NGIMU User Manual

Version 1.5

Public Release

Document updates

This document is continuously being updated to incorporate additional information requested by users and new features made available in software and firmware updates. Please check the x-io Technologies website for the latest version of this document and device firmware.

Document version history

Date	Document version	Description
16 Oct 2019	1.5	• Update photos of board and plastic housing
24 Jul 2019	1.4	• Update RSSI sample rate • Remove altimeter as future feature • Add units to linear and earth acceleration descriptions • Remove processor from temperature message • Add battery low indication to LED behaviour table
07 Nov 2017	1.3	• Update the button information • Add analogue inputs section • Replace mechanical drawings with links to website • Update description of the LED indicating SD card status
10 Jan 2017	1.2	• Add send rates, sample rates, and timestamps section • Describe OSC time tag in more detail • Add auxiliary serial interface section • Add appendix for integration of a GPS module
19 Oct 2016	1.1	• Add description of the LED indicating SD card activity • Fix footnote error in overview section
23 Sep 2016	1.0	• Indicate that button must be held for half a second to switch on • Update description of OSC argument overloading • Include percentage in RSSI message • Update plastic housing photo and mechanical drawing • Add AHRS initialise and zero commands • Add altitude message
19 May 2016	0.6	• Add echo command • Add RSSI message • Add magnitudes message
29 Mar 2016	0.5	• Add communication protocol section • Correct analogue input voltage range to 3.1 V • Update LED section • Update annotated photo of board • Update plastic housing photo • Update mechanical drawing of board
19 Nov 2015	0.4	• Update photo and mechanical drawing of latest prototype plastic housing

		• Include mechanical drawing of board
30 Jun 2015	0.3	• Correct serial pinout tables • Mark pin 1 on annotated photo of board
9 Jun 2015	0.2	• Include photo and mechanical drawing of latest prototype plastic housing • Small tables are not split across pages
12 May 2015	0.1	• Update photo of prototype plastic housing
10 May 2015	0.0	• Initial release

Table of Contents

1. Overview

The Next Generation IMU (NGIMU) is a compact IMU and data acquisition platform that combines on-board sensors and data processing algorithms with a broad range of communication interfaces to create a versatile platform well suited to both real-time and data-logging applications.

The device communicates using OSC and so is immediately compatible with many software applications and straight forward to integrate with custom applications with libraries available for most programming languages.

1.1. On-board sensors & data acquisition

- Triple-axis gyroscope (±2000°/s, 400 Hz sample rate)
- Triple-axis accelerometer (±16g, 400 Hz sample rate)
- Triple-axis magnetometer (±1300 µT)
- Barometric pressure (300-1100 hPa)
- Humidity
- Temperature[1]
- Battery voltage, current, percentage, and time remaining
- Analogue inputs (8 channels, 0-3.1 V, 10-bit, 1 kHz sample rate)
- Auxiliary serial (RS-232 compatible) for GPS or custom electronics/sensors
- Real-time clock and

1.2. On-board data processing

- All sensors are calibrated
- AHRS fusion algorithm provides a measurement of orientation relative to the Earth as a quaternion, rotation matrix, or Euler angles
- AHRS fusion algorithm provides a measurement of linear acceleration
- All measurements are timestamped
- Synchronisation of timestamps for all devices on a Wi-Fi network[2]

1.3. Communication interfaces

- USB
- Serial (RS-232 compatible)
- Wi-Fi (802.11n, 5 GHz, built-in or external antennae, AP or client mode)
- SD card (accessible as an external drive via USB)

1.4. Power management

- Power from USB, external supply or battery
- Battery charging via USB or external supply
- Sleep timer

[1] On-board thermometers are used for calibration and are not intended to provide an accurate measurement of ambient temperature.

[2] Synchronisation requires additional hardware (Wi-Fi router and synchronisation master).

- Motion trigger wake up
- Wake up timer
- 3.3 V supply for user electronics (500 mA)

1.5. Software features

- Open-source GUI and API (C#) for Windows
- Configure device settings
- Plot real-time data
- Log real-time data to file (CSV file format for use with Excel, MATLAB, etc.)
- Maintenance and calibration tools**Error! Bookmark not defined.**

2. Hardware

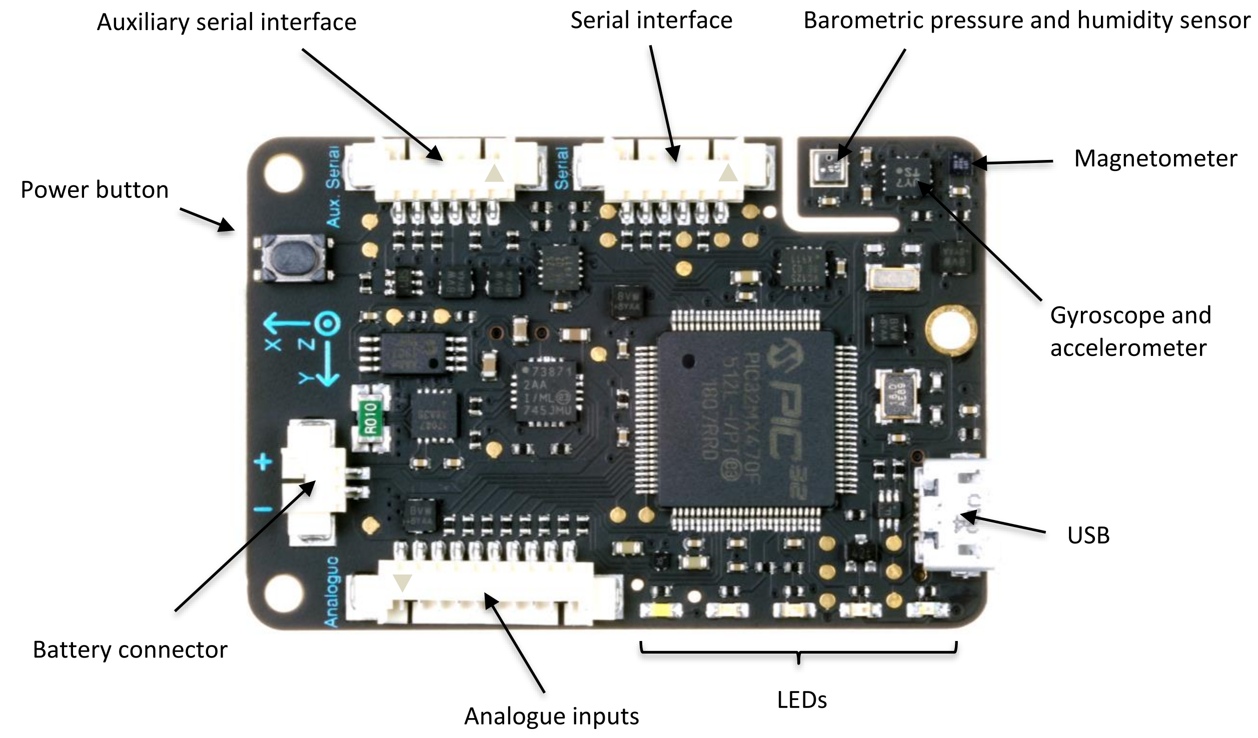

Auxiliary serial interface
Serial interface
Barometric pressure and humidity sensor
Magnetometer
Power button
Gyroscope and accelerometer
USB
Battery connector
Analogue inputs
LEDs

Figure 1: Top view of board

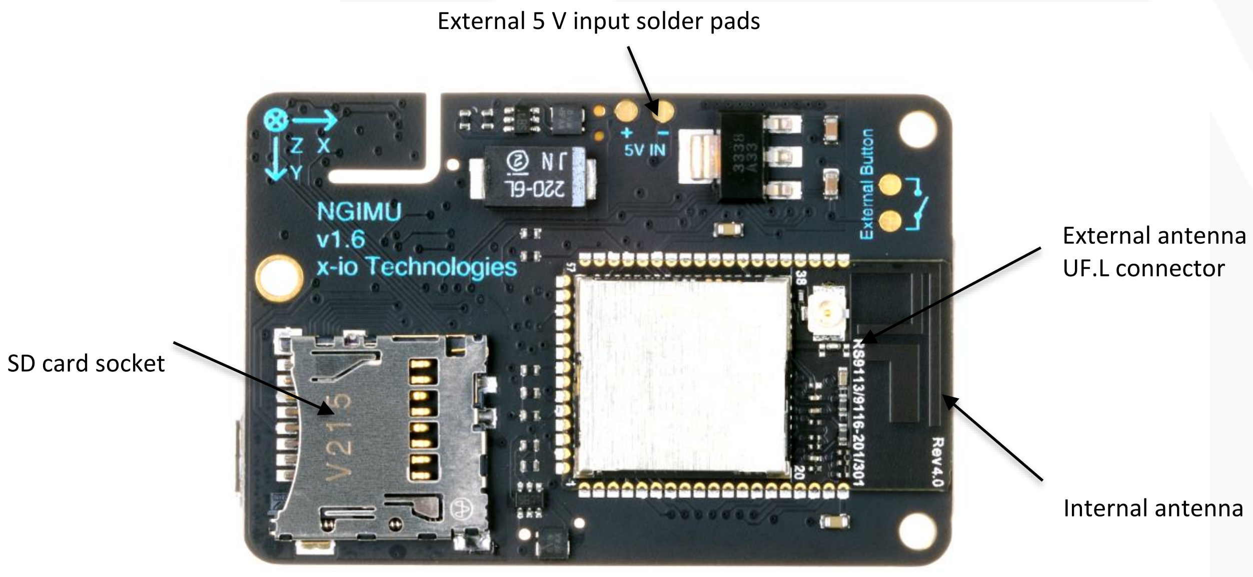

External 5 V input solder pads
External antenna UF.L connector
SD card socket
Internal antenna
NGIMU
v1.6
x-io Technologies

Figure 2: Bottom view of board

2.1. Power button

The power button is primarily used to turn the device on and off (sleep mode). Pressing the button while the device is off will turn it on. Pressing and holding the button for 2 seconds while it is on will turn it off.

The button can also be used as a data source by the user. The device will send a timestamped button message each time the button is pressed. This may provide a convenient user input for real-time applications or a useful means of marking events when logging data. See Section 7.1.1 for more information.

2.2. LEDs

The board features 5 LED indicators. Each LED is a different colour and has a dedicated role. Table 1 list the role and associated behaviour of each LED.

Colour	Indicates	Behaviour
White	Wi-Fi status	**Off** - Wi-Fi disabled **Slow flashing (1 Hz)** - Not connected **Fast flashing (5 Hz)** - Connected and waiting for IP address **Solid** - Connected and IP address obtained
Blue	-	-
Green	Device status	Indicates that the device is switched on. Will also blink each time the button is pressed or a message is received.
Yellow	SD card status	**Off** – No SD card present **Slow flashing (1 Hz)** – SD card present but not in use **Solid** - SD card present and logging in progress
Red	Battery charging	**Off** - Charger not connected **Solid** - Charger connected and charging in progress **Flashing (0.3 Hz)** - Charger connected and charging complete **Fast flashing (5 Hz)** - Charger not connected and battery less than 20%

Table 1: LED behaviour

Sending an identify command to the device will cause all the LEDs to rapidly flash for 5 seconds. This may be of use when trying to identify a specific device within a group of multiple devices. See Section 7.3.6 for more information.

The LEDs may be disabled in the device settings. This may be of use in applications where light from the LEDs is undesirable. The identify command may still be used when the LEDs are disabled and the green LED will still blink each time the button is pressed. This allows the user to check if the device is switched on while the LEDs are disabled.

2.3. Auxiliary serial pinout

Table 2 lists the auxiliary serial connector pinout. Pin 1 is physically marked on the connector by a small arrow, see Figure 1.

Pin	Direction	Name
1	N/A	Ground
2	Output	RTS
3	Output	3.3 V output
4	Input	RX
5	Output	TX
6	Input	CTS

Table 2: Auxiliary serial connector pinout

2.4. Serial pinout

Table 3 lists the serial connector pinout. Pin 1 is physically marked on the connector by a small arrow, see Figure 1.

Pin	Direction	Name
1	N/A	Ground
2	Output	RTS
3	Input	5 V input
4	Input	RX
5	Output	TX
6	Input	CTS

Table 3: Serial connector pinout

2.5. Analogue inputs pinout

Table 4 lists the analogue inputs connector pinout. Pin 1 is physically marked on the connector by a small arrow, see Figure 1.

Pin	Direction	Name
1	N/A	Ground
2	Output	3.3 V output
3	Input	Analogue channel 1
4	Input	Analogue channel 2
5	Input	Analogue channel 3
6	Input	Analogue channel 4
7	Input	Analogue channel 5
8	Input	Analogue channel 6
9	Input	Analogue channel 7
10	Input	Analogue channel 8

Table 4: Analogue input connector pinout

2.6. Connector part numbers

All board connectors are 1.25 mm pitch Molex PicoBlade™ Headers. Table 5 lists each part number used on the board and the recommended part numbers of the corresponding mating connectors. Each mating connector is created from a plastic housing part and two or more crimped wires.

Board connector	Part number	Mating part number
Battery	Molex PicoBlade™ Header, Surface Mount, Right-Angle, 2-way, P/N: 53261-0271	Molex PicoBlade™ Housing, Female, 2-way, P/N: 51021-0200 Molex Pre-Crimped Lead Single-Ended PicoBlade™ Female, 304mm, 28 AWG, P/N: 06-66-0015 (×2)
Auxiliary serial / Serial	Molex PicoBlade™ Header, Surface Mount, Right-Angle, 6-way, P/N: 53261-0671	Molex PicoBlade™ Housing, Female, 6-way, P/N: 51021-0600 Molex Pre-Crimped Lead Single-Ended PicoBlade™ Female, 304mm, 28 AWG, P/N: 06-66-0015 (×6)
Analogue inputs	Molex PicoBlade™ Header, Surface Mount, Right-Angle, 10-way, P/N: 53261-1071	Molex PicoBlade™ Housing, Female, 10-way, P/N: 51021-1000 Molex Pre-Crimped Lead Single-Ended PicoBlade™ Female, 304mm, 28 AWG, P/N: 06-66-0015 (×10)

Table 5: Board connector part numbers

2.7. Board dimensions

A 3D STEP file and mechanical drawing detailing all board dimensions are available on the x-io Technologies website.

3. Plastic housing

The plastic housing encloses the board with a 1000 mAh battery. The housing provides access to all board interfaces and is translucent so that the LED indicators may be seen. Figure 3 shows the board assembled with 1000 mAh battery in plastic housing.

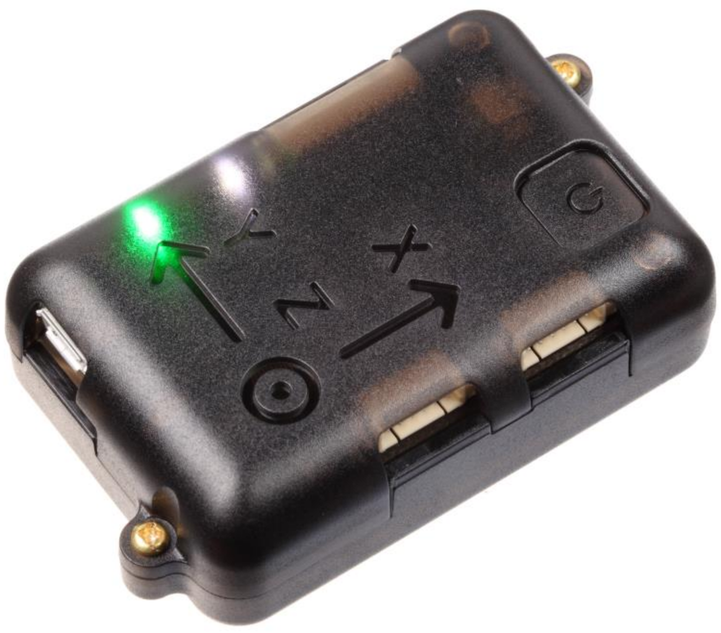


Figure 3: Board assembled with 1000 mAh battery in plastic housing

A 3D STEP file and mechanical drawing detailing all housing dimensions are available on the x-io Technologies website.

4. Analogue inputs

The analogue inputs interface is used to measure voltages and obtain data from external sensors that provide measurements as an analogue voltage. For example, a resistive force sensor can be arranged in a potential divider circuit to provide measurements of force as an analogue voltage. Voltage measurements are sent by the device as timestamped analogue inputs messages as described in Section 7.1.13.

The analogue inputs pinout is described in Section 2.3, and the part numbers for a mating connector are listed in Section 2.6.

4.1. Analogue inputs specification

- Number of channels: **8**
- ADC resolution: **10-bit**
- Sample rate: **1000 Hz**
- Voltage range: **0 V to 3.1 V**

4.2. 3.3 V supply output

The analogue input interface provides a 3.3 V output which may be used to power external electronics. This output is switched off when the device enters sleep mode to prevent the external electronics from draining the battery when the device is not active.

5. Auxiliary serial interface

The auxiliary serial interface is used to communicate with external electronics via a serial connection. For example, Appendix A describes how a GPS module may be connected directly to the auxiliary serial interface to log and stream GPS data alongside existing sensor data. Alternatively, a microcontroller connected to the auxiliary serial interface can be used to add general purpose input/output functionality.

The auxiliary serial interface pinout is described in Section 2.3, and the part numbers for a mating connector are listed in Section 2.6.

5.1. Auxiliary serial specification

- Baud rate: **7 bps to 12 Mbps**
- RTS/CTS hardware flow control: **enabled/disabled**
- Invert data lines (for RS-232 compatibility): **enabled/disabled**
- Data: **8-bit (no party)**
- Stop bits: 1
- Voltage: **3.3 V (inputs are tolerant of RS-232 voltages)**

5.2. Sending data

Data is sent from the auxiliary serial interface by sending an auxiliary serial data message to the device. See Section 7.1.15 for more information.

5.3. Receiving data

Data received by the auxiliary serial interface is sent by the device as an auxiliary serial data message as described in Section 7.2.1. Received bytes are buffered before being sent together in a single message when one of the following conditions is met:

- The number of bytes stored in the buffer matches the *buffer size*
- No bytes have been received for more than the *timeout* period
- Reception of a byte equal to the *framing character*

The *buffer size*, *timeout*, and *framing character* can be adjusting in the device settings. An example use of these settings is to set the *framing character* to the value of a new-line character

('\n', decimal value 10) so that each ASCII string, terminated by a new-line character, received by auxiliary serial interface is sent as a separate time-stamped message.

5.4. OSC passthrough

If OSC passthrough mode is enabled then the auxiliary serial interface will not send and receive in the way described in Sections 5.2 and 5.3. Instead, the auxiliary serial interface will send and receive OSC packets encoded as SLIP packets. OSC content received by the auxiliary serial interface is forwarded to all active communication channels as a timestamped OSC bundle. OSC messages received via any active communication channel that are not recognised will be forwarded to the auxiliary serial interface. This allows direct communication with third party and custom serial-based OSC devices through messages sent and received alongside existing OSC traffic.

The NGIMU Teensy I/O Expansion Example demonstrates how a Teensy (an Arduino compatible microcontroller) connected to the auxiliary serial interface can be used to control LEDs and provide sensor data using OSC passthrough mode.

5.5. RTS/CTS hardware flow control

If RTS/CTS hardware flow control is not enabled in the device settings then the CTS input and RTS output may be controlled manually. This provides a general purpose digital input and output which may be used to interface to external electronics. For example: to detect the pressing of a button or to control an LED. The RTS output state is set by sending an auxiliary serial RTS message to the device as described in Section 7.2.2. A timestamped auxiliary serial CTS message is sent by the device each time the CTS input states changes as described in Section 7.1.16.

5.6. 3.3 V supply output

The auxiliary serial interface provides a 3.3 V output which may be used to power external electronics. This output is switched off when the device enters sleep mode to prevent the external electronics from draining the battery when the device is not active.

6. Send rates, sample rates, and timestamps

The device settings allow the user to specify the send rate of each measurement message type, for example: sensors message (Section 7.1.2), quaternion message (Section 7.1.4), etc. The send rate has no effect on the sample rate of the corresponding measurements. All measurements are acquired internally at the fixed sample rates listed in Table 6. The timestamp for each measurement is created when the sample is acquired. The timestamp is therefore a reliable measurement, independent of the latency or buffering associated with a given commutation channel.

Measurement	Sample Rate
Gyroscope	400 Hz
Accelerometer	400 Hz
Magnetometer	20 Hz
Barometric pressure	25 Hz
Humidity	25 Hz
Processor temperature	1 kHz
Gyroscope and accelerometer temperature	100 Hz
Environmental sensor temperature	25 Hz
Battery (percentage, time to empty, voltage, current)	5 Hz
Analogue inputs	1 kHz
RSSI	2 Hz

Table 6: Fixed internal sample rates

If a specified send rate is greater than the sample rate of the associate measurement then measurements will be repeated within multiple messages. Repeated measurements can be identified as a repeated timestamp. It is possible to specify send rates that exceed the bandwidth of a communication channel. This will result in messages being lost. Timestamps should be used to ensure that the receiving system is robust to lost messages.

7. Communication protocol

All communication is encoded as OSC. Data sent over UDP uses OSC as per the OSC v1.0 specification. Data set over USB, serial or written to the SD card is OSC encoded as SLIP packets as per the OSC v1.1 specification. The OSC implementation uses the following simplifications:

- OSC messages sent to the device may use numerical argument types (int32, float32, int64, OSC time tag, 64-bit double, character, boolean, nil, or infinitum) interchangeably, and blob and string argument types interchangeably.
- OSC address patterns sent to the device may not contain any special characters: '?', '*', '[]', or '{}'.
- OSC messages sent to the device may be sent within OSC bundles. However, message scheduling will be ignored.

7.1. Data from device

All data sent from the device is sent as a timestamped OSC bundle containing a single OSC message. All data messages, with the exception of the button, auxiliary serial and serial messages, are sent continuously at the send rates specified in the device settings.

The timestamp of an OSC bundle is an OSC time tag. This is a 64-bit fixed-point number. The first 32 bits specify the number of seconds since 00:00 on January 1st, 1990, and the last 32 bits specify fractional parts of a second to a precision of about 200 picoseconds. This is the representation used by Internet NTP timestamps. An OSC time tag can be converted to a decimal value of seconds by first interpreting the value as a 64-bit unsigned integer and then dividing this value by 2^{32}. It is important that this calculation is implemented using a double-precision floating-point type otherwise the lack of precision will result in significant errors.

7.1.1. Button message

OSC address: `/button`

The button message is sent each time the power button is pressed. The message contains no arguments.

7.1.2. Sensors

OSC address: `/sensors`

The sensors message contains measurements from the gyroscope, accelerometer, magnetometer, and barometer. The message arguments are summarised in Table 7.

Argument	Type	Description
1	float32	Gyroscope x axis in °/s
2	float32	Gyroscope y axis in °/s
3	float32	Gyroscope z axis in °/s
4	float32	Accelerometer x axis in g
5	float32	Accelerometer y axis in g
6	float32	Accelerometer z axis in g
7	float32	Magnetometer x axis in µT
8	float32	Magnetometer y axis in µT
9	float32	Magnetometer z axis in µT
10	float32	Barometer in hPa

Table 7: Sensor message arguments

7.1.3. Magnitudes

OSC address: `/magnitudes`

The magnitudes message contains measurements of the gyroscope, accelerometer, and magnetometer magnitudes. The message arguments are summarised in Table 8: Magnitudes message arguments.

Argument	Type	Description
1	float32	Gyroscope magnitude in °/s
2	float32	Accelerometer magnitude in g
3	float32	Magnetometer magnitude in µT

Table 8: Magnitudes message arguments

7.1.4. Quaternion

OSC address: `/quaternion`

The quaternion message contains the quaternion output of the on-board AHRS algorithm describing the orientation of the device relative to the Earth (NWU convention). The message arguments are summarised in Table 9.

Argument	Type	Description
1	float32	Quaternion w element
2	float32	Quaternion x element
3	float32	Quaternion y element
4	float32	Quaternion z element

Table 9: Quaternion message arguments

7.1.5. Rotation matrix

OSC address: `/matrix`

The rotation matrix message contains the rotation matrix output of the on-board AHRS algorithm describing the orientation of the device relative to the Earth (NWU convention). The message arguments describe the matrix in row-major order as summarised in Table 10.

Argument	Type	Description
1	float32	Rotation matrix xx element
2	float32	Rotation matrix xy element
3	float32	Rotation matrix xz element
4	float32	Rotation matrix yx element
5	float32	Rotation matrix yy element
6	float32	Rotation matrix yz element
7	float32	Rotation matrix zx element
8	float32	Rotation matrix zy element
9	float32	Rotation matrix zz element

Table 10: Rotation matrix message arguments

7.1.6. Euler angles

OSC address: `/euler`

The Euler angles message contains the Euler angle output of the on-board AHRS algorithm describing the orientation of the device relative to the Earth (NWU convention). The message arguments are summarised in Table 11.

Argument	Type	Description
1	float32	Roll (x) angle in degrees
2	float32	Pitch (y) angle in degrees
3	float32	Yaw/heading (z) angle in degrees

Table 11: Euler angle message arguments

7.1.7. Linear acceleration

OSC address: `/linear`

The linear acceleration message contains the linear acceleration output of the on-board sensor fusion algorithm describing gravity-free acceleration in the sensor coordinate frame. The message arguments are summarised in Table 12.

Argument	Type	Description
1	float32	Acceleration in the sensor x axis in g
2	float32	Acceleration in the sensor y axis in g
3	float32	Acceleration in the sensor z axis in g

Table 12: Linear acceleration message arguments

7.1.8. Earth acceleration

OSC address: `/earth`

The Earth acceleration message contains the Earth acceleration output of the on-board sensor fusion algorithm describing gravity-free acceleration in the Earth coordinate frame. The message arguments are summarised in Table 13.

Argument	Type	Description
1	float32	Acceleration in the Earth x axis in g
2	float32	Acceleration in the Earth y axis in g
3	float32	Acceleration in the Earth z axis in g

Table 13: Earth acceleration message arguments

7.1.9. Altitude

OSC address: `/altitude`

The altitude message contains the measurement of altitude above sea level. The message argument is summarised in Table 14.

Argument	Type	Description
1	float32	Altitude above sea level in m

Table 14: Altitude message argument

7.1.10. Temperature

OSC address: `/temperature`

The temperature message contains the measurements from each of the device's on-board temperature sensors. The message arguments are summarised in Table 15.

Argument	Type	Description
1	float32	Gyroscope/accelerometer temperature in °C
2	float32	Barometer temperature in °C

Table 15: Temperature message arguments

7.1.11. Humidity

OSC address: `/humidity`

The humidity message contains the relative humidity measurement. The message argument is summarised in Table 16.

Argument	Type	Description
1	float32	Relative humidity in %

Table 16: Humidity message argument

7.1.12. Battery

OSC address: `/battery`

The battery message contains the battery voltage and current measurements as well as the states of the fuel gauge algorithm. The message arguments are summarised in Table 17.

Argument	Type	Description
1	float32	Battery level in %
2	float32	Time to empty in minutes
3	float32	Battery voltage in V
4	float32	Battery current in mA
5	string	Charger state

Table 17: Battery message arguments

7.1.13. Analogue inputs

OSC address: `/analogue`

The analogue inputs message contains measurements of the analogue inputs voltages. The message arguments are summarised in Table 18.

Argument	Type	Description
1	float32	Channel 1 voltage in V
2	float32	Channel 2 voltage in V
3	float32	Channel 3 voltage in V
4	float32	Channel 4 voltage in V
5	float32	Channel 5 voltage in V
6	float32	Channel 6 voltage in V
7	float32	Channel 7 voltage in V
8	float32	Channel 8 voltage in V

Table 18: Analogue inputs message arguments

7.1.14. RSSI

OSC address: `/rssi`

The RSSI message contains the RSSI (Receive Signal Strength Indicator) measurement for the wireless connection. This measurement is only valid if the Wi-Fi module is operating in client mode. The message arguments are summarised in Table 19.

Argument	Type	Description
1	float32	RSSI measurement in dBm
2	float32	RSSI measurement as a percentage where 0% to 100% represents the range -100 dBm to -50 dBm.

Table 19: RSSI message argument

7.1.15. Auxiliary serial data

OSC address: `/auxserial`

The auxiliary serial message contains the data received through the auxiliary serial interface. The message argument may be one of two types depending on the device settings as summarised in Table 20.

Argument	Type	Description
1	blob	Data received through the auxiliary serial interface.
1	string	Data received through the auxiliary serial interface with all null bytes replaced with the character pair "`/0`".

Table 20: Auxiliary serial data message argument

7.1.16. Auxiliary serial CTS input

OSC address: `/auxserial/cts`

The auxiliary serial CTS input message contains the CTS input state of the auxiliary serial interface when hardware flow control is disabled. This message is sent each time the state of the CTS input changes. The message argument is summarised in Table 21.

Argument	Type	Description
1	boolean	CTS input state. False = low, True = high.

Table 21: Auxiliary serial CTS input message argument

7.1.17. Serial CTS input

OSC address: `/serial/cts`

The serial CTS input message contains the CTS input state of the serial interface when hardware flow control is disabled. This message is sent each time the state of the CTS input changes. The message argument is summarised in Table 22.

Argument	Type	Description
1	boolean	CTS input state. False = low, True = high.

Table 22: Serial CTS input message argument

7.2. Data to device

Data is sent to the device as OSC messages. The device will not send an OSC message in response.

7.2.1. Auxiliary serial data

OSC address: `/auxserial`

The auxiliary serial message is used to send data (one or more bytes) from the auxiliary serial interface. This message may only be sent if 'OSC passthrough' mode is not enabled. The message argument is summarised in Table 23.

Argument	Type	Description
1	OSC-blob / OSC-string	Data to be transmitted from the auxiliary serial interface

Table 23: Auxiliary serial data message arguments

7.2.2. Auxiliary serial RTS output

OSC address: `/auxserial/rts`

The auxiliary serial RTS message is used to control the RTS output of the auxiliary serial interface. This message may only be sent if hardware flow control is disabled. The message argument is summarised in Table 24.

Argument	Type	Description
1	Int32/float32/boolean	RTS output state. 0 or false = low, non-zero or true = high.

Table 24: Auxiliary serial RTS output message arguments

7.2.3. Serial RTS output

OSC address: `/serial/rts`

The serial RTS message is used to control the RTS output of the serial interface. This message may only be sent if hardware flow control is disabled. The message argument is summarised in Table 25.

Argument	Type	Description
1	Int32/float32/boolean	RTS output state. 0 or false = low, non-zero or true = high.

Table 25: Serial RTS output message arguments

7.3. Commands

All commands are sent as OSC messages. The device will confirm reception of a command by sending an identical OSC message back to the host.

7.3.1. Set time

OSC address: `/time`

The set time command sets the date and time on the device. The message argument is an OSC-timetag.

7.3.2. Mute

OSC address: `/mute`

The mute command inhibits the sending of all data messages listed in Section 7.1. Command confirmation messages and setting read/write response messages will still be sent. The device will remain muted until an unmute command is sent.

7.3.3. Unmute

OSC address: `/unmute`

The unmute command will undo the mute state described in Section 7.3.2.

7.3.4. Reset

OSC address: `/reset`

The reset command will perform a software reset. This is equivalent to switching the device off and then on again. The software reset will be performed 3 seconds after the command is received to ensure that the host is able to confirm the command before it is executed.

7.3.5. Sleep

OSC address: `/sleep`

The sleep command will put the device into sleep mode (switched off). The device will not enter sleep mode until 3 seconds after the command is received to ensure that the host is able to confirm the command before it is executed.

7.3.6. Identify

OSC address: `/identify`

The identify command will cause all the LEDs to rapidly flash for 5 seconds. This may be of use when trying to identify a specific device within a group of multiple devices.

7.3.7. Apply

OSC address: `/apply`

The apply command will force the device to immediately apply all pending settings that have been written but not yet applied. The confirmation of this command is sent after all settings have been applied.

7.3.8. Restore default

OSC address: `/default`

The restore default command will reset all device settings to their factory default values.

7.3.9. AHRS initialise

OSC address: `/ahrs/initialise`

The AHRS initialise command will reinitialise the AHRS algorithm.

7.3.10. AHRS zero yaw

OSC address: `/ahrs/zero`

The AHRS zero yaw command will zero the yaw component of the current orientation of the AHRS algorithm. This command may only be issued if the magnetometer is ignored in the AHRS settings.

7.3.11. Echo

OSC address: `/echo`

The echo command may be sent with any arguments and the device will respond with an identical OSC message.

7.4. Settings

Device settings are read and written using OSC messages. The settings tab of the device software provides access to all device settings and includes detailed documentation for each setting.

7.4.1. Read

Settings are read by sending an OSC message with the corresponding setting OSC address and no arguments. The device will respond with an OSC message with the same OSC address and the current setting value as an argument.

7.4.2. Write

Settings are written by sending an OSC message with the corresponding setting OSC address and an argument value. The device will respond with an OSC message with the same OSC address and the new setting value as an argument.

Some setting writes are not applied immediately because this may result in loss of communication with the device if a setting affecting the communication channel is modified. These settings are applied 3 seconds after the last write of any setting.

7.5. Errors

The device will send error messages as an OSC message with the OSC address: `/error` and a single string argument.

A. Integrating a GPS module with the NGIMU

This section describes how to integrate an off-the-shelf GPS module with the NGIMU. The NGIMU is compatible with any serial GPS module, the "Adafruit Ultimate GPS Breakout - 66 channel w/10 Hz updates - Version 3" was chosen here for the purposes of demonstration. This module can be purchased from Adafruit or any other their distributors.

A.1. Hardware setup

The CR1220 coin cell battery clip and auxiliary serial interface connecter wires must be soldered to the GPS module board. The auxiliary serial interface connector part numbers are detailed in Section 2.6. The required connections between the auxiliary serial port and the GPS module are described in Table 26. Figure 5 shows the assembled GPS module with connector for auxiliary serial interface.

Auxiliary serial pin	GPS module pin
Ground	"GND"
RTS	Not connected
3.3 V output	"3.3V"
RX	"TX"
TX	"RX"
CTS	Not connected

Table 26: Auxiliary serial interface connections to the GPS module

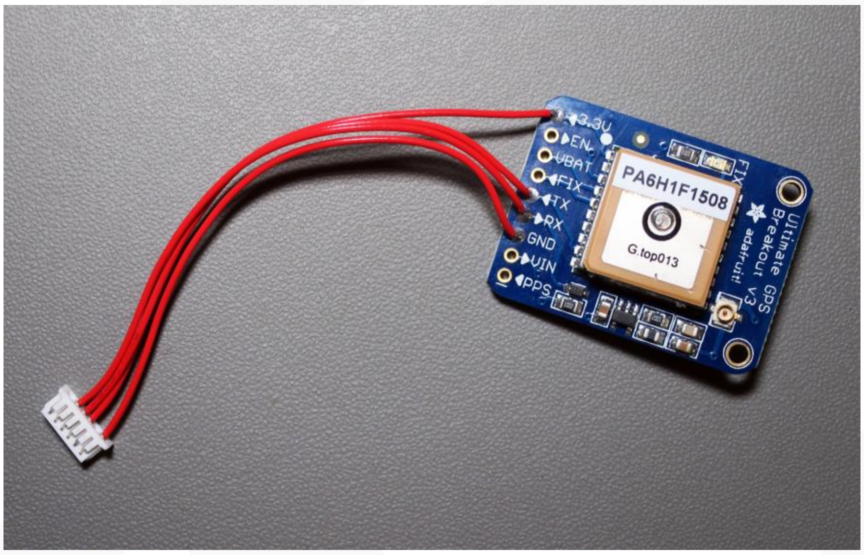
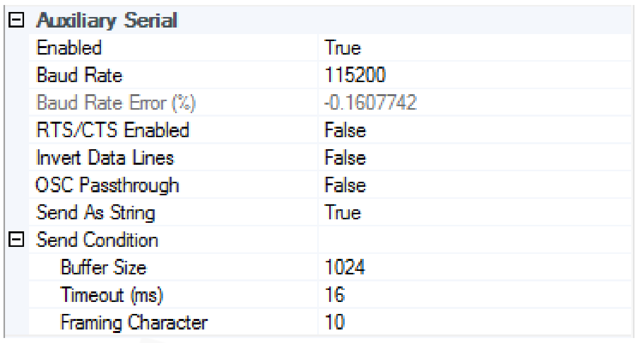


Figure 4: Assembled GPS module with connector for auxiliary serial interface

The CR1220 coin cell battery is necessary to preserve GPS module settings and to power the real-time clock while external power is not present. The GPS module will lose power each time the NGIMU is switched off. The real-time clock significantly reduces the time required to obtain a GPS lock. The battery can be expected to last approximately 240 days.

A.2. NGIMU settings

The auxiliary serial baud rate setting must be set to 9600. This is the default baud rate of the GPS module. The GPS module sends data in separate ASCII packets, each terminated by a new-line character. The auxiliary serial framing character setting must therefore be set to 10 so that each ASCII packet is timestamped and transmitted/logged by the NGIMU separately. The auxiliary serial 'send as string' setting must be enabled so that packets are interpreted as strings by the NGIMU software. All other settings should be left at default values so that the settings match those shown in Figure 5.

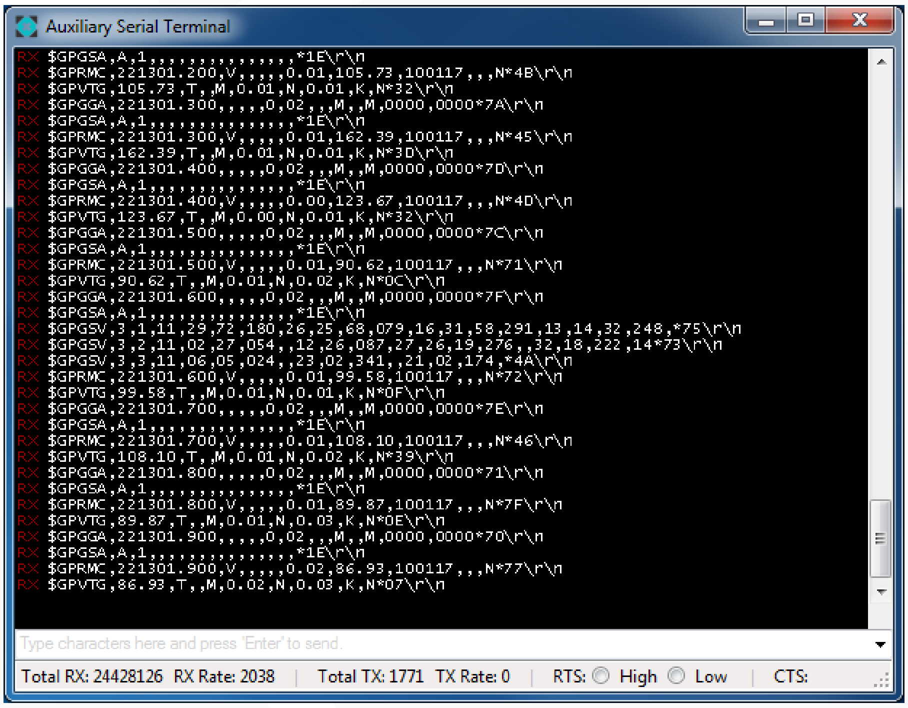


⊟ **Auxiliary Serial**	
Enabled	True
Baud Rate	115200
Baud Rate Error (%)	-0.1607742
RTS/CTS Enabled	False
Invert Data Lines	False
OSC Passthrough	False
Send As String	True
⊟ Send Condition	
Buffer Size	1024
Timeout (ms)	16
Framing Character	10

Figure 5: Auxiliary serial interface settings configured for a GPS module

A.3. Viewing and processing GPS data

Once the NGIMU settings have been configured as described in Section A.2, GPS data will be received and forwarded to all active communication channels as a timestamped auxiliary serial data message as described in Section 7.1.15. The NGIMU GUI can be used to view incoming GPS data using the Auxiliary Serial Terminal (under the Tools menu). Figure 6 shows incoming GPS data after a GPS fix has been achieved. The module may take tens of minutes to achieve a fix when powered for the first time.



```
RX $GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n
RX $GPRMC,221301.200,V,,,,,0.01,105.73,100117,,,N*4B\r\n
RX $GPVTG,105.73,T,,M,0.01,N,0.01,K,N*32\r\n
RX $GPGGA,221301.300,,,,,0,02,,,M,,M,0000,0000*7A\r\n
RX $GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n
RX $GPRMC,221301.300,V,,,,,0.01,162.39,100117,,,N*45\r\n
RX $GPVTG,162.39,T,,M,0.01,N,0.01,K,N*3D\r\n
RX $GPGGA,221301.400,,,,,0,02,,,M,,M,0000,0000*7D\r\n
RX $GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n
RX $GPRMC,221301.400,V,,,,,0.00,123.67,100117,,,N*4D\r\n
RX $GPVTG,123.67,T,,M,0.00,N,0.01,K,N*32\r\n
RX $GPGGA,221301.500,,,,,0,02,,,M,,M,0000,0000*7C\r\n
RX $GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n
RX $GPRMC,221301.500,V,,,,,0.01,90.62,100117,,,N*71\r\n
RX $GPVTG,90.62,T,,M,0.01,N,0.02,K,N*0C\r\n
RX $GPGGA,221301.600,,,,,0,02,,,M,,M,0000,0000*7F\r\n
RX $GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n
RX $GPGSV,3,1,11,29,72,180,26,25,68,079,16,31,58,291,13,14,32,248,*75\r\n
RX $GPGSV,3,2,11,02,27,054,,12,26,087,27,26,19,276,,32,18,222,14*73\r\n
RX $GPGSV,3,3,11,06,05,024,,23,02,341,,21,02,174,*4A\r\n
RX $GPRMC,221301.600,V,,,,,0.01,99.58,100117,,,N*72\r\n
RX $GPVTG,99.58,T,,M,0.01,N,0.01,K,N*0F\r\n
RX $GPGGA,221301.700,,,,,0,02,,,M,,M,0000,0000*7E\r\n
RX $GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n
RX $GPRMC,221301.700,V,,,,,0.01,108.10,100117,,,N*46\r\n
RX $GPVTG,108.10,T,,M,0.01,N,0.02,K,N*39\r\n
RX $GPGGA,221301.800,,,,,0,02,,,M,,M,0000,0000*71\r\n
RX $GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n
RX $GPRMC,221301.800,V,,,,,0.01,89.87,100117,,,N*7F\r\n
RX $GPVTG,89.87,T,,M,0.01,N,0.03,K,N*0E\r\n
RX $GPGGA,221301.900,,,,,0,02,,,M,,M,0000,0000*70\r\n
RX $GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n
RX $GPRMC,221301.900,V,,,,,0.02,86.93,100117,,,N*77\r\n
RX $GPVTG,86.93,T,,M,0.02,N,0.03,K,N*07\r\n
```

Type characters here and press 'Enter' to send.
Total RX: 24428126 RX Rate: 2038 | Total TX: 1771 TX Rate: 0 | RTS: High Low | CTS:

Figure 6: In coming GPS data displayed in the Auxiliary Serial Terminal

The default GPS module settings provide GPS data in four NMEA packet types: GPGGA, GPGSA, GPRMC, and GPVTG. The NMEA Reference Manual provides detailed description of the data contained in each of these packets.

The NGIMU software can be used to log real-time data as CSV files or to convert data logged to the SD card file to CSV files. GPS data is provided in the auxserial.csv file. The file contains two columns: the first column is the timestamp of a given NMEA packet generated by the NGIMU when the packet was received from the GPS module, and the second column is the NMEA packet. The user must handle the importing and interpretation this data.

A.4. Configuring for 10 Hz update rate

The GPS module default settings send data with a 1 Hz update rate. The module can be configured to send data with a 10 Hz update rate. This is achieved by sending command packets to adjust the settings as described in Sections A.4.1 and A.4.2. Each command packet may be sent using the NGIMU GUI's Auxiliary Serial Terminal (under the Tools menu). The GPS module will revert to default settings if the battery is removed.

The command packets described in this section are created as per the GlobalTop PMTK command packet documentation with checksums calculated using an on-line NMEA checksum calculator.

A.4.1. Step 1 – Change baud rate to 115200

Send the command packet "`$PMTK251,115200*1F\r\n`" to the GPS module. The incoming data will then appear as 'garbage' data because the current auxiliary serial baud rate of 9600 does not match the new GPS module baud rate of 115200. The auxiliary serial baud rate setting must then be set to 115200 in the NGIMU settings before for the data appears correctly again.

A.4.2. Step 2 – Change output rate to 10 Hz

Send the command packet "`$PMTK220,100*2F\r\n`" to the GPS module. The GPS module will now send data with a 10 Hz update rate.

A.4.3. Saving GPS module settings

The GPS module will save settings automatically. However, the GPS module will revert to default settings if the battery is removed.