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 and STM32MP2 microprocessor on the STM32MP157F-DK2 and STM32MP257F-EV1 discovery kit to demonstrate GNSS-based applications.

X-LINUX-GNSS1 includes user space application (app) and Yocto recipe (meta) for the Teseo-LIV3FL/Teseo-LIV4F based X-STM32MP-GNSSx boards.

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

It also contains the user space application for NMEA protocol support and POSIX thread for task scheduling.

It also has a separate application for RTK (Real Time Kinematics Library and corresponding QT Application) using GNSS and RTCM3 Messages.

Package also contains EKF (Extended Kalman Filter) application which fuses IMU data + GPS data using EKF to improve accuracy.

The source code can be ported to any Linux platform using X-STM32MP-GNSSx boards.

1 Glossary

NMEA: National Marine Electronics Association
GNSS: Global navigation satellite system
RTCM: The Radio Technical Commission for Maritime Services

2 X-LINUX-GNSS1 overview

The X-LINUX-GNSS1 software provides a user space application running on STM32MP257F-EV1 and STM32MP157F-DK2 for the X-STM32MP-GNSS1 expansion board based on the Teseo-LIV3FL tiny global navigation satellite system (GNSS) module and for the X-STM32MP-GNSS2 expansion board based on Teseo-LIV4F tiny GNSS module.

The software package contains the following modules:

- User space applications : x-linux-gnss1-app
- Meta Yocto recipe : meta-st-x-linux-gnss1 (for gnss app and RTK Lib app)

User Applications are meant for Starter Package (see Section 6: References) but can be built with Developer Package (see Section 6: References) as well.

Meta Yocto Recipes are meant for Distribution Package (see Section 6: References).

Each software module can run independently to acquire the GNSS NMEA data from the X-STM32MP-GNSS1/X-STM32MP-GNSS2 over UART or I²C.

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

Figure 2. RTKLib QT application architecture

The device tree of both X-STM32MP-GNSS1 and X-STM32MP-GNSS2 has been modified to configure UART1 and I2C1 on 40 Pin GPIO Header of the STM32MP257F-EV1 and STM32MP157F-DK2 connector. For UART, the underlying /dev/ttySTM1 is enabled, whereas /dev/i2c-1 is enabled for I²C to retrieve the Data from the GNSS module.

Data from MEMS (ISM330DHCX, IIS2MDC) and Environmental (ILPS22QS) Sensors are fetched from the corresponding Linux Kernel Drivers

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 and Linux Kernel Drivers for reading Sensors data through I2C.

The software also uses 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 3. X-STM32MP-GNSS1 40 pin hardware connections with STM32MP157F-DK2 board

Figure 4. X-STM32MP-GNSS2 40 pin hardware connections with STM32MP157F-DK2 board

Figure 5. X-STM32MP-GNSS2 40 pin hardware connections with STM32MP257F-EV1 board

2.1 Features

- Standalone applications to read the NMEA data over UART and I²C
- Example RTK Lib and QT Application based on RTCM3 (X-STM32MP-GNSS1) Messages
- Complete software to build applications on Linux using X-STM32MP-GNSS1/X-STM32MP-GNSS2 board connected to 40 pin header of STM32MP257F-EV1 and STM32MP157F-DK2 board
- 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

2.2 Architecture

The software package runs on the ARM Cortex-A7 core of the STM32MP157F-DK2 and Arm Cortex-A35 of the STM32MP257F-EV1. The X-LINUX-GNSS1 interacts with the lower layers interfaces - I^2C and USART exposed by the Linux software framework.

Figure 6. X-LINUX-GNSS1 application architecture in Linux environment

2.3 Software package structure

The X-LINUX-GNSS1 release package contains Linux user space application C examples, Python example, the device tree and a Yocto layer recipe.

Any application can be run independently in the Application folder to retrieve the GNSS NMEA data.

Figure 7. X-LINUX-GNSS1 package Structure

Figure 8. **X-LINUX-GNSS Yocto Recipe Structure**

Figure 9. **X-LINUX-GNSS User Space Application Structure**

2.3.1 gnss_i2c, gnss_uart, gnss_x_linux

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

2.3.2 RTK Library Application

This application is used to get the RTK Application data using GNSS and RTCM3 messages from X-STM32MP-GNSS1 board. It has been provided in the form of patch.

To use this RTCM3 messages must be enabled as explained in the Section 5.2.2.1.

2.3.3 Python code application

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

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

3 Hardware setup

Step 1. Set the X-STM32MP-GNSS1 and X-STM32MP-GNSS2 jumpers as shown in the figures below.

Figure 10. X-STM32MP-GNSS1 expansion board on top of 40 Pin GPIO Header

Figure 11. X-STM32MP-GNSS2 expansion board on top of 40 Pin GPIO Header

Tip: Keep jumpers as shown in the image.

Step 2. Connect the antenna provided with X-STM32MP-GNSSx board. For X-STM32MP-GNSS1 use INPAQ B3G02G-S3-XX-A while with X-STM32MP-GNSS2 use Taoglas EAHP.50.01.0100D GNSS Antenna.

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

The STM32MP157F-DK2 and STM32MP257F-EV1 is powered by the USB Type-C cable.

4 Software setup

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

4.1 PC/laptop requirements

For the software setup, you need:

- a Linux® PC/laptop running Ubuntu® 18.04 or 20.04 or 22.04
- detailed instructions at https://wiki.st.com/stm32mpu/wiki/PC_prerequisites

4.2 Installing the SDK

The software package contains the binaries inside x-linux-gnss1-app\Application\Binaries folder which can be transfered to the STM32MPU Board.

Install the SDK to build the software package.

4.3 Downloading the kernel sources (developer package)

Downloading the kernel sources (see Section 6: References) is required to build the user space application along with the device tree. The software package already contains the binaries (dtb) which can be transferred using scp command.

KERNEL SOURCE PATH = <path to Linux kernel>/linux-stm32mp-5.15.67-r0/ linux-5.15.67$

4.4 Downloading the distribution package

This is required to build the recipes and create STM32MP1/ STM32MP2 image which has GNSS application and device tree settings embedded (see Section 6: References).

4.5 Connecting to the discovery kit

To transfer the built binaries (application, device trees) to the STM32MP157F-DK2 and STM32MP257F-EV1 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).

You can also transfer it using minicom or hyper terminal Utility like teraterm (using zmodem).

5 Building and running the example

The code can be built using simple make command 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.

5.1 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-GNSS1) from www.st.com and extract it.

```
$tar xvf X-LINUX-GNSS1_VX.Y.Z
```

You will get the X-LINUX-GNSS1_VX.Y.Z folder.

5.1.1 How to build the gnss_app

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

Step 2. Download the kernel sources as described in Section 4.3.

Step 3. Copy the dts file from the directory: x-linux-gnss1-app\linux-kernel\patch\5.15\kernel to the kernel source directory at <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
```

Step 5. Build the device tree.

Step 6. Once the dtbs are built, copy them to the STM32MPU Board via hotspot or Wi-Fi.

Follow deploy steps from wiki page of modify, rebuild and reload linux kernel for relevant MPU Board (see Section 6: References).

Step 7. cd to the gnss_x_linux location and run make

```
$cd path to/X-LINUX-GNSS_VX.Y.Z/x-linux-gnss1-app\Application\Source
```

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

Step 8. Transfer the application to STM32MP157F-DK2.

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

Step 9. On the STM32MPU Board, sync and reboot.

```
$sync
$reboot
```

Step 10. Run the application.

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

Figure 12. STM32MP157F-DK2 - running the application

Step 11. From the menu above, select 11 to get the GPS coordinates or select 20 to upload the GNSS location to asset tracking dashboard.

Figure 13. Selecting option 11 from the gnss_app

5.1.2 How to build C utility gnss_uart and gnss_i2c

Step 1. Enter the <Path to > /X-LINUX-GNSS1_VX.Y.Z\Application\Source\gnss_uart and run make gnss_uart or gnss_i2c will be created in the same directory of Makefile.

Step 2. Transfer it to STM32MPU Board using scp and make it executable using chmod +x command

```
$cd /
#for GNSS1 expansion Board, baud rate 9600

./gnss_uart_read 9600

#for GNSS2 expansion Board, baud rate 115200

./gnss_uart_read 115200
```

Figure 14. Running gnss_uart

```
$ ./gnss_i2c_read
```

Figure 15. **Running 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
```

5.1.3 How to run the Python code

Step 1. Install the pyserial.

Figure 16. **Installing Python dependencies - pyserial**

```
$wget https://files.pythonhosted.org/packages/1e/7d/ae3f0a63f41e4d2f6cb66a5b57197850
f919f59e558159a4dd3a818f5082/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 17. Installing Python dependencies - pynmea2

```
$wget https://files.pythonhosted.org/packages/88/b9/a0fed4563f5c73eb8f4d7bb115a45586
3c5327ae824ac1772e2a4b1b95ee/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 STM32MPU Board.

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

Step 4. Run the Python example.

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

Figure 18. Running the Python example

5.1.4 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.

To upload the GNSS data to dashboard you need device Id, API Key and the End point which you have to put in the creds.conf file.

Figure 19. **DSH-ASSETRACKING asset tracking dashboard homepage**

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

Figure 20. **DSH-ASSETRACKING login page**

Step 2. Create a device (device name and device ID) from the [**Devices**] tab. The device id created here to be put in the creds.conf file

Figure 21. DSH-ASSETRACKING - Devices tab

Figure 22. DSH-ASSETRACKING - device name and ID - Select HTTP Client

Figure 23. **DSH-ASSETRACKING - device name and ID**

Figure 24. **DSH-ASSETRACKING - Submit**

Figure 25. DSH-ASSETRACKING - Device On Dashboard

Step 3. Create the API key which will be used to send data to DSH-ASSETRACKING. Click [SETTINGS] Tab

Figure 26. DSH-ASSETRACKING - new API key generation

Figure 27. DSH-ASSETRACKING - Copy API Key

Step 4. 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_VX.Y.Z/Application/Source/gnss_x_linux/Sources/cloud
$vi creds.conf
```

Figure 28. Modifying creds.conf

Step 5. Modify the data in the 'cloud_comm_https.c' file as per the device ID created for the device as:

```
pgps->device_id = 123456(device ID)
$cd /path-to/x-linux-gnss1-app/Application/Source/gnss_x_linux/Sources/cloud
```

Figure 29. Modifying cloud_comm_https.c

Figure 30. cloud_comm_https.c file modified

Step 6. Build the application (if the creds file only has been modified, this step is not required).

Step 7. Repeat the steps done to build and deploy the gnss_app.

Step 8. Enter option 20 to upload the data to the cloud and make sure the STM32MPU Board is connected to the Internet.

Figure 31. 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 32. gnss_app sending data to DSH-ASSETRACKING over http

Figure 33. gnss_app data in real time

5.2 Using BitBake (for distribution package)

5.2.1 Building and deploying Yocto layer for gnss Application

Step 1. Download the Distribution Package (see Section 6: References) and do a bitbake.

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

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

Figure 34. End user license agreement

After accepting, you will access the build directory.

Step 3. Download and copy meta-gnss1 yocto recipe into meta-st folder `cp -rf ~/X-LINUX-GNSS1-VX.Y.Z/meta-st-x-linux-gnss1/Layers/meta-st-x-linux-gnss1/meta-gnss1/ ../layers/meta-st/`

Step 4. Add a layer (recipes-gnss1).

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

Step 5. Add "gnss1" in IMAGE_INSTALL: append line at the end of the layer.conf.

```
vi ../layers/meta-st/meta-st-openstlinux/conf/layer.conf
IMAGE_INSTALL:append = "gnss1"
```

You can see the added layer by typing (optional):

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

Figure 35. Showing all meta layers

Step 6. Add the Sources path (location containing CMakeLists.txt) inside gnss1_0.1.bbappend as shown below.

```
Path to X-LINUX-GNSS1\x-linux-gnss1-app\Application\Source\gnss_x_linux\Sources
```

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
```

Step 7. Build the ST image.

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

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

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

Step 8. Flash the binary following the instructions at https://wiki.st.com/stm32mpu/wiki/Getting_started.

Step 9. Check if the file below is present on the discovery kit.

```
$ ls -l /dev/ttySTM1
```

Step 10. Run the application.

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

or

```
gnss_app
```

5.2.2 Building and deploying Yocto layer for RTK Lib application

Make sure to add the gnss1 layer before.

5.2.2.1 Configure RTCM3 messages on X-STM32MP-GNSS1

This section describes how to enable RTCM3 messages on Teseo-LIV3FL module on X-STM32MP-GNSS1 board. RTKLIB currently supports only RTCM3 messages.

Step 1. Connect to Board using FTDI (shown in red color below).

Enable the RTCM3 message by sending RTCM enable message directly to Teseo-LIV3FL module. Connect FTDI cable or connector to the board as shown below (use the image from the slide 1). The cable/connector is connected to Laptop running Teseo-Suite.

Connect hardware as shown below:

Figure 36. Rtklib: hardware setup

Step 2. Launch Teseo Suite:

X-STM32MP-GNSS1 board will be automatically detected (if not check connections again and make sure FTDI VCP drivers are installed on Windows PC. After installing drivers reconnect FTDI cable to PC and relaunch Teseo-suite).
Below UI will be seen if connection is successful.

Figure 37. rtklib Teseo suite

Step 3. Send RTCM3 Enable Message to X-STM32MP-GNSS1 board

Step 3a. Click on Eye icon as shown below, NMEA decoding window will appear showing messages received by GNSS1 module.

Figure 38. rtklib: eye icon

Step 3b. Click on Edit button of NMEA decoding Window

Figure 39. rtklib: Edit button

New GUI as shown in Figure 40 will appear.

Step 3c. write any name into NMEA scripts and Edited script box, Press enter, execute NMEA script option will come. Then write below script contents into script box.

```
$PSTMSETPAR,1200,0x1000000,2
$PSTMSETPAR,1227,0x8,2
$PSTMSETPAR,1200,0x10000800,1
$PSTMSETPAR,1227,0x20000,1
$PSTMSETPAR,1201,0x0
$PSTMSETPAR,1228,0x0
$PSTMSETPAR,1201,0x2
$PSTMSETPAR,1228,0x0
$PSTMSAVEPAR
$PSTMSRR
```

Figure 40. rtklib: rtcm3 script

Execute script, wait for some time and then close Teseo-Suite Tool and disconnect FTDI cable.

Figure 41. rtklib: execute script

5.2.2.2 *Adding RTKLIB into distribution build*

Step 1. Download and copy meta-gnss1-rtklib yocto recipe into meta-st folder.

Step 2. Add a layer (recipes-gnss1-rtklib).

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

Step 3. Add "gnss1-rtklib" in IMAGE_INSTALL: append line at the end of the layer.conf.

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

Figure 42. gnss1-rtklib

```
LAYERVERSION_st-openstlinux = "1"
LAYERSERIES_COMPAT_st-openstlinux = "kirkstone"

INHERIT += "check-st-openstlinux-compatibility"
# OpenSTLinux compatibility version
ST_OSTL_COMPATIBILITY_VERSION_st-openstlinux = "4.1"
IMAGE_INSTALL:append = "gnss1 gnss1-rtklib"

~
```

Step 4. Add "gnss1-rtklib" inside the st-image-userfs.bb

```
vi ../layers/meta-st/meta-st-stm32mp/recipes-st/images/st-image-userfs.bb

PACKAGE_INSTALL += " \
packagegroup-st-demo \
gnss1-rtklib \
"
```

Step 5. Build QT image.

```
$ bitbake st-example-image-qtwayland
```

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

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

Step 6. Flash the binary following the instructions at https://wiki.st.com/stm32mpu/wiki/Getting_started.

5.2.2.3 *Launching and Using RTKLIB Application*

Step 1. Move over to RTKLib launch icon and click on it.

Figure 43. rtklib: launch rtklib

RTKLIB Application will be launched and below GUI will appear

Figure 44. rtklib: main menu

Step 2. Click on Rover Option, Configurations Choice menu will come. Click on CUSTOMIZED CONFIGURATIONS Tab.

Figure 45. rtklib: configurations

Step 3. Select Custom2 configuration.

Figure 46. rtklib: custom2

Stream UI will come as shown below.
In-Byte and In-Bps should be incremental.

Figure 47. rtklib: stream UI

Step 4. Click at random place on Stream UI, Figure 49. rtklib: satellites UI will appear.

Figure 48. rtklib: Stream menu

Step 5. Click on satellites to see satellite info.

Figure 49. rtklib: satellites

Step 6. Click on azimuth to see azimuth data.

Figure 50. rtklib: azimuth

Step 7. Click on elevation to see elevation data.

Figure 51. rtklib: elevation

Step 8. To stop , click anywhere in UI and click on STOP button. main menu will come again.

Figure 52. rtklib: stop

Step 9. To close, click on Close GUI option.

Figure 53. rtklib: close

5.2.3 Building and deploying Yocto layer for EKF application

Make sure to add the gnss1 layer before.

5.2.3.1 Enabling GPS Velocity Data in X-STM32MP-GNSS1 and X-STM32MP-GNSS2 Board

Note: Make sure GNSS1 yocto package is flashed onto board, only then ttySTM1 will be visible, if building gnss1 and gnss1-ekf together, this step can be done after flashing image and before running EKF.

Also make sure X-STM32MP-GNSS1/GNSS2 is connected to MPU.

Step 1. For X-STM32MP-GNSS1 expansion , give following one commands on STM32MPU Terminal.

Note: Use single quotes only not double quotes in commands.

```
echo '$PSTMSETPAR,1228,1,1' > /dev/ttySTM1
echo '$PSTMSAVEPAR'  > /dev/ttySTM1
echo '$PSTMSRR'  > /dev/ttySTM1
```

For X-STM32MP-GNSS2 expansion board , give following one commands on STM32MPU Terminal

Note: Use single quotes only not double quotes in commands.

```
echo '$PSTMSETPAR,1228,4000,1' > /dev/ttySTM1
echo '$PSTMSAVEPAR' > /dev/ttySTM1
echo '$PSTMSRR'  > /dev/ttySTM1
```

Step 2. Validate velocity data is enabled.

Step 2a. Take setup to outdoor, antenna connected to GNSS expansion board.

Step 2b. Issue below command on MPU.

```
cat /dev/ttySTM1 | grep PSTMPV
```

Note:

1. *For X-STM32MP-GNSS1, PSTMPV messages will be seen for X-STM32MP-GNSS2, PSTMPVRAW messages will be seen if PSTMPV/PSTMPVRAW messages are seen, it confirms velocity data has been successfully enabled.*
2. *For X-STM32MP-GNSS2, velocity data feature is available for firmware version > = 4.6.6.5.5. in case older FW version is present, to get latest FW version can get in touch with support team.*

5.2.3.2 Adding EKF into distribution build

Step 1. Download and copy meta-gnss1-ekf yocto recipe into meta-st folder.

Step 2. Add a layer (recipes-gnss1-ekf).

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

Step 3. Add "gnss1-ekf" in IMAGE_INSTALL: append line at the end of the layer.conf.

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

Figure 54. Add gnss1-ekf in recipe

```
# 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-openstlinux"
BBFILE_PATTERN_st-openstlinux := "^${LAYERDIR}/"
BBFILE_PRIORITY_st-openstlinux = "5"

LAYERDEPENDS_st-openstlinux = "qt5-layer"

# Set a variable to get the openstlinux location
OPENSTLINUX_BASE = "${LAYERDIR}"

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

INHERIT += "check-st-openstlinux-compatibility"
# OpenSTLinux compatibility version
ST_OSTL_COMPATIBILITY_VERSION_st-openstlinux = "5.0"

IMAGE_INSTALL:append = "gnss1  gnss1-ekf"
```

Step 4. Add "gnss1-ekf" inside the st-image-userfs.bb

```
vi ../layers/meta-st/meta-st-stm32mp/recipes-st/images/st-image-userfs.bb
PACKAGE_INSTALL += " \
packagegroup-st-demo \
gnss1-ekf \
"
```

Step 5. Build QT image.

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

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

Step 6. Flash the binary following the instructions at https://wiki.st.com/stm32mpu/wiki/Getting_started.

5.2.3.3 *Launching and using EKF application*

Step 1. Launch Terminal and execute script to run EKF application (run_ekf_gnss1.sh for X-STM32MP-GNSS1 and run_ekf_gnss2.sh for X-STM32MP-GNSS2).

Figure 55. EKF: launch ekf script

Step 2. On console user can see log Start logging data signifying logging has started.

Figure 56. EKF: logging started

```
Stream : Closed

/dev/ttySTM1 is connected successfully!!!

-------------------------------------------------------
Stream : Closed
Test OUTDOOR macro case
readGPS_thread started

/dev/pts/9 is connected successfully!!!
gpsThread
serial config done
no serial Data
non zero value found:7729.491211, 2829.294922
^Z[1]+  Stopped                 run_ekf.sh
root@stm32mp1:~# run_ekf.sh
MAG: 0
ACCEL: 2
GYRO 1

/dev/pts/10 is connected successfully!!!

/dev/pts/11 is connected successfully!!!

/dev/pts/12 is connected successfully!!!
ttyName: /dev/ttySTM1magn device node:/sys/bus/iio/devices/iio:device0/

tmagnet id:0
unable to open:/sys/bus/iio/devices/iio:device0/in_magn_z_raw
accel device node:/sys/bus/iio/devices/iio:device2/

accel ID: 2gyro device node:/sys/bus/iio/devices/iio:device1/

gyro ID:1imu options passed correctly

/dev/ttySTM1 is connected successfully!!!
Stream : Closed

/dev/ttySTM1 is connected successfully!!!

-------------------------------------------------------
Stream : Closed
Test OUTDOOR macro case
readGPS_thread started

/dev/pts/13 is connected successfully!!!
gpsThread
serial config done
non zero value found:7729.509277, 2829.291992
new latlong found:7729.509277, 2829.292236
Start logging Data,as new latlong value recieved!!!
```

Step 3. One can move around and both GPS + EKF corrected data and raw is collected and stored in a gnss.csv file at location from where command was launched.

Step 4. Once user is done collecting data, can press CTRL+Z to stop the data recording. and then can run stop_ekf.sh to kill the recording process.

Figure 57. EKF: logging started

Step 5. Copy data from MP board to external machine to visualize data using matplotlib.

Figure 58. EKF: copy data

```
root@stm32mp1:/home/weston#
root@stm32mp1:/home/weston# ls /dev/sd*
/dev/sda   /dev/sda1
root@stm32mp1:/home/weston# mount /dev/sda1 /media
[ 1568.589308] FAT-fs (sda1): Volume was not properly unmounted. Some data may b
e corrupt. Please run fsck.
root@stm32mp1:/home/weston# cp ./gnss.csv /media/
root@stm32mp1:/home/weston# umount /media/
root@stm32mp1:/home/weston#
```

Step 6. Visualize data using analyze_lonlat.py and arg_plot_coords.py python scripts present in gnss1 package.

Note:
1. *Check py scripts and make sure correct paths are provided*
2. *Make sure python plugins matplotlib, numpy and pandas are installed*

Figure 59. EKF: analyze lat-long distance(meters)

Figure 60. EKF: plot raw lat-long comparison

Data shown in sample figure was collected while walking, can see spikes in raw GPS data where as GPS+EKF data is more stable and accurate.

6 References

Table 1. References

Package	Reference
Starter package	
STM32MP157F-DK2	https://www.st.com/en/embedded-software/stm32mp1starter.html
STM32MP257x-EV1	https://www.st.com/en/embedded-software/stm32mp2starter.html
Developer Package	
STM32MP157F-DK2	https://wiki.st.com/stm32mpu/wiki/STM32MPU_Developer_Package
STM32MP257x-EV1	https://wiki.st.com/stm32mpu/wiki/STM32MPU_Developer_Package
Distribution Package	
STM32MP157F-DK2	https://wiki.st.com/stm32mpu/wiki/STM32MPU_Distribution_Package#Installing_the_OpenSTLinux_distribution
STM32MP257x-EV1	https://wiki.st.com/stm32mpu/wiki/STM32MPU_Distribution_Package#Installing_the_OpenSTLinux_distribution
Modify and build kernel and device tree	
STM32MP157F-DK2	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
STM32MP257x-EV1	https://wiki.st.com/stm32mpu/wiki/Getting_started/STM32MP2_boards/STM32MP257x-EV1/Develop_on_Arm%C2%AE_Cortex%C2%AE-A35/Modify,_rebuild_and_reload_the_Linux%C2%AE_kernel

Revision history

Table 2. 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 4.2.1 Building and deploying Yocto layer for gnss Application.
22-Nov-2023	3	Updated Section Introduction, Section 1 X-LINUX-GNSS1 overview, Section 2 Hardware setup and Section 4 Building and running the example.
18-Dec-2023	4	Added references to X-STM32MP-GNSS2 and Teseo-LIV4F. Updated *Section Introduction, Section 1 X-LINUX-GNSS1 overview, Section 1.1 Features, Section 2 Hardware setup* and *Section 4.2.2.1 Configure RTCM3 messages on X-STM32MP-GNSS1.*
29-Apr-2024	5	Modified *Section Introduction.* Added *Section 4.2.3: Building and deploying Yocto layer for EKF application, Section 4.2.3.1: Adding EKF into distribution build* and *Section 4.2.3.2: Launching and using EKF application.*
06-Nov-2024	6	Updated Section Introduction, Section 2: X-LINUX-GNSS1 overview, Section 2: X-LINUX-GNSS1 overview, Section 2.2: Architecture, Section 3: Hardware setup, Section 4.3: Downloading the kernel sources (developer package), Section 4.4: Downloading the distribution package and Section 5: Building and running the example. Added Section 6: References.

Contents

1 Glossary 2
2 X-LINUX-GNSS1 overview. 3
2.1 Features 7
2.2 Architecture 8
2.3 Software package structure 8
2.3.1 gnss_i2c, gnss_uart, gnss_x_linux 9
2.3.2 RTK Library Application. 9
2.3.3 Python code application 9
3 Hardware setup. 10
4 Software setup 12
4.1 PC/laptop requirements 12
4.2 Installing the SDK 12
4.3 Downloading the kernel sources (developer package). 12
4.4 Downloading the distribution package 12
4.5 Connecting to the discovery kit 12
5 Building and running the example 13
5.1 Using Makefile (for starter package). 13
5.1.1 How to build the gnss_app 13
5.1.2 How to build C utility gnss_uart and gnss_i2c 15
5.1.3 How to run the Python code 17
5.1.4 Maps and asset tracking 18
5.2 Using BitBake (for distribution package) 25
5.2.1 Building and deploying Yocto layer for gnss Application 25
5.2.2 Building and deploying Yocto layer for RTK Lib application 27
5.2.3 Building and deploying Yocto layer for EKF application. 38
6 References 44
Revision history 45
List of figures. 47

List of figures

Figure 1. X-LINUX-GNSS1 user space application architecture . . . 3
Figure 2. RTKLib QT application architecture . . . 4
Figure 3. X-STM32MP-GNSS1 40 pin hardware connections with STM32MP157F-DK2 board . . . 5
Figure 4. X-STM32MP-GNSS2 40 pin hardware connections with STM32MP157F-DK2 board . . . 5
Figure 5. X-STM32MP-GNSS2 40 pin hardware connections with STM32MP257F-EV1 board . . . 6
Figure 6. X-LINUX-GNSS1 application architecture in Linux environment . . . 8
Figure 7. X-LINUX-GNSS1 package Structure . . . 8
Figure 8. X-LINUX-GNSS Yocto Recipe Structure . . . 9
Figure 9. X-LINUX-GNSS User Space Application Structure . . . 9
Figure 10. X-STM32MP-GNSS1 expansion board on top of 40 Pin GPIO Header . . . 10
Figure 11. X-STM32MP-GNSS2 expansion board on top of 40 Pin GPIO Header . . . 10
Figure 12. STM32MP157F-DK2 - running the application . . . 14
Figure 13. Selecting option 11 from the gnss_app . . . 14
Figure 14. Running gnss_uart . . . 16
Figure 15. Running gnss_i2c . . . 17
Figure 16. Installing Python dependencies - pyserial . . . 17
Figure 17. Installing Python dependencies - pynmea2 . . . 18
Figure 18. Running the Python example . . . 18
Figure 19. DSH-ASSETRACKING asset tracking dashboard homepage . . . 19
Figure 20. DSH-ASSETRACKING login page . . . 19
Figure 21. DSH-ASSETRACKING - Devices tab . . . 20
Figure 22. DSH-ASSETRACKING - device name and ID - Select HTTP Client . . . 20
Figure 23. DSH-ASSETRACKING - device name and ID . . . 21
Figure 24. DSH-ASSETRACKING - Submit . . . 21
Figure 25. DSH-ASSETRACKING - Device On Dashboard . . . 22
Figure 26. DSH-ASSETRACKING - new API key generation . . . 22
Figure 27. DSH-ASSETRACKING - Copy API Key . . . 22
Figure 28. Modifying creds.conf . . . 23
Figure 29. Modifying cloud_comm_https.c . . . 23
Figure 30. cloud_comm_https.c file modified . . . 23
Figure 31. Running gnss_app and enabling cloud upload (option 20) . . . 24
Figure 32. gnss_app sending data to DSH-ASSETRACKING over http . . . 25
Figure 33. gnss_app data in real time . . . 25
Figure 34. End user license agreement . . . 26
Figure 35. Showing all meta layers . . . 26
Figure 36. Rtklib: hardware setup . . . 27
Figure 37. rtklib Teseo suite . . . 28
Figure 38. rtklib: eye icon . . . 29
Figure 39. rtklib: Edit button . . . 29
Figure 40. rtklib: rtcm3 script . . . 30
Figure 41. rtklib: execute script . . . 30
Figure 42. gnss1-rtklib . . . 31
Figure 43. rtklib: launch rtklib . . . 32
Figure 44. rtklib: main menu . . . 33
Figure 45. rtklib: configurations . . . 33
Figure 46. rtklib: custom2 . . . 34
Figure 47. rtklib: stream UI . . . 34
Figure 48. rtklib: Stream menu . . . 35
Figure 49. rtklib: satellites . . . 35
Figure 50. rtklib: azimuth . . . 36
Figure 51. rtklib: elevation . . . 36
Figure 52. rtklib: stop . . . 37
Figure 53. rtklib: close . . . 37

Figure 54. Add gnss1-ekf in recipe . . . 39
Figure 55. EKF: launch ekf script . . . 40
Figure 56. EKF: logging started . . . 41
Figure 57. EKF: logging started . . . 41
Figure 58. EKF: copy data . . . 42
Figure 59. EKF: analyze lat-long distance(meters) . . . 43
Figure 60. EKF: plot raw lat-long comparison . . . 43

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.

© 2024 STMicroelectronics – All rights reserved