User Manual: Android GNSS Raw Data to RINEX 3 Converter

Jorge Hernández Olcina[1]*, Ana B. Anquela Julián[1], and Ángel E. Martín Furones[1]

[1]Cartographic Engineering Department, Universitat Politècnica de València, ES.

*E-mail: jorherol@doctor.upv.es, jorgeho1995@gmail.com

Manual Version: 1.0.1

1. Introduction

1.1 Overview:

The Android GNSS Raw Data to RINEX 3 Converter is a specialized tool designed to transform raw Global Navigation Satellite System (GNSS) data collected from Android devices into RINEX 3 format. This conversion is essential for post-processing applications such as precise positioning, navigation, and mapping.

1.2 Key Features:

- Converts Android GNSS raw data to RINEX 3.
- Compatible with GEA and GNSSLogger.
- Real-time progress monitoring during conversion.

2. System Requirements

2.1 Hardware Requirements:

- Android device with GNSS capability.
- Computer with USB connectivity for data transfer.

2.2 Software Requirements:

- Android 7.0 (Nougat) or higher on the mobile device.
- GEA or GNSSLogger app.
- Python 3.o or higher installed on the computer.

3. Installation

First, make sure you have at least Python version 3.6 installed on your system:

```
python --version
```

Then, make sure to set up pip, the Python package installer, navigate to the main directory (where *pyproject.toml* is located) and install the tool using the following command.

```
python -m pip install .
```

This command is used to install a Python package from the current directory.

Here's a breakdown of the command:

- python: Calls the Python interpreter.
- -m pip: Uses the pip module to install packages.
- install: Specifies that you want to install a package.
- .: Refers to the current directory. This indicates that you want to install the package located in the current directory.

When you run this command, it installs the package in a standard (non-editable) mode. In other words, any changes you make to the source code after installation won't automatically affect the installed package. You'd need to reinstall the package to apply code changes.

Ensure that the *pyproject.toml* file is present in the current directory which contains the necessary information for packaging, such as the package name, version, dependencies, etc.

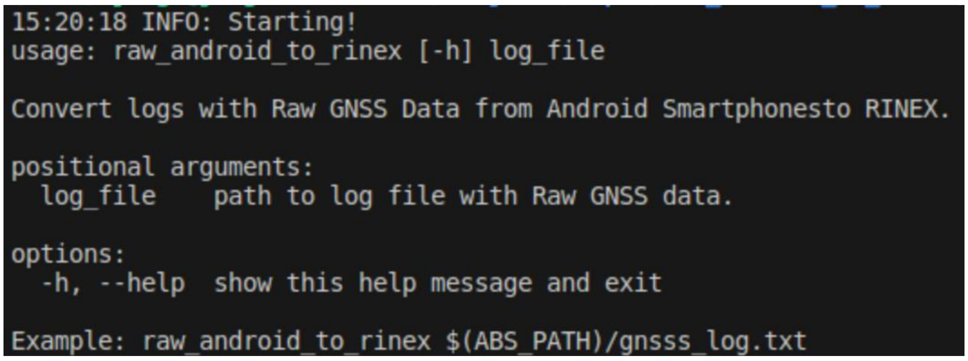
4. Usage

4.1 How to run *'raw_android_to_rinex'* tool.

4.1.1 Using command line:

The script has a small help that explains the different options of the program. This help is accessed through the command:

```
raw_android_to_rinex --help
```



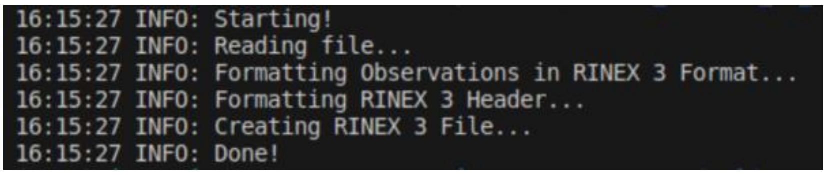
```
15:20:18 INFO: Starting!
usage: raw_android_to_rinex [-h] log_file

Convert logs with Raw GNSS Data from Android Smartphonesto RINEX.

positional arguments:
  log_file    path to log file with Raw GNSS data.

options:
  -h, --help  show this help message and exit

Example: raw_android_to_rinex $(ABS_PATH)/gnsss_log.txt
```

Figure 1. *Help message.*

Process the data file and generate a RINEX file:

```
raw_android_to_rinex INPUT_FILE
```

Replace INPUT_FILE with the path to the input file.



```
16:15:27 INFO: Starting!
16:15:27 INFO: Reading file...
16:15:27 INFO: Formatting Observations in RINEX 3 Format...
16:15:27 INFO: Formatting RINEX 3 Header...
16:15:27 INFO: Creating RINEX 3 File...
16:15:27 INFO: Done!
```

Figure 2. *Tool log when a file is processed.*

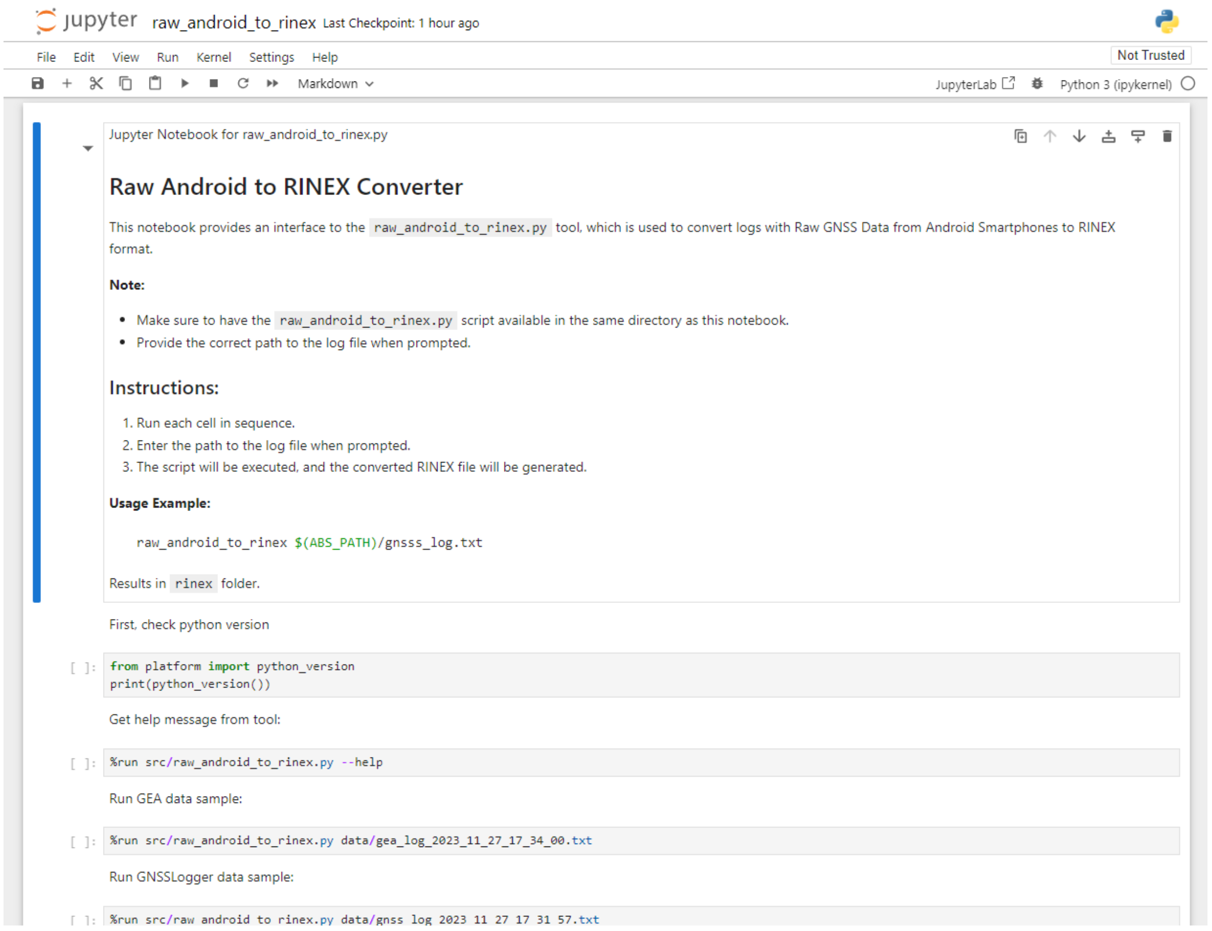
4.1.2 Using Jupyter Notebook example:

This section provides comprehensive instructions on utilizing the tool through a Jupyter Notebook. The accompanying Jupyter Notebook example serves as a hands-on guide to demonstrate the tool's functionality.

- Prerequisites: Before proceeding, ensure that you have the following prerequisites installed:
    1. Python (version 3.6 or higher)
    2. Jupyter Notebook
    3. *raw_android_to_rinex* tool (downloaded and installed)
- Getting Started
    1. Open Jupyter Notebook: Launch Jupyter Notebook from your terminal or command prompt.

       ```
       Jupyter notebook
       ```

    2. Navigate to the directory where you downloaded the example notebook and open it.


jupyter
raw_android_to_rinex Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Not Trusted
Markdown
JupyterLab
Python 3 (ipykernel)

Jupyter Notebook for raw_android_to_rinex.py

Raw Android to RINEX Converter

This notebook provides an interface to the `raw_android_to_rinex.py` tool, which is used to convert logs with Raw GNSS Data from Android Smartphones to RINEX format.

Note:

- Make sure to have the `raw_android_to_rinex.py` script available in the same directory as this notebook.
- Provide the correct path to the log file when prompted.

Instructions:

1. Run each cell in sequence.
2. Enter the path to the log file when prompted.
3. The script will be executed, and the converted RINEX file will be generated.

Usage Example:

```
raw_android_to_rinex $(ABS_PATH)/gnsss_log.txt
```

Results in `rinex` folder.

First, check python version

```
[ ]: from platform import python_version
     print(python_version())
```

Get help message from tool:

```
[ ]: %run src/raw_android_to_rinex.py --help
```

Run GEA data sample:

```
[ ]: %run src/raw_android_to_rinex.py data/gea_log_2023_11_27_17_34_00.txt
```

Run GNSSLogger data sample:

```
[ ]: %run src/raw_android_to_rinex.py data/gnss_log_2023_11_27_17_31_57.txt
```

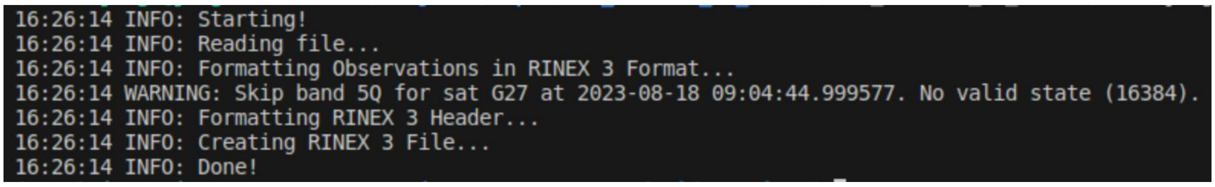
Figure 3. *Jupyter Notebook example.*

3. Run each cell and check the results: The tool will process the raw Android GNSS data and generate RINEX files in the specified output directory.
4. Review the Results: Verify the converted RINEX files in the designated output directory. The filenames typically follow the RINEX naming conventions.

4.2 Tool log

The tool outputs a series of messages to the console, providing the user with information regarding the execution status. There are three distinct message types:

- Info: This type relays informative messages about the execution status.
- Warning: This message appears when one of the observables is deemed invalid and is consequently not utilized. Importantly, this doesn't impact the execution process.
- Error: This message surfaces when an error occurs during execution, resulting in the tool's inability to generate any results.



```
16:26:14 INFO: Starting!
16:26:14 INFO: Reading file...
16:26:14 INFO: Formatting Observations in RINEX 3 Format...
16:26:14 WARNING: Skip band 5Q for sat G27 at 2023-08-18 09:04:44.999577. No valid state (16384).
16:26:14 INFO: Formatting RINEX 3 Header...
16:26:14 INFO: Creating RINEX 3 File...
16:26:14 INFO: Done!
```

Figure 4. *Warning example. In this specific scenario, the tool issues a warning indicating that band 5Q for satellite G27 is being skipped because it is in an invalid state with a value of 16384.*

4.3 Data Preprocessing

Before using the tool, ensure that the GNSS raw data is properly logged by the Android device. The data should be in a text file format with GNSS measurements. An example data file can be found in the data folder of the repository.

5. Usage for developers

Install the package in editable mode:

```
python -m pip install -e .
```

This command is used to install a Python package in "editable" or "development" mode. The -e flag stands for "editable".

Make sure you run this command in the directory where the Python package's *pyproject.toml* file is located. This file contains information about the package and is necessary for the installation process.

After running this command, you should see the package installed in editable mode, and any changes you make to the code will take effect immediately without the need for reinstalling.

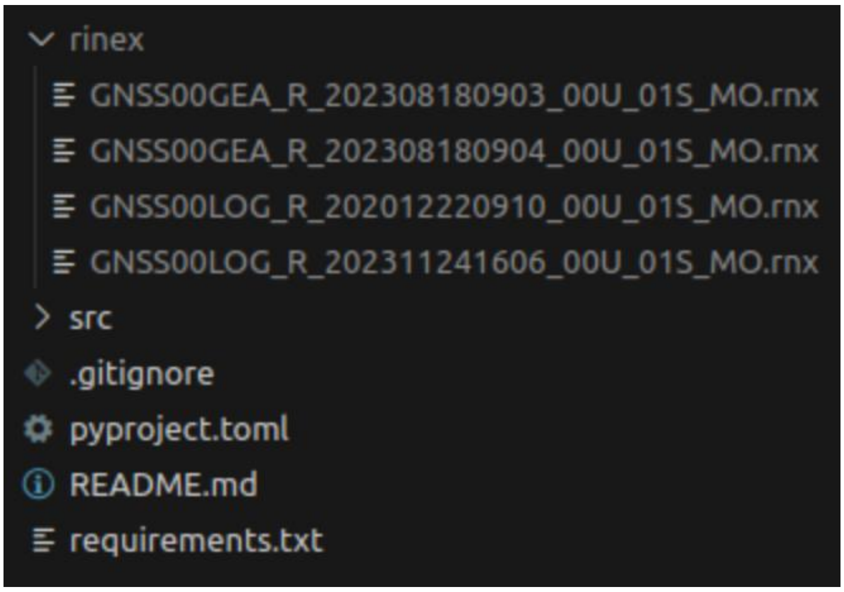
If you want to make contributions to the code base, install recommended dependencies to ensure Code Style Guidelines:

```
python -m pip install -r requirements.txt
```

6. Output

6.1 RINEX 3 Output:

The converted RINEX 3 files will be stored in a specified output directory. The output directory, named ‘rinex’, will be generated in the same directory where you execute the ‘raw_android_to_rinex’ command.



```
rinex
  GNSS00GEA_R_202308180903_00U_01S_MO.rnx
  GNSS00GEA_R_202308180904_00U_01S_MO.rnx
  GNSS00LOG_R_202012220910_00U_01S_MO.rnx
  GNSS00LOG_R_202311241606_00U_01S_MO.rnx
src
.gitignore
pyproject.toml
README.md
requirements.txt
```

Figure 5. *Example of generated RINEX files. In this instance, the tool has been executed in the same directory as the installation (where the downloaded file is extracted). The tool creates a folder named 'rinex' and accumulates all the outputs within it.*

6.2 File Naming Convention:

The RINEX file name convention generally includes information such as the station or site identifier, the start time of the data collection, and other relevant details. Here's a breakdown of the main components of a RINEX 3.05 file name that are specific for this tool. You can find the rest of them in Table A1 of RINEX 3.0.5 document [1]:

1. Station Identifier: This is a unique identifier for the GPS station or site where the data was collected. It could be a combination of letters and numbers that uniquely identifies the specific location. In the output of this tool, we will find two different Station Identifiers:
   a. GNSS00GEA: If raw data was logged using GEA.
   b. GNSS00LOG: If raw data was logged using GNSSLogger.
2. Session Start Time: The date and time at which the data collection session started. This is usually represented in the format YYYYMMDDHHMM, where YYYY is the four-digit year, MM is the two-digit month, DD is the two-digit day, HH is the two-digit hour (in UTC), and MM is the two-digit minute.

Thank you for choosing the Android GNSS Raw Data to RINEX 3 Converter!

Appendix A: Raw To RINEX Data

A sequence of computations is required to derive the Pseudorange, Carrier Phase, Doppler measurements, and temporal values. As elucidated by the GSA [2, 3], given the unavailability of these measurements directly from the Android API, their determination requires the following methodology [2, 3]:

GPS Time Generation: In context of Android 7 or higher, the direct provision of GNSS time is unavailable. However, an internal hardware clock and bias towards the true GPS time (expressed in nanoseconds) are offered, provided that the receiver has gauged the GNSS reference time. In instances where the receiver ascertains the GPS time, the computation can be conducted as follows:

$$GnssTime = TimeNanos - (FullBiasNanos + BiasNanos)[ns] \tag{1}$$

where $TimeNanos$ is the value of the GNSS receiver's internal hardware clock, expressed in nanoseconds; $FullBiasNanos$ signifies the bias between the receiver's clock and GPS time in nanoseconds; and $BiasNanos$ is the sub-nanosecond bias of the clock. If the receiver has inferred time via a non-GPS constellation, the calculated GPS time can be derived using the following formula:

$$GpsTime = TimeNanos - (FullBiasNanos + BiasNanos) - InterSystemsBias \tag{2}$$

where $InterSystemsBias$ denotes the disparity between the GPS and GNSS times used in the time estimation process. For instance, if the Galileo System Time is employed for time estimation, $InterSystemsBias$ is represented by the GPS to Galileo time offset (GGTO).

Pseudorange Generation: The Android system does not provide direct pseudoranges but offers all the essential parameters for their computation. The formulation of pseudoranges hinges on temporal disparity, specifically the time lapse between the instances of reception (measurement time) and transmission.

$$\rho = \frac{(t_{Rx} - t_{Tx})}{1E9} \cdot c \tag{3}$$

where t_{Tx} represents the received GNSS satellite time at the measurement juncture or the emission time, which corresponds to the GNSS reference time at which the signal is dispatched. t_{Rx} denotes the measurement time or received time, and c symbolizes the velocity of light in vacuum. t_{Tx} is furnished by the Android system and is structured as

$$t_{Tx} = ReceivedSvTimeNanos[ns] \tag{4}$$

where $ReceivedSvTimeNanos$ denotes the emission time in ns.

The acceptable scope of t_{Rx} can be reconstructed in the form:

$$t_{Rx_{GNSS}} = GnssTime + TimeOffsetNanos \tag{5}$$

Here, $TimeOffsetNanos$ signifies the time offset at which the measurement was captured, denoted in nanoseconds. This value specifies the measurement timing accuracy.

Carrier Phase Measurements: Android supplies carrier phase measurements as $AccumulatedDeltaRangeMeters$ (ADRM) represented in meters.

Table 1. *AccumulatedDeltaRangeState* [2, 3].

ADR State	Value	Description
UNKNOWN	0	The state is invalid or unknown.
VALID	1	The state is valid.
RESET	2	A reset is detected.
CYCLE_SLIP	4	A cycle slip is detected.
HALF_CYCLE_RESOLVED	8	Half cycle ambiguity is resolved at time t.
HALF_CYCLE_REPORTED	16	Half cycle ambiguity is reported at time t.

These measurements are inherently ambiguous and lack temporal information, indicating that the receiver can only quantify the cycles that occur between epochs. In the event of a cycle slip, this count was forfeited. The veracity of the carrier measurements is indicated through the $AccumulatedDeltaRangeState$ (ADRS) field, which offers several flags detailed in **Table 1**. For the computational process, only valid measurements in this category were considered.

Doppler Measurements: The Doppler shift arising from satellite motion can be deduced from the $PseudorangeRateMetersPerSecond$ parameter, provided that the pseudorange rate registered at the timestamp or received time is in meters per second (m/s). Notably, this value, presented in Hertz (Hz), does not encompass adjustments for receiver and satellite clock frequency discrepancies, making it an “uncorrected” value. A positive “uncorrected” value signifies the satellite's motion away from the receiver. The connection between the “uncorrected”, “pseudorange rate” and the “doppler shift” is articulated through the equation:

$$doppler = -\frac{PseudorangeRateMetersPerSecond}{k} \tag{6}$$

Here, k is a constant contingent on the centre frequency of the signal, for instance, f_c for L1 at 1575.42e6 Hz and the speed of light c . Thus, k is represented by c/f_c, which denotes the wavelength.

References

[1] Romero, I. (2020). RINEX The Receiver Independent Exchange Format Version 3.05. IGS/RTCM RINEX WG Chair ESA/ESOC/Navigation Support Office. Retrieved from https://files.igs.org/pub/data/format/rinex305.pdf

[2] European GNSS Agency, Using GNSS raw measurements on Android devices - White paper, Publications Office, 2017. https://data.europa.eu/doi/10.2878/449581

[3] Raw GNSS Measurements. (2023). Android Developers. https://developer.android.com/develop/sensors-and-location/sensors/gnss Accessed 23 Nov 2023