

**LR1110 Transceiver
Geolocation Demo
User Guide**

Table of Contents

List of Figures	4
1 Introduction	5
2 LR1110 Geolocation Demo Overview	6
3 LR1110 EVK Software Package.....	8
4 Python Host Application Installation	9
4.1 Prerequisites	9
4.1.1 Ensure You Can Run Python.....	9
4.1.2 Ensure You Can Run Pip.....	9
4.2 Installing The Software.....	9
4.3 Checking The Installation	10
4.3.1 The Programs Are Installed And Executable.....	10
4.3.2 The Programs Can Communicate With The LR1110 Dev Kit.....	10
4.4 Troubleshooting.....	11
4.4.1 Issues With Python	11
4.4.2 Issues With Pip.....	12
4.4.3 Issues With The Installed Software.....	13
4.4.4 Issues With Usbconnectioncheck	13
4.5 Addendum.....	15
4.5.1 Path Environment Variable.....	15
4.6 Additional Resources.....	15
5 LR1110 Geolocation Demo Execution.....	16
5.1 LR1110 Information Screen.....	17
5.2 Wi-Fi Passive Scanning	18
5.3 GNSS Scanning	21
6 Python Application.....	24
6.1 Python Application Execution	24
7 Geolocation Results Visualization.....	25
8 Almanac Update Process	28
8.1 Introduction	28
8.2 Obtaining The Latest Almanac	28
8.3 Updating The Almanac In The EVK.....	29
9 Field Fests	30

9.1	Field Test PC Host Application.....	30
9.1.1	Execution Process	30
9.1.2	Jobs File.....	31
9.2	Post Processing Application	32
10	Revision History	33

List of Figures

Figure 1: LR1110 EVK and Demo Start Screen	6
Figure 2: LR1110 Geolocation Demo Architecture	7
Figure 3: LR1110 EVK Software Package	8
Figure 4: Wi-Fi Scanning Results With Reversed Geocoding	19
Figure 5: Wi-Fi Scanning Results.....	19
Figure 6: Wi-Fi Scanning Configuration Menu	20
Figure 7: GNSS Autonomous and Assisted Scanning Results.....	21
Figure 8: GNSS Assisted Scanning Results- GPS and BeiDou Detected Satellites	22
Figure 9: GNSS Autonomous (left) and Assisted (right) Configuration Screen	22
Figure 10: Sniffing Diode.....	23
Figure 11: Wi-Fi GLS (left) and GNSS DAS (right) Services in LoRa Cloud	24
Figure 12: Results.kml File	25
Figure 13: Geolocation Results Visualization Map	26
Figure 14: Wi-Fi Geolocation Results with Metadata	27
Figure 15: GNSS Geolocation Results with Metadata.....	27
Figure 16: Almanacs Need Update Warning.....	28
Figure 17: Field Test Results Folder	30
Figure 18: Field Test Jobs.json Structure	31

1 Introduction

LR1110 is a long range, ultra-low power device aimed to enhance LoRa®-based localization applications. In addition to Wi-Fi and GNSS geolocation capabilities, it supports LoRa® and (G)FSK modulations, and is fully compatible with previous generations of LoRa® radios. It is able to transmit either up to +22dBm on the High-Power PA, or up to +15dBm on the Low Power PA path, and supports a continuous low power operation in the 150MHz-960MHz ISM bands.

A demo application allows demonstrating the geolocation capabilities of the LR1110. This application note aims at describing the different features of this demo for the Transceiver Use Case of the LR1110. It also provides a quick guide in order to help the user to setup for this demo.

The next section of this document shows the global architecture of the LR1110 geolocation demo, introducing its different constituents: the LR1110 Evaluation Kit and the Python host application suite. Section 4 describes the installation process of the Python host application, as a prerequisite for the geolocation demo execution. It can be skipped once the installation process is complete. This section also provides a troubleshooting guide in case of issues during the installation.

Sections 5, to 8 describe in details the different execution steps and the functionalities of the LR1110 geolocation demo, on both the Evaluation Kit and the Python host application levels. Section 9 provides guidance on how to use the LR1110 geolocation demo to perform geolocation tests in the field.

It is recommended to read this User Guide in conjunction with the following documents:

- LR1110 Datasheet
- AN1200.56 LR1110 Evaluation Kit

2 LR1110 Geolocation Demo Overview

The LR1110 geolocation demo runs on the LR1110 evaluation kit hardware (EVK), with the TFT display plugged. It requires the EVK binary execution file `lr1110_evk_vx.y.z.bin` to be loaded in the STM32 Nucleo board.

The GNSS and the Wi-Fi antennas have both to be plugged on the appropriate SMA headers. The demo focuses on demonstrating GNSS and Wi-Fi geolocation. The LoRa® link is unused, therefore the LoRa® antenna is not necessary.

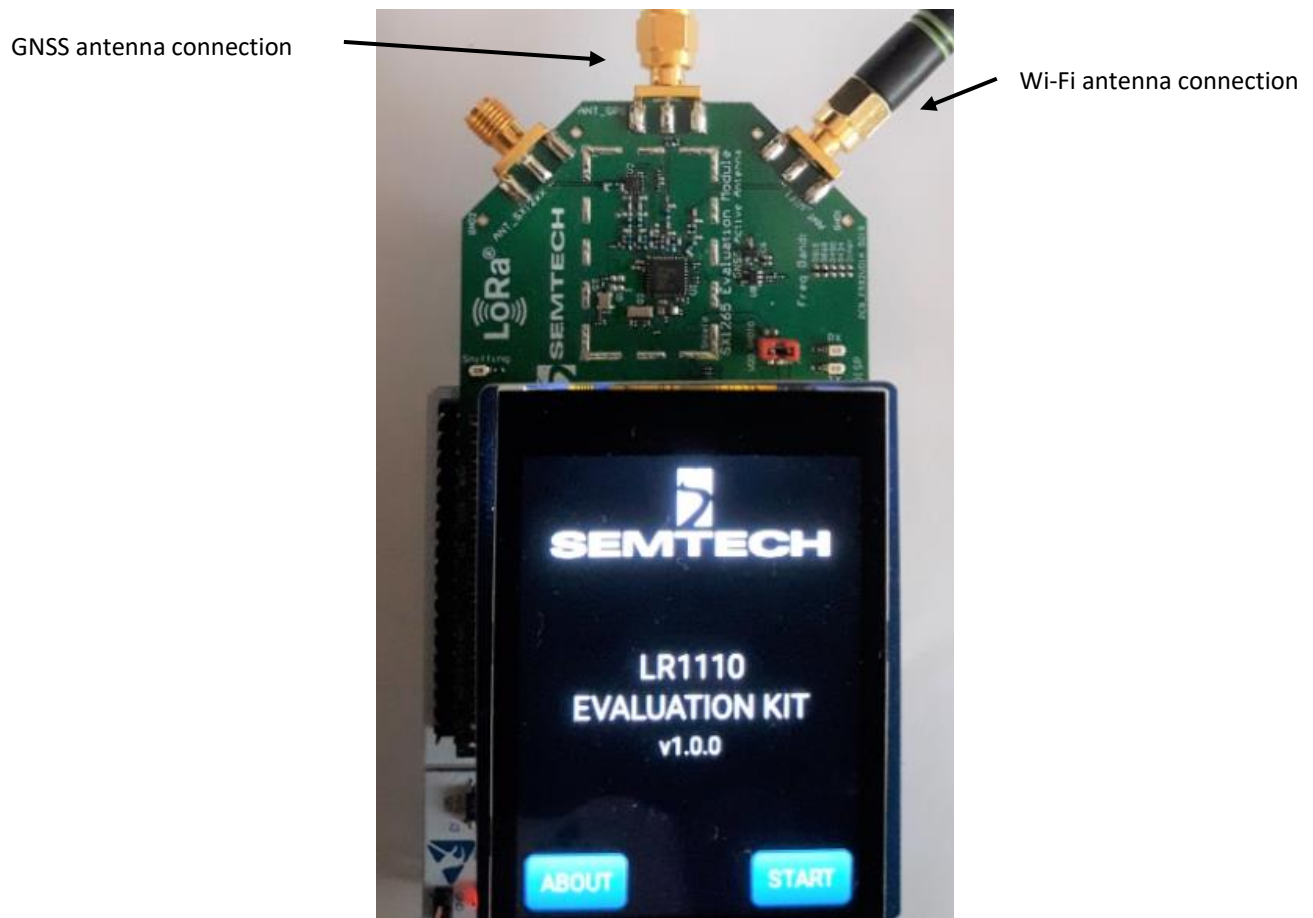


Figure 1: LR1110 EVK and Demo Start Screen

The LR1110 geolocation requires the geolocation server to calculate the device geolocation based on the Wi-Fi or GNSS scanning results. A reverse geocoding is also implemented, in order to display the address of the calculated geolocation on the LR1110 evaluation kit display.

The overall LR1110 geolocation demo architecture is shown in the Figure 2: LR1110 Geolocation Demo Architecture.

The communication with the geolocation server is ensured by a Python host application running on a PC. The communication between the LR1110 evaluation kit and the PC is a serial communication over the USB. The connectivity with the geolocation servers can also be enabled thanks to a LoRa or LoRaWAN® link in the final application.

An optional KML output allows to display the geolocation results on a map for both the Wi-Fi and GNSS geolocations.

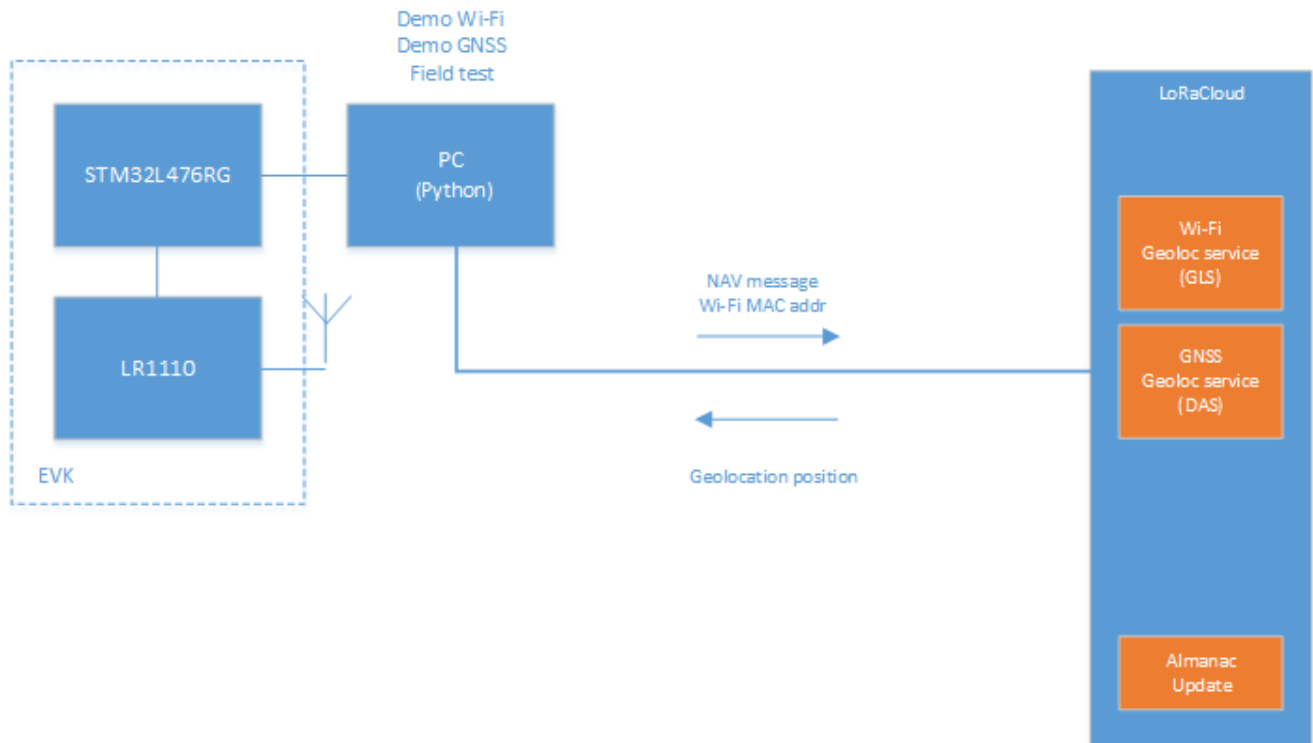


Figure 2: LR1110 Geolocation Demo Architecture

3 LR1110 EVK Software Package

The LR1110 EVK software package and the EVK execution file can be downloaded for the Semtech LR1110 webpage, at the following location:

www.semtech.com/products/wireless-rf/lora-transceivers/lr1110

The archive contains the following files, as shown in Figure 3: LR1110 EVK Software Package.

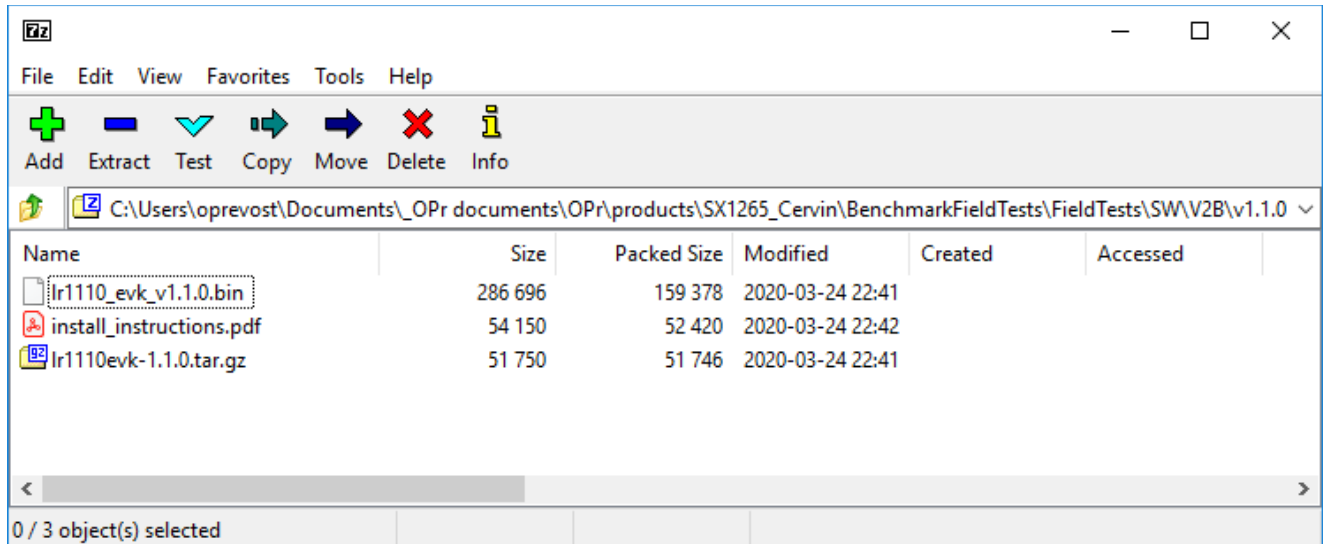


Figure 3: LR1110 EVK Software Package

- lr1110_evk_vx.y.z.bin is the EVK binary execution file. It has to be loaded in the STM32 Nucleo board.
- lr1110-evk_vx.y.z.tar.gz is the Python Host Application installation package. Please refer to section 4 Python Host Application Installation for a detailed installation process.

Once installed, the Python Host application package contains the following Python applications, whose usage is described in the following sections of this document:

- AlmanacUpdate : update the LR1110 Almanac data
- FieldTestPost : send the LR1110 field test results to the geolocation server for post-processing
- Lr1110Demo : LR1110 geolocation demo
- Lr1110FieldTest : LR1110 field test application
- UsbConnectionCheck : troubleshooting tool to check the USB connection between the Python host application and the EVK

4 Python Host Application Installation

This section describes the installation process of the Python host application. This section can be skipped once the Python host application is installed.

4.1 Prerequisites

The LR1110 EVK Python Host applications are mainly written in Python. It requires Python to be installed on the machine that will execute the code. The installation process described here should not require any elevated privileges or super user privileges on Windows. Python can be installed in the `AppData` folder of a regular user.

4.1.1 Ensure You Can Run Python

Open a command line and type:

```
python --version
```

You MUST have at least Python 3.5 to use this software. If it is the case, proceed directly with the section `Ensure You Can Run Pip`.

If it is not the case, there are two possibilities:

1. You get an error on previous command stating you don't have Python at all.
 - a. you actually never installed Python: Refer to section 4.4.1.1 Python Is Not Installed
 - b. you did install Python but it is not recognize: Refer to section 4.4.1.2 Python Is Installed But Not Found
2. You have a Python version that is older than 3.5: Refer to section 4.4.1.3 Python Is Too Old

4.1.2 Ensure You Can Run Pip

The `pip` program is a package installer for Python. To validate you have `pip` installed with the correct version, start a shell and enter the following:

```
pip --version
```

It must returns a version number, and its location and the Python version it is referring to. Ensure that the Python version corresponds to the one you plan to use (at least Python 3.5), and that it corresponds to the Python folder you plan to use.

Depending on the execution of the Python installation process, the user might need to replace the `pip` command by `python3 -m pip`, such as below:

```
python3 -m pip --version
```

If you get an error at this stage, refer to section 4.4.2 Issues With Pip

4.2 Installing The Software

NOTE At this point, it is considered that you have a correct Python installation.

The installation process described here supposes that you have downloaded the source distribution package of the host software (typically named `lr1110evk-<version_number>.tar.gz`).

In a shell, navigate in the folder that contains the source distribution file and type:

```
pip install lr1110evk-<version_number>.tar.gz
```

This command must not fail. If it does fail, ensure the pip program is correctly installed: refer to the previous section, or check that your `PATH` environment variable contains the folder where the pip.exe is installed.

4.3 Checking The Installation

4.3.1 The Programs Are Installed And Executable

The source distribution package of the host software `lr1110evk-<version_number>.tar.gz` does actually install several projects. To validate the installation, the user can just ensure that one of them is correctly installed, for example the `Lr1110Demo` application.

To do so, type in a shell:

```
Lr1110Demo --version
```

It must return a version number. If it is not the case, refer to section 4.4.3 Issues With The Installed Software.

4.3.2 The Programs Can Communicate With The LR1110 Dev Kit

Here, it is assumed that the command `Lr1110Demo --version` did return the correct version number.

The purpose of this section is to use the dedicated host software `UsbConnectionCheck` (that has been installed in the `Lr1110-evk` Python package). This dedicated tool implements a light exchange that validates the communication between the Python host application and a Nucleo L476RG board.

4.3.2.1 Using The Usbconnectioncheck Software

Only a Nucleo L476RG board is required. The LR1110 shield is not required, nor the touch screen.

First of all, ensure the communication host software is correctly installed by typing the following in a shell:

```
UsbConnectionCheck --version
```

The above must return the same version number as for `Lr1110Demo --version`.

Then, plug the Nucleo board to your computer using a USB cable, open a shell and type the following:

```
UsbConnectionCheck
```

This command starts the communication check program (it will run forever, you can hit `Ctrl-C` to stop it).

The exchanges that should occur between host and embedded are the following:

1. the embedded code is sending periodically (almost every second) a token on the USB port
2. the host software waits for this message and sends back a response (it prints `Got message from embedded. Sending question...` on the screen indicating it received the token)
3. the embedded code gets the response and sends back another message to the host software

4. the host software gets this new message and prints `Got response from embedded. Communication ok.` to indicate that the exchange ran successfully

All the messages are exchanged at 921600 baud.

The trace on the shell must be:

```
$ UsbConnectionCheck
```

```
Started
```

```
Waiting for embedded to send command...
```

```
Got message from embedded. Sending question...
```

```
Got response from embedded. Communication ok.
```

```
Got message from embedded. Sending question...
```

```
Got response from embedded. Communication ok.
```

```
Got message from embedded. Sending question...
```

```
Got response from embedded. Communication ok.
```

```
...
```

If you don't see the above trace or get an error, go to section 4.4.4 Issues With Usbconnectioncheck.

4.4 Troubleshooting

4.4.1 Issues With Python

4.4.1.1 Python Is Not Installed

The following [article](#) provides a good step by step install process for Python. Hereafter is an extract of the main steps.

If you are running Windows:

- Navigate to [the Python Download page](#)
- Download the installer corresponding to Windows
- Execute it (it should not require any elevated privileges)
- Restart the PC.

IMPORTANT NOTICE If you are running Windows, don't forget to check the box to update the PATH environment variable. Refer to this section 4.5.1 Path Environment Variable

4.4.1.2 Python Is Installed But Not Found

If you have Python installed but it is not recognized when entering `python --version`, then it is probably because the above important notice has not been followed.

To fix it, you first need to locate your current Python install folder. Search for the file `python.exe`, which is probably located under `C:\Users\\AppData\Local\Programs\Python\Python<version>\`.

Then ensure that it corresponds to Python version at least 3.5. To do so, either trust the folder path that may display a folder like `PythonX.Y/`, or start a shell, navigate to the folder containing the `python.exe` file, and execute `python --version`.

If you figure out the Python version is older than 3.5, then proceed to a complete new Python installation following section 4.4.1.1 Python Is Not Installed. Your actual Python will not be erased.

If the Python version you have is of version at least 3.5, then you simply have to update the `PATH` environment variable. Let's say that the absolute path of your `python.exe` file is `C:\wonderfull\path\python.exe`. Then you have to add the two following folders to the `PATH` environment variable:

- `C:\wonderfull\path\`
- `C:\wonderfull\path\Scripts\`

4.4.1.3 Python Is Too Old

In this case, you can follow section 4.4.1.1 Python Is Not Installed. It will not remove the actual install of Python.

However, you must ensure that, after the install, the command `python --version` actually calls the newer install (and not the older one). To do so, simply start a new shell (yes, you must open the shell **AFTER** the end of the installation) and type

```
python --version
```

If the Python version is the one you just installed, you're good to go. Otherwise, you will have to modify the `PATH` environment variable so that the new `python.exe` is found but not the older one.

To do so, simply remove from the `PATH` environment variable the folder corresponding to the old Python path, and ensure the folder of the new one is indeed in the `PATH` variable.

After fixing the `PATH`, you must start a new shell and try the command `python --version` again. It must return the new version of Python.

The former contains the `python.exe`, and the later will contain all command line programs that will be installed by this project.

4.4.2 Issues With Pip

4.4.2.1 Pip Is Not Installed

If you have Python version at least 3.4, the tool [ensurepip](#) can be used to bootstrap pip. Running the following command in a shell to use it and install pip (the argument `--default-pip` is here to install the pip script, otherwise it install only `pipX` and `pipX.Y`):

```
python -m ensurepip --user --default-pip
```

If that still don't work, you have to download the [get-pip.py script](#) and execute it by

```
python get-pip.py
```

Now, typing `pip --version` in a new shell must give you the correct information.

4.4.2.2 Unknown Option Version

If this command fails with an error similar to:

```
$ pip --version
Unknown option: version
```

It is probably that the `pip` command is not the one from Python, but from Perl. To validate that point, run the following in a shell:

```
where pip
```

It should return a path similar to `... \python\python<version>\Scripts\pip.exe`. But in the case of the incorrect `pip`, it may return a path like `... \perl\bin\pip.bat`.

To overcome this issue, you can try an alternate name of Python's `pip` program:

```
pip3 --version
```

If the above returns the correct version number associated with the correct Python installation, you can keep following this guide by replacing all calls to `pip` program by the alternate name `pip3`.

4.4.3 Issues With The Installed Software

4.4.3.1 Command Not Found

If you get an error indicating that the command `Lr1110Demo` is unknown, you probably have an issue with the `PATH` environment variable.

To figure it out, try to find the entry point of the `Lr1110Demo` software in the Python distribution folder. It probably is under `C:\Users\<username>\AppData\Local\Programs\Python\Python<version>\Scripts\Lr1110Demo.exe`.

If you do find the `Lr1110Demo.exe` at that location, then your `PATH` is probably badly configured. Modify it to include the folder `C:\Users\<username>\AppData\Local\Programs\Python\Python<version>\Scripts\`.

For more details about the environment variable and how to change it, you can have a look on

<https://superuser.com/questions/284342/what-are-path-and-other-environment-variables-and-how-can-i-set-or-use-them>.

If the file is not here, there are several possible problems. Either the install process failed, or you have another Python distribution installed, and the software got installed in another place.

In the later, you have to ensure you install the software for the correct Python version, with the correct `pip` program.

4.4.4 Issues With Usbconnectioncheck

4.4.4.1 Issue: No Port Connected

If you get a Python exception trace like:

```
$ UsbConnectionCheck
Traceback (most recent call last):
  File ".../UsbConnectionCheck", line 8, in <module>
    sys.exit(entry_point_connection_tester())
  File ".../SerialHandlerConnectionTest.py", line 65, in entry_point_connection_te
```

```
ster
  launch_test()
  File ".../SerialHandlerConnectionTest.py", line 41, in launch_test
    port = serial_handler.discover(SerialHandler.DISCOVER_PORT_REGEX)
  File ".../SerialHandler.py", line 94, in discover
    regexp)
LR1110FieldTest.SerialExchange.SerialHandler.SerialHandlerExceptionNoPorts: No port
connected that could be STM32 (used regexp '(STM.*)|(374B)')
```

It means that there is no Nucleo board detected. It may not be connected. If it is indeed connected, try with another Nucleo board.

4.4.4.2 Issue: Too Many Ports

If you get a trace like:

```
$ UsbConnectionCheck
Traceback (most recent call last):
  File ".../UsbConnectionCheck", line 8, in <module>
    sys.exit(entry_point_connection_tester())
  File ".../SerialHandlerConnectionTest.py", line 65, in entry_point_connection_te
ster
  launch_test()
  File ".../SerialHandlerConnectionTest.py", line 41, in launch_test
    port = serial_handler.discover(SerialHandler.DISCOVER_PORT_REGEX)
  File ".../SerialHandler.py", line 97, in discover
    regexp, ports)
LR1110FieldTest.SerialExchange.SerialHandler.SerialHandlerExceptionTooManyPorts: T
oo many ports could be the STM32 (used regexp '(STM.*)|(374B)'): [..., ...]
```

then you may possibly have several Nucleo board connected. Disconnect all but one, and relaunch the UsbConnectionCheck tool.

4.4.4.3 Issue: Never Received Message

If the trace stays stuck to the following:

```
$ UsbConnectionCheck
Started
Waiting for embedded to send command...
...
```

then it means that either the embedded is broken (firmware is not executing), or the link is broken. This may also happen if there are several Nucleo Board connected.

Check that the COM port receives messages from the embedded by opening a Terminal Emulator software on the Nucleo COM port configured with 921600 baud. Check that the word !TEST_HOST is received almost every second.

4.4.4.4 Issue Embedded -> Host

If the trace looks like:

```
Unexpected data received from embedded: '<something>'. Communication KO!
```

where <something> is any set of characters, then the channel from embedded to host is corrupted, or not on the correct baud.

Check that there is no other tool or shell listening on the Nucleo COM port. The baud is a hard configuration on the system so it should not be an issue.

4.4.4.5 Issue Host -> Embedded

If the trace looks like:

```
$ UsbConnectionCheck
Started
Waiting for embedded to send command...
Got message from embedded. Sending question...
Got message from embedded. Sending question...
Got message from embedded. Sending question...
...
```

where there is never the line `Got response from embedded`. Communication ok, it suggests that the embedded never get the response from host (that one that is sent on step 2.).

In this case check the configuration of the COM port. Ensure the Nucleo board has not been modified to disconnect the Serial 2 from the ST-Link.

4.5 Addendum

4.5.1 Path Environment Variable

Any software that is installed on a system can be accessed from a shell by entering its name (and not its complete absolute path) if the folder that contains it is referenced in the `PATH` environment variable.

Therefore, when you type `Python --version` in a shell, the shell search for a file called `python` (or `python.exe`) in all folders referenced by `PATH` environment variable.

In this project there are several programs to consider. First, there is the Python interpreter. Second is the set of command line programs that are installed by this project (`LR1110Demo`, `LR1110FieldTest`, and al.). These are called *entry points* (it is the equivalent of `main` function in C). All these programs must reside in a folder that is in the `PATH` environment variable in order to be accessible from command line.

The `python.exe` interpreter resides in a folder usually located in `C:\Users\\AppData\Local\Programs\Python\Python<version>\`. The *entry points* reside usually in `C:\Users\\AppData\Local\Programs\Python\Python<version>\Scripts`.

Therefore, a Python-based software that is installed in the `Scripts\` folder is accessible from the shell only if the `PATH` environment variable contains the folder.

If at some point you figure out that an installed Python software is not accessible from command line, consider checking the value of the `PATH` environment variable.

4.6 Additional Resources

- <https://docs.python.org/3.7/using/windows.html>
- <https://packaging.python.org/tutorials/installing-packages/>

5 LR1110 Geolocation Demo Execution

The button “START” of the startup screen opens the LR1110 EVK main menu, giving the choice between a “Radio test modes” application, and the LR1110 Geolocation demo (“demonstrator” menu), as shown in Figure 4: LR1110 EVK and Geolocation Demo Main Menu.

The LR1110 geolocation demo runs 3 modes: Wi-Fi Scan (a.k.a Wi-Fi Passive Scanning), and 2 GNSS modes for outdoor geolocation: GNSS autonomous and GNSS assisted.

Please note that the two GNSS modes require the Python host application to be running to be active (“GO!” button in blue). Otherwise, the two “GO!” buttons for the GNSS autonomous and GNSS assisted are greyed out.



Figure 4: LR1110 EVK and Geolocation Demo Main Menu

5.1 LR1110 Information Screen

The button “ABOUT” allows to display the LR1110 Geolocation demo information screen (ABOUT screen), as shows in Figure 5: Demo ABOUT Screen hereafter.



Figure 5: Demo ABOUT Screen

The HW and SW versions of the various components of the demo are as follows:

- LR1110 EVK: version of the EVK firmware, running on the STM32L476 MCU of the Nucleo board.
- LR1110 TYPE: LR1110 version, as returned by the LR1110 GetVersion command. 0x01 stands for Transceiver use case.
- LR1110 HARDWARE: LR1110 chip version.
- LR1110 FIRMWARE: LR1110 internal firmware version.

5.2 Wi-Fi Passive Scanning

The main screen of the Wi-Fi Passive Scanning demo allows to configure the Wi-Fi Passive Scanning Wi-Fi type and channels pressing the “CONFIG” button, or to perform a Wi-Fi Passive Scanning using the saved Scanning configuration pressing the “START” button.

In any menu, the “BACK” button allows returning to the previous menu.



Figure 6: Wi-Fi Scanning Main Menu

The scan is running until 5 MAC addresses have been detected or 10 retrials on each channel. The Wi-Fi timeout has been set to 105ms. All the available MAC addresses might not be detected by the LR1110, due to the nature of the LR1110 Wi-Fi Passive Scanning algorithm, which optimizes the Wi-Fi MAC addresses detection for the fastest and lowest power consumption geolocation. The “sniffing” diode on the EVK is on during the Scanning process.

At the end of the Wi-Fi scanning process, the number of MAC addresses found on the configured channels is displayed on the screen, as well as the total amount of energy and time needed to detect those MAC addresses.

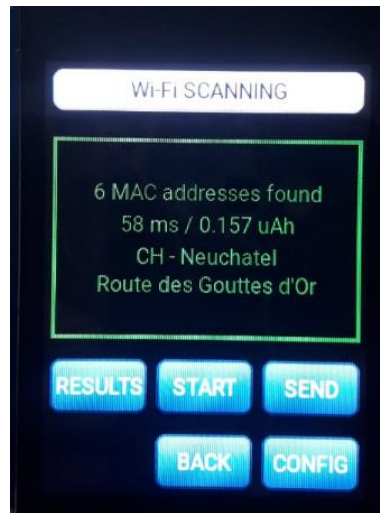


Figure 4: Wi-Fi Scanning Results With Reversed Geocoding

The “START” button triggers a new Wi-Fi scanning, whereas the “SEND” button sends the scanning results to the Python host application running on the PC. The Python software creates and sends a request to the geolocation server and waits for its response. The response from the geolocation server is then propagated back to the EVK. The geolocation demo displays the device resolved location, converted from the GPS coordinates calculated by the solver by reverse geocoding.

The “RESULTS” button shows the detected MAC addresses per channel and Wi-Fi type, with the corresponding RSSI value. The arrows allow navigating between the different result pages.

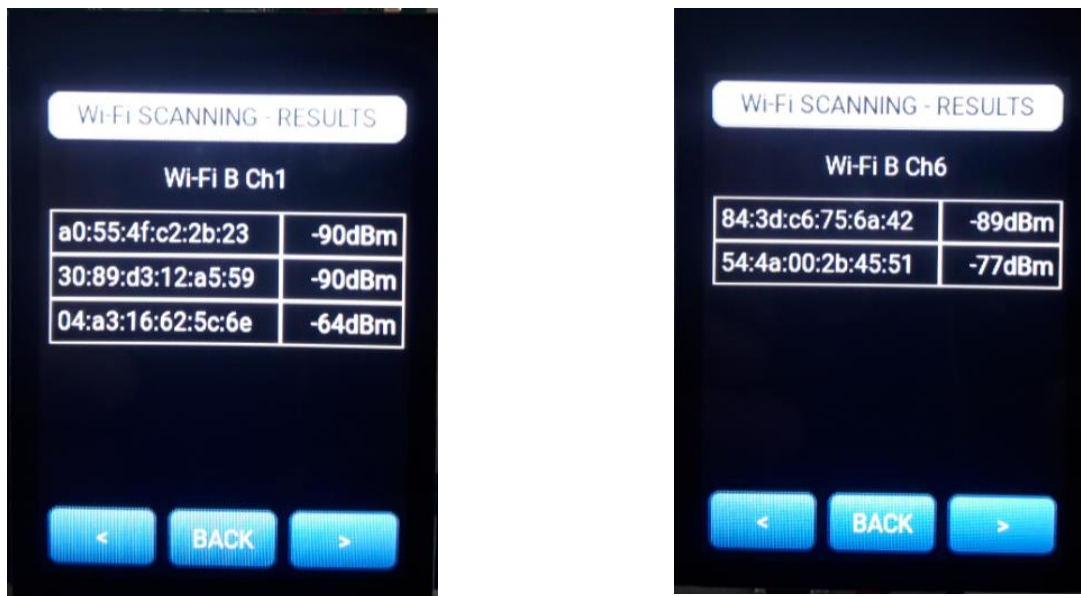


Figure 5: Wi-Fi Scanning Results

The “CONFIG” button of the Wi-Fi Scanning demo main menu calls the configuration menu, as shown in Figure 6: Wi-Fi Scanning Configuration Menu.



Figure 6: Wi-Fi Scanning Configuration Menu

The Wi-Fi Passive Scanning can be performed either for the one of the Wi-Fi types 802.11b, g or n, or for all the 3 types successively. For the selected Wi-Fi type, the user can select any subset of the 14 Wi-Fi channels.

The "SAVE" buttons saves the current configuration. By default, the Wi-Fi Scanning is performed for the Wi-Fi b type, on the 3 channels 1, 6, 11. The default configuration can be restored via the "DEFAULT" button.

5.3 GNSS Scanning

2 GNSS scanning modes are implemented:

- In GNSS autonomous mode the LR1110 searches and decodes the signal from the strong satellites. This mode does not require any assistance data. It can be used for indoor / outdoor detection. It requires connectivity with the host application to provide the current capture time with a 1sec precision.
- The GNSS assisted mode allows searching for all the visible satellites. It requires connectivity with the host application to provide assistance information (current capture time with a 1sec precision, and approximate location with 150km accuracy). It requires also connectivity with the geolocation server for calculation of the device position. Up-to-date LR1110 almanac data allows achieving minimum capture time.

The “START” button triggers a new GNSS capture, whereas the “SEND” button sends the scanning results to the geolocation server, and displays the device location thanks to reverse geocoding. The “sniffing” diode on the EVK is on during the Scanning process.

For each mode, the LR1110 geolocation demo displays time and energy needed for the capture.



Figure 7: GNSS Autonomous and Assisted Scanning Results

The “RESULTS” button displays the list of the detected satellites, with the corresponding C/N₀.

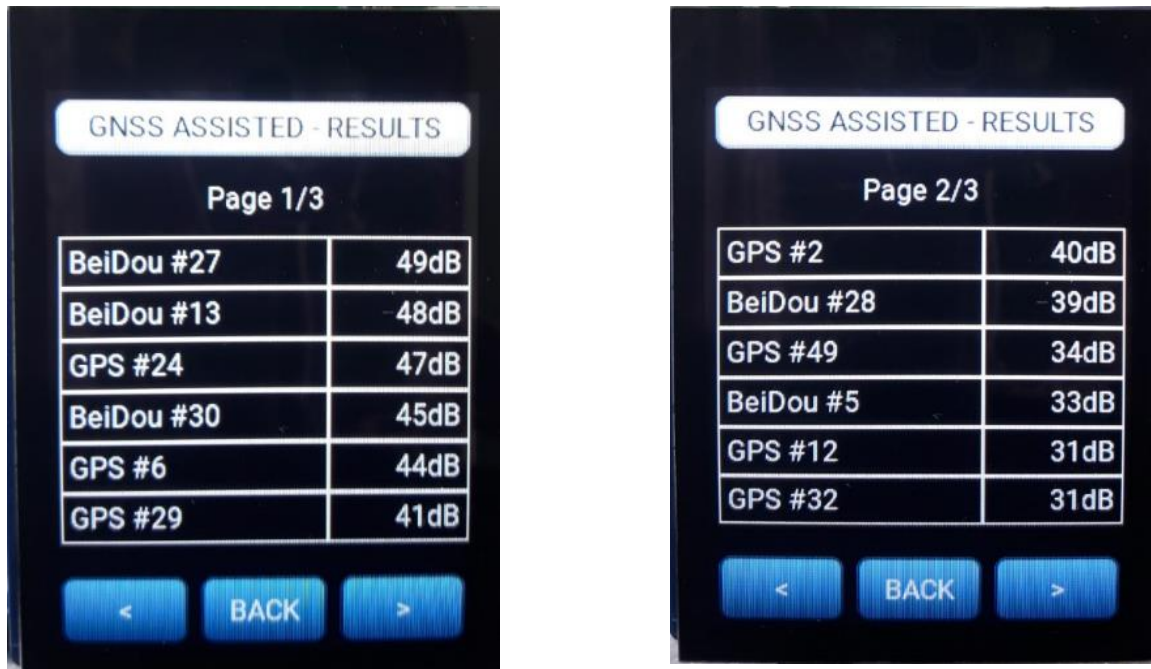


Figure 8: GNSS Assisted Scanning Results- GPS and BeiDou Detected Satellites

The “CONFIG” buttons allows configuring the GNSS Scanning, as shown in Figure 9: GNSS Autonomous (left) and Assisted (right) Configuration Screen.

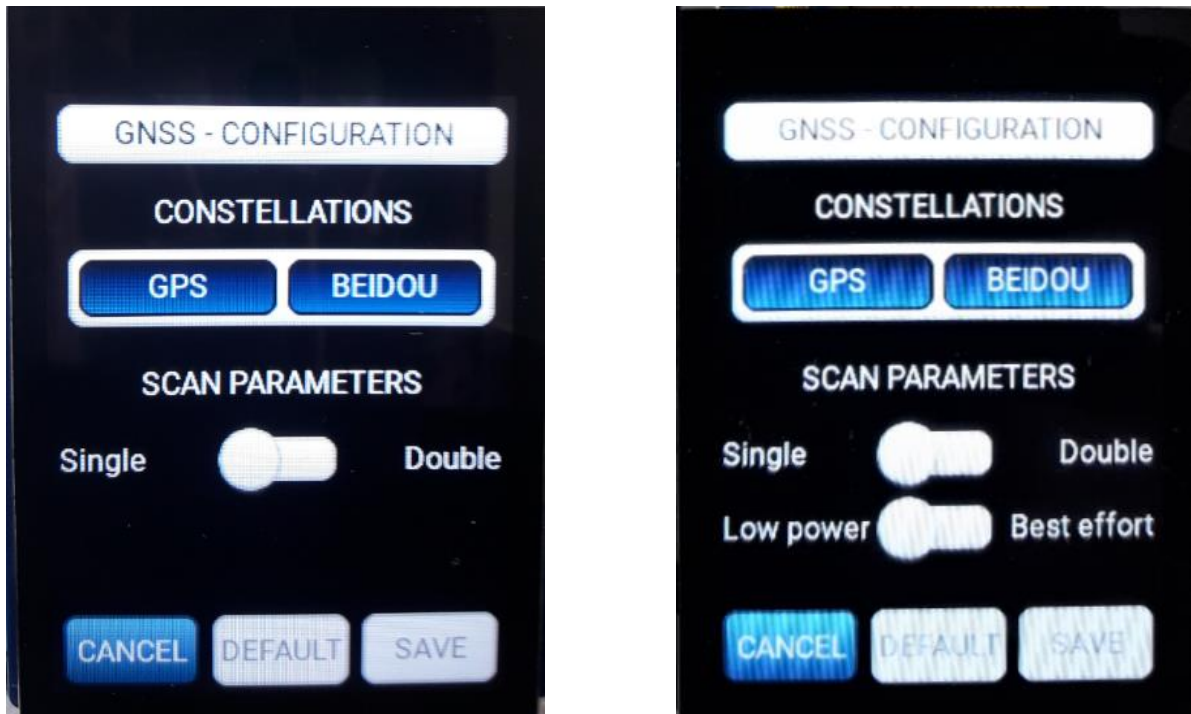


Figure 9: GNSS Autonomous (left) and Assisted (right) Configuration Screen

For each of the GNSS modes autonomous or assisted, the user can select the constellation: either GPS, or BeiDou, or both GPS and BeiDou. When both GPS and BeiDou constellations are selected, there is a delay of 4 seconds between the Scans, during which the LR1110 is put in sleep mode to achieve the lowest power consumption.

The Scan can then be either a Single Scan, or a Double Scan. In the case of a Double scan, there is a 30 seconds sleep period between the first two Scans, as shown by the “Sniffing” diode pattern, as shown in Figure 10: Sniffing Diode.



Figure 10: Sniffing Diode

6 Python Application

A Python host application allows connectivity with the geolocation server in order to send the scanning results and retrieve the geolocation. It also provides the LR1110 device assistance information (approximate position and time) for the GNSS assisted mode.

For a complete installation and troubleshooting guide, please refer to the section 4 Python Host Application Installation.

6.1 Python Application Execution

In order to run the Python host application, type the following command and arguments in a command window:

```
Lr1110Demo -d <COM_PORT> -r <COORDINATES_ASSISTED_LR1110> <COORDINATES_EXACT>  
<TOKEN_WIFI_GLS> <TOKEN_GNSS_DAS>
```

with:

- <COM_PORT>: Nucleo COM port (COMx on windows, /dev/ttyACMx on linux)

-r : reverse geocoding (optional). Displays the resolved location address.

- <COORDINATES_ASSISTED_LR1110>: approximate position (150km accuracy) for the GNSS assisted scan. The format is <DECIMAL_LATITUDE>,<DECIMAL_LONGITUDE>,<DECIMAL_ALTITUDE>. Please note that there is no space between the commas. For example, for Semtech Neuchatel office, the coordinates will be 47.006,6.966,400.

- <COORDINATES_EXACT>: real coordinates in order for the solver to calculate and display the geolocation error. The format is the same as previously.

- <TOKEN_WIFI_GLS>: HTTP header token from the LoRa Cloud Geo Location Service (GLS), in order to authenticate the requests. Can be obtained via https://www.loracloud.com/portal/geolocation/token_management

- <TOKEN_GNSS_DAS>: HTTP header token from the LoRa Cloud Device & Application Services (DAS), in order to authenticate the requests. Can be obtained via https://www.loracloud.com/portal/device_management/tokens.

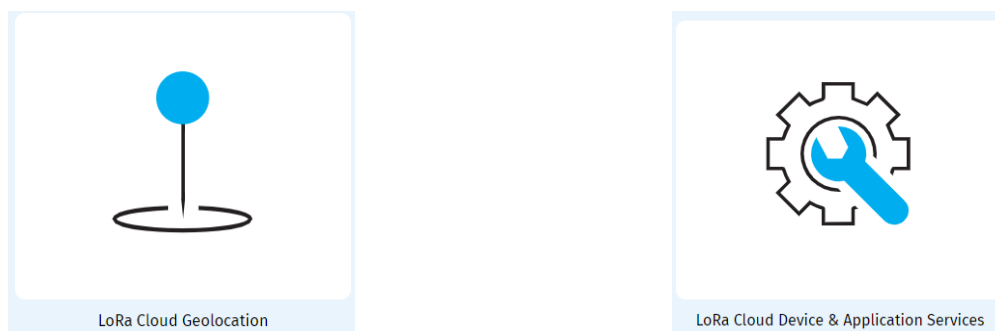


Figure 11: Wi-Fi GLS (left) and GNSS DAS (right) Services in LoRa Cloud

Example:

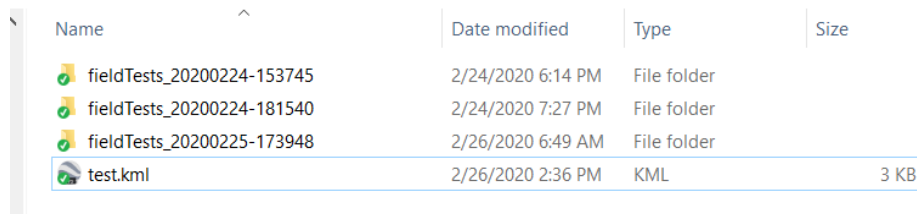
Nucleo board enumerated on COM60, approximate location in Neuchatel, Switzerland:

```
Lr1110Demo -d COM60 -r 47.006,6.966,400 47.006229,6.966572,440 <TOKEN_WIFI_GLS>  
<TOKEN_GNSS_DAS>
```


The command `Lr1110Demo --help` provides additional information on the LR1110 demo application parameters.

7 Geolocation Results Visualization

The Geolocation demo creates a Keyhole Markup Language file `test.kml`, containing the results of the geolocation. This file is generated at the file location where the Python host application program is executed:



Name	Date modified	Type	Size
fieldTests_20200224-153745	2/24/2020 6:14 PM	File folder	
fieldTests_20200224-181540	2/24/2020 7:27 PM	File folder	
fieldTests_20200225-173948	2/26/2020 6:49 AM	File folder	
test.kml	2/26/2020 2:36 PM	KML	3 KB

Figure 12: Results.kml File

This kml file contains not only the coordinates of the user calculated via the Wi-Fi and GNSS solvers, but also some metadata relative to the Wi-Fi (list of Wi-Fi MAC addresses and RSSI) and GNSS captures (NAV message).

The geolocation results contained in the kml file can be visualized on the main 2D maps and 3D Earth browsers, as shown in Figure 13: Geolocation Results Visualization Map .

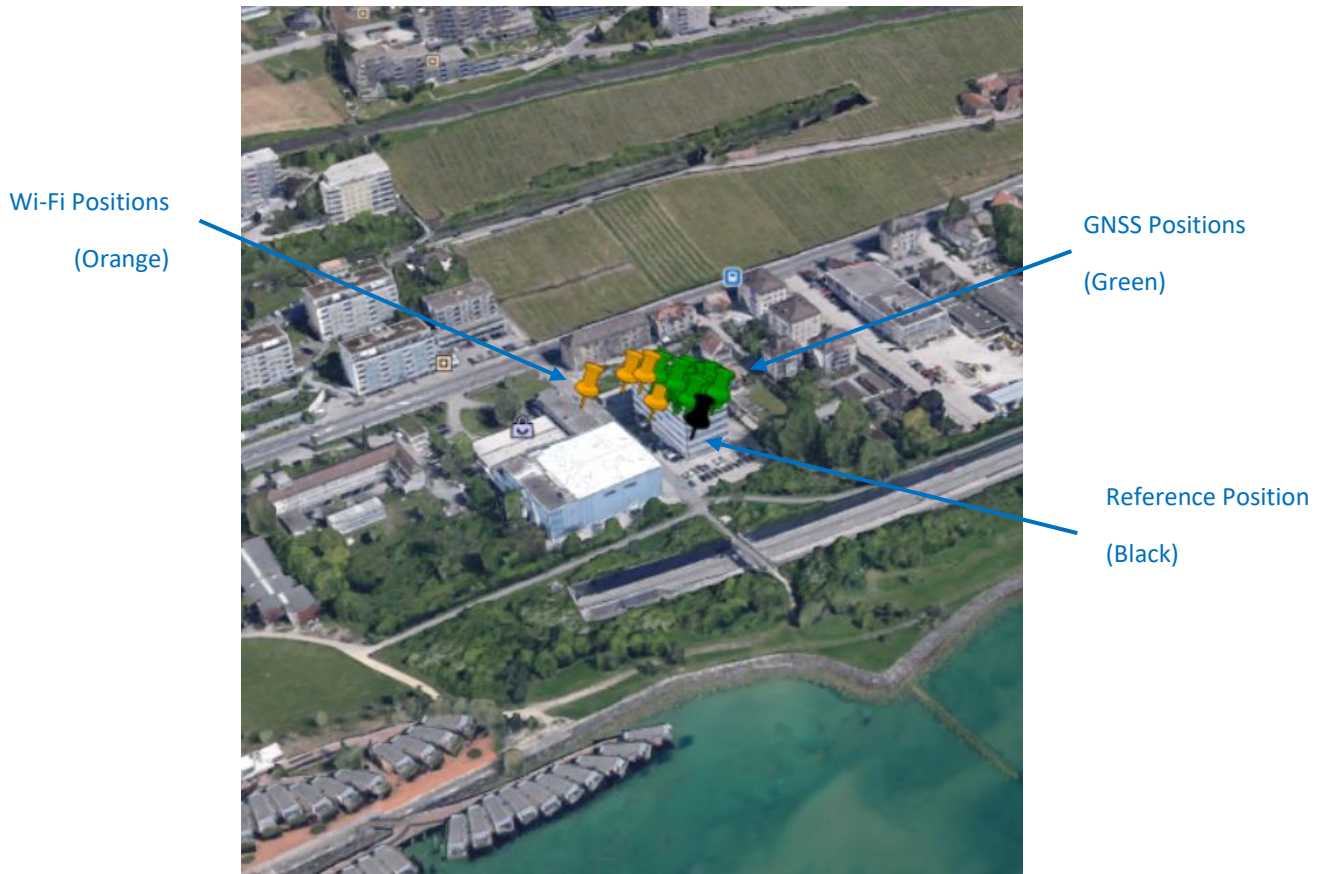


Figure 13: Geolocation Results Visualization Map

Clicking on the Wi-Fi or GNSS points allows displaying the associated Wi-Fi and GNSS geolocation results, as shown in Figure 14: Wi-Fi Geolocation Results with Metadata and Figure 15: GNSS Geolocation Results with Metadata here below.

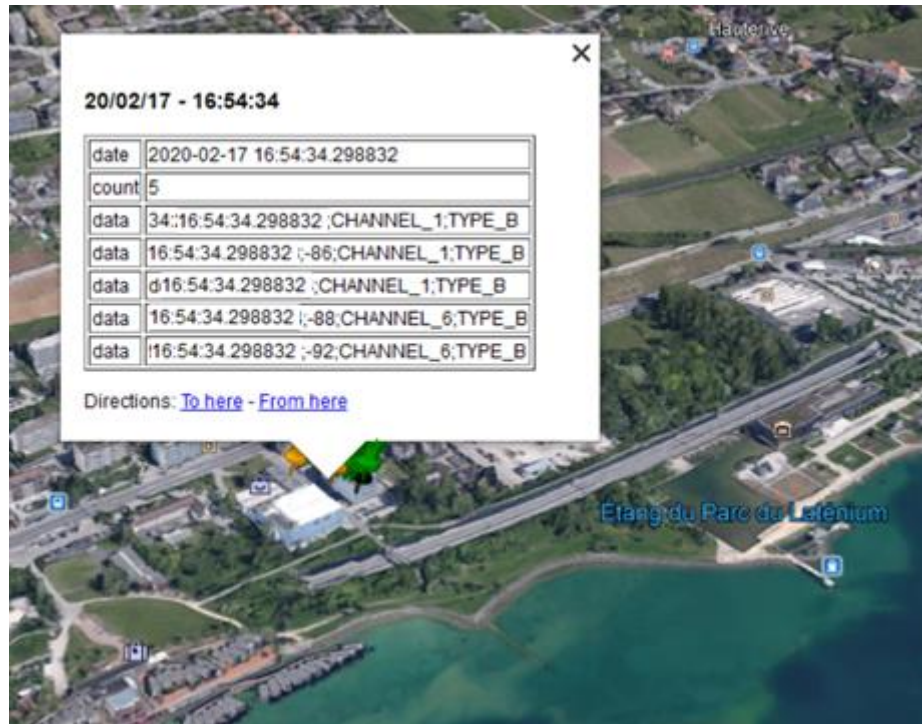


Figure 14: Wi-Fi Geolocation Results with Metadata

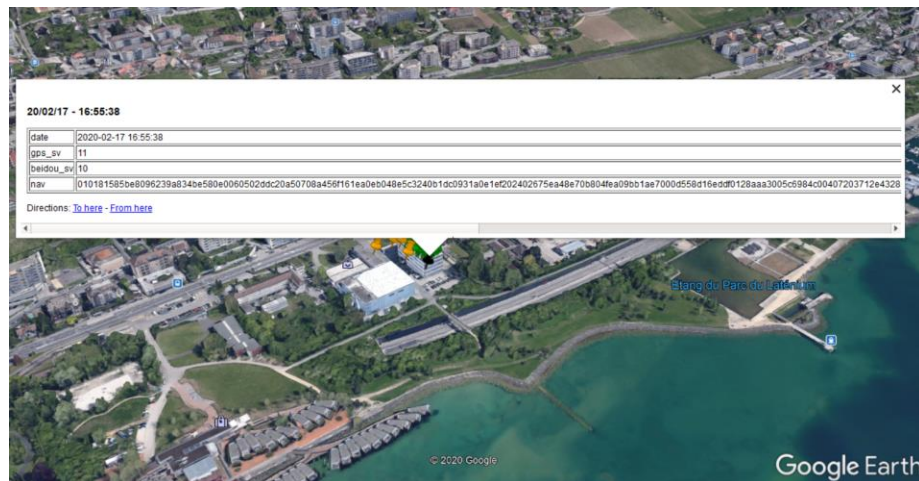


Figure 15: GNSS Geolocation Results with Metadata

8 Almanac Update Process

8.1 Introduction

The Almanac are a set of parameters used to estimate the position of the visible SVs, at the time and location of the scan. Almanac data not up-to-date imply an increase of the capture time and of the energy spent for the scan. LR1110 Almanac data must be updated every three months, though it is strongly advised to update it every month for optimum scanning performance. Monthly Almanac images are available from LoRaCloud.

While performing a GNSS scan, a warning message is displayed on EVK screen and on the Python execution trace if Almanac data is at least 31 days old, as shown in Figure 16: Almanacs Need Update Warning. This warning message will not prevent the execution of the GNSS scan if the Almanac data is less to 3 months old, but the GNSS scan will not be executed if the Almanac data is more than 3 months old.



Figure 16: Almanacs Need Update Warning

8.2 Obtaining The Latest Almanac

The Almanac are monthly distributed by Semtech LoRaCloud server. (Refer to https://www.loracloud.com/documentation/device_management?url=v1.html#fetch-monthly-almanac-image for details). This requires the GNSS_DAS token.

8.3 Updating The Almanac In The EVK

A python tool is available to download the almanac file obtained at previous step: AlmanacUpdate.

The usage is as follows:

```
AlmanacUpdate -d <COM_PORT> <TOKEN_GNSS_DAS>
```

Please note that the EVK must be connected on USB and reset prior using the AlmanacUpdate command.

The output displays the different steps of the almanac update process. It firsts detects the EVK, then it downloads the content of the Almanac from the LoRaCloud server. Lastly, it validates the almanac update by checking the almanac CRC.

The command `AlmanacUpdate --help` provides additional information on the LR1110 AlmanacUpdate application parameters.

9 Field Fests

The LR1110 demo is a portable setup, which allows performing tests in the field. The LR1110 demo kit needs then to be connected via USB to a laptop, where a PC host application loops the Wi-Fi and GNSS scanning results, and stores the results locally. At the end of the field test, the results can be sent to the geolocation server for post processing and analysis.

The field test setup uses 3 software pieces:

- A dedicated LR1110 dev kit firmware (.bin file), running on the LR1110 Nucleo board of the dev kit.
- The PC host application controlling the field test execution and storing the test results.
- A post-processing PC application sending the test results to the geolocation server

9.1 Field Test PC Host Application

9.1.1 Execution Process

In order to start the field test PC host application, enter the following in a command window:

```
Lr1110FieldTest -f <RESULT_FOLDER> <JSON_FILE>
```

Example:

```
Lr1110FieldTest -f TestsResults jobswifi.json
```

The Field test PC application executes the scanning tasks configured in the jobs *.json* file, and stores the results in the configured *Result_Folder*. At every startup of the *LR1110FieldTest* host application, it creates a time-stamped folder containing a copy of the jobs *.json* file, the scanning results *results.res* and a trace of the program execution *logging.log*, as shown in Figure 17: Field Test Results Folder hereafter:

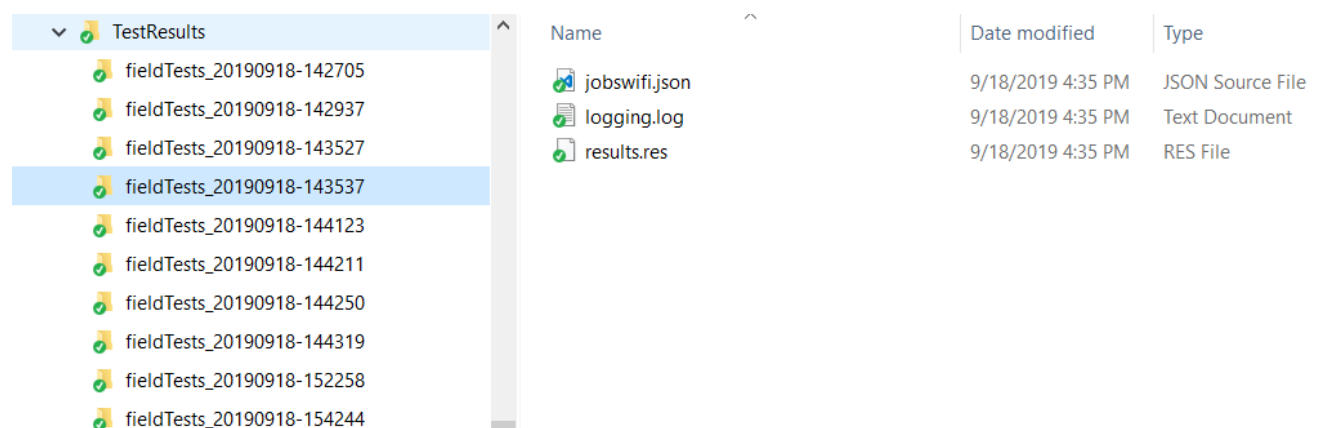


Figure 17: Field Test Results Folder

By default, the *jobs.json* file is located in the Python application installation path below:

C:\Users\username\AppData\Local\Programs\Python\Python37-32\Lib\site-packages\LR1110FieldTest\assets

For convenience, it is possible to create a dedicated Result folder in any (simpler) location, and execute the *LR1110FieldTest* host application from this folder.

To get some help on the *LR1110FieldTest* application, type:

```
LR1110FieldTest --help
```

9.1.2 Jobs File

The field test execution is split into scanning tasks -called a job- , which are sequentially executed until the end of the job *.json* file is reached. Depending on the *infinite_loops* parameter, the whole jobs list is either looped indefinitely, or executed only once. Each job is independently parameterized (e.g. Wi-Fi channels to scan, Wi-Fi type, ...) and can be independently repeated (*n_iterations* parameter).

The parameter *intermediary mode* is deprecated, and will be removed in a future version of the Python host application.

```
{
  "infinite_loops": true,
  "jobs": [
    {
      "name": "WiFi Scan",
      "n_iterations": 1,
      "reset_before_job_start": false,
      "intermediary_mode": "none",
      "wifi_channels": [
        "CHANNEL_1",
        "CHANNEL_6",
        "CHANNEL_11"
      ],
      "wifi_types": [
        "TYPE_B"
      ],
      "wifi_nbr_retrial": 10,
      "wifi_max_result_per_scan": 5,
      "wifi_timeout": 110,
      "wifi_mode": "beacon_and_packet",
      "wifi_api": "wifi_scan"
    },
    {
      "name": "GNSS assisted single",
      "n_iterations": 1,
      "reset_before_job_start": false,
      "intermediary_mode": "none",
      "gnss_assisted_option": "best_effort",
      "gnss_assisted_capture_mode": "single",
      "gnss_assisted_nb_satellite": 0,
      "gnss_assisted_antenna_selection": "no_selection",
      "gnss_assisted_constellations": ["gps", "beidou"],
      "assisted_coordinate": {
        "latitude": 49.458652,
        "longitude": 11.035677,
        "altitude": 300
      }
    }
  ]
}
```

Figure 18: Field Test Jobs.json Structure

9.2 Post Processing Application

At the end of the field test, the `FieldTestPost` program allows sending the field test data to the geolocation server. The different parameters are the result file, where to store the coordinates returned by the server, the assisted GNSS coordinates, the exact coordinates for the location error calculation, an ID for the visualization webpage, and an identification token:

```
FieldTestPost <RESULT_FILE> <SAVE_FILE_FOR_SERVER_RESPONSE> <ASSISTED_COORDINATE>  
<TOKEN_WIFI_GLS> <TOKEN_GNSS_DAS>
```

with:

- `<RESULT_FILE>`: .res file generated by the Python host application during the field tests execution, sent to the geolocation server for the position resolution.

- `<SAVE_FILE_FOR_SERVER_RESPONSE>`: output file, containing the coordinates calculated by the geolocation solvers.

- `<ASSISTED_COORDINATE>`: approximate position (150km accuracy), used by the geolocation solver as initial value for the geolocation calculation. The format is `<DECIMAL_LATITUDE>,<DECIMAL_LONGITUDE>,<DECIMAL_ALTITUDE>`. Please note that there is no space between the commas. For example, for Semtech Neuchatel office, the coordinates will be 47.006,6.966,400.

- `<TOKEN_WIFI_GLS>`: HTTP header token from the LoRa Cloud Geo Location Service, in order to authenticate the requests. Can be obtained via https://www.loracloud.com/portal/geolocation/token_management (Refer to Figure 11: Wi-Fi GLS (left) and GNSS DAS (right) Services in LoRa Cloud).

- `<TOKEN_GNSS_DAS>`: HTTP header token from the LoRa Cloud Device & Application Services, in order to authenticate the requests. Can be obtained via https://www.loracloud.com/portal/device_management/tokens (Refer to Figure 11: Wi-Fi GLS (left) and GNSS DAS (right) Services in LoRa Cloud).

```
Example:      FieldTestPost      ./results.res      test_20190918      47.006,6.966,400  
<TOKEN_WIFI_GLS> <TOKEN_GNSS_DAS>
```

The `FieldTestPost` program automatically selects the server to contact depending on the nature of the results (GNSS or Wi-Fi).

More information help on the `FieldTestPost` application can be obtained via:

```
FieldTestPost --help
```


10 Revision History

Revision	Date	Applicable to	Modifications
1.0	March-2020	Use Case: 01 FW Version: 03.02 or later	Initial Version



Important Notice

Information relating to this product and the application or design described herein is believed to be reliable, however such information is provided as a guide only and Semtech assumes no liability for any errors in this document, or for the application or design described herein. Semtech reserves the right to make changes to the product or this document at any time without notice. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. Semtech warrants performance of its products to the specifications applicable at the time of sale, and all sales are made in accordance with Semtech's standard terms and conditions of sale.

SEMTECH PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS, OR IN NUCLEAR APPLICATIONS IN WHICH THE FAILURE COULD BE REASONABLY EXPECTED TO RESULT IN PERSONAL INJURY, LOSS OF LIFE OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. INCLUSION OF SEMTECH PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE UNDERTAKEN SOLELY AT THE CUSTOMER'S OWN RISK. Should a customer purchase or use Semtech products for any such unauthorized application, the customer shall indemnify and hold Semtech and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs damages and attorney fees which could arise.

The Semtech name and logo are registered trademarks of the Semtech Corporation. The LoRa® Mark is a registered trademark of the

Semtech Corporation. All other trademarks and trade names mentioned may be marks and names of Semtech or their respective companies. Semtech reserves the right to make changes to, or discontinue any products described in this document without further notice. Semtech makes no warranty, representation or guarantee, express or implied, regarding the suitability of its products for any particular purpose. All rights reserved.

© Semtech 2020

Contact Information

Semtech Corporation
Wireless & Sensing Products
200 Flynn Road, Camarillo, CA 93012
E-mail: sales@semtech.com
Phone: (805) 498-2111, Fax: (805) 498-3804
www.semtech.com