need one of these kits.

[CY8CKIT-062S2-43012](#) – A PSoC™ 6-2M MCU and a CYW43012 Wi-Fi + Bluetooth® Combo



- |   |  |
|---|--|
| 1. Power LED (LED1)   | 19. CYW43012 VDDIO current measurement jumper (J17)                                    |
| 2. KitProg3 USB connector (J6)  | 20. Infineon serial NOR flash memory (S25FL512S, U3)                                   |
| 3. PSoC™ 6 MCU VDD power selection jumper (J14)                       | 21. Infineon PSoC™ 6 (2M) MCU with CYW43012 Carrier Module (CY8CMOD-062S2-43012, MOD1) |
| 4. KitProg3 programming mode selection button (SW3)                   | 22. CYW43012 based Murata Type 1LV module  |
| 5. PSoC™ 6 MCU VDD current measurement jumper (J15)                   | 23. Wi-Fi/Bluetooth® antenna   |
| 6. PSoC™ 6 MCU VDDIO2 and CYW43012 VDDIO power selection jumper (J16) | 24. PSoC™ 6 MCU  |
| 7. PSoC™ 6 MCU VDDIO0 current measurement jumper (J19)                | 25. Infineon serial Ferroelectric RAM (CY15B104QSN, U4)                                |
| 8. External power supply VIN connector (J5)                           | 26. CYW43012 VBAT current measurement jumper (J8)                                      |
| 9. PSoC™ 6 MCU user buttons (SW2 and SW4)                             | 27. CYW43012 VBAT power selection jumper (J9):   |
| 10. Potentiometer connection jumper (J25)                             | 28. PSoC™ 6 MCU user LEDs (LED8 and LED9)  |
| 11. Potentiometer (R1)  | 29. PSoC™ 6 I/O header (J21, J22, J24)   |
| 12. Arduino-compatible power header (J1)                              | 30. RGB LED (LED5)   |
| 13. PSoC™ 6 MCU reset button (SW1)                                    | 31. Wi-Fi/Bluetooth® GPIO header (J23)   |
| 14. PSoC™ 6 MCU debug and trace header (J12)                          | 32. PSoC™ 6 USB device connector (J7)  |
| 15. PSoC™ 6 MCU program and debug header (J11)                        | 33. Optional USB Host power supply header (J10)  |
| 16. Arduino Uno R3-compatible I/O headers (J2, J3, and J4)            | 34. KitProg3 status LED (LED2)   |
| 17. CAPSENSE™ slider (SLIDER) and buttons (N0 and BTN1)               | 35. KitProg3 (PSoC™ 5LP MCU) programmer and debugger (CY8C5868LTI-LP039, U2)           |
| 18. PSoC™ 6 MCU VDDIO2 current measurement jumper (J18)               | 36. MicroSD Card holder (J20) (on back of board)                                       |

[CY8CPROTO-062-4343W](#) or [CY8CPROTO-062-43439](#) – PSoC™ 6 Wi-Fi BT Prototyping Kit



- |   |   |
|---|---|
| 1. KitProg3 USB connector (J8)  | 14. Digilent® Pmod™ SPI compatible I/O header (J16)           |
| 2. KitProg3 programming mode selection button (SW3)                   | 15. Digilent® Pmod™ I2S2 compatible I/O header (J15)          |
| 3. KitProg3 (PSoC 5LP) programmer and debugger (CY8C5868LT-LP039, U1) | 16. CapSense buttons  |
| 4. KitProg3 status LED (LED2)   | 17. CapSense slider   |
| 5. KitProg3 I/O headers (J6, J7)                                      | 18. PSoC 6 MCU program and debug header (J14)                 |
| 6. KitProg3 5-pin programming header (J4)                             | 19. PSoC 6 MCU user button (SW2)                              |
| 7. PSoC 6 MCU I/O headers (J1, J2)                                    | 20. Power selection jumper (J3)                               |
| 8. PSoC 6 MCU user LED (LED4)   | 21. PSoC 6 USB device Connector (J10)                         |
| 9. PSoC 6 MCU (CY8C624ABZI-D44)                                       | 22. External power supply connector (J17)                     |
| 10. Cypress PSoC 6 Wi-Fi-BT Module (CY8CMOD-062-4343W, U15)           | 23. PDM microphones (U8, U9)                                  |
| 11. CYW4343W based Murata Type 1DX Module (LBEE5KL1DX)                | 24. Thermistor (RT1)  |
| 12. Reset button (SW1)  | 25. Power LEDs (LED1, LED3)                                   |
| 13. On-board peripheral headers (J5, J11, J12 and J13)                | 26. Cypress 512-Mbit serial NOR flash memory (S25HL512T, U11) |
|   | 27. microSD Card holder (J9)                                  |

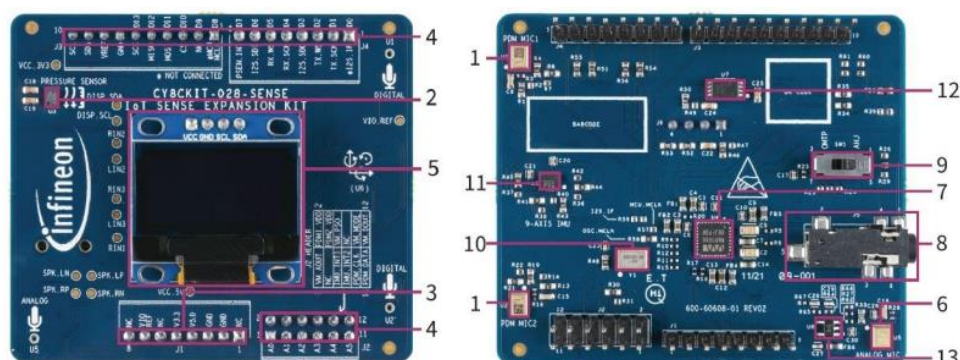
**Note:** The user button shares the same pin on this kit as the Wi-Fi host wake pin. Therefore, if you use both Wi-Fi and the user button, you must disable the host wake feature by adding `CY_WIFI_HOST_WAKE_SW_FORCE=0` to the `DEFINES` variable in the Makefile.

**Note:** While the CY8CPROTO-062-4343W is shown here, the CY8CPROTO-062S2-43439 is identical except for the connectivity device contained in the Wi-Fi/Bluetooth® module.



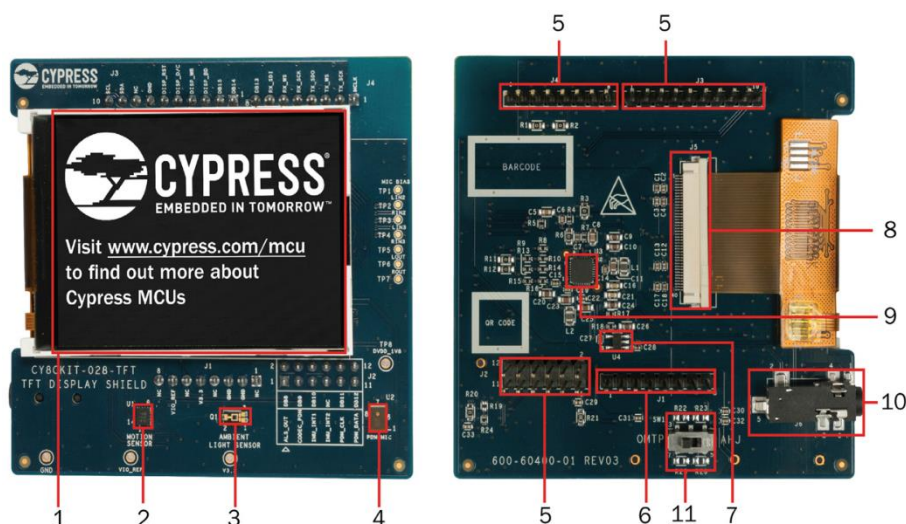
## 1.4.2 Arduino compatible shields

### CY8CKIT-028-SENSE



- |  |   |
|--|---|
| 1. XENSIV™ digital MEMS microphones (U1, U2)           | 8. 3.5-mm stereo audio jack socket (J5)           |
| 2. XENSIV™ digital barometric air pressure sensor (U3) | 9. Audio jack socket type selection switch (SW1): |
| 3. Arduino™ UNO R3 compatible power header (J1)        | 10. Crystal oscillator (Y1)                       |
| 4. Arduino™ UNO R3 compatible I/O headers (J2, J3, J4) | 11. Bosch 9-axis absolute orientation sensor (U6) |
| 5. OLED module (ACC6)                                  | 12. I2C level translator (U7)                     |
| 6. Vesper piezoelectric MEMS analog microphone (U5)    | 13. Preamplifier (U8)                             |
| 7. Audio codec (U4)                                    |   |

### CY8CKIT-028-TFT



- |  |  |
|--|--|
| 1. 2.4-inch TFT display                        | 7. TFT display power control load switch (U4)    |
| 2. Motion Sensor (U1)                          | 8. TFT display connector (J5)                    |
| 3. Ambient Light Sensor (Q1)                   | 9. Audio CODEC (U3)                              |
| 4. PDM microphone (U2)                         | 10. Audio Jack (J6)                              |
| 5. Arduino compatible I/O headers (J2, J3, J4) | 11. Audio Jack Selection (OMTP/AHJ) Switch (SW1) |
| 6. Arduino compatible power header (J1)        |  |

## 1.5 Introduction to ModusToolbox™ for Connectivity and ModusToolbox™ for Wi-Fi run-time software

In addition to the PSoC™ 6, the development kit used in this course has a CYW43012 connectivity device that supports both Wi-Fi and Bluetooth®. Let's face it, most electronic devices these days have some sort of wireless connectivity. In this chapter we will focus on the Wi-Fi piece of the puzzle. The connection between the PSoC™ 6 and the connectivity device is done using SDIO for Wi-Fi (or SPI on some devices) and an HCI UART interface for Bluetooth®. Furthermore, wireless co-existence between Wi-Fi and Bluetooth® is supported so both functions can operate simultaneously.

Infineon has several cloud solutions depending on how you decide to manage your connected devices. The ModusToolbox™ for Connectivity solution was built for customers with their own cloud device management back end, whether hosted on AWS, Google, Azure, AliCloud, or any other cloud infrastructure.

If you are using Amazon Web Services IoT Core for your device management, then you may prefer to use our more customized AWS IoT Core solution. Likewise, if you are using Arm Pelion for your device management, you may prefer our customized Arm Pelion solution. Note that if you are using Amazon's cloud services for storage but not their IoT Core device management, ModusToolbox™ for Connectivity is a great solution for you, as you will see in some of the upcoming exercises.

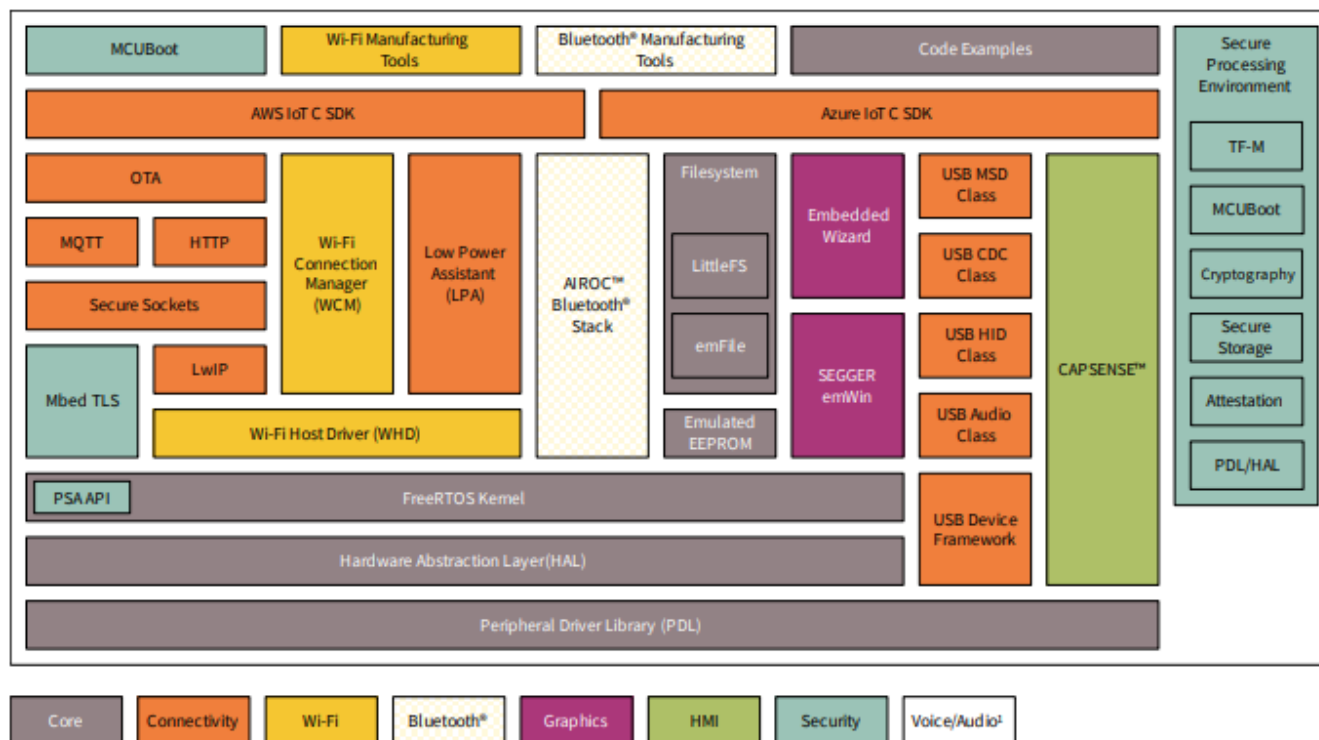
ModusToolbox™ for Wi-Fi provides the low-level features such as the Wi-Fi Connection Manager and Wi-Fi Host Driver while ModusToolbox™ for Connectivity provides a Secure Socket layer and support for application layer cloud protocols such as MQTT and HTTP.

While ModusToolbox™ for Wi-Fi and ModusToolbox™ for Connectivity provide core functionality including connectivity, security, firmware upgrade support, and application layer protocols like MQTT, they are also flexible so you can modify or extend them to match your needs.

## 1.6 ModusToolbox™ for Connectivity and ModusToolbox™ for Wi-Fi run-time software library descriptions

The run-time software is distributed as a collection of libraries that work together to help you easily get your IoT device up and running on the Cloud. Some of the libraries were written by Infineon, while others are industry standard open source libraries. As you have seen, these can be pulled into a ModusToolbox™ application easily by using the Library Manager.

The complete set of run-time software libraries fit together with the core PSoC™ 6 libraries as shown below. The Connectivity and Wi-Fi libraries are shown in orange and yellow, respectively.



Most libraries are available as GitHub repositories. These “repos” contain source files, readme files, and documentation such as an API reference.

The following subsections describe the ModusToolbox™ for Connectivity and ModusToolbox™ for Wi-Fi libraries.

### 1.6.1 Wi-Fi Core FreeRTOS lwIP MbedTLS

This library is a collection of the libraries that any Wi-Fi application will need. By adding this library to an application, you get:

- Wi-Fi Host Driver (WHD) - Embedded Wi-Fi Host Driver that provides a set of APIs to interact with Infineon WLAN chips.
- WHD BSP Integration – Functions to simplify connecting the WHD library to a BSP containing a WLAN chip.



- Wi-Fi Connection Manager - Makes Wi-Fi connections easier and more reliable by implementing Wi-Fi Protected Setup (WPS) to simplify the secure connection to a Wi-Fi access point (AP) and by providing a monitoring service to detect problems and keep connections alive.
- FreeRTOS - FreeRTOS kernel, distributed as standard C source files with configuration header file, for use with the PSoC™ 6 MCU.
- CLib Support - This is a support library that provides the necessary hooks to make C library functions such as malloc and free thread safe. This implementation is specific to FreeRTOS.
- Connectivity Utilities – A collection of general-purpose middleware utilities including JSON parser, linked list functions, string functions, network helper functions, logging functions, and error code utilities. The network helper functions provides functions to perform tasks like converting strings to IP addresses and vice versa. Header files are provided in the library to include the required set of functions. For example, the network helper functions are included via the header file `cy_nw_helper.h` while JSON parsing is included via the header file `cy_json_parser.h`.
- lwIP - A Lightweight open-source TCP/IP stack. lwIP stands for "lightweight IP".
- lwIP FreeRTOS Integration – Contains FreeRTOS dependencies needed by the Lightweight open-source TCP/IP stack
- lwIP Network Interface Integration - An integration layer that links the lwIP network stack with the underlying Wi-Fi host driver (WHD) and Ethernet driver. This functionality is included via the header file `cy_network_mw_core.h`.
- MbedTLS - An open source, portable, easy to use, readable and flexible SSL library, that has cryptographic capabilities.
- MbedTLS Acceleration – Contains MbedTLS hardware accelerated basic cryptography functions.
- RTOS Abstraction Layer - Minimalistic RTOS-agnostic kernel interface allowing middleware to be used in multiple ecosystems, such as Arm's Mbed OS and Amazon's FreeRTOS. It is not recommended for use by the end application - the user should write code for their RTOS of choice such as FreeRTOS.
- Secure Sockets - Network abstraction APIs for underlying lwIP network stack and MbedTLS security library. The secure sockets library eases application development by exposing a socket like interface for both secure and non-secure socket communication.
- WPA3 External Supplicant - Supports WPA3 SAE authentication using HnP (Hunting and Pecking Method) and H2E (Hash to Element Method) using RFC.
- Predefined configuration files for FreeRTOS, lwIP and MbedTLS for typical embedded IoT use-cases.

The Library Manager name for the Wi-Fi Middleware Core library is *wifi-core-freertos-lwip-mbedtls*. It includes the other libraries listed above automatically.

## 1.6.2 HTTP

This is actually two separate libraries – one for HTTP servers and one for HTTP clients. The libraries enable both secure (https) and non-secure (http) modes of connection. They support RESTful HTTP methods: HEAD, GET, PUT, and POST.

The Library Manager names for these libraries are *http-client* and *http-server*.

## 1.6.3 MQTT

This library includes the open source AWS IoT device SDK embedded C library plus some glue to get it to work seamlessly in ModusToolbox™ for Connectivity. It is based on MQTT client v3.1.1 and supports QoS levels 0 and 1. Both secure and non-secure TCP connections can be used.

The Library Manager name for this library is *mqtt*.

### 1.6.4 Over-The-Air (OTA) Bootloading

The OTA toolkit library is an extensible solution based on MCUBoot that can be modified to work with any third-party or custom IoT device management software. With it you can rapidly create efficient and reliable OTA schemes. It currently supports OTA over MQTT and HTTP.

The Library Manager name for this library is *ota-update*.

### 1.6.5 Bluetooth®/Bluetooth® LE Hosted Stack (BTSTACK and FreeRTOS Wrapper)

In addition to the great Wi-Fi support in ModusToolbox™ for Wi-Fi, you can use the ModusToolbox™ for Bluetooth® functionality in the 43xxx combo device to enable Bluetooth® LE for your device. For example, it can easily be used to enable Wi-Fi onboarding so that you can safely and quickly connect your device to a Wi-Fi network using Bluetooth® LE to select the network and enter the password. One of the ModusToolbox™ code examples will show you that exact thing.

In the Library Manager, you can include the *btstack-integration* library which will include the BTSTACK library automatically.

### 1.6.6 Low Power Assistant (LPA)

The LPA is a library and associated settings in the Device Configurator that allow you to configure a PSoC™ 6 Host and WLAN (Wi-Fi / Bluetooth® Radio) device for optimized low-power operation. With LPA you can achieve the most aggressive power budgets by placing the host device into sleep or deep sleep modes while networks are quiet or when there is traffic that can be handled by the connectivity device.

The Library Manager name for this library is *LPA*.

## 1.7 Task/Thread Priorities

The Wi-Fi libraries use an RTOS to accomplish their tasks. You have the option to choose which RTOS to use as it is part of the application configuration. In this class we will use FreeRTOS. Regardless of which RTOS you use, you must make sure that your tasks/threads are not so high in priority that they interfere with Wi-Fi operation. In the case of FreeRTOS, your tasks should use a priority of 3 or lower to ensure that Wi-Fi works properly. A priority of 3 is equivalent to a priority of `CY_THREAD_PRIORITY_NORMAL` from the RTOS abstraction library. If you use a different RTOS, be sure to set your tasks/threads to the equivalent priority or lower.

## 1.8 Path Lengths

Windows has a path length limitation of 260 characters. This isn't usually a problem, but some of the libraries used for IoT functionality have very long paths. Therefore, you may run into a build error if the path to your application is too long. For example, the HTTP client and MQTT libraries include the aws-iot-device-sdk-embedded-C library as a dependency. When that library is included in an application, the build directory will have a file in this location:

```
<path_to_application>\build\CY8CKIT-062S2-43012\Debug\ext\mtb_shared\aws-iot-device-sdk-embedded-C\202103.00\libraries\aws\ota-for-aws-iot-embedded-sdk\source\dependency\3rdparty\tinycbor\src\cborencoder_close_container_checked.d
```


That's 209 characters, so only 51 characters are left for the path to the application (which by default is `C:\Users\<user_name>\mtw\<application_name>`).

If you see build errors indicating that file cannot be found in the build directory, it is likely due to the path length. You can shorten the name of the application or move your workspace to a location with fewer characters in the path (e.g. `C:\mtw`).

## 1.9 Documentation

The best place to find documentation for ModusToolbox™ for Connectivity and ModusToolbox™ for Wi-Fi is inside the individual libraries themselves. You can look on GitHub directly, you can open the documentation from a library that you have downloaded, or you can open the documentation for a downloaded library from inside the Eclipse IDE for ModusToolbox.

For example, here is the *README.md* for the Wi-Fi Host Driver library as seen on GitHub:

 README.md

## Wi-Fi Host Driver (WHD)

### Overview

The WHD is an independent, embedded Wi-Fi Host Driver that provides a set of APIs to interact with Infineon WLAN chips. The WHD is an independent firmware product that is easily portable to any embedded software environment, including popular IoT frameworks such as Mbed OS and Amazon FreeRTOS. Therefore, the WHD includes hooks for RTOS and TCP/IP network abstraction layers.

The [release notes](#) detail the current release. You can also find information about previous versions.

### Supported bus interface

| Interface | 4343W | 43438 | 4373 | 43012 | 43439 | 43907 |
|-----------|-------|-------|------|-------|-------|-------|
| SDIO      | ○     | ○     | ○    | ○     | ○     |       |
| SPI       | ○     | ○     |      |       | ○     |       |
| M2M       |       |       |      |       |       | ○     |

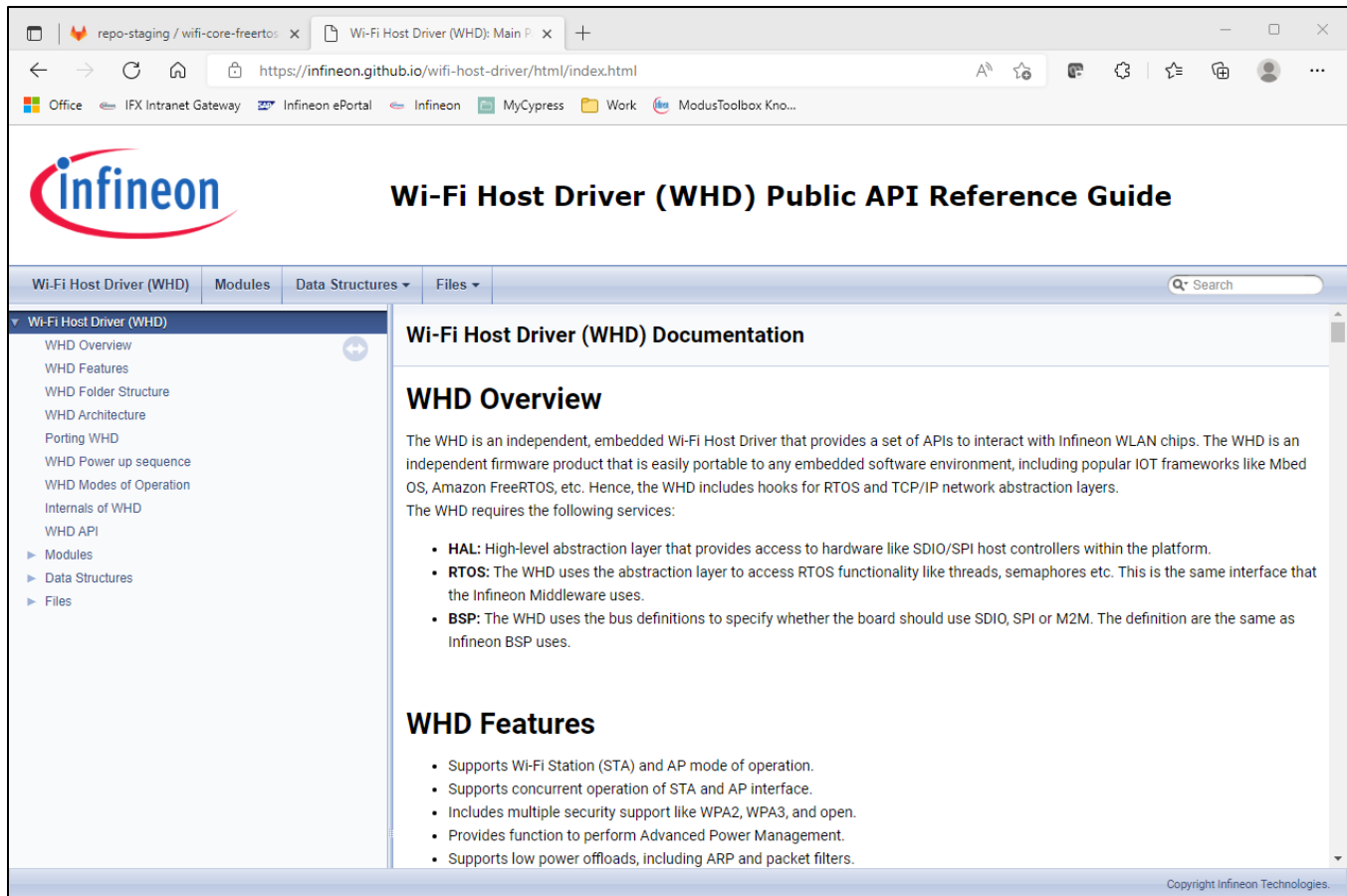
### WPA3 AP mode support

| Security | 4343W | 43438 | 4373 | 43012 | 43439 | 43907 |
|----------|-------|-------|------|-------|-------|-------|
| WPA3     |       |       | ○    |       | ○     |       |

### More information

- [Wi-Fi Host Driver API Reference Manual and Porting Guide](#)
- [Wi-Fi Host Driver Release Notes](#)
- [Infineon Technologies](#)

Once you have downloaded a library, you can go to the *docs* directory or you can open it from the Quick Panel inside Eclipse. In either case, you will typically find an *api\_reference\_manual.html* file with full descriptions of the API.



## 1.10 Code examples

It is always easier to start with an existing application rather than starting with a blank slate. Infineon provides a wealth of code examples for just that reason.

All Infineon code examples are hosted on GitHub so ultimately that's where you will find them, but there are a few ways to get there:

1. Inside the project creator tool look under the Wi-Fi category.
2. Directly from the [Infineon GitHub website](#)
3. From the [Infineon website](#)

The first method is recommended if you want to create an application from the code example since it handles all of the BSP and library downloads automatically.

Of course, you can learn a ton by looking at the source code and reading the comments from the examples. On top of that, every Infineon code example has a README.md file that explains the example, tells you which kits it will run on, shows how to set it up, explains how to use it, and explains the design. The README.md is always a good starting point.



You will need a markdown reader to get the most out of the README.md documents. There are plugins for most web browsers as well as an extension for VS Code. Markdown will also show inside the Eclipse IDE, but some things may be formatted strangely since Eclipse doesn't support everything that Markdown can do.

☰ README.md

## WLAN low power

This code example demonstrates the low-power operation of a host MCU and a WLAN device using the network activity handlers provided by the [low power assistant \(LPA\) middleware](#).

The code example connects to a configured network. After connecting to the network successfully, the example configures the WLAN device in a power-save mode, suspends the network stack, and puts the host MCU in a wait state. During this wait state, the host MCU enters a low-power state and wakes up on any network activity detected on the MAC interface.

[View this README on GitHub.](#)

[Provide feedback on this code example.](#)

## Requirements

- [ModusToolbox™ software](#) v3.0 or later (tested with v3.0)
- Board support package (BSP) minimum required version: 4.0.0
- Programming language: C
- Associated parts: All PSoC™ 6 MCU parts, [AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip](#), [AIROC™ CYW4343W Wi-Fi & Bluetooth® combo chip](#)

## Supported toolchains (make variable 'TOOLCHAIN')

- GNU Arm® embedded compiler v10.3.1 ( `GCC_ARM` ) - Default value of `TOOLCHAIN`
- Arm® compiler v6.16 ( `ARM` )
- IAR C/C++ compiler v9.30.1 ( `IAR` )

## Supported kits (make variable 'TARGET')

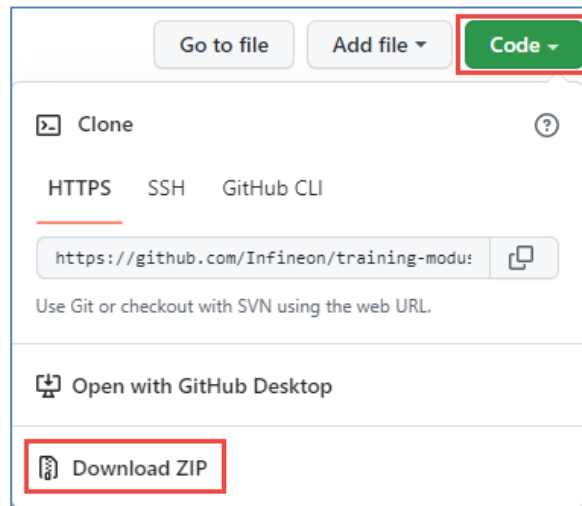
- [PSoC™ 6 Wi-Fi Bluetooth® prototyping kit](#) ( `CY8CPROTO-062-4343W` ) – Default value of `TARGET`

## 1.11 Exercises

### Exercise 1: Download Class Material

In this exercise, you will download the class material from GitHub. This will give you local access to the manuals and projects.

- ☐ 1. Use a Web browser to go to the class GitHub site at: <https://github.com/Infineon/training-modustoolbox-level3-wifi>
- ☐ 2. Click the Code button.



- ☐ 3. Click the Download ZIP button to download the repo to your local disk to a convenient location and then unzip it.

*Note: If you are familiar with Git operations, you can choose to clone the repository to your local disk using the URL instead of downloading a ZIP file.*

#### **Trademarks**

All referenced product or service names and trademarks are the property of their respective owners.

**Published by**  
**Infineon Technologies AG**  
**81726 Munich, Germany**

**© 2024 Infineon Technologies AG.**  
**All Rights Reserved.**

#### **IMPORTANT NOTICE**

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenhheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office ([www.infineon.com](http://www.infineon.com)).

#### **WARNINGS**

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.