debug innovations 0101001101

Mataki-Classic User Guide

Version 1

This document provides an in depth guide to the Mataki-Classic GPS tracker, including descriptions of hardware features and how to effectively use it for experiments.

Version History

Version	Date	Changes
1	10 April 2018	First Release

Related Documents

PuTTY Installation and Configuration
Mataki Support Board User Guide
Mataki-Classic Programming Guide

Contents

1. Introduction

Mataki-Classic tags are the largest tags in the Mataki range. They weigh approximately 10g excluding a battery and can act as both a tracker and a basic base station. In addition to a GPS and radio, they also have a temperature and a barometric pressure sensor from which altitude can be calculated and a 3 axis accelerometer.

Unlike Mataki-Lite, Mataki-Classic tags do not support user-written tracker scripts. Instead, the GPS, Radio and Accelerometer sample rates are controlled with user settings, whereas temperature, pressure, light level and battery voltage are sampled at the same time as GPS fixes are taken. This means the possible behaviours for the tag are less complex than with Mataki-Lite.

When used as a tracker, the tag can upload its log data to either another Mataki-Classic acting as a base station or to a Pi-Base base station.

When acting as a base station, logs are collected from nearby tags and stored as a cumulative log, which can then be read using a support board connected to a PC. The cumulative log can then be split into individual logs for each tag using a utility program called the log converter.

1.1. Tag Versions / History

The original concept for Mataki tags came from Robin Freeman of Microsoft Research in Cambridge, UK. V1-V3 PCBs were developed by Microsoft but they were not used in animal studies due to functional issues. V4 tags were developed by Debug Innovations to Microsoft specifications and are not compatible with V5 tags. A small quantity were produced and limited animal testing was done. Consequently, PCBs before V5 are of historical interest only and this guide only covers the V5 range of tags.

V5 PCBs were developed by Debug Innovations. At the time of writing there have been 6 PCB versions shown in the table below. They are all the same size and shape and they all fit in the same support board. The table below shows the main differences between tag versions...

PCB Version	Changes
V5	First version - uses a PA6B GPS, SCP1000 temp/pressure sensor, CMA3000 accelerometer and has a 16MB flash (for the log)
V5.1	Bug fixes - On/Off switch legend was incorrect on V5
V5.11	Flash upgraded to 32MB
V5.2	Temp/Pressure sensor changed to BMP180 due to obsolescence Accelerometer changed to LSM303 due to obsolescence
V5.3	Upgraded to PA6C GPS which uses less power
V5.4	Temp/Pressure sensor changed to BMP280 due to obsolescence Added an SMA antenna connector option and DC blocking capacitor Some PCBs have PA6H GPS modules which are lighter

The original firmware was developed at Berlin University under contract from Microsoft. In June 2014, Debug Innovations obtained the source code and a license from Microsoft and took over the Mataki firmware development. The firmware supplied by Debug Innovations will run on all the V5 range. It auto-detects the PCB version and adapts its device drivers and behaviour accordingly. The log converter utility was written by Robin Freeman and is informally maintained by Debug Innovations to keep it in step with firmware changes.

Unless otherwise stated, this document refers to V5.4 PCBs running V5.4.4 firmware.

2. Specifications

Size	44 x 21.75mm
Weight	Approx. 10g
Radio Base Frequency	868MHz (Europe) 916MHz (USA)
Max. Transmit Power (at antenna)	10mW (+10dBm)
Battery (not supplied)	3.6V Lithium
Log Capacity	932066 Entries *
Battery Voltage Sensor	✓
GPS Position	✓
Light Sensor	✓
Accelerometer	✓
Temperature / Pressure Sensor	✓
Current Consumption	Typical Figures
Sleeping	30 µA
GPS Module	26 mA **
Radio	19 mA ***

* V5 and V5.1 PCBs have half the log capacity (466032 entries)
** During acquisition, tracking current is ~20mA
PA6B GPS on PCBs prior to V5.3 takes ~48mA(Acq), 37mA(Trk)
*** When receiving

3. Firmware Applications

Mataki-Classic tags can be programmed with different applications. Unlike Mataki-Lite scripts, changing applications involves replacing the entire tag firmware and removes all the tag's settings. For details of the programming procedure, see the Mataki-Classic Programming Guide.

There are three applications supplied in the standard firmware distribution. These provide the two main roles undertaken by Mataki-Classic tags (tracker and base station) and a third application is for testing the tag hardware.

3.1. Tracker Application

This is the default application loaded on to new tags and is the main operating mode for Mataki-Classic tags. GPS fixes and sensor readings are taken at set intervals and stored as logs in flash memory. In between readings, the tags enter a sleep state where they consume a minimum amount of power.

Periodically, the tag attempts to contact the base station. If contact is made, a series of radio messages are exchanged where the tag responds to requests from the base station for log entries it doesn't currently have, resulting in the transfer of the tag's entire log to the base station (possibly over more than one session). When the base station has all the logs, it instructs the tag to re-boot itself.

The period between sensor readings (GPS, temp/pressure, light level, battery voltage, accelerometer) and radio activity is set with commands through a connected PuTTY terminal. See section 10 for more details.

3.2. Base Application

This application allows a Mataki-Classic tag to act as a base station to collect log data from tags running the tracker application. Only one tag should be running as a base station at any one time and place. Multiple base stations will interfere with each other and limit the amount of logs collected.

Mataki-Classic base stations can also collect logs from Mataki-Lite trackers. The resulting cumulative log contains entries from both Classic and Lite tags so all tags, whatever the type, need to have unique IDs to be able to separate the individual tag logs.

When operating as a base station, the radio is always on. Most of the time it is in receive mode, listening for heartbeats, consuming around 20mA. The sensors are not used by the base application so no measurement periods need to be set. Likewise, because the radio is always on, no radio setting is necessary.

Batteries for base station tags typically need to be much larger in order to support the extra power consumed by the continuous radio use (approx 500mAh capacity per day).

For good radio range, it is important to use the best antenna possible, located as high up as possible and clear of any obstacles. V5.4 tags have an SMA patch on the PCB which can be used to connect an external antenna. Older tags should use wire antennas for the base station.

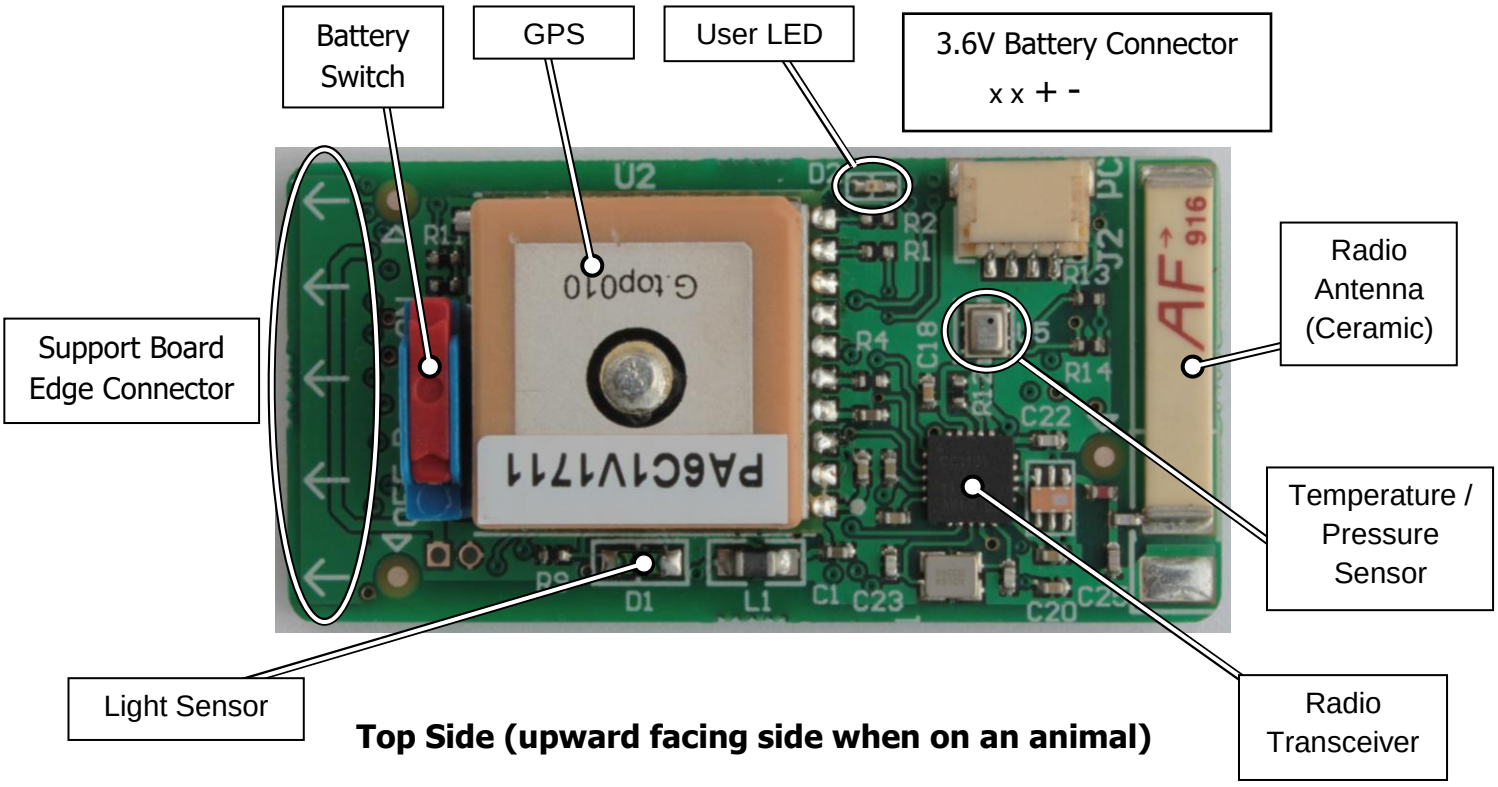
3.3. Service Application

This is a special application designed for hardware testing. If there are any suspected hardware faults on a tag, then load this firmware to help diagnose any issues.

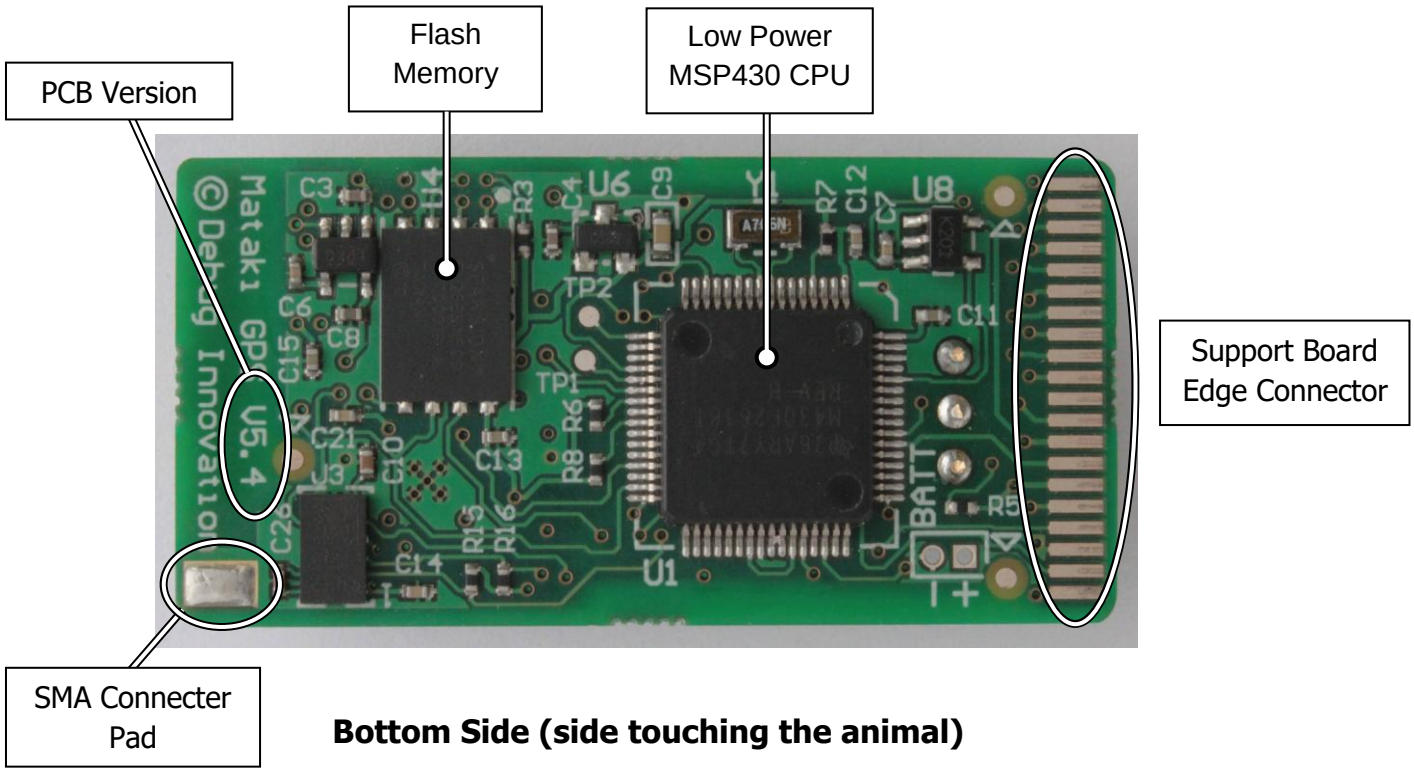
Initially, a series of hardware tests are run automatically, quickly diagnosing any basic faults. Then a radio test sends packets to another tag running the service application to test the radio works.

After the tests have finished, the application enters the normal command mode where extra commands are available that can help to test the on-board devices. For example, a GPS fix can be obtained and displayed, the accelerometer can be tested interactively and the radio can be set to generate a carrier.

4. Tag Hardware



Top Side (upward facing side when on an animal)



Bottom Side (side touching the animal)

4.1. Board Orientation

Components are carefully laid out such that the bottom side is mostly flat and can rest on an animal.

The GPS antenna (the metal square on top of the GPS receiver) must be able to 'see sky' to receive a satellite signal so it is on the top side.

The light sensor obviously needs to be located on the top side too. Take care not to cover the light sensor with wrapping.

The temperature / pressure sensor is designed to sense air pressure and has a small hole in the top. It can't be used for water depth measurements and consideration needs to be given to waterproofing in a way that it isn't completely air-tight. For diving species, you may choose not to use the pressure sensor.

4.2. Comparison with Mataki-Lite

Both Mataki-Classic and Mataki-Lite use the support board, so the width of both tags is the same, but Mataki-Classic is 10mm longer. The support board edge connector is the same, as are most of the pin functions, see Mataki Support Board User Guide for further information.

Mataki-Classic has an On/Off switch, which Mataki-Lite does not.

Mataki-Classic is approximately 3 times heavier than Mataki-Lite. This is mainly down to the much larger GPS module, but some of this can be attributed to the extra on-board hardware and larger PCB.

Mataki-Classic has a large flash memory chip for log storage. This is required in the base station role in order to contain all the logs from all the tracker tags. Since Mataki-Lite can't be used as a base station, it requires a lot less flash, so the CPU's internal flash storage is used.

Mataki-Classic has an accelerometer and air pressure/temperature sensor. Mataki-Lite is designed primarily for GPS tracking so doesn't have these sensors. However, it does retain the light sensor and battery voltage measurement features.

Both tags use a UART interface to communicate with the PC and a PuTTY terminal as a command interface. However, on Mataki-Classic tags, the command interface is used to control the logging settings while the logging and radio communications continues in the background. On Mataki-Lite the command interface is used to write and run a script which does the logging and communicates with the base station only when it is running.

4.3. Electronics

Battery Power

Power is supplied from a 3.6V battery connected to J2 (see photo in section 4). The red On/Off switch controls the power when running from batteries. This is regulated down to 3.3V for the CPU (always on) and the GPS (enabled by the firmware when needed). The regulators are located on the bottom side of the PCB.

If a battery is connected to a tag, it can be charged by plugging the tag into a support board and using SW2. The charger supplies a maximum of 100mA so it is recommended to only use the charger with batteries of 100mAh or greater capacity (1C charge rate).

Support Board Power

When plugged into a support board, power can be supplied from a battery in the normal way if the red On/Off switch is in the ON position or supplied by a 3.8V battery substitute controlled by SW1 on the support board if the red On/Off switch is in the OFF position. See Support Board User Guide for more information.

Alternative Power Sources for Base Stations

Base stations consume more power than trackers because their radios are always on. If you want to operate a base station for a long period, it can be mains powered or powered from a larger type of battery e.g. a car battery or USB power pack but these sources are at the wrong voltage for Mataki-Classic tags.

Mataki-Classic requires a 3.6V to 4.2V at 100mA. However, switch-on surges may require more current and/or a large capacitor e.g. 1000µF across the power rails.

To step down a 5V USB supply to 4V, a linear regulator can be used. If a charger is used, please make sure the output current is sufficient to supply the switch-on surge, or connect a battery and/or a capacitor in parallel.

To step down from a 12V car battery, a switched mode regulator is recommended. However, it should be encased in a metal box and the input and output connections filtered so they don't produce large amounts of RF noise which will reduce the radio sensitivity.

CPU

The CPU is a Texas Instruments MSP430 device. Unlike Mataki-Lite, the logs are not stored in the CPU flash memory. However the settings **are** stored in the CPU flash and will be lost when the device is re-programmed.

Flash Memory

Because of the need to store logs from many tags when in base station mode, the logs on Mataki-Classic are stored in a flash memory chip with a 32Mb capacity giving enough space for nearly 1 million log entries.

GPS
The GPS is a low power, high sensitivity device from GlobalTop (PA6C).

Light and Sea Sensors
Light sensor measurements can be set to be logged. An example use would be to detect when an animal goes into a burrow. A pin on the edge connector is used as an analogue input to detect sea water conduction to GND or other pins, this can also be logged.

Accelerometer
Accelerometer measurements can be set to be logged when tracking.

Temperature/Barometric Sensor
Temperature and absolute air pressure readings can be logged when tracking. The pressure sensor is sensitive enough to measure altitude if the pressure reading at sea level is known at the time of the reading.

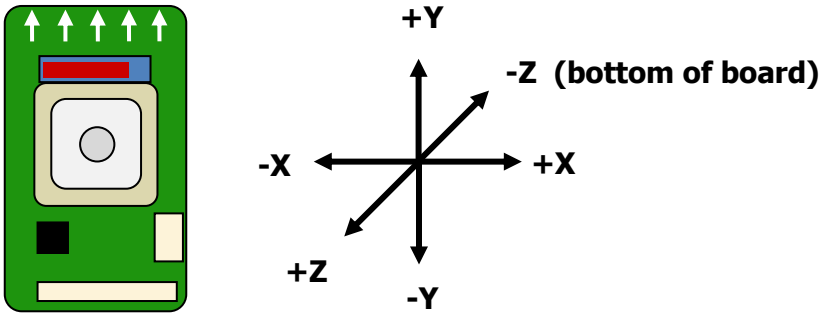
Magnetometer
While there is an on-board magnetometer, this is only used for testing and only accessible in the service mode.

Radio and Antenna
See section 4.6-4.8.

User LED
There is a red LED on the top of the board. This comes on briefly when the board is reset.

4.4. Accelerometer and Magnetometer Axes

Both the accelerometer and magnetometer axes are the same, and match the Android standard for devices. Relative to the tag, they look like so:



4.5. Battery

Mataki-Classic is supplied as a circuit board. You will need a battery for each tag and the means to attach it to your animal of choice. Because the animal dictates the attachment method and the size of battery, we do not supply these parts.

The power is supplied from a 3.6V Lithium-Ion or Lithium-Polymer rechargeable battery. The battery connector is a JST SM04B-SRSS-TB connector (Farnell part number 183-0839) with the battery connected on pin 1 (negative) and pin 2 (positive). There is also a small pair of holes next to the battery switch, where a battery can be directly soldered (positive and negative are marked on the bottom of the board).

It is recommended that pre-made cables are bought for this, such as: https://www.digikey.com/short/qjd0n9, as the tools for crimping are very expensive. There is also a small pair of holes next to the battery switch (shown below), where a battery can be directly soldered (positive and negative are marked on the bottom of the board).

WARNING

Check the polarity of the battery very carefully before inserting as connecting it the wrong way round is likely to cause permanent damage to the tag

4.6. Radio

The radio transceiver operates in the ISM (Industrial, Scientific & Medical) band. No license is required to transmit in this band provided you stick to the published rules. In Europe this band is 868-870MHz (generally known as 868MHz) and in America it is 902-982MHz (generally known as 915MHz). You must use the correct base frequency setting and antenna for the region you are using it in.

The radio operates as follows...

Tracker Application

- Period between base station contacts is set with the `setradio` command or the `sync` command can be used to initiate contact
- When contact is made, tracker tags automatically upload their logs to the base station
- A 10 minute timeout is used for loss of contact with the base station
- The time is set from the GPS

Base Application

- The radio is always on, in receive mode, so no contact settings are needed
- Collects logs from nearby tracker tags when a heartbeat is received
- After uploading logs, the tracker is told to reset
- Logs from trackers are stored in a cumulative log file which can be read using PuTTY and decoded using the log converter utility
- The `trackers` command gives an overview of the cumulative log
- The base station time needs to be set correctly for the log entries to be in the correct order and, if Mataki-Lite trackers are being used, to set their time. It has to be UTC (GMT) time as that is what GPS uses.

Service Application

- The radio is only used in the radio test. A single tag with ID 100 operates as a slave listening device which replies to incoming radio packets from other tags being tested.
- Extra commands are available to test the radio: `carrier`, `channel` and `txpower`

The part of the firmware that deals with radio messages supports different message protocols which you may see references to on the PuTTY terminal – this table describes all of the protocols used by Mataki-Classic tags...

Protocol	Function
1	Used to send/receive log text
2	Used for remote command requests
3	Used for remote command responses
10	Used for requesting missing logs to fill gaps
11	Used for sending missing logs for gap requests
12	Used for requesting a range of missing logs
13	Used for sending missing logs for range requests
91	Used for Master to Slave test messages in the Service App. radio test
92	Used for Slave to Master test messages in the Service App. radio test
100	Used to send heartbeats
101	Used to receive base station setup instructions

4.7. Radio Compatibility

Mataki-Classic radio messages are compatible with Mataki-Lite tags and mixed operation is supported (Classic and Lite operating at the same time using the same base station). The logs produced by Mataki-Lite and Mataki-Classic tags are identical.

All tags, whatever the type, must have unique IDs or the base station logs will mix up the tags. Compatibility between tags is only possible if the firmware versions are compatible. The following table shows which firmware versions are compatible with which PCBs and Mataki-Classic base stations. Each row is a compatible set and uses compatible radio messages.

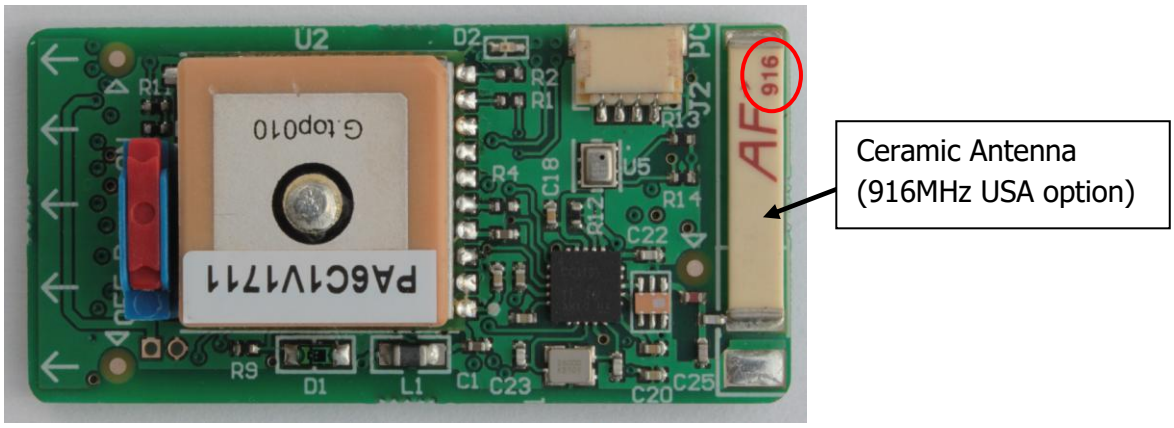
Mataki-Classic Firmware Version	Mataki-Classic PCB Version	Mataki-Lite Firmware Version
5.3.1	Up to V5.3	-
5.3.2	Up to V5.3	-
5.3.3	Up to V5.3	1.1.1
5.4.4	All	1.2.2

In general we recommend running the latest firmware on all tags. See Mataki-Classic Programming Guide for instructions on how to load new firmware.

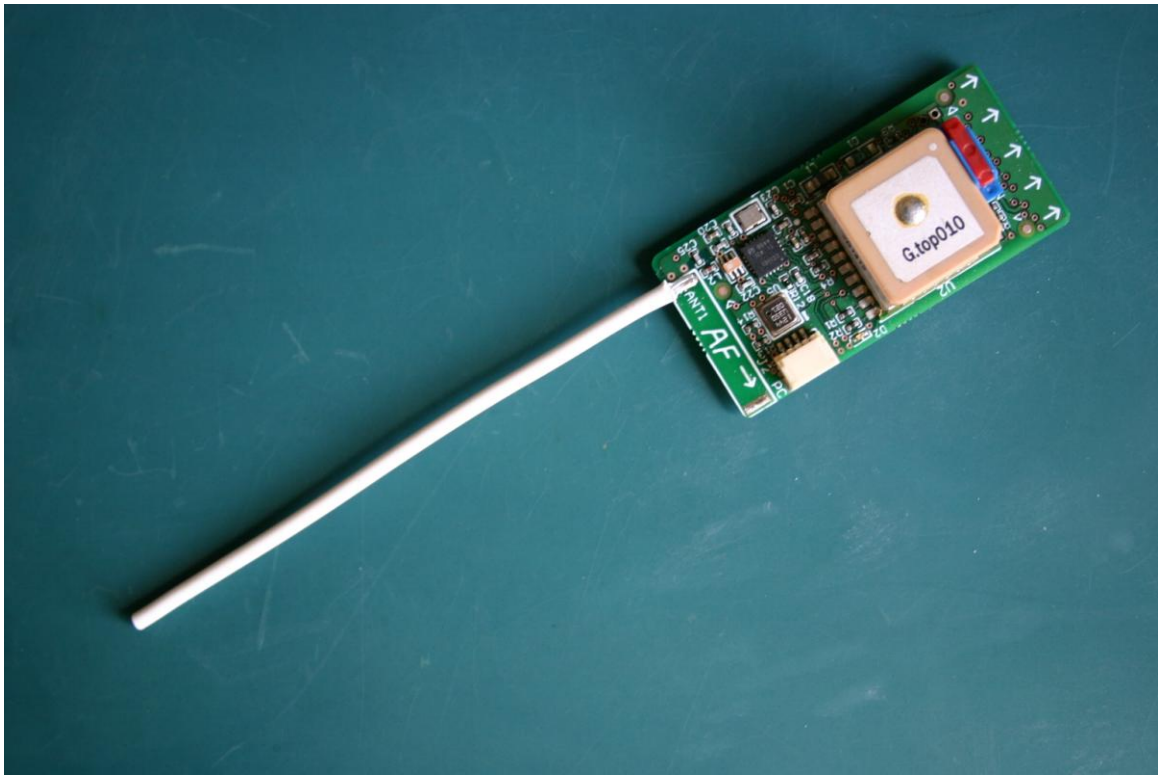
Mataki-Classic tags running V5.4.4 firmware are also compatible with Pi-Base base stations.

4.8. Antenna Options

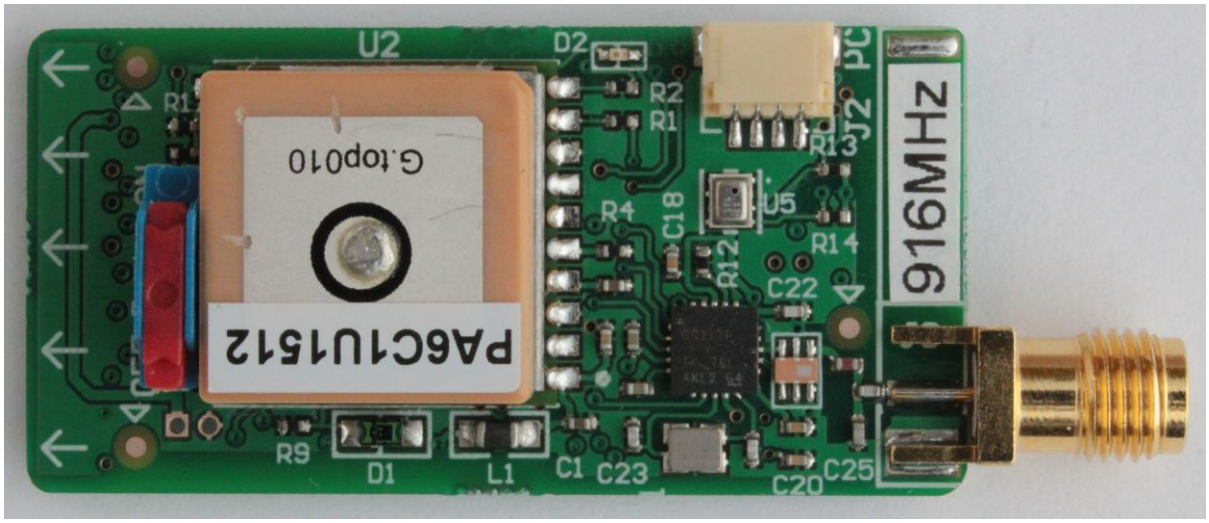
Mataki-Classic tags can be ordered with a variety of radio antennas. The smallest is the ceramic antenna shown below. It is available in 2 different frequencies indicated by the number on the part in the red circle (868 for Europe, 916 for USA).



The ceramic antenna can be replaced by a wire as shown below. This will typically have better performance than a ceramic antenna but is larger and can be heavier. If you fit it yourself, the correct length is 85mm for 868MHz (Europe) and 80mm for 915MHz (USA) measured from the start of the pad/wire.



V5.4 PCBs have the option to fit an SMA connector as shown below.



The SMA connector is intended for base stations, so a larger antenna can be used. Care must be taken not to exceed the radiated power limits at the antenna, so if you connect a high gain antenna for example, you may be operating outside the ISM rules. Due to the high frequencies used, long coaxial cables may result in a loss of signal - if your antenna is more than a few metres away from the tag, consider moving the tag closer.

The SMA connector can be supplied with new tags or fitted later. It is available from DigiKey, part number 931-1175. Note that only one (the lower) side of the SMA ground leg is used (on both PCB sides) as the other (upper) side overlaps with the ceramic antenna patch. V5.4 tags also have a 100pF DC blocking capacitor at the antenna connection so antennas with a DC path can be used. This means that almost any 50Ω antenna for the correct frequency can be used.

5. Base Station Communication

When a tag contacts a base station, a lot happens in a short space of time. This section is an overview of the process.

Two radio channels are used to communicate with the base station. The actual frequencies depend on the base frequency setting but the channel numbers are the same and are used in the following description. The basic idea is that tags contact the base station on one channel then switch to a different channel to upload their logs. During this process, other tags which try to contact the base station cannot interfere with the first tag's data transfer.

Initially the tag's radio is off. When the required time elapses the radio chip is turned on and a heartbeat is sent out...

- The tag broadcasts a heartbeat message on channel 16. The heartbeat contains basic information about the tag (ID, type, firmware version etc) and the number of log entries it has.
- The base station uses this information to compare with its own copy of the tag's logs and decide which log entries it doesn't have. To avoid searching its log every time, the base station keeps statistics on every tag which are built from the log when the base station starts up. If the base station has all the tag's logs, it sends a reset message to the tag causing the tag to reset itself.
- If there are logs to get, the base station sends a message to the tag to switch to channel 20. Then the base station pings the tag on the new channel to check the tag got the message and has switched channels.
- The ping message contains information about the base station (ID, type, firmware version etc). When the tag receives a ping message, it sends a ping response back.
- Once a connection has been established on channel 20, the base station sends a 'keep alive' message to the tag. This indicates to the tag how long it should wait before giving up if contact is lost. Different types of base station send different values due to the way they work. A Mataki-Classic base station sends 10 minutes as it might spend 10 minutes searching its own log without sending a radio message, whereas a Pi-Base base station sends 10 seconds as it has enough RAM to cache the whole log. Mataki-Classic tags running the Tracker Application ignore this message and always use a 10 minute loss of contact timeout period.

The base station then asks for the log entries it doesn't have. This is a two step process...

- In the first step, a 'range request' message is sent from the base station to the tag. This basically has a start and end point for the logs it wants e.g. Logs 101-225. The base station asks for every log between the highest entry it has and the tag's log size reported in the heartbeat message.

- The tag then sends back a message for each missing log entry with a short delay between each message. As each log entry is received, the base station adds the missing entry to its own log and updates the log statistics. The base station may not receive all of the logs but the next step takes care of the problem.
- In the second step, the base station sends a 'gap request' to the tag. This consists of a log entry number and a bitmap of the logs after that number identifying which logs the tag needs to send (gaps in the base station's log for this tag). Unlike a range request, this can quickly patch up the holes in the log because not all the logs are requested. However, due to the maximum size of the radio messages, the bitmap is limited to about 400 logs so multiple gap requests are required to patch up holes that are a large number of logs apart.
- As in range requests, the tag sends the requested logs to the base station.
- At the end of the process, the base station evaluates the data it has received and decides whether to ask again for any log entries it hasn't received. If not, the next time this tag is seen, the base station will try again.
- Note that the base station's log isn't in the same order as the tag's log and it contains logs for all the tags it has contacted. This jumbled log is unscrambled by the log converter script on a Mataki-Classic base station. On a Pi-Base base station, the logs are sent to a Mataki server which unscrambles the data and plots separate maps on its web pages for the different tags as the logs are received.

Lastly, the base station tells the tag to reset itself. The base station switches back to channel 16 and waits for other tags.

6. Start Up Messages

On start up, a large amount of text is displayed on the PuTTY terminal. We will explain it in sections. Some lines are time-stamped. The text below has had the time stamps removed for clarity. A lot can be learned from the start up messages - we recommend reading through this section as it contains information that is not explained elsewhere.

```
MATAKI Radio Tracker and Sensor Platform V5.4.4
```

This first line is the sign on message and firmware version.

V5.4.4 can be interpreted as follows: V5.4 means 'supports PCB version up to and including PCB V5.4' and the .4 on the end means this is the fourth release of firmware. So the next release will be V5.4.5 unless it supports another board e.g. V5.5.5. The last number never resets, so V5.3.27 has the same functionality as V5.4.27 except it supports V5.4 PCBs.

If the last number is zero, it is a development build, not a release build, and it may have unusual behaviour!

The version number can be obtained at any time from a PuTTY terminal with the use of the `ver` command.

```
##############################
###    DEVELOPMENT BUILD    ###
##############################
```

Development build only: This warning sign is displayed. Development builds may have unexpected behaviour. On release builds this message is not shown.

```
Copyright (c) 2014-2018 Debug Innovations Ltd.
Portions Copyright (c) Microsoft Corporation.
Portions Copyright (c) 2012 Freie Universitaet Berlin, Tomasz
Naumowicz.
All rights reserved.
```

The copyright information.

```
MATAKI PCB : V5.4
BMP280 temp/press sensor selected
LSM303 accelerometer selected
```

This identifies information about the hardware. The PCB number matches the PCB number on the back of the board. The firmware supports multiple PCB versions, temperature/pressure sensors, and accelerometers. It can then auto-detect which version of each of these it has and alter its behaviour accordingly.

```
App: 'Tracker' V2.2 for MSP430 on AGPS5
Compiled: 16:54:12.000 26-02-2018 UTC
```

This identifies the current firmware application (Tracker, Base or Service) running on the device, the version number, and when it was compiled.


```
Node ID: 1
```

At this point the node ID has been read from memory. Not that by this line the `[0:L]` has changed to `[1:L]`.

```
SMCLK: 3964928 Hz
```

This is an internal hardware clock that needs calibrating. It should be close to 4MHz.

```
Radio 0 On
 Radio 0 (CC1101): Part Number 0, Version 20
```

The radio chip is turned on and the part and version registers are read. The driver expects to see a CC1101 device.

```
 Radio base freq: 868000000Hz
 Radio freq. cal: 0Hz
```

The radio's frequency is set. The base frequency comes from the `fbase` setting. The frequency calibration value is always 0 on Mataki-Classic.

```
Radio 0 is in RX
Radio: Registered handler for protocol 1
Radio: Registered handler for protocol 2
Radio: Registered handler for protocol 3
```

The radio is in receive (RX) mode and radio handlers are registered. See the table in section 4.6 for details on the different protocols.

```
Initializing Log
 Flash Init...
 Found S25FL256S device
 Capacity: 32M bytes, 64K bytes/sector
Flash Init: OK
```

The log and flash memory are prepared for use. The flash memory device is reported alongside some basic details.

```
 Log: Total slots 932066 (slot size 36)
 Log: Slots used 2565
Log Engine Init: OK
```

The log is read. The total space available and current number of logs is reported.

```
 Radio: Registered handler for protocol 12
 Radio: Registered handler for protocol 10
```

More radio handlers are registered. See the table in section 4.6 for details on the different protocols.

```
Log Sync Init: OK
```

The log syncing has been initialised properly.


```
Initializing Temp/Press sensor
Initializing Accelerometer
```

The temperature/pressure sensor and accelerometer have been initialised.

```
Battery: 3.76V
```

This gives the current voltage of the attached battery. It can be read again later with the `batt` command. See section 4.5 for more details on batteries.

```
 GPS Init...
GPS On
```

The GPS module is initialised and then turned on.

```
 GPS: Trying 38400 baud
 GPS: Trying 9600 baud
 GPS: Detected at 9600 baud
 GPS: Reconfiguring to 38400...
GPS: Comms established
```

An auto baud rate detection process takes place to establish communication with the GPS module. The baud rate is then raised to 38400 to allow for a higher performance.

```
 GPS: NMEA Configuration...
 GPS: NMEA configured: GGL: 0 RMC: 5 VTG 0 GGA: 1 GSA: 0
      GSV: 0 GRS: 0 GST: 0
 GPS: Position Update Rate: 1 Hz
GPS Init: OK
GPS Off
```

The NMEA parser is configured and reports the interpreted NMEA data. The GPS is then switched back off.

```
Init sensors: OK
```

All of the sensors have initialised correctly.

```
Radio 0 Off
```

The radio is switched back off.

```
System init complete
```

The initialisation of the system has finished.


```
Startup delay set, press C to cancel
  -> Start in 1 mins
```

Only when start up delay is enabled: Gives a countdown to when the tag will start up main operation. As indicated, pressing the 'C' key will cancel the delay and the application will start immediately. The start up delay can be set and read with the `stdly` command.

```
Radio 0 On
Radio: Registered handler for protocol 101
```

The radio and a further protocol initialised. On the Base Application the radio stays on.

When running the Tracker Application:

```
GPS On
 Registered NMEA Handler
Radio 0 Off
```

The radio is turned off, the GPS is turned on and the first fix is attempted.

```
Accelerometer measurements will start in 10s
```

This says when the accelerometer will start taking its first measurements. This corresponds to the settings that can be set/read with the `setacc` command. By default the accelerometer is disabled, which gives an appropriate message.

```
Use the 'abort' command to stop logging
App. started
```

The application has started. On trackers, the `abort` command will disable logging.

```
Got GPS Time
```

When a valid GPS time is found: This message indicates the tag has got a valid time from the GPS satellites.

```
Got GPS Fix: 52.298381 -0.006581
GPS Off
```

When a valid GPS fix (position) is found: This message indicates the tag has got a valid position fix from the GPS satellites.

When running the Base Application:

```
Use the 'abort' command to stop the base station starting (30
seconds)
App. started
```

The hardware and firmware are fully initialised and the application code has started running. On a base station, there is a 30 second window where the `abort` command can be entered to stop the base from starting normal operation. If the base application is aborted, the base station will still display start-up messages and heartbeats from trackers but it will not respond to them. No more tracker log entries will be added so the base station log can be read without being modified by trackers.

7. Tracker Application Specifics

7.1. Sensor Setting/Reading Examples

7.1.1. Accelerometer

There are two commands when configuring the accelerometer, which are:

- `acc` (Note: This is only required for boards with a CMA3000 device)
    - In this example, the preset sample rate chosen is 40Hz with an 8g measurement range, so the issued command is `acc 1`
    - The response after issuing this command is: `[1:acc] OK`
- `setacc`
    - In this example, the accelerometer will be configured to start taking readings 600 seconds (10 minutes) after power up, with 5 seconds of readings every 300 seconds (5 minutes)
    - The issued command for this is `setacc 600 300 5`
    - The response from this is: `[1:setacc] {600,300,5} Changes saved, reset to activate`

When accelerometer readings are taken, no messages are displayed.

7.1.2. GPS

To configure the GPS:

- `setgps`
    - In this example, the GPS will be configured to start taking readings 30 seconds after power up, with readings every 300 seconds (5 minutes). The maximum time waiting for a fix is set to 60 seconds, and the GPS will be left on for 10 seconds collecting additional fixes after the first valid fix.
    - The issued command for this is `setgps 30 120 60 10`
    - The response from this is: `[1:setgps] {30,120,60,10} Changes saved, reset to activate`

A typical set of messages when taking GPS readings looks like so:

```
GPS On
 Registered NMEA Handler
Got GPS Time
Got GPS Fix: 52.298381 -0.006581
GPS Off
```

In this sequence the GPS is first turned on, with the NMEA handler registered to process the data. If the GPS time is different to the current tag time, the tag time is set to the GPS time and a fix is collected and logged. The GPS is then turned off.

7.1.3. Temperature/Pressure and Light Sensors

The frequency of temperature/pressure and light readings is tied to the frequency of GPS readings. When a GPS reading takes place, the measurements are taken. An example of the output after a reading of the temperature/pressure looks like this:

```
Temp/Pressure Logged
```

Light sensor readings are also taken simultaneously, but provide no text output.

7.2. Sync Process

When synchronising logs with a nearby base station, there are many steps that take place. Here, we go through them to show what happens.

```
Sending Heartbeat
Radio 0 On
```

The tracker sends out a heartbeat containing some basic details, which any nearby base stations will pick up and reply to.

```
Connecting to {1} on channel 20...
Uploading log to base, suspending tracking and all
measurements
```

A base station has responded to the heartbeat. Suspend all logging and measurements in order to start syncing logs with the connected base station.

```
Switching to base station mode on channel 20...
Type: Mataki-Base, FW: V5.4.4
Connected to {1} on channel 20
```

The tag is asked to switch to a new radio channel for the log sync. The base also sends some information about its firmware version, which is displayed.

```
Requesting newest data from {1}
```

The latest logs are now going to be synced.

```
Received 3021/3021 new entries from {1}
```

The number of collected logs is reported. In this instance, none of the logs failed to make it to the base.

```
Requesting missing data from {1}
```

At this point, the base asks for any missing data from its records about this tag.

```
Requested 0, received 0 (0 in total)
```

The number of missing logs requested and sent is reported.

```
Got enough data for now
Base station told us to reset
```

All the data that can be collected at this point has been collected. The base finishes the transaction by requesting that the tag performs a reset.

8. Base Configuration Specifics

8.1. Battery/Log Reports

Periodically, the base station will report its own battery level and the number of logs in the flash. A typical message of this type will look like so:

```
Battery: 3.80V, Log Size: 4440 (0%)
```

8.2. Sync Process

When synchronising logs with a connected tag, there are many steps that take place. Here, we go through them to show what happens.

```
Heartbeat from {1} [-76dBm +9kHz]: Uptime: 625(0h), Log Size:
3021, New Logs: 203, Batt: 3.78V
```

Received a heartbeat from a nearby tag. The signal level and frequency offset of the message are reported alongside log details and a battery level.

```
Type: Mataki-Classic, FW: V5.4.4
```

The firmware version of the connected tag is also reported.

```
Connecting to {1} on channel 20...
ping...
Connected to {1} on channel 20
```

In the first message, a command is sent to the connected tag asking it to switch to a new radio channel to sync logs. Radio pings are then repeatedly sent on the new channel until the other end has responded which causes the connected message to be displayed.

```
Ping RX at tag [-74dBm -11kHz] Reply RX at base [-74dBm +9kHz]
```

The ping response also contains details on the level and frequency offset of the ping received by the tag.


```
Requesting newest data from {1}
Local Seq Num:  1188
Target NodeID:  1
First Seq Num:  1
Last Seq Num:   3021
............................................................
............................................................
Received 3021/3021 new entries from {1}
```

Here, the base decides which logs need to collected in the range request. It displays the first and last sequence numbers, and sends the range request. Dots are printed whilst the data comes in to indicate data is being collected. A report on the data collected is then displayed.

```
Requesting missing data from {1}
Scheduled the operation. Wait for the schedule to complete!

3021 items remaining, searching log...
2589 items remaining, searching log...
2157 items remaining, searching log...
1725 items remaining, searching log...
1293 items remaining, searching log...
861 items remaining, searching log...
429 items remaining, searching log...

Schedule processing completed
```

Now a gap request needs to take place. The first step is to look through the logs for any missing entries. As the log is searched the number of logs left to look through is periodically reported.

```
Requested 0, received 0 (0 in total)
```

A message reporting the number of logs requested and received in the gap request phase is displayed.

```
Got enough data for now
Got 3021/3021 new and 0/0 missing from {1}
```

All the data that can be collected is now with the base. Display a summary of the collected logs.

```
Resetting tracker
```

A reset command is issued to the connected tag.


```
Returning to channel 16
Waiting for new trackers...
```

The sync is now complete. Return to the heartbeat channel and wait for the next tag.

9. Tag Start Up

9.1. Powering a Tag

Mataki-Classic tags can be powered from two sources...

- A Battery
    - Plug the battery into the tag (checking the polarity first!)
    - Switch the battery switch on the tag to the On position
    - Normal behaviour begins after initialisation (only takes a few seconds)
- A Support Board
    - Insert the tag into the support board
    - Switch SW1 (On/Off) into the On position to provide power to the tag (not necessary if a battery is already attached and the power is)
    - If a battery is attached, SW2 (Charger) will charge it

10. Important Settings

Please carefully check the following settings before tag deployment, as errors in these may cause issues for the tags functionality. The exact behaviour for the commands necessary is in the command reference section of this document.

10.1. Node ID

It is important to set a unique ID for each tag before deployment, or the base will get confused which tag it is communicating with. For help on which ID should be assigned to a tag, see the Mataki Platform Overview document.

To assign a tag an ID, use the `id` command at the command prompt. This value is stored in the flash and will be saved between device resets.

10.2. Radio Base Frequency

The base frequency for radio interactions (used as channel 0) must be set depending on the region where the tag will be deployed. The suggested bands are described in the radio portion of this document.

To set the base frequency, use the `fbase` command at the command prompt. The device requires a reset for the base frequency to be set. This value is stored in the flash and will be saved between device resets.

10.3. Start Up Delay

In order to save battery, a start up delay can be programmed onto the tag. This causes the tag to go to sleep for the specified time, only waking up when it is time to start taking readings.

This could be used, for example, to set a 1 hour delay while walking up to a site where the tags will be deployed with the batteries inserted instead of having to plug the batteries into the tags at the deployment site.

It may be cancelled through the PuTTY terminal by pressing the 'C' key.

To set the start up delay, use the `stdly` command at the command prompt to specify the delay time in minutes. This value is stored in the flash and will be saved between device resets.

10.4. Time

A tracker needs to know the time so it can accurately track when measurements or system events take place. In the Tracker Application, this is set when GPS time is obtained, to keep the time accurate.

A base station needs to know the time so it can give it to Mataki-Lite tags starting up, and keep an accurate chronology of tag heartbeats.

This is set with the `time` command, with time in the format: HH:MM:SS DD.MM.YYYY. Note that the time has to be UTC as that is what the GPS system uses.

See the commands reference for more details.

10.5. Format

When starting a new experiment it's usually a good idea to format a device with either the `format` or `f_reset` (format and reset) commands in order to remove any logs currently on the device.

Note: This process cannot be undone. Only format a tag if you are sure you have collected the data from.

10.6. Accelerometer (tracker only)

The behaviour of accelerometer readings must be set. The settings for this can be set with the `setacc` command. The fields available are: power up delay, time gap between measurements, and logging time. 0 can be used in the last 2 fields to disable accelerometer readings. On tags with a CMA3000 device, the use of the `acc` command may also be required. See the commands reference for more details.

10.7. Radio settings (tracker only)

The behaviour of attempted radio contact with a base station must be set. This is done with the `setradio` command. The available fields are: power up delay, time gap between RX, and timeout on no signal. See the commands reference for more details.

10.8. GPS (tracker only)

The behaviour of GPS readings must be set. This is done with the `setgps` command. The available fields are: power up delay, time gap between fixes, max wait for fix, and logging time after fix. See the commands reference for more details.

11. Collecting Base Data

11.1. Collecting Logs

To get the data collected by a Mataki-Classic base station follow these steps:

- Ensure PuTTY has log output enabled (see PuTTY installation guide for how to do this)
- Insert the base station tag into the support board and reset it
- Enter `abort` when prompted to stop the tag behaving as a base station so other tags can't interrupt the reading process
- Enter the `read` command into terminal (this could take some time to finish depending on how many logs are on the tag)
- Check PuTTY log file contains the data displayed

11.2. Converting Log Data

Once the data has been extracted from the base station tag, the raw data needs to be converted to more suitable files. A Python command line utility called `ConvertMatakiLogs.py` has been provided to automate this. Use it like the example below to output the converted data to a .zip file in the same directory:

```
C:\> ConvertMatakiLogs.py example/data.log
```

WARNING

Once a base has been formatted to remove the logs the data is non-recoverable. Only format the base when you are sure the data has all been extracted.

12. Commands Reference

Below is a list of the commands and a description of how they work. Where argument numbers and syntax are omitted, the commands only perform one function and can be accessed with just the command and no arguments (e.g. entering `help` without any arguments will display the help text). If a command is not available on all applications, the limitations are described under the command. The `help` command only displays valid commands for the current application.

---

abort

Tracker & Base Application Only

Tracker Application: Stops all automatic logging. Still allows user to interact with the device without scheduled events taking place, which is useful for testing.

Base Application: Stops the base responding to radio messages from trackers. The base station should be put into this state when the log needs to be read.

---

acc

Tracker Application Only

This is only applicable to CMA3000 devices.

0 arguments: Gives the current sample rate and measurement range, and the valid settings for this command, like so:

ACC mode: Sampling Rate: 40 Hz, Range: 8 g
Usage: acc [Sampling Rate (Hz) and Measurement Range (g)]
1: {40,8}
2: {100,8}
3: {400,8}
4: {100,2}
5: {400,2}

1 argument: Sets the sampling rate and the measurement range to one of 5 presets for debugging with the accelerometer. The 5 presets (that can be used with the values 1-5) are:

- 40Hz, 8g
- 100Hz, 8g
- 400Hz, 8g
- 100Hz, 2g
- 400Hz, 2g

Syntax:

```
acc [mode]
```

Increasing the sample rate of accelerometer readings will increase battery usage.

---

acci

Service Application Only

Performs an interactive accelerometer test, where a live readout of the current accelerometer measurements in each axis is displayed.

"Failed to power up" is given if the accelerometer device will not start. "Read failed" is displayed if at any point a reading is unable to be taken.

It is impossible to break out of this mode. A device reset is required.

app

Prints the current application loaded onto the tag, alongside the version number and when it was compiled.

An example is something like:

```
'Tracker' 2.2 compiled 16:54:12.000 26-02-2018
```

base

Tracker Application Only

0 arguments: Switches into a mode to listen to base stations commands on the current channel. The use of this function is deprecated.

1 argument: Switches into a mode to listen to base station commands on the given channel. The use of this function is deprecated.

Syntax:

```
base [channel]
```

batt

Takes a reading of the current battery voltage then displays it in volts. For example:

3.98V

carrier

Service Application Only

Generates a carrier wave on the current channel.

Care must be taken when generating carrier signals, as local radio laws may prevent generating them for too long at too high a power. Check local regulations before using this.

The only ways to stop the signal are resetting the tag, or using the `channel` command to change the radio channel.

channel

Service Application Only

0 arguments: Returns the current radio channel.

1 argument: Sets the radio to a given channel.

Syntax:

```
channel [new channel]
```

fbase

0 arguments: Gives the current radio base frequency in MHz.

1 argument: Sets the new radio base frequency in MHz. Only positive integer inputs are valid.

Syntax:

```
fbase [new base frequency]
```

Careful selection of the base frequency is important to adhere to local radio laws. See the radio section of this document for details on which base frequencies should be used.

fix

Service Application Only

Tries to obtain a GPS fix. NMEA sentences are displayed. Does not automatically switch off after a fix is found.

format

Removes all logs from the current tag.

Warning: This process is irreversible. Do not format the tag unless you are sure you want to remove the logs.

f_reset

Removes all logs from the current tag, followed by a reset.

Warning: This process is irreversible. Do not format the tag unless you are sure you want to remove the logs.

full

Service Application Only

Performs a thorough flash memory test to check its health.

Displays "Page n FAIL" (where n is the page it failed at) and terminates if there was a fault in any page.

Note: Any data stored in the flash will be lost.

gps

Tracker & Service Application Only

0 arguments: Gives the valid actions for this command, like so:

Usage: gps [operation]
1: Full NMEA debug mode (one way, reset to disable)
2: Toggle NMEA preview mode
3: Toggle Power of the GPS Chip
4: Toggle NMEA Debug Handler
5: Enable GPS, Start Logging (one way)
6 X: Set the position update rate to X (X = 1, 4, 5, or 10)

1 (2 on action 6) argument(s): Performs one of 6 GPS debugging actions:

- 1: Full NMEA debug mode
  - Powers on the GPS module
  - Sets the position update rate to 10Hz
  - Enables the display of raw NMEA data
- 2: Toggle NMEA preview
  - Toggles the display of raw NMEA data
- 3: Toggle GPS power
  - Toggles the power state of the GPS module
- 4: Toggle NMEA debug
  - Toggles the display of processed NMEA sentences
  - On a checksum failure, the message "CHKSUM FAIL" is displayed
- 5: Enable GPS, start logging
  - Powers on the GPS module
  - Enables logging of NMEA sentences
- 6: Set the position update rate
  - Requires a second argument equal to either 1, 4, 5, or 10
  - Updates the update rate of positions taken with the GPS to the given value in Hz.

Syntax:

```
gps [action][update rate (if action 6)]
```

help

Displays a list of valid commands for the current firmware application loaded onto the tag together with a short description.

id

0 arguments: Gives the current node ID of the tag.

1 argument: Sets the node ID of the tag. Rejects invalid IDs. See Mataki Platform Overview for details on picking IDs.

Syntax:

```
id [new ID]
```

light

Takes a reading of the current light level from the light sensor and gives the ADC value. Possible values are between 0-4095 (where 0 is dark and 4095 is bright).

logsize

Gives the amount of log space used.

An example return message would be something like:

```
158 of 932066 slots used (0%)
```

magi

Service Application Only

Performs an interactive magnetometer test, where a live readout of the current magnetometer measurements in each axis is displayed.

"No magnetometer found" if unable to find the magnetometer device. "Failed to power up" is given if the magnetometer device will not start. "Read failed" is displayed if at any point a reading is unable to be taken.

It is impossible to break out of this mode. A device reset is required.

read

Prints a list of all the logs on the current tag. This may take quite a while depending on the number of logs stored.

For tracker or service mode tags, this is all the logs collected since the flash was last formatted. For base mode tags, this contains all the logs of tags that it has contacted.

This is the command used for getting data from base stations. To see how this process works, see the collecting base data section in this document.

readbin

Similar to the `read` command, but outputs the logs in a binary format.

reset

Forces the device to reset.

rr

Gives the reason for the last reset.

The valid reason codes and their meanings are:

- 0: Reason unknown
- 1: Watchdog guard violation
- 2: Requested (e.g. the `reset` command)
- 3: Radio requested
- 4: Watchdog

time

0 arguments: Gives the tag's stored time. An example time would be:

14:18:08:123 22.02.2018

1 argument: Sets the new time for the tag. The input format is:

HH:MM:SS DD.MM.YYYY

Syntax:

`time [new time]`

Important:

All times are in UTC. Using local time will not work because the trackers are set to UTC when they get GPS time. To maintain chronology of events/logs, base stations must also be set to UTC time.

tp

Takes a reading from the temperature and barometric sensor and give the reading in Celsius and Pascal.

An example return value is:

Temperature : 20.0 C
Pressure : 100200 Pa

If the reading failed, then the message "Fail" is returned.

trackers

Base Application Only

Displays a list of trackers that have at least one log entry stored on this base, alongside the number of log entries, how complete the stored log is, and the last fix (if there is one).

An example list may look like:

Tag 1 : 57 log entries (100%)
Tag 9 : 551 log entries (100%), Last Fix: 50.979093 -0.000858 at 15:05:42.000 07-03-2018

txpower

Service Application Only

0 arguments: Returns the current radio transmit power.

1 argument: Sets the radio transmit power. The accepted values are (in dBm): -30, -20, -10, 0, 10, and 12.

Syntax:

```
txpower [level]
```

setacc

Tracker Application Only

0 arguments: Gives the current settings for accelerometer readings. Any values that are 0 represent infinity (e.g. power up delay of 0 means the device will never switch on). An example is shown here:

```
Power up delay: 60s
Gap time:       60s
Log time:        5s
```

3 arguments: Changes the accelerometer reading settings. Any value set to 0 represent an infinite time. The settings are (all in seconds):

- Power up delay
  - How long after the device first starts the first measurement should be taken
- Gap between measurements
  - How long to sleep between measurements
- Logging time
  - Maximum time to spend measuring and logging acceleration data

Syntax:

```
setacc [power up delay][gap between
       measurements][logging time]
```

setgps

Tracker Application Only

0 arguments: Gives the current settings for the GPS. Any values that are 0 represent infinity (e.g. power up delay of 0 means the device will never switch on). An example is shown here:

```
Power up delay: 60s
Fix Gap:        120s
Max. Wait:      60s
Log Time:       10s
```

4 arguments: Changes the GPS reading settings. Any value set to 0 represent an infinite time. The settings are (all in seconds):

- Power up delay
  - How long after the device first starts the first heartbeat should happen
- Gap between fixes
  - How long to wait between turning the GPS on and trying to get a fix
- Maximum wait for fix
  - The maximum time to wait for a valid fix from the GPS module
- Logging time after fix
  - How long to leave the GPS module on for after getting a valid fix (therefore collecting a few more fixes)

Syntax:

```
setgps [power up delay][gap between
       fixes][max wait for fix][logging
       time after fix]
```

GPS is a very battery intensive operation. Carefully consider how often you want to take readings.

setradio

Tracker Application Only

0 arguments: Gives the current settings for the radio. Any values that are 0 represent infinity (e.g. power up delay of 0 means the device will never switch on). An example is shown here:

```
Power up delay: 60s
RX gap:         60s
Timeout:        10s
```

3 arguments: Changes the accelerometer reading settings. Any value set to 0 represent an infinite time. The settings are (all in seconds):

- Power up delay
  - How long after the device first starts the first heartbeat should happen
- Gap between RX
  - Time between sending heartbeats
- Timeout
  - How long to leave the radio on waiting for a response from a base

Syntax:

```
setradio [power up delay][gap between
         RX][timeout on no signal]
```

More frequent heartbeats will increase battery usage.

stdly

0 arguments: Gives the current start up delay of the tag in minutes.

1 argument: Sets the start up delay for the tag in minutes. Only takes effect after a restart.

Syntax:

```
stdly [start up delay in minutes]
```

Start up delay keeps the CPU in a low power state for the time specified. On a connected PuTTY terminal, a message saying "Startup delay set, press C to cancel", followed by a time till the device will start, will show.

This is useful when inserting the batteries just before attaching the tag to the target animal is impractical. It has a negligible effect on the battery life.

suppress

Tracker Application Only

0 arguments: If GPS down time is configured, then it is displayed. An example is shown here:

Current setting: {10,13}

2 arguments: Configures GPS down time on a tag. If configured, then the GPS is not turned on between 2 sets of given times. An example usage might be if the target animal is asleep between 2 points, and taking GPS measurements would be wasteful of battery.

The arguments are 2 24 hour clock times (only whole hours are accepted), indicating the start and end times of the GPS suppression. An example might be "suppress 10 13" which would turn suppress the GPS between the hours of 10am and 1pm.

Syntax:

`suppress [`*`start time`*`][`*`end time`*`]`

sync

Tracker Application Only

Sends a heartbeat and attempts to sync with a base station. This is normally automatically handled (set up by using the settings controlled with `setradio`).

uptime

Gives the time elapsed since the device was started. An example would be:

01d 10:53:06

ver

Gives the firmware version. An example would be:

V5.4.4

V5.4.4 can be interpreted as follows: V5.4 means 'supports PCB version up to and including PCB V5.4' and the .4 on the end means this is the fourth release of firmware. So the next release will be V5.4.5 unless it supports another board e.g. V5.5.5. The last number never resets, so V5.5.27 has the same functionality as V5.4.27 except it supports V5.3 PCBs.

