



epSDK Zephyr Agora53

Getting started with the epSDK Zephyr Example

QuickStart Guide

Embedded Planet Inc. | v1.0.0 | 25 SEPT 2026



www.embeddedplanet.com

CONTENTS

1. ABOUT THIS MANUAL	3
2. ACRONYMS & ABBREVIATIONS.....	3
3. INTRODUCTION.....	4
3.1. epSDK ZEPHYR INTRODUCTION	4
3.2. HARDWARE	4
4. BUILD EXAMPLE	5
4.1. SET UP.....	5
4.1.1. Install GITHUB DESKTOP	5
4.1.2. Install IDE.....	5
4.1.3. Install nRF Connect	5
4.1.4. Install Terminal (Optional).....	6
4.2. BUILDING THE PROJECT	7
4.2.1. Load Source Files	7
4.2.2. Configure VS Code Project.....	7
4.3. PROGRAMING	9
4.4. TERMINAL INTERFACE	10
5. epSDK EVALUATION	11
5.1. Sending Data to Cloud	11
5.2. BLE FUOTA	12
6. TROUBLESHOOTING	14
7. UNINSTALL SOFTWARE.....	14
8. DOCUMENT HISTORY	16
9. CONTACT EMBEDDED PLANET	16

1. ABOUT THIS MANUAL

The epSDK Zephyr QuickStart Guide provides details for running the Embedded Planet Agora53 example project. For elaboration on drawings, software, or other specific product details there may be other sources of information to which this document points as reference. For the latest documentation, including document & certification updates, please always refer to the Embedded Planet documentation page: <https://www.embeddedplanet.com/product-documentation>

The firmware for the Agora53 example is located within a public Github repository, [EmbeddedPlanet/epSDK-Zephyr-Agora53 \(github.com\)](https://github.com/EmbeddedPlanet/epSDK-Zephyr-Agora53). The example contains the demo static library file in the *library* folder and the library file headers in the *include* folder. The board specific files are located within the *boards* folder. The project should be initially compiled without any changes.

2. ACRONYMS & ABBREVIATIONS

Term	Description
EP	Embedded Planet, Inc.
epSDK	Embedded Planet Software Development Kit
nRF Connect	Nordic Semiconductor Software Development Kit
Zephyr	Real-time operating system and associated libraries for embedded systems
IDE	Integrated Development Environment
VS Code	Visual Studio Code IDE
UART	Universal Asynchronous Receiver-Transmitter communication port
FUOTA	Firmware Upgrade Over the Air

3. INTRODUCTION

The epSDK Zephyr Agora 53 Example shows users how to integrate the epSDK static library files to send data to the cloud over cellular, run a BLE FUOTA, flash an LED and print messages on the debug UART port. The public archive used in this example demonstrates a small portion of epSDK available for licensing. The QuickStart guide will go over the necessary steps from reviewing the EP product box contents to compiling and running the example.

3.1. epSDK ZEPHYR INTRODUCTION

The epSDK for Zephyr provides library files for sensors and communication interfaces, significantly reducing development time and testing costs. The epSDK integration with a user's application is through simple user-friendly API calls.

The intended use of the epSDK is with EP supplied hardware and Zephyr OS running on a Nordic nRF5340 microcontroller. Users can configure the communication interfaces and drivers through compiler definitions that get passed to the libraries. Sensors, Utilities and Communication Interfaces are contained within the epSDK. Users develop their individual applications but are provided with examples to get started. Individual drivers and communication interfaces are portable to other hardware using Zephyr on an nRF5340 microcontroller. Contact EP to discuss the full epSDK and your application.

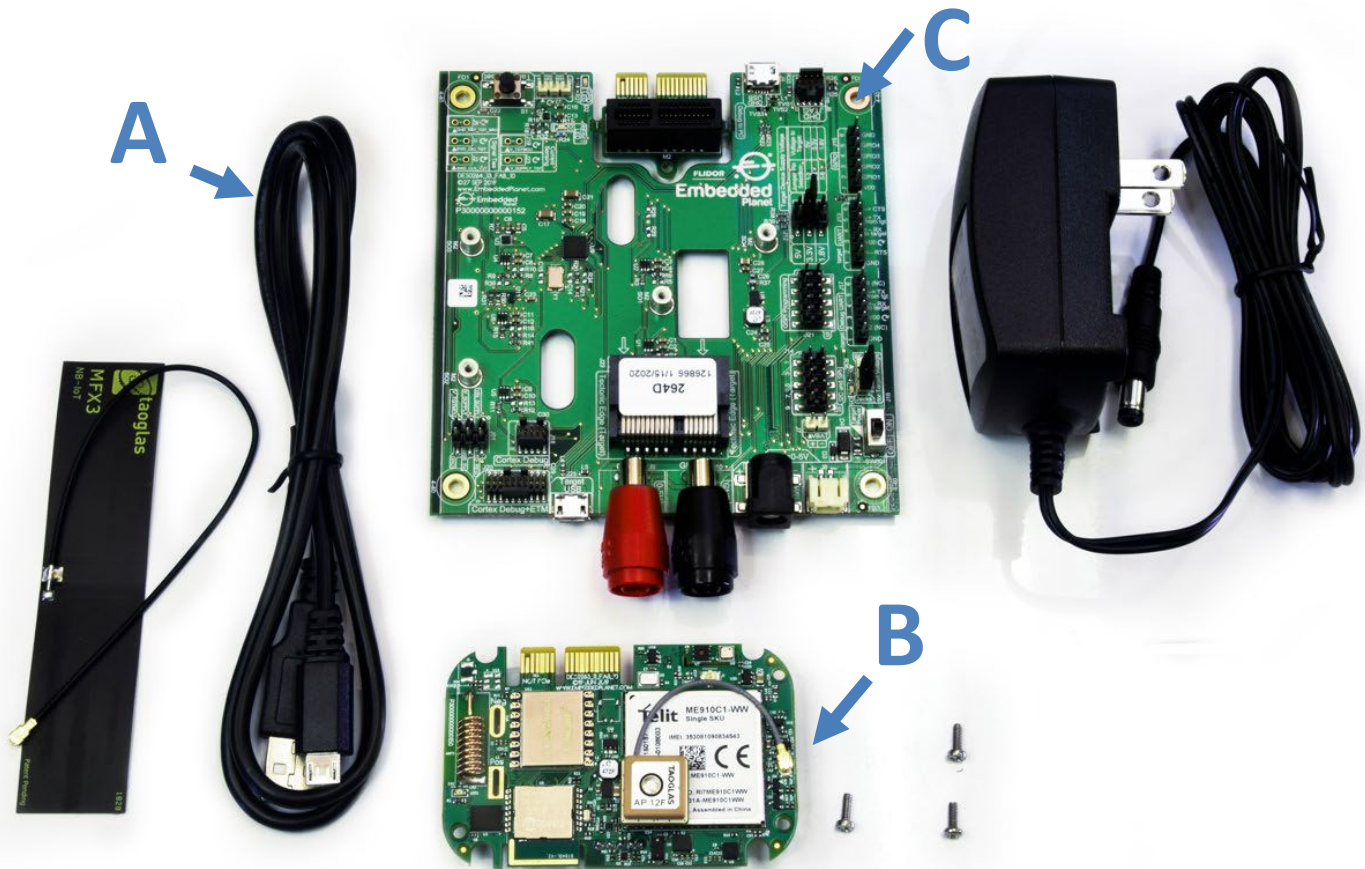
App	User Created				
Comm Interfaces	BLE	HTTP over Cellular (CatM1/NB-IoT)	I2C	SPI	QSPI
	Humidity	Temperature	Pressure	Gas	GNSS
Sensors	Time of Flight	6-axis IMU			
	Crypto	Compact Payload	LED		
General					
Bootloader	mcuBoot				
OS	Zephyr OS				
Vendor Drivers	nRF Connect SDK				
Hardware	NRF5340				

epSDK for Zephyr

3.2. HARDWARE

The blinky example is intended for use with an EP Agora53 device connected to an EP Flidor carrier board. The kit can be purchased through the EP website: [Agora53 Product Development Kit | Embedded Planet](#).

The kit includes the following:



The items A, B, and C will be required for this example. Item A is the micro-USB cable. Item B is the Agora53. Item C is the Flidor programming and debugger board.

4. BUILD EXAMPLE

The following details setting up the compiler, compiling the example, and running the example on the target hardware.

4.1. SET UP

Users must have nRF Connect tools, GitHub Desktop repository tools, and an IDE for building the example. The build instructions are dependent on the PC operating system; the following details describe working on a Windows machine. Linux and MAC machines require minor adjustments.

4.1.1. INSTALL GITHUB DESKTOP

Download the latest Github Desktop version: [Download GitHub Desktop | GitHub Desktop](#)

Execute the downloaded file and follow the prompts for a default installation.

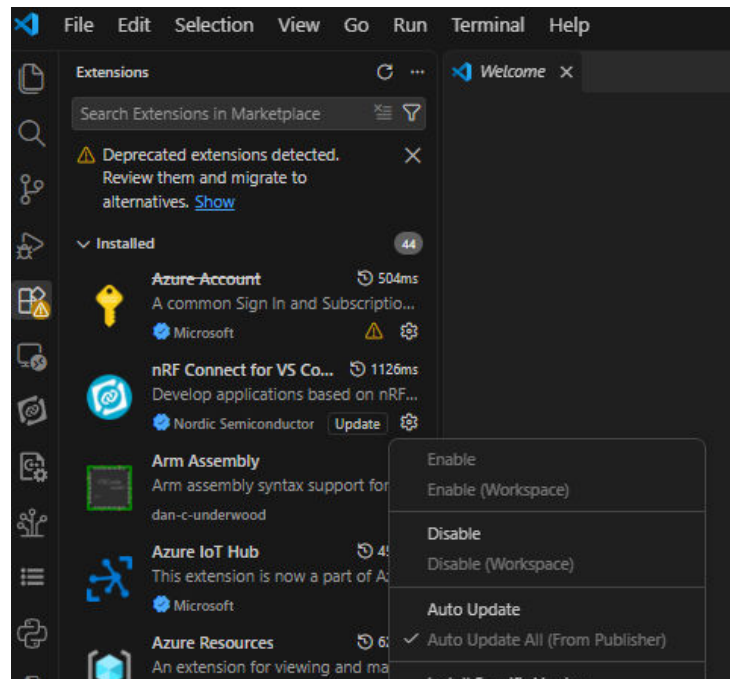
4.1.2. INSTALL IDE

Download the latest version of Visual Studio Code: <https://code.visualstudio.com/download>

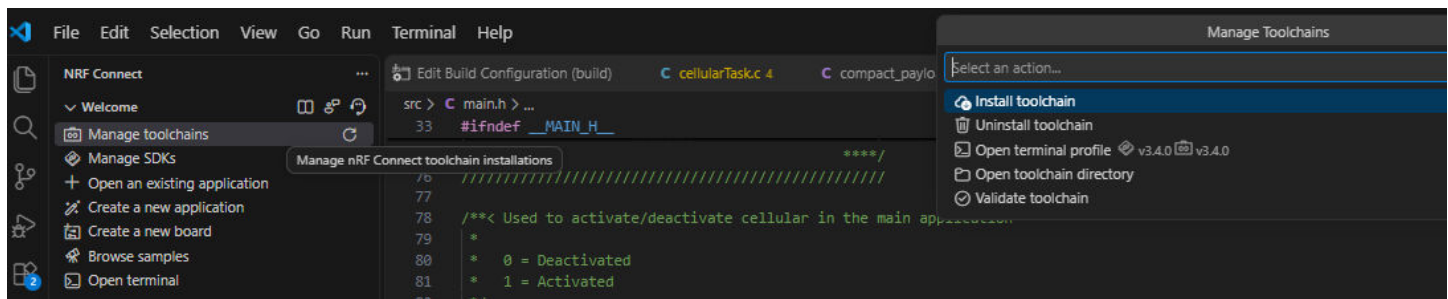
Execute the downloaded file and follow the prompts for a default installation with deviations below.

4.1.3. INSTALL NRF CONNECT

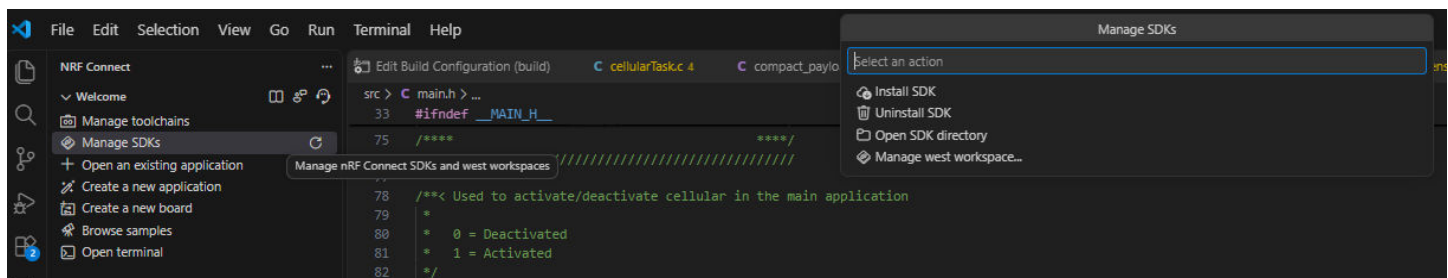
Download version v2026.7.2128 of the nRF Connect for VS Code Extension within Visual Studio Code. Ensure *Auto Update* is not active by selecting the settings button, as there have been compatibility issues with newer versions.



Open the nRF Connect Extension and select *Manage toolchains*, followed by *Install toolchain* to install v3.4.0.



Open the nRF Connect Extension and select *Manage SDKs*, followed by *Install SDK* to install v3.4.0.



4.1.4. INSTALL TERMINAL (OPTIONAL)

A serial port monitor can be used to view debug data from the processor. The instructions here are for using Tera Term.

Download the latest version of Tera Term: <https://github.com/TeraTermProject/teraterm/releases>

Execute the downloaded file and follow the prompts for a default installation.

4.2. BUILDING THE PROJECT

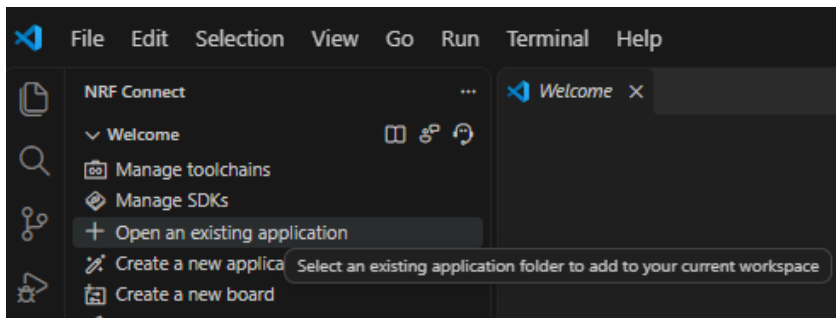
The following steps load the source files to your machine and create a programming file.

4.2.1. LOAD SOURCE FILES

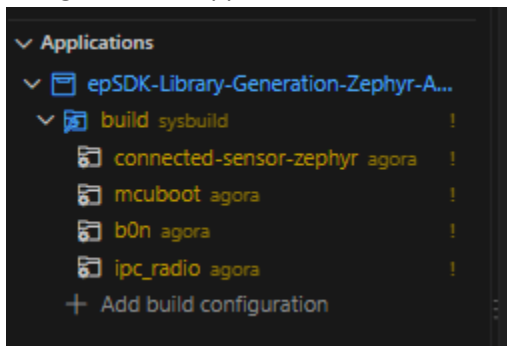
1. Open GitHub Desktop and select *Clone a repository*
2. Navigate to the URL for the desired epSDK Zephyr GitHub repository and set the *Local path* to the location provided on the local computer
3. Select *Clone* to copy the project folder and files to your local computer

4.2.2. CONFIGURE VS CODE PROJECT

1. Open VS Code followed by the nRF Connect Extension
2. Select *Open an Existing application*



3. Navigate to the *Cloned* repository location on the local machine and open the folder
4. Navigate to the *Application* section on the left side, hover over select the “...” for *build* to edit the build configuration



5. Ensure the configuration matches the screen capture below. Select *Generate and Build* to start the compilation

Edit Build Configuration (build)

SDK ⓘ ⚠

nRF Connect SDK v3.4.0 LTS c:\ncs\v3.4.0

Toolchain ⓘ

nRF Connect SDK Toolchain v3.4.0 c:\ncs\toolchains\dc366a1

Board target ⓘ ⚠ Revision ⓘ

agora/nrf5340/cpuapp Revision

Compatible Nordic SoC Nordic Kits Custom All

Base configuration files (Kconfig fragments) ⓘ

Select... Browse

Extra Kconfig fragments ⓘ

Select... Browse

Base Devicetree overlays ⓘ

Select... Browse

Extra Devicetree overlays ⓘ

Select... Browse

Snippets ⓘ

Select...

Optimization level (size, speed, or debugging) ⓘ

Use project default

☒ Include debug thread information ⓘ

Extra CMake arguments ⓘ

-Dconnected-sensor-zephyr_DEBUG_THREAD_INFO=On Remove

Add Argument

☐ Generate only ⓘ

System build (sysbuild) ⓘ

Build system default Use sysbuild No sysbuild

Generate and Build

6. The example will go through the compilation process. The first compilation may take some time to initialize and complete. After successful completion, the memory regions are displayed along with the hex and bin files names.

```

Memory region      Used Size  Region Size  %age Used
  FLASH:           39200 B      48 KB      79.75%
   RAM:           285288 B     440 KB     63.32%
  IDT_LIST:          0 B       32 KB      0.00%

Generating files from C:/Users/golobm/Documents/GitHub/epSDK-Zephyr-Agora53-Pvt/build/mcu
[34/47] C:\WINDOWS\system32\cmd.exe /C "cd /D C:\Users\golobm\Documents\GitHub\epSDK-Zeph
[35/47] Creating signature of application
[36/47] No install step for 'mcuboot'
[37/47] Creating data to be provisioned to the Bootloader, storing to net_provision.hex
[38/47] Completed 'mcuboot'
[39/47] Creating validation for zephyr.hex, storing to signed_by_b0_ipc_radio.hex
[41/47] Generating ../b0n_provision_merged.hex
[43/47] Generating ../merged_agora_nrf5340_cpuapp.hex
[44/47] Generating ../signed_by_mcuboot_and_b0_ipc_radio.bin
image.py: sign the payload
[45/47] Generating ../signed_by_mcuboot_and_b0_ipc_radio.hex
image.py: sign the payload
[46/47] Generating ../dfu_application.zip
[47/47] Generating ../merged_agora_nrf5340_cpunet.hex
* Terminal will be reused by tasks, press any key to close it.

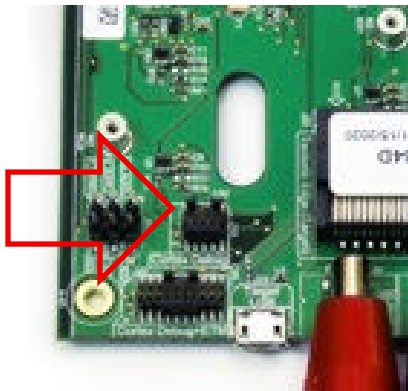
```

7. The build files including the net core hex, app core hex, and app core DFU zip files are created in the build folder:
../build

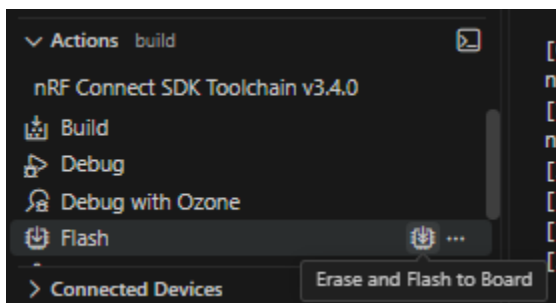
4.3. PROGRAMING

Once the build files are generated, it is used to update the firmware on the Agora53. The *merged_agora_nrf5340_cpuapp.hex* is used for updating the app core and the *merged_agora_nrf5340_cpunet.hex* is used for updating the net core.

1. Connect a [J-Link programmer](#) and [ARM-JTAG 20-10 adapter](#) to the Flidor. The location on the Flidor for plugging in the adapter cable is below:



2. Select the programming button in the *Actions* window to program the application and network cores of the NRF5340



- If multiple programming devices are connected to a computer, user need to select which programmer to use

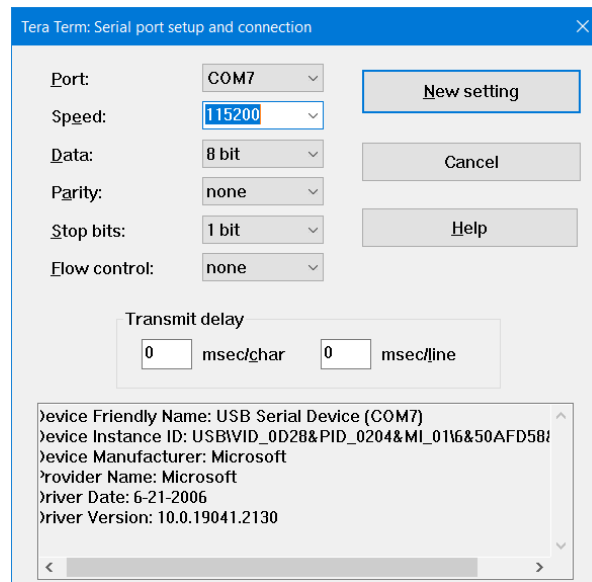


- The VS Code terminal will display a *Flashed Successfully* message for each core. Upon completion, a *Terminal will be used by task..* message is displayed

```
Parsing image file.
Verifying programming.
Verified OK.
Applying pin reset.
-- runners.nrfjprog: Board(s) with serial number(s) 50101497 flashed successfully.
* Terminal will be reused by tasks, press any key to close it.
```

4.4. TERMINAL INTERFACE

- Debugging data is available through the USB port by using a terminal program (Tera Term).
- When opening the program for the first time select “Serial” and the COM port attached to the Flidor (typically the only selection available).
- Select “Setup”, then “Serial Port”. Change the “Speed” to “115200”, then press “New setting”.



- The setup is then saved by clicking on “Setup”, “Save setup”, and following the prompts for overwriting the existing INI file.
- Application information is then displayed on the screen.

```
Tera Term - [disconnected] VT
File Edit Setup Control Window Help
*** Booting nRF Connect SDK v3.4.0-99553055607b ***
*** Using Zephyr OS v4.4.0-bf801e4e3d19 ***

-----
*      Agora53 epSDK      *
*      U0.0.1             *
-----

I: HW Platform: Nordic Semiconductor <0x0002>
I: HW Variant: nRF53x <0x0003>
I: Firmware: Standard Bluetooth controller <0x00> Version 200.12506 Build 888207
I: HCI transport: IPC
I: Identity: E9:C1:A1:04:F1:BC <random>
I: HCI: version 6.3 <0x11> revision 0x2039, manufacturer 0x0059
I: LMP: version 6.3 <0x11> subver 0x2039
I: Bluetooth peripheral initialized

I: Advertising successfully started
I: Waiting for IMEI to be received
```

- The application can be restarted by pressing the reset button on the Flidor.



5. epSDK EVALUATION

This project has a minimal version of the epSDK to send data to the cloud over cellular and run a FUOTA over BLE. The epSDK exposes the headers of the underlying libraries. The headers provide detailed information for each function use and the tagging required for the “mouseover” text descriptions while coding.

```
nrfx_err_t set_time(uint32_t seconds)
Sets the system time initial timestamp

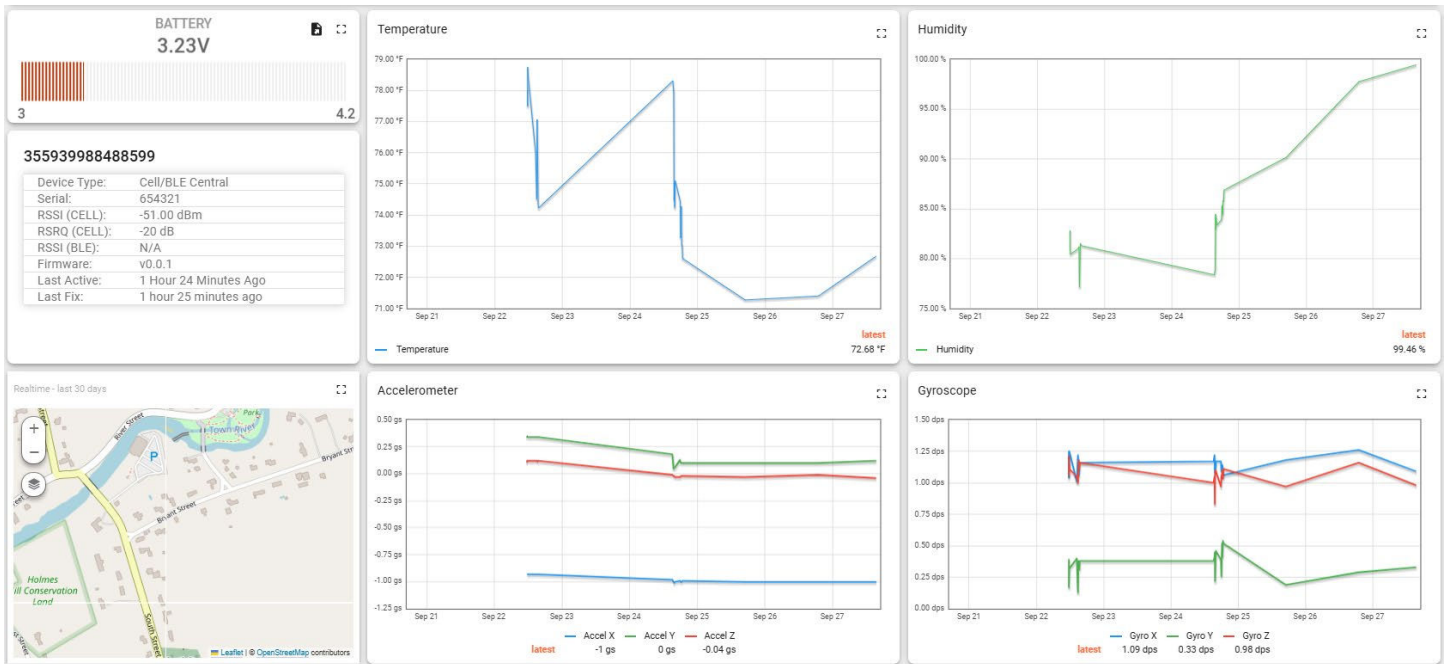
Parameters:
seconds – Can be anything, but should be synced with a time source to the current Unix epoch. Time in seconds elapsed since January 1, 1970. Often initialized to 0 prior to syncing to track time between start and sync.

Returns:
nrfx_err_t 1 for failure to create semaphore. For all others refer to nrfx_err_t codes

set_time(0);
```

5.1. Sending Data to Cloud

This project sends onboard sensor data for the GNSS, 6-axis IMU, and environmental sensors to the cloud. User shall receive a Thingsboard link upon purchase for viewing data.

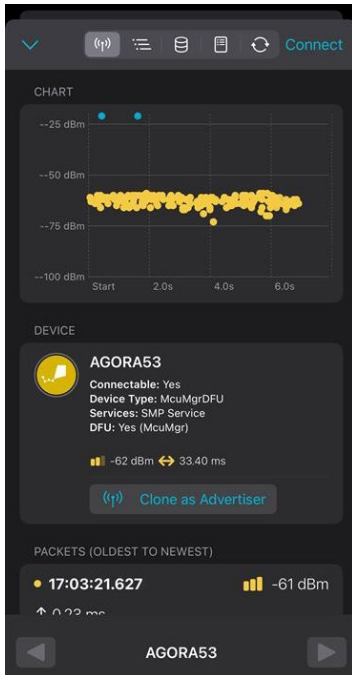


5.2. BLE FUOTA

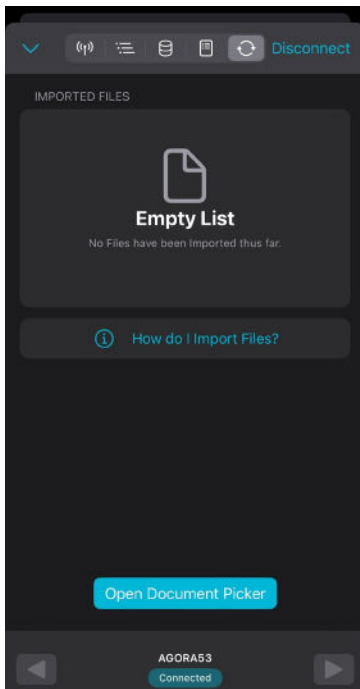
This project can run a BLE DFU FUOTA to upgrade the application core of the NRF5340. The DFU zip file is created as part of the build process. The file is located at `/build/dfu_application.zip`. Users must download the NRF Connect application on their mobile device. The following process was executed on an Apple iPhone, the process has been verified on an Android phone as well. The zip file must be copied onto the mobile device. The DFU will be active on the Agora53 for the first 60s of boot. If additional time is needed, the user shall reboot the device. Once the zip file is loaded on a mobile device, the following process can be used to run the DFU update process.

1. Open the NRF Connect application on a mobile device
2. Reboot the Agora53 device

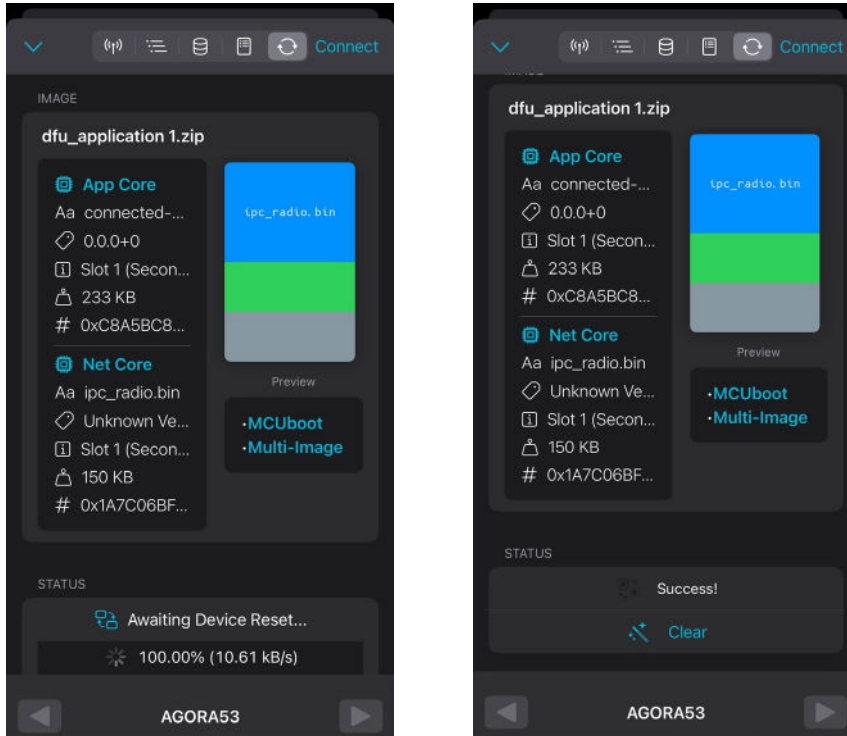
3. Within 30s connect to the device within the application. The device will be named *Agora53*. Select the *Connect* button to connect with the device



4. Navigate to the *file* tab within the application. Select *Open Document Picker* and select the zip file load onto mobile device



5. Application will start the upgrade process. Once complete, the Agora53 will be commanded to reboot



6. TROUBLESHOOTING

Third party software used for this QuickStart guide is updated at intervals beyond our control. If you find software does not operate as described in this manual, please contact EP for assistance.

This guide was written and tested using the following software versions:

GitHub Desktop: version 3.6.6

Visual Studio Code: version 1.139.1

Tera Term: version 4.102

nRF Connect Extension: v2026.7.2128

nRF Connect Toolchain: v3.4.0

nRF Connect SDK: v3.4.0

7. UNINSTALL SOFTWARE

Some of the software used in this Quickstart guide has unique removal processes documented here. Before uninstalling close all programs and power shell windows.

Removing GitHub Desktop:

Open “Add or Remove Programs”: type “add or remove” in the Windows search bar and select “Add or Remove Programs”.

From the list select “GitHub Dekstop”.

Select “Uninstall”.

Follow the questions to remove the program.

Removing IDE:

The Visual Studio Code is removed using the normal Windows removal technique.

Open “Add or Remove Programs”: type “add or remove” in the Windows search bar and select “Add or Remove Programs”.

From the list select “Microsoft Visual Studio Code (User)”.

Select “Uninstall”.

Follow the questions to remove the program.

Removing Terminal:

Tera Term is removed using the normal Windows removal technique.

Open “Add or Remove Programs”: type “add or remove” in the Windows search bar and select “Add or Remove Programs”.

From the list select “Tera Term version x.xx”.

Select “Uninstall”.

Follow the questions to remove the program.

8. DOCUMENT HISTORY

Revision	Author	Description	Date
1.0.0	M. Golob	Initial Release	25 Sep 2026

9. CONTACT EMBEDDED PLANET

Embedded Planet

31225 Bainbridge Rd, Suite N

Solon, OH 44139

Phone: 216.245.4180

Fax: 216.292.0561

www.embeddedplanet.com

Sales: sales@embeddedplanet.com

Information Requests: info@embeddedplanet.com

Technical Support: support@embeddedplanet.com