Following are the software prerequisites to get started:

1. Windows/Linux PC
2. Wi-Fi Access Point
3. Minicom/TeraTerm VT Utility installed on PC (add a download link)
4. SDK package

### Software Development Kit

#### Download the Software Development Package (SDK)

To download the software development package, execute the following steps:

1. Go to the InnoPhase website (<https://innophaseiot.com/>) and click on Register.

A screenshot of a computer

Description automatically generated

1. Provide the appropriate details to register onto the InnoPhase Customer Portal.

**Note**: InnoPhase requires a signed MNDA and Development Tool License Agreement (DTLA) to be signed prior to granting access to the Customer Portal.

1. On successfully registering to the Customer Portal, the following screen will appear:

A screenshot of a computer

Description automatically generated

1. Navigate to the Software Tab and download the appropriate software package(s):

A screenshot of a computer

Description automatically generated

#### SDK Package Contents Walkthrough

The FreeRTOS SDK software package (sdk\_x.y.zip) provides software APIs with ready-to-run firmware examples to support quick evaluation and development of MQTT/HTTP/HTTPS/AWS and AZURE IoT Cloud applications with Talaria TWO.

Folder details are as follows:

1. **apps**: production ready example applications that the user can run on Talaria TWO. Following applications are available in the release package:

|  |  |
| --- | --- |
| **Protocol** | **Description** |
| alarm | Describes alarm functionality in Talaria TWO |
| helloworld | Basic helloworld application |
| ssbl | Demonstrates running and switching between multiple applications on Talaria TWO using SSBL |
| stw\_multi\_proto | Serial to Wi-Fi application used in hosted mode of operation |
| gordon.elf | Flashing utility using UART interface |

Table 1: Application in the release package

1. **binaries**: The product folder consists of firmware image for AT commands.
2. **bins**: Firmware images which can be used to understand the protocols and functionality supported on Talaria TWO.
3. **build**: Make configuration files. These files include linker scripts and make file rules used for firmware configuration.
4. **components**: Files which serve as common components which all applications can make use of. It acts as a library, where an application like HTTP, SNTP, etc., can link to this library and directly use the function. The following protocols are available in the components folder serving as a library for applications:
   1. alarm
   2. checksum
   3. cipher
   4. device\_ifc
   5. fota
   6. http
   7. json
   8. mdns
   9. mqtt
   10. out
   11. prov
   12. secureboot
   13. sntp
   14. ssl\_wrap
   15. utils
   16. websocket
5. **conf**: Debugger configuration files.
6. **doc**:

The doc folder contains the following sub-folders:

* 1. reference\_guides
     1. api\_reference\_guide: API reference guide describes the programmers APIs.
     2. bootargs\_reference\_guide: Bootargs reference guide describes the use of Talaria TWO boot arguments (bootargs) in different scenarios using different feature and protocols.
     3. cli\_reference\_guide: CLI reference guide serves as a reference guide for Talaria TWO CLI commands, its usage, use case or examples in different scenarios supported by Talaria TWO modules.
  2. user\_guides
     1. ug\_evb\_a: Talaria TWO Evaluation Board an overview of the evaluation board explaining its key features and functions
     2. ug\_eclipse\_setup\_windows: Eclipse setup in Windows describes developing an application using Eclipse and Talaria TWO SDK
     3. ug\_eclipse\_setup\_linux: Eclipse setup in Linux describes the procedure to build and debug an application on Eclipse IDE using Talaria TWO SDK
     4. ug\_env\_setup\_linux: Environment set-up with Talaria TWO for Linux describes setting up the development environment for Talaria TWOTM SDK on an Ubuntu VirtualBox with a Windows 10 host
     5. ug\_wsl: Windows Subsystem for Linux describes developing an application using Windows Subsystem for Linux (WSL) and Talaria TWO SDK
     6. ug\_coredump\_generation\_and\_anaysis: Coredump generation and analysis describes generating a coredump file used for analyzing the cause of the crash.
     7. ug\_debugging\_using\_gdb: Debugging using GDB describes the procedure for debugging the applications using GDB to work with OpenOCD
     8. porting\_guide\_innoos\_to\_freeRTOS: Porting guide describes the procedure to port InnoOS code to FreeRTOS.
     9. ug\_firmware\_sdk: Firmware SDK user guide describes developing applications for the Talaria TWO device.
     10. ug\_memory\_mapping: Memory mapping describes the flash memory mapping of standard applications with details on the flash memory layout.

1. **examples**: Demo/sample applications that the user can run on Talaria TWO firmware. Following examples are available:

|  |  |
| --- | --- |
| **Protocol** | **Description** |
| at\_custom\_cmd | Demonstrates custom AT commands which the user can use apart from the standard commands |
| ble\_beacons | Example codes describing the basic concept of Eddystone Beacon   * Eddystone UID * Eddystone URL * Eddystone TLM |
| ble\_wifi\_bridge | Example code for receiving a text message from a connected BLE client and publishing it to a CloudMQTT broker |
| chip\_monitor | Describes the application for fetching the changes in the values of device core temperature, Voltage of VBAT, external ADC and estimated current consumption of Talaria TWO device |
| crash\_handling | Example code for using the crash handler API to handle and debug error cases |
| gpio | Example codes to use the GPIO interface. the interface for GPIO |
| http\_client | Example codes for using HTTP client APIs to connect to HTTP servers in secured (HTTPS) and non-secured way |
| i2c | Demonstrates usage of I2C on Talaria TWO |
| ifttt | Example application for using Talaria TWO with IFTTT |
| lp\_scan | Demonstrates the basics of the Low Power Wi-Fi scan feature |
| mdns | Demonstrates using the mDNS APIs provided by the mDNS module |
| mqtt | Example codes for using the publish/subscribe operation of MQTT in both secured and non-secured modes |
| prov | A demo Provisioning application using BLE for provisioning AP credentials at Talaria TWO from a mobile application |
| secure\_files | Example application for reading and writing encrypted files from/to the filesystem |
| unassoc | Example codes describing Wi-Fi un-associated mode transmission APIs available in the SDK, call-back events, notifications and associated data structures |
| using\_ble | Introduction to BLE APIs through code samples consisting of a server and client application |
| using\_filesystem | Demonstrates using the filesystem APIs to show case the filesystem functionalities on the Talaria TWO EVK |
| using\_sntp | Demonstrates fetching time from NTP server using SNTP |
| using\_wifi | Example codes describing the Wi-Fi connection manager APIs |
| watchdog\_timer | Demonstrates managing Talaria TWO watchdog timer using the functions provided by the watchdog driver |
| wcm\_pm | Demonstrates the Wi-Fi Connection Manager power management APIs |
| websocket | Demonstrates using the WebSocket client APIs provided by the WebSocket module |

Table 2: Example applications

1. **FreeRTOS-Kernel**: FreeRTOS include files.
2. **include**: SDK include files (.h files) for all applications/examples in the release package.
3. **libs**: SDK library files which can be used by all applications/examples.
4. **pc\_tools**: Programming (Download) tool and Multi-Purpose Demo tool (available for both Windows and Linux platforms).
5. **root\_fs**: Contains the filesystem image.
6. **script**: Helper scripts/utilities used to achieve multiple functionalities like program, read, write and so on.
7. **tools**: Certain tools, supporting files and scripts are used for development on Talaria TWO. This directory further contains the following directories:
   1. fcc\_test – tool used for Wi-Fi/BLE testing and to set up required test parameters.
   2. fletcher32 - tool used to create checksum files for checking the integrity of the configuration files. Currently part.json, boot.json and fota\_config.json files’ integrity is checked using the checksum.
   3. mklittlefs - tool used to create the file system image.
   4. partition\_files - flash partition files used with SSBL setup (ssbl\_part\_table.json) and without SSBL setup (standard\_part\_table.json).