
# Thetis User Manual

v1.0

October 31, 2023

Braidan Duffy

# Contents

## Acknowledgements

I would like to acknowledge and thank Dr. Sebastian Madgwick, his company xio-Technologies, and his staff for their efforts in developing small embedded data loggers. The x-IMU3 became a major inspiration when it was announced earlier in 2023 and Seb was willing to chat with me about Thetis's design and its flaws. He has also been open to me using the x-IMU3 API on my board and the x-IMU3 user manual as a basis for this manual. His contributions and advice directly influenced version 2.0 of the firmware and the hardware revision F6. I am grateful for his correspondence and look forward to continuing our discussions on a variety of ideas.

# 1 Overview

Thetis is a nine Degree of Freedom (DOF) Inertial Measurement Unit (IMU) that is capable of reporting the acceleration, rotation rate, orientation, and position of a body. It was developed as part of a thesis project for the Ocean Engineering Department at the Florida Institute of Technology. The device is designed to be used in classroom and laboratory environments or out in the field, wherever inertial data logging is desired.

Thetis can communicate with a host computer over a USB or Wi-Fi connection, streaming data in real-time to an operator. For more embedded applications, Thetis has an on-board battery and microSD card storage, allowing it to remotely data log without an operator. A simple user interface allows for a single operator to deploy Thetis in any condition.

**Sensors**

- Gyroscope, ±2000 °/s, 100 Hz
- Accelerometer, ±32 g, 100 Hz
- Magnetometer, ±16 gauss, 100 Hz

**Calibration**

- 15-parameter calibration for: axis sensitivity, axis offset, inter-axis misalignment, and package misalignment.
- Hard-iron and soft-iron calibration
- On-board gyroscope bias correction algorithm

**AHRS**

- Algorithm outputs:
  - Quaternion
  - Rotation matrix[1]
  - Euler angles
  - Linear acceleration[1]
  - Earth acceleration[1]
- Linear acceleration rejection[1]
- Magnetic distortion rejection[1]
- 100 Hz update rate
- Static accuracy:
  - 0.5 ° RMS inclination[2]
  - 1 ° RMS heading[2]

**Communication**

- USB (CDC)
- TCP (Wi-Fi)[1]
- UDP (Wi-Fi)

**Wi-Fi**

- Client and AP mode
- Single band (2.4 GHz)
- WPA/WPA2-Personal

**Data logging**

- Supports microSDs up to 32 GB[3]
- Start/stop logging remotely[1]
- Start/stop logging with physical button
- USB download[4]
- Wi-Fi download[5]
- CSV output

**Battery**

- Internal battery charged by USB
- 9 hours data logging[6]
- 4 hours Wi-Fi client[7]

**Housing**

- IP67
- Chassis mount via two M3 screws or bolts

**Software GUI**

- Connect to multiple xio-compatible IMUs (e.g. x-IMU3, Thetis)
- Real-time data graphs and 3D view
- Log data to CSV
- Windows, macOS, Ubuntu

**Software API**

- Rust, C, C++, C#, Python
- Code examples for other languages available

[1] Supported, but not yet implemented.
[2] Remains to be validated.
[3] The product is supplied with an 4 GB microSD that can be upgraded by the user.
[4] USB downloading is currently in development and not yet supported.
[5] Wi-Fi downloading is currently in development and not yet supported.
[6] Estimated
[7] Estimated, assuming constant transmission.

# 2 Hardware

## 2.1 Board

Board components are annotated in Figure 1. A detailed mechanical drawing describing the board dimensions is available in the Appendix.

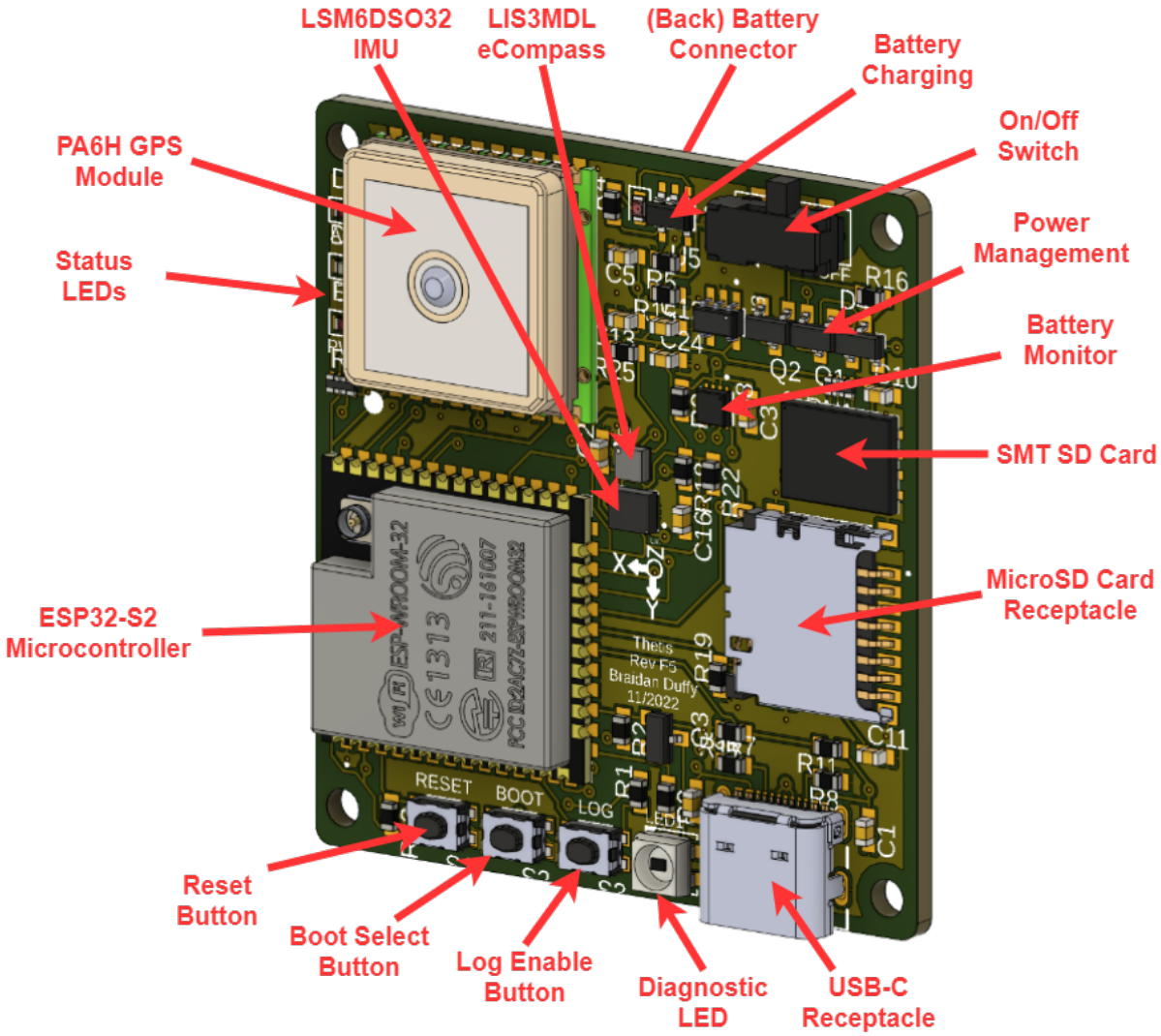



Figure 1: Revision F5 of Thetis with callouts for important components

### 2.1.1 Interface Pads

The PCB interface is on the back of the board beneath the EPS32-S2 microcontroller. Its pinout is annotated in Figure 2. The interface pads are just bare copper that are routed to their appropriate components. This allows for easier programming, diagnostics, and probing of the board during assembly and testing. It is recommended to use a carrier board and pogo pins to break these pads out for use. A mechanical drawing for the pad layout is in the Appendix.

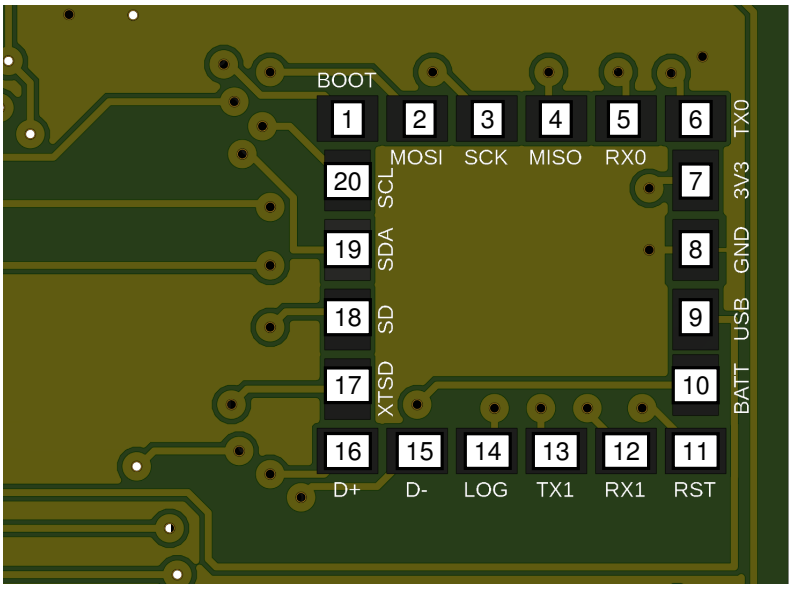



Figure 2: PCB interface pinout

| Pad | Name | Bus | Type |
|---|---|---|---|
| 1 | Boot Select | GPIO | Input |
| 2 | MOSI | SPI | Output |
| 3 | SCK | SPI | Clock |
| 4 | MISO | SPI | Input |
| 5 | RX0 | UART0 | Input |
| 6 | TX0 | UART0 | Output |
| 7 | 3V3 | | Power |
| 8 | GND | | Power |
| 9 | VUSB | | Power |
| 10 | VBATT | | Power |

Table 1: PCB interface pinout

| Pad | Name | Bus | Type |
|---|---|---|---|
| 11 | Reset | GPIO | Input |
| 12 | RX1 | UART1 | Input |
| 13 | TX1 | UART1 | Output |
| 14 | Log | GPIO | Input |
| 15 | D- | USB | Data |
| 16 | D+ | USB | Data |
| 17 | XTSD CS | SPI | Output |
| 18 | microSD CS | SPI | Output |
| 19 | SDA | I2C | Data |
| 20 | SCL | I2C | Clock |

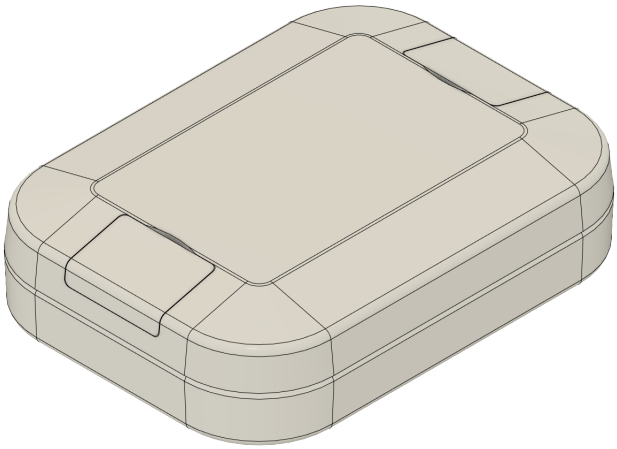
Table 2: PCB interface pinout

2

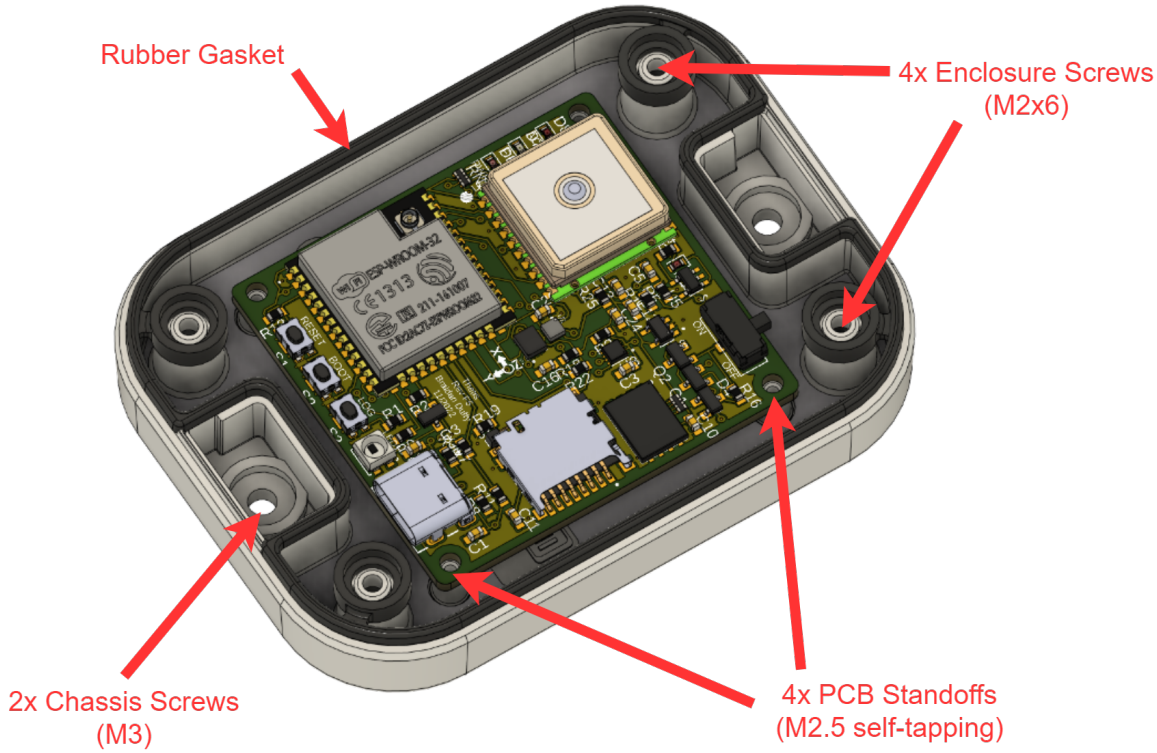
**Warning** - Incorrect connections to the PCB interface may cause permanent damage. This interface should only be used by an experienced engineer with the appropriate carrier board.

## 2.2 Housing

The housing components are annotated in Figure 3. A detailed mechanical drawing describing the housing dimensions is available in the Appendix.



(a) Sealed enclosure ready for deployment




(b) Enclosure with the top removed

Figure 3: Thetis enclosure

### 2.2.1 IP67 rating

The Ingress Protection 67 (IP67) rating is an international standard that describes the ability of the housing to protect against the ingress of solid particles and water. The first digit, 6 indicates complete protection against dust and solid particles. The second digit, 7 indicates protection from water for a maximum depth of 1 meter for up to 30 minutes.

In practical terms, this means that the housing can be used outdoors in all weather conditions and that it will survive accidental or temporary submersion in water. The housing has been tested in submerged applications up to 20-feet for 30 minutes, however typical applications should not exceed the IP rating. Always check that the gasket is in place, the 4 enclosure screws are suitably tightened, and the case is not cracked.

**Warning** - Over-torquing the enclosure screws may result in the plastic around them cracking. The enclosure screws should be finger tight with an extra quarter turn. Additionally, all four screws must be present for the case to be sealed - any less and the enclosure will leak when submerged!

# 3 Technical Specification

## 3.1 Mechanical

### 3.1.1 Board

| Characteristic | Value | Notes |
|---|---|---|
| Size | 1.82 × 1.74 × 0.48 in | 1 |
| Weight | 3.2 g | - |

Table 3: Board mechanical specification

**Notes**

1. A detailed mechanical drawing describing the board dimensions and locations of key components is available in the Appendix.

| Characteristic | Value | Notes |
|---|---|---|
| Size | 3.15 × 2.36 × 0.79 in | 1 |
| Weight | 50 g | - |

Table 4: Housing mechanical specification

**Notes**

1. A detailed mechanical drawing describing the housing dimensions and locations of key components is available in the Appendix.

## 3.2 Temperature

### 3.2.1 No battery

| Characteristic | Value | Notes |
|---|---|---|
| Operating | -40℃ to 85℃ | 1, 2 |
| Storage | -40℃ to 105℃ | - |

Table 5: Temperature specification (no battery)

**Notes**

1. The operating temperature of the device will always be greater than the surroundings due to heat generated by electronics.
2. The specified accuracy of the device is not achieved over the full operating temperature range. See Section 4 on page 15 for more information.

### 3.2.2 With battery

| Characteristic | Value | Notes |
|---|---|---|
| Operating (discharging) | -20 ℃ to 60 ℃ | 1, 2 |
| Operating (charging) | 0 ℃ to 45 ℃ | 1, 2, 3 |
| Storage | -20 ℃ to 25 ℃ | - |

Table 6: Temperature specification (with battery)

**Notes**

1. The operating temperature of the device will always be greater than the surroundings due to heat generated by electronics.
2. The specified accuracy of the device is not achieved over the full operating temperature range. See Section 4 on page 15 for more information.
3. Charging at temperatures below 0 ℃ will reduce the capacity and cycle life of the battery.

## 3.3 Sensors

### 3.3.1 Gyroscope

| Characteristic | Value | Notes |
|---|---|---|
| Range | ±2000 °/s | - |
| Resolution | 16-bit, 0.061 °/s | - |
| Sample rate | 100 Hz ±0.3% | 1 |

Table 7: Gyroscope specification

**Notes**

1. Each sample includes a timestamp for a reliable measurement of time independent of the sample rate error. See Section 8 on page 37 for more information.

### 3.3.2 Accelerometer

| Characteristic | Value | Notes |
|---|---|---|
| Range | ±32 g | - |
| Resolution | 16-bit, 488 µg | - |
| Sample rate | 100 Hz ±0.3% | 1 |
| Accuracy at 1 g | ±5 mg | 2, 3 |

Table 8: Accelerometer specification

**Notes**

1. Each sample includes a timestamp for a reliable measurement of time independent of the sample rate error. See Section 8 on page 37 for more information.
2. The accelerometer error is evaluated as the deviation of the measured magnitude of gravity for a 360° rotation around each axis aligned to the horizontal. The magnitude is calculated as $\sqrt{x^2 + y^2 + z^2}$.

3. Accuracy is specified for the calibrated temperature only. See Section 4 on the next page for more information.

#### 3.3.3 Magnetometer

| Characteristic | Value | Notes |
|---|---|---|
| Range | ±16 Gauss | - |
| Sample rate | 80 Hz ±8% | 1 |
| Noise | Unknown | - |
| Accuracy at 1 a.u. | Unknown | 2, 3, 4 |

Table 9: Magnetometer specification

**Notes**

1. Each sample includes a timestamp for a reliable measurement of time independent of the sample rate error. See Section 8 on page 37 for more information.
2. The calibrated magnetometer units are arbitrary units (a.u.). 1 a.u. is equal to the magnitude of the ambient magnetic field during calibration, approximately 45 µT.
3. The magnetometer error is evaluated as the deviation of the measured magnitude of the ambient magnetic field for a 360° rotation around each axis aligned to the vertical. The magnitude is calculated as $\sqrt{x^2 + y^2 + z^2}$.
4. Accuracy is specified for the calibrated temperature only. See Section 4 on the next page for more information.

### 3.4 AHRS

#### 3.4.1 Update rate

| Characteristic | Value | Notes |
|---|---|---|
| Update rate | 64 Hz ±0.3% | 1, 2 |

Table 10: AHRS update rate

**Notes**

1. Each update includes a timestamp for a reliable measurement of time independent of the update rate error. See Section 8 on page 37 for more information.
2. The Attitude Heading Reference System (AHRS) update rate is fixed independent of message rate settings. AHRS outputs are not averaged when the message rate is less than the update rate.

#### 3.4.2 Static accuracy

| Characteristic | Value | Notes |
|---|---|---|
| Inclination | 0.5° RMS | 1, 2 |
| Heading | 1° RMS | 1, 2 |

Table 11: AHRS static accuracy

**Notes**

1. Static accuracy is specified as the Root Mean Square (RMS) error for a 360° rotation around each axis.
2. Accuracy is specified for the calibrated temperature only. See Section 4 for more information.

### 3.5 Data logger capacity

The data logger capacity can be determined with the following equation:

$$t_{\text{samples}} = \frac{N_{\text{storage}}[\text{Bytes}]}{198[\text{Bytes}] \times 64[\text{s}^{-1}] \times 3600\left[\frac{\text{s}}{\text{h}}\right]} \tag{1}$$

This yields the following times the data logger can record for with a given microSD card size:

| Size | Value |
|---|---|
| 1 GB | 22 hours |
| 4 GB | 88 hours |
| 8 GB | 175 hours |
| 16 GB | 350 hours |
| 32 GB | 701 hours |

Table 12: Data logger record times based on capacity of the microSD card

## 4 Calibration

Each Thetis instrumentation board is calibrated during production to achieve the specified accuracy. The calibration process uses custom equipment and open-source algorithms to calculate calibration parameters specific to each Thetis board. See Chapter 4 in the thesis for more information. Calibration is performed at room temperature. Accuracy will be reduced for operating temperatures that deviate from this temperature.

### 4.1 Inertial sensors

The inertial sensors are the gyroscope and accelerometer. Each inertial sensor is calibrated for axis sensitivity, axis offset, inter-axis misalignment, and package misalignment. The inertial calibration model is described by Equation (2) where $\boldsymbol{i}_c$ is the calibrated inertial measurement obtained from the uncalibrated inertial measurement, $\boldsymbol{i}_u$, given the misalignment matrix, $M$, the sensitivity diagonal matrix, $S$, and the offset vector, $\boldsymbol{b}$. The inertial calibration model is expanded as Equation (3) to express the model as 15 scalar quantities. The units of $\boldsymbol{i}_c$, $\boldsymbol{i}_u$, and $\boldsymbol{b}$ are degrees per second for the gyroscope, and g for the accelerometer. $M$ and $S$ are ratios and therefore have no units. The calibration parameters $M$, $S$, and $\boldsymbol{b}$ for each inertial sensor can be accessed as device settings.

$$\boldsymbol{i}_c = MS(\boldsymbol{i}_u - \boldsymbol{b}) \tag{2}$$

$$\begin{bmatrix} i_{cx} \\ i_{cy} \\ i_{cz} \end{bmatrix} = \begin{bmatrix} m_{xx} & m_{xy} & m_{xz} \\ m_{yx} & m_{yy} & m_{yz} \\ m_{zx} & m_{zy} & m_{zz} \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \left( \begin{bmatrix} i_{ux} \\ i_{uy} \\ i_{uz} \end{bmatrix} - \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \right) \tag{3}$$

### 4.2 Magnetometer

The magnetometer is calibrated for soft and hard iron distortion. Soft iron characteristics are distortions that alter the intensity and direction of the magnetic field measured by the magnetometer. Soft iron calibration also accounts for magnetometer axis sensitivity, inter-axis misalignment, and package misalignment. Hard

iron characteristics are unintended magnetic fields generated by the device that offset magnetometer measurements. Hard iron calibration also accounts for the magnetometer axis offset.

The magnetometer calibration model is described by Equation (4) where $\boldsymbol{m}_c$ is the calibrated magnetometer measurement obtained from the uncalibrated magnetometer measurement, $\boldsymbol{m}_u$, given the soft iron matrix, $W^{-1}$, the hard iron vector, $\boldsymbol{v}$. The magnetometer calibration model is expanded as Equation (5) to express the model as 12 scalar quantities. The units of $\boldsymbol{m}_c$, $\boldsymbol{m}_u$, and $\boldsymbol{v}$ are a.u.. $W^{-1}$ is a ratio and therefore has no units. The calibration parameters $W^{-1}$, $\boldsymbol{v}$ can be accessed as device settings.

$$\boldsymbol{m}_c = W^{-1}\boldsymbol{m}_u - \boldsymbol{v} \tag{4}$$

$$\begin{bmatrix} m_{c,x} \\ m_{c,y} \\ m_{c,z} \end{bmatrix} = \begin{bmatrix} w_{xx} & w_{xy} & w_{xz} \\ w_{yx} & w_{yy} & w_{yz} \\ w_{zx} & w_{zy} & w_{zz} \end{bmatrix} \begin{bmatrix} m_{u,x} \\ m_{u,y} \\ m_{u,z} \end{bmatrix} - \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{5}$$

# 5 Human Interfaces

There are multiple human interface components on the Thetis board, as shown in Figure 4.

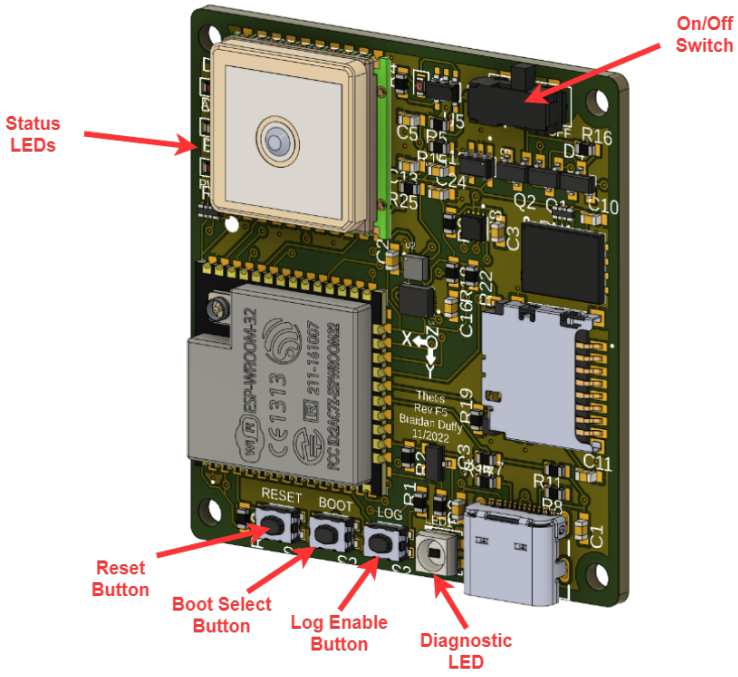



Figure 4: Thetis Revision F5 with callouts for human interface components

## 5.1 Buttons

Along the bottom edge are three buttons: reset, boot select, and log enable.

**The reset button** will perform a power-on reset of the microcontroller when pressed, restarting the firmware and clearing any errors.

**The boot select button** will put the microcontroller into the bootloader (safe/programming) mode if pressed during a reset or power cycle.

**The log enable button** will start or stop logging when pressed for half of a second. The current logging status is shown by the LEDs.

## 5.2 Primary RGB LEDs

The main diagnostic Red Green Blue (RGB) Light-Emitting Diode (LED) indicates the mode and status of Thetis using different colors and flashing behaviors.

### 5.2.1 Booting (Purple)

A steady purple LED, as shown in Figure 5 indicates that Thetis is switched on and booting.

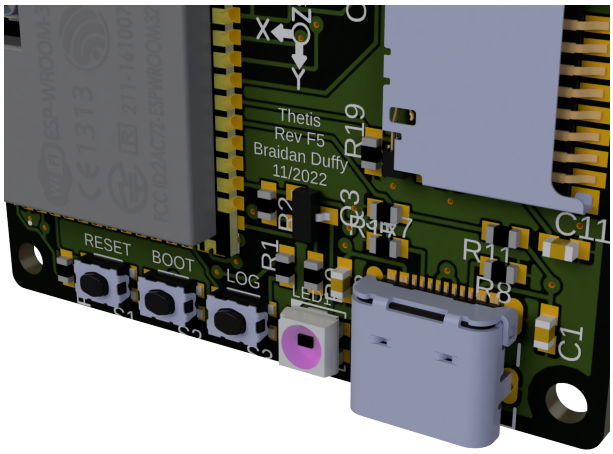



Figure 5: a steady purple LED indicating that Thetis is switched on and booting

### 5.2.2 Standby (Yellow)

An steady yellow LED, as shown in Figure 6 indicates that Thetis is switched on and in standby mode. This mode is active until the device passes a self-test, its sensors report that they are calibrated, and the fusion algorithm is online and stable. Currently, this mode is bypassed and ignored.

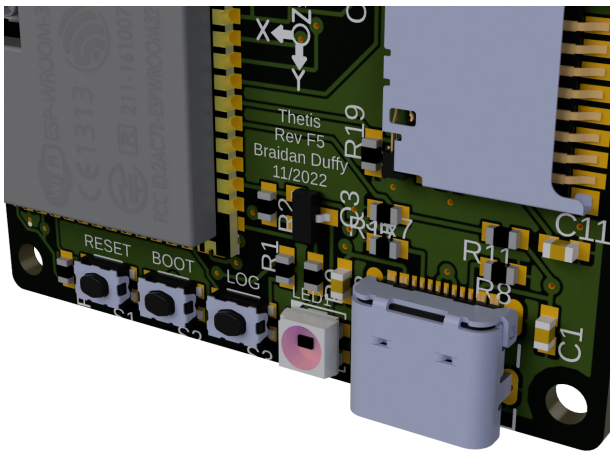
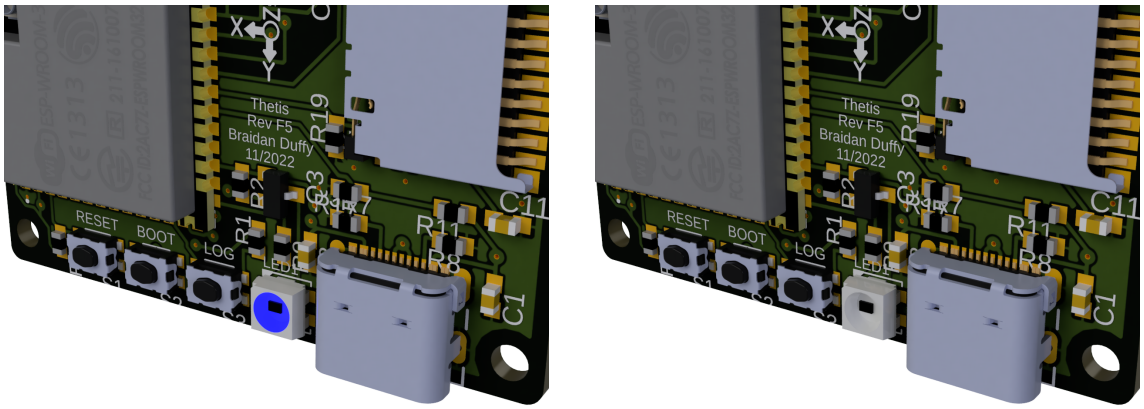



Figure 6: A steady yellow LED indicating that Thetis is switched on and in standby mode

### 5.2.3 Ready, No GPS (Blue)

A blinking (once per second) blue LED, as shown in Figure 7 on the next page indicates that Thetis is online, calibrated, and ready to start logging. However, it has not yet acquired a GPS fix. This mode will still allow logging, but the position messages will either not be populated or not accurate.




(a) The LED is blue for approximately 250 milliseconds

(b) Then turns off for 250 milliseconds

Figure 7: A blinking blue LED indicating that Thetis is switched on and ready to log data without a GPS fix

### 5.2.4 Logging, No GPS (Blue)

A steady blue LED, as shown in Figure 8 indicates that Thetis is switched on and logging data without a GPS fix. This mode will still log data as expected, but the position messages will either not be populated or not accurate.

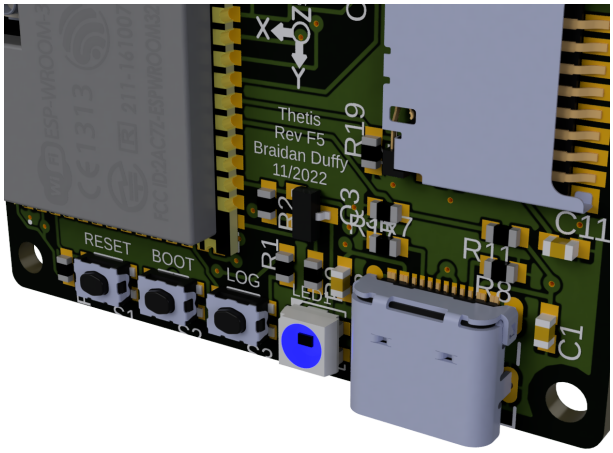
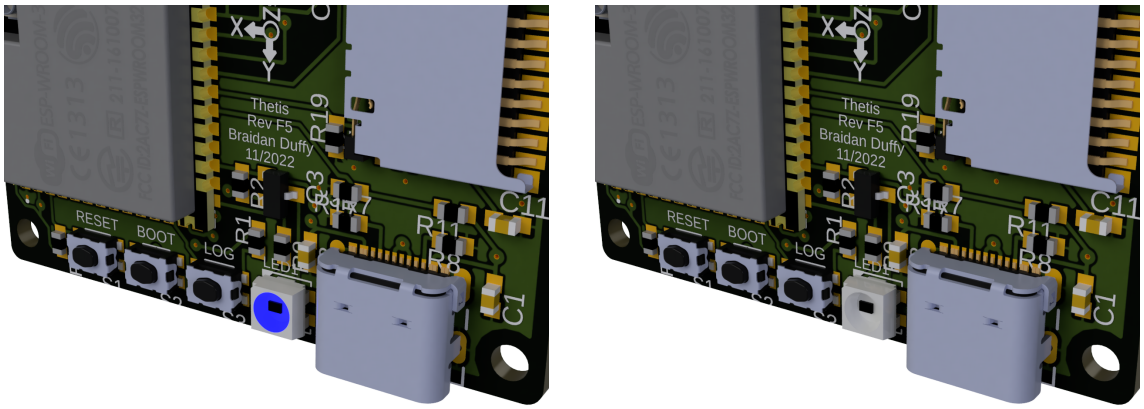



Figure 8: A steady blue LED indicating that Thetis is switched on and logging without a GPS fix

### 5.2.5 Ready, GPS (Green)

A blinking (once per second) green LED, as shown in Figure 9 on the next page indicates that Thetis is online, calibrated, and ready to start logging with a valid GPS fix. This will populate the position message with reasonably-accurate GPS data.




(a) The LED is green for 250 milliseconds

(b) Then turns off for 250 milliseconds

Figure 9: A blinking green LED indicating that Thetis is switched on and ready to log data with a GPS fix

### 5.2.6 Logging, GPS (Green)

A steady green LED, as shown in Figure 10 indicates that Thetis is switched on and logging data with a GPS fix. Position messages will be populated with reasonably-accurate GPS data in this mode.

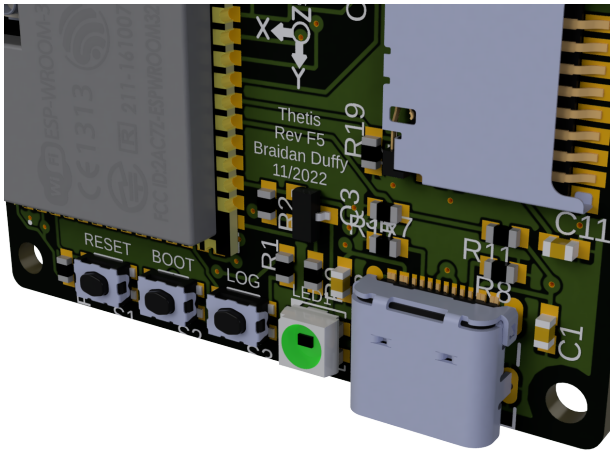
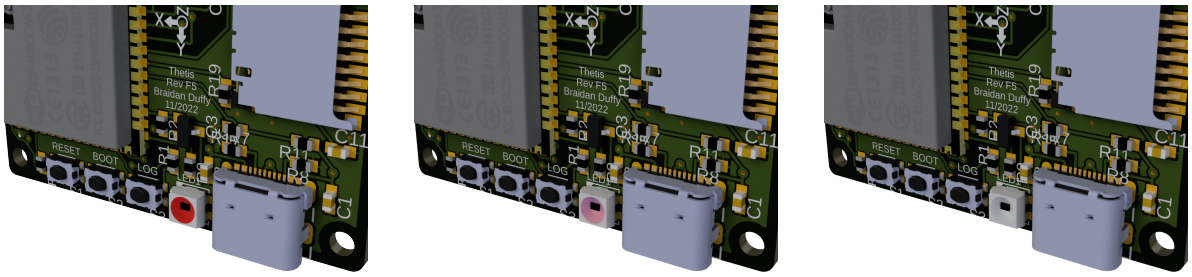



Figure 10: A steady green LED indicating that Thetis is switched on and logging with a GPS fix

### 5.2.7 Error (Red and Yellow)

When Thetis enters an error state, the main RGB LED will begin flashing red and yellow error codes based on the error type. The red blinks will precede the yellow blinks and there will be a short interval between code flashes. In the error state, Thetis will lock up and lose all functionality until the battery dies. The device will require a full power cycle or reset to clear the error code. In-depth diagnostic information may be available if data is being streamed to a host computer through the USB port and serial monitor terminal (e.g. PuTTY).




(a) The LED is red with 250 millisecond blinks for a certain number of times

(b) Then turns yellow with 250 millisecond blinks for a certain number of times

(c) Then turns off for one second

Figure 11: A blinking red and yellow LED indicates that Thetis is switched on and in an error state according to Table 13

| Error | Red | Yellow | Description |
|---|---|---|---|
| General | 1 | 1 | General failure state; unhandled or unknown error |
| Settings | 1 | 2 | Board is initialized with or encounters invalid settings |
| Low Battery | 2 | 1 | Battery voltage is below the threshold limit and the system has not automatically shutdown |
| Battery Monitor | 2 | 2 | The battery monitoring IC fails to initialize |
| Temperature | 2 | 3 | The reported system temperature exceeds a threshold[8] |
| Card Mount | 3 | 1 | The data log filesystem fails to mount to the microSD card |
| Card Type | 3 | 2 | The data log filesystem initializes, but reports an incorrect microSD card type or format |
| File Operations | 3 | 3 | A requested filesystem operation fails to execute |
| Wi-Fi Radio | 4 | 1 | The Wi-Fi radio fails to initialize or encounters errors |
| BlueTooth Radio | 4 | 2 | The BlueTooth radio fails to initialize or encounters errors[8] |
| GPS Radio | 4 | 3 | The GPS radio fails to initialize or encounters an error |
| IMU | 5 | 1 | The IMU fails to initialize or encounters an error |
| Magnetometer | 5 | 2 | The magnetometer fails to initialize or encounters an error |
| Fusion | 5 | 3 | The sensor fusion algorithm encounters an error[8] |
| CANbus | 6 | 1 | The CANbus transceiver fails to initialize or encounters an error[8] |
| Device ID | 6 | 2 | The device ID is not properly configured[8] |

Table 13: Thetis error code table

[8] not currently implemented

### 5.2.8 User control

The on-board RGB LED can be controlled by the user using the strobe and colour commands. See Section 7.1.14 on page 25 and Section 7.1.15 on page 25 for more information.

## 5.3 Secondary Diagnostic LED

In addition to the main RGB LED, Thetis also has four other monochromatic diagnostic LEDs on the left and top edge of the board. From top to bottom, these LEDs are for battery charging, activity, GPS fix, and power. Their locations are highlighted in Figure 12

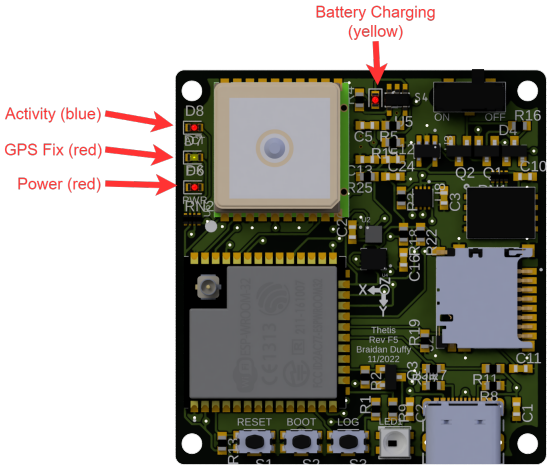



Figure 12: Additional monochromatic diagnostic LEDs present on Thetis

**The battery charging LED** illuminates orange when the battery is plugged in, Universal Serial Bus (USB) is connected, and the battery is not at 4.2V. The board does not need to be on to charge the battery.

**The activity LED** is blue and typically only illuminated when the device is logging. This behavior can be changed in the firmware.

**The Global Positioning System (GPS) fix LED** blinks red once per second when the GPS is acquiring a signal. Once the signal is acquired, it blinks quickly once every 15 seconds.

**The power LED** illuminates red when the 3V3 bus on the board is active; nominally when the battery or USB is plugged in and the switch turned on.

# 6 Data logger

Thetis can function as a stand-alone data logger by streaming real-time data to a file on the micro Secure Digital (microSD). Files created by the data logger use the .bin extension and can be downloaded from the device to be converted to Comma-Separated Values (CSV) files using the product software.

The data logger will create a new file in the root directory on the microSD each time the device boots. Files will never be overwritten or deleted by the data logger. If the microSD becomes full then the data logger will stop and Thetis will indicate an error.

## 6.1 Start and stop

The data logger is started or stopped by the “Log Enable” button (See Section 5.1 on page 16 for more information.). As soon as the button is held for half of a second, the device will immediately begin logging. When the operator wishes to stop logging, the “Log Enable” button can be held again for another half second or until the “Activity” LED and RGB LED reflect the “Ready” state.

**Warning** - The data file is not written to the microSD card filesystem until the device is commanded to stop logging. DO NOT reset or power off the device when logging, otherwise the data may not be saved or will be corrupted!

### 6.2 File name

The file name format is “log_CCC.bin” where “CCC” is a counter. The counter runs between “000” and “999”. On startup, Thetis scans the microSD card and increments the counter until the next available number is encountered. When it exceeds its maximum value, the system will throw a filesystem initialization error. Even if the device is powered on, but does not log, a new log file will be created. In testing environments, this may lead to multiple empty files that can obscure an actual desired data file.

### 6.3 File contents

The contents of the file is a byte stream as per the communication protocol described in Section 7.

## 7 Communication protocol

All communication interfaces use the same communication protocol. The byte stream is therefore identical for USB, User Datagram Protocol (UDP), and the files created by the data logger. The communication protocol consists of two message types:

- Command messages
- Data messages

All messages are terminated by a Line Feed (LF) control character. This termination byte will not appear anywhere else in a message and so can be used to divide a byte stream into individual messages. Table 14 describes the different ways that the control character LF may be referred to throughout this document.

| Control character | Abbreviation | String | Hex | Decimal |
|---|---|---|---|---|
| Line Feed | LF | “\n” | 0x0A | 10 |

Table 14: Control characters LF representations

The first byte of a message indicates the message type. Command messages start with the character “{” (0x7B in hex, 123 in decimal). Data messages start with either an uppercase character or a byte value greater than 0x80 (128 in decimal) depending on the message.

### 7.1 Command messages

Command messages are sent to Thetis to read and write settings and execute commands. All command messages are a JavaScript Object Notation (JSON) object containing a single key/value pair, terminated by the control character LF. The control character LF must not appear anywhere else in a command message. Thetis will acknowledge each received command message by sending a command message with the same key to the host.

The key used by command messages sent to Thetis **is case sensitive** and must be exactly as stated in the API. For example, “serialNumber” will work, but “Serial Number” and “serial_number” are not valid keys for a command message to read the device serial number.

#### 7.1.1 Read all settings command

The read all settings command is sent to Thetis to read all the setting values. The key is “readAll” and the value is null. See Section 10.1 on page 39 for a complete list of settings. Thetis will acknowledge a read all settings command by sending a write setting command to the host for each of its settings.

**Example:** {"readAll":null}\n

### 7.1.2 Read JSON file command

The read JSON file command is sent to Thetis to read out the settings file stored in its non-volatile memory. This file is a JSON file containing all the settings as key-value pairs. See Section 10.1 on page 39 for a complete list of settings. Thetis will acknowledge a read JSON file command by sending a stream reading out the entire contents of the settings file.

**Example:** {"readJSON":null}\n

### 7.1.3 Read setting command

The read setting command is sent to Thetis to read a setting value. The key is the setting key and the value is null. See Section 10.1 on page 39 for a complete list of settings. Thetis will acknowledge a read setting command by sending a write setting command to the host.

**Example:** {"serialNumber":null}\n

### 7.1.4 Write setting command

The write setting command is sent to Thetis to write a setting value, or sent from Thetis to the host in response to a read setting command. The key is the setting key and the value is the setting value. See Section 10.1 on page 39 for a complete list of settings. Thetis will acknowledge a write setting command by sending a setting write command back to the host, indicating the new settings value. Thetis will not apply new settings until two seconds after the most recent write setting command or default command was received.

**Example:** {"deviceName":"Thetis-000"}\n

### 7.1.5 Default command

The default command is sent to Thetis to set all settings to default values. The key is “default” and the value is null. Thetis will not apply new settings until two seconds after the most recent write setting command or default command was received.

**Example:** {"default":null}\n

### 7.1.6 Apply command

The apply command is sent to Thetis to apply all settings. The key is “apply” and the value is null. This command can be sent after a write setting or default command to apply settings immediately instead of after a two second delay.

**Example:** {"apply":null}\n

### 7.1.7 Save command

The save command is sent to Thetis to save all settings to Electrically Erasable Programmable Read-Only Memory (EEPROM). The key is “save” and the value is null. The command acknowledgement will not be sent until the save is complete. This may take up to 300 milliseconds. The save command is unnecessary in most applications because Thetis will automatically save all settings on shutdown.

**Example:** `{"save":null}\n`

### 7.1.8 Read time command

The read time command is sent to Thetis to read the date and time of the Real-Time Clock (RTC). The key is "time" and the value is null. Thetis will acknowledge a read time command by sending a write time command to the host.

**Example:** `{"time":null}\n`

### 7.1.9 Write time command

The write time command is sent to Thetis to write the date and time of the RTC, or sent from Thetis to the host in response to a read time command. The key is "time" and the value is a string expressing the date and time in the format "YYYY-MM-DD hh:mm:ss" where each delimiter can be any non-numerical character. Thetis will acknowledge a write time command by sending a write time command back to the host, indicating the new date and time.

**Example:** `{"time":"2020-01-01 00:00:00"}\n`

### 7.1.10 Ping command

The ping command is sent to Thetis to trigger a ping response. The key is "ping" and the value is null. Thetis will acknowledge a ping command by sending a ping response to the host.

**Example:** `{"ping":null}\n`

### 7.1.11 Ping response

The ping response is sent from Thetis to the host in response to the ping command. The key is "ping" and the value is a JSON object containing three key/value pairs indicating the communication interface, device name, and device serial number. The keys are "interface", "deviceName", and "serialNumber", respectively and all values are string types.

**Example*:**

```
{
  "ping": {
    "interface":     "USB",
    "deviceName":    "x-IMU3",
    "serialNumber":  "0123-4567-89AB-CDEF"
  }
}\n
```

* The actual JSON will not include any whitespace.

### 7.1.12 Reset command

The reset command is sent to Thetis to reset Thetis. The key is "reset" and the value is null. A reset is equivalent to switching Thetis off and then on again. Thetis will reset two seconds after receiving this command.

**Example:** `{"reset":null}\n`

#### 7.1.13 Shutdown command

The shutdown command is sent to Thetis to switch Thetis off. The key is “shutdown” and the value is null. Thetis will shutdown two seconds after receiving this command.

**Example:** `{"shutdown":null}\n`

#### 7.1.14 Strobe command

The strobe command is sent to Thetis to strobe the LED bright white for 5 seconds. The key is “strobe” and the value is null. This command can be used to quickly locate a specific x-IMU3 when using multiple x-IMU3s.

**Example:** `{"strobe":null}\n`

#### 7.1.15 Colour command

The colour command is sent to Thetis to set the LED colour. The key is “colour” or “color” and the value is either a RGB hex triplet expressed as a string, or null. Setting the colour will override the normal LED behaviour. A value of null will restore the normal behaviour.

**Example:** `{"colour":"FFFFFF"}\n`

#### 7.1.16 Initialise AHRS command

The initialise AHRS command is sent to Thetis to reinitialise the AHRS algorithm. The key is “initialise” or “initialize” and the value is null. This command in unnecessary for normal operation and typically only used during calibration and testing.

**Example:** `{"initialise":null}\n`

#### 7.1.17 Heading command

The heading command is sent to Thetis to set the heading of the orientation measurement provided by the AHRS algorithm. The key is “heading” and the value is a number equal to the heading in degrees. The heading command can only be used if the magnetometer is ignored in the AHRS settings.

**Example:** `{"heading":0}\n`

#### 7.1.18 Serial accessory command

The serial accessory command is sent to Thetis to transmit data to a serial accessory when the serial interface is in serial accessory mode. The key is “accessory” and the value is the data expressed as a string of up to 256 characters. Longer strings will be truncated to the maximum size. The string escape sequence “\u” can be used to express any byte value as per the JSON specification.

**Example:** `{"accessory":"hello \u0077\u006F\u0072\u006C\u0064"}\n`

#### 7.1.19 Note command

The note command is sent to Thetis to generate a timestamped notification message containing a user-defined string. The key is “note” and the value is the string of up to 127 characters. Longer strings

will be truncated to the maximum size. This command can be used to create timestamped notes of events during data logging.

**Example:** `{"note":"Something happened."}\n`

#### 7.1.20 Format command

The format command is sent to Thetis to format the microSD. The key is “format” and the value is null. The command acknowledgement will not be sent until the format is complete. This will take approximately 3 seconds for an 8 GB microSD. Larger SD microSDs will take longer to format. Formatting will erase all data on the SD card.

**Example:** `{"format":null}\n`

#### 7.1.21 Self-test command

The self-test command is sent to Thetis to perform a self-test. The key is “test” and the value is null. Thetis will acknowledge a self-test command by sending a self-test response to the host once the self-test is complete. This may take up to 5 seconds. Thetis must be stationary during the self-test.

**Example:** `{"test":null}\n`

#### 7.1.22 Self-test response

The self-test response is sent from Thetis to the host in response to the self-test command. The key is “test” and the value is a JSON object containing multiple key/value pairs. Each key/value pair indicates the result of a test. Each key is the test name and the value is the string ”Passed” if the test was passed.

**Example*:**

```
{
  "test": {
    "EEPROM":                "Passed",
    "RTC":                   "Passed",
    "Inertial":              "Passed",
    "Magnetometer":          "Passed",
    "High-g Accelerometer":  "Passed",
    "Battery":               "Passed",
    "SD Card":               "Passed",
    "Wireless":              "Passed",
  }
}\n
```

* The actual JSON will not include any whitespace.

#### 7.1.23 Bootloader command

The bootloader command is sent to Thetis to put Thetis in bootloader mode. The key is “bootloader” and the value is null. Thetis will enter bootloader mode two seconds after receiving this command.

**Example:** `{"bootloader":null}\n`

#### 7.1.24 Factory command

The factory command is sent to Thetis to enable factory mode. The key is “factory” and the value is null. In factory mode, read-only settings can be written using the write setting command and the erase command will be enabled.

**Example:** `{"factory":null}\n`

> **Warning** - Incorrect use of this command may permanently damage the device. Do not use this command unless instructed by customer support.

#### 7.1.25 Erase command

The erase command is sent to Thetis to erase the EEPROM. The key is “erase” and the value is null. The command acknowledgement will not be sent until the erase is complete. This will take approximately 700 milliseconds. This command can only be used if factory mode is enabled.

**Example:** `{"erase":null}\n`

> **Warning** - Incorrect use of this command may permanently damage the device. Do not use this command unless instructed by customer support.

### 7.2 Data messages

Data messages are sent from Thetis to the host to provide timestamped measurements, serial accessory data, notifications, and error messages. Data messages will be either American Standard Code for Information Interchange (ASCII) or binary, depending on the device settings.

ASCII data messages consist of multiple comma-separated values terminated by the control character character LF. The first value is a single uppercase character indicating the message type. The second value is the timestamp in microseconds. The remaining values are arguments specific to the message type.

Binary data messages are a sequence of bytes terminated by the control character LF. The first byte of the sequence indicates the message type. The value of this byte is equal to 0x80 plus the first character of the equivalent ASCII message. The next eight bytes are the timestamp in microseconds expressed as a 64-bit unsigned integer. The remaining bytes are arguments specific to the message type. Numerical types use little-endian ordering. Byte stuffing is used to remove all occurrences of the control character LF prior to the termination byte.

#### 7.2.1 Byte stuffing

Byte stuffing ensures that the termination byte value, 0x0A, only occurs at the end of a binary data message. This is achieved by replacing all occurrences of the termination byte prior to termination with an escape sequence. This process is identical to Serial Line Internet Protocol (SLIP) except that the “END” byte value is defined as 0x0A. Table 15 lists the values used by the byte stuffing process.

| Hex | Decimal | Name | Description |
|---|---|---|---|
| 0x0A | 10 | END | Message termination |
| 0xDB | 219 | ESC | Message escape |
| 0xDC | 220 | ESC_END | Transposed message termination |
| 0xDD | 221 | ESC_ESC | Transposed message escape |

Table 15: Values used by the byte stuffing process

Byte stuffing is achieved by the following:

- Replace each occurrence of END in the original message with the two byte sequence: ESC, ESC_END.
- Replace each occurrence of ESC in the original message with the two byte sequence: ESC, ESC_ESC.

The byte stuffing process will not modify the END that terminates the message. Table 16 demonstrates byte stuffing for example byte sequences terminated as binary data messages.

| Example | Before byte stuffing | After byte stuffing |
|---|---|---|
| 1 | 45 58 41 4D 50 4C 45 0A | 45 58 41 4D 50 4C 45 0A |
| 2 | 45 0A 41 4D 50 4C 45 0A | 45 DB DC 41 4D 50 4C 45 0A |
| 3 | 45 58 DB 4D 50 4C 45 0A | 45 58 DB DD 4D 50 4C 45 0A |
| 4 | 45 58 41 4D 50 DB 0A 0A | 45 58 41 4D 50 DB DD DB DC 0A |

Table 16: Byte stuffing examples

### 7.2.2 Inertial message

The inertial message provides timestamped gyroscope and accelerometer measurements. Inertial messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "I" and the arguments are six numerical values expressed to four decimal places. The first byte of a binary message is 0xC9 (equal to 0x80 + "I") and the arguments are six contiguous 32-bit floats. The message arguments are described in Table 17.

| Argument | Description |
|---|---|
| 1 | Gyroscope X axis in degrees per second |
| 2 | Gyroscope Y axis in degrees per second |
| 3 | Gyroscope Z axis in degrees per second |
| 4 | Accelerometer X axis in g |
| 5 | Accelerometer Y axis in g |
| 6 | Accelerometer Z axis in g |

Table 17: Inertial message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Gyroscope X axis = 0
2. Gyroscope Y axis = 0
3. Gyroscope Z axis = 0
4. Accelerometer X axis = 0
5. Accelerometer Y axis = 0
6. Accelerometer Z axis = 1

**ASCII example:** `I,1000000,0.0000,0.0000,0.0000,0.0000,0.0000,1.0000\n`

**Binary example:** `C9 40 42 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 3F 0A`

### 7.2.3 Magnetometer message

The magnetometer message provides timestamped magnetometer measurements. Magnetometer messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "M" and the arguments are three numerical values expressed to four decimal places. The first byte of a binary message is 0xCD (equal to 0x80 + "M") and the arguments are three contiguous 32-bit floats. The message arguments are described in Table 18.

| Argument | Description |
|---|---|
| 1 | Magnetometer X axis in a.u. |
| 2 | Magnetometer Y axis in a.u. |
| 3 | Magnetometer Z axis in a.u. |

Table 18: Magnetometer message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Magnetometer X axis = 1
2. Magnetometer Y axis = 0
3. Magnetometer Z axis = 0

**ASCII example:** `M,1000000,1.0000,0.0000,0.0000\n`

**Binary example:** `CD 40 42 0F 00 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 0A`

### 7.2.4 Position Message

The position message provides timestamped GPS data from the on-board radio. The message are sent once per second and will only be valid or populated when the GPS radio has a valid fix. The first value of an ASCII message is the character "P" and the arguments are numerical values. The first byte of a binary message is 0xD0 (equal to 0x80 + "P") and the arguments are several 32-bit longs and a couple 8-bit characters. The message argument are described in Table 19

| Argument | Description |
|---|---|
| 1 | GPS fix valid |
| 2 | Number of satellites |
| 3 | HDOP |
| 4 | Latitude in microdegrees |
| 5 | Longitude in microdegrees |
| 6 | Speed in meters per second |
| 7 | Course in millidegrees |

Table 19: Position message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. GPS fix valid = 1
2. Number of satellites = 12

3. HDOP = 7
4. Latitude = 280652369
5. Longitude = -806229767
6. Speed = 0
7. Course = 0

**ASCII example:** `P,1000000,1,12,7,280652369,-806229767,0,0\n`

**Binary example:** `50 40 42 0F 00 00 00 00 00 01 0C 07 10 BA 6A 51 30 0E 17 07 00 00 00 00 00 0A`

### 7.2.5 Quaternion message

The quaternion message provides timestamped measurements of the orientation of Thetis relative to the Earth. Quaternion messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "Q" and the arguments are four numerical values expressed to four decimal places. The first byte of a binary message is 0xD1 (equal to 0x80 + "Q") and the arguments are four contiguous 32-bit floats. The message arguments are described in Table 20.

| Argument | Description |
|---|---|
| 1 | Quaternion W element |
| 2 | Quaternion X element |
| 3 | Quaternion Y element |
| 4 | Quaternion Z element |

Table 20: Quaternion message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Quaternion W element = 1
2. Quaternion X element = 0
3. Quaternion Y element = 0
4. Quaternion Z element = 0

**ASCII example:** `Q,1000000,1.0000,0.0000,0.0000,0.0000\n`

**Binary example:** `D1 40 42 0F 00 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 0A`

### 7.2.6 Rotation matrix message

The rotation matrix message provides timestamped measurements of the orientation of Thetis relative to the Earth. Rotation matrix messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "R" and the arguments are nine numerical values expressed to four decimal places. The first byte of a binary message is 0xD2 (equal to 0x80 + "R") and the arguments are nine contiguous 32-bit floats. The message arguments are described in Table 21 on the next page.

| Argument | Description |
|---|---|
| 1 | Rotation matrix XX element |
| 2 | Rotation matrix XY element |
| 3 | Rotation matrix XZ element |
| 4 | Rotation matrix YX element |
| 5 | Rotation matrix YY element |
| 6 | Rotation matrix YZ element |
| 7 | Rotation matrix ZX element |
| 8 | Rotation matrix ZY element |
| 9 | Rotation matrix ZZ element |

Table 21: Rotation matrix message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Rotation matrix XX element = 1
2. Rotation matrix XY element = 0
3. Rotation matrix XZ element = 0
4. Rotation matrix YX element = 0
5. Rotation matrix YY element = 1
6. Rotation matrix YZ element = 0
7. Rotation matrix ZX element = 0
8. Rotation matrix ZY element = 0
9. Rotation matrix ZZ element = 1

**ASCII example:**
```
R,1000000,1.0000,0.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000,1.0000\n
```

**Binary example:**
```
D2 40 42 0F 00 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00
00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 3F 0A
```

### 7.2.7 Euler angles message

The Euler angles message provides timestamped measurements of the orientation of Thetis relative to the Earth. Euler angles messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character “A” and the arguments are three numerical values expressed to four decimal places. The first byte of a binary message is 0xC1 (equal to 0x80 + “A”) and the arguments are three contiguous 32-bit floats. The message arguments are described in Table 22.

| Argument | Description |
|---|---|
| 1 | Roll angle in degrees |
| 2 | Pitch angle in degrees |
| 3 | Yaw angle in degrees |

Table 22: Euler angles message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Roll angle = 0
2. Pitch angle = 0
3. Yaw angle = 0

**ASCII example:** `A,1000000,0.0000,0.0000,0.0000\n`

**Binary example:** `C1 40 42 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A`

### 7.2.8 Linear acceleration message

The linear acceleration message provides timestamped measurements of linear acceleration and the orientation of Thetis relative to the Earth. Linear acceleration messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "L" and the arguments are seven numerical values expressed to four decimal places. The first byte of a binary message is 0xCC (equal to 0x80 + "L") and the arguments are seven contiguous 32-bit floats. The message arguments are described in Table 23.

| Argument | Description |
|---|---|
| 1 | Quaternion W element |
| 2 | Quaternion X element |
| 3 | Quaternion Y element |
| 4 | Quaternion Z element |
| 5 | Linear acceleration X axis in g |
| 6 | Linear acceleration Y axis in g |
| 7 | Linear acceleration Z axis in g |

Table 23: Linear acceleration message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Quaternion W element = 1
2. Quaternion X element = 0
3. Quaternion Y element = 0
4. Quaternion Z element = 0
5. Linear acceleration X axis = 0
6. Linear acceleration Y axis = 0
7. Linear acceleration Z axis = 0

**ASCII example:** `L,1000000,1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000\n`

**Binary example:** `CC 40 42 0F 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A`

### 7.2.9 Earth acceleration message

The Earth acceleration message provides timestamped measurements of Earth acceleration and the orientation of Thetis relative to the Earth. Earth acceleration messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "E" and the arguments are seven numerical values expressed to four decimal places. The first byte of a binary message is 0xC5 (equal to 0x80 + "E") and the arguments are seven contiguous 32-bit floats. The message arguments are described in Table 24 on the following page.

| Argument | Description |
|---|---|
| 1 | Quaternion W element |
| 2 | Quaternion X element |
| 3 | Quaternion Y element |
| 4 | Quaternion Z element |
| 5 | Earth acceleration X axis in g |
| 6 | Earth acceleration Y axis in g |
| 7 | Earth acceleration Z axis in g |

Table 24: Earth acceleration message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Quaternion W element = 1
2. Quaternion X element = 0
3. Quaternion Y element = 0
4. Quaternion Z element = 0
5. Earth acceleration X axis = 0
6. Earth acceleration Y axis = 0
7. Earth acceleration Z axis = 0

**ASCII example:** `E,1000000,1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000\n`

**Binary example:**
```
C5 40 42 0F 00 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A
```

### 7.2.10 AHRS status message

The AHRS status message provides timestamped indications of the AHRS status flags. An AHRS status message is sent each time a status flag changes. AHRS status messages are only sent if AHRS messages are enabled in the device settings. The first value of an ASCII message is the character “U” and the arguments are four numerical values expressed to four decimal places. The first byte of a binary message is 0xD5 (equal to 0x80 + “U”) and the arguments are four contiguous 32-bit floats. The message arguments are described in Table 25.

| Argument | Description |
|---|---|
| 1 | Initialising |
| 2 | Angular rate recovery |
| 3 | Acceleration recovery |
| 4 | Magnetic recovery |

Table 25: AHRS status message arguments (all arguments are Boolean, see Table 26 on the next page)

| Value | Boolean |
|---|---|
| 0 | False |
| !0 | True |

Table 26: Boolean argument values

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Initialising = 1
2. Angular rate recovery = 0
3. Acceleration recovery = 0
4. Magnetic recovery = 0

**ASCII example:** `U,1000000,1.0000,0.0000,0.0000,0.0000\n`

**Binary example:** `D5 40 42 0F 00 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 0A`

### 7.2.11 High-g accelerometer message

The High-g accelerometer message provides timestamped high-g accelerometer measurements. High-g accelerometer messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character “H” and the arguments are three numerical values expressed to four decimal places. The first byte of a binary message is 0xC8 (equal to 0x80 + “H”) and the arguments are three contiguous 32-bit floats. The message arguments are described in Table 27.

| Argument | Description |
|---|---|
| 1 | High-g accelerometer X axis in g |
| 2 | High-g accelerometer Y axis in g |
| 3 | High-g accelerometer Z axis in g |

Table 27: High-g accelerometer message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. High-g accelerometer X axis = 0
2. High-g accelerometer Y axis = 0
3. High-g accelerometer Z axis = 1

**ASCII example:** `H,1000000,0.0000,0.0000,1.0000\n`

**Binary example:** `C8 40 42 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 3F 0A`

### 7.2.12 Temperature message

The temperature message provides timestamped temperature measurements. Temperature messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character “T” and the argument is a numerical value expressed to four decimal places. The first byte of a binary message is 0xD4 (equal to 0x80 + “T”) and the argument is a 32-bit float. The message arguments are described in Table 28 on the next page.

| Argument | Description |
|---|---|
| 1 | Temperature in degrees Celsius |

Table 28: Temperature message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Temperature = 25

**ASCII example:** `T,1000000,25.0000\n`

**Binary example:** `D4 40 42 0F 00 00 00 00 00 00 00 41 C8 0A`

### 7.2.13 Battery message

The battery message message provides timestamped measurements of the battery level and charger status. Battery message messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character “B” and the arguments are four numerical values expressed to four decimal places. The first byte of a binary message is 0xC2 (equal to 0x80 + “B”) and the arguments are four contiguous 32-bit floats. The message arguments are described in Table 29.

| Argument | Description |
|---|---|
| 1 | Battery percentage |
| 2 | Battery voltage in volts |
| 3 | Charging status (See Table 30) |

Table 29: Battery message arguments

| Charging status | Description |
|---|---|
| 0 | Not connected |
| 1 | Charging |
| 2 | Charging complete |

Table 30: Charging status enumeration

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Percentage = 100
2. Voltage = 4.2
3. Charging status = 2

**ASCII example:** `B,1000000,100.0000,4.2000,2.0000\n`

**Binary example:** `C2 40 42 0F 00 00 00 00 00 00 00 C8 42 66 66 86 40 00 00 00 40 0A`

### 7.2.14 RSSI message

The Received Signal Strength Indicator (RSSI) message provides timestamped Wi-Fi RSSI measurements. RSSI messages are sent continuously at the message rate configured in the device settings. RSSI messages will only be sent if Thetis is connected in Wi-Fi client mode. The first value of an ASCII message is the character "W" and the arguments are two numerical values expressed to four decimal places. The first byte of a binary message is 0xD7 (equal to 0x80 + "W") and the arguments are two contiguous 32-bit floats. The message arguments are described in Table 31.

| Argument | Description |
|---|---|
| 1 | RSSI percentage |
| 2 | RSSI power in dBm |

Table 31: RSSI message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. RSSI percentage = 100
2. RSSI power = -50

**ASCII example:** `W,1000000,100.000,-50.0000\n`

**Binary example:** `D7 40 42 0F 00 00 00 00 00 00 00 C8 42 00 00 48 C2 0A`

### 7.2.15 Serial accessory message

The serial accessory message provides timestamped received serial accessory data. Serial accessory messages are sent as serial accessory data is received as configured in the device settings. The first value of an ASCII message is the character "S" and the argument is the received data. Received byte values less than 0x20 or greater than 0x7E will be replaced with the character "?" so that the argument is a string of printable characters. The string is not null-terminated. The first byte of a binary message is 0xD3 (equal to 0x80 + "S") and the argument is the unmodified received data. The message arguments are described in Table 32.

| Argument | Description |
|---|---|
| 1 | Received serial accessory data |

Table 32: Serial accessory message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Data = 0x61 0x62 0x63 0x31 0x32 0x33 0xF1 0xF2 0xF3

**ASCII example:** `S,1000000,abc123???\n`

**Binary example:** `D3 40 42 0F 00 00 00 00 00 61 62 63 31 32 33 F1 F2 F3 0A`

### 7.2.16 Notification message

The notification message provides timestamped notifications of system events. Notification messages may be sent by Thetis at any time and cannot be disabled. The first value of an ASCII message is the character "N". The first byte of a binary message is 0xCE (equal to 0x80 + "N"). The argument of both ASCII and binary

messages is a string of printable characters. The string is not null-terminated. The message arguments are described in Table 33.

| Argument | Description |
|---|---|
| 1 | Notification string |

Table 33: Notification message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. String = This is a notification message.

**ASCII example:** `N,1000000,This is a notification message.\n`

**Binary example:** `CE 40 42 0F 00 00 00 00 00 54 68 69 73 20 69 73 20 61 20 6E 6F 74 69 66 69 63 61 74 69 6F 6E 20 6D 65 73 73 61 67 65 2E 0A`

### 7.2.17 Error message

The error message provides timestamped notifications of errors. Error messages may be sent by Thetis at any time and cannot be disabled. The first value of an ASCII message is the character "F". The first byte of a binary message is 0xC6 (equal to 0x80 + "F"). The argument of both ASCII and binary messages is a string of printable characters. The string is not null-terminated. The message arguments are described in Table 34.

| Argument | Description |
|---|---|
| 1 | Error string |

Table 34: Notification message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. String = This is an error message.

**ASCII example:** `F,1000000,This is an error message.\n`

**Binary example:** `C6 40 42 0F 00 00 00 00 00 54 68 69 73 20 69 73 20 61 6E 20 65 72 72 6F 72 20 6D 65 73 73 61 67 65 2E 0A`

# 8 Sample rates, message rates, and timestamps

This section describes the relationship between sample rates and message rates, and the role of timestamps in synchronisation.

## 8.1 Sample rates

The sample rate is the rate at which measurements are sampled by a measurement source. For example, an Analog-to-Digital Converter (ADC). All sample rates are fixed and cannot be adjusted by the user. Each measurement source has an independent sample clock. The sample rate and associated data messages for each measurement source are listed in Table 35 on the following page.

| Measurement source | Sample rate | Sample rate error | Data messages |
|---|---|---|---|
| Inertial sensor | 400 Hz | ±0.3% | Inertial, Quaternion, Rotation matrix Euler angles, Linear acceleration, Earth acceleration, Temperature |
| Magnetometer | 20 Hz | ±8% | Magnetometer |
| High-g accelerometer | 1600 Hz | ±5% | High-g accelerometer |
| Battery voltage | 5 Hz | - | Battery |
| RSSI | 1 Hz | - | RSSI |

Table 35: Sample rate and associated data messages for each measurement source

## 8.2 Message rates

The message rate is the rate at which a data message is sent. The message rate of each data message type is configured by a separate message rate divisor in the device settings. A message rate divisor is a positive integer that a fixed sample rate is divided by to obtain the message rate. For example, if the inertial message rate divisor is 4 then the inertial message rate will be 100 messages per second. A message rate divisor of 0 will disable the sending of that data message type. See Section 10.1.78 on page 52 to Section 10.1.84 on page 53 for a detailed description and examples for each message rate divisor setting.

## 8.3 Sample averaging

If a message rate divisor is greater than 1 then the measurements in each data message will be the average of the most recent *n* samples where *n* equal to the message rate divisor. The timestamp of the data message will be that of the most recent sample. For example, if the inertial message rate divisor is 8 then the measurements in each inertial message will be the average of 8 samples and the timestamp of the message will be that of the $8^{th}$ sample.

## 8.4 Timestamps

The timestamp of a data message indicates the time at which a measurement was obtained. For example, when an ADC conversion completes. Timestamps are therefore not affected by the sample rate error or the latency of a communication interface. Applications that involve time-dependent calculations such as numerical integration or interpolation should not infer timing from the nominal sample rate and should instead use the timestamp of each measurement. A timestamp is the number of microseconds since Thetis was switched on, with a resolution of one microsecond.

## 8.5 Synchronisation

Multiple x-IMU3s operating on the same Wi-Fi network will automatically synchronise so that the timestamps from all x-IMU3s is the number of microseconds since the first x-IMU3 was switched on. The sample clocks of synchronised x-IMU3s will remain asynchronous. If an application requires synchronous sampling then this must be achieved in post-processing through interpolation and resampling.

# 9 Network announcement message

The network announcement message is used by a host to discover and connect to x-IMU3s on the same network. The message is continuously broadcast by the x-IMU3 on UDP port 10000 at a fixed rate of one message per second. Each message provides the device name, serial number, Wi-Fi and battery status, as well as the device settings required for a host to establish a Transmission Control Protocol (TCP) or UDP connection. The message is a single JSON object. The key/value pairs are described in Table 36 on the next page.

| Key | Value type | Description |
|---|---|---|
| “sync” | number | Used for synchronisation |
| “name” | string | Device name |
| “sn” | string | Device serial number |
| “ip” | string | x-IMU3 IP address |
| “port” | number | TCP port |
| “send” | number | UDP send port (x-IMU3 sends to this port) |
| “receive” | number | UDP receive port (x-IMU3 receives on this port) |
| “rssi” | number | RSSI percentage (-1 in Wi-Fi AP mode) |
| “battery” | number | Battery percentage |
| “status” | number | Charging status (See Table 30 on page 35) |

Table 36: Network announcement message key/value pairs

**Example*:**

```
{
  "sync":      0,
  "name":      "x-IMU3",
  "sn":        "0123-4567-89AB-CDEF",
  "ip":        "192.168.1.1",
  "port":      7000,
  "send":      8000,
  "receive":   9000,
  "rssi":      100,
  "battery":   100,
  "status":    2
}
```

* The actual JSON will not include any whitespace.

# 10 Device Settings

## 10.1 Individual Settings

### 10.1.1 Calibration date (read-only)

**Description:** Calibration date.

**JSON key:** `"calibrationDate"`

**JSON value type:** string

**Default value:** “Unknown”

### 10.1.2 System clock calibration (read-only)

**Description:** System clock calibration.

**JSON key:** `"systemClockCalibration"`

**JSON value type:** number

**Default value:** 1.0

#### 10.1.3 RTC calibration (read-only)

| | |
|---|---|
| **Description:** | RTC calibration. |
| **JSON key:** | `"rtcCalibration"` |
| **JSON value type:** | number |
| **Default value:** | 1.0 |

#### 10.1.4 Battery voltmeter sensitivity (read-only)

| | |
|---|---|
| **Description:** | Battery voltmeter sensitivity. |
| **JSON key:** | `"batteryVoltmeterSensitivity"` |
| **JSON value type:** | number |
| **Default value:** | 1.0 |

#### 10.1.5 Gyroscope misalignment (read-only)

| | |
|---|---|
| **Description:** | Gyroscope misalignment. |
| **JSON key:** | `"gyroscopeMisalignment"` |
| **JSON value type:** | array of 9 numbers |
| **Default value:** | [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0] |

#### 10.1.6 Gyroscope sensitivity (read-only)

| | |
|---|---|
| **Description:** | Gyroscope sensitivity. |
| **JSON key:** | `"gyroscopeSensitivity"` |
| **JSON value type:** | array of 3 numbers |
| **Default value:** | [1.0, 1.0, 1.0] |

#### 10.1.7 Gyroscope offset (read-only)

| | |
|---|---|
| **Description:** | Gyroscope offset. |
| **JSON key:** | `"gyroscopeOffset"` |
| **JSON value type:** | array of 3 numbers |
| **Default value:** | [0.0, 0.0, 0.0] |

#### 10.1.8 Accelerometer misalignment (read-only)

| | |
|---|---|
| **Description:** | Accelerometer misalignment. |
| **JSON key:** | `"accelerometerMisalignment"` |
| **JSON value type:** | array of 9 numbers |
| **Default value:** | [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0] |

#### 10.1.9 Accelerometer sensitivity (read-only)

**Description:** Accelerometer sensitivity.

**JSON key:** `"accelerometerSensitivity"`

**JSON value type:** array of 3 numbers

**Default value:** [1.0, 1.0, 1.0]

#### 10.1.10 Accelerometer offset (read-only)

**Description:** Accelerometer offset.

**JSON key:** `"accelerometerOffset"`

**JSON value type:** array of 3 numbers

**Default value:** [0.0, 0.0, 0.0]

#### 10.1.11 Soft iron matrix (read-only)

**Description:** Soft iron matrix.

**JSON key:** `"softIronMatrix"`

**JSON value type:** array of 9 numbers

**Default value:** [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]

#### 10.1.12 Hard iron offset (read-only)

**Description:** Hard iron offset.

**JSON key:** `"hardIronOffset"`

**JSON value type:** array of 3 numbers

**Default value:** [0.0, 0.0, 0.0]

#### 10.1.13 High-g accelerometer misalignment (read-only)

**Description:** High-g accelerometer misalignment.

**JSON key:** `"highGAccelerometerMisalignment"`

**JSON value type:** array of 9 numbers

**Default value:** [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]

#### 10.1.14 High-g accelerometer sensitivity (read-only)

**Description:** High-g accelerometer sensitivity.

**JSON key:** `"highGAccelerometerSensitivity"`

**JSON value type:** array of 3 numbers

**Default value:** [1.0, 1.0, 1.0]

#### 10.1.15 High-g accelerometer offset (read-only)

| | |
|---|---|
| **Description:** | High-g accelerometer offset. |
| **JSON key:** | `"highGAccelerometerOffset"` |
| **JSON value type:** | array of 3 numbers |
| **Default value:** | [0.0, 0.0, 0.0] |

#### 10.1.16 Device name

| | |
|---|---|
| **Description:** | Device name. |
| **JSON key:** | `"deviceName"` |
| **JSON value type:** | string |
| **Default value:** | “x-IMU3” |

#### 10.1.17 Serial number (read-only)

| | |
|---|---|
| **Description:** | Serial number. |
| **JSON key:** | `"serialNumber"` |
| **JSON value type:** | string |
| **Default value:** | “Unknown” |

#### 10.1.18 Firmware version (read-only)

| | |
|---|---|
| **Description:** | Firmware version. |
| **JSON key:** | `"firmwareVersion"` |
| **JSON value type:** | string |
| **Default value:** | “Unknown” |

#### 10.1.19 Bootloader version (read-only)

| | |
|---|---|
| **Description:** | Bootloader version. |
| **JSON key:** | `"bootloaderVersion"` |
| **JSON value type:** | string |
| **Default value:** | “Unknown” |

#### 10.1.20 Hardware version (read-only)

| | |
|---|---|
| **Description:** | Hardware version. |
| **JSON key:** | `"hardwareVersion"` |
| **JSON value type:** | string |
| **Default value:** | “Unknown” |

#### 10.1.21 Serial mode

**Description:** Serial mode.

**JSON key:** `"serialMode"`

**JSON value type:** number

**Default value:** 0

#### 10.1.22 Serial baud rate

**Description:** Serial baud rate.

**JSON key:** `"serialBaudRate"`

**JSON value type:** number

**Default value:** 115200

#### 10.1.23 Serial RTS/CTS enabled

**Description:** Serial Request To Send (RTS)/Clear To Send (CTS) enabled.

**JSON key:** `"serialRtsCtsEnabled"`

**JSON value type:** true or false

**Default value:** false

#### 10.1.24 Serial accessory number of bytes

**Description:** Serial accessory number of bytes.

**JSON key:** `"serialAccessoryNumberOfBytes"`

**JSON value type:** number

**Default value:** 1024

#### 10.1.25 Serial accessory termination byte

**Description:** Serial accessory termination byte.

**JSON key:** `"serialAccessoryTerminationByte"`

**JSON value type:** number

**Default value:** 10

#### 10.1.26 Serial accessory timeout

**Description:** Serial accessory timeout.

**JSON key:** `"serialAccessoryTimeout"`

**JSON value type:** number

**Default value:** 100

#### 10.1.27 Wireless mode

**Description:** Wireless mode.

**JSON key:** `"wirelessMode"`

**JSON value type:** number

**Default value:** 2

#### 10.1.28 Wireless firmware version (read-only)

**Description:** Wireless firmware version.

**JSON key:** `"wirelessFirmwareVersion"`

**JSON value type:** string

**Default value:** “Unknown”

#### 10.1.29 External antennae enabled

**Description:** External antennae enabled.

**JSON key:** `"externalAntennaeEnabled"`

**JSON value type:** true or false

**Default value:** false

#### 10.1.30 Wi-Fi region

**Description:** Wi-Fi region.

**JSON key:** `"wiFiRegion"`

**JSON value type:** number

**Default value:** 2

#### 10.1.31 Wi-Fi MAC address (read-only)

**Description:** Wi-Fi Media Access Control (MAC) address.

**JSON key:** `"wiFiMacAddress"`

**JSON value type:** string

**Default value:** 0

#### 10.1.32 Wi-Fi IP address (read-only)

**Description:** Wi-Fi Internet Protocol (IP) address.

**JSON key:** `"wiFiIPAddress"`

**JSON value type:** string

**Default value:** 0

#### 10.1.33 Wi-Fi client SSID

**Description:** Wi-Fi client Service Set Identifier (SSID).

**JSON key:** `"wiFiClientSsid"`

**JSON value type:** string

**Default value:** “”

#### 10.1.34 Wi-Fi client key

**Description:** Wi-Fi client key.

**JSON key:** `"wiFiClientKey"`

**JSON value type:** string

**Default value:** “”

#### 10.1.35 Wi-Fi client channel

**Description:** Wi-Fi client channel.

**JSON key:** `"wiFiClientChannel"`

**JSON value type:** number

**Default value:** 0

#### 10.1.36 Wi-Fi client DHCP enabled

**Description:** Wi-Fi client Dynamic Host Configuration Protocol (DHCP) enabled.

**JSON key:** `"wiFiClientDhcpEnabled"`

**JSON value type:** true or false

**Default value:** true

#### 10.1.37 Wi-Fi client IP address

**Description:** Wi-Fi client IP address.

**JSON key:** `"wiFiClientIPAddress"`

**JSON value type:** string

**Default value:** “192.168.1.2”

#### 10.1.38 Wi-Fi client netmask

**Description:** Wi-Fi client netmask.

**JSON key:** `"wiFiClientNetmask"`

**JSON value type:** string

**Default value:** “255.255.255.0”

#### 10.1.39 Wi-Fi client gateway

**Description:** Wi-Fi client gateway.
**JSON key:** `"wiFiClientGateway"`
**JSON value type:** string
**Default value:** “192.168.1.1”

#### 10.1.40 Wi-Fi AP SSID

**Description:** Wi-Fi Access Point (AP) SSID.
**JSON key:** `"wiFiAPSsid"`
**JSON value type:** string
**Default value:** “”

#### 10.1.41 Wi-Fi AP key

**Description:** Wi-Fi AP key.
**JSON key:** `"wiFiAPKey"`
**JSON value type:** string
**Default value:** “”

#### 10.1.42 Wi-Fi AP channel

**Description:** Wi-Fi AP channel.
**JSON key:** `"wiFiAPChannel"`
**JSON value type:** number
**Default value:** 36

#### 10.1.43 Wi-Fi AP IP address

**Description:** Wi-Fi AP IP address.
**JSON key:** `"wiFiAPIPAddress"`
**JSON value type:** string
**Default value:** 0x0101A9C0

#### 10.1.44 TCP port

**Description:** TCP port.
**JSON key:** `"tcpPort"`
**JSON value type:** number
**Default value:** 7000

#### 10.1.45 UDP IP address

**Description:** UDP IP address.
**JSON key:** `"udpIPAddress"`
**JSON value type:** string
**Default value:** 0

#### 10.1.46 UDP send port

**Description:** UDP send port.
**JSON key:** `"udpSendPort"`
**JSON value type:** number
**Default value:** 0

#### 10.1.47 UDP receive port

**Description:** UDP receive port.
**JSON key:** `"udpReceivePort"`
**JSON value type:** number
**Default value:** 9000

#### 10.1.48 UDP low latency

**Description:** UDP low latency.
**JSON key:** `"udpLowLatency"`
**JSON value type:** true or false
**Default value:** false

#### 10.1.49 Synchronisation enabled

**Description:** Synchronisation enabled.
**JSON key:** `"synchronisationEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.50 Synchronisation network latency

**Description:** Synchronisation network latency.
**JSON key:** `"synchronisationNetworkLatency"`
**JSON value type:** number
**Default value:** 1500

#### 10.1.51 Bluetooth address (read-only)

**Description:** Bluetooth address.
**JSON key:** `"bluetoothAddress"`
**JSON value type:** number
**Default value:** 0

#### 10.1.52 Bluetooth name

**Description:** Bluetooth name.
**JSON key:** `"bluetoothName"`
**JSON value type:** string
**Default value:** “”

#### 10.1.53 Bluetooth pin code

**Description:** Bluetooth pin code.
**JSON key:** `"bluetoothPinCode"`
**JSON value type:** string
**Default value:** “”

#### 10.1.54 Bluetooth discovery mode

**Description:** Bluetooth discovery mode.
**JSON key:** `"bluetoothDiscoveryMode"`
**JSON value type:** number
**Default value:** 2

#### 10.1.55 Bluetooth paired address (read-only)

**Description:** Bluetooth paired address.
**JSON key:** `"bluetoothPairedAddress"`
**JSON value type:** number
**Default value:** 0

#### 10.1.56 Bluetooth paired link key (read-only)

**Description:** Bluetooth paired link key.
**JSON key:** `"bluetoothPairedLinkKey"`
**JSON value type:** number
**Default value:** 0

#### 10.1.57 Data logger enabled

**Description:** Data logger enabled.
**JSON key:** `"dataLoggerEnabled"`
**JSON value type:** true or false
**Default value:** false

#### 10.1.58 Data logger file name prefix

**Description:** Data logger file name prefix.
**JSON key:** `"dataLoggerFileNamePrefix"`
**JSON value type:** string
**Default value:** “”

#### 10.1.59 Data logger file name time enabled

**Description:** Data logger file name time enabled.
**JSON key:** `"dataLoggerFileNameTimeEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.60 Data logger file name counter enabled

**Description:** Data logger file name counter enabled.
**JSON key:** `"dataLoggerFileNameCounterEnabled"`
**JSON value type:** true or false
**Default value:** false

#### 10.1.61 Data logger max file size

**Description:** Data logger max file size.
**JSON key:** `"dataLoggerMaxFileSize"`
**JSON value type:** number
**Default value:** 0

#### 10.1.62 Data logger max file period

**Description:** Data logger max file period.
**JSON key:** `"dataLoggerMaxFilePeriod"`
**JSON value type:** number
**Default value:** 0

#### 10.1.63 Axes alignment

**Description:** Axes alignment.
**JSON key:** `"axesAlignment"`
**JSON value type:** number
**Default value:** 0

#### 10.1.64 Gyroscope offset correction enabled

**Description:** Gyroscope offset correction enabled.
**JSON key:** `"gyroscopeOffsetCorrectionEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.65 AHRS axes convention

**Description:** AHRS axes convention.
**JSON key:** `"ahrsAxesConvention"`
**JSON value type:** number
**Default value:** 0

#### 10.1.66 AHRS gain

**Description:** AHRS gain.
**JSON key:** `"ahrsGain"`
**JSON value type:** number
**Default value:** 0.5

#### 10.1.67 AHRS ignore magnetometer

**Description:** AHRS ignore magnetometer.
**JSON key:** `"ahrsIgnoreMagnetometer"`
**JSON value type:** true or false
**Default value:** false

#### 10.1.68 AHRS acceleration rejection enabled

**Description:** AHRS acceleration rejection enabled.
**JSON key:** `"ahrsAccelerationRejectionEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.69 AHRS magnetic rejection enabled

**Description:** AHRS magnetic rejection enabled.
**JSON key:** `"ahrsMagneticRejectionEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.70 Binary mode enabled

**Description:** Binary mode enabled.
**JSON key:** `"binaryModeEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.71 USB data messages enabled

**Description:** USB data messages enabled.
**JSON key:** `"usbDataMessagesEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.72 Serial data messages enabled

**Description:** Serial data messages enabled.
**JSON key:** `"serialDataMessagesEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.73 TCP data messages enabled

**Description:** TCP data messages enabled.
**JSON key:** `"tcpDataMessagesEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.74 UDP data messages enabled

**Description:** UDP data messages enabled.
**JSON key:** `"udpDataMessagesEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.75 Bluetooth data messages enabled

**Description:** Bluetooth data messages enabled.
**JSON key:** `"bluetoothDataMessagesEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.76 Data logger data messages enabled

**Description:** Data logger data messages enabled.
**JSON key:** `"dataLoggerDataMessagesEnabled"`
**JSON value type:** true or false
**Default value:** true

#### 10.1.77 AHRS message type

**Description:** AHRS message type.
**JSON key:** `"ahrsMessageType"`
**JSON value type:** number
**Default value:** 0

#### 10.1.78 Inertial message rate divisor

**Description:** Inertial message rate divisor.
**JSON key:** `"inertialMessageRateDivisor"`
**JSON value type:** number
**Default value:** 8

#### 10.1.79 Magnetometer message rate divisor

**Description:** Magnetometer message rate divisor.
**JSON key:** `"magnetometerMessageRateDivisor"`
**JSON value type:** number
**Default value:** 1

#### 10.1.80 AHRS message rate divisor

**Description:** AHRS message rate divisor.
**JSON key:** `"ahrsMessageRateDivisor"`
**JSON value type:** number
**Default value:** 8

#### 10.1.81 High-g accelerometer message rate divisor

| | |
|---|---|
| **Description:** | High-g accelerometer message rate divisor. |
| **JSON key:** | `"highGAccelerometerMessageRateDivisor"` |
| **JSON value type:** | number |
| **Default value:** | 32 |

#### 10.1.82 Temperature message rate divisor

| | |
|---|---|
| **Description:** | Temperature message rate divisor. |
| **JSON key:** | `"temperatureMessageRateDivisor"` |
| **JSON value type:** | number |
| **Default value:** | 5 |

#### 10.1.83 Battery message rate divisor

| | |
|---|---|
| **Description:** | Battery message rate divisor. |
| **JSON key:** | `"batteryMessageRateDivisor"` |
| **JSON value type:** | number |
| **Default value:** | 5 |

#### 10.1.84 RSSI message rate divisor

| | |
|---|---|
| **Description:** | RSSI message rate divisor. |
| **JSON key:** | `"rssiMessageRateDivisor"` |
| **JSON value type:** | number |
| **Default value:** | 1 |

#### 10.1.85 FTP! enabled

| | |
|---|---|
| **Description:** | **FTP!** (**FTP!**) enabled. |
| **JSON key:** | `"ftpEnabled"` |
| **JSON value type:** | true or false |
| **Default value:** | false |

#### 10.1.86 FTP! username

| | |
|---|---|
| **Description:** | **FTP!** username. |
| **JSON key:** | `"ftpUsername"` |
| **JSON value type:** | string |
| **Default value:** | “” |

#### 10.1.87 Log print level

**Description:** Log print level.
**JSON key:** `"logPrintLevel"`
**JSON value type:** number
**Default value:** 0

#### 10.1.88 Log file level

**Description:** Log file level.
**JSON key:** `"logFileLevel"`
**JSON value type:** number
**Default value:** 2

#### 10.1.89 GPS RTC sync enabled

**Description:** GPS RTC sync enabled.
**JSON key:** `"gpsRtcSyncEnabled"`
**JSON value type:** true or false
**Default value:** false

#### 10.1.90 Accelerometer range

**Description:** Accelerometer range.
**JSON key:** `"accelerometerRange"`
**JSON value type:** number
**Default value:** 2

#### 10.1.91 Gyroscope range

**Description:** Gyroscope range.
**JSON key:** `"gyroscopeRange"`
**JSON value type:** number
**Default value:** 0

#### 10.1.92 IMU data rate

**Description:** IMU data rate.
**JSON key:** `"imuDataRate"`
**JSON value type:** number
**Default value:** 5

#### 10.1.93 Magnetometer performance mode

**Description:** Magnetometer performance mode.
**JSON key:** `"magnetometerPerformanceMode"`
**JSON value type:** number
**Default value:** 1

#### 10.1.94 Magnetometer operation mode

**Description:** Magnetometer operation mode.
**JSON key:** `"magnetometerOperationMode"`
**JSON value type:** number
**Default value:** 0

#### 10.1.95 Magnetometer data rate

**Description:** Magnetometer data rate.
**JSON key:** `"magnetometerDataRate"`
**JSON value type:** number
**Default value:** 1

#### 10.1.96 Magnetometer range

**Description:** Magnetometer range.
**JSON key:** `"magnetometerRange"`
**JSON value type:** number
**Default value:** 0

#### 10.1.97 Gauge reset voltage

**Description:** Gauge reset voltage.
**JSON key:** `"gaugeResetVoltage"`
**JSON value type:** number
**Default value:** 2.5

#### 10.1.98 Gauge activity threshold

**Description:** Gauge activity threshold.
**JSON key:** `"gaugeActivityThreshold"`
**JSON value type:** number
**Default value:** 0.15

#### 10.1.99 Gauge hibernation threshold

| | |
|---|---|
| **Description:** | Gauge hibernation threshold. |
| **JSON key:** | `"gaugeHibernationThreshold"` |
| **JSON value type:** | number |
| **Default value:** | 0.1 |

#### 10.1.100 Gauge alert minimum voltage

| | |
|---|---|
| **Description:** | Gauge alert minimum voltage. |
| **JSON key:** | `"gaugeAlertMinimumVoltage"` |
| **JSON value type:** | number |
| **Default value:** | 3.0 |

#### 10.1.101 Gauge alert maximum voltage

| | |
|---|---|
| **Description:** | Gauge alert maximum voltage. |
| **JSON key:** | `"gaugeAlertMaximumVoltage"` |
| **JSON value type:** | number |
| **Default value:** | 4.2 |

#### 10.1.102 Fusion update rate

| | |
|---|---|
| **Description:** | Fusion update rate. |
| **JSON key:** | `"fusionUpdateRate"` |
| **JSON value type:** | number |
| **Default value:** | 20 |

# Glossary

**a.u.** arbitrary units . . . 14

**ADC** Analog-to-Digital Converter . . . 37

**AHRS** Attitude Heading Reference System . . . 14

**AP** Access Point . . . 46

**API** Application Programming Interface

**ASCII** American Standard Code for Information Interchange . . . 27

**CDC** Communications Device Class

**CS** Chip Select

**CSV** Comma-Separated Values . . . 21

**CTS** Clear To Send . . . 43

**DOF** Degree of Freedom . . . 7

**DHCP** Dynamic Host Configuration Protocol . . . 45

**EEPROM** Electrically Erasable Programmable Read-Only Memory . . . 23

**GPIO** General Purpose Input Output

**GPS** Global Positioning System . . . 21

**GUI** Graphical User Interface

**HDOP** Horizontal Dillution of Precision

**I2C** Inter-Integrated Circuit

**IC** Integrated Circuit

**IMU** Inertial Measurement Unit . . . 7

**IP** Internet Protocol . . . 44

**IP67** Ingress Protection 67 . . . . . 11

**JSON** JavaScript Object Notation . . . . . 22

**LED** Light-Emitting Diode . . . . . 17

**LF** Line Feed . . . . . 22

**MAC** Media Access Control . . . . . 44

**MISO** Master In Slave Out

**MOSI** Master Out Slave In

**RGB** Red Green Blue . . . . . 17

**RMS** Root Mean Square . . . . . 15

**RSSI** Received Signal Strength Indicator . . . . . 36

**RTC** Real-Time Clock . . . . . 24

**RTS** Request To Send . . . . . 43

**RX** Receive

**microSD** micro Secure Digital . . . . . 21

**SCK** Signal Clock (SPI)

**SCL** Signal Clock (I2C)

**SDA** Signal Data

**SLIP** Serial Line Internet Protocol . . . . . 27

**SPI** Serial Peripheral Interface

**SSID** Service Set Identifier . . . . . 45

**TCP** Transmission Control Protocol . . . . . 38

**TX** Transmit

**UDP** User Datagram Protocol . . . . . 22

**UART** Universal Asynchronous Receiver-Transmitter

**USB** Universal Serial Bus . . . . . 21

# Document version history

| Version | Date | Changes |
|---|---|---|
| v1.0 | October 31, 2023 | • Converted original x-IMU3 user manual (v1.5) to Thetis user manual and released |

# Disclaimer

The information in this document pertains to information related to x-io Technologies products. This information is provided as a service to our customers, and may be used for information purposes only.

x-io Technologies assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at x-io Technologies' sole discretion without any prior notice to anyone. x-io Technologies is not committed to updating this document in the future.