NGIMU User Manual

Version 1.0

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
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 calendar

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
- Altimeter fusion algorithm provides a measurement of altitude[Error! Bookmark not defined.]
- All measurements are timestamped
- Synchronisation of timestamps for all devices on a Wi-Fi network[Error! Bookmark not defined.,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

[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).

- Sleep timer
- Motion trigger wake up
- Wake up timer[Error! Bookmark not defined.]
- 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

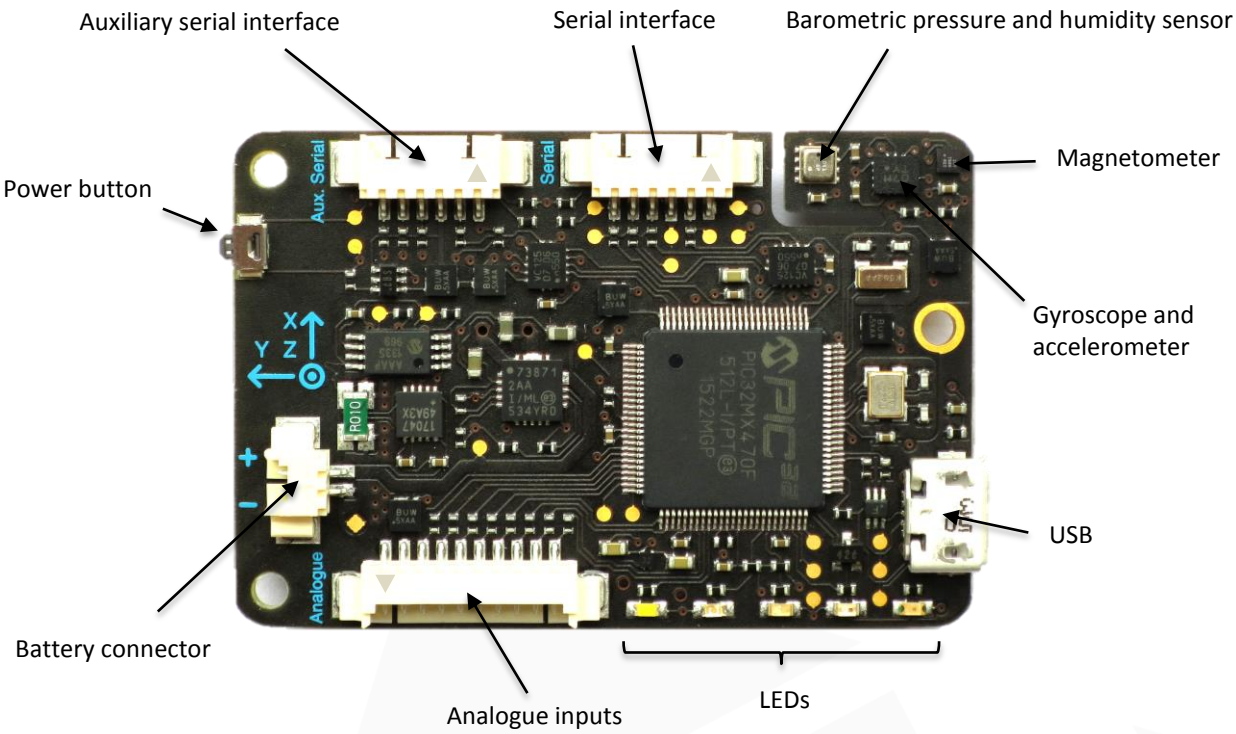

Figure 1: Top view of board

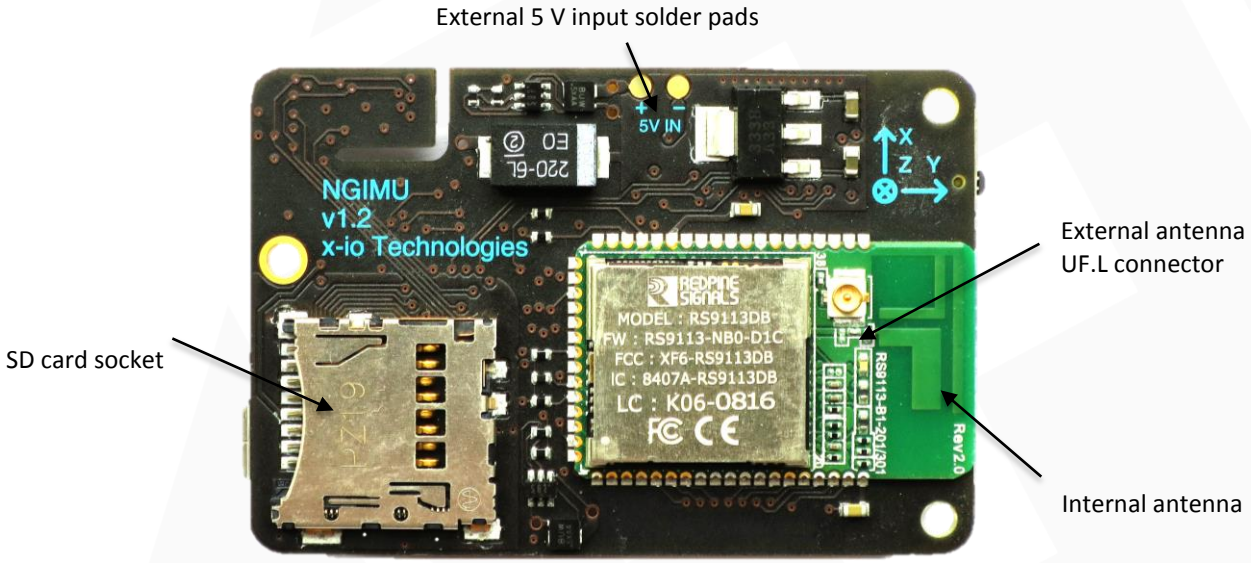

Figure 2: Bottom view of board

2.1. Power button

The power button is primarily used to turn the device on and off (sleep mode). Pressing and holding the button for half a second while the device is off will turn it on. Pressing and holding the button for 3 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 (`/button`) 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.

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	-	-
Red	Battery charging	**Off** - Charger not connected
		Solid - Charger connected and charging in progress
		Flashing (0.3 Hz) - Charger connected and charging complete

Table 1: LED behaviour

Sending an 'identify' message (`/identify`) 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.

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 input pinout

Table 4 lists the analogue input 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

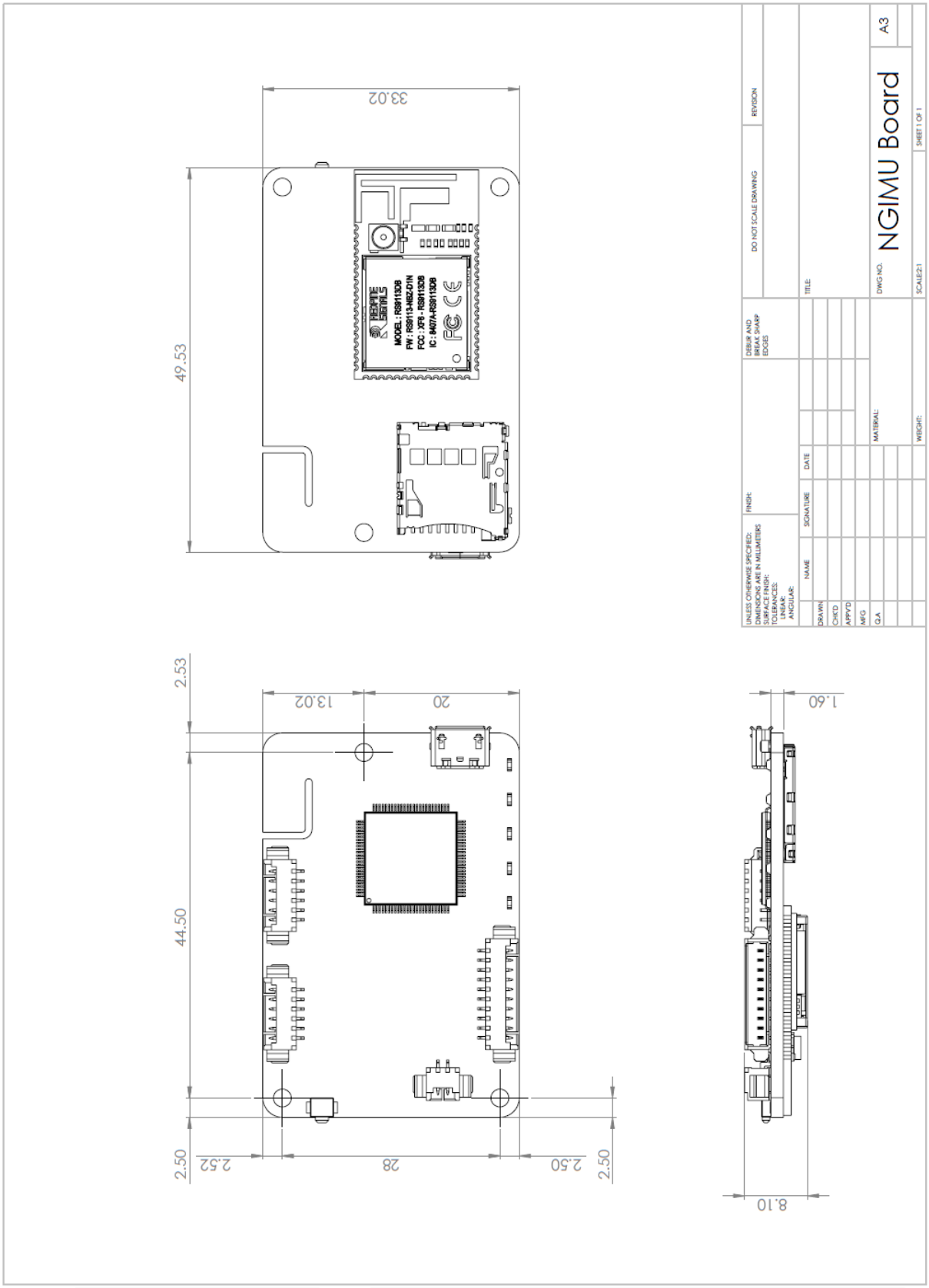

Figure 3: Mechanical drawing of the board

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 4 shows the board assembled with 1000 mAh battery in plastic housing. Figure 5 is a mechanical drawing of the plastic housing indicating all dimensions.

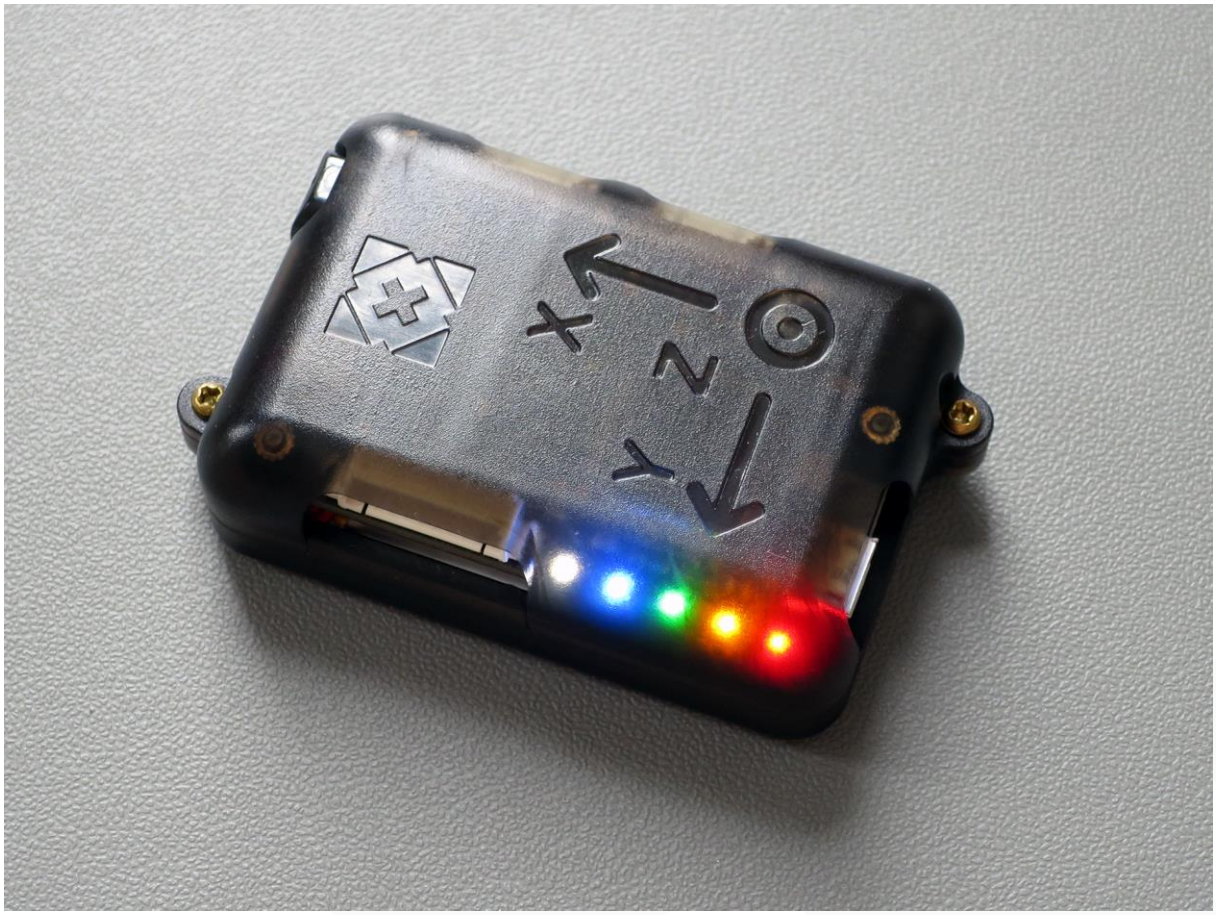

Figure 4: Board assembled with 1000 mAh battery in plastic housing

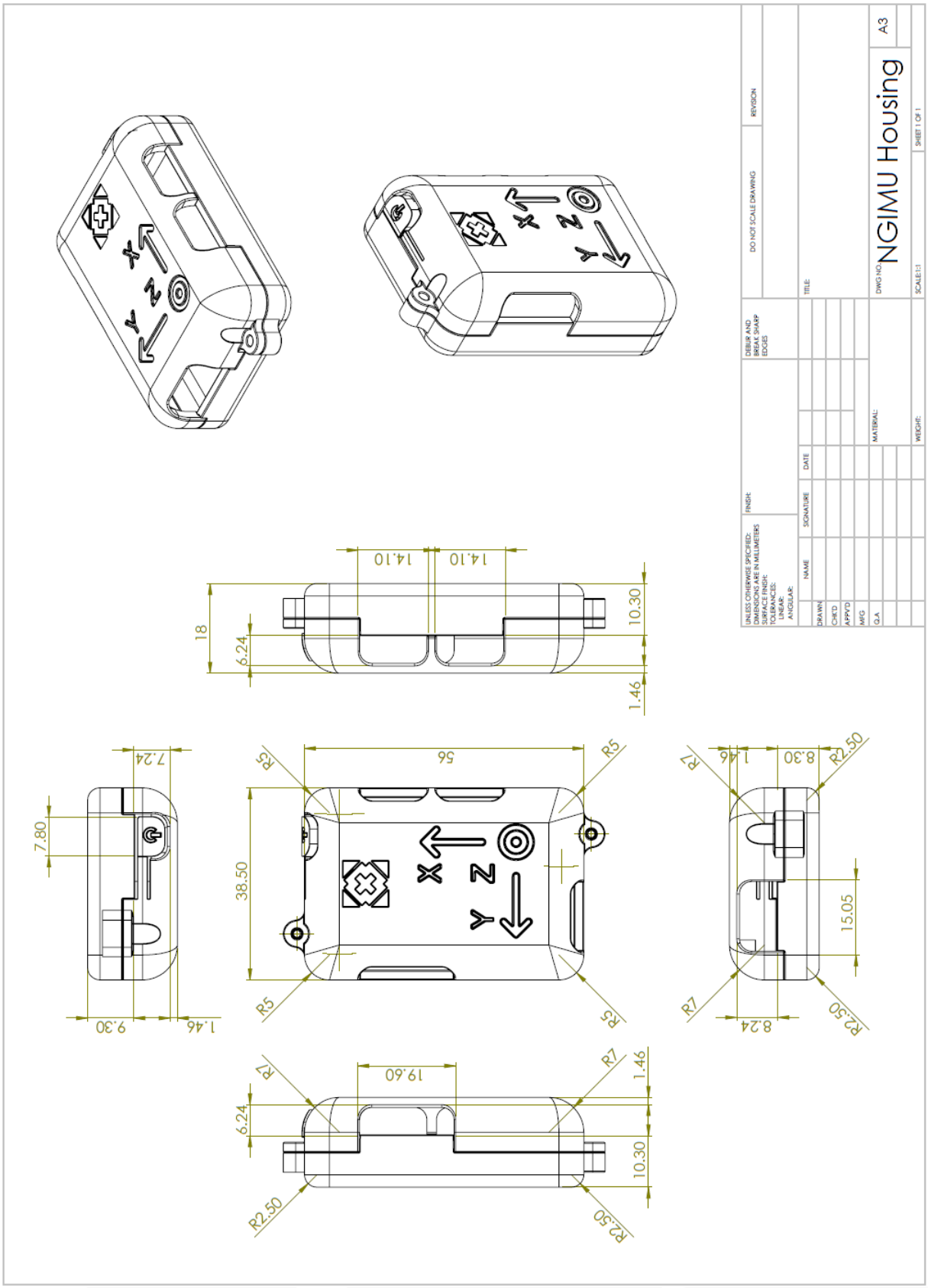

Figure 5: Mechanical drawing of the plastic housing

4. 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.

4.1. 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 value describing the date and time as the number of seconds since 00:00 on January 1st, 1990.

4.1.1. Button message

OSC address: `/button`

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

4.1.2. Sensors

OSC address: `/sensors`

The sensor message contains measurements from the gyroscope, accelerometer, magnetometer, and barometer. The message arguments are summarised in Table 6.

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 6: Sensor message arguments

4.1.3. Magnitudes

OSC address: `/magnitudes`

The magnitudes message contains measurements of the gyroscope, accelerometer, and magnetometer magnitudes. The message arguments are summarised in Table 7: 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 7: Magnitudes message arguments

4.1.4. 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 8.

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

Table 8: Quaternion message arguments

4.1.5. 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 9.

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 9: Rotation matrix message arguments

4.1.6. 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 10.

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 10: Euler angle message arguments

4.1.7. 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 11.

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

Table 11: Linear acceleration message arguments

4.1.8. 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 12.

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

Table 12: Earth acceleration message arguments

4.1.9. Altitude

OSC address: `/altitude`

The altitude message contains the measurement of altitude above sea level. The message argument is summarised in Table 13.

Argument	Type	Description
1	float32	Altitude above sea level in m

Table 13: Altitude message argument

4.1.10. Temperature

OSC address: `/temperature`

The temperature message contains the measurements from each of the devices on-board temperature sensors. The message arguments are summarised in Table 14.

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

Table 14: Temperature message arguments

4.1.11. Humidity

OSC address: `/humidity`

The humidity message contains the relative humidity measurement. The message argument is summarised in Table 15.

Argument	Type	Description
1	float32	Relative humidity in %

Table 15: Humidity message argument

4.1.12. 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 16.

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 16: Battery message arguments

4.1.13. Analogue inputs

OSC address: `/analogue`

The analogue inputs message contains measurements of the analogue inputs voltages. The message arguments are summarised in Table 17.

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 17: Analogue inputs message arguments

4.1.14. 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 18.

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 18: RSSI message argument

4.1.15. 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 19.

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 19: Auxiliary serial data message argument

4.1.16. 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 20.

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

Table 20: Auxiliary serial CTS input message argument

4.1.17. 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 21.

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

Table 21: Serial CTS input message argument

4.2. Data to device

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

4.2.1. 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 22.

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

Table 22: Auxiliary serial data message arguments

4.2.2. 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 23.

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

Table 23: Auxiliary serial RTS output message arguments

4.2.3. 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 24.

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

Table 24: Serial RTS output message arguments

4.3. 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.

4.3.1. Set time

OSC address: `/time`

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

4.3.2. Mute

OSC address: `/mute`

The mute command inhibits the sending of all data messages listed in section 4.1. Command confirmation messages and setting read/write response messages will still be sent. The device will remain muted until an unmute command is sent.

4.3.3. Unmute

OSC address: `/unmute`

The unmute command will undo the mute state described in section 4.3.2.

4.3.4. 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.

4.3.5. 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.

4.3.6. 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.

4.3.7. 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.

4.3.8. Restore default

OSC address: `/default`

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

4.3.9. AHRS initialise

OSC address: `/ahrs/initialise`

The AHRS initialise command will reinitialise the AHRS algorithm.

4.3.10. 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.

4.3.11. Echo

OSC address: `/echo`

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

4.4. 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.

4.4.1. 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.

4.4.2. 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.

4.5. Errors

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

