Getting started with X-LINUX-GNSS1 package for developing GNSS applications on Linux OS

Introduction

X-LINUX-GNSS1 is an STM32 MPU OpenSTLinux software expansion package that runs on the Arm Cortex®-A7-based core of the STM32MP1 microprocessor on the STM32MP157F-DK2 discovery kit to demonstrate GNSS-based applications.

X-LINUX-GNSS1 includes user space application, a device tree for the Teseo-LIV3F global navigation satellite system (GNSS) device, a library for NMEA protocol support and POSIX thread for task scheduling to ensure better asynchronous message parsing.

The software contains various application modules to retrieve the NMEA GNSS data and upload it to DSH-ASSETRACKING.

The source code can be ported to any Linux platform.

1 X-LINUX-GNSS1 overview

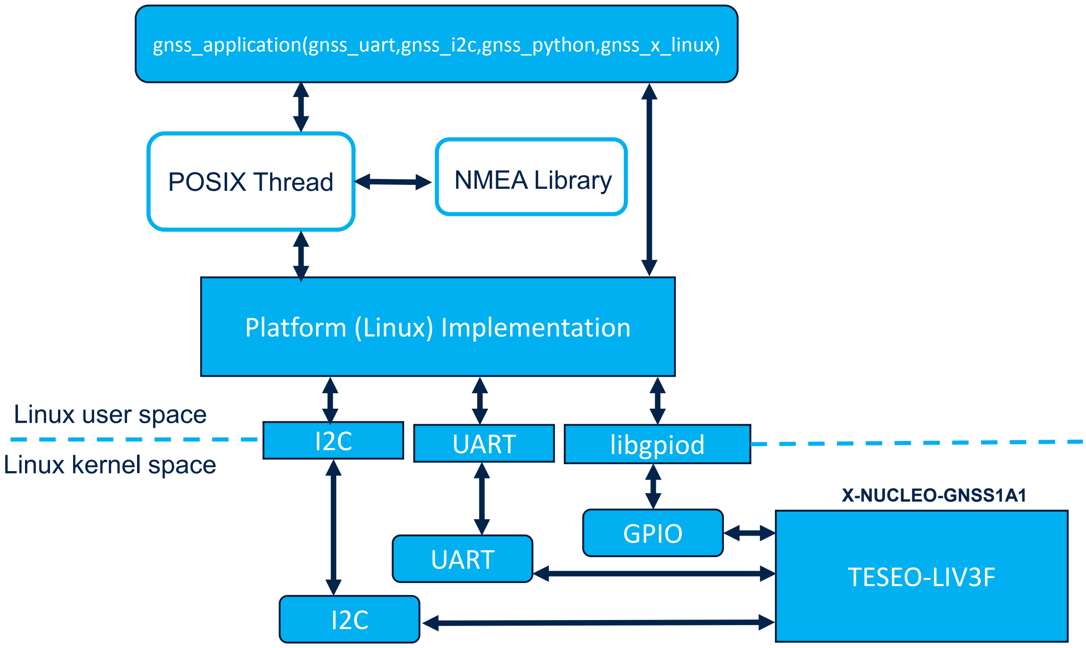
The X-LINUX-GNSS1 software provides a user space application running on STM32MP157F-DK2 for the X-NUCLEO-GNSS1A1 expansion board based on the Teseo-LIV3F tiny global navigation satellite system (GNSS) module.

The software package contains the following modules:

1. gnss_app (x-linux-gnss)
2. C utility(gnss_uart and gnss_i2c)
3. A Python utility (gnss_pynmea2.py)

Each software module can run independently to acquire the GNSS NMEA data from the X-NUCLEO-GNSS1A1 over UART and I²C.

Figure 1. X-LINUX-GNSS1 user space application architecture

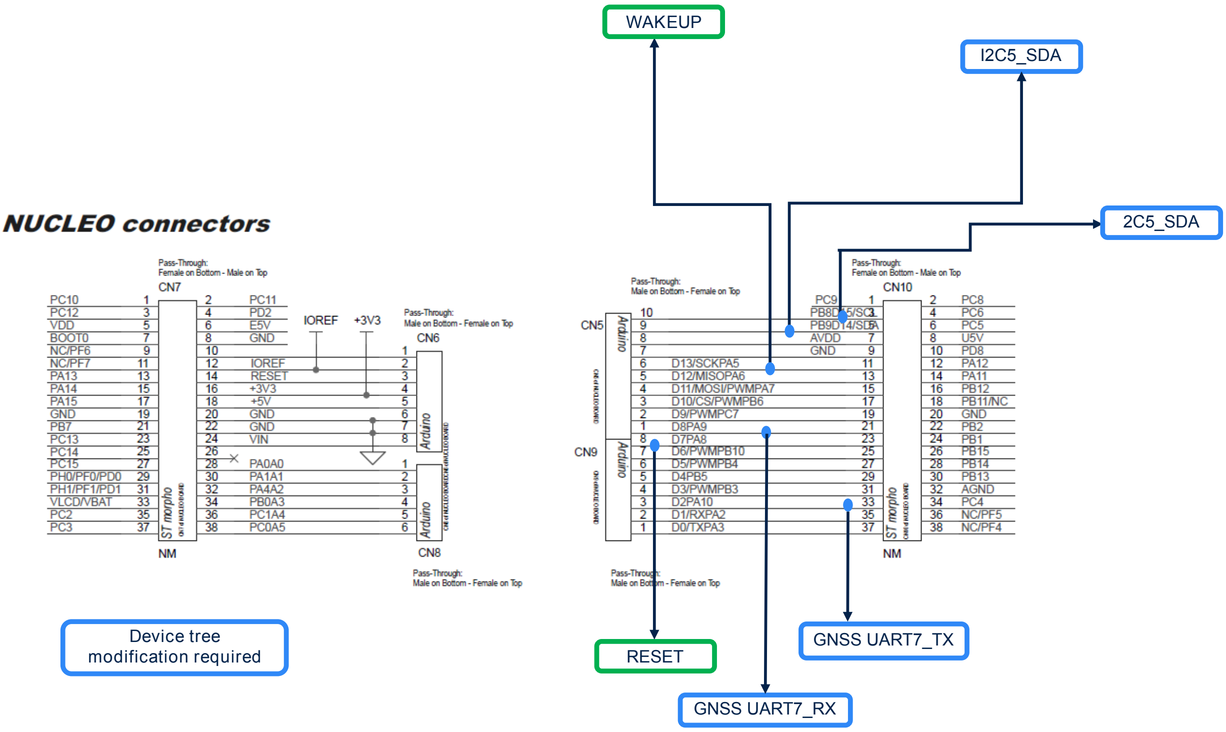


The X-NUCLEO-GNSS1A1 device tree has been modified to configure the UART7 and I2C5 on the Arduino connector. For UART, the underlying dev/ttySTM2 is enabled, whereas /dev/i2c-1 is enabled for I²C.

The X-LINUX-GNSS1 software interacts with the lower layer peripheral drivers (I²C and UART) through the user space application. It uses termios for UART and file descriptor reading for the I²C peripheral.

The software also exploits POSIX thread to run two parallel tasks (Consumer Task and Console Task). The Consumer Task acquires the NMEA data, parse them and populates the NMEA data structure. The Console Task reads the input from the user application and provides the information from the populated NMEA data structure, such as position, speed, elevation, etc., based on the provided inputs.

Figure 2. **X-NUCLEO-GNSS1A1 hardware connections with required tree modification**



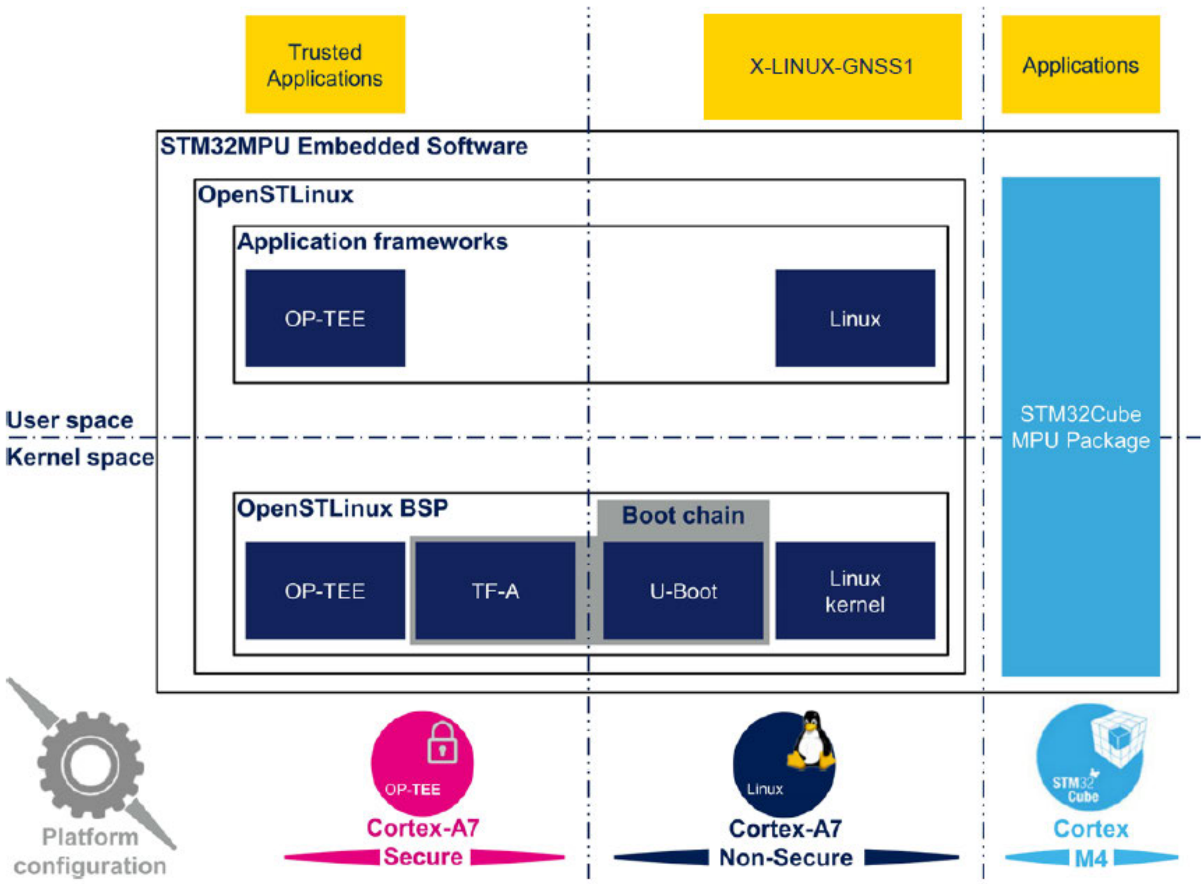
1.1 Features

- Standalone applications to read the NMEA data over UART and I²C
- Complete software to build applications on Linux using Teseo-LIV3F and Teseo-VIC3DA GNSS module
- Middleware for the NMEA protocol
- POSIX thread task scheduling to ensure better asynchronous message parsing
- Easy portability across different Linux platforms
- Application example to retrieve and parse GNSS data and send them to DSH-ASSETRACKING for live tracking
- Python example to read the NMEA data over UART

1.2 Architecture

The software package runs on the ARM Cortex-A7 core of the STM32MP157F-DK2. The X-LINUX-GNSS1 interacts with the lower layers libraries and SPI lines exposed by the Linux software framework.

Figure 3. **X-LINUX-GNSS1 application architecture in Linux environment**



1.3 Software package structure

The X-LINUX-GNSS_V1.0.0 release package contains Linux user application C examples, a Python example, the device tree and a Yocto layer recipe.

You can run any application independently in the Application folder to retrieve the GNSS NMEA data.

Figure 4. **X-LINUX-GNSS1_V1.1.0 release package structure - top level**

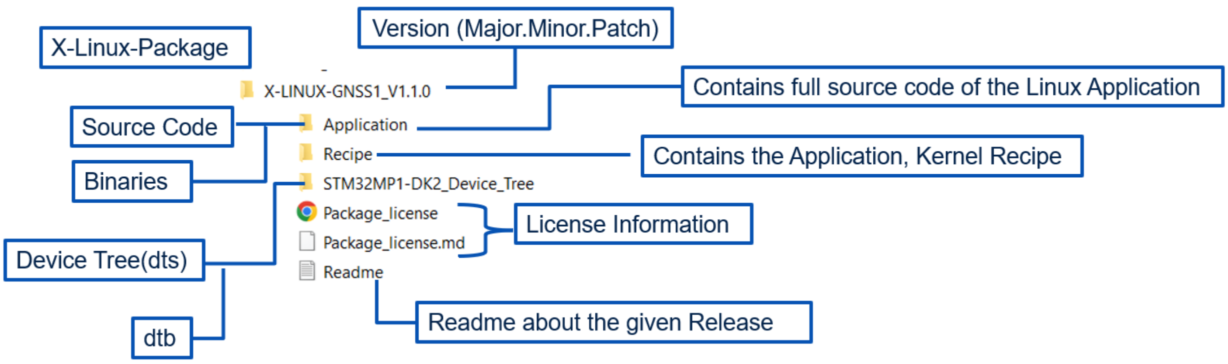


Figure 5. **X-LINUX-GNSS_V1.1.0 release package structure - application folders**

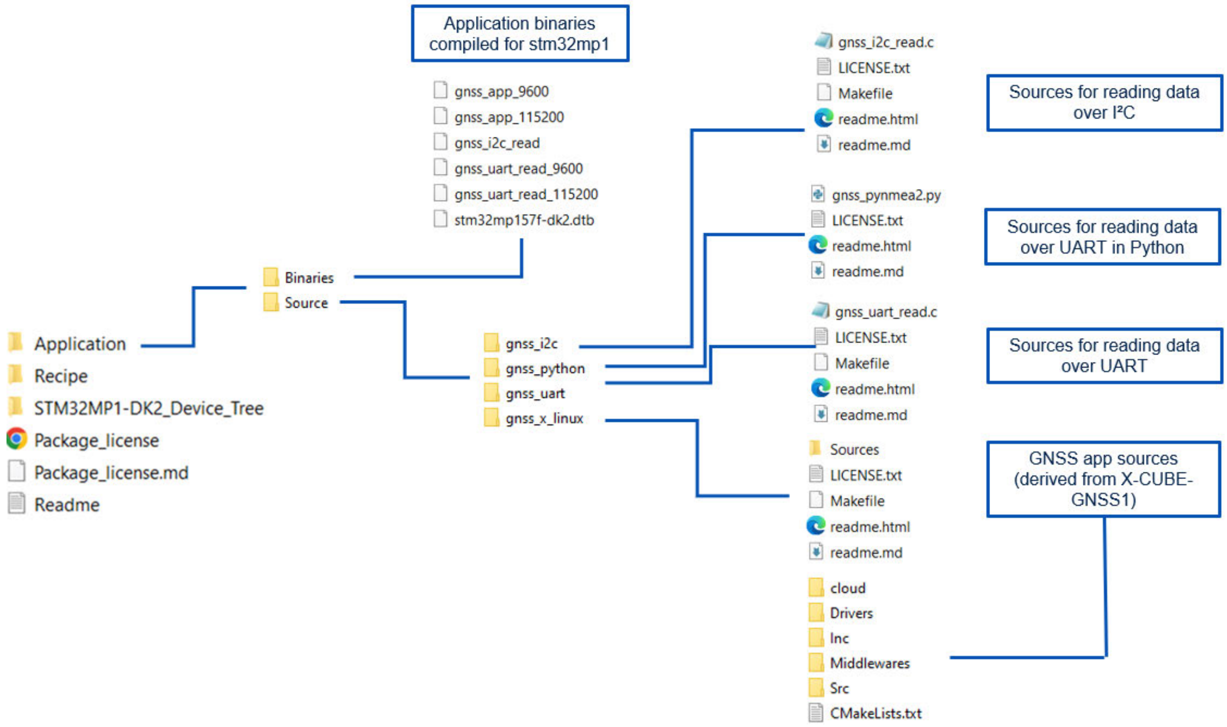
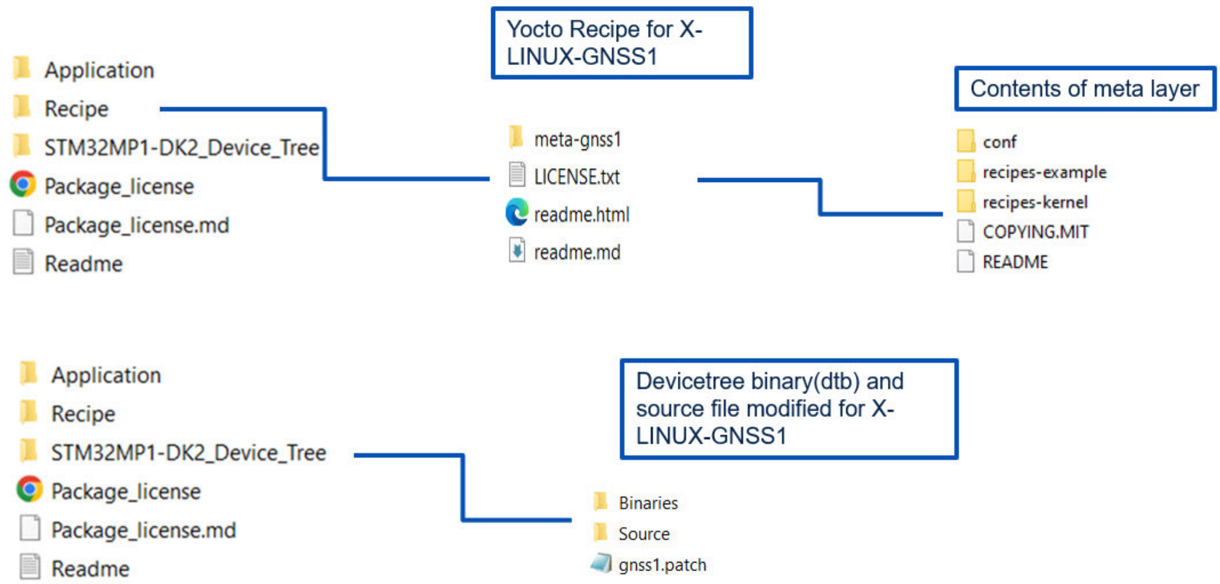


Figure 6. **X-LINUX-GNSS_V1.1.0 release package structure - GNSS other folders**

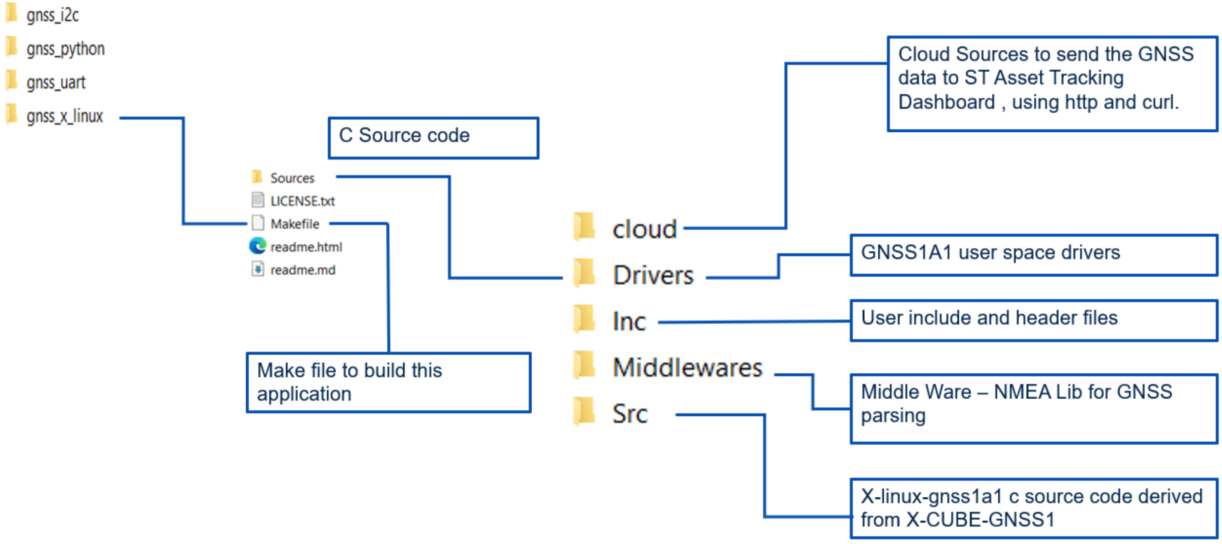


1.3.1 gnss_app

This application accesses the GPS data over UART (/dev/ttySTM2) and I²C (/dev/i2c-1) interface. The settings to enable UART and I²C are provided separately in the device tree file folder.

Through the gnss_app you can upload the data to the cloud (DSH-ASSETRACKING).

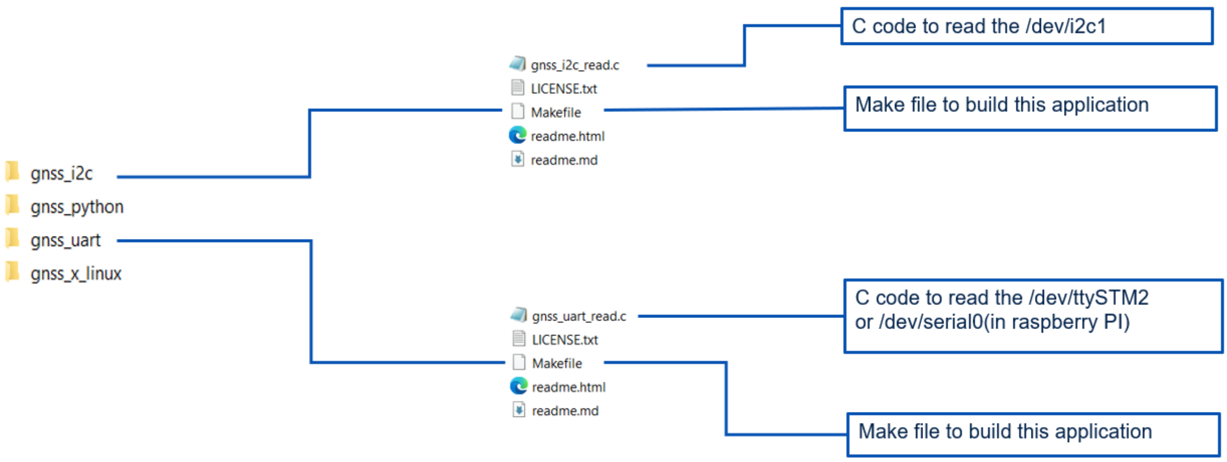
Figure 7. **gnss_app folder structure**



1.3.2 C utility

This Linux user space C application reads data from UART (/dev/ttySTM2) and I²C (/dev/i2c-1) interface.

Figure 8. **C utility - gnss-uart and gnss-i2c**

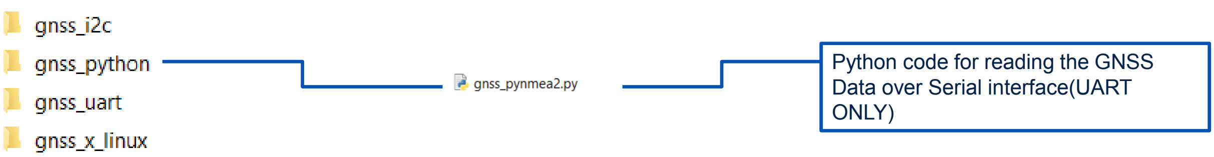


1.3.3 Python code application

This application is a basic Python code to read data from UART.

Important: *You need to install pyserial and pynmea2 library before using this application.*

Figure 9. **Python code application structure**



2 Hardware setup

The X-LINUX-GNSS1 is compatible with the X-NUCLEO-GNSS1A1 expansion board which can be directly plugged on the STM32MP157F-DK2 discovery kit Arduino connectors.

Step 1. Set the X-NUCLEO-GNSS1A1 jumpers as shown in the figures and the tables below.

Figure 10. X-NUCLEO-GNSS1A1 expansion board jumper settings

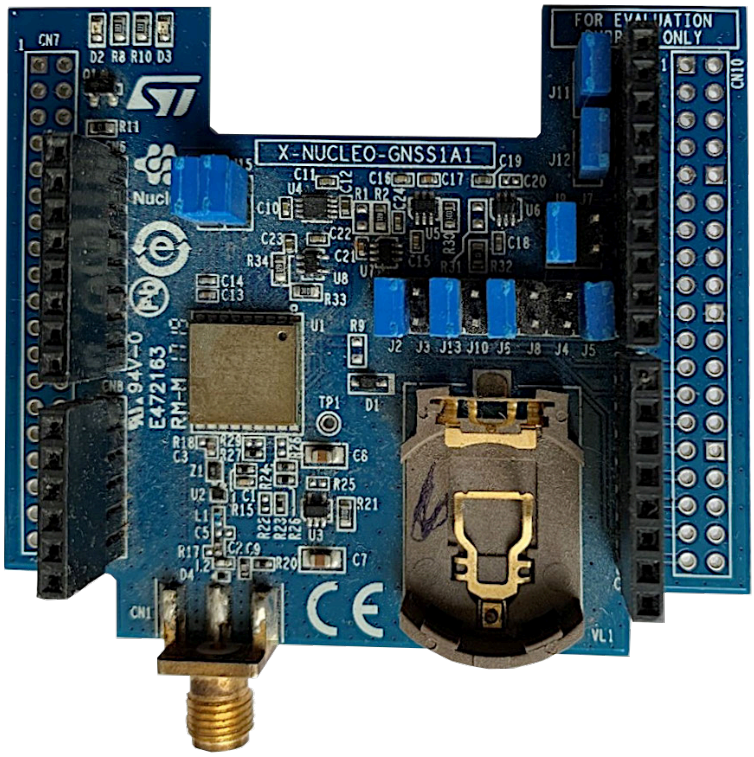


Figure 11. X-NUCLEO-GNSS2A1 expansion board jumper settings

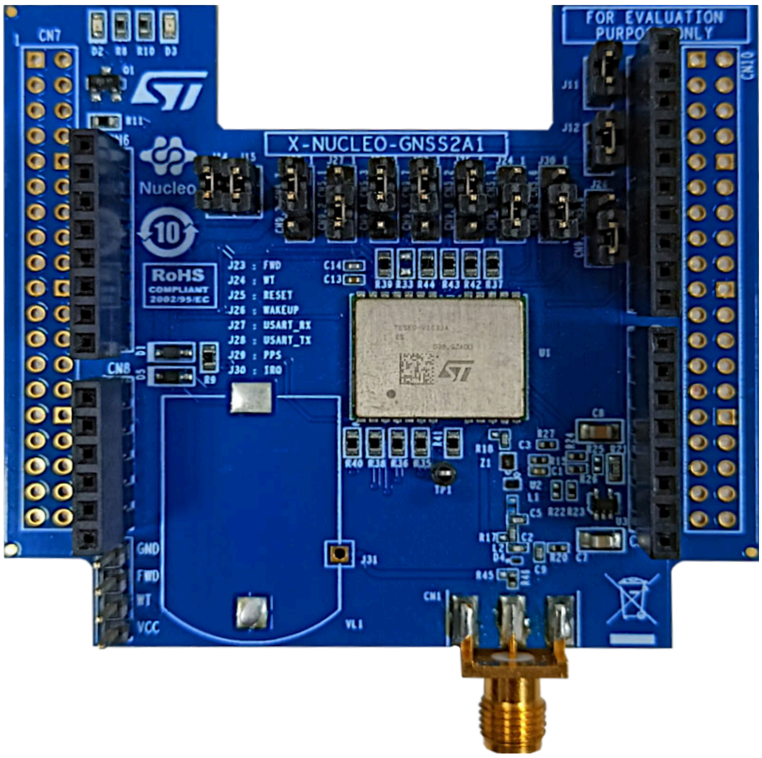


Table 1. **X-NUCLEO-GNSS1A1 jumper configuration**

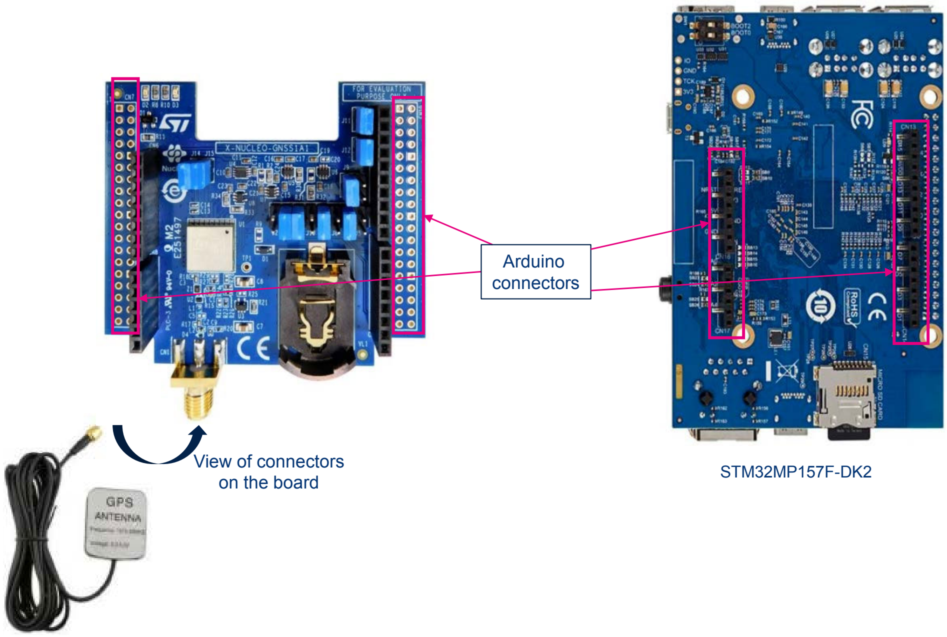
Signal	Arduino connector	Discovery kit	Jumper	Configuration
I2C-SCL	D15	PB8	J11	Closed
I2C-SDA	D14	PB9	J12	Closed
Wakeup	D13	PA5	J9	Closed
Wakeup	D4	PB5	J7	Open
Reset	D9	PC7	J10	Open
Reset	D7	PA8	J13	Closed
PPS	D6	PB10	J6	Closed
PPS	D2	PA10	J8	Open
UART-RX	D8	PA9	J3	Open
UART-TX	D2	PA10	J4	Open
UART-RX	D1	PA2	J2	Closed
UART-TX	D0	PA3	J5	Open

Table 2. **X-NUCLEO-GNSS2A1 jumper configuration**

Signal	Jumper	Configuration
I^2C-SCL	J11	Closed
I^2C-SDA	J12	Closed
VCC-VCC_IO	J14	Closed
V14Bat	J15	Closed
SYS_FWD	J23	1-2
SYS_WHEELTICK	J24	2-3
SYS_RESETn	J25	1-2
SYS_WAKEUP	J26	2-3
UART-RX	J27	2-3
UART-TX	J28	1-2
SYS_PPS	J29	1-2
SYS_IRQ	J30	2-3

Step 2. Plug the expansion board onto the discovery kit Arduino connectors.

Figure 12. X-NUCLEO-GNSS1A1 and STM32MP157F-DK2 connection



Step 3. Connect the GPS/GLONASS/Beidou antenna provided with X-NUCLEO-GNSS1A1.

Tip: *Keep the antenna outdoor for better reception.*

The STM32MP157F-DK2 is powered by the USB Type-C cable.

3 Software setup

This section describes the software setup required to build, flash, transfer, and run the GNSS application.

3.1 PC/laptop requirements

For the software setup, you need:

- a Linux® PC/laptop running Ubuntu® 18.04 or 20.04
- detailed instructions at https://wiki.st.com/stm32mpu/wiki/PC_prerequisites

3.2 Installing the SDK

The software package contains the binaries you can transfer using scp command.

Install the SDK to help you build a customized application, by following the instructions at https://wiki.st.com/stm32mpu/wiki/Getting_started/STM32MP1_boards/STM32MP157x-DK2/Develop_on_Arm%C2%AE_Cortex%C2%AE-A7/Install_the_SDK.

3.3 Downloading the kernel sources (developer package)

Downloading the kernel sources is required to build the device tree. The software package already contains the binaries (dtb) which can be transferred using scp command. For the complete guide on how to download the kernel sources, see https://wiki.st.com/stm32mpu/wiki/Getting_started/STM32MP1_boards/STM32MP157x-DK2/Develop_on_Arm%C2%AE_Cortex%C2%AE-A7/Modify,_rebuild_and_reload_the_Linux%C2%AE_kernel#

KERNEL SOURCE PATH
= ~/STM32MPU_workspace/STM32MP15-Ecosystem-v3.0.0/Developer-Package/stm32mp1-openstlinux-5.10-dunfell-mp1-21-03-31/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.10.10-r0/linux-5.10.10$

3.4 Downloading the distribution package

This is required to build the recipes and create STM32MP1 images which have GNSS application and device tree settings embedded (see https://wiki.st.com/stm32mpu/wiki/STM32MP1_Distribution_Package).

3.5 Connecting to the discovery kit

To transfer the built binaries (application, device trees) to the STM32MP157F-DK2 discovery kit from your PC/laptop, you can transfer the binaries either by hotspot (https://wiki.st.com/stm32mpu/wiki/How_to_configure_a_wlan_interface_on_hotspot_mode) or via Wi-Fi connectivity (https://wiki.st.com/stm32mpu/wiki/How_to_setup_wifi_connection).

4 Building and running the example

The code can be built using simple Makefile utility for the starter package or using bitbake for the distribution package. For Python, no building/compiling is required but it is dependent on pyserial and pynmea2 package which needs to be installed.

4.1 Using Makefile (for starter package)

Step 1. Download the X-LINUX-GNSS1 package.

Step 2. Create a directory named “gnss”.

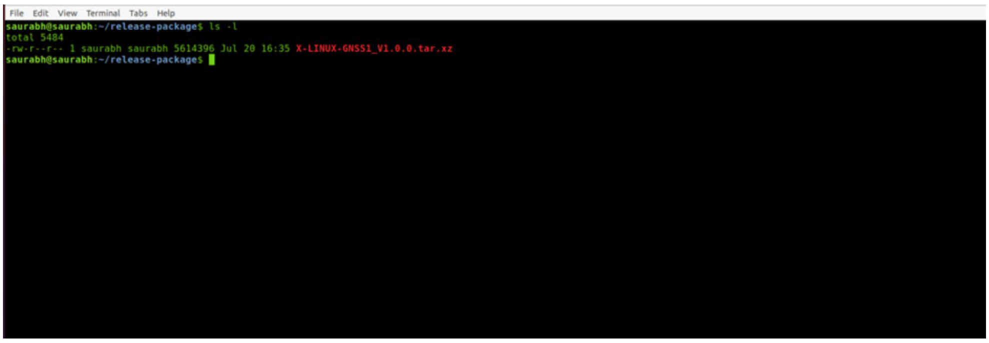
```
$mkdir gnss
$cd gnss
```

Step 3. Download or clone the package (X-LINUX-GNSS_V1.1.0.tar.xz) from www.st.com and extract it.

```
$tar xvf X-LINUX-GNSS_V1.1.0.tar.xz
```

You will get the X-LINUX-GNSS1_V1.1.0 folder.

Figure 13. Cloning the package



4.1.1 How to build the gnss_app

Step 1. Modify the device tree or copy it from the folder provided.

Step 2. Copy the dts file in the directory: X-LINUX-GNSS1_V1.1.0/ STM32MP1-DK2_Device_Tree/Source to the kernel source directory at <KERNEL SOURCE PATH>/ arch/arm/boot/dts/.

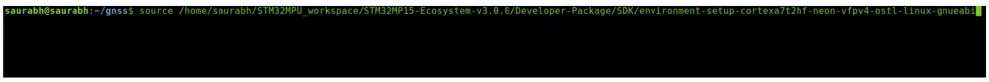
Step 3. Download the kernel sources as described in Section 3.3 .

```
$cd path-to/X-LINUX-GNSS1_V1.1.0/Application/Source/gnss_x_linux/Sources
$cp stm32mp157f-dk2.dts <KERNEL SOURCE PATH>/ arch/arm/boot/dts
```

Step 4. Source the path of the SDK (previously downloaded and installed).

```
$source <SDK PATH>/SDK/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi
```

Figure 14. Sourcing the SDK path



Step 5. Build the device tree.

```
$cd <KERNEL SOURCE PATH>
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
$make arch=ARM menuconfig
$make ARCH=arm uImage vmlinux dtbs LOADADDR=0xC2000040 (Optional)
$make ARCH=arm modules (Optional)
```

Step 6. Once the dtbs are built, copy them to the STM32MP157F-DK2 via hotspot or Wi-Fi.

```
$scp <KERNEL SOURCE PATH>/arch/arm/boot/dts/stm32mp157f_dk2.dtb root@192.168.72.1:/
boot
```

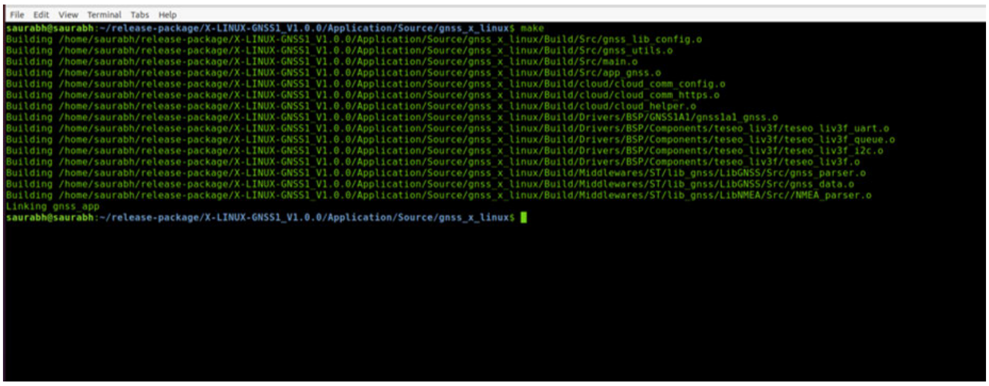
Step 7. Build the gnss_app.

```
$cd gnss
```

Step 8. Build the gnss_app. cd to the gnss_x_linux location.

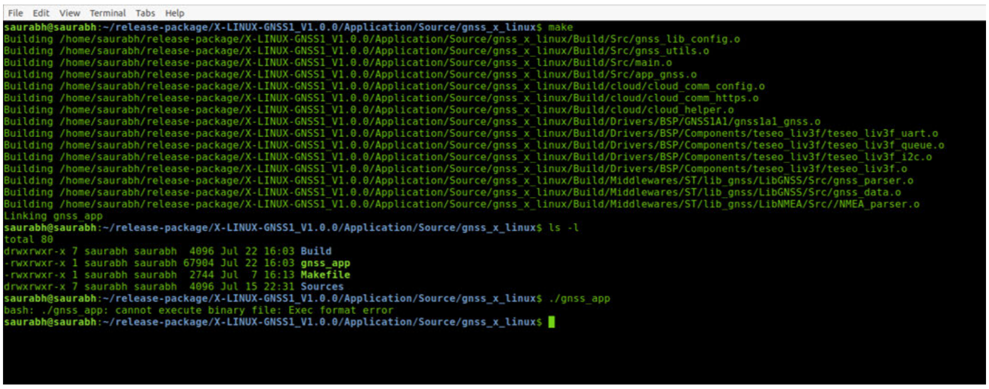
```
$cd path to/X-LINUX-GNSS1_V1.1.0/Application/Source/gnss_x_linux
```

Figure 15. STM32MP157F-DK2 - building the application



The executable (gnss_app) will be built in the same location of Makefile.

Figure 16. gnss_app created



Step 9. Transfer the application to STM32MP157F-DK2.

```
scp gnss_app root@192.168.72.1:/
```

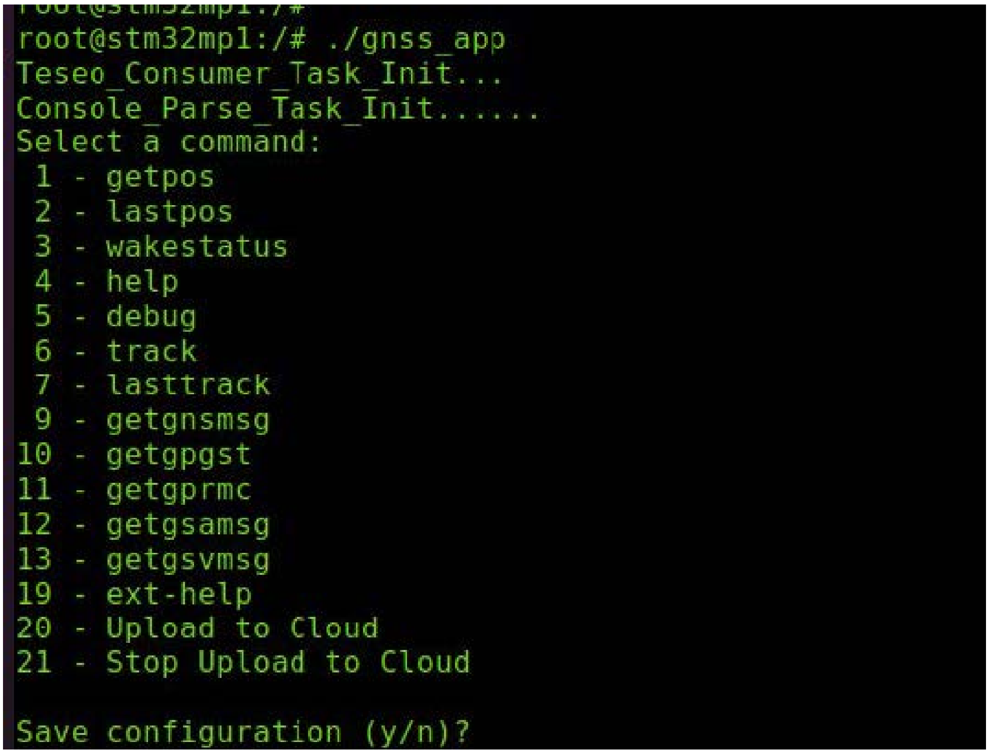
Step 10. On the STM32MP157F-DK2, sync and reboot.

```
$sync
$reboot
```

Step 11. Run the application.

```
$cd /
$./gnss_app
```

Figure 17. STM32MP157F-DK2 - running the application



Step 12. From the menu above, select 11 to get the GPS coordinates.

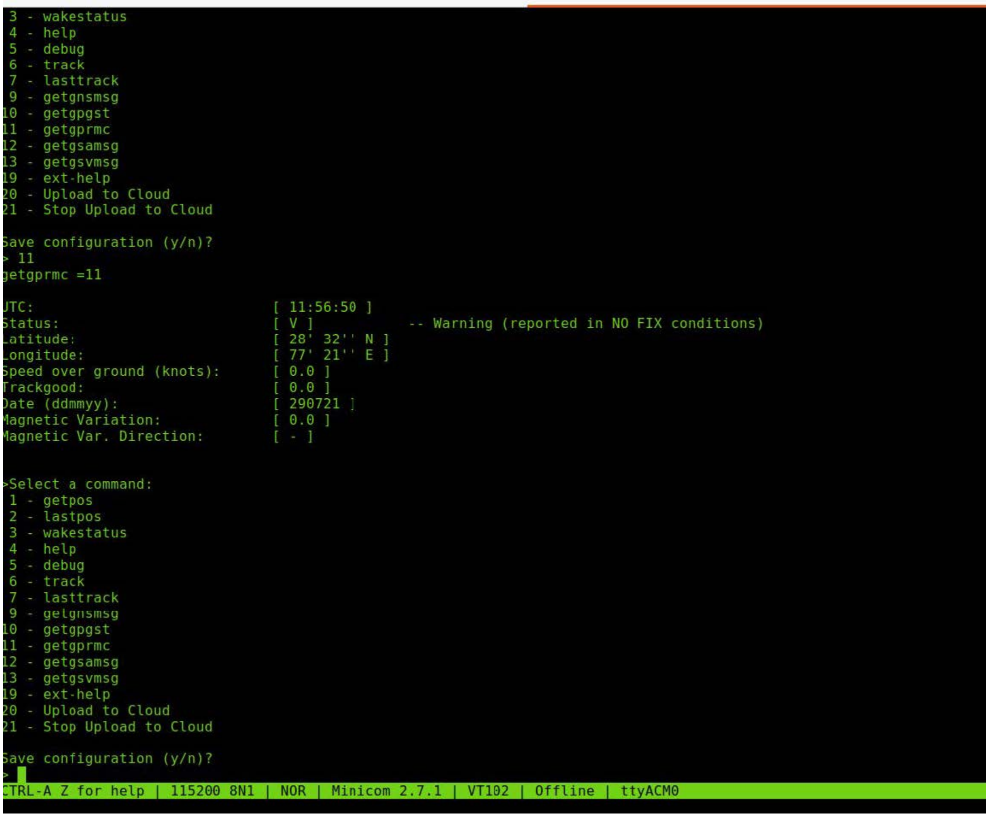


Figure 18. Selecting option 11 from the gnss_app

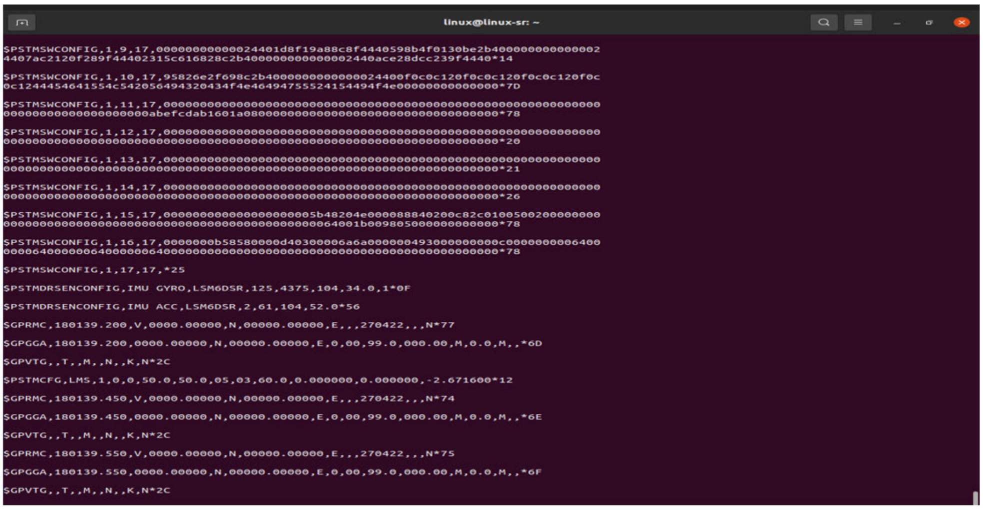
4.1.2 How to build C utility gnss_uart and gnss_i2c

Step 1. Enter the <Path to > /X-LINUX-GNSS1_V1.1.0\Application\Source\gnss_uart.
gnss_uart will be created in the same directory of Makefile.

Step 2. Transfer it to STM32MP157F-DK2 using scp.

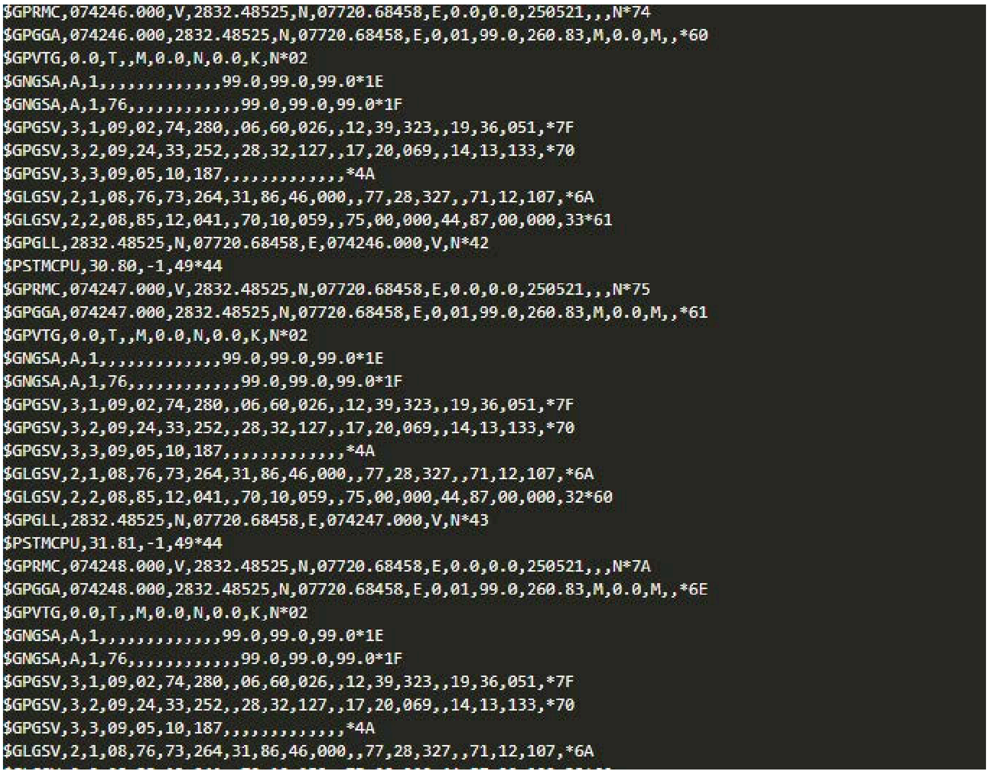
```
$cd /
$ ./gnss_uart_read
```

Figure 19. Running gnss_uart



```
$ ./gnss_i2c_read
```

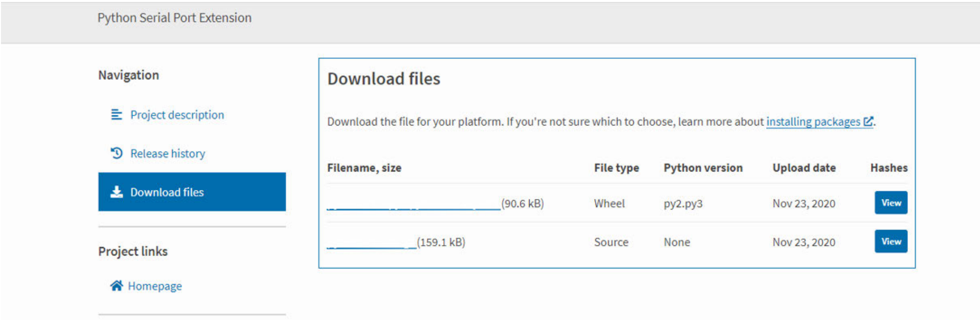
Figure 20. Running gnss_i2c



4.1.3 How to run the Python code

Step 1. Install the pyserial.

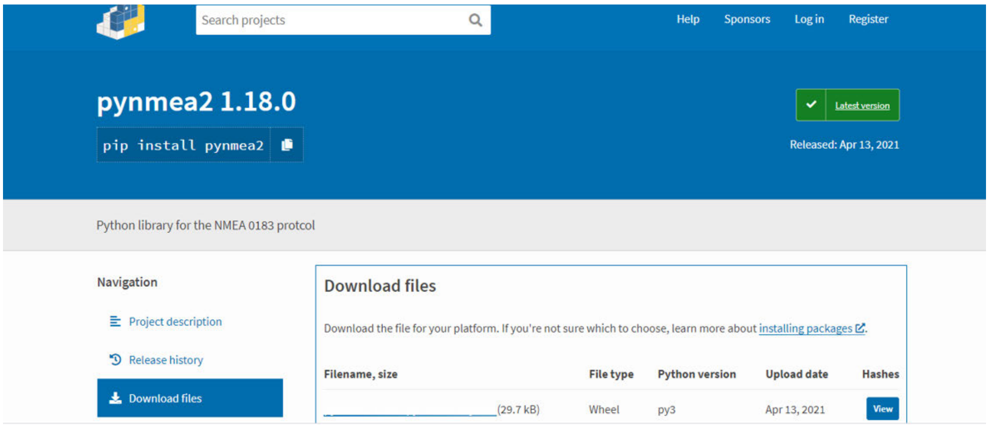
Figure 21. Installing Python dependencies - pyserial




```
$wget https://files.pythonhosted.org/packages/1e/7d/
ae3f0a63f41e4d2f6cb66a5b57197850f919f59e558159a4dd3a818f5082/pyserial-3.5.tar.gz
$tar xvf pyserial-3.5.tar.gz
$cd pyserial-3.5
$python setup.py install or python3 setup.py install
```

Step 2. Install pynmea2.

Figure 22. Installing Python dependencies - pynmea2




```
$wget https://files.pythonhosted.org/packages/88/b9/
a0fed4563f5c73eb8f4d7bb115a455863c5327ae824ac1772e2a4b1b95ee/pynmea2-1.18.0.tar.gz
$tar xvf pynmea2-1.18.0.tar.gz
$cd pynmea2-1.18.0
$python setup.py install or python3 setup.py install
```

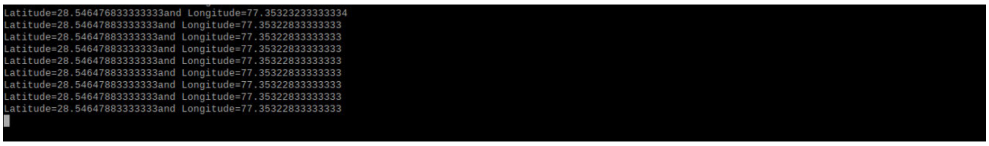
Step 3. Enter the Python folder and copy the gnss_pynmea2.py file to STM32MP157F-DK2.

```
$scp gnss_pynmea2.py root@192.168.72.1:/
```

Step 4. Run the Python example.

```
$cd /
$python3 gnss_pynmea2.py
```

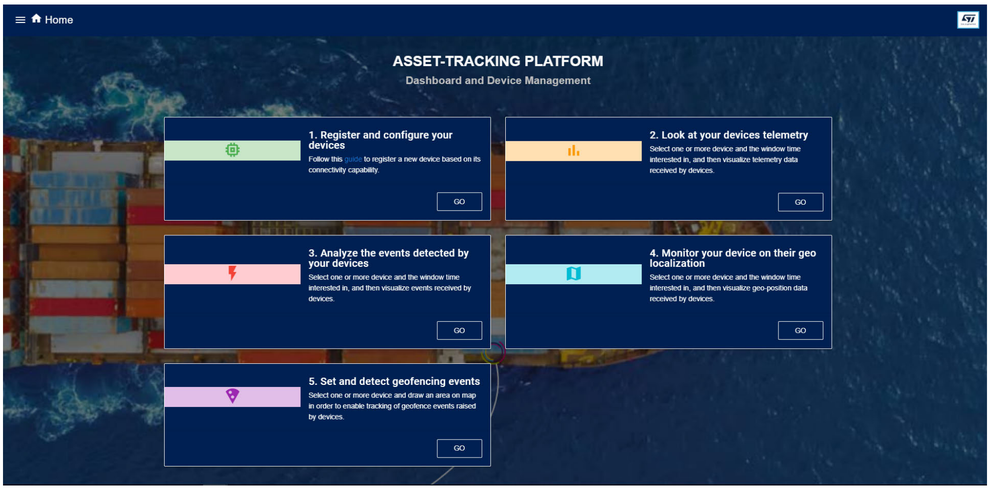
Figure 23. **Running the Python example**



4.1.4 Maps and asset tracking

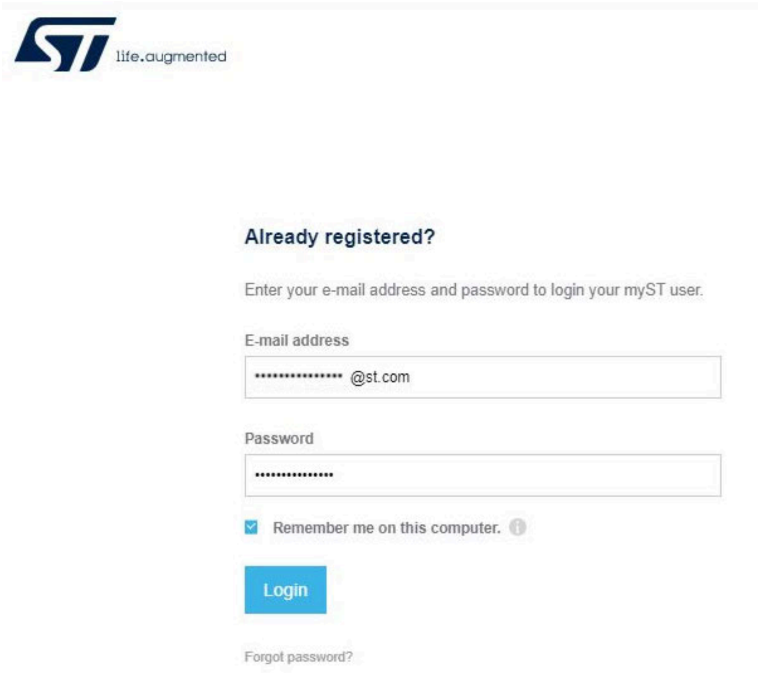
X-LINUX-GNSS1 provides an example to send GNSS data to the cloud over http. DSH-ASSETRACKING displays GNSS data in real-time.

Figure 24. **DSH-ASSETRACKING asset tracking dashboard homepage**



Step 1. Login to or create an account at https://dsh-assetracking.st.com/#/login.

Figure 25. DSH-ASSETRACKING login page



Step 2. Create a device (device name and device ID) from the [**Devices**] tab.

Figure 26. DSH-ASSETRACKING - Devices tab

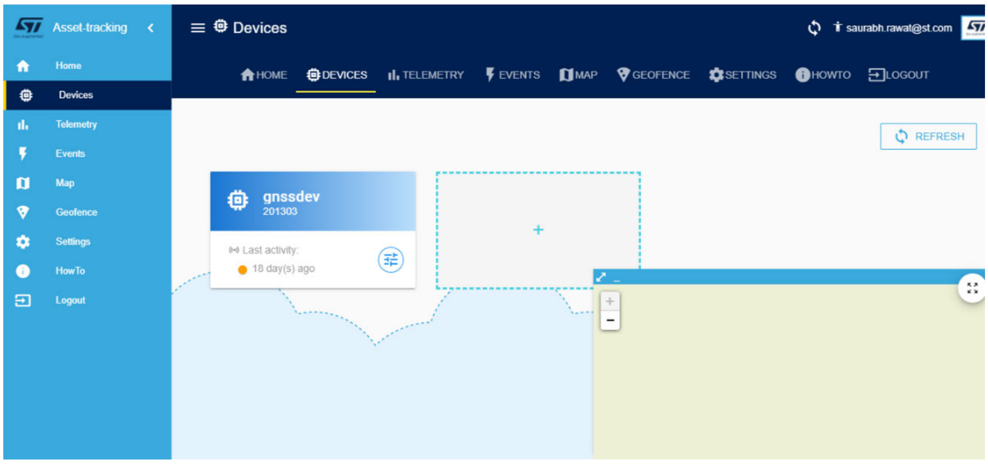
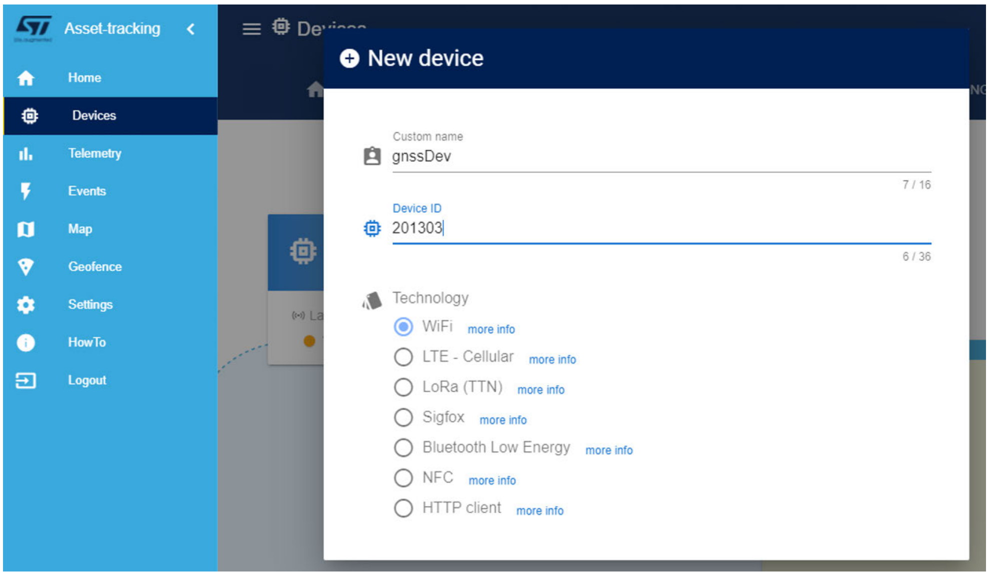
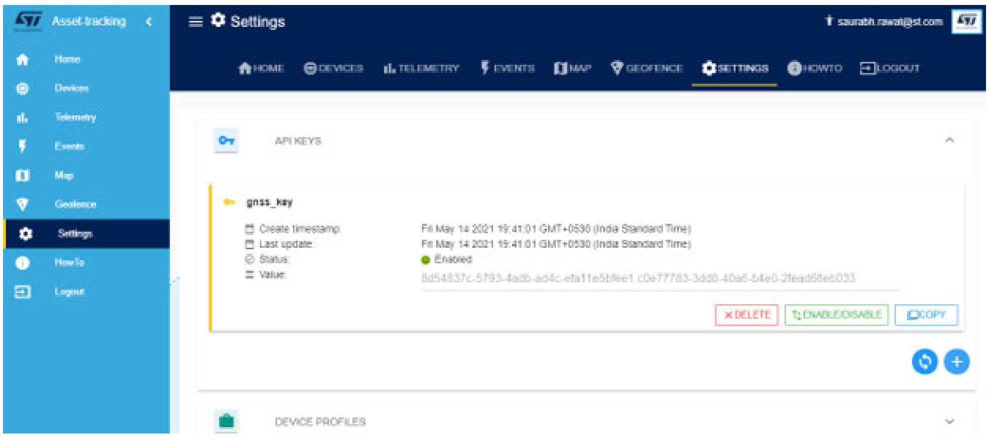


Figure 27. DSH-ASSETRACKING - device name and ID creation



Step 3. Create the API key which will be used to send data to DSH-ASSETRACKING.

Figure 28. DSH-ASSETRACKING - new API key generation



Step 4. Note down the end-point.

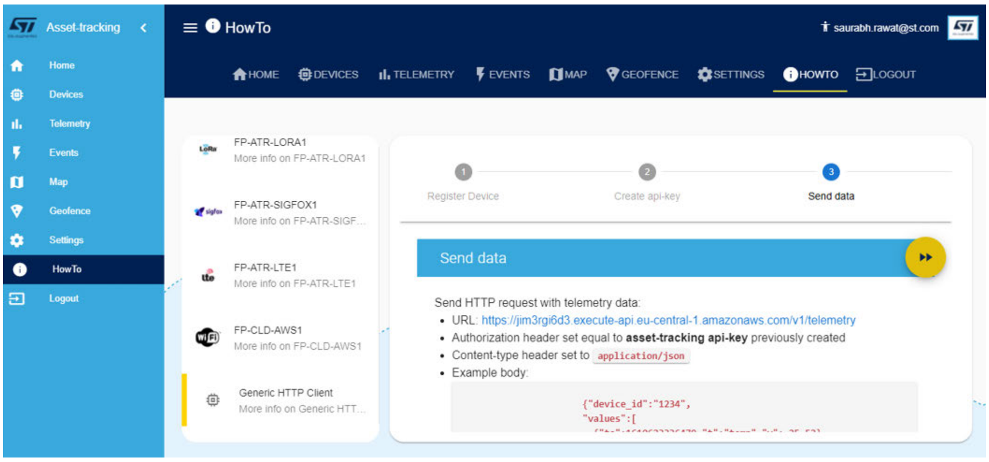


Figure 29. **DSH-ASSETRACKING - location of the end-point**

Step 5. Modify the data in the creds.conf file as per the end-point, device ID and the API key.

```
$cd /path-to/X-LINUX-GNSS1_V1.1.0/Application/Source/gnss_x_linux/Sources/cloud
$vi creds.conf
```

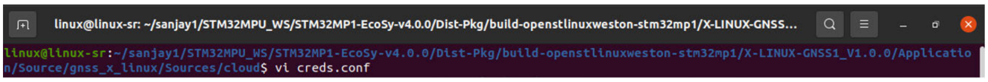


Figure 30. **Modifying creds.conf**

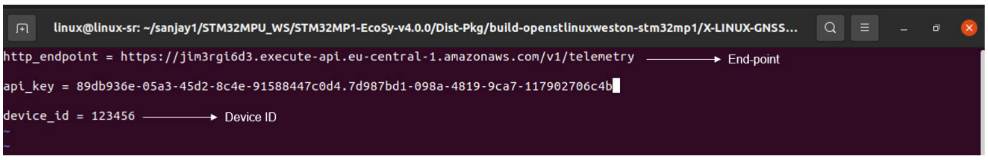


Figure 31. **Add credentials in the creds.conf file**

Step 6. Modify the data in the 'cloud_comm_https.c' file as per the device ID created for the device as:

```
pgps->device_id = 123456
$cd /path-to/X-LINUX-GNSS1_V1.1.0/Application/Source/gnss_x_linux/Sources/cloud
$vi cloud_comm_https.c
```

Figure 32. **Modifying cloud_comm_https.c**

Figure 33. **cloud_comm_https.c file modified**

Step 7. Build the application (if the creds file only has been modified, this step is not required).

Step 8. Repeat the steps done to build and deploy the gnss_app.

Step 9. Enter option 20 to upload the data to the cloud and make sure the STM32MP157F-DK2 is connected to the Internet.

Figure 34. Running gnss_app and enabling cloud upload (option 20)

```
Save configuration (y/n)?
> 11
getgprmc =11

UTC:                          [ 08:19:17 ]
Status:                       [ V ]          -- Warning (reported in NO FIX conditions)
Latitude:                     [ 28' 32'' N ]
Longitude:                    [ 77' 21'' E ]
Speed over ground (knots):    [ 0.0 ]
Trackgood:                    [ 0.0 ]
Date (ddmmyy):                [ 110621 ]
Magnetic Variation:           [ 0.0 ]
Magnetic Var. Direction:      [ - ]


>Select a command:
 1 - getpos
 2 - lastpos
 3 - wakestatus
 4 - help
 5 - debug
 6 - track
 7 - lasttrack
 8 - getfwver
 9 - getgnsmsg
10 - getgpgst
11 - getgprmc
12 - getgsamsg
13 - getgsvmsg
19 - ext-help
20 - Upload to Cloud
21 - Stop Upload to Cloud
```

You will get the below logs and the live tracking on the DSH-ASSETRACKING.

Figure 35. Gnss_app sending data to DSH-ASSETRACKING over http

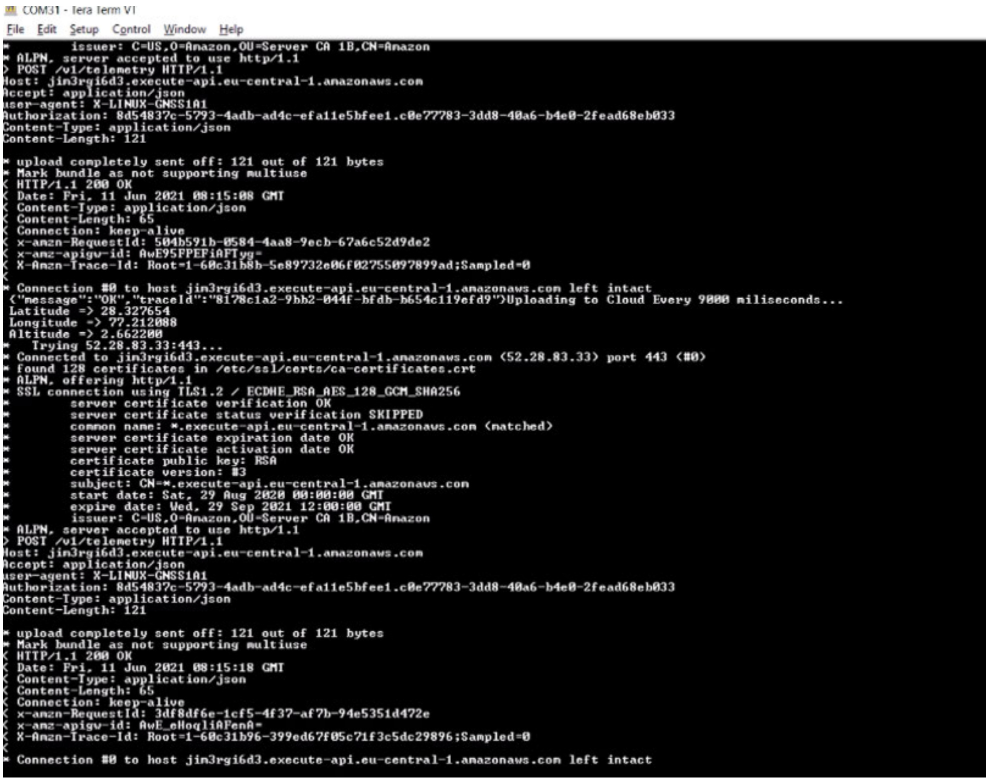
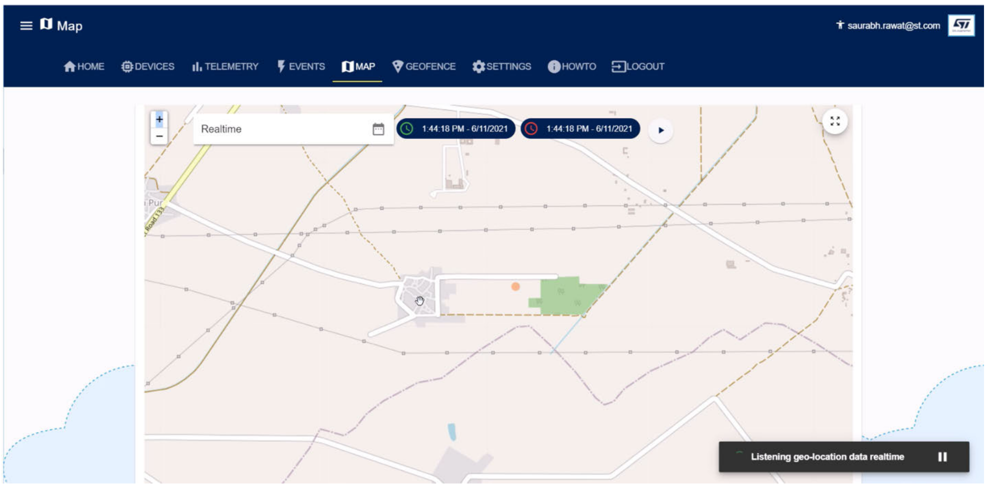


Figure 36. Gnss_app data in real-time



4.2 Using BitBake (for distribution package)

Step 1. Download the distribution package and do a bitbake.

```
$DISTRO=openstlinux-weston MACHINE=stm32mp1 source layers/meta-st/scripts/
envsetup.sh
$ bitbake st-image-weston
```

Step 2. Accept the EULA by typing "y" and pressing enter.

Figure 37. End user license agreement

After accepting, you will access the build directory.

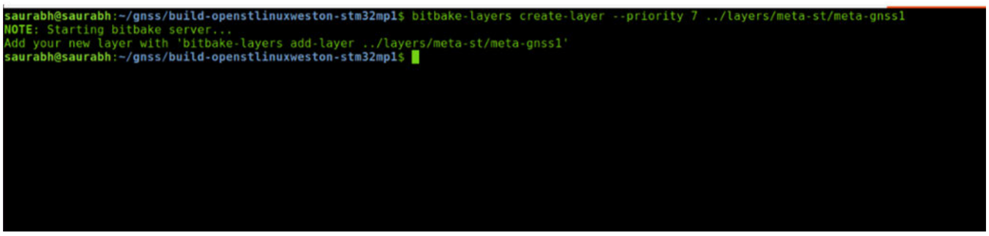
Step 3. Download the gnss package from www.st.com in the build directory as shown below.

Figure 38. gnss package download

Step 4. Create a layer (meta-gnss1).

```
$bitbake-layers create-layer --priority 7 ../layers/meta-st/meta-gnss1
```

Figure 39. meta-gnss1 layer creation



Step 5. Add a layer (recipes-gnss1).

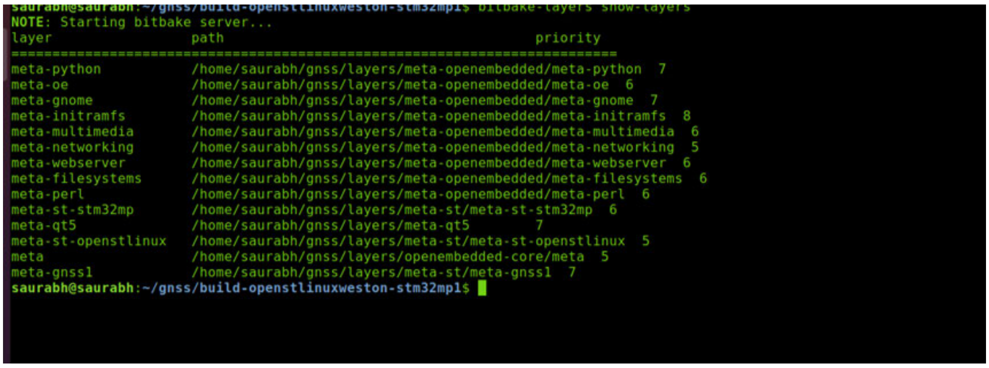
```
$ bitbake-layers add-layer ../layers/meta-st/meta-gnss1
```

Figure 40. Adding meta-gnss1 layer

You can see the added layer by typing:

```
$bitbake-layers show-layers
```

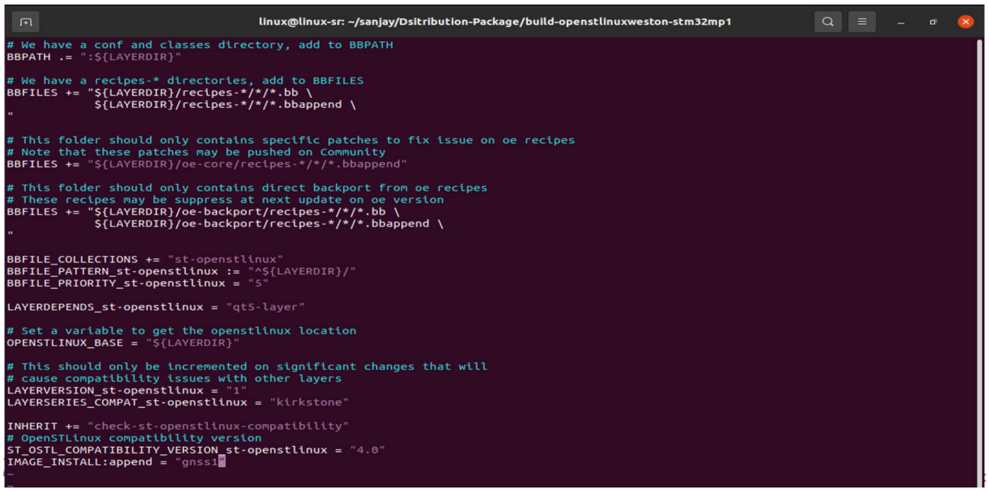
Figure 41. Showing all meta layers



Step 6. Use IMAGE_INSTALL:append line at the end of the layer.conf.

```
$vi ../layers/meta-st/meta-st-openstlinux/conf/layer.conf
```

Figure 42. IMAGE_INSTALL:append



Step 7. Delete the meta-gnss1 created by the tool.

```
$  rm -rf ../layers/meta-st/meta-gnss1/
```

Figure 43. Deleting default meta-gnss1

Step 8. Copy the layer provided by the X-LINUX-GNSS1 package.

```
$ cp  -rf X-LINUX-GNSS1_V1.1.0/Recipe/meta-gnss1/   ../layers/meta-st/
```

Figure 44. Copying the X-LINUX-GNSS1 meta-st layer

Step 9. Add the Sources path (location containing CMakeLists.txt) inside gnss1_0.1.bbappend as shown below.

```
path-to/openstlinux-5.15-yocto-kirkstone-mp1-v22.06.15/Distribution-
Package/build-openstlinuxweston-stm32mp1/x-linux-gnss/X-LINUX-GNSS_V1.X.Y/
Application/Source/gnss_x_linux/Sources
```

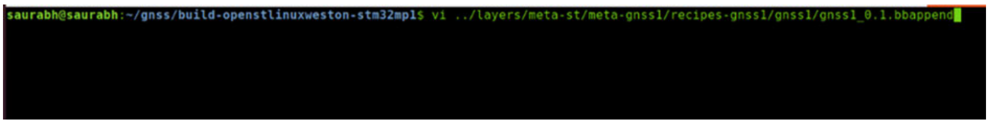


Figure 45. Modifying bbapend file

which is inside layers/meta-st/meta-gnss1/recipes-gnss1/gnss1/gnss1_0.1.bbappend:

```
$vi ../layers/meta-st/meta-gnss1/recipes-gnss1/gnss1/gnss1_0.1.bbappend
```

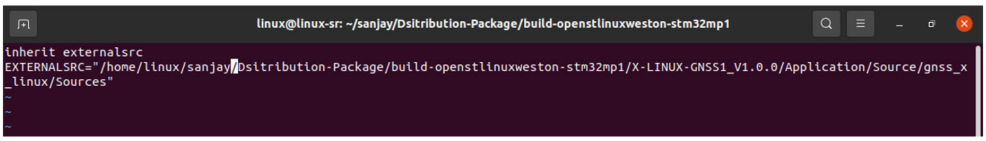


Figure 46. Adding external source path in bbapend file

Step 10. Update “dunfell” with “kirkstone” inside ‘layers/meta-st/meta-gnss1/conf/layer.conf’.

```
$ vi ../layers/meta-st/meta-gnss1/conf/layer.conf
```

Figure 47. Modifying inside ‘..layers/meta-st/meta-gnss1/conf/layer.conf’

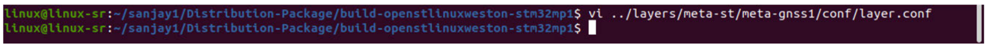
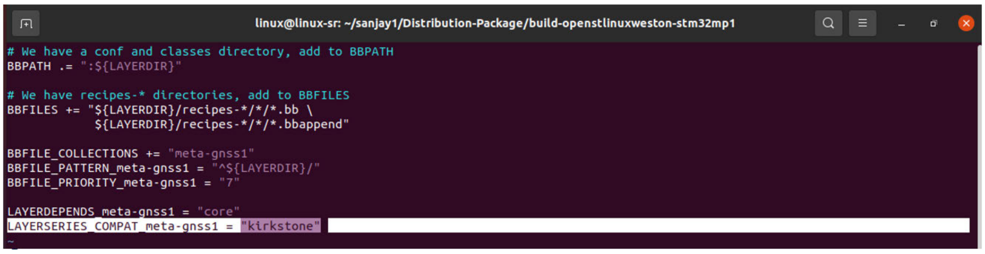


Figure 48. Update “dunfell ” with “ kirkstone”



Step 11. Change ‘_’ with ‘:’ inside “layers/meta-st/meta-gnss1/recipes-kernel/linux/linux -stm32mp_%.bbappend” file

```
SRC_URI:append = " file://gnss1.patch"
FILESEXTRAPATHS:prepend := "${THISDIR}/${PN}:"
PACKAGE_ARCH = “${MACHINE_ARCH}”
$ vi ../layers/meta-st/meta-gnss1/recipes-kernel/linux/linux-stm32mp_%.bbappend
```

Figure 49. Modifying inside ‘../layers/meta-st/meta-gnss1/recipes-kernel/linux/linux-stm32mp_%.bbappend’

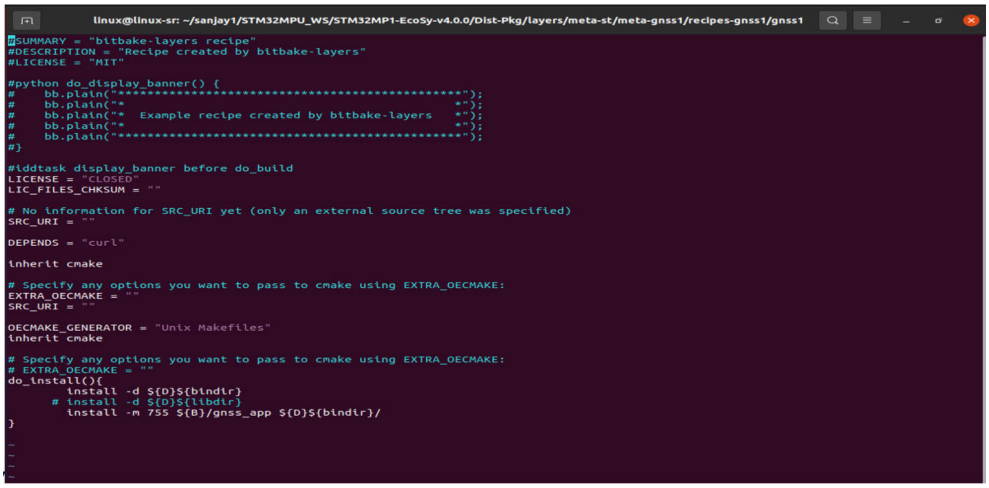
Figure 50. Updating ‘_’ with ‘:’

Step 12. Delete “install -d ${D}${libdir}” inside “ layers/meta-st/meta-gnss1/recipes- gnss1/gnss1/gnss1_0.1.bb”.

```
# install -d ${D}${libdir}
$ vi ../layers/meta-st/meta-gnss1/recipes-gnss1/gnss1/gnss1_0.1.bb
```

Figure 51. **Modifying inside ‘../layers/meta-st/meta-gnss1/recipes-gnss1/gnss1/gnss1_0.1.bb’**

Figure 52. **Delete “install -d ${D}${libdir}”**



Step 13. Build the ST image.

```
$ bitbake st-image-weston
```

Figure 53. **Building the ST image**

New images (including FlashLayout_sdcard_stm32mp157f-dk2-trusted.tsv and FlashLayout_sdcard_stm32mp157f-dk2-trusted) will be created in the tmp-glibc/deploy/images/stm32mp1/ directory.

```
$cd tmp-glibc/deploy/images/stm32mp1/
```

Step 14. Flash the binary following the instructions at https://wiki.st.com/stm32mpu/wiki/STM32MP15_Discovery_kits_Starter_Package#Image_flashing.

Step 15. Check if the file below is present on the discovery kit.

```
$ ls -l /dev/ttySTM2
```

Step 16. Run the application.

```
$ /usr/bin/gnss_app
```

or

```
gnss_app
```

Revision history

Table 3. Document revision history

Date	Revision	Changes
02-Aug-2021	1	Initial release.
01-Dec-2022	2	Updated Section 2 Hardware setup, Section 4.1.4 Maps and asset tracking, and Section 4.2 Using BitBake (for distribution package).

Contents

List of tables

List of figures

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2022 STMicroelectronics – All rights reserved