

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

Introduction

The X-LINUX-GNSS1 is a software package for STM32MP157F-DK2 Board. The software runs on STM32MP1 MPU and includes user space application, device tree for the Teseo-LIV3F global navigation satellite system (GNSS) device, library for the NMEA (National Marine Electronics Association,) protocol support and POSIX Thread for task scheduling to ensure better asynchronous message parsing.

The software comes with sample implementations of user space applications running on the STM32MP1 board with [X-NUCLEO-GNSS1A1](#) connected to the Arduino Connector on UART and I2C. NMEA library is used to parse the GPS NMEA data that provide several GPS parameters like – latitude, longitude, elevation, speed etc. It contains platform specific Device Tree Modification for STM32MP1 as well.

The source code is designed for portability across a wide range of processing units running Linux .

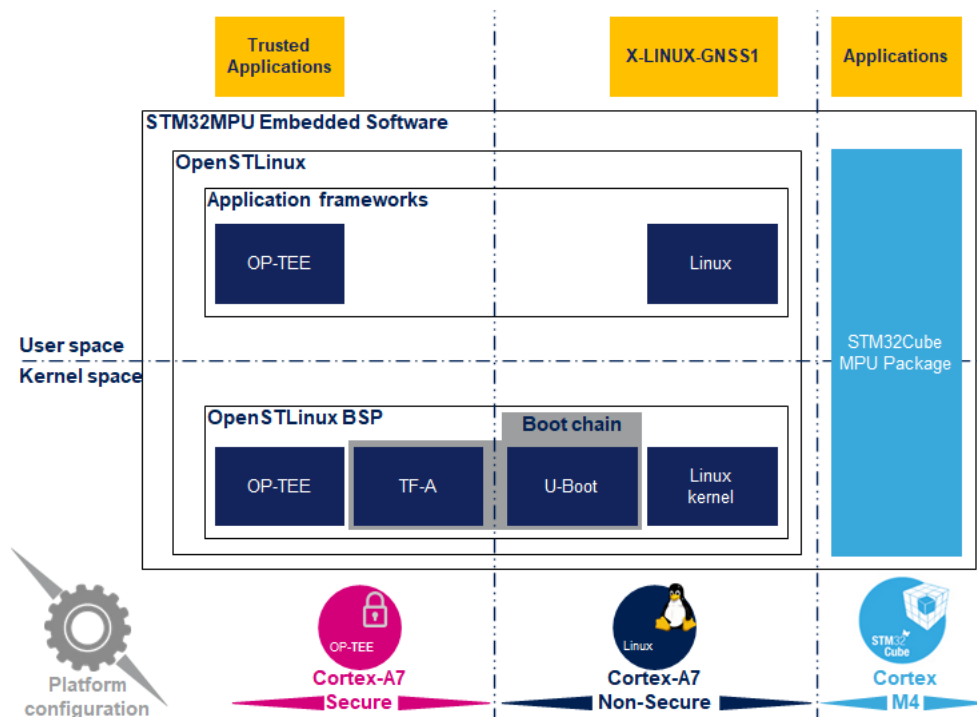


Figure 1 X-LINUX Software Architecture Diagram

1. X-LINUX-GNSS1 overview

The X-LINUX-GNSS1 software provides user space application running on STM32MP157F-DK2 for X-NUCLEO-GNSS1A1 expansion board based on the [Teseo-LIV3F](#) tiny Global Navigation Satellite System (GNSS) module. The software package has 3 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 be run independently to fetch the GNSS NMEA data from the X-NUCLEO-GNSS1A1 over UART and I2C.

The device tree for STM32MP157F-DK2 board has been modified to configure the UART7 and I2C5 on the Arduino Connector. For UART, underlying dev/ttySTM2 is enabled while for I2C, /dev/i2c-1 is enabled. The X-LINUX-GNSS1 software interacts with the lower layer peripheral drivers (I2C and UART) using [termios](#) for UART and file descriptor reading for I2C peripheral.

It uses POSIX Thread to run two parallel tasks – Consumer Task and Console Task. Consumer Task fetches the NMEA data, parse it and populates the NMEA data structure. Console Task reads the input from the user and provides the information from the populated NMEA Data structure like position, speed, elevation, wakeup status, etc. based on the provided inputs.

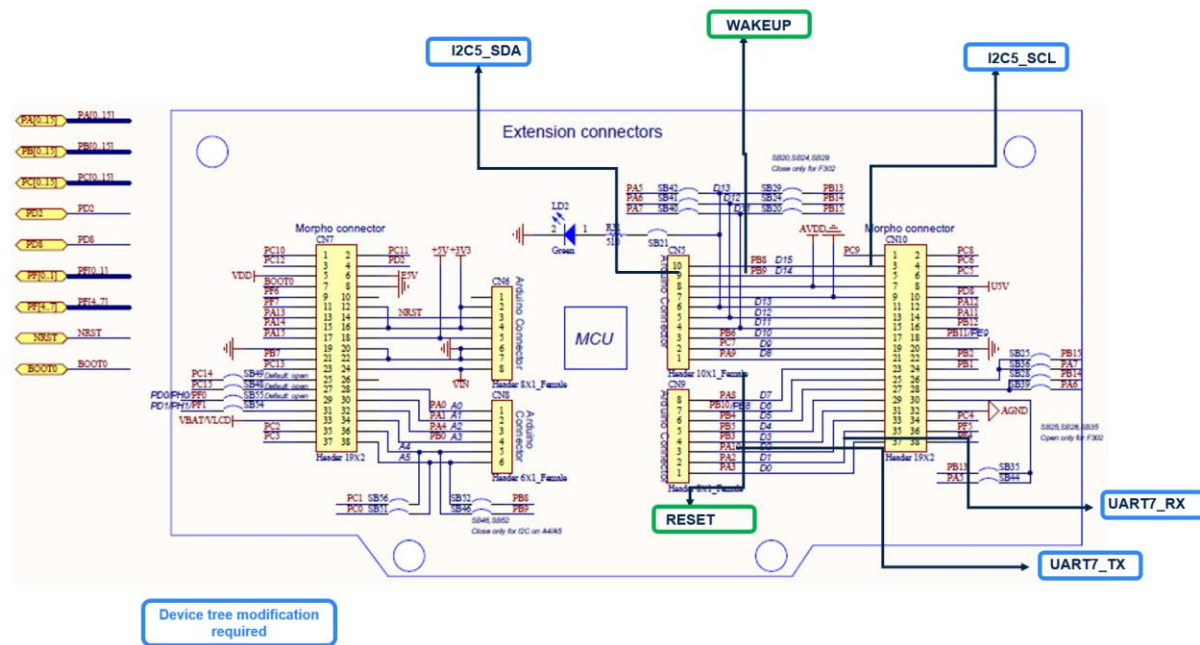


Figure 2 Hardware Connections

1.1 Features

Below are the main features of the X-LINUX-GNSS1

- Standalone applications to read the NMEA data over UART and I2C. (gnss_uart and gnss_i2c)
- Complete software to build applications using Teseo-LIV3F GNSS device on Linux.
- Middleware for the NMEA protocol
- POSIX Thread task scheduling to ensure better asynchronous message parsing.
- Easy portability across different Linux Platforms

- Sample application example to retrieve and parse GNSS data and send to [ST Asset Tracking Dashboard](#) for live tracking.
- Python Example(gnss_pynmea2.py)
- Free, user-friendly license terms

1.2 Architecture

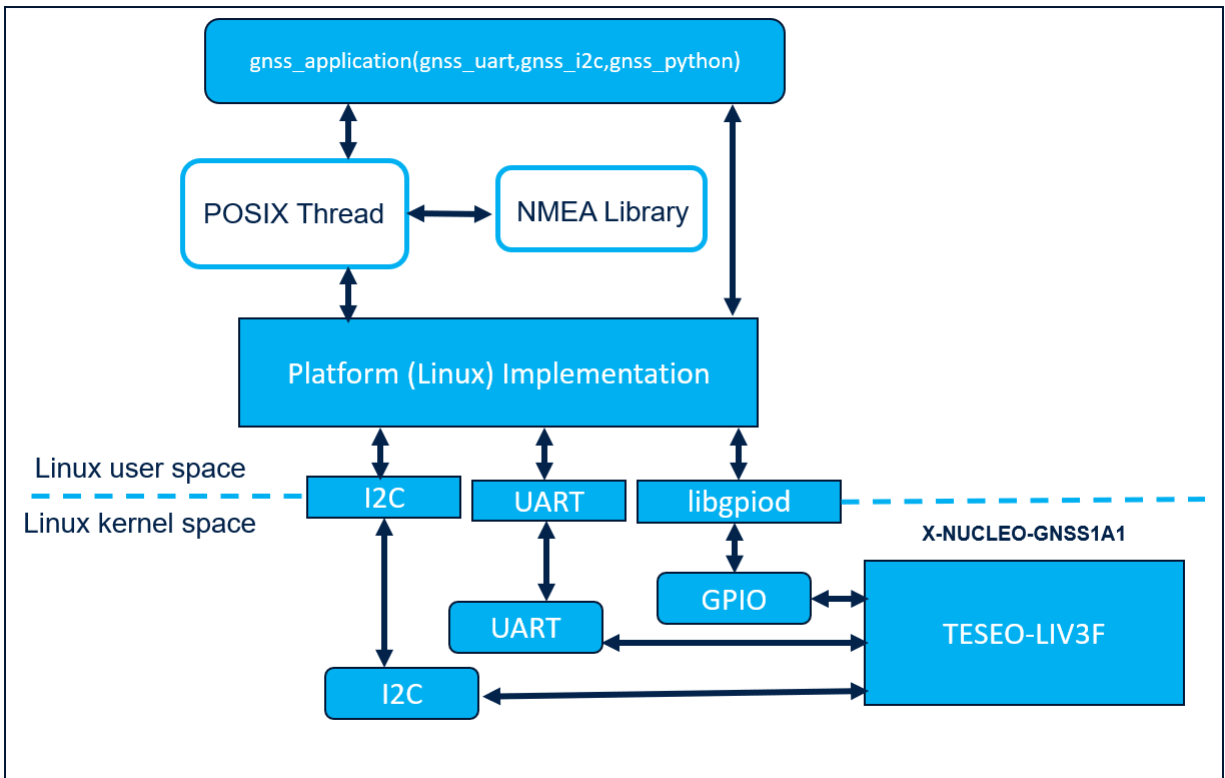
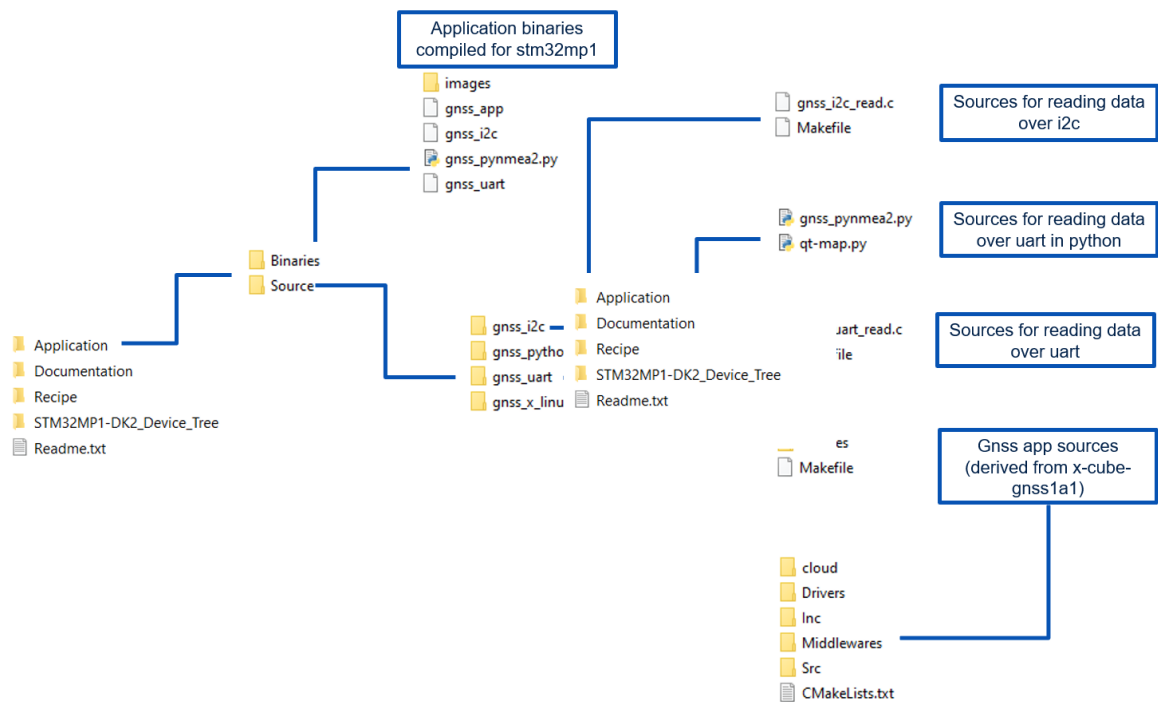
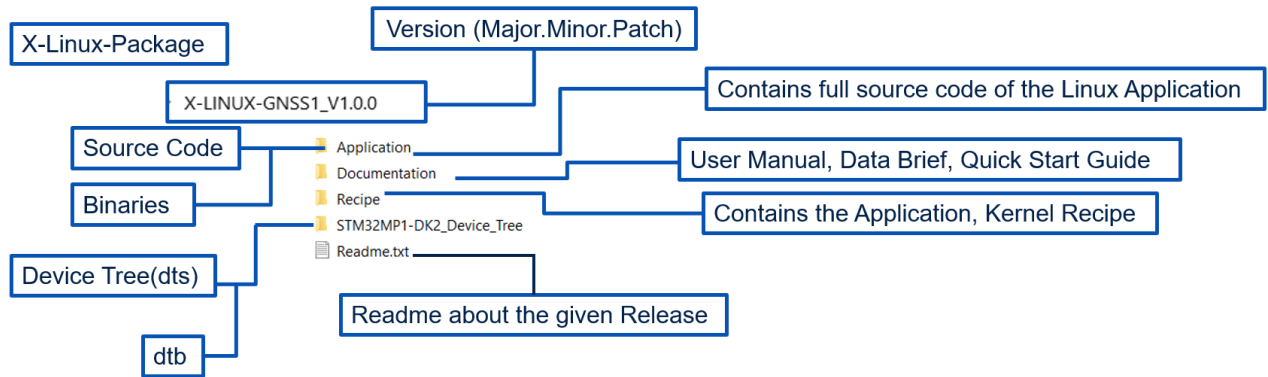


Figure 3 Application Architecture

1.3 Software Package Structure

Below is the folder structure of the release package. The release package has Linux user application C examples, python example, device tree and Yocto layer recipe. User can run any of the application independently inside the Application folder to retrieve the GNSS NMEA data.



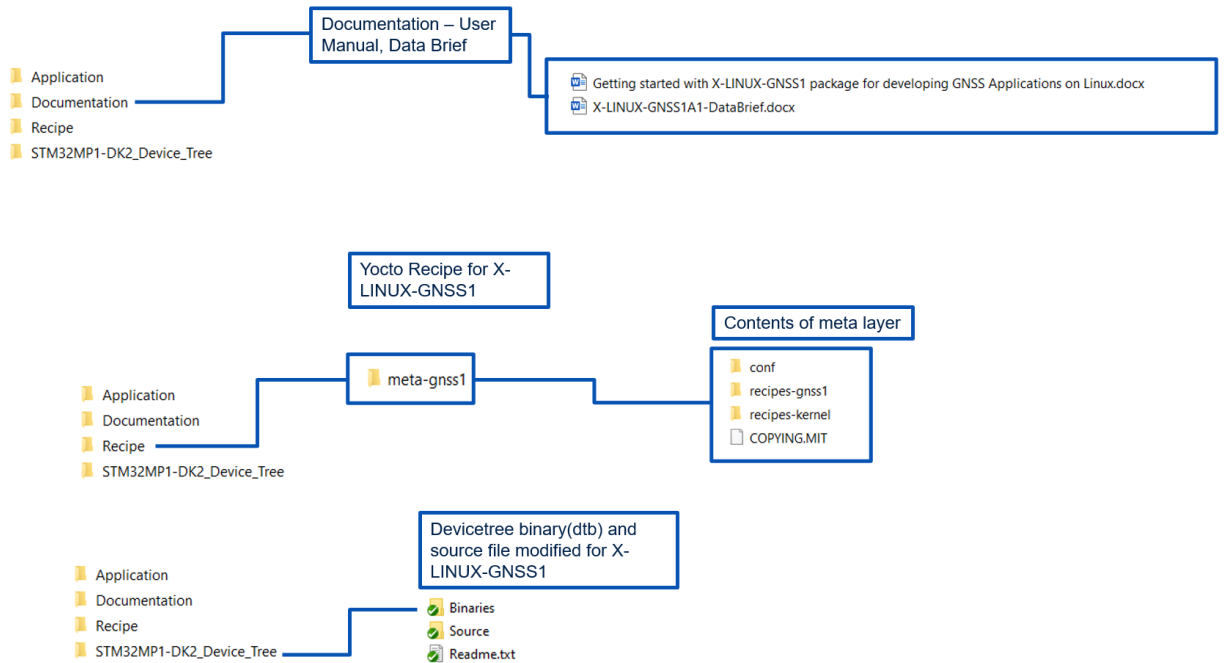


Figure 6 GNSS Other Folder Structures

1.3.1 GNSS_APP

This Application accesses the GPS data over UART (/dev/ttySTM2) and I2C(/dev/i2c-1) interface. The settings for enabling the UART and I2C is provided separately in the device tree file folder. It has also provision to upload the data to the cloud (ST Asset Tracking Dashboard)

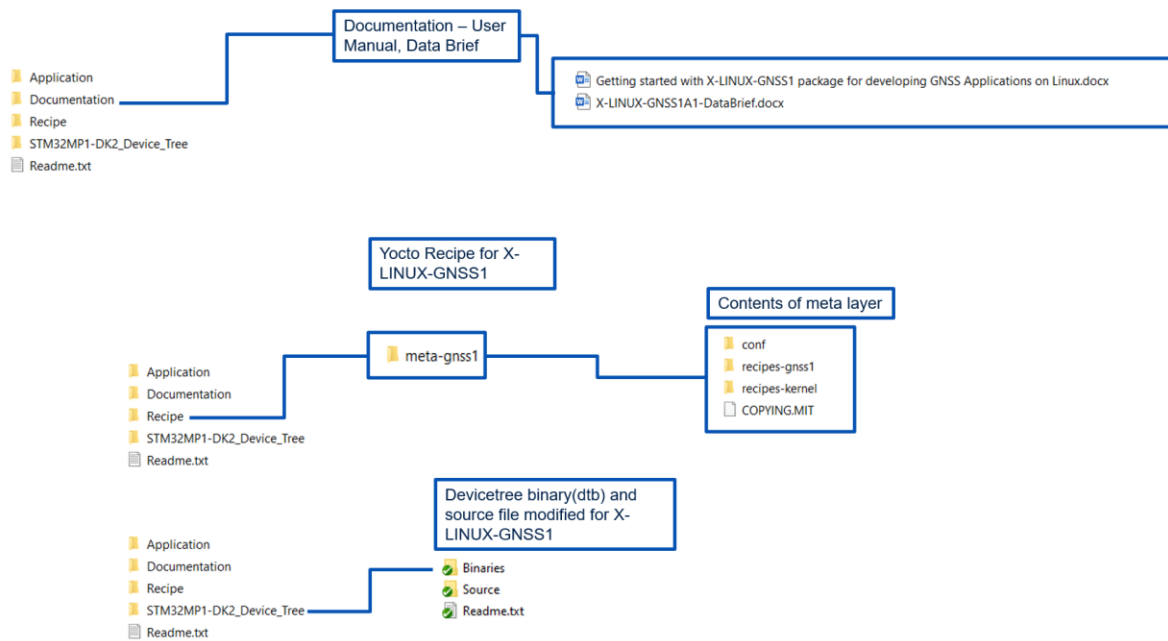


Figure 7 gnss_app

1.3.2 C Utility

This Application is Linux user space C application to read the data from the UART (/dev/ttySTM2) and I2C (/dev/i2c-1) interface.

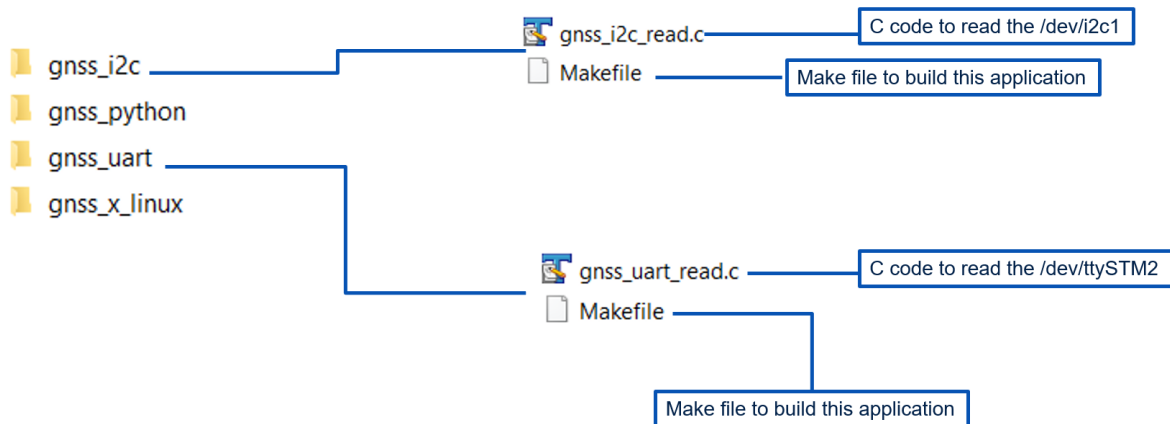


Figure 8 C utility – gnss-uart and gnss-i2c

1.3.3 Python Code

This Application is a basic python code to read the data from UART. It uses [pyserial](#) and [pynmea2](#) library. You need to have these two dependencies installed before using this application.

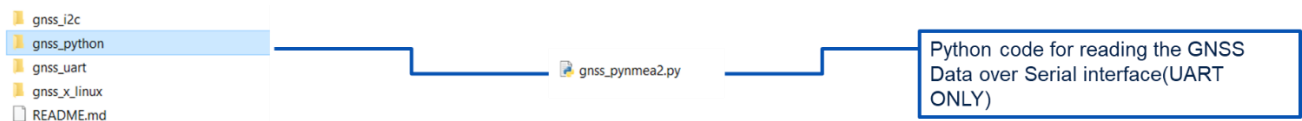


Figure 9 Python Code

2. Hardware Setup

The Software is compatible with [X-NUCLEO-GNSS1A1](#) board which can be directly plugged on the Arduino connector. Keep the jumper settings as shown below in the figure 10 and plug it on Arduino connectors of the STM32MP157F-DK2 board.

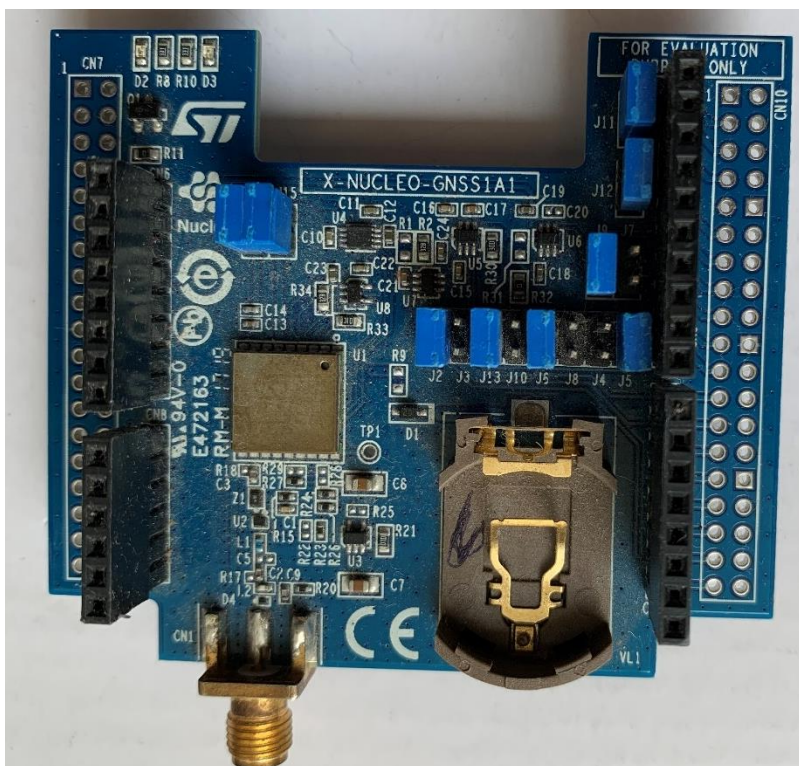


Figure 10 Hardware Setup – Jumper Settings

Signal	Arduino	Nucleo 64	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 1 Jumper Settings

Connect the GPS/GLONASS/Beidou antenna provided with X-NUCLEO-GNSS1A1. Keep the antenna outdoor for better reception. The board will receive power from the STM32MP157F-DK2 board from USB type C cable.

3. Software Setup

The section describes the software setup, which is required for building, flashing, transferring, and running the GNSS application.

3.1 Recommended PC prerequisites

A Linux® PC running under Ubuntu® 18.04 or 20.04 is to be used. The developer can follow the below link.

https://wiki.st.com/stm32mpu/wiki/PC_prerequisites

3.2 Installing the SDK

This is required to build the application package. The package contains the binaries which you can transfer using scp command. In case customization is needed in the Application, installing SDK will help in building it. The developer can follow the below link.

https://wiki.st.com/stm32mpu/wiki/Getting_started/STM32MP1_boards/STM32MP157x-DK2/Develop_on_Arm_Cortex-A7/Install_the_SDK

3.3 Downloading the kernel Sources (Developer Package)

This is required to build the device tree. The package already contains the binaries(dtb) which can be transferred using scp command. The developer can follow the below link.

https://wiki.st.com/stm32mpu/wiki/Getting_started/STM32MP1_boards/STM32MP157x-DK2/Develop_on_Arm_Cortex-A7/Modify_rebuild_and_reload_the_Linux_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 creating STM32MP1 images which has GNSS application and device tree settings embedded.

The developer can follow the below link to download the distribution package.

https://wiki.st.com/stm32mpu/wiki/STM32MP1_Distribution_Package

3.5 Connecting to the Board.

This is required to transfer the built binaries (application, device trees) to the STM32MP157F-DK2 board from the development PC. The developer can transfer the binaries either by Hotspot method (https://wiki.st.com/stm32mpu/wiki/How_to_configure_a_wlan_interface_on_hotspot_mode) or using the Wi-Fi connectivity (https://wiki.st.com/stm32mpu/wiki/How_to_setup_wifi_connection)

4. Building and Running the example

This section explains the method to build and run the software package. The code can be built using simple Makefile utility for Starter Package or using bitbake for Distribution package. For python, no building/compiling is required but it is dependent on pyserial and pynmea2 package which needs to be installed.

Below conventions are used below:

#Descriptive comments	Comment describing steps
\$command	Development or Host PC/machine command prompt. Text after \$ is command
\$command	STM32MP1 command prompt. Text after \$ is command
STM32MP1	STM32MP157F-DK2 Board

4.1 Using Makefile (For Starter Package)

Download the X-LINUX-GNSS1 package as a first step. Create a directory by name “gnss”

```
$mkdir gnss
```

```
$cd gnss
```

```
# Download or clone the package (X-LINUX-GNSS_V1.0.0.tar.xz) from www.st.com and extract it
```

```
$tar xvf X-LINUX-GNSS_V1.0.0.tar.xz
```

You will get X-LINUX-GNSS1_V1.0.0 folder.

```

saurabh7@vmi613801:~/gnss$ ls
x-linux-gnssal
saurabh7@vmi613801:~/gnss$ cd x-linux-gnssal/
saurabh7@vmi613801:~/gnss/x-linux-gnssal$ ls
README.md  X-LINUX-GNSS1_V1.0.0
saurabh7@vmi613801:~/gnss/x-linux-gnssal$ ls -l
total 8
-rw-rw-r-- 1 saurabh7 saurabh7 1115 Jun 26 22:57 README.md
drwxrwxr-x 6 saurabh7 saurabh7 4096 Jun 26 22:57 X-LINUX-GNSS1_V1.0.0
saurabh7@vmi613801:~/gnss/x-linux-gnssal$

```

Figure 11 Cloning the Package

4.1.1 gnss_app

For running the gnss_app, the below steps are to be followed.

1. Modify the device tree or copy it from the folder provided (see below where?)
2. Build the device tree and transfer it to STM32MP157F-DK2
3. Build the gnss_app
4. Transfer the gnss_app executable over Wi-Fi or hotspot to STM32MP157F-DK2 board

Modify the device tree or copy it from the folder provided (see below)

#Copy the dts file in the directory: X-LINUX-GNSS1_V1.0.0/ STM32MP1-DK2_Device_Tree/Source to the kernel source directory at <KERNEL SOURCE PATH>/ arch/arm/boot/dts/

Download the Kernel Sources as described in section 3.1.2.

```
$cd path-to/X-LINUX-GNSS1_V1.0.0/Application/Source/gnss_x_linux/Sources
```

```
$cp stm32mp157f-dk2.dts <KERNEL SOURCE PATH>/ arch/arm/boot/dts
```

#Source the path of the SDK . You have already downloaded and Installed the SDK in steps above. – 3.1 .Source
<SDK PATH>/SDK/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi

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

```

saurabh@saurabh:~/gnss$ source /home/saurabh/STM32MPU_workspace/STM32MP15-Ecosystem-v3.0.0/Developer-Package/SDK/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi

```

#Build the device tree

```
$cd <KERNEL SOURCE PATH>
```

```
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

```
$make arch=ARM menuconfig
```


\$reboot

#Running the Application

\$cd /

\$/gnss_app

```
Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Dec 23 2019, 02:06:26.
Port /dev/ttyACM0, 17:23:02

Press CTRL-A Z for help on special keys

♦
root@stm32mp1:~#
root@stm32mp1:~#
root@stm32mp1:~#
root@stm32mp1:~# cd /
root@stm32mp1:~#
root@stm32mp1:~# ./gnss_app
Teseo_Consumer_Task_Init...
Console_Parse_Task_Init.....
Select a command:
 1 - getpos
 2 - lastpos
 3 - wakestatus
 4 - help
 5 - debug
 6 - track
 7 - lasttrack
 8 - getfwver
 9 - getgsmmsg
10 - getpgst
11 - getprmc
12 - getsamsg
13 - getsvmsg
19 - ext-help
20 - Upload to Cloud
21 - Stop Upload to Cloud

Save configuration (y/n)?
>
```

Figure 13 STM32MPU - Run Application

```

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 - getfwrver
9 - getgnsmsg
10 - getpgpst
11 - getgprmc
12 - getgsamsg
13 - getgsvmmsg
19 - ext-help
20 - Upload to Cloud
21 - Stop Upload to Cloud

```

Figure 14 gnss_app Selecting the options - Select 11 to get the GPS Co-ordinates

4.1.2 C Utility

The steps are same as building gnss_app

Enter the <Path to > /X-LINUX-GNSS1_V1.0.0\Application\Source\gnss_uart and do make.

gnss_uart will be created in the same directory where Makefile is present. Transfer it to STM32MP1-DK2 using scp .

```

$cd /
$./gnss_app

```

```

Read 61:$GPGSV,3,2,09,24,33,250,,28,32,128,,17,19,070,,14,12,133,*7E

Read 1:

Read 40:$GPGSV,3,3,09,05,11,187,,,,,,,,,,,,,*48

Read 1:

Read 63:$GLGSV,2,1,08,76,72,261,30,86,45,001,,77,29,326,,71,13,106,*60

Read 1:

Read 64:$GLGSV,2,2,08,85,12,042,,70,10,058,,75,00,000,45,87,00,000,25*65
Read 1:

Read 1:

Read 52:$GPGLL,2832.48525,N,07720.68458,E,074459.000,V,N*4A

Read 1:

Read 24:$PSTMCPU,36.35,-1,49*4C

Read 1:

Read 69:$GPRMC,074500.000,V,2832.48525,N,07720.68458,E,0.0,0.0,250521,,,N*71

Read 1:

Read 75:$GPGGA,074500.000,2832.48525,N,07720.68458,E,0,02,99.0,260.83,M,0.0,M,,*66

```

Figure 15 running gnss_uart

\$/ gnss_i2c

```

$GPRMC,074246.000,V,2832.48525,N,07720.68458,E,0.0,0.0,250521,,,N*74
$GPGGA,074246.000,2832.48525,N,07720.68458,E,0,01,99.0,260.83,M,0.0,M,,*60
$GPVTG,0.0,T,,M,0.0,N,0.0,K,N*02
$GNGSA,A,1,,,,,,,,,,,,,99.0,99.0,99.0*1E
$GNGSA,A,1,76,,,,,,,,,,,,,99.0,99.0,99.0*1F
$GPGSV,3,1,09,02,74,280,,06,60,026,,12,39,323,,19,36,051,*7F
$GPGSV,3,2,09,24,33,252,,28,32,127,,17,20,069,,14,13,133,*70
$GPGSV,3,3,09,05,10,187,,,,,,,,,,,,,*4A
$GLGSV,2,1,08,76,73,264,31,86,46,000,,77,28,327,,71,12,107,*6A
$GLGSV,2,2,08,85,12,041,,70,10,059,,75,00,000,44,87,00,000,33*61
$GPGLL,2832.48525,N,07720.68458,E,074246.000,V,N*42
$PSTMCPU,30.80,-1,49*44
$GPRMC,074247.000,V,2832.48525,N,07720.68458,E,0.0,0.0,250521,,,N*75
$GPGGA,074247.000,2832.48525,N,07720.68458,E,0,01,99.0,260.83,M,0.0,M,,*61
$GPVTG,0.0,T,,M,0.0,N,0.0,K,N*02
$GNGSA,A,1,,,,,,,,,,,,,99.0,99.0,99.0*1E
$GNGSA,A,1,76,,,,,,,,,,,,,99.0,99.0,99.0*1F
$GPGSV,3,1,09,02,74,280,,06,60,026,,12,39,323,,19,36,051,*7F
$GPGSV,3,2,09,24,33,252,,28,32,127,,17,20,069,,14,13,133,*70
$GPGSV,3,3,09,05,10,187,,,,,,,,,,,,,*4A
$GLGSV,2,1,08,76,73,264,31,86,46,000,,77,28,327,,71,12,107,*6A
$GLGSV,2,2,08,85,12,041,,70,10,059,,75,00,000,44,87,00,000,32*60
$GPGLL,2832.48525,N,07720.68458,E,074247.000,V,N*43
$PSTMCPU,31.81,-1,49*44
$GPRMC,074248.000,V,2832.48525,N,07720.68458,E,0.0,0.0,250521,,,N*7A
$GPGGA,074248.000,2832.48525,N,07720.68458,E,0,01,99.0,260.83,M,0.0,M,,*6E
$GPVTG,0.0,T,,M,0.0,N,0.0,K,N*02
$GNGSA,A,1,,,,,,,,,,,,,99.0,99.0,99.0*1E
$GNGSA,A,1,76,,,,,,,,,,,,,99.0,99.0,99.0*1F
$GPGSV,3,1,09,02,74,280,,06,60,026,,12,39,323,,19,36,051,*7F
$GPGSV,3,2,09,24,33,252,,28,32,127,,17,20,069,,14,13,133,*70
$GPGSV,3,3,09,05,10,187,,,,,,,,,,,,,*4A
$GLGSV,2,1,08,76,73,264,31,86,46,000,,77,28,327,,71,12,107,*6A

```

Figure 16 running gnss_i2c

4.1.3 Python Code

Installing the dependencies on STM32MPU

You need to install the [pyserial](#) and [pynmea2](#) for this.

#Installing pyserial

Python Serial Port Extension

Navigation

Project description

Release history

Download files

Project links

Homepage

Download files

Download the file for your platform. If you're not sure which to choose, learn more about [installing packages](#).

Filename, size	File type	Python version	Upload date	Hashes
pyserial-3.5-cp35-cp36-macosx10.7-tar.gz (90.6 kB)	Wheel	py2.py3	Nov 23, 2020	View
pyserial-3.5.tar.gz (159.1 kB)	Source	None	Nov 23, 2020	View

Figure 17 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
```

#installing pynmea2

4.1.4 Maps and Asset Tracking

X-LINUX-GNSS1 provides an example to send GNSS data to the cloud over http. [ST Asset Tracking Dashboard](#) is used to display the live GNSS data . Below are the steps to getting started with sending GNSS data to cloud

#Create a login at ST Asset Tracking Dashboard: <https://dsh-assettracking.st.com/> , its free

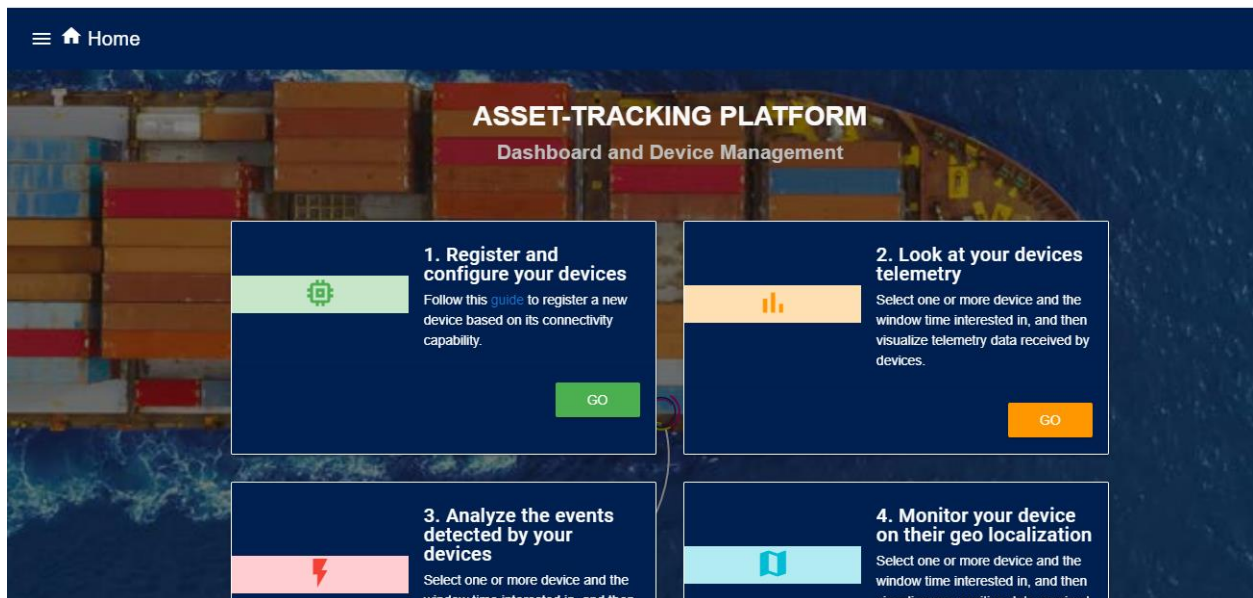


Figure 20 ST Asset Tracking Dashboard

Login or create an account at <https://dsh-assettracking.st.com/#/login>


Already registered?

Enter your e-mail address and password to login your myST user.

E-mail address

saurabh.rawat@st.com

Password

☒ Remember me on this computer. 

Login

[Forgot password?](#)

Figure 21 ST Asset Tracking Login

Once logged in , create a device (device name and device id) from the Devices Tab

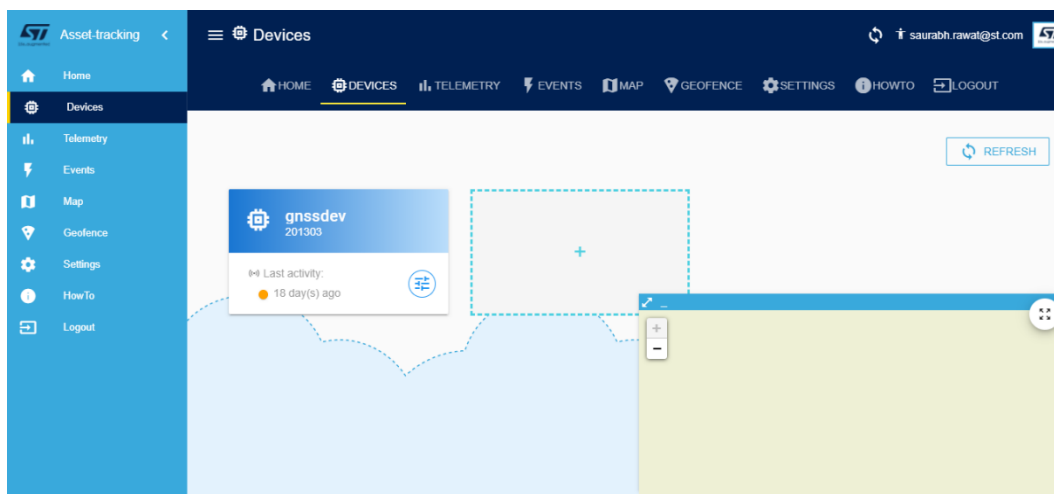


Figure 22 ST Asset Tracking Dashboard Home

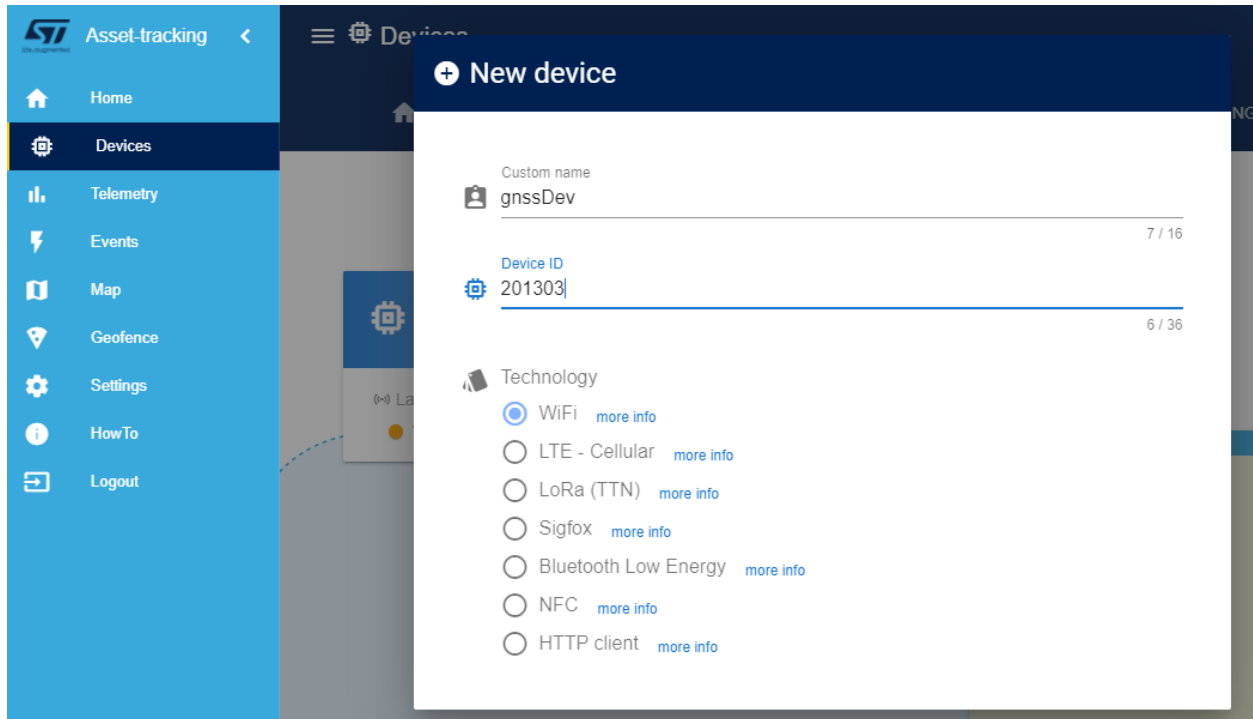
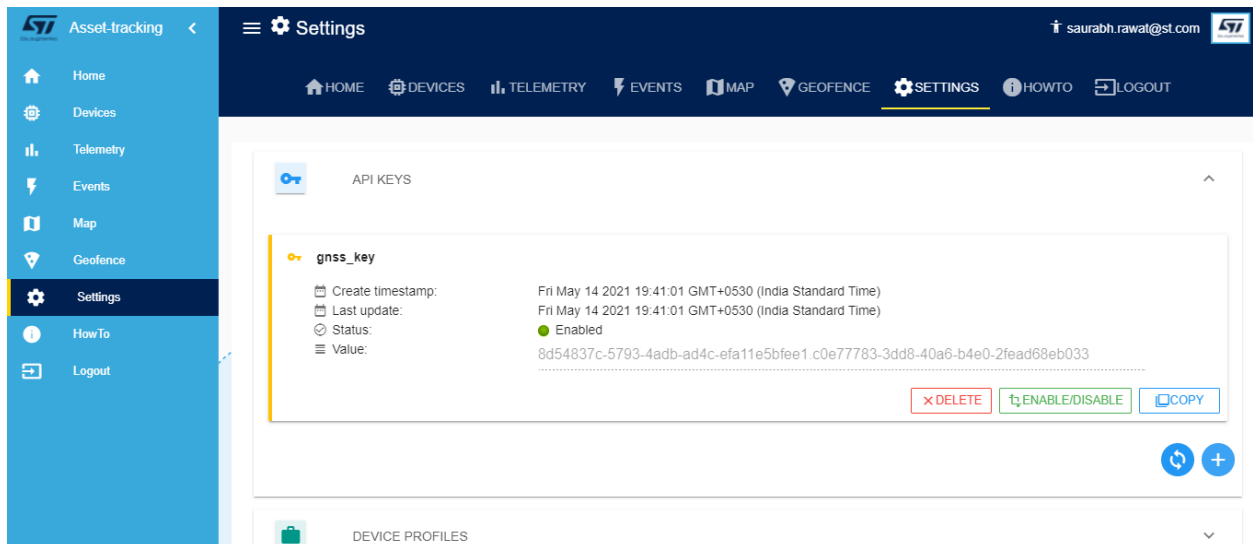


Figure 23 ST Asset Tracking Dashboard Creating a device ID and device Name

#Create API Key which will be used to send data to this Asse Tracking Dashboard



Location of endpoint:



```
$cd /path-to/X-LINUX-GNSS1_V1.0.0/Application/Source/gnss_x_linux/Sources/cloud
```

```
saurabh7@vmi613801:~/gnss/x-linux-gnssal/X-LINUX-GNSS1_V1.0.0/Application/Source/gnss_x_linux/Sources/cloud$ ls -l
total 44
-rw-rw-r-- 1 saurabh7 saurabh7 280 Jun 26 22:57 cloud_comm_common.h
-rw-rw-r-- 1 saurabh7 saurabh7 4222 Jun 26 22:57 cloud_comm_config.c
-rw-rw-r-- 1 saurabh7 saurabh7 625 Jun 26 22:57 cloud_comm_config.h
-rw-rw-r-- 1 saurabh7 saurabh7 4340 Jun 26 22:57 cloud_comm_https.c
-rw-rw-r-- 1 saurabh7 saurabh7 428 Jun 26 22:57 cloud_comm_https.h
-rw-rw-r-- 1 saurabh7 saurabh7 1168 Jun 26 22:57 cloud_helper.c
-rw-rw-r-- 1 saurabh7 saurabh7 192 Jun 26 22:57 creds.conf
-rw-rw-r-- 1 saurabh7 saurabh7 1336 Jun 26 22:57 gps_parser.h
-rw-rw-r-- 1 saurabh7 saurabh7 1292 Jun 26 22:57 README.md
saurabh7@vmi613801:~/gnss/x-linux-gnssal/X-LINUX-GNSS1_V1.0.0/Application/Source/gnss_x_linux/Sources/cloud$
```

#modify creds.conf file device_id,api_key and endpoint and Transfer it to STM32MP1-DK2 at the same place where gnss_app was transferred

```
http_endpoint = https://jim3rgi6d3.execute-api.eu-central-1.amazonaws.com/v1/telemetry
```

```
api_key = 8d54837c-5793-4adb-ad4c-efa11e5bfee1.c0e77783-3dd8-40a6-b4e0-2fead68eb033
```

```
device_id = 201303
```

```
http_endpoint = https://jim3rgi6d3.execute-api.eu-central-1.amazonaws.com/v1/telemetry
```

```
api_key = 8d54837c-5793-4adb-ad4c-efa11e5bfee1.c0e77783-3dd8-40a6-b4e0-2fead68eb033
```

```
device id = 201303
```

```
$scp /path-to/X-LINUX-GNSS1_V1.0.0/Application/Source/gnss_x_linux/Sources/cloud/creds.conf  
root@192.168.72.1:/
```

The steps are same as for building and deploying the gnss_app

Enter option “20” to upload the data to cloud. Make sure your STM32MP1-DK2 board is connected to internet

```
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 - getfiver  
9 - getgsmmsg  
10 - getpgst  
11 - getgprmc  
12 - getgsamsg  
13 - getgsvmmsg  
19 - ext-help  
20 - Upload to Cloud  
21 - Stop Upload to Cloud
```

Figure 26 Running gnss app and enabling cloud upload option (“20”)

And then you will get the below logs and the live tracking on the Asset Tracking Dashboard.

```
COM31 - Tera Term VT
File Edit Setup Control Window Help
* issuer: C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
* ALPN, server accepted to use http/1.1
> POST /v1/telemetry HTTP/1.1
Host: jin3rgi6d3.execute-api.eu-central-1.amazonaws.com
Accept: application/json
user-agent: X-LINUX-CMS101
Authorization: 8d54837c-5793-4adb-ad4c-efaf1e5hfeel.c0e77783-3dd8-48a6-b4e0-2fead68eb033
Content-Type: application/json
Content-Length: 121

* upload completely sent off: 121 out of 121 bytes
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Fri, 11 Jun 2021 08:15:08 GMT
< Content-Type: application/json
< Content-Length: 65
< Connection: keep-alive
< x-amzn-RequestId: 504b591b-0584-4aa8-9ecb-67a6c52d9de2
< x-amz-apigw-id: AwE35PFE2aPnA=
< X-Amzn-Trace-Id: Root=1-60c31b96-5e89732e06f02755097899ad;Sampled=0

* Connection #0 to host jin3rgi6d3.execute-api.eu-central-1.amazonaws.com left intact
{"message": "OK", "traceId": "8178c1a2-9bb2-044f-bfdb-b654c119efd9"}Uploading to Cloud Every 9000 milliseconds...
Latitude -> 28.227554
Longitude -> 77.213088
Altitude -> 2.662200
* Trying 52.28.83.33:443...
* Connected to jin3rgi6d3.execute-api.eu-central-1.amazonaws.com (52.28.83.33) port 443 (#0)
* found 128 certificates in /etc/ssl/certs/ca-certificates.crt
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
* server certificate verification OK
* server certificate status verification SKIPPED
* common name: *.execute-api.eu-central-1.amazonaws.com (matched)
* server certificate expiration date OK
* server certificate activation date OK
* certificate public key: RSA
* certificate version: #3
* subject: CN=*.execute-api.eu-central-1.amazonaws.com
* start date: Sat, 29 Aug 2020 00:00:00 GMT
* expire date: Wed, 29 Sep 2021 12:00:00 GMT
* issuer: C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
* ALPN, server accepted to use http/1.1
> POST /v1/telemetry HTTP/1.1
Host: jin3rgi6d3.execute-api.eu-central-1.amazonaws.com
Accept: application/json
user-agent: X-LINUX-CMS101
Authorization: 8d54837c-5793-4adb-ad4c-efaf1e5hfeel.c0e77783-3dd8-48a6-b4e0-2fead68eb033
Content-Type: application/json
Content-Length: 121

* upload completely sent off: 121 out of 121 bytes
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Fri, 11 Jun 2021 08:15:18 GMT
< Content-Type: application/json
< Content-Length: 65
< Connection: keep-alive
< x-amzn-RequestId: 3d48af6e-1cf5-4f37-af7b-94e5351d472e
< x-amz-apigw-id: AwE_ehloq1l8PnA=
< X-Amzn-Trace-Id: Root=1-60c31b96-399ed67f05c71f3c5dc27896;Sampled=0

* Connection #0 to host jin3rgi6d3.execute-api.eu-central-1.amazonaws.com left intact
```

Figure 27 Gnss app sending data to ST Asset Tracking Dashboard over HTTP

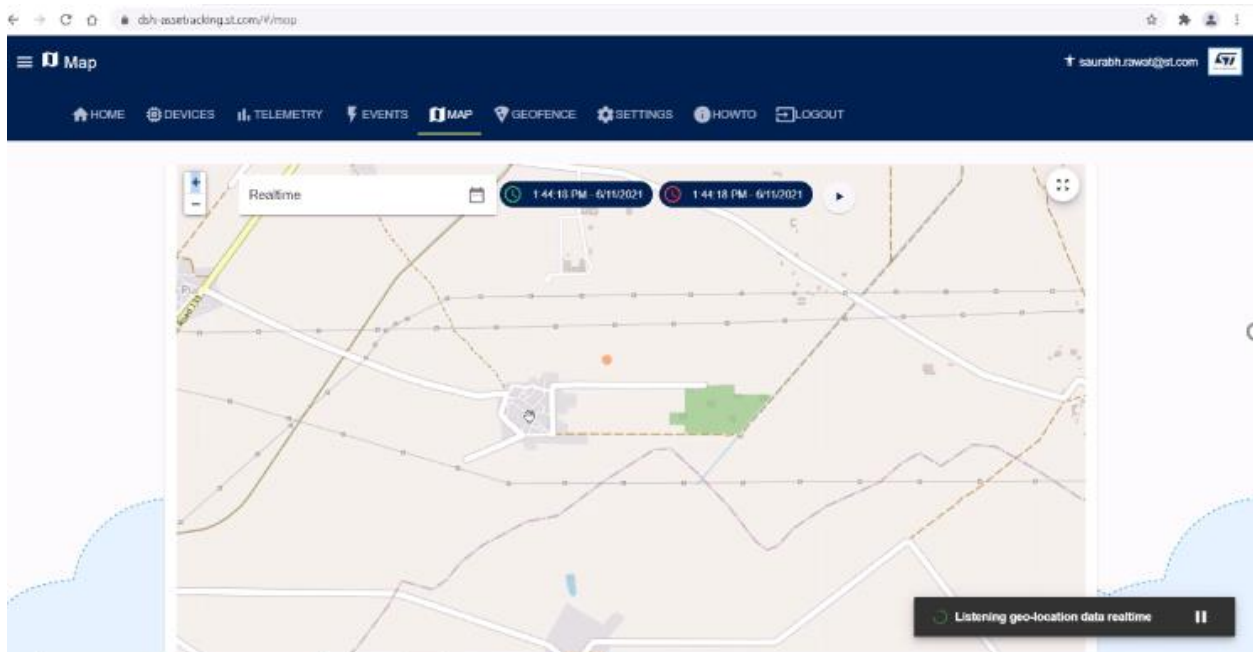
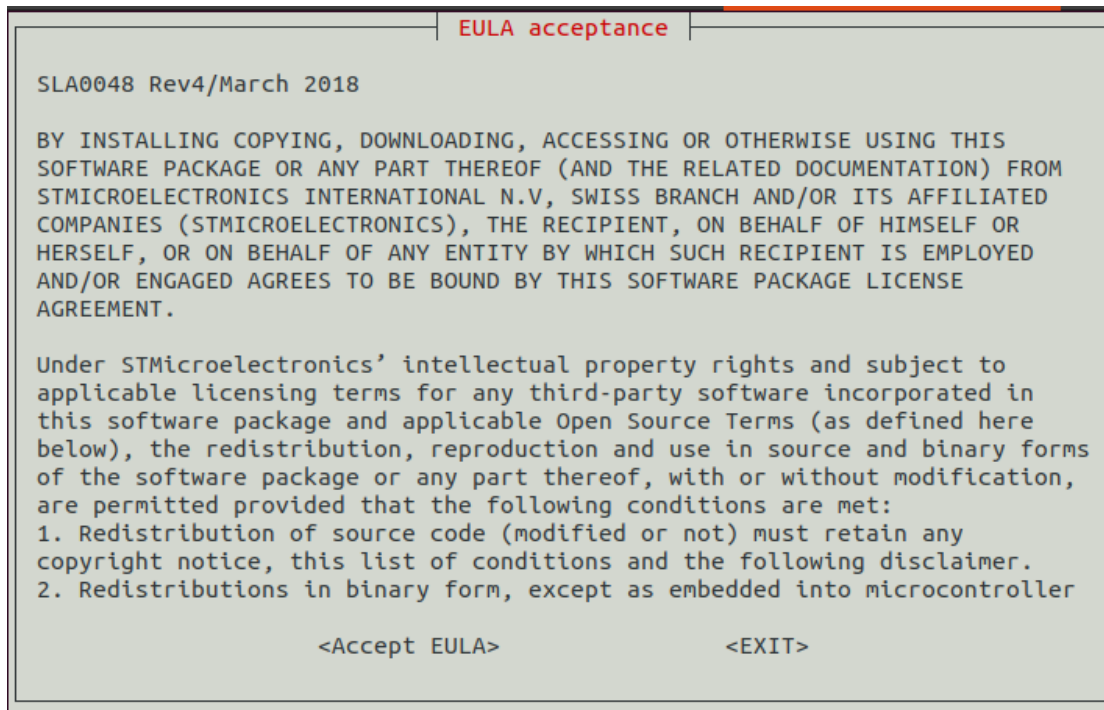


Figure 28 Gnss app live data

4.2 Using BitBake (For Distribution Package)

Below is an example to add the gnss_app as meta layer. Same procedure can be used to add C Utility as meta layer.

#Download the Distribution Package and do bitbake. Doing bitbake for the best time will show the EULA accept prompt, Accept it (type “y” and press enter)



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

```
$bitbake st_image_weston
```



```
saaurabh7@vmi613801: ~/dp/openstlinux-5.10-dunfell-mp1-21-03-31

#####
BROADCOM BCM43XX

SOFTWARE LICENSE AGREEMENT

The accompanying software in binary code form ("Software"), is licensed to you, or, if you are accepting on behalf of an entity, the entity and its affiliates exercising rights hereunder ("Licensees"), subject to the terms of this software license agreement ("Agreement"), unless Licensee and Broadcom Corporation ("Broadcom") execute a separate written software license agreement governing use of the Software. ANY USE, REPRODUCTION, OR DISTRIBUTION OF THE SOFTWARE CONSTITUTES LICENSEE'S ACCEPTANCE OF THIS AGREEMENT.

1. License. Subject to the terms and conditions of this Agreement, Broadcom hereby grants to Licensee a limited, non-exclusive, non-transferable, royalty-free license: (i) to use and integrate the Software with any other software; and (ii) to reproduce and distribute the Software complete, unmodified, and as provided by Broadcom, solely for use with Broadcom proprietary integrated circuit product(s) sold by Broadcom with which the Software was designed to be used, or their successors.

2. Restrictions. Licensee shall distribute Software with a copy of this Agreement. Licensee shall not remove, efface or obscure any copyright or trademark notices from the Software. Reproductions of the Broadcom copyright notice shall be included with each copy of the Software, except where such Software is embedded in a manner not readily accessible to the end user. Licensee shall not: (i) use, license, sell or otherwise distribute the Software except as provided in this Agreement; (ii) attempt to modify in any way, reverse engineer, decompile or disassemble any portion of the Software; or (iii) use the Software or other material in violation of any applicable law or regulation, including but not limited to any regulatory agency. This Agreement shall automatically terminate upon Licensee's failure to comply with any of the terms of this Agreement. In such event, Licensee will destroy all copies of the Software and its component parts.

3. Ownership. The Software is licensed and not sold. Title to and ownership of the Software, including all intellectual property rights thereto, and any portion thereof remain with Broadcom or its licensors. Licensee hereby covenants that it will not assert any claim that the Software created by or for Broadcom infringe any intellectual property right owned or controlled by Licensee.

4. Disclaimer. THE SOFTWARE IS OFFERED "AS IS," AND BROADCOM PROVIDES AND GRANTS AND LICENSEE RECEIVES NO SUPPORT AND NO WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, BY STATUTE, COMMUNICATION OR CONDUCT WITH LICENSEE, OR OTHERWISE. BROADCOM SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A SPECIFIC PURPOSE, OR NONINFRINGEMENT CONCERNING THE SOFTWARE OR ANY UPGRADES TO OR DOCUMENTATION FOR THE SOFTWARE. WITHOUT LIMITATION OF THE ABOVE, BROADCOM GRANTS NO WARRANTY THAT THE SOFTWARE IS ERROR-FREE OR WILL OPERATE WITHOUT INTERRUPTION, AND NO GRANTS NO WARRANTY REGARDING ITS USE OR THE RESULTS THEREFROM INCLUDING, WITHOUT LIMITATION, ITS CORRECTNESS, ACCURACY, OR RELIABILITY. TO THE MAXIMUM EXTENT PERMITTED BY LAW, IN NO EVENT SHALL BROADCOM OR ANY OF ITS LICENSORS HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER FOR BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE) OR OTHERWISE, ARISING OUT OF THIS AGREEMENT OR USE, REPRODUCTION, OR DISTRIBUTION OF THE SOFTWARE, INCLUDING BUT NOT LIMITED TO LOSS OF DATA AND LOSS OF PROFITS, EVEN IF SUCH PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY.

5. Export Laws. LICENSEE UNDERSTANDS AND AGREES THAT THE SOFTWARE IS SUBJECT TO UNITED STATES AND OTHER APPLICABLE EXPORT-RELATED LAWS AND REGULATIONS AND THAT LICENSEE MAY NOT EXPORT, RE-EXPORT OR TRANSFER THE SOFTWARE OR ANY DIRECT PRODUCT OF THE SOFTWARE EXCEPT AS PERMITTED UNDER THOSE LAWS. WITHOUT LIMITING THE FOREGOING, EXPORT, RE-EXPORT, OR TRANSFER OF THE SOFTWARE TO CUBA, IRAN, NORTH KOREA, SUDAN, AND SYRIA IS PROHIBITED.

Do you accept the EULA you just read? (y/n) y
```

#You will be inside the build directory. Once bitbake is done, download the gnss package from www.st.com inside the build directory as shown in figure below.

```
saaurabh@saaurabh:~/gnss/build-openstlinuxweston-stm32mp1$ ls
conf  X-LINUX-GNSS1 V1.0.0  X-LINUX-GNSS1 V1.0.0.tar.xz
saaurabh@saaurabh:~/gnss/build-openstlinuxweston-stm32mp1$
```

#Create a layer : meta-gnss1

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

```
saaurabh@saaurabh:~/gnss/build-openstlinuxweston-stm32mp1$ bitbake-layers create-layer --priority 7 ../layers/meta-st/meta-gnss1
NOTE: Starting bitbake server...
Add your new layer with 'bitbake-layers add-layer ../layers/meta-st/meta-gnss1'
saaurabh@saaurabh:~/gnss/build-openstlinuxweston-stm32mp1$
```

#Add a layer a layer : recipes-gnss1

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

```
saaurabh@saaurabh:~/gnss/build-openstlinuweston-stm32mp1$ bitbake-layers add-layer ../layers/meta-st/meta-gnss1
NOTE: Starting bitbake server...
saaurabh@saaurabh:~/gnss/build-openstlinuweston-stm32mp1$
```

#See the added layer layer : meta-gnss1

\$bitbake-layers show-layers

```
saaurabh@saaurabh:~/gnss/build-openstlinuweston-stm32mp1$ bitbake-layers show-layers
NOTE: Starting bitbake server...
layer      path      priority
-----
meta-python /home/saurabh/gnss/layers/meta-openembedded/meta-python 7
meta-oe     /home/saurabh/gnss/layers/meta-openembedded/meta-oe 6
meta-gnome  /home/saurabh/gnss/layers/meta-openembedded/meta-gnome 7
meta-initramfs /home/saurabh/gnss/layers/meta-openembedded/meta-initramfs 8
meta-multimedia /home/saurabh/gnss/layers/meta-openembedded/meta-multimedia 6
meta-networking /home/saurabh/gnss/layers/meta-openembedded/meta-networking 5
meta-webserver /home/saurabh/gnss/layers/meta-openembedded/meta-webserver 6
meta-filesystems /home/saurabh/gnss/layers/meta-openembedded/meta-filesystems 6
meta-perl    /home/saurabh/gnss/layers/meta-openembedded/meta-perl 6
meta-st-stm32mp /home/saurabh/gnss/layers/meta-st/meta-st-stm32mp 6
meta-qt5     /home/saurabh/gnss/layers/meta-qt5 7
meta-st-openstlinu /home/saurabh/gnss/layers/meta-st/meta-st-openstlinu 5
meta        /home/saurabh/gnss/layers/openembedded-core/meta 5
meta-gnss1   /home/saurabh/gnss/layers/meta-st/meta-gnss1 7
saaurabh@saaurabh:~/gnss/build-openstlinuweston-stm32mp1$
```

#Add the IMAGE_INSTALL_append line as shown below in the layer.conf at the end

\$vi ../layers/meta-st/meta-st-openstlinu/conf/layer.conf

IMAGE_INSTALL_append += "gnss1"

```
# We have a conf and classes directory, add to BBPATH
BBPATH += "${LAYERDIR}"

# We have a recipes-* directories, add to BBFILES
BBFILES += "${LAYERDIR}/recipes-*//*/*.bb \
           ${LAYERDIR}/recipes-*//*/*.bbappend \
           "

# This folder should only contains specific patches to fix issue on oe recipes
# Note that these patches may be pushed on Community
BBFILES += "${LAYERDIR}/oe-core/recipes-*//*/*.bbappend"

# This folder should only contains direct backport from oe recipes
# These recipes may be suppress at next update on oe version
BBFILES += "${LAYERDIR}/oe-backport/recipes-*//*/*.bb \
           ${LAYERDIR}/oe-backport/recipes-*//*/*.bbappend \
           "

BBFILE_COLLECTIONS += "st-openstlinu"
BBFILE_PATTERN_st-openstlinu := "${LAYERDIR}/"
BBFILE_PRIORITY_st-openstlinu = "5"

LAYERDEPENDS_st-openstlinu = "qt5-layer"

# Set a variable to get the openstlinu location
OPENSTLINU_BASE = "${LAYERDIR}"

# This should only be incremented on significant changes that will
# cause compatibility issues with other layers
LAYERVERSION_st-openstlinu = "1"
LAYERSERIES_COMPAT_st-openstlinu = "dunfell"

INHERIT += "check-st-openstlinu-compatibility"
# Openstlinu compatibility version
ST_OSTL_COMPATIBLTY_VERSION_st-openstlinu = "3.0"
IMAGE_INSTALL_append += "gnss1"
~
~
```

#Delete Completely meta-gnss1 that is created by the tool and we copy the meta-gnss1 downloaded from github / www.st.com

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

```
saurabh@saurabh:~/gnss/build-openstlinuxweston-stm32mp1$ rm -rf ../layers/meta-st/meta-gnss1/
```

#And copy the layer provided from the X-Linux package

```
$ cp -rf X-LINUX-GNSS1_V1.0.0/Recipe/meta-gnss1/ ../layers/meta-st/.
```

```
saurabh@saurabh:~/gnss/build-openstlinuxweston-stm32mp1$ cp -rf X-LINUX-GNSS1_V1.0.0/Recipe/meta-gnss1/ ../layers/meta-st/
```

#Add the Sources path(Location where CMakeLists.txt is present) inside gnss1_0.1.bbappend as shown below. (by default in the package what path will be?) If no path is present the user need

```
# "/path-to/openstlinux-5.10-dunfell-mp1-21-03-31/build-openstlinuxweston-stm32mp1/x-linux-gnss/X-LINUX-GNSS1_V1.0.0/Application/Source/gnss_x_linux/Sources"
```

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
```

```
saurabh@saurabh:~/gnss/build-openstlinuxweston-stm32mp1$ vi ../layers/meta-st/meta-gnss1/recipes-gnss1/gnss1/gnss1_0.1.bbappend
```

```
inherit externalsrc
EXTERNALSRC="/home/saurabh/gnss/build-openstlinuxweston-stm32mp1/X-LINUX-GNSS1_V1.0.0/Application/Source/gnss_x_linux/Sources"
```

#Build the ST image

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

```
saurabh@saurabh:~/gnss/build-openstlinuxweston-stm32mp1$ bitbake st-image-weston
NOTE: Started PRServer with DBfile: /home/saurabh/gnss/build-openstlinuxweston-stm32mp1/cache/prserv.sqlite3, IP: 127.0.0.1, PORT: 46665, PID: 1698786
Parsing recipes: 4% |#####| ETA: 0:04:44
```

#New Images will be formed in the tmp-glibc/ deploy/images/stm32mp1/ directory

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

#FlashLayout_sdcard_stm32mp157f-dk2-trusted.tsv and FlashLayout_sdcard_stm32mp157f-dk2-trusted will be created besides other images

Follow below link (https://wiki.st.com/stm32mpu/wiki/STM32MP15_Discovery_kits_-_Starter_Package#Image_flashing) to flash the binary.

#check if below file is present on discovery kit

\$ ls -l /dev/ttySTM2

#Run the application

\$ /usr/bin/gnss_app or directly gnss_app

5.0 Revision History

Table 2 Revision History

Date	Version	Changes
6 th June 2021	1.0	Initial Release

Contents

1. X-LINUX-GNSS1 overview.....	1
1.1 Features	2
1.2 Architecture	3
1.3 Software Package Structure.....	3
1.3.1 GNSS_APP	5
1.3.2 C Utility.....	6
1.3.3 Python Code.....	6
2. Hardware Setup	7
3. Software Setup.....	9
3.1 Recommended PC prerequisites.....	9
3.2 Installing the SDK	9
3.3 Downloading the kernel Sources (Developer Package)	9
3.4 Downloading the Distribution Package.....	9

3.5 Connecting to the Board	10
4.Building and Running the example	10
4.1 Using Makefile (For Starter Package).....	10
4.1.1 gnss_app	11
4.1.2 C Utility.....	14
4.1.3 Python Code.....	15
4.1.4 Maps and Asset Tracking	18
4.2 Using BitBake (For Distribution Package)	24
5.0 Revision History	28

List of Figures

Figure 1 X-LINUX Software Architecture Diagram	1
Figure 2 Hardware Connections	2
Figure 3 Application Architecture	3
Figure 4 Release Package Structure Top Level	4
Figure 5 Application Folder Overview	4
Figure 6 GNSS Other Folder Structures	5
Figure 7 gnss_app	5
Figure 8 C utility – gnss-uart and gnss-i2c	6
Figure 9 Python Code	6
Figure 10 Hardware Setup – Jumper Settings.....	7
Figure 11 Cloning the Package	11
Figure 12 STM32MPU - Build Application	12
Figure 13 STM32MPU - Run Application	13
Figure 14 gnss_app Selecting the options - Select 11 to get the GPS Co-ordinates	14
Figure 15 running gnss_uart	15
Figure 16 running gnss_i2c.....	15
Figure 17 Installing Python Dependencies - Pyserial	16
Figure 18 Installing Python Dependencies – Pynmea2.....	17
Figure 19 Python example running	17
Figure 20 ST Asset Tracking Dashboard.....	18
Figure 21 ST Asset Tracking Login.....	19

Figure 22 ST Asset Tracking Dashboard Home	20
Figure 23 ST Asset Tracking Dashobard Creating a device ID and device Name	20
Figure 24 ST Asset Tracking Generating new API Key	21
Figure 25 ST Asset Tracking - Note down the endpoint	21
Figure 26 Running gnss app and enabling cloud upload option (“20”)	22
Figure 27 Gnss app sending data to ST Asset Tracking Dashboard over HTTP	23
Figure 28 Gnss app live data.....	23

List of Tables

Table 1 Jumper Settings.....	8
Table 2 Revision History	28