Point One Standard Dev Kit
User Manual

V1.4
2/1/2022

Table of contents

1. Introduction

This document provides technical guidance for the Point One Navigation Standard Dev Kit. The Point One Standard Dev Kit allows the user to collect precise positioning data using Point One FusionEngine API for various applications. It is available in a small miniPCIe form factor using Quectel's LG69T GNSS receiver module.

During operation, the device will produce a 10 Hz standalone or RTK navigation solution. Results can be output using the Point One FusionEngine (https://github.com/PointOneNav/fusion-engine-client) and/or NMEA-0813 protocols.

For RTK operation, corrections must be supplied to the device using RTCMv3 MSM4-7 messages. Corrections may be passed directly to the device via either serial UART. Corrections can be provided by a local base station, Point One's Polaris corrections network, or an alternative NTRIP server using either the Point One Navigation desktop application or the included Python `p1_runner` command-line application.

Point One Navigation Desktop application is freely available to interface with the Standard Dev Kit for configuration, data collection, and evaluation. The Desktop application is available for Linux, Windows, or Mac OS.

Similar to the Desktop application, the Python `p1_runner` application may be used on the command line to connect to the device over serial and performs the following:

- Connect to a Polaris or NTRIP server and provide incoming RTCM corrections data to the device over serial
  - NMEA GPGGA position messages automatically will be forwarded to the NTRIP server to associate the device with a corrections data stream
- Extract and display position solutions
- Relay positioning data on a TCP port for use with user applications
- Log FusionEngine, RTCM, NMEA data produced by the device for post-processing and diagnostic purposes

In addition, the provided Python `config_tool` command-line application may be used for configuring device settings.

This document provides information so that a user will be be able to:

- Connect the board to the host computer and GNSS antenna
- Configure device parameters and communication
- Install the Standard Dev Kit in a vehicle (if applicable)
- Run the device using the desktop or command-line applications and collect data
- Upgrade device firmware

2. Requirements

- Hardware
  - Point One Standard Dev Kit
  - L1/L5 GNSS Antenna
  - USB-to-miniPCIe Carrier board or miniPCIe interface to host computer
- Software
  - LG69T AM, AP, or AH release package containing the latest firmware and tools
  - Windows, Linux (kernel v5.9 or later), or Mac OS
  - Point One Navigation Desktop application
  - Latest firmware version available at https://pointonenav.com/docs/#standard-dev-kit
- Connectivity
  - The Point One Standard Dev Kit should be connected to a host computer with internet connectivity to receive GNSS corrections for RTK operation

NOTE: DRIVERS ARE REQUIRED FOR WINDOWS AND MACOS

When using Windows, install the CP210x Windows Drivers v6.7.6 or later from https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers. The default universal driver included with Windows can cause significant unexpected data loss on some machines.

When using Mac, install the CP210x VCP Mac OSX Driver v6.0.0 or later from https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers.

When using Linux, use the CP210x driver included with Linux kernel version 5.9 or later (included in Ubuntu 21.04 and later). Before 5.9, the driver had an issue that can cause unexpected data loss on some machines.

The Poinit One Standard Dev Kit offers two different types of Firmware available for use: AM and AP.

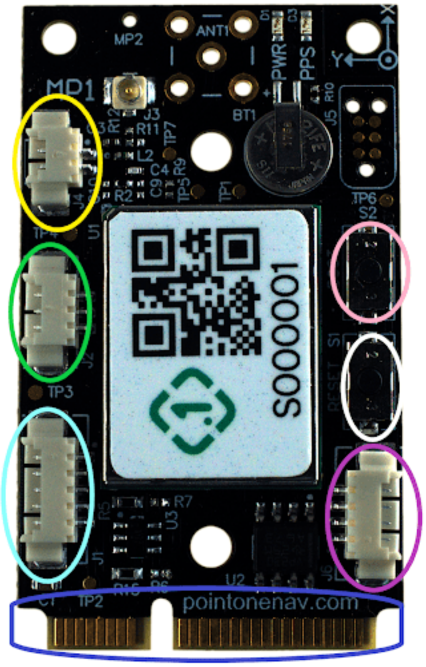
The AM firmware uses the on board GNSS receiver and supporting machine to stream RTK corrections and provide precise location.

The AP firmware integrates the GNSS receiver with the onboard IMU and optional wheel speed/tick sensor data in order to provide Dead Reckoning abilities in tough GNSS environments where it is difficult to provide an accurate position with GNSS alone. This firmware is only compatible with LG69T-AP modules.

By default, boards are flashed with the latest -AM firmware. See Section 7.1 to update firmware using the Point One Desktop Application to the latest version.

3. Point One Standard Dev Kit Interfaces

3.1 Hardware Interface



3.1.1 Point One Standard Dev Kit Connectors

🟦 J1: 5-pin connector (PPS, wheel tick, wheel direction)
🟩 J2: 3-pin connector (CAN bus data)
🟨 J4: 2-pin connector (battery backup)
🟪 J6: 4-pin connector (GPIO, 3.3Vout)
🟦 MiniPCIe
⬜ S1: Reset button
🟪 S2: Boot button

Port	Input	Outputs (default)
miniPCIe	• Point One FusionEngine control messages • NMEA control messages. • RTCM RTK corrections data (1005/6, MSM4-7) • Power	• Point One FusionEngine messages • NMEA-0183 solution messages
2-pin Connector [J4] (WM15251)	• Battery Backup (3.3V) *Note: this is not required if the soldered down battery is present.*	
3-pin Connector [J2] (WM15255)	• CAN bus data	
4-pin Connector [J6] (WM15259)	• GPIOA • GPIOB	• GPIOA • GPIOB • 3.3V
5-pin Connector [J1] (WM15263)	• Wheel Tick (0-5.5V tolerant) • Wheel Direction (0-5.5V tolerant)	• PPS

3.1.2 Connector Pinouts

[J1] 5-pin connector:

- Pin 1: Wheel Tick (0-5.5V tolerant)
- Pin 2: PPS Out
- Pin 3: Wheel Direction (0-5.5V tolerant)
- Pin 4: Open
- Pin 5: GND

[J2] 3-pin connector:

- Pin 1: CAN_L
- Pin 2: CAN_H
- Pin 3: GND

[J4] 2-pin connector:

- Pin 1:GND
- Pin 2: BAT (3.3V)

[J6] 4-pin connector:

- Pin 1: GPIO_A
- Pin 2: GPIO_B
- Pin 3: 3.3VOUT
- Pin 4: GND

3.2 Communications Interface

The Point One Standard Dev kit supports Point One FusionEngine messages and NMEA-0183 messages, over USB on the miniPCIe connector. There are two UARTs available on the LG69T device.

By default, serial UART1 is configured to output NMEA-0183 messages at 460800 baud. UART2 is configured to output an interleaved stream of FusionEngine, NMEA-0183, and RTCM messages. Output configuration can be changed using FusionEngine configuration messages (see 6.2.4 Enabling/Disabling Output Message Types). Both serial ports are configured for 8 bits, no parity, 1 stop bit (8N1).

During operation, the device will produce a 10 Hz position solution after initialization.

Note: In Windows, UART1 is named "Standard COM Port" and UART2 is named "Enhanced COM Port". In Linux, UARTs 1 and 2 typically appear as /dev/ttyUSB1 and /dev/ttyUSB0 respectively.

Note: We highly encourage the use of the Point One FusionEngine binary protocol over legacy NMEA-0813 messages. The FusionEngine protocol includes more detailed information than is available in standard NMEA messages.

The device can be operated with either the Point One Desktop Application, or the `p1_runner` command-line application included with the release. Both applications include a built-in NTRIP client, support for Point One's Polaris corrections network, and data logging support for diagnostics.

The device supports the following input and output message types:

Data	Mode
RTCM RTK Corrections (1005/1006, MSM4-7)	Input
RTCM Ephemeris (1019, 1020, 1042, 1045)	Input
FusionEngine Solution Messages	Output
FusionEngine Control Messages	Input/Output
FusionEngine Diagnostic Messages	Output
NMEA-0183 Solution Data	Output
NMEA Control Messages (Quectel Proprietary)	Input
RTCM Diagnostic Data	Output
RTCM RTK Corrections (mirrored)	Output

When messages from multiple protocols are enabled, they will be interleaved in a single data stream. Note that some NMEA clients may not support mixing non-ASCII data.

If desired, the output configuration can be changed using FusionEngine SetMessageOutputRate messages.

Corrections data and control messages may be sent to either UART, however input messages for a single protocol may not be sent to both UARTs simultaneously (for instance RTK corrections may only be sent to one port at a time).

The UARTs are configured to operate at 460800 baud by default. The UART baud rate may be configured by issuing a FusionEngine SetConfig message using the `config_tool` application. See 6.2.6 Changing The Serial Baud Rate.

3.3 Supported Message Types

The device natively supports the Point One FusionEngine protocol for solution output and command/control, as well as the NMEA-0183 standard for solution output.

The following table lists the various messages supported for each protocol, along with their default configuration for UART1 and UART2. See 6.2.4 Enabling/Disabling Output Message Types for details on changing the message configuration on each UART.

Message	**Mode**	**UART1**	**UART2**
Point One FusionEngine Protocol https://github.com/PointOneNav/fusion-engine-client			
Navigation Solution			
Pose (10000)	Output	Off	100 ms
GNSSInfo (10001)	Output	Off	100 ms
GNSSSatellite (10002)	Output	Off	500 ms
PoseAux (10003)	Output	Off	Off
CalibrationStatus (10004)	Output	Off	10 sec (*)
Device Control			
CommandResponse (13000)	Output	**	**
MessageRequest (13001)	Input		
ResetRequest (13002)	Input		
EventNotification (13004)	Output	***	On Change
ShutdownRequest (13005)	Input		
FaultControl (13006)	Input		
Configuration			
VersionInfo (13003)	Output	**	30 sec
SetConfig (13100)	Input		
GetConfig (13101)	Input		
SaveConfig (13102)	Input		

ConfigResponse (13103)	Output	**	**
GetMessageOutputRate (13221)	Input		
SetMessageOutputRate (13220)	Input		
MessageOutputRateResponse (13222)	Output	**	**
Sensor Measurements			
IMUMeasurement (11000)	Output	Off	On Change (*)
Vehicle Data Measurements			
WheelSpeedMeasurement (11101)	Input/Output	Off	On Change (*)
VehicleSpeedMeasurement (11102)	Input/Output	Off	On Change (*)
WheelTickMeasurement (11103)	Input	Off	On Change (*)
VehicleTickMeasurement (11104)	Input/Output	Off	On Change (*)
ROS Compatibility			
ROSPose (12000)	Output	Off	Off
ROSGPSFix (12010)	Output	Off	Off
ROSIMU (12011)	Output	Off	Off
NMEA-0183			
GGA	Output	100 ms	100 ms
GSV	Output	100 ms	100 ms
GSA	Output	100 ms	100 ms
GLL	Output	100 ms	100 ms
RMC	Output	100 ms	100 ms
VTG	Output	100 ms	100 ms
PQTMGNSS	Output	On Boot	On Boot
PQTMVER	Output	On Boot	On Boot
PQTMVERNO	Input/Output	Off	Off
P1CALSTATUS	Output	10 sec (*)	Off

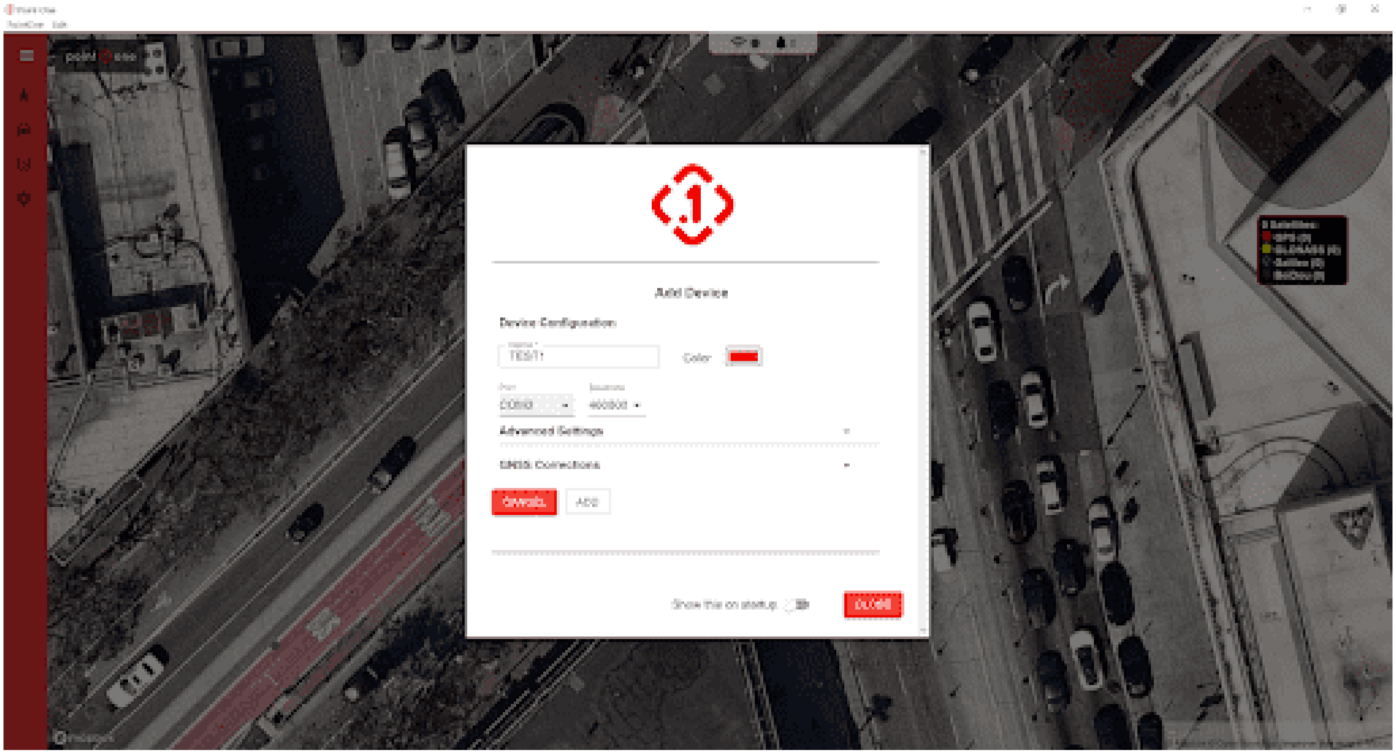
P1MSG	Output	On Change	On Change

** AP firmware only.*
*** Responses to commands will be sent on the interface the command was received, regardless of configuration.*
**** By default, FusionEngine EventNotificationMessages are only sent to UART1 in case of a fatal error.*

The complete list of FusionEngine messages and message format details can be found at http://pointonenav.com/files/fusion-engine-message-spec. Reference code is available for C++ and Python: https://github.com/PointOneNav/fusion-engine-client.

4. Quick Start Guide

1. Download and Install Point One Desktop Application, and USB drivers:
    a. Mac and Windows versions available at https://pointonenav.com/docs/#standard-dev-kit.
2. Plug in the Standard Dev Kit into your computer, connect a GNSS antenna (L1//L5) to the device, and launch the Desktop application.



Add Device

Device Configuration

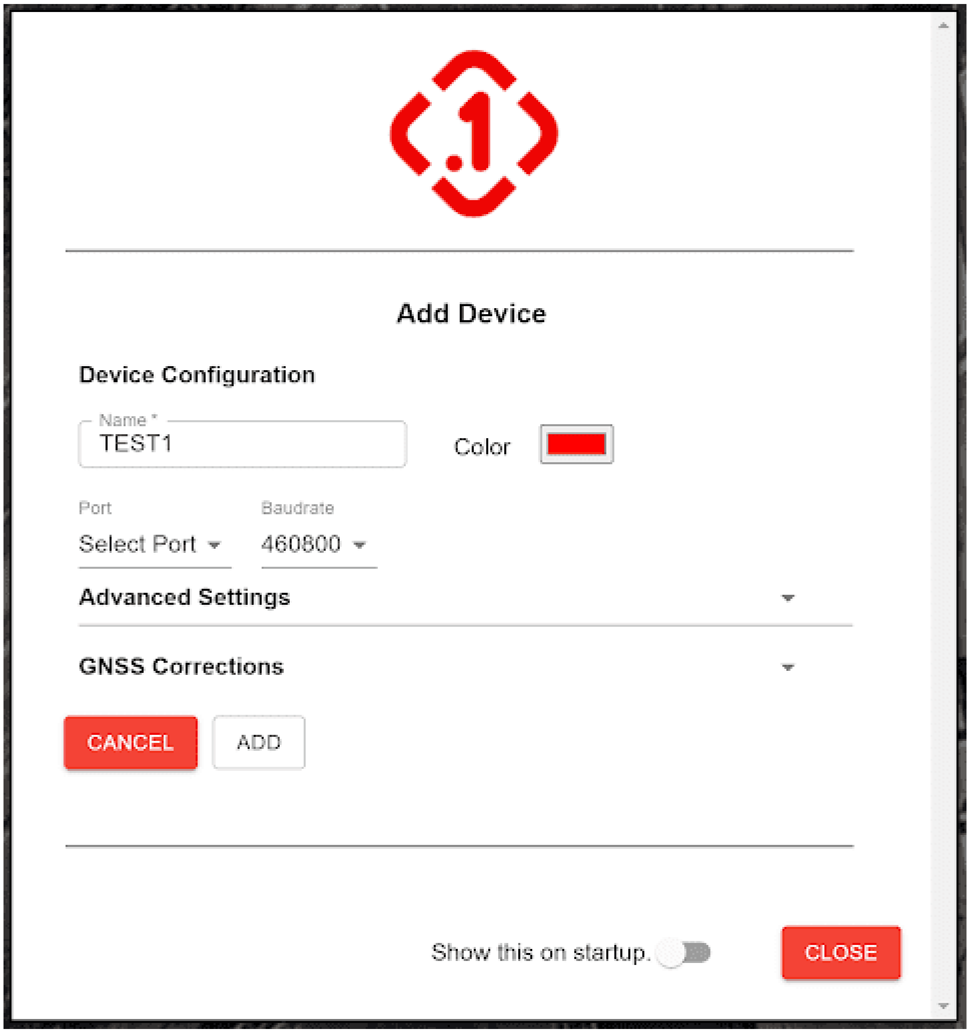
Name * TEST1

Color

Port COM3

Baudrate 460800

3. Add Device, with any name desired, and select the serial port associated with your Dev Kit

a. 

Add Device

Device Configuration

Name * TEST1

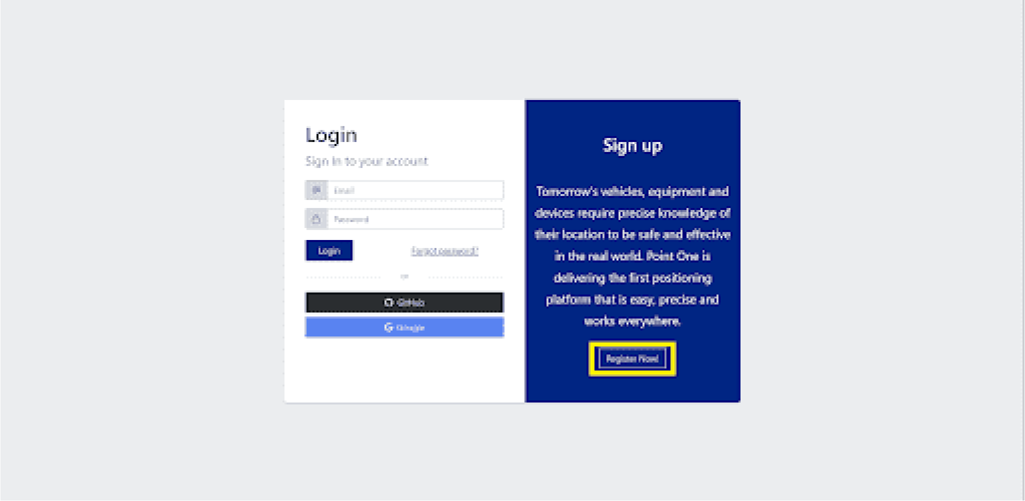
Color 🟥

Port: Select Port

Baudrate: 460800

4. Expand GNSS Corrections menu, and enter Polaris or NTRIP credentials, or “Sign Up” in order to get corrections. To sign up for Polaris Credentials, visit https://app.pointonenav.com/login and complete the following steps:

i. Register via Menu below



Login

Sign in to your account

Email

Password

Login

Forgot password?

or

GitHub

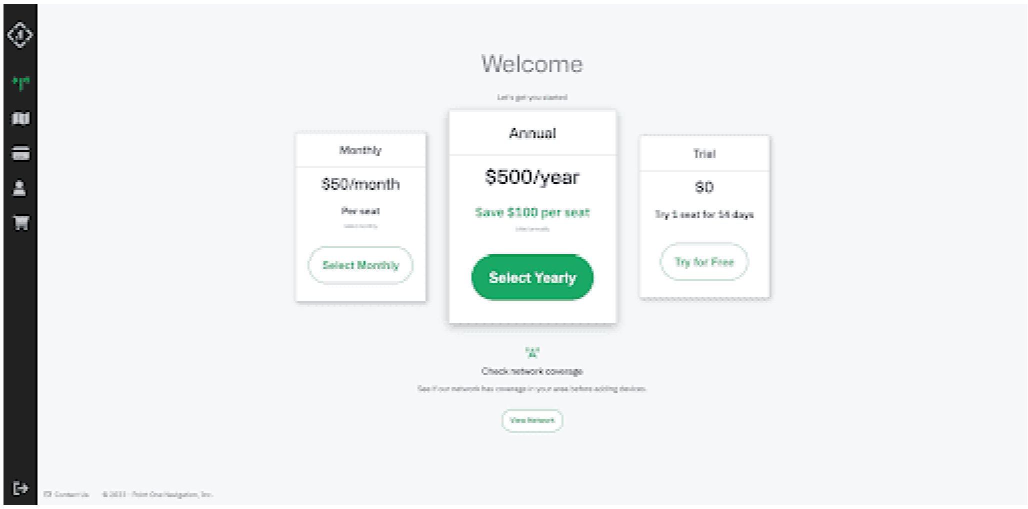
Google

Sign up

Tomorrow's vehicles, equipment and devices require precise knowledge of their location to be safe and effective in the real world. Point One is delivering the first positioning platform that is easy, precise and works everywhere.

Register Now!

ii. Verify email, and log in, Welcome page will offer options below, in order to get corrections “seat”:



Welcome

Let's get you started

Monthly	Annual	Trial
$50/month	$500/year	$0
Per seat	Save $100 per seat	Try 1 seat for 14 days
Select Monthly	Select Yearly	Try for Free

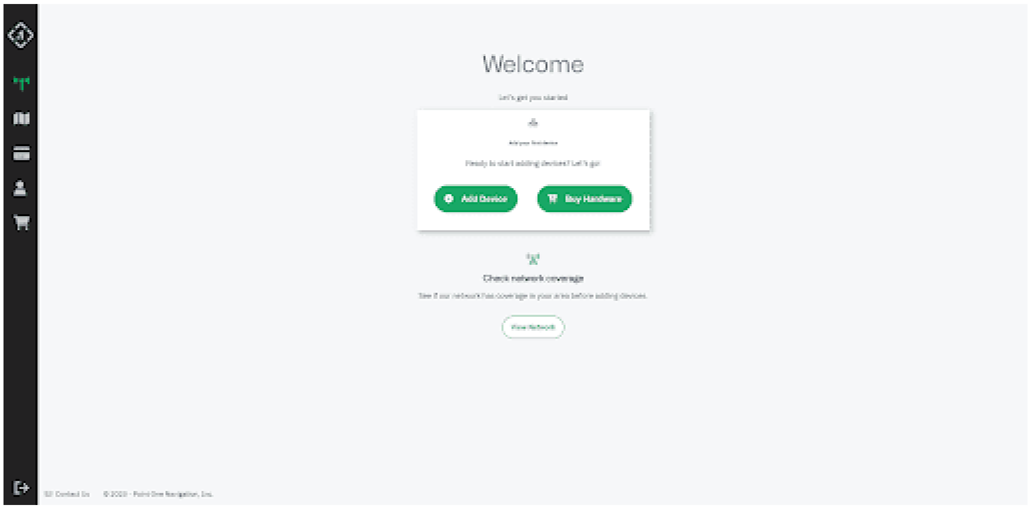
Check network coverage

See if our network has coverage in your area before adding devices.

View Network

Contact Us © 2021 · Point One Navigation, Inc.

iii. After acquiring seat, navigate to “Devices” on the left menu, and select “Add Device”:



Welcome

Let's get you started

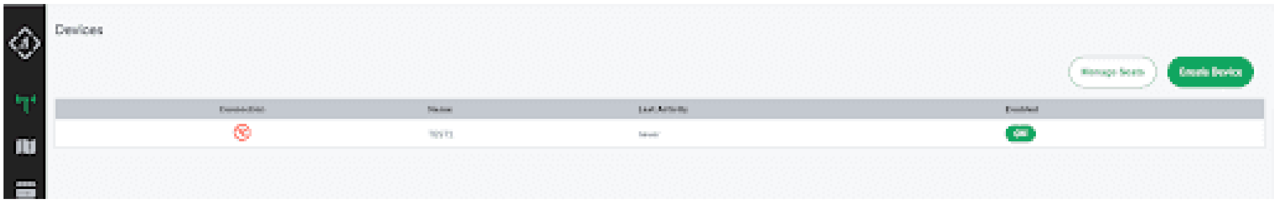
Add your first device

Ready to start adding devices? Let's go!

Check network coverage

See if our network has coverage in your area before adding devices.

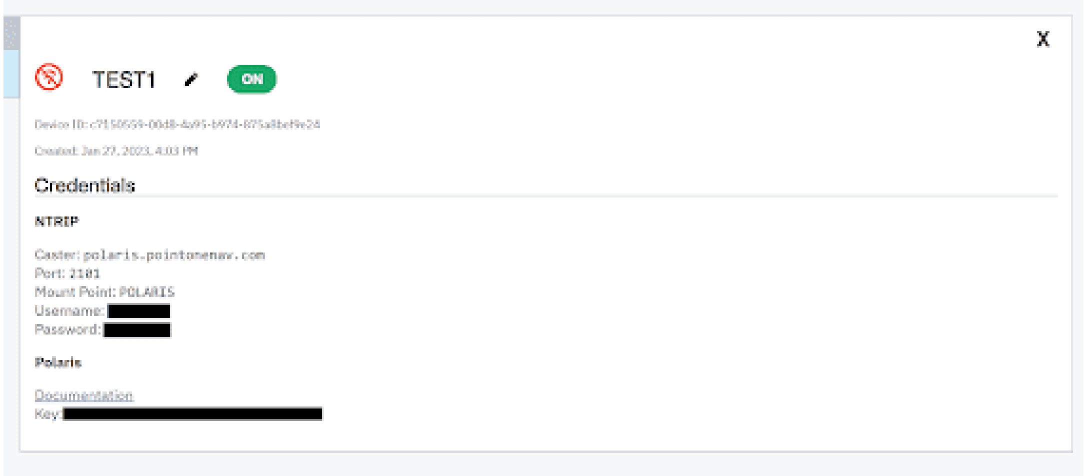
iv. Add new device with any unique device name (we recommend using the same name that you chose in step 3)



Devices

Connection	Name	Last Activity	Enabled
⊘	TEST1	Never	ON

v. Click device to get NTRIP and Polaris details:



TEST1 ON

Device ID: c7150559-00d8-4a95-b974-875a8bef9e24

Created: Jan 27, 2023, 4:03 PM

Credentials

NTRIP

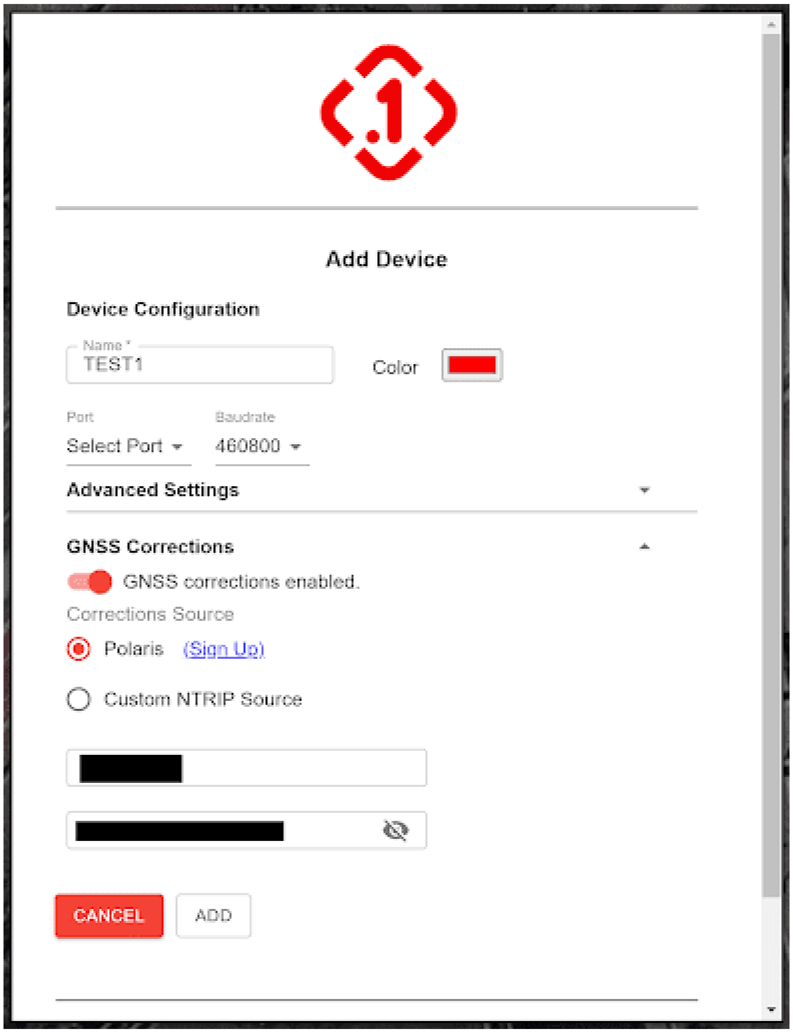
Caster: polaris.pointonenav.com
Port: 2101
Mount Point: POLARIS
Username: [redacted]
Password: [redacted]

Polaris

Documentation
Key: [redacted]

vi. Congratulations, you now have RTK corrections ready for use.

b. Back in the Desktop Application, input credentials from Polaris into the Add Device wizard, and click “ADD”.



Add Device

Device Configuration

Name * TEST1

Color 🟥

Port: Select Port

Baudrate: 460800

Advanced Settings

GNSS Corrections

GNSS corrections enabled.

Corrections Source

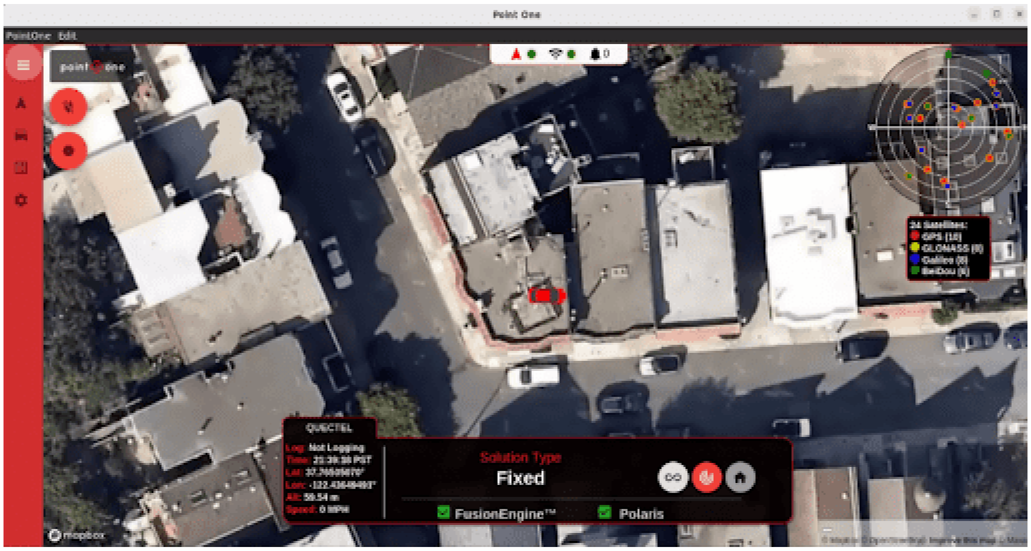
● Polaris (Sign Up)

○ Custom NTRIP Source

[redacted]

[redacted]

5. For best results, the GNSS antenna must be outside in an open sky environment, away from trees or large buildings.
6. Click Connect Device on the top left corner to see RTK-corrected position, with an RTK Fix Status (<10cm accuracy)
    a. Initial GNSS signal acquisition may take a minute while the receiver acquires satellites and constellation information from a cold start.



24 Satellites:
🟥 GPS (10)
🟨 GLONASS (8)
🟦 Galileo (8)
🟩 BeiDou (6)

Log: Not Logging
Time: 21:39:38 PST
Lat: 37.76605670°
Lon: -122.43648493°
Alt: 59.54 m
Speed: 0 MPH

Solution Type
Fixed

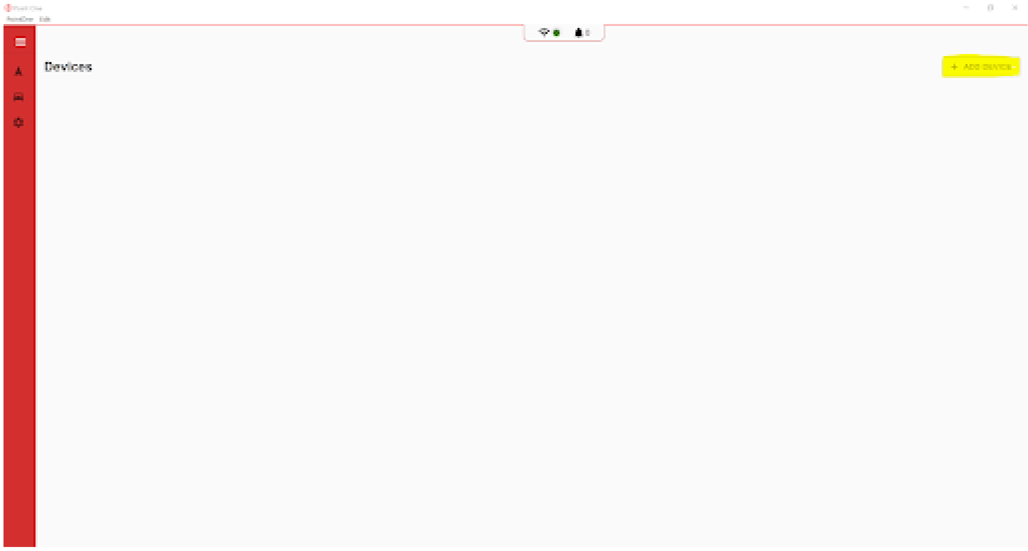
☑ FusionEngine™ ☑ Polaris

5. Desktop Application Configuration/Use

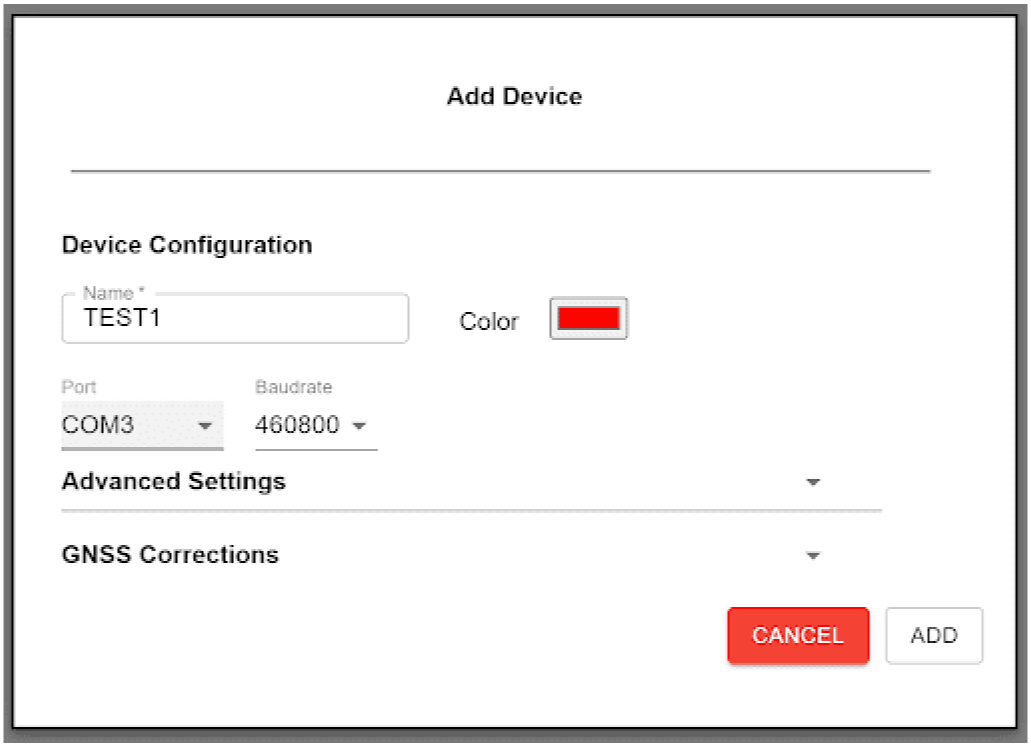
Note: If you have followed the quick start guide above, you can skip this section.

5.1 Device Setup

1. Navigate through the Desktop Application to Devices (car symbol in the menu)
2. Connect the device to your computer and click “Add Device” on the top right corner



3. Input Device Name, select color, Select port, and Baud rate to 460800 before clicking “Add”


Add Device

Device Configuration

Name *: TEST1

Color: 🟥

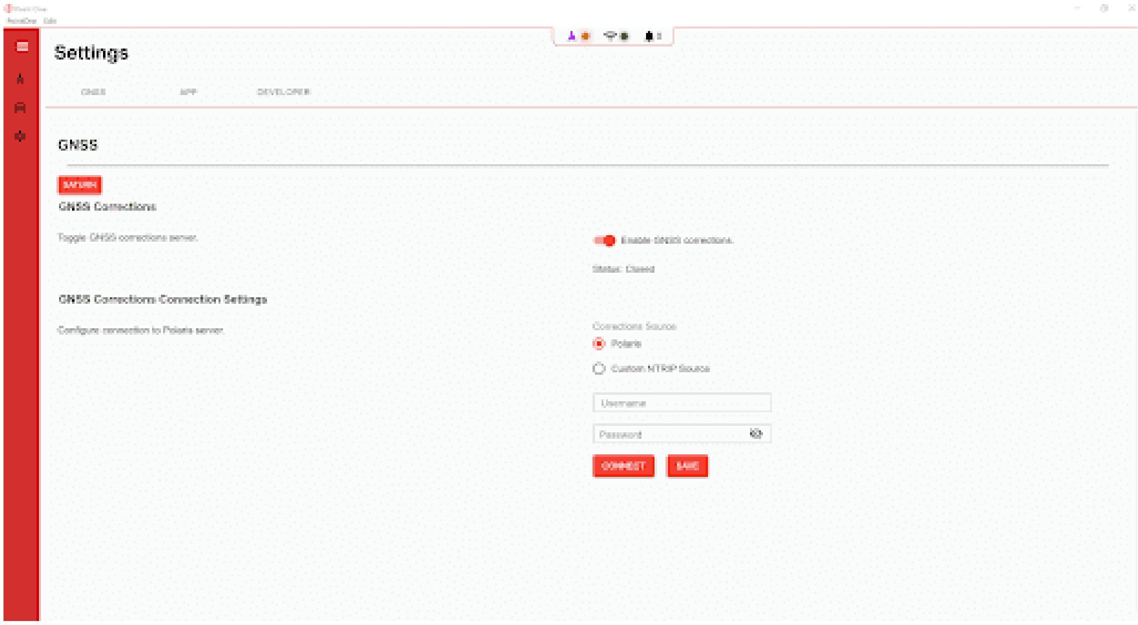
Port: COM3

Baudrate: 460800

5.2 GNSS Corrections Settings

In order to achieve <10 cm accuracy with the Point One Dev Kit, it is necessary to have RTK corrections enabled on the application. Point One offers RTK corrections via our website (https://app.pointonenav.com/start). In order to input Polaris credentials in the application, follow steps below:

1. Navigate through the Desktop Application, Settings>GNSS, and enable GNSS corrections



GNSS

GNSS Corrections

Toggle GNSS corrections server.

Enable GNSS corrections.

Status: Closed

GNSS Corrections Connection Settings

Configure connection to Polaris server.

Corrections Source

● Polaris

○ Custom NTRIP Source

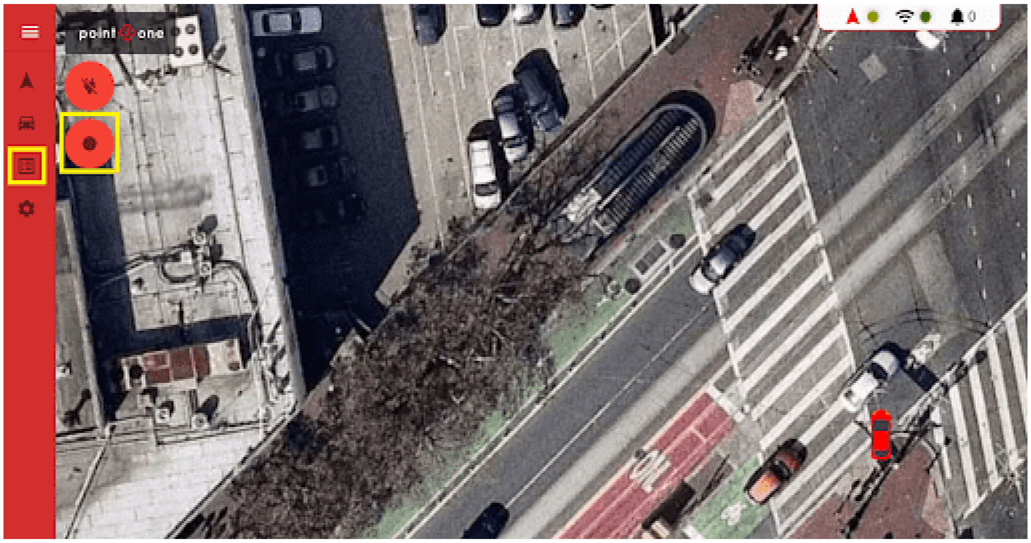
Username

Password

2. Enter Polaris Credentials received from Point One Website on Username/Password prompt, and “Save”

5.3 Navigating/Logging

After connecting the device, you can record logs by clicking the “Record” button under the “Connect” button on the Map Menu. Select file paths for all collected logs, and manage them through the Log Data Menu on the left. This records raw, NMEA, and Fusion Engine data throughout the course of the log which can be used with Point One’s open source visualization tools, and can be provided to Point One for support and diagnostic purposes.


point one
0

6. Command-Line Python Applications

In addition to the Point One Desktop Application, the device can be operated and configured using the `p1_runner` and `config_tool` command-line applications included in the release package.

The client applications are written in Python 3. They include a pip `requirements.txt` file, which details the dependencies needed to run the application.

For Windows users, the release includes pre-compiled executables that may be used instead of installing and configuring Python.

We strongly encourage the use of a Python virtual environment for managing dependencies. The following steps create a virtual environment, activate it, and install the requirements in Linux or Mac OS:

```
$ cd p1_runner
$ python3 -m venv venv
$ source venv/bin/activate
$ pip install -I -r requirements.txt
```

Once the virtual environment is created, it can be entered again by sourcing the activate script for your operating system (i.e. activate for Linux/Mac, activate.bat for Windows).

If desired, it is also possible to install the requirements directly in the system Python installation:

```
$ pip3 install -r requirements.txt
```

6.1 p1_runner For AP And AM Firmware

6.1.1 Running With The Included p1_runner Python Application

1. Connect a GNSS antenna to the device.
2. Connect a USB cable from the device to your host computer.
3. If used, activate the Python virtual environment.
   Linux/Mac: `$ source venv/bin/activate`
   Windows: `> venv\Scripts\activate.bat`
4. Run the Python client application to connect the device
   Linux/Mac: `$ python3 bin/runner.py`
   Windows: `> python bin/runner.py`
   - By default, this will use the following serial devices:

	RTK Corrections / NMEA Output	Sensor Data / NMEA Output
Linux	/dev/ttyUSB0	/dev/ttyUSB1
Windows	COM1	COM2

  - If the serial devices enumerate using different names, use the `--device-port` and `--corrections-port` arguments to specify the correct ports.
  - Note: In Linux, you may need to add yourself to the `dialout` group to access the serial ports as a non-root user.

6.1.2 Receiving RTK Corrections

For precision applications, you must provide GNSS RTK corrections data. The Python client can be configured to connect to Point One's Polaris corrections service or to an NTRIP server to receive corrections and relay them to the device.

To enable Polaris corrections, use the `--polaris` argument, providing an NTRIP password assigned by Point One. In addition, you must specify an ID that uniquely identifies the device. The username may only contain letters, numbers, dashes, or underscores and can be at most 32 characters. For example:

```
$ python3 bin/runner.py --device-id my-device --polaris abcd1234
```

Important: There cannot be two concurrent connections to the Point One Polaris NTRIP service with the same username and password, doing so will lead to undefined behavior.

You can get a username and password following the instructions in Section 4 of this document or by signing up at https://app.pointonenav.com/register.

To use another NTRIP service, use the `--ntrip` argument, specifying URL, mountpoint, and optionally username and password. For example:

```
$ python3 bin/runner.py --ntrip
http://169.125.0.1:2101,my_mountpoint,my_username,my_password
```

6.1.3 Outputting Data Over TCP

If desired, the `p1_runner` application can be configured to output data from the device to a user application over a TCP or websocket connection. To enable, specify a TCP port as follows:

```
$ python3 bin/runner.py --tcp 12345
```

To use a websocket connection, specify the following:

```
$ python3 bin/runner.py --websocket 12345
```

User applications may then connect to port 12345 and receive the generated FusionEngine and NMEA messages.

6.1.4 Data Logging

The Python client application records all sensor measurements and generated output from the device into a log directory. By default, logs are stored in `~/logs` on Linux or `%HOME%\Documents\logs` (usually `My Documents\logs`) on Windows.

Logs are useful for post process analysis (using the FusionEngine open source tools available at https://github.com/PointOneNav/fusion-engine-client), and for post-run diagnostics that can be performed by the Point One support team.

6.2 config_tool For AP And AM Firmware

6.2.1 Running The Configuration Utility

The following sections describe various configuration parameters available to the user. These parameters may be set using the provided Python configuration utility. To use the configuration utility:

1. Configure your Python environment as described in section 6. Command Line Python Applications.
2. Connect a USB cable from the USB_UART connector to your host computer.
3. If used, activate the Python virtual environment.
   Linux: `$ source venv/bin/activate`
   Windows: `> venv\Scripts\activate.bat`
4. Run the Python configuration utility using the commands described in the sections below to write the values to the device. The configuration utility takes an action argument, followed by additional optional parameters:
   ```
     $ python3 bin/config_tool.py COMMAND [OPTIONS...]
   ```

For example, to enable FusionEngine PoseMessages and disable NMEA GGA messages on UART1 run the following:

```
  $ python3 bin/config_tool.py apply uart1_message_rate fusion_engine pose
on
  $ python3 bin/config_tool.py apply uart1_message_rate nmea gga off
  $ python3 bin/config_tool.py save
```

6.2.2 Saving Changes

Parameter changes take effect immediately after issuing an `apply` command:

```
      $ python3 bin/config_tool.py apply ...
```

Applied settings are not saved to persistent storage automatically, and will be reset after a power cycle. To save settings to persistent storage, issue a `save` command:

```
      $ python3 bin/config_tool.py save
```

6.2.3 Enabling The Software Watchdog (Reset On Crash)

The LG69T software includes a watchdog timer that will automatically restart the device if a crash or other fatal error is detected. **The watchdog timer is not enabled by default.**

To enable the watchdog timer, issue the following command:

```
$ python3 bin/config_tool.py apply watchdog_enabled true
```

6.2.4 Enabling Output Message Types And Changing The Output Rate

By default, UART1 and UART2 are configured to output a different set of FusionEngine and NMEA message types. See section 3.3 Supported Message Types for the complete list.

For each UART, you may change the enabled set of messages at any time using `config_tool.py` by setting sending appropriate `*_message_rate` command, specifying:

- The interface to be configured (UART1 or UART2)
- The protocol to be configured (FusionEngine, NMEA, or RTCM)
- The message desired type (GGA, PoseMessage, etc.)
- The desired status or message rate (off, on change, 500 ms interval, etc.)

For example, to enable FusionEngine Pose messages on UART1, issue the following command:

```
$ python3 bin/config_tool.py apply uart1_message_rate fusion_engine pose on
```

Similarly, to disable NMEA GSV messages on UART1, issue the following command:

```
$ python3 bin/config_tool.py apply uart1_message_rate nmea gsv off
```

Another example, to change the UART2 output from the default 10 Hz rate to 1 Hz, issue the following command:

```
$ python3 bin/config_tool.py apply uart2_message_rate all 1s
```

For a complete list of options and their values, run:

```
$ python3 bin/config_tool.py apply uart1_message_rate --help
```

6.2.5 Enabling Diagnostic Output

When requesting support for an issue, you must enable diagnostic output from the device, which will be captured along with any other messages you have configured. The diagnostic data is necessary when sending a log to Point One for assistance.

Note: When enabled, this feature will automatically enable all diagnostic message types, including RTCM data and FusionEngine PoseMessages, and will override any individual message settings set for those messages.

To enable diagnostic output on UART1 (or UART2, replacing 1 with 2 below), issue the following command:

```
$ python3 bin/config_tool.py apply uart1_output_diagnostics true
```

Alternatively, you can send a FusionEngine `SetConfig` message setting the parameter `UART1_OUTPUT_DIAGNOSTICS_MESSAGES` to true. See https://github.com/PointOneNav/fusion-engine-client for details.

6.2.6 Changing The Serial Baud Rate

To change the the UART1 serial baud rate from the default 460800 to 115200, issue the following command:

```
$ python3 bin/config_tool.py apply uart1_baud 115200
```

> Note: The `p1_runner.py` and `config_tool.py` tools both assume a baud rate of 460800 by default. If you specify a different baud rate, you must supply the `--baud-rate RATE` argument when calling the scripts.

6.2.7 Resetting the Device

The device supports multiple levels of reset. The included `config_tool` application may be used to request a reset at any time, or the user may request a reset by sending a FusionEngine `ResetRequest` message.

Reset Type	**Description**
`nav_engine`	Reset the navigation engine and position estimate. Do not reset ephemeris data or RTK corrections data, and do not cold start the Teseo GNSS receiver.
`cold`	Reset the position estimate, ephemeris data, and RTK corrections, and cold start the Teseo GNSS receiver
`config`	Reset user configuration parameters back to factory default values. Also performs a calibration reset if the device orientation/lever arm settings are not default.
`factory`	Reset all settings and state data back to factory defaults, and cold start the Teseo GNSS receiver.

For example, to request a cold start, run the following command:

```
$ python3 bin/config_tool.py reset cold
```

6.3 config_tool For AP

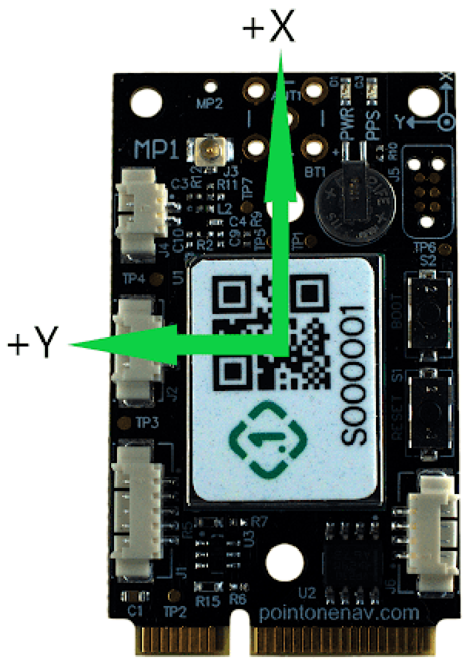
The AP Firmware makes use of the physical constraints of the vehicle, an IMU as well as the same high performance GNSS engine in the AM firmware. These additional constraints and sensors make it necessary to configure the orientation of the device (see 6.3.1 Device (IMU) Orientation) and the GNSS and device lever arms (see 6.3.2 Lever Arms), referenced to the center of the rear axle of the vehicle.

1. Configure your Python environment as described in section 6. Command Line Python Applications
2. Connect a USB cable from the USB_UART connector to your host computer.
3. If used, activate the Python virtual environment.
   Linux: `$ source venv/bin/activate`
   Windows: `> venv\Scripts\activate.bat`
4. Run the Python configuration utility using the commands described in the sections below to write the values to the device. The configuration utility takes an action argument, followed by additional optional parameters:
   ```
   $ python3 bin/config_tool.py COMMAND [OPTIONS...]
   ```

For example, to configure the GNSS antenna lever arm to [0.5, -0.3, 1.1] meters, the device lever arm to [0.2, 0.4, 0.5] meters and the device orientation facing the left side of the vehicle, run the following:

```
$ python3 bin/config_tool.py apply gnss 0.5 -0.3 1.1
$ python3 bin/config_tool.py apply device 0.2 0.4 0.5
$ python3 bin/config_tool.py apply orientation left
$ python3 bin/config_tool.py save
```

6.3.1 Device (IMU) Orientation



Both the Point One Standard Dev Kit and the GNSS antenna must be mounted **rigidly** to the vehicle. You must specify the rough orientation of the device relative to the vehicle, and then the device calibration procedure will estimate any differences between the device mounting angles and the vehicle axes.

We recommend installing the Dev Kit with the +Y axis towards the left side of the vehicle and the top of the circuit board facing upward. In this orientation, the IMU +x axis will align with the forward axis of the vehicle. This is the default orientation setting.

If you install it in a different orientation, make sure to specify the correct direction of the device +x and +z axes when configuring the device settings using the following command:

```
$ python3 bin/config_tool.py apply orientation x [z]
```

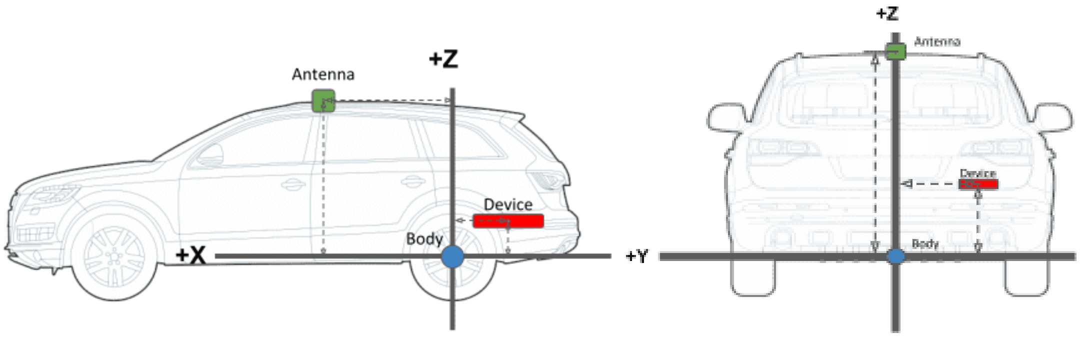
Supported values for the x and z parameters are:

- `forward` – Device axis aligned with vehicle forward direction of motion
- `backward` – Device axis aligned with vehicle backward direction of motion
- `left` – Device axis pointed towards the left side of the vehicle
- `right` – Device axis pointed towards the right side of the vehicle
- `up` – Device axis pointed upward
- `down` – Device axis pointed downward

For example, to configure the device with the +x axis facing the left side of the vehicle (PCIe connector facing to the right), run the following command:

```
    $ python3 bin/config_tool.py apply orientation left
```

6.3.2 Lever Arms



Once installed, you must measure the lever arms from the vehicle body frame to both the device and the GNSS antenna. For best performance, it is important to measure the lever arms as accurately as possible, ideally to better than 5cm of accuracy. The following lever arms may be specified:

- **GNSS antenna lever arm**: the location of the GNSS antenna, measured with respect to the body frame
- **Device lever arm**: the location of the device, measured with respect to the body frame
- **Output lever arm**: The desired output location, measured with respect to the body frame
  - *Recommended: set equal to the GNSS antenna lever arm to generate output at the location of the antenna.*

To set a lever arm, run the `queue` command, specifying the lever arm type -- `gnss`, `device`, `output` -- along with the X, Y, and Z, lever arm values in meters:

```
    $ python3 bin/config_tool.py apply gnss 0.5 -0.3 1.1
```

The vehicle body frame is a right-handed coordinate system centered at the middle of the rear axle of the vehicle and is oriented as follows:

+x - Toward the front of the vehicle
+y - Toward the left side of the vehicle
+z - Up

The lever arms are defined as the vector to the sensor from the vehicle body frame origin, resolved in the body frame. For instance, if the device is located behind and above the rear axle, toward the right-hand side of the vehicle (as in the diagram above), then the device lever arm will have a negative x component, a negative y component, and positive z component. Ideally these measurements will be accurate to better than 5 cm.

The lever arms should be measured at the nominal phase center of the antenna and center of the QR code label on the Point One Standard Dev Kit.

Note: If the Dev Kit is moved even slightly, the existing calibration will no longer be valid and the device must be recalibrated by issuing a calibration reset command.

6.3.3 Hardware Wheel Tick Configuration

To use an external wheel tick signal, you must configure the following parameters:

- The capture edge for the wheel tick signal (off, rising edge, or falling edge)
- The behavior of the direction signal (off, active high, or active low; optional)
- The scale factor to convert wheel ticks to meters, calculated from the tick encoder resolution and the vehicle's nominal wheel diameter:

  *Scale [meters / tick] = (Diameter [meters] / 2) * Resolution [radians / ticks]*

Warning: You must **not** enable hardware tick capture if a wheel tick signal is **not** available. Instead, set the wheel tick mode to OFF. Similarly, if a vehicle direction signal is not available, you must set the direction type to OFF to indicate that the signal is not connected.

Configuring the device to expect wheel ticks or direction when a voltage signal is not present will result in extremely poor dead reckoning performance.

For example:

```
$ python3 bin/config_tool.py apply hardware_tick_config
    --tick-mode rising_edge --tick-direction forward_active_high
    --wheel-ticks-to-m 0.05
$ python3 bin/config_tool.py apply hardware_tick_config
    --tick-mode falling_edge --tick-direction off
    --wheel-ticks-to-m 0.13
```

For a complete list of options and their values, run:

```
$ python3 bin/config_tool.py apply hardware_tick_config --help
```

Alternatively, you can send a FusionEngine `SetConfig` message containing a `HardwareTickConfig` payload. See https://github.com/PointOneNav/fusion-engine-client for details.

Note: The wheel diameter estimate will be automatically and continually refined as part of the calibration process.

6.3.4 Software Wheel Speed Configuration

Where available, the device can be provided wheel speed measurements through software by sending one of the following FusionEngine messages to either UART:

- `WheelSpeedMeasurement` - Individual speeds for 2 or more wheels
- `VehicleSpeedMeasurement` - A single along-track (forward/backward) speed for the vehicle

See https://github.com/PointOneNav/fusion-engine-client for a complete description of the messages and configuration details.

In order to use software wheel speeds, you must first configure the following parameters depending on the type of wheel data being provided:

- The wheel sensor type: differential wheel speeds (i.e., both wheels on the rear axle), or a single vehicle speed measurement
- The applied speed type, i.e., which wheels to use: front wheels, rear wheels, single vehicle speed
- The steering type (if using the steered wheels)
- The vehicle steering ratio (if using the steered wheels)
- The nominal wheel measurement interval

In addition, you must also specify the following vehicle properties:

- The wheelbase - the distance between front and rear wheels (if using front wheels)
- The front track width - the distance between the front wheels (if using front wheels)
- The rear track width - the distance between the rear wheels (if using rear wheels)

For best performance, the recommended configuration is differential rear wheel speeds.

You can specify the above parameters using `config_tool.py`. For a complete list of options and their values, run:

```
$ python3 bin/config_tool.py apply wheel_config --help
```

Alternatively, you can send FusionEngine `SetConfig` messages containing `VehicleDetailsConfig` and `WheelConfig` payloads. See https://github.com/PointOneNav/fusion-engine-client for details.

6.3.4.1 Example: Differential Rear Wheel Speeds (Unsteered)

```
$ python3 bin/config_tool.py apply wheel_config
    --wheel-sensor-type=wheel_speed
    --applied-speed-type=rear_wheels
    --wheel-update-interval=0.1
```

6.3.4.2 Example: Differential Front Wheel Speeds (Steered)

```
$ python3 bin/config_tool.py apply wheel_config
    --wheel-sensor-type=wheel_speed
    --applied-speed-type=front_wheels
    --wheel-update-interval=0.1
    --steering-type=front
    --steering-ratio=5.1
```

6.3.4.3 Example: Single Vehicle Speed Measurement

```
$ python3 bin/config_tool.py apply wheel_config
    --wheel-sensor-type=vehicle_speed
    --applied-speed-type=vehicle_body
    --wheel-update-interval=0.1
```

6.3.5 Software Wheel Tick Configuration

Similar to section 6.3.4 Software Wheel Speed Configuration, you can also provide wheel encoder tick count measurements, which measure change in angle as the vehicle's wheels rotate. To use wheel tick data, send one of the following FusionEngine messages to either UART:

- `WheelTickMeasurement` - Individual tick counts for 2 or more wheels
- `VehicleTickMeasurement` - A single along-track (forward/backward) tick count for the vehicle

See https://github.com/PointOneNav/fusion-engine-client for a complete description of the messages and configuration details.

In order to use software wheel ticks, you must configure the following parameters depending on the type of wheel data being provided:

- The wheel sensor type: differential (i.e., multiple wheels) or single vehicle tick counts
- The applied speed type, i.e., which wheels to use: front wheels, rear wheels, single vehicle speed
- The steering type (if applicable)
- The vehicle steering ratio (if applicable)
- The nominal wheel measurement interval
- The scale factor to convert wheel ticks to meters (if using ticks), calculated from the tick encoder resolution and the vehicle's wheel diameter:

  *Scale [meters / tick] = (Diameter [meters] / 2) * Resolution [radians / ticks]*
- The maximum tick value before the tick count rolls over
- Whether or not the tick value is signed or unsigned
- If the ticks increase as the vehicle drives in either direction, or if they decrease when the vehicle drives backward

In addition, you must also specify the following vehicle properties:

- The wheelbase - the distance between front and rear wheels (if using front wheels)
- The front track width - the distance between the front wheels (if using front wheels)
- The rear track width - the distance between the rear wheels (if using rear wheels)

For best performance, the recommended configuration is differential rear wheel ticks.

You can specify the above parameters using `config_tool.py`. For a complete list of options and their values, run:

```
$ python3 bin/config_tool.py apply wheel_config --help
```

Alternatively, you can send FusionEngine `SetConfig` messages containing `VehicleDetailsConfig` and `WheelConfig` payloads. See https://github.com/PointOneNav/fusion-engine-client for details.

6.3.5.1 Example: Differential Rear Wheel Ticks (Unsteered)

```
     $ python3 bin/config_tool.py apply wheel_config
         --wheel-sensor-type=ticks
         --applied-speed-type=rear_wheels
         --wheel-update-interval=0.1
         --meters-per-tick=0.05
         —-wheel-tick-max-value=255
         --wheel-ticks-signed=false
         --wheel-ticks-always-increase=true
```

6.4 Testing Dead Reckoning (Fault Control)

For testing purposes, the device supports a number of “fault controls” that can be enabled at any time to simulate specific behaviors. For testing dead reckoning performance, you can simulate a GNSS outage as follows:

```
     $ python3 bin/config_tool.py fault gnss off
```

It is highly recommended that you simulate GNSS outages through software control instead of disconnecting the antenna. Simulating an outage in software allows GNSS data to be captured in the log for diagnostic purposes and avoids unnecessary wear on the antenna connector.

7. P1DK Firmware Versions

The Point One Standard Dev Kit offers two different types of Firmware available for use: AM and AP.

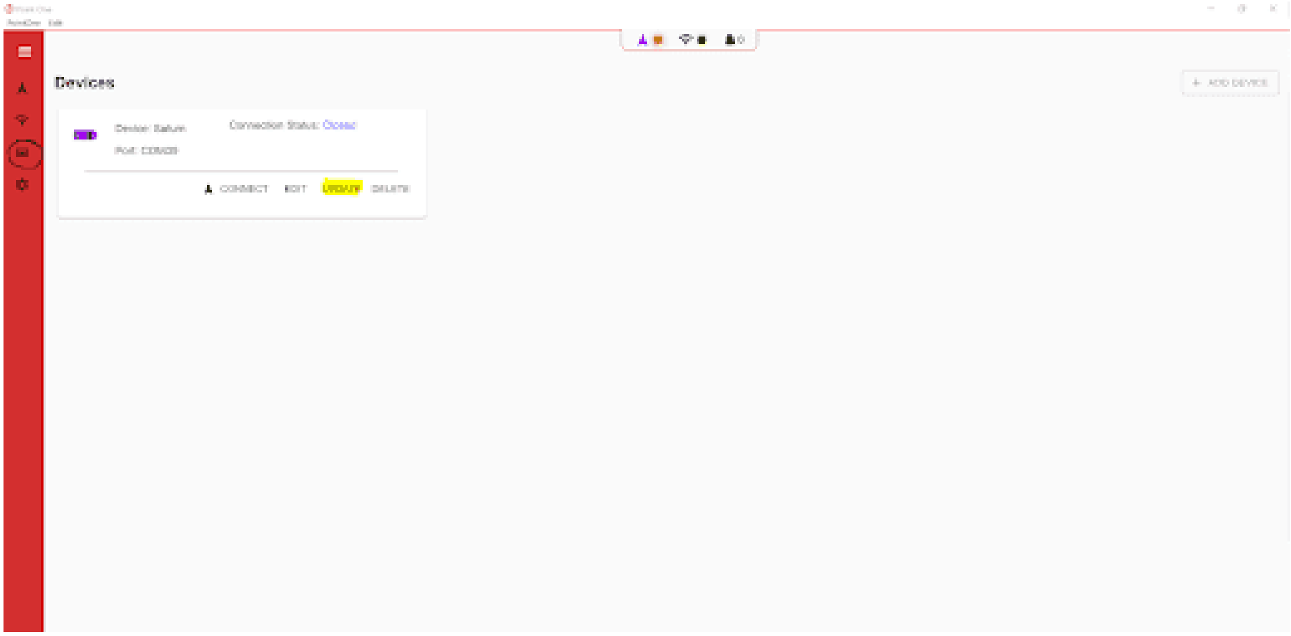
The AM firmware uses the on board GNSS receiver and supporting machine to stream RTK corrections and provide precise location.

The AP firmware integrates the GNSS receiver with the onboard IMU and optional wheel speed/tick sensor data in order to provide Dead Reckoning abilities in tough GNSS environments where it is difficult to provide an accurate position with GNSS alone. This firmware is only compatible with LG69T-AP modules.

7.1 Firmware Updates

Regardless of what firmware is preferred, you may use the Point One Desktop Application to upgrade firmware.

1. In Desktop Application, go to Devices Tab, and select Update



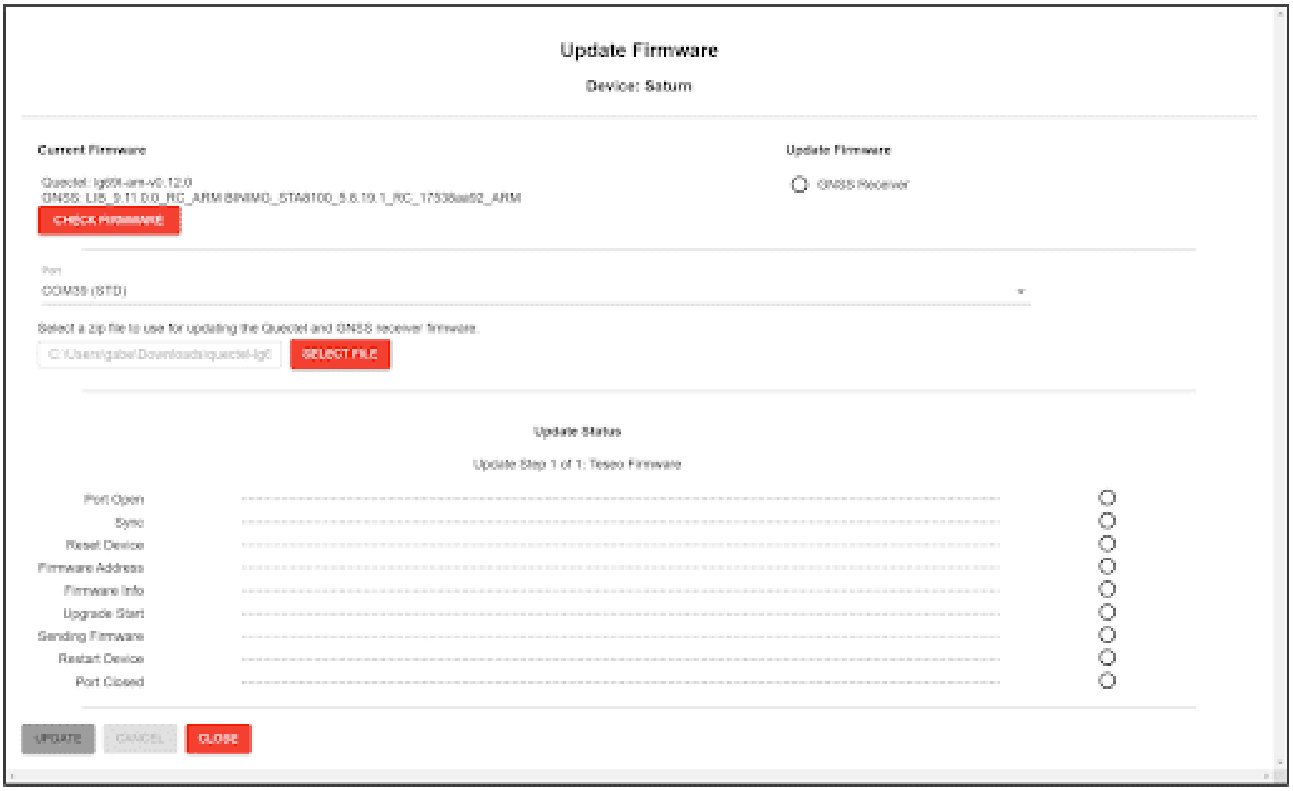
Devices

Device: Saturn

Port: COM20

Connection Status: Closed

2. Click “Select File”, and select the appropriate firmware version (AM or AP), named quectel-lg69t-aX.A.B.C.p1fw, and click Update



Update Firmware

Device: Saturn

Current Firmware

Quectel: lg69t-am-v0.12.0
GNSS: LIB_5.11.0.0_RC_ARM BINIMG_STA8100_5.8.19.1_RC_17538ae82_ARM

Update Firmware

○ GNSS Receiver

Port
COM38 (STD)

Select a zip file to use for updating the Quectel and GNSS receiver firmware.

C:\Users\gabe\Downloads\quectel-lg6

Update Status

Update Step 1 of 1: Teseo Firmware

Port Open ○
Sync ○
Reset Device ○
Firmware Address ○
Firmware Info ○
Upgrade Start ○
Sending Firmware ○
Restart Device ○
Port Closed ○

Follow prompts on screen on when to reboot, and Update should be completed in ~5 minutes.

8. NMEA Message Definitions

The following sections describe proprietary NMEA message formats supported by the device firmware.

8.1 NMEA $PQTM* Definition

The PQTM* NMEA messages are proprietary messages defined for use by Quectel. See Quectel NMEA message documentation for details.

8.2 NMEA $P1CALSTATUS Definition

This is a proprietary NMEA message defined by Point One and used to convey the current status of the device calibration process. It is defined as follows:

```
$P1CALSTATUS,<stage>,<state_verified>,<gyro_percent_complete>,<accel_percent_complete>,<mounting_angle_percent_complete>
```

where the fields are:

- `<stage>` - The current calibration stage:
    - 0 – Unknown/not started
    - 1 – Initial mounting angle convergence (performance may be slightly degraded)
    - 2 – Final mounting angle convergence
    - 255 – Done
- `<state_verified>` - 1 if the navigation state has been verified after a hot start and calibration can proceed, 0 otherwise
- `<gyro_percent_complete>` - Gyroscope correction calibration status: 0=incomplete, 1=complete
- `<accel_percent_complete>` - Accelerometer correction calibration completion percentage
- `<mounting_angle_percent_complete>` - IMU mounting angle correction completion percentage

8.3 NMEA $P1MSG Definition

This is a proprietary NMEA message defined by Point One and used to send notifications to the user:

```
$P1MSG,<system_time_ns>,<severity>,<message>
```

Appendix A. Firmware Version History

AM 0.15.0, AP 0.12.0 (2022-2-1)

New Features

- **[lg69t]** Added a watchdog to restart the device if the IMU or Teseo GNSS receiver stops operating
- **[lg69t]** Halt operation in restricted geographic regions (P1-97)

Fixes

- **[lg69t]** Fixed handling of NMEA PQTMVERNO requests (P1-102)
- **[lg69t]** Fixed possible data loss during set configuration operations
- **[lg69t-am]** Improved yaw estimation when moving slowly in high multipath
- **[lg69t-am]** Reduced drift after driving under overhead obstructions
- **[lg69t-ap]** Fixed possible IMU failure during heavy CPU usage
- **[lg69t-ap]** Fixed memory corruption after reboot when wheel ticks are enabled
- **[lg69t-ap]** Fixed high CPU usage when applying Doppler measurements

AM 0.14.2, AP 0.11.2 (2022-1-30)

Changes

- **[p1_runner]** Renamed `p1_runner` tool, formerly `quectel_runner`

Fixes

- **[p1_runner, config_tool]** Fixed crash due to incorrect dependency
- **[config_tool]** Fixed incorrect handling of positional boolean arguments (e.g., `config_tool.py apply uart1_diagnostics_enabled true`)

AM 0.14.1, AP 0.11.1 (2022-1-12)

Changes

- **[lg69t]** Improved cold start TTFF when a minimal number of signals are present (P1-28)
- **[lg69t]** Improved initial position if RTK corrections are available
- **[lg69t]** Disabled PQTMGNSS and PQTMVER,SUB messages on boot (P1-102)
- **[lg69t-ap]** Improved use of Doppler measurements
- **[lg69t-ap]** Disabled FusionEngine IMU measurement output on UART1 by default
- **[lg69t, config_tool]** Added rate control support for NMEA PQTM version and GNSS messages (P1-102)

Fixes

- **[lg69t]** Fixed high CPU usage before initialization
- **[lg69t]** Reduced CPU usage during RTK float operation
- **[lg69t]** Fixed incorrect week number when starting up with ephemeris data saved ½-1 week ago

AM 0.14.0, AP 0.11.0 (2022-12-23)

Changes

- **[lg69t]** Reduced noise in low-quality position solutions prior to system initialization
- **[lg69t]** Minor improvements related to retaining RTK fixes while approaching signal obstructions
- **[lg69t-ap]** Reduced discontinuities when resuming GNSS updates after dead reckoning
- **[lg69t-ap]** Improved drift behavior while stationary in high multipath (P1-93)
- **[lg69t-ap]** Made yaw initialization more robust

Fixes

- **[lg69t]** Prevented some spurious resets in high multipath
- **[lg69t]** Fixed a rare issue causing occasional corrections data loss due to timestamp precision
- **[lg69t]** Fixed possible crash when saving configuration to flash
- **[lg69t-ap]** Fixed issues with uninitialized and uncalibrated IMU biases
- **[lg69t-ap]** Fixed accelerometer bias status going back to 0 after calibration completed

AM 0.13.0, AP 0.10.0 (2022-12-2)

New Features

- **[lg69t]** Added support for FusionEngine ROS Pose, GPSFix, and IMU messages
- **[lg69t-ap]** Added support for FusionEngine WheelSpeedMessage and VehicleSpeedMessage output
- **[teseo]** Updated Teseo firmware to 5.8.19.1-20221201, disabling SIGQM to reduce UART output bandwidth

Changes

- **[lg69t]** Improved TTFF and initial filter position accuracy in high multipath or low C/N0 environments (P1-31, P1-32, P1-40, P1-41, P1-66, P1-87)
- **[lg69t]** Improved position drift in high multipath environments

Fixes

- **[lg69t]** Resolved issue preventing RTK fixing in high multipath while stationary (P1-64)
- **[lg69t]** Fixed infrequent crash due to stack overflow when requesting a cold start (P1-37)
- **[lg69t]** Fixed CRC errors on FusionEngine VehicleTickMessages

- **[lg69t-ap]** Fixed hardware wheel tick handling when going in reverse

AM 0.12.1, AP 0.9.1 (2022-11-4)

New Features

- **[lg69t]** Added FusionEngine protocol set/get message rate messages for "current" transport
- **[lg69t]** Added new FusionEngine FaultControlMessage, capable of injecting crashes/COCOM limits violations, and enabling/disabling GNSS at runtime (P1-83)
- **[config_tool]** Added fault control commands (disable GNSS, force a crash, etc.) (P1-83)
- **[teseo]** Updated Teseo firmware 5.8.19.1-1024 to include output of SIGQM messages

Changes

- **[lg69t]** Revised tightly coupled measurement tuning for Teseo receiver
- **[lg69t]** Improved position and velocity estimation while multipath environment is degrading
- **[lg69t]** Report prior solution type while stationary and not incorporating GNSS measurements due to high multipath (P1-90)
- **[lg69t]** Populate user output from Teseo MSM prior to availability of week number

Fixes

- **[lg69t]** Fixed missing user response if buffer is too small to hold get rate request response
- **[lg69t]** Fixed missing response to unsupported user commands (P1-94)
- **[lg69t]** Fixed handling of unsupported transports and protocols in set/get rate requests (P1-94)
- **[lg69t]** Fixed handling of unsupported message IDs for FusionEngine protocol in get rate requests (P1-94)
- **[lg69t]** Wait for cold start completion before using ephemeris from Teseo (P1-28)
- **[lg69t-am]** Fixed missing output after a cold start
- **[lg69t-am]** Aligned GGA timestamps to exact 1/10 second boundaries (P1-91)
- **[lg69t-am]** Prevent some spurious resets in difficult environments (P1-84)

Version 0.12.0 (2022-10-6)

New Features

- **[lg69t]** Added support for setting/querying output rates for multiple messages in one command

Changes

- **[lg69t]** Improved performance in high multipath and obstructed environments (driving under overhead roads, overpasses, etc.) (P1-22, P1-84)
- **[lg69t]** Improved position integrity check performance
- **[teseo]** Updated Teseo firmware to version 5.8.19.1-0930 to resolve issues tracking low C/N0 signals (P1-68, P1-84, P1-86)

Fixes

- **[lg69t]** Resolved slow position recovery after drifts during outages
- **[lg69t]** Fixed rare unexpected reset while navigating in extremely high multipath (P1-84)

AM 0.11.0, AP 0.8.0 (2022-9-9)

New Features

- **[lg69t]** Detect wheel tick update rate automatically (P1-36)
- **[lg69t]** Added apply-and-save support to SetMessageRate command

Changes

- **[lg69t]** Changed default GSA/GSV rate back to 10 Hz (P1-77.1)
- **[lg69t]** Turned off Relative ENU Position message by default
- **[lg69t]** Made minor navigation improvements in multipath and Non-Line-of-Sight environments (P1-65)
- **[lg69t]** Stopped sending boot messages on reset
- **[lg69t]** Enabled watchdog by default (P1-44)
- **[lg69t]** Updated required Teseo GNSS receiver firmware version (P1-39, P1-63)
- **[lg69t]** Made filter resets caused by internal monitoring less severe
- **[lg69t]** Send CrashLog periodically.

Fixes

- **[lg69t]** Fixed rate control support for NMEA RMC messages
- **[lg69t]** Output NMEA RMC before GGA so date is always available
- **[lg69t]** Reject measurements for BDS GEO satellites with erroneous range from receiver
- **[lg69t]** Fixed missing P1/GPS timestamps in user output after certain reset events
- **[lg69t]** Fixed potential over-length string in CoCom limit message
- **[lg69t]** Route version messages only to configured destination port (P1-77.2)
- **[lg69t]** Fixed a bug affecting SetConfig request with save
- **[lg69t]** Fixed RTCM MSM decoding bug affecting B1C signals
- **[lg69t]** Handle delayed cold starts of internal Teseo GNSS receiver

AM 0.10.1, AP 0.7.1 (2022-8-16)

Changes

- **[lg69t]** Added low-speed dynamics handling in multipath environments

Fixes

- **[lg69t]** Fixed hang on navigation engine reset

AM 0.10.0, AP 0.7.0 (2022-8-12)

New Features

- **[lg69t, config_tool]** Added rate control support for individual message types
- **[quectel_runner]** Added hot/warm/cold reset support (default hot start automatically)

Changes

- **[lg69t]** Enable watchdog automatic restart by default
- **[lg69t]** Improved time to first fix (TTFF)
- **[lg69t]** Improved positioning output in very low C/N0 environments

Fixes

- **[lg69t]** Fixed missing NMEA and FusionEngine output when solution type is invalid
- **[lg69t]** Fixed incorrect GSA/GSV results in high-multipath or low-C/N0 environments
- **[lg69t]** Fixed missing DOP values in NMEA output
- **[lg69t]** Fixed unexpected FusionEngine VersionInfo response timeouts
- **[lg69t]** Fixed possible crash when issuing a reset request
- **[lg69t]** Fixed hang while capturing crash log information after an error

AM 0.9.0, AP 0.6.1 (2022-7-28)

New Features

- **[lg69t-am]** Added FusionEngine control messages for enabling/disabling individual messages on each UART
- **[lg69t, config_tool]** Added watchdog to restart automatically on crash (disabled by default; to enable: `config_tool.py apply watchdog_enabled true`)

Changes

- **[lg69t-ap]** Improved dead reckoning and stationary detection performance
- **[config_tool]** Changed message rate control syntax and added additional documentation

Fixes

- **[lg69t]** Added workaround for Teseo incorrect MSM multi-message bit bug in highly challenged environments resulting in duplicate or dropped measurements
- **[lg69t]** Fixed possible time lag and measurement rejection after very long GNSS outages (10+ minutes)
- **[lg69t-am]** Resolved issues causing lower-than-expected RTK fixing rate
- **[lg69t-am]** Resolved issues with dropped satellites when incoming MSM data has a very large number of signals

- **[lg69t-am]** Fixed issue where RTK corrections were unavailable or degraded for a while after one or more MSM messages was dropped
- **[lg69t-am]** Fixed unexpected halt behavior during out-of-memory error handling
- **[lg69t-am]** Fixed rapidly toggling solution type when entering parking garages and other highly challenged environments
- **[lg69t-am]** Fixed possible crash when entering a parking garage or tunnel and previously tracking a very large number of GNSS signals
- **[lg69t-ap]** Fixed missing FusionEngine IMU measurement output
- **[lg69t-ap]** Fixed empty FusionEngine version string in VersionInfo message
- **[config_tool]** Fixed broken read all command (`config_tool.py read`)

AP 0.6.0 (2022-7-22)

New Features

- **[lg69t-ap]** Output hardware wheel tick data using FusionEngine VehicleTickMeasurement messages for logging and display
- **[lg69t-ap]** Added FusionEngine control messages for enabling/disabling individual messages on each UART

Changes

- **[lg69t-ap]** Improved dead reckoning performance with and without wheel speed data
- **[lg69t-ap]** Reduced latency of calculated position solutions
- **[lg69t-ap]** Improved yaw and vehicle direction initialization in highly challenged GNSS environments
- **[lg69t-ap]** Improved position integrity checking and unexpected resets in highly challenged GNSS environments
- **[lg69t-ap]** Output FusionEngine IMUMeasurement messages on both UARTs by default (UART1 now contains NMEA + FusionEngine IMUMeasurement by default)

Fixes

- **[lg69t-ap]** Resolved issues causing lower-than-expected RTK fixing rate
- **[lg69t-ap]** Resolved issue causing with duplicate or misaligned GPS timestamps
- **[lg69t-ap]** Resolved issues with dropped satellites when incoming MSM data has a very large number of signals
- **[lg69t-ap]** Fixed issue where RTK corrections were unavailable or degraded for a while after one or more MSM messages was dropped
- **[lg69t-ap]** Fixed unexpected halt behavior during out-of-memory error handling
- **[lg69t-ap]** Fixed possible crash when entering a parking garage or tunnel and previously tracking a very large number of GNSS signals
- **[lg69t-ap]** Resolved issues causing vehicle to drive backward when entering parking garages or tunnels

AM 0.8.5, AP 0.5.5 (2022-6-19)

Changes

- **[lg69t]** Output NMEA $P1MSG and FusionEngine EventNotification messages on fatal errors
- **[lg69t]** Enable software reset after a fatal error
- **[lg69t]** Added additional diagnostic and profiling data

Fixes

- **[lg69t]** Fixed out-of-memory error with very large numbers of signals

AM 0.8.4, AP 0.5.4 (2022-6-14)

Changes

- **[lg69t]** Adjusted memory pool allocations

Fixes

- **[lg69t]** Fix crash with large number of signals
- **[lg69t]** Fix crash logging when debug probe is not attached

AM 0.8.3, AP 0.5.3 (2022-6-9)

Changes

- **[lg69t]** Output NMEA $P1MSG and FusionEngine EventNotification messages at 1 Hz after a CoCom limit is reached, and continue to output "invalid" pose messages
- **[lg69t]** Enable software reset of CoCom limit status
- **[teseo]** Updated to Teseo firmware (5.8.18.1;BETA0427) to address low C/N0 tracking issues

Fixes

- **[lg69t]** Fixed bug preventing use of carrier phase and RTK fixing

AM 0.8.2, AP 0.5.2 (2022-6-7)

Fixes

- **[lg69t]** Fixed possible initialization failure after a long GNSS outage
- **[lg69t]** Fixed incorrect GPS timestamps in FusionEngine output
- **[lg69t]** Fixed incorrect elevation/azimuth for satellites without ephemeris data in NMEA GSV messages

AM 0.8.1, AP 0.5.1 (2022-6-3)

New Features

- **[lg69t-ap]** Added wheel tick pulse capture support

Fixes

- **[lg69t]** Fixed potential crash when too many signals are available with corrections

AM 0.8.0, AP 0.5.0 (2022-5-28)

New Features

- **[lg69t]** Added new FusionEngine RelativeENUPositionMessage support to output the rover position relative to a local RTK base station
- **[lg69t]** Apply US DoC COCOM limits
- **[lg69t]** Added support for software-commanded processor reboot
- **[lg69t-ap]** Added software shutdown command to save calibration data
- **[config_tool]** Automatically detect the LG69T serial ports in quectel_runner and config_tool
- **[config_tool]** Added Mac OS support

Changes/Improvements

- **[lg69t]** Significant performance improvements in high-multipath environments (dense urban canyons, driving under overhead roads/bridges, etc.)
- **[lg69t-ap]** Enable the use of standalone and non-fixed GNSS solutions for calibration
- **[lg69t-ap]** Added support for up to 100 Hz wheel speed data
- **[lg69t-ap]** Improved calibration time
- **[lg69t-ap]** Improved DR direction detection when wheel speeds are not available
- **[lg69t-ap]** Report a reduced-quality position solution in highly challenged environments where the system cannot initialize
- **[quectel_runner]** Warn if there are no FusionEngine messages in the quectel_runner input
- **[quectel_runner]** Include FusionEngine and NMEA data counts in the quectel_runner status print
- **[teseo]** Updated to latest Teseo firmware (5.8.18.1;BETA0510) to resolve signal quality and other issues

Fixes

- **[lg69t]** Fixed gap in data when a large burst of RTCM corrections data is sent by the user (multiple seconds of data accumulated and sent at once)
- **[lg69t]** Fixed system startup error when only GPS is present
- **[lg69t]** Fixed possible crash if there are very few signals available
- **[lg69t]** Fixed crash when the UART baud rate is changed
- **[lg69t]** Fixed crash if last flash sector set to 0x00

- **[bootloader]** Updated to Quectel bootloader version 1.0.2 to resolve possible flash corruption when upgrading Teseo firmware
- **[config_tool]** Fixed handling of user-specified serial port
- **[config_tool]** Output quectel_runner position updates based on P1 time when GPS time isn't available
- **[config_tool]** Fixed distutils deprecated warning in Python 3.10+

AM 0.7.2-rc1 (2022-5-18)

Changes/Improvements

- **[teseo]** Updated to latest Teseo firmware (5.8.18.1;BETA0510) to resolve signal quality and other issues

Fixes

- **[lg69t]** Fixed occasional out-of-order sequence numbering for FusionEngine messages

AM 0.7.1 (2022-5-15)

New Features

- **[lg69t-am]** Report a reduced-quality position solution in highly challenged environments where the system cannot initialize
- **[lg69t-am]** Automatically detect the LG69T serial ports in quectel_runner and config_tool

Changes/Improvements

- **[lg69t-am]** Reduced the CPU clock speed to 300 MHz to lower power consumption
- **[quectel_runner]** Warn if there are no FusionEngine messages in the quectel_runner input
- **[quectel_runner]** Include FusionEngine and NMEA data counts in the quectel_runner status print

Fixes

- **[lg69t-am]** Fixed crash if there are very few signals available
- **[lg69t-am]** Fixed crash when the UART baud rate is changed
- **[quectel_runner]** Output quectel_runner position updates based on P1 time when GPS time isn't available
- **[quectel_runner]** Fixed distutils deprecated warning in Python 3.10+

AM 0.7.0, AP 0.4.0 (2022-5-6)

New Features

- **[lg69t]** Added GNSS measurement engine reset option; automatically restart the Teseo receiver when on cold start requests
- **[lg69t]** Added combined configuration apply-and-save support (FusionEngine SetConfigMessage)
- **[lg69t-ap]** Added external wheel speed support via FusionEngine messages
- **[quectel_runner, config_tool]** Added baud rate controls to Python applications
- Include QGNSS and GNSS Flash Tool in the release

Changes/Improvements

- **[lg69t]** Improved multipath performance and removal of large outliers and non-line-of-sight signals
- **[lg69t]** Output 3 integer digits with leading zeros for NMEA COG values
- **[lg69t]** Adjusted Teseo startup checks to improve time to first fix (TTFF)
- **[lg69t-ap]** Improved dead reckoning performance
- **[lg69t-ap]** Added intermediate calibration stage to reduce initial time to accurate position solutions
- **[quectel_runner]** Added additional options to quectel_runner and changed default behavior for log and output formats

Fixes

- **[lg69t]** Stability improvements
- **[lg69t-ap]** Fixed calibration save and load issues

AM 0.6.8, AP 0.3.3 (2022-4-21)

Changes/Improvements

- **[lg69t]** Reduced LG69T CPU clock frequency to 400 MHz to lower power consumption and operating temperature
- **[lg69t]** Increased NMEA RMC COG field precision to 2 decimal places
- **[lg69t-ap]** Improved performance of integrity check after hot start initialization

Fixes

- **[lg69t]** Fixed use of invalid measurements when Teseo clock bias is not yet stabilized

AM 0.6.7, AP 0.3.2 (2022-3-21)

Fixes

- **[lg69t]** Fixed missing Python files

AM 0.6.6, AP 0.3.1 (2022-3-20)

New Features

- **[lg69t-ap]** Added FusionEngine and NMEA calibration status output messages
- **[lg69t-ap]** Added support for ST ASM330 IMU

Changes/Improvements

- **[lg69t]** Improved non-differential GNSS position performance when non-L1 code biases are not available
- **[lg69t-ap]** Resolved stability issues due to memory usage
- **[lg69t-ap]** Increased IMU mounting angle tolerance
- **[lg69t-ap]** Improved navigation engine position integrity monitoring

Fixes

- **[lg69t]** Fixed GNSS engine failure when switching from RTK to standalone operation in a denied environment
- **[lg69t]** Fixed use of anomalous GNSS satellites marked as "healthy"
- **[lg69t]** Fixed NMEA course over ground (COG) calculation
- **[lg69t]** Fixed issue with performance rate profiling metrics
- **[lg69t-ap]** Resolved GNSS performance issues before calibration has begun to converge
- **[lg69t-ap]** Resolved unexpected issue causing some IMU data to be discarded
- **[lg69t-ap]** Resolved dynamic memory usage issues related to measurement reordering
- **[lg69t-ap]** Fixed unexpected cold start when warm start was requested
- **[quectel_runner, config_tool]** Fixed issues with pre-compiled Windows .exe files for `quectel_runner` and `config_tool` applications

AP 0.3.0 (2022-2-28)

New Features

- **[lg69t-ap]** Added automatic device calibration support
- **[lg69t-ap]** Added automatic navigation state save and load support
- **[lg69t-ap]** Added navigation/calibration reset support
- **[lg69t-ap]** Added temperature data logging support

Changes/Improvements

- **[lg69t-ap]** Performance improvements, particularly when operating in challenging environments
- **[lg69t-ap]** Resolved stability issues due to memory usage

AM 0.6.5 (2022-2-25)

Improvements/Changes

- **[lg69t-am]** Include HDOP value in NMEA output
- **[lg69t-am]** Estimate track angle (course over ground) in NMEA output when moving

Fixes

- **[lg69t-am]** Resolved a possible initialization failure in high multipath environments

AM 0.6.4 (2022-2-18)

Fixes

- **[lg69t-am]** Resolved an issue preventing system integrity check resets
- **[lg69t-am]** Fixed issue preventing logging of configuration settings data

AM 0.6.3 (2022-2-8)

Fixes

- **[lg69t-am]** Resolved unexpected resets and solution gaps caused by internal resets due to multipath
- **[lg69t-am]** Fixed empty solution messages (no timestamps or signal tracking status) when an internal reset occurs

AM 0.6.2 (2022-1-23)

Improvements/Changes

- **[lg69t-am]** Output NMEA $PQTMGNSS and $PQTMVER messages on startup, not $PQTMVERNO
- **[lg69t-am]** Output "invalid" FusionEngine and NMEA solution messages on startup before the GNSS receiver starts tracking

Fixes

- **[lg69t-am]** Resolved stability issues

AM 0.6.1 (2022-1-14)

Improvements/Changes

- **[lg69t-am]** Output $PQTMVERNO messages immediately on startup
- **[lg69t-am]** Output empty NMEA GSA/GSV messages immediately, provided the Teseo has time available
- **[lg69t-am]** Changed the default output configuration for UARTs 1 and 2
- **[lg69t-am]** Increased the RTCM MSM max signal types threshold to account for VRS base stations tracking many signals

Fixes

- **[lg69t-am]** Fixed incorrect firmware version in FusionEngine version info message
- **[lg69t-am]** Fixed NMEA GSV # messages count when # SVs is divisible by 4
- **[lg69t-am]** Fixed timing issue when there are outages in the measurement data
- **[lg69t-am]** Fixed possible system hang when outputting data to the user

AM 0.6.0 (2022-1-7)

New Features

- **[lg69t-am]** Added user control for routing of message protocols to both UARTs
- **[lg69t-am]** Allow all input protocols (FusionEngine and NMEA) on both UARTs
- **[lg69t-am]** Added ionosphere delay model support (requires Teseo firmware update)
- **[lg69t-am]** Added Quectel-controllable firmware version and support for NMEA $PQTMVERNO command

Improvements/Changes

- **[lg69t-am]** Disabled use of B2a for standalone operation (accurate group delay information not available from Teseo)
- **[lg69t-am]** Improved fixing performance

Fixes

- **[lg69t-am]** Addressed unexpected position resets and fixed position issues in high multipath environments
- **[lg69t-am]** Resolved NMEA-0183 output format issues

AM 0.5.0 (2021-12-13)

New Features

- **[lg69t-am]** Added FusionEngine configuration message support, including serial baud rate control
- **[lg69t-am]** Added FusionEngine control, version, and and diagnostic messages

Improvements/Changes

- **[lg69t-am]** Allow the use of pseudorange measurements down to 20 dB-Hz
- **[lg69t-am]** Performance improvements
- **[lg69t-am]** Include geoid undulation and MSL height in NMEA position output
- **[lg69t-am]** Validate the Teseo receiver firmware version on startup, and include it in the FusionEngine version message

Fixes

- **[lg69t-am]** Fixed missing solution output when no BeiDou signals are available
- **[lg69t-am]** Fixed reported position and velocity standard deviation in FusionEngine pose output
- **[lg69t-am]** Fixed out-of-memory issue when receiving ephemeris data from the RTCM corrections source

AM 0.4.3 (2021-11-24)

New Features

- **[lg69t-am]** Added additional system profiling/monitoring support

Improvements/Changes

- **[lg69t-am]** Store full 64b timestamps in RTCM 999 SENS messages to avoid rollover ambiguity if data is dropped by the host computer

Fixes

- **[lg69t-am]** Fixed RTCM 1006 support for base station ID 1 (e.g., LG69T-AS)
- **[lg69t-am]** Fixed issues with base station data when GPS is not present
- **[lg69t-am]** Increased RTCM max signal type limit, needed to use data from certain base stations
- **[lg69t-am]** Ignore bogus RTCM MSM lock times if all identical (seen on LG69T-AS)

AP 0.2.0 (2021-10-29)

New Features

- **[lg69t-ap]** Added lever arm specification support
- **[lg69t-ap]** Enabled RTCM MSM5 and MSM6 message types
- **[lg69t-ap]** Added navigation engine integrity monitoring support
- **[lg69t-ap]** Added FusionEngine pose and GNSS info output messages

Fixes

- **[lg69t-ap]** Fixed occasional discarding of IMU measurements due to latency
- **[lg69t-ap]** Fixed potential hang when reading incoming Teseo data
- **[lg69t-ap]** Fixed potential crash when a lot of GNSS signals are available
- **[lg69t-ap]** Discard Teseo 1006 messages to avoid conflict with incoming corrections
- **[lg69t-ap]** Improved memory usage and stability

AM 0.4.2 (2021-10-27)

New Features

- **[lg69t-am]** Enabled RTCM MSM5 and MSM6 message types

Fixes

- **[lg69t-am]** Corrected possible numerical instability when restarting the ambiguity search
- **[lg69t-am]** Fixed rare initialization failure in certain locations due to satellite geometry

AP 0.1.0 (2021-10-15)

Initial version (GNSS/INS+RTK)

Features

- GNSS and IMU sensor fusion support
- GNSS RTK support using supplied RTCM3 MSM4-7 corrections
- NMEA-0183 and Point One FusionEngine message formats supported

AM 0.4.1 (2021-10-5)

New Features

- **[lg69t-am]** Added FusionEngine system profiling messages

Improvements/Changes

- **[lg69t-am]** Improved RTK fixing performance when newly fixed in occluded environments

AM 0.4.0 (2021-10-1)

New Features

- **[lg69t-am]** Added Quectel bootloader with Teseo programming support
- **[lg69t-am]** Added support for RTCM non-single receiver operation (i.e., asynchronous constellations)
- **[lg69t-am]** Added QGNSS usage and data logging instructions

Improvements/Changes

- **[lg69t-am]** Improved RTK fixing performance after long outages
- **[lg69t-am]** Always output leading digits for NMEA latitude/longitude degree values
- **[lg69t-am]** Added NTRIP v1 support to the `quectel_runner` Python application
- **[lg69t-am]** Extended BeiDou PRN range for updated RTCM MSM message definition
- **[lg69t-am]** Increased max supported simultaneous BeiDou satellites
- **[quectel_runner]** Added note about Windows and Linux driver version requirements for the CP210x USB/serial device

Fixes

- **[lg69t-am]** Detect invalid and discard measurements from Teseo receiver
- **[quectel_runner]** Fixed Python handling of NTRIP server hostname without leading http://

AM 0.3.0 (2021-8-20)

New Features

- **[lg69t-am]** Added BeiDou B2a support (must be supported by base station for use during RTK operation)
- **[lg69t-am]** Include station ID and corrections data age in NMEA GPGGA output when RTK corrections are used

Improvements/Changes

- **[lg69t-am]** Improved RTK performance when very few signals are available
- **[lg69t-am]** Fixed rare issue affecting use of certain signal types during standalone operation
- **[lg69t-am]** Stability improvements, including resolution of fragmentation issue noted in 0.2

AM 0.2.0 (2021-8-23)

New Features

- **[lg69t-am]** Added RTCM 999 IMU data output (not used)
- **[lg69t-am]** Added RTCM 4050 device reset support
- **[lg69t-am]** Added NMEA GSV, GSA, GLL, VTG, RMC support
- **[lg69t-am]** Output NMEA on both UART1 and UART2

Improvements/Changes

- **[lg69t-am]** Resolved signal availability issues for RTK operation
- **[lg69t-am]** Resolved various performance issues for standalone and RTK operation
- **[lg69t-am]** Improved position integrity check robustness
- **[lg69t-am]** Resolved memory issues related to NMEA output
- **[quectel_runner]** Changed Python `--ntrip` and `--polaris` options to use comma-separated values

Known Issues

- **[lg69t-am]** System halts from memory fragmentation during RTK operation in open sky environments after extended periods of time

AM 0.1.0 (2021-7-23)

Initial version (GNSS+RTK navigation)