Getting started with X-LINUX-GNSS1A1 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) devfice, 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](https://www.st.com/en/ecosystems/x-nucleo-gnss1a1.html) 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 .

Graphical user interface, diagram

Description automatically generated

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](https://www.st.com/en/positioning/teseo-liv3f.html) 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 layers peripheral drivers ( I2C and UART) using user space application. It uses [termios](https://wiki.st.com/stm32mpu/wiki/TTY_tools) 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.

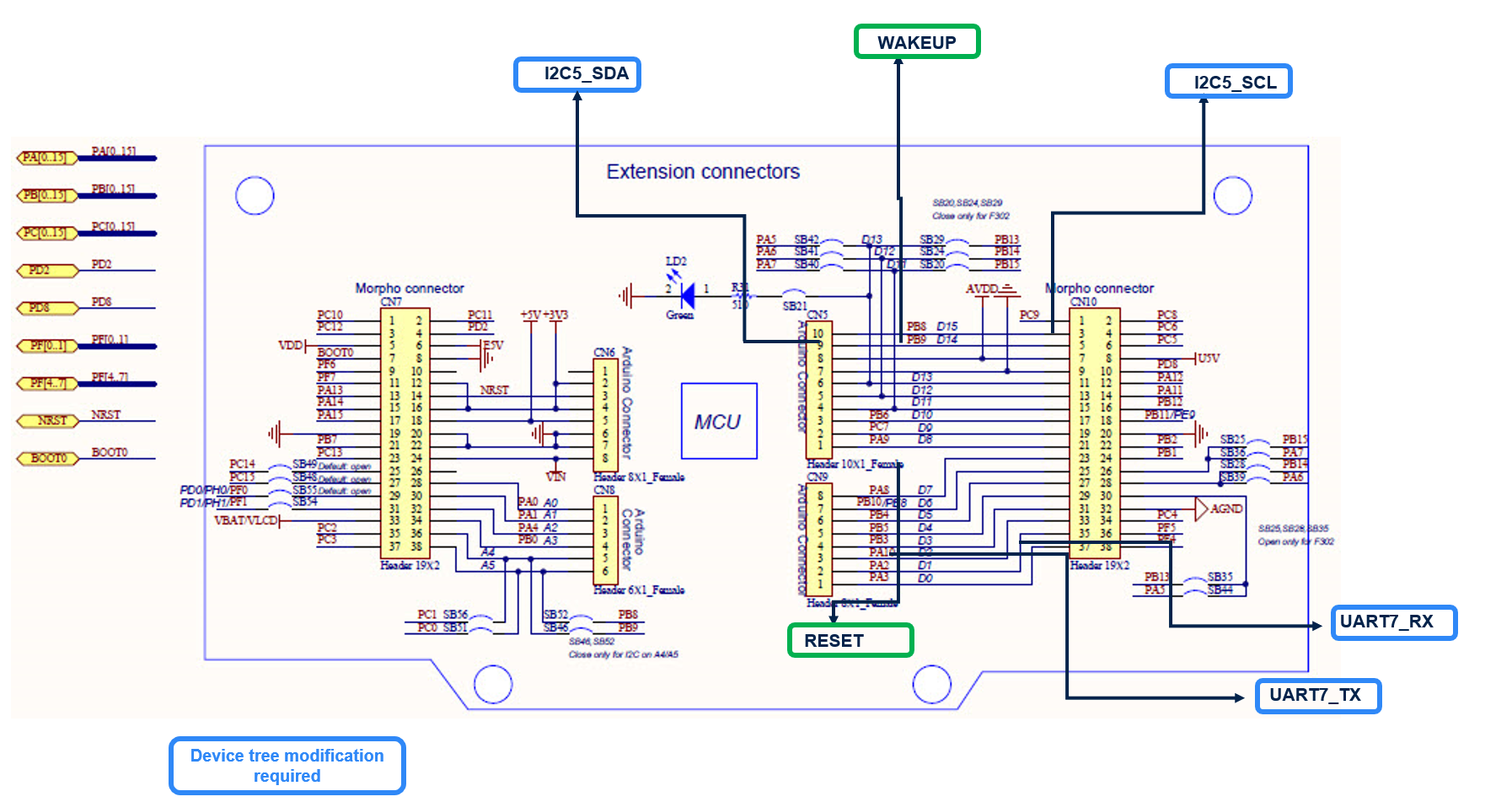


Figure Hardware Connections

## 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](https://www.st.com/en/embedded-software/dsh-assetracking.html) for live tracking.
* Python Example(gnss\_pynmea2.py)
* Free, user-friendly license terms

## 1.2 Architecture

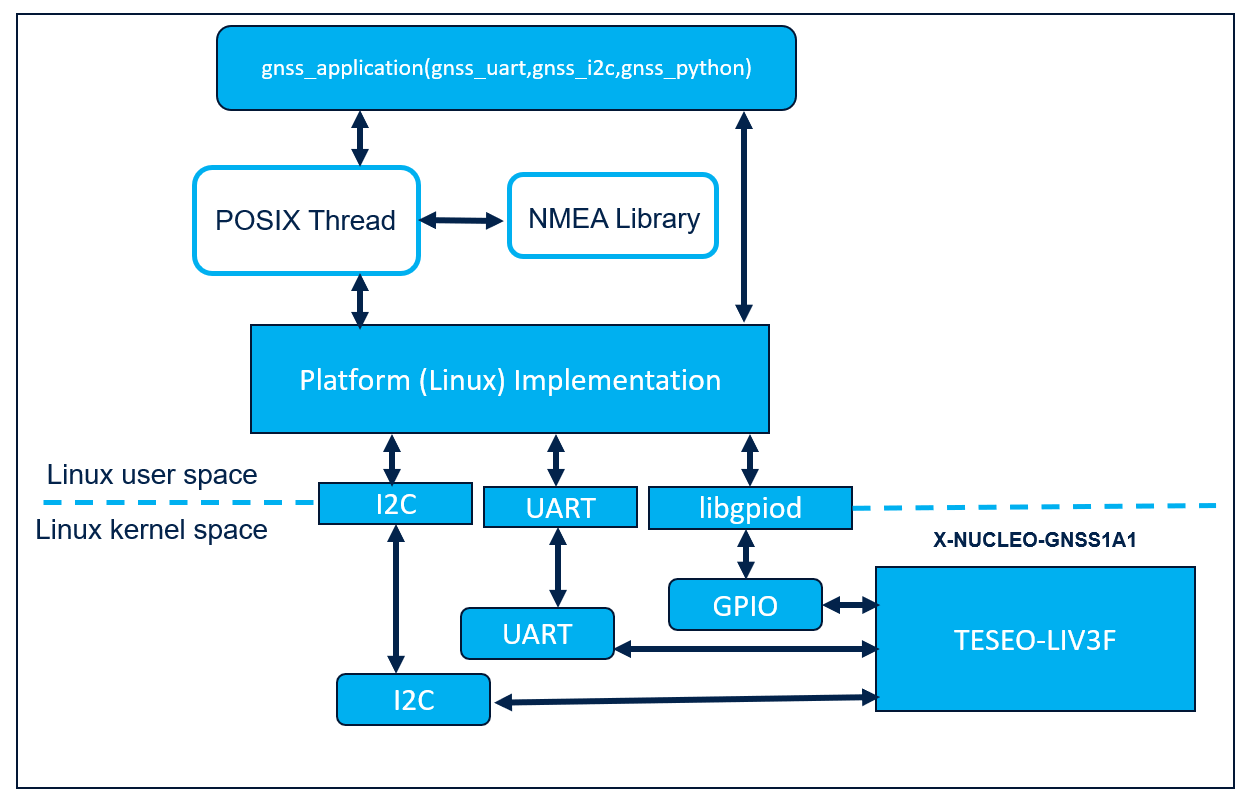


Figure Application Architecture

## 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.

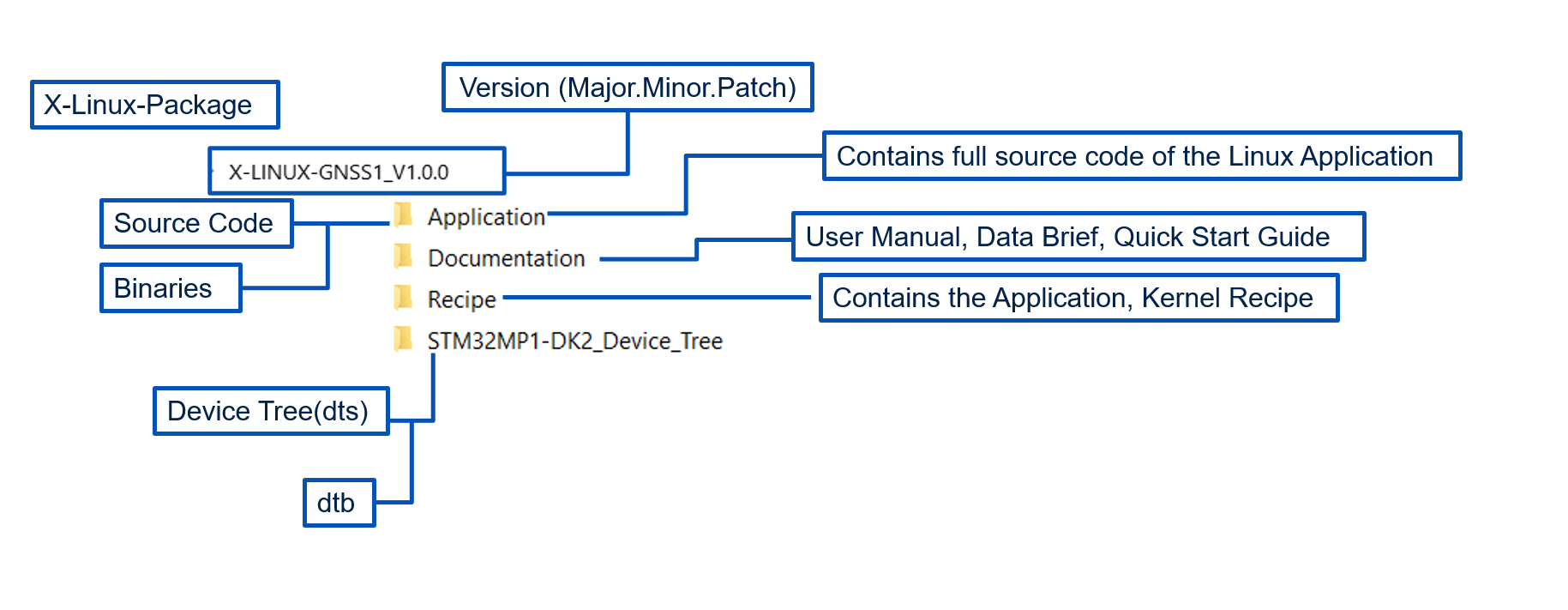


Figure Release Package Structure Top Level

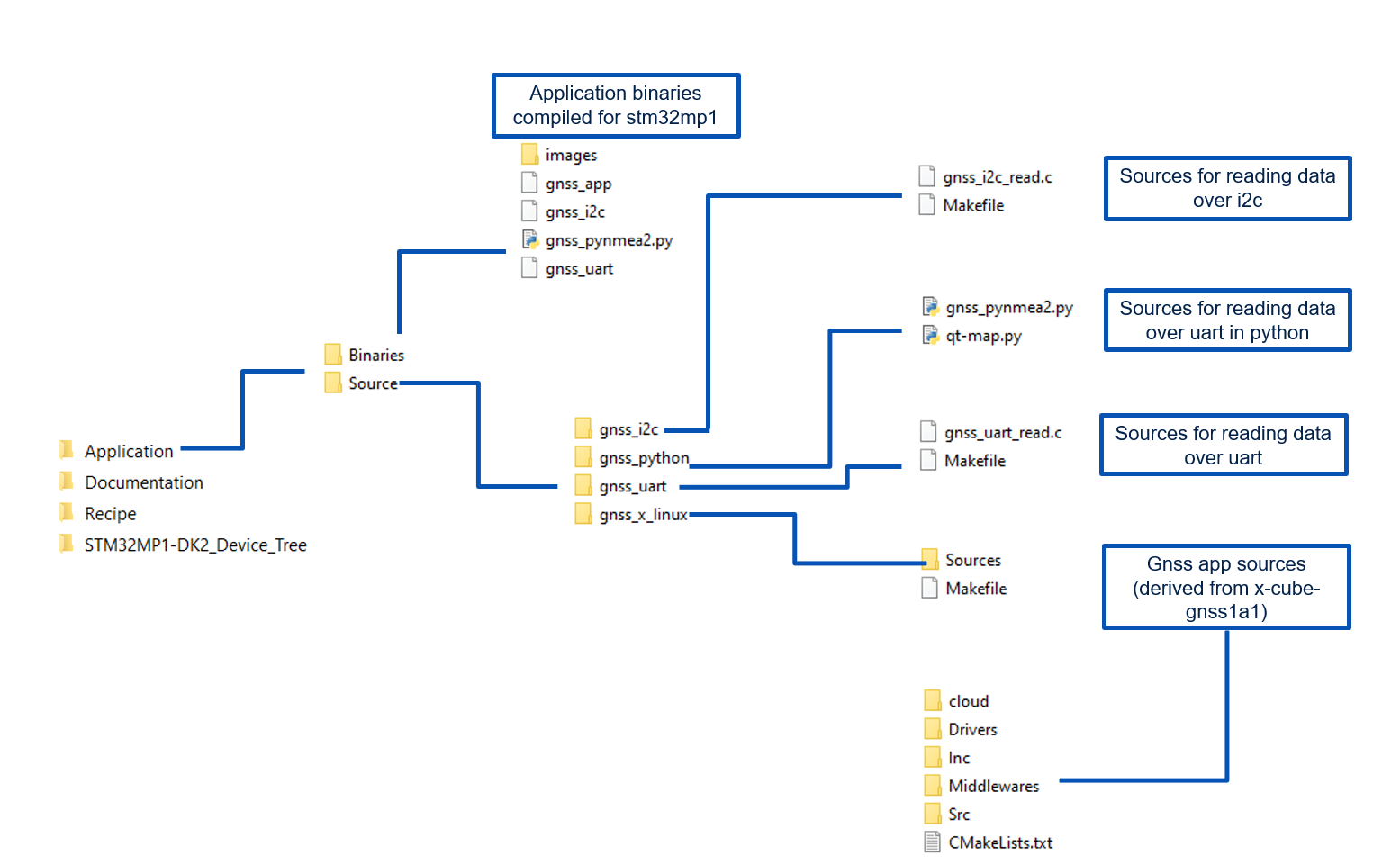


Figure Application Folder Overview

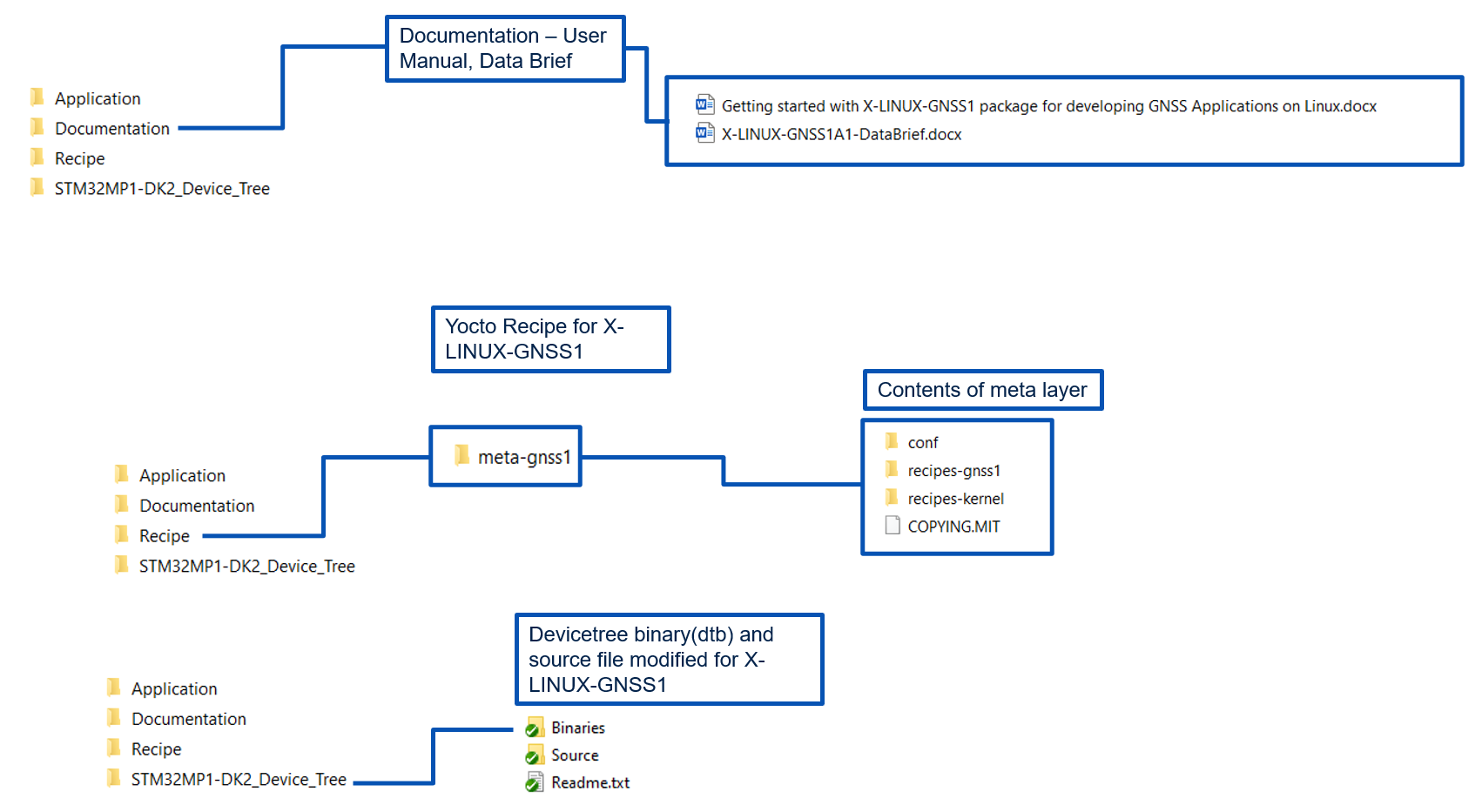


Figure GNSS Other Folder Structures

### 1.3.1 GNSS\_APP

This Application accesses the GPS data over UART (/dev/ttySTM32) 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)

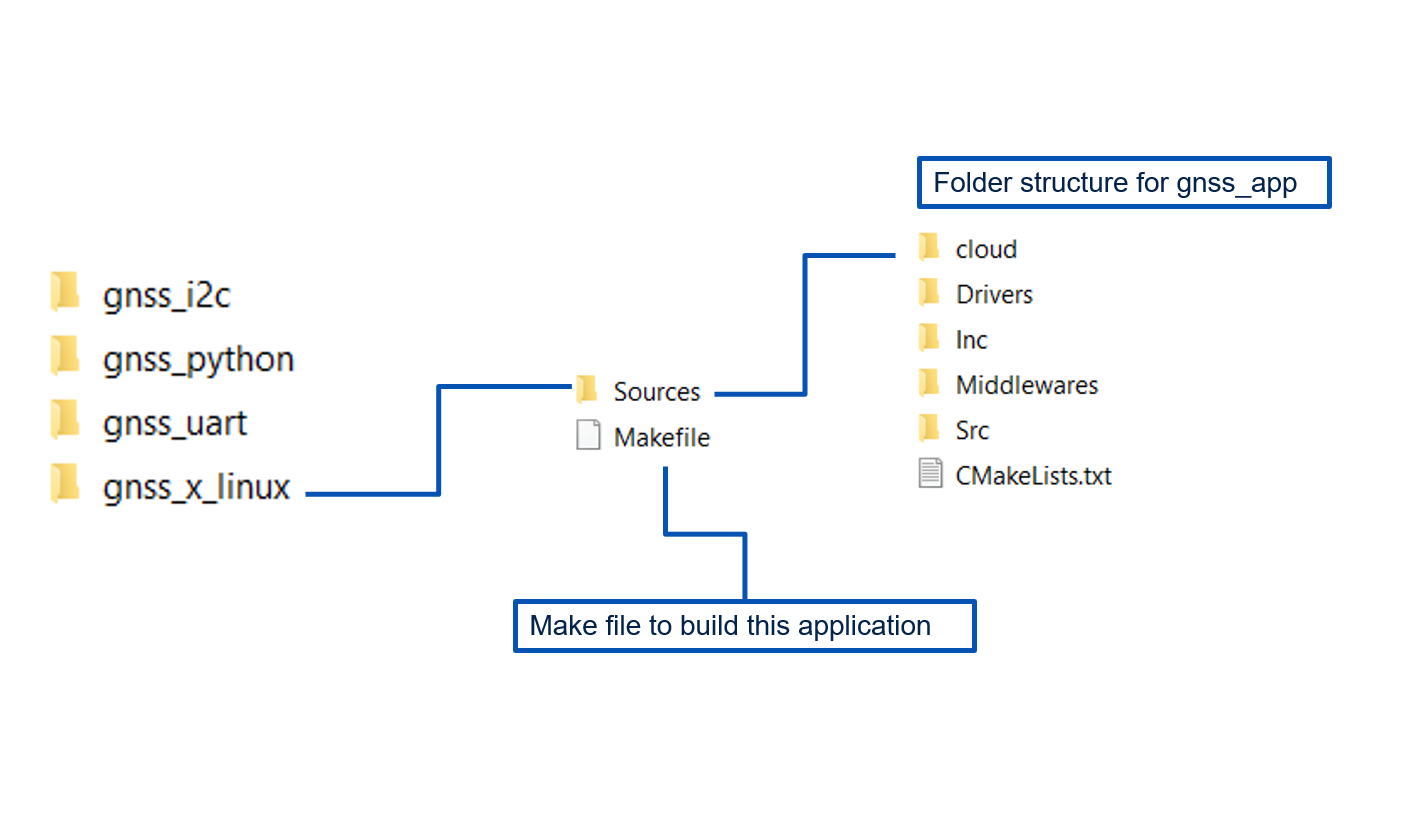


Figure gnss\_app

### 1.3.2 C Utility

This Application is Linux user space C application to read the data from the UART (/dev/ttySTM32) and I2C (/dev/i2c-1) interface.

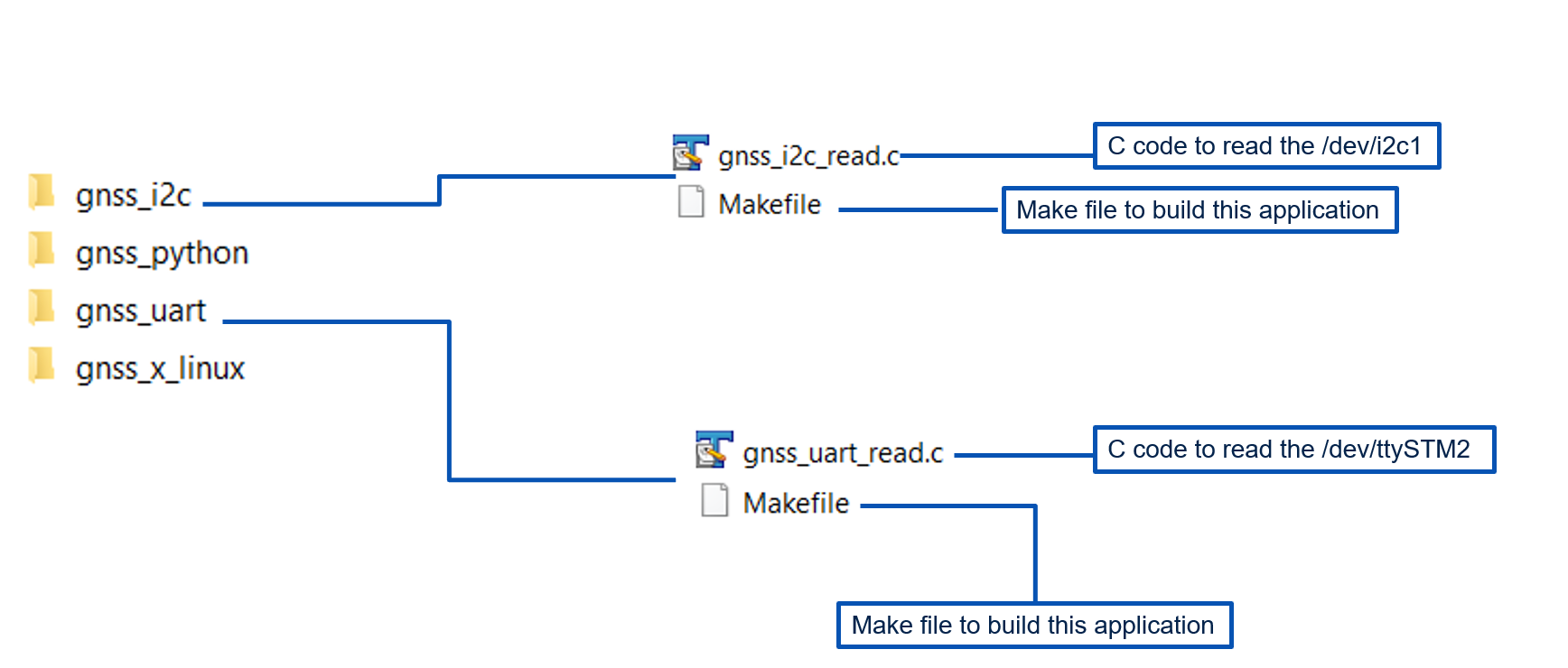


Figure 8 C utility – gnss-uart and gnss-i2c

### 1.3.3 Python Code

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

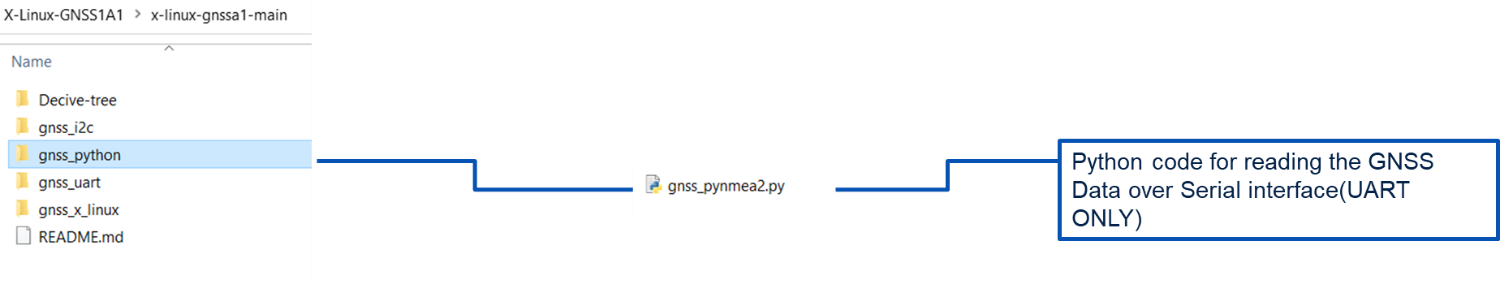


Figure 9 Python Code

# 2.Hardware Setup

The Software is compatible with [X-NUCLEO-GNSS1A1](https://www.st.com/en/ecosystems/x-nucleo-gnss1a1.html) 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.

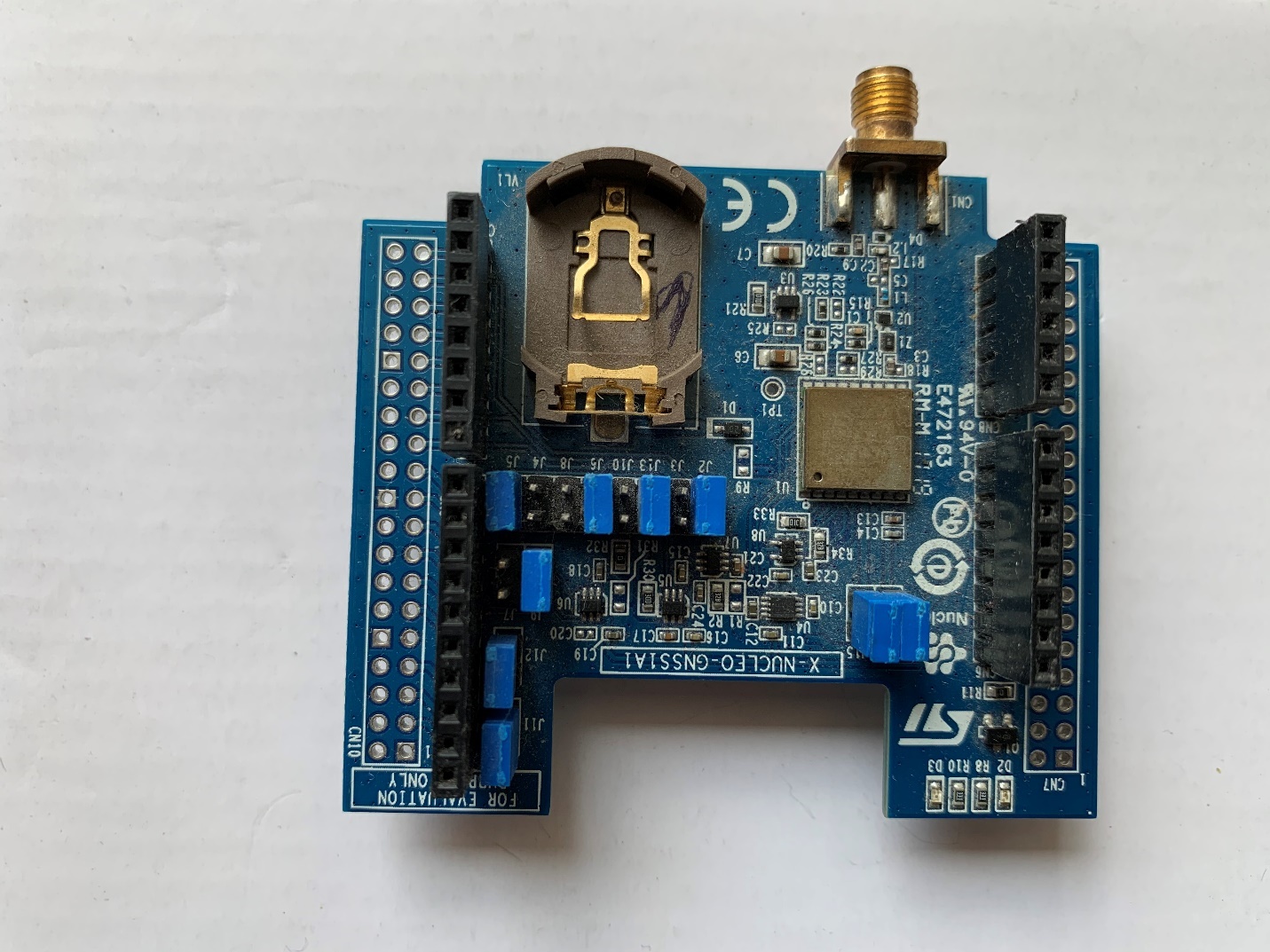


Figure Hardware Setup – Jumper Settings

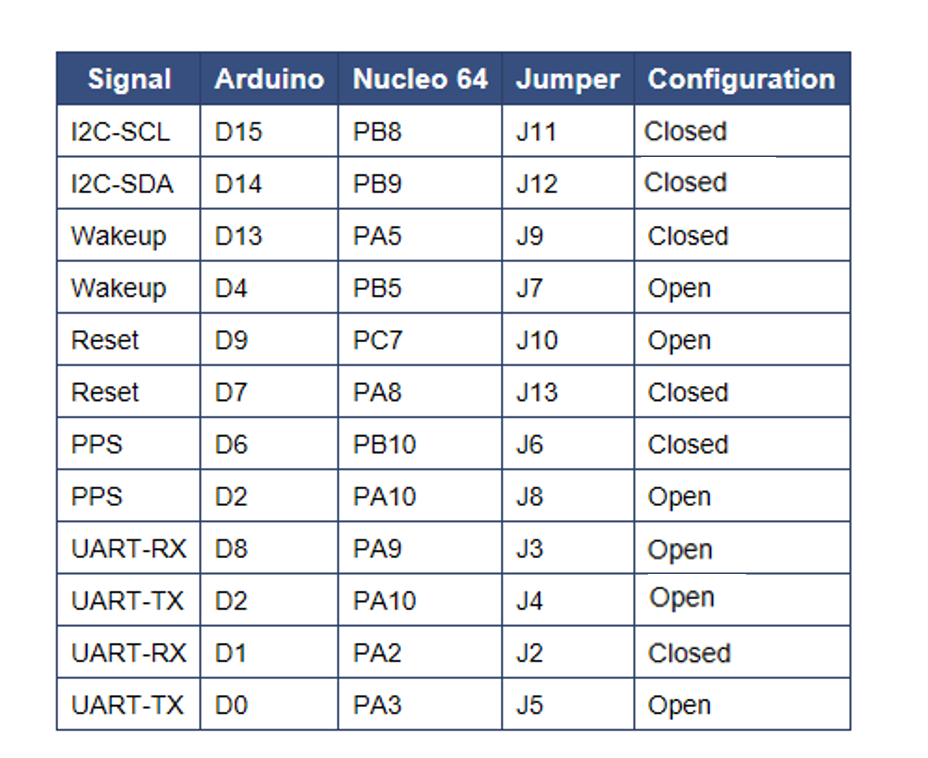


Table 1 Jumper Settings

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%C2%AE_Cortex%C2%AE-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%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 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](http://www.st.com) and untar it

$tar xvf tar xvf X-LINUX-GNSS\_V1.0.0.tar.xz

You will get X-LINUX-GNSS1\_V1.0.0 folder.

Text

Description automatically generated

Figure 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

A picture containing text, screenshot, electronics

Description automatically generated

#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)

#Once the dtbs are built, copy them to the STM32MP1-DK2 board. Make sure to connect to it using hotspot mode or over WIFI.

$scp <KERNEL SOURCE PATH>/arch/arm/boot/dts/stm32mp157f\_dk2.dtb [root@192.168.72.1:/boot](mailto:root@192.168.72.1:/boot)

#Build the gnss\_app

$cd gnss

#build the gnss\_app. cd to the gnss\_x\_linux location and do make

$cd path to/X-LINUX-GNSS1\_V1.0.0/Application/Source/gnss\_x\_linux

A picture containing text, electronics, screenshot, computer

Description automatically generated

Figure 12 STM32MPU - Build Application

The executable (gnss\_app) will be formed in the same location where Makefile is present.

A picture containing text, electronics, computer, screenshot

Description automatically generated

#Transfer the application to STM32MP1

scp gnss\_app [root@192.168.72.1:/](mailto:root@192.168.72.1:/)

#On STM32MP1 board sync and reboot

$sync

$reboot

#Running the Application

$cd /

$./gnss\_app

Text

Description automatically generated

Figure 13 STM32MPU - Run Application

Text

Description automatically generated

Figure 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

Text

Description automatically generated

Figure running gnss\_uart

$./ gnss\_i2c

Text

Description automatically generated

Figure running gnss\_i2c

### 4.1.3 Python Code

Installing the dependencies on STM32MPU

You need to install the [pyserial](https://pypi.org/project/pyserial/#files) and [pynmea2](https://pypi.org/project/pynmea2/#files) for this.

#Installing pyserial

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

Figure 18 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

#Enter the python folder, copy the readmap.py file to stm32mp1

$Python3 readmap.py

#Running the python code

Text

Description automatically generated

Figure 19 Python example running

### 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](https://dsh-assetracking.st.com/) 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-assetracking.st.com/> , its free

Graphical user interface, website

Description automatically generated

Figure ST Asset Tracking Dashboard

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

Graphical user interface, application

Description automatically generated

Figure ST Asset Tracking Login

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

Graphical user interface

Description automatically generated

Figure ST Asset Tracking Dashboard Home

Graphical user interface, application, Teams

Description automatically generated

Figure ST Asset Tracking Dashobard Creating a device ID and device Name

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

Graphical user interface, text, application, email

Description automatically generated

Figure ST Asset Tracking Generating new API Key

Location of endpoint:

Graphical user interface, text, application, website

Description automatically generated

Figure ST Asset Tracking - Note down the endpoint

#Modify the data in the creds.conf file as per the endpoint, device Id and the API Key

$cd /path-to/X-LINUX-GNSS1\_V1.0.0/Application/Source/gnss\_x\_linux/Sources/cloud

$vi creds.conf

Graphical user interface, text

Description automatically generated

#modify creds.conf file device\_id,api\_key and endpoint

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 = 201303Graphical user interface, text

Description automatically generated

#build the Application (in case only creds file is modified, this step is not required)

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

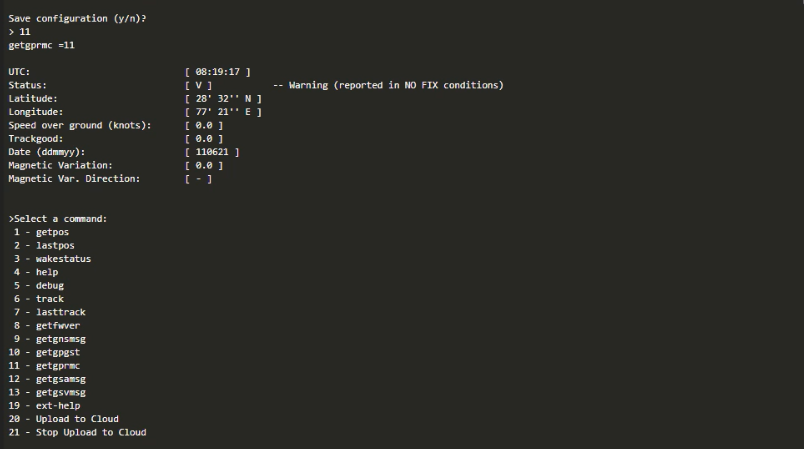


Figure Running gnss app and enabling cloud upload option (“20”)

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

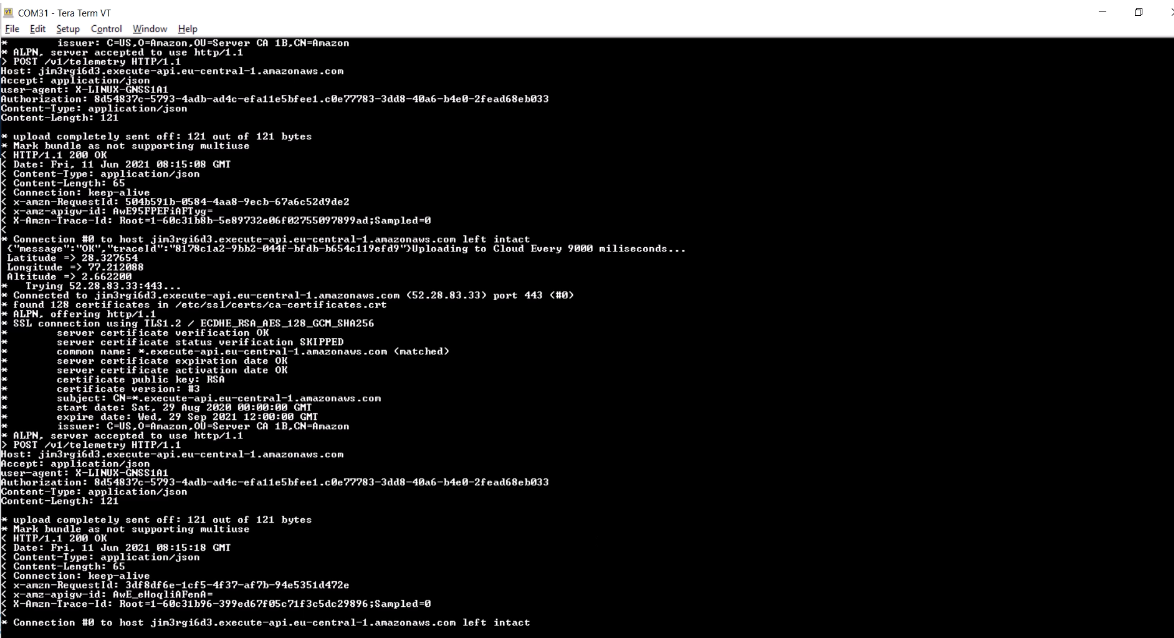


Figure Gnss app sending data to ST Asset Tracking Dashboard over HTTP

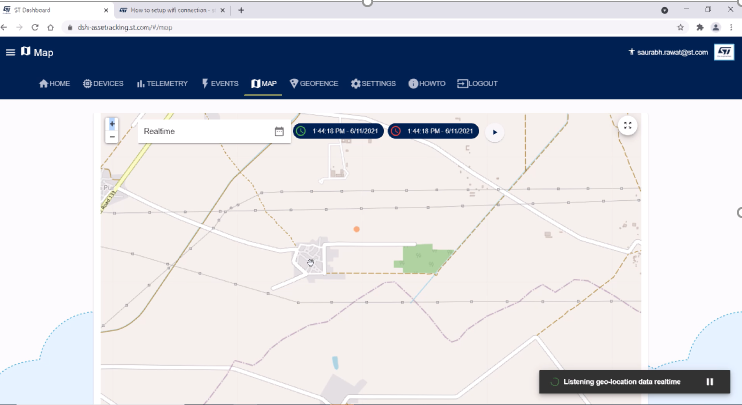


Figure Gnss app live data

## 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)

Text

Description automatically generated

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

$bitbake st\_image\_weston

Graphical user interface, text

Description automatically generated

#You will be inside the build directory. Once bitbake is done, download the gnss package from [www.st.com](http://www.st.com) inside the build directory.

$ git clone <git url> or download the package from www.st.comText

Description automatically generated

#Create a layer : meta-gnss1

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

A screenshot of a computer

Description automatically generated with medium confidence

#Add a layer a layer : recipes-gnss1

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

A screenshot of a computer

Description automatically generated with medium confidence

#See the added layer layer : meta-gnss1

$bitbake-layers show-layers

Graphical user interface, text

Description automatically generated

#Add the IMAGE\_INSTALL\_append line as shown below in the layer.conf at the end

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

IMAGE\_INSTALL\_append += "gnss1"

Text

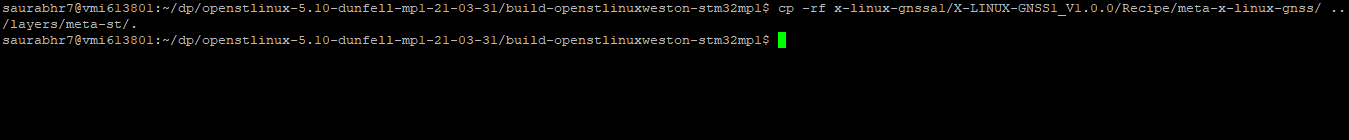
Description automatically generated

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

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

#And copy the layer provided from the x-linux-package

$ cp -rf X-LINUS-GNSS1/X-LINUX-GNSS\_V1.0.0/Recipe/meta-gnss1/ /path-to-layers/layers/meta-st/.

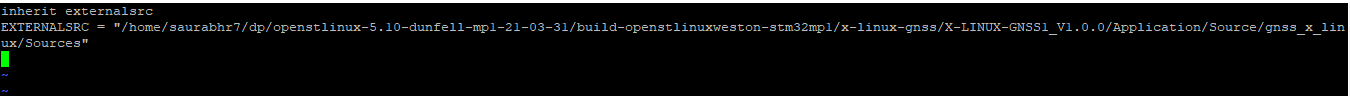


#Add the Sources path(Location where CMakeLists.txt is present) inside gnss1\_0.1.bbappend as shown below. ( by default in the package what path will be?) If no path is present the user need

# “/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-x-linux-gnss/recipes-gnss1/gnss1/gnss1\_0.1.bbappend

$vi ../layers/meta-st/meta-gnss1/recipes-gnss1/gnss1/gnss1\_0.1.bbappend



#Build the ST image

$ bitbake st-image-weston

#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 |
| 6th June 2021 | 1.0 | Initial Release |

**Contents**

[1. X-LINUX-GNSS1 overview 1](#_Toc77280010)

[1.1 Features 2](#_Toc77280011)

[1.2 Architecture 3](#_Toc77280012)

[1.3 Software Package Structure 3](#_Toc77280013)

[1.3.1 GNSS\_APP 5](#_Toc77280014)

[1.3.2 C Utility 6](#_Toc77280015)

[1.3.3 Python Code 6](#_Toc77280016)

[2.Hardware Setup 7](#_Toc77280017)

[3. Software Setup 9](#_Toc77280018)

[3.1 Recommended PC prerequisites 9](#_Toc77280019)

[3.2 Installing the SDK 9](#_Toc77280020)

[3.3 Downloading the kernel Sources (Developer Package) 9](#_Toc77280021)

[3.4 Downloading the Distribution Package 9](#_Toc77280022)

[3.5 Connecting to the Board 10](#_Toc77280023)

[4.Building and Running the example 10](#_Toc77280024)

[4.1 Using Makefile (For Starter Package) 10](#_Toc77280025)

[4.1.1 gnss\_app 11](#_Toc77280026)

[4.1.2 C Utility 14](#_Toc77280027)

[4.1.3 Python Code 15](#_Toc77280028)

[4.1.4 Maps and Asset Tracking 18](#_Toc77280029)

[4.2 Using BitBake (For Distribution Package) 24](#_Toc77280030)

[5.0 Revision History 28](#_Toc77280031)

**List of Figures**

[Figure 1 X-LINUX Software Architecture Diagram 1](#_Toc77280236)

[Figure 2 Hardware Connections 2](#_Toc77280237)

[Figure 3 Application Architecture 3](#_Toc77280238)

[Figure 4 Release Package Structure Top Level 4](#_Toc77280239)

[Figure 5 Application Folder Overview 4](#_Toc77280240)

[Figure 6 GNSS Other Folder Structures 5](#_Toc77280241)

[Figure 7 gnss\_app 5](#_Toc77280242)

[Figure 8 C utility – gnss-uart and gnss-i2c 6](#_Toc77280243)

[Figure 9 Python Code 6](#_Toc77280244)

[Figure 10 Hardware Setup – Jumper Settings 7](#_Toc77280245)

[Figure 11 Cloning the Package 11](#_Toc77280246)

[Figure 12 STM32MPU - Build Application 12](#_Toc77280247)

[Figure 13 STM32MPU - Run Application 13](#_Toc77280248)

[Figure 14 gnss\_app Selecting the options - Select 11 to get the GPS Co-ordinates 14](#_Toc77280249)

[Figure 15 running gnss\_uart 15](#_Toc77280250)

[Figure 16 running gnss\_i2c 15](#_Toc77280251)

[Figure 17 Installing Python Dependencies - Pyserial 16](#_Toc77280252)

[Figure 18 Installing Python Dependencies – Pynmea2 17](#_Toc77280253)

[Figure 19 Python example running 17](#_Toc77280254)

[Figure 20 ST Asset Tracking Dashboard 18](#_Toc77280255)

[Figure 21 ST Asset Tracking Login 19](#_Toc77280256)

[Figure 22 ST Asset Tracking Dashboard Home 20](#_Toc77280257)

[Figure 23 ST Asset Tracking Dashobard Creating a device ID and device Name 20](#_Toc77280258)

[Figure 24 ST Asset Tracking Generating new API Key 21](#_Toc77280259)

[Figure 25 ST Asset Tracking - Note down the endpoint 21](#_Toc77280260)

[Figure 26 Running gnss app and enabling cloud upload option (“20”) 22](#_Toc77280261)

[Figure 27 Gnss app sending data to ST Asset Tracking Dashboard over HTTP 23](#_Toc77280262)

[Figure 28 Gnss app live data 23](#_Toc77280263)

**List of Tables**

[Table 1 Jumper Settings 8](#_Toc77280339)

[Table 2 Revision History 28](#_Toc77280340)