Biot Motion Tracking System Architecture Description and API Specification

Licence

Considerations

1. The system is currently a prototype undergoing active development.

2. I intend that the ideas, hardware, software and associated documentation developed as part of this project be available for anyone to replicate and use in the spirit of the MIT Software license.

Basically I mean that to mean: if you want to build, implement, experiment with, use, market, make money from, donate expertise in, sell expertise in or modify a system based on this work, you should be free to do so, providing you appropriately acknowledge its source and respect the above intention.

The Purpose of this document

This document defines the basic architecture of the system and specifies a set of APIs that describe how the components should communicate with each other.

Defining the APIs allows for various hardware and software components of the system to be changed and developed without breaking the rest of the system as the modified components will still communicate using the same protocols.

Table of Contents

Document History

Date	Version	Comments
2016-09-23	release-v-1.0	JK

Overview

The system is designed to track the physical orientation of objects.

The primary focus is on tracking and recording the movements of human limbs however the system may also be used for a variety of applications where there is a need to measure and record the movements of assemblies of objects.

The system measures individual limb orientation using small sensors attached to each of the limbs (or components) being monitored. Each sensor (called a Biot node or 'Biot') contains an Inertial Measurement Unit (IMU) that is used to determine the its current 3 dimensional orientation in space.

Each Biot communicates its orientation to a software interface that collates the information from all the participating Biot Nodes.

A user interface application interacts with the software interface via a REST based Web Servive API to provide features such as:

1. a visual display of the current orientation of the limbs (or components) being tracked.

2. the ability to create and display graphs describing the orientation of limbs and changes in limb orientations over time.

3. the ability to compare orientations and changes in orientation against a previously recorded (or pre defined) set of movements.

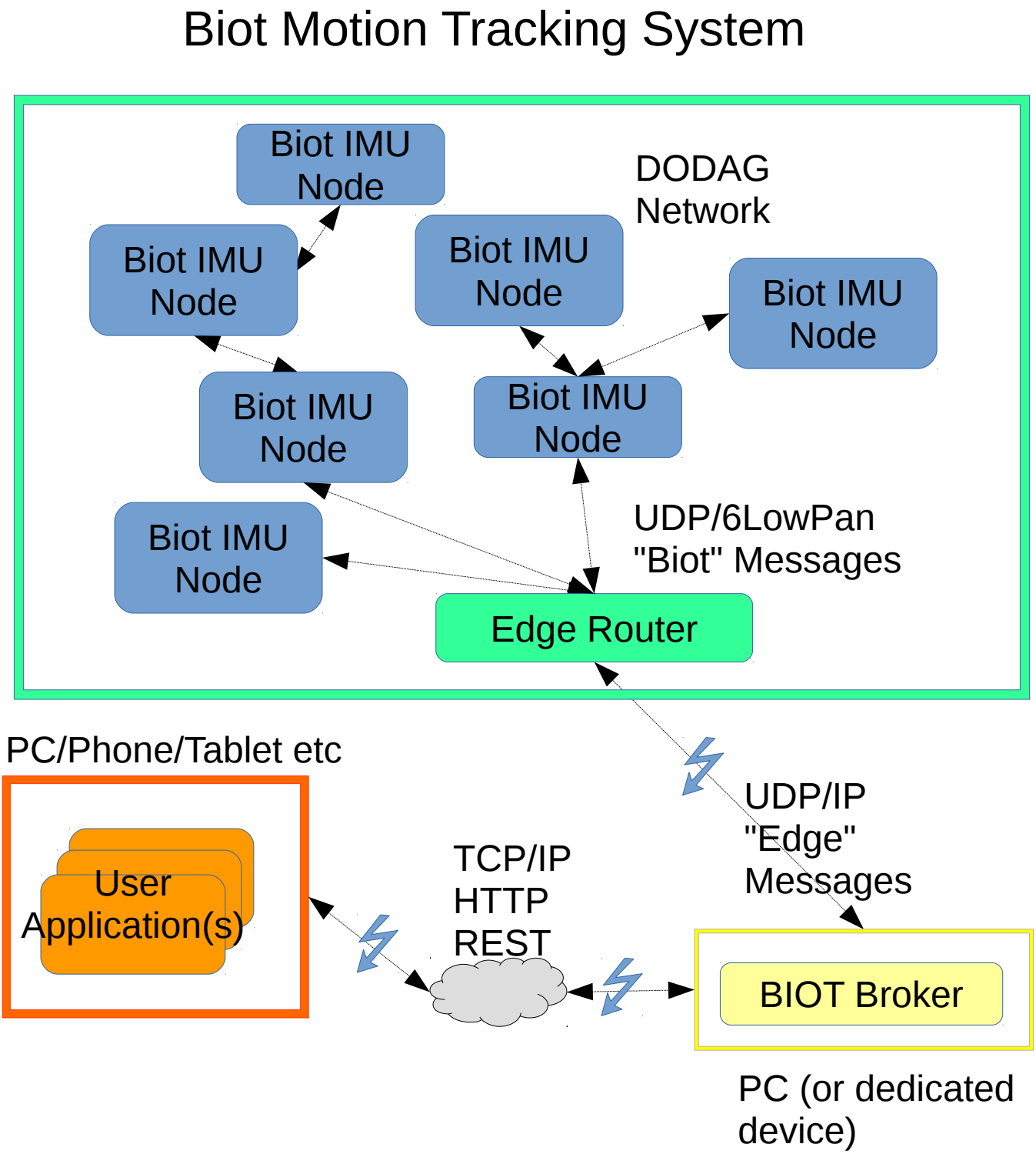
4. the ability to export numerical movement data for use with other analytical tools

5. the ability to record the limb movements for later playback and analysis.

Whilst the motivation behind this project is to produce tools that can assist in therapies for people with disabilities or physical injuries, it is not intended that the system be limited to only those kinds of applications.

More Detailed Description

Drawing 1: Components of the System



The Biot Nodes or 'Biots'

The system assumes there will be at least one (and typically a number of) Biot Nodes attached to various limbs (or other objects) whose orientation is to be tracked.

Each Biot is powered by its own battery. A Biot consists of several components all contained on a single small Printed Circuit Board (PCB). The components include:

- an IMU device that uses a combination of tiny electronic gyroscopes, accelerometers and magnetometers that respond to the physical orientation of the Biot node.
- a microprocessor that controls the IMU device and is constantly reading and processing the IMU outputs to generate a mathematical model representation of the node's current orientation.
- a very low power wireless device that can communicate with other components in the system.
- a power and a status LED that by flashing at various rates can be used to indicate the node's connection status and to visually identify itself in response to a request from a user.

The Biots communicate via a 6LowPAN wireless network. 6LowPAN is a wireless network protocol designed for use with very low power wireless devices. Being low power the physical distance a node can reliably communicate over is quite limited however the protocol is designed to be able to dynamically route and relay messages between intermediate neighbouring nodes allowing the network to potentially span greater physical distances than any individual nodes wireless range.

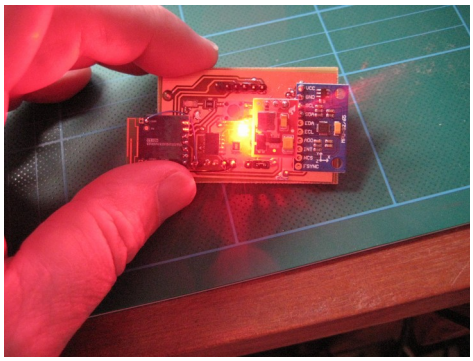


Illustration 1: An early prototype Biot Node

The Edge Router (or 'ER')

The 6LowPAN wireless network has one special node that is a central point to which all Biot data is sent, this special node is called an “Edge Router”. Nodes may communicate directly with the edge router if they are within range or if not can pass data to the edge router via intermediate Biot nodes.

In addition to communicating with the Biots, the Edge Router also has a UDP/IP interface to allow access from conventional network nodes and devices. The Edge Router acts as a gateway between the network of Biots and the wider world.

The Edge Router is a small device that can either be mounted somewhere on the body or nearby to the network of Biot nodes ('nearby' being roughly within 10m-20m).

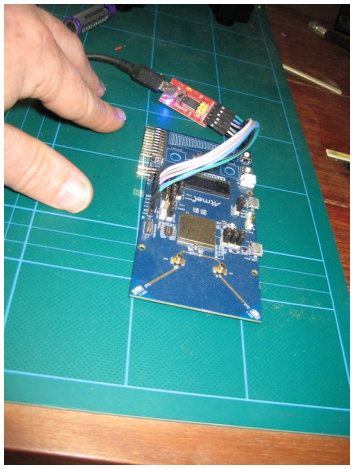


Illustration 2: The prototype Edge Router

The Biot Broker Application (or 'Broker')

The Biot Broker is a software service that sends and receives messages between other applications and the Biot Node network. It talks to both the Edge Router and to other applications (such as a User Interface Application)

The Biot Broker provides a REST Web Service API that UI developers can use to get information about the orientations of the various Biot nodes and to send control requests to the Biot network.

The Broker is a Node JS application that uses the Node Restify module. It currently runs on the same PC as the User Interface however it could potentially be set up to run on the Edge Router at some point in the future.

The User Interface Application

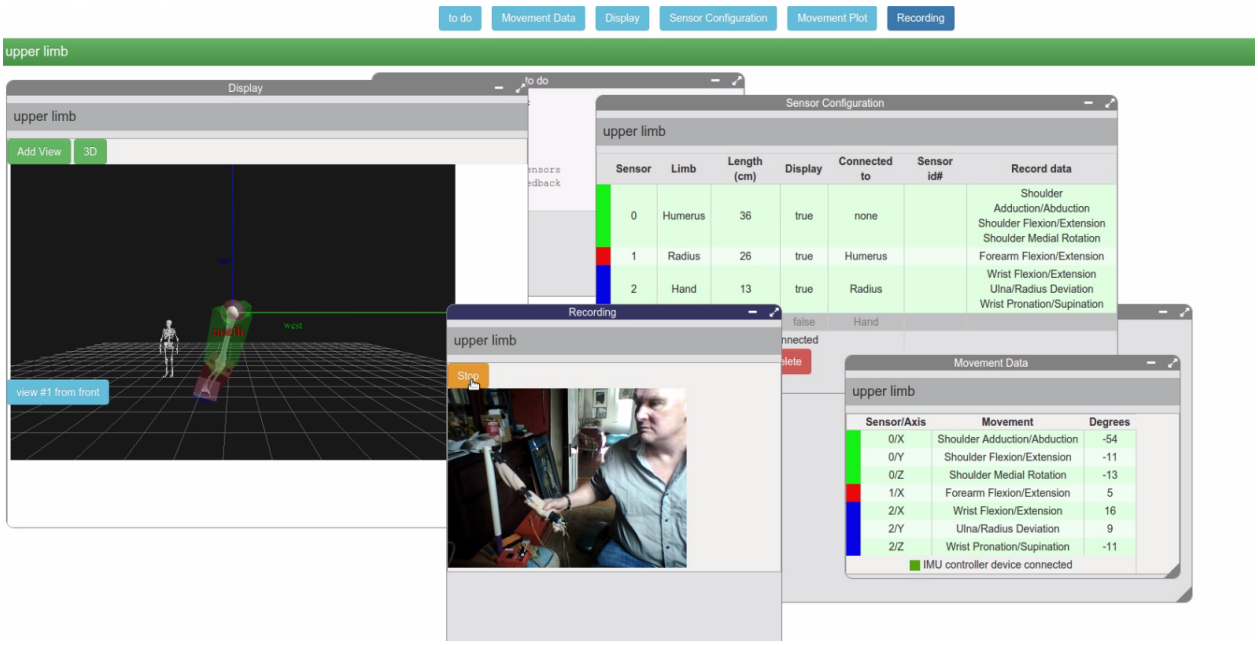
The Biot Broker Web Service API potentially allows a variety of different User Applications to read data from and control the behaviour of the network of Biot Nodes.

The current system implements a User Interface Application that provides a 3D representation of limbs with the ability to control, monitor and record the motions of the limbs. It is implemented as a Browser Based Web Application requiring no special software to be installed on the user's computer and also allowing the UI to be viewed on various devices like phone's, tablets etc.

Currently the UI being included as part of the standard system is a web browser application hosted somewhere that has network connectivity to the Biot Broker application. It uses the Angular2 Application Framework and is written in TypeScript.

Although the system currently provides a default UI, there is nothing to stop anyone writing their own UI applications. Application developers can access the Biot system using the Broker REST API.

Illustration 3: Early prototype of the User Interface



Intended Future Simplification of Structure

Currently the system is set up using a PC to host both the Broker and the application that provides the UI.

At some point the edge router and the host the Biot Broker runs on should be set up together on a small physical device such as Raspberry Pi or similar. Currently they are implemented as separate systems, with the Biot Broker software running on a PC and the PC connected by cable to the Edge Router device. The software demands of the Biot Broker could be handled by a small device like a Raspberry Pi or similar and this device could also potentially be used to implement the Edge Router.

This device could also provide the web server and host the backend code that provides the UI meaning no special hardware/PC setup outside of the Biot Nodes and the Edge Router device would be required.

This means that once the Biots are connected to a person's limbs and once the combined device (that now would provide the ER and host the Broker and Web Server components) is turned on, user's would just need to go to the Web Address provided by the device to start using the system.

Specification of Communication Interfaces and behaviours.

Biot Node Functionality

Each Node runs the RIOT OS operating system (see: https://riot-os.org/).

Each Node has, for development and diagnostic use, an ATMEL Cortex programming header for updating the firmware and a serial port for connecting a terminal using an FTDI cable. The terminal allows access to a simple command line shell running on the Biot. Normal operations don't require these features to be used.

A node will associate itself with a suitable 6LowPAN Biot network when it detects one.

A node constantly keeps track of its currently calculated orientation.

A node will transmit unbidden its current orientation, calibration values and status at regular intervals to the Edge Router

A node can dynamically calculate and update correction parameters for its magnetometers to account for hard and soft iron errors.

A node can set its magnetometer correction parameters to the values sent to it from the Edge Router (to allow the system to cache calibrations between sessions for faster start up).

A node maintains its own current 'system time'

A node will adjust and synchronise its own system time to be the same as the Edge Router's system time if sent a time value from the edge router.

A node normally blinks its status light every second (using its system time). All nodes that have synchronised to the same system time therefore will blink in unison.

A node can turn its status LED on or off or rapidly flash it for a few seconds to make itself temporarily conspicuous.

Edge Router Functionality

The Edge Router is a device that can communicate on both the 6LowPAN Biot Node Network and a UDP/IP network. It acts as a physical gateway between the Biot Node mesh and the wider world (normally the "wider world" would be a Biot Broker application – see further down for information on the Broker).

The Edge Router acts as the 6LowPAN DODAG Root node.

The Edge router will translate control messages sent from the Broker application and direct them as appropriate to one or more of its Biot nodes.

The Edge Router will translate and pass on data sent to it from Biot nodes to the Broker.

The Edge router will periodically instruct the Biot Nodes in its mesh to synchronise their system time with the ER's system time.

Biot Broker Functionality

The Biot Broker is a software service that provides a REST Based Web Service API to User Applications that wish to obtain data from and to send configuration to, a mesh of Biot Nodes.

The Broker keeps a dynamic cache of Biot Node data built and maintained from any data messages it receives from the Edge Router.

If sent appropriate REST requests from a user application, the Broker can return data on one or more of the Biot Nodes back to the User Application based on the cached information it holds.

If sent appropriate REST requests from a user application, it can construct and send Edge Control requests to the Edge Router to control the Biot Nodes.

Component Interactions

Types of Interactions

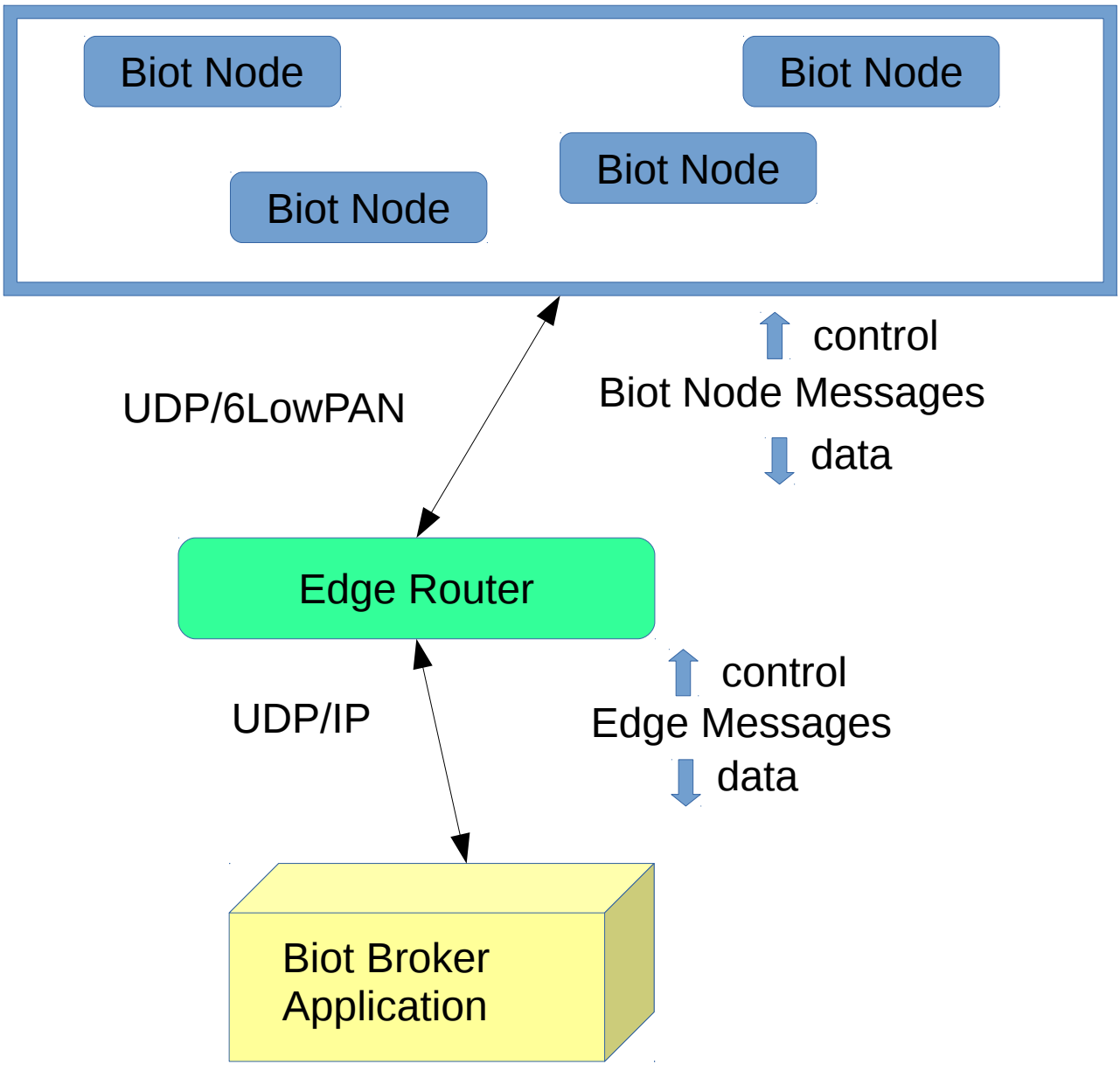
There are 3 types of messages/communication protocols between the various components.

1. **Biot Node Messages** (for communication between the Biots and the ER)

2. **Edge Messages** (between the ER and the Broker)

3. **REST API** methods (between User Interfaces and the Broker).

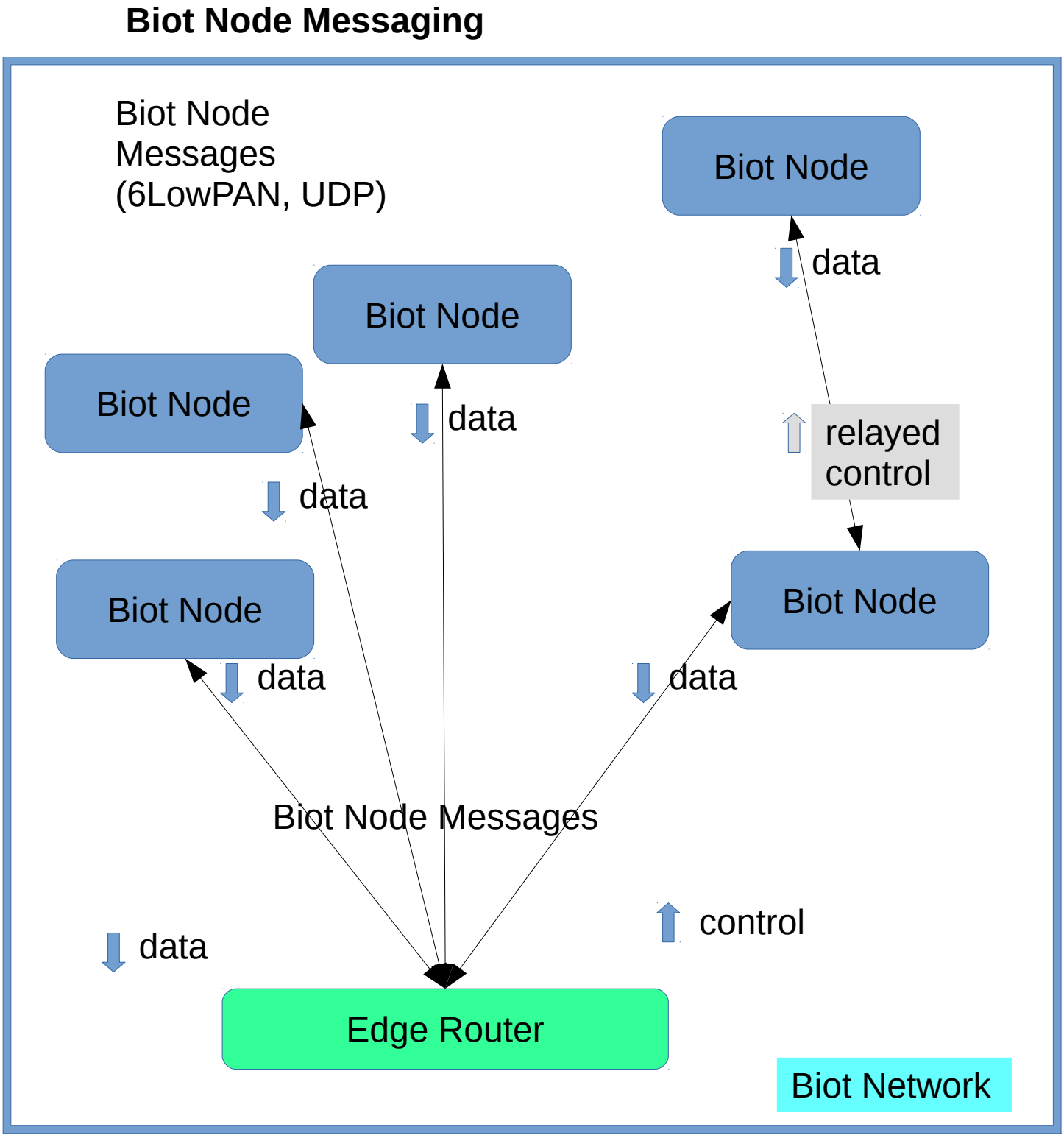
The first 2 protocols (Biot Node and Edge messages) involve sending and consuming messages. The third is a call/response API style interface.



Biot Node communications:

Biot Node's can only communicate with an Edge Router (or with other Biot Nodes if there is a need to relay messages).

Drawing 2: Biot Node Communications



Biot node communications are all UDP using the 6LowPAN protocol. With UDP there is no guarantee of messages being delivered.

The most important data flow in the system is the maintenance of up to data node orientations.

For this reason, rather than the ER sending nodes a request for their data and waiting for a response (where there is a risk of the request and/or the associated response being lost), the Biots autonomously (and regularly) push their current orientation (and other) data to the wider system to attempt to keep the system up to date with each Biot's current state.

Types of Biot Node Messages

There are 2 categories of **Biot Node Messages:**

Data Messages and **Control Messages**.

Data messages are generated by Nodes and directed to the Edge Router..

Control messages come from the Edge Router and are directed to one or more Nodes.

The messages consist of 2 fields (message type and message value) separated by the '#' symbol.

Biot Node Data Messages

Data messages contain either Orientation, Magnetometer Calibration or Status data. They are identified by a 2 character code, for future enhancements, the first character should be the letter 'd' (for data) to help when diagnosing issues.

Examples of Data messages.

Data Message	example	description
Orientation	`do#653472:-2.987:0.88:1.1000:0.289`	contains node timestamp and orientation w,x,y,z quaternion values
Calibration	`dc#-89:-86:-82:88:3:54`	contains node magnetometer calibration values
Status	`ds#111:200:1`	contains IMU DOF settings, the current unsolicited data update time interval and auto calibration mode

Data Messages are sent autonomously from nodes at a rate that is configurable for each node.

Biot Node Control Messages

Control Messages are used to change a Node's behaviour. They can be used to tell a node to:

- adjust its LED display
- adjust its system time
- adjust its IMU DOF settings
- adjust its current calibration parameters
- adjust its frequency of data sending
- reboot itself

They are identified by a 4 character code, for future enhancements, the first character should be the letter 'c' (for control) to assist in diagnosing issues.

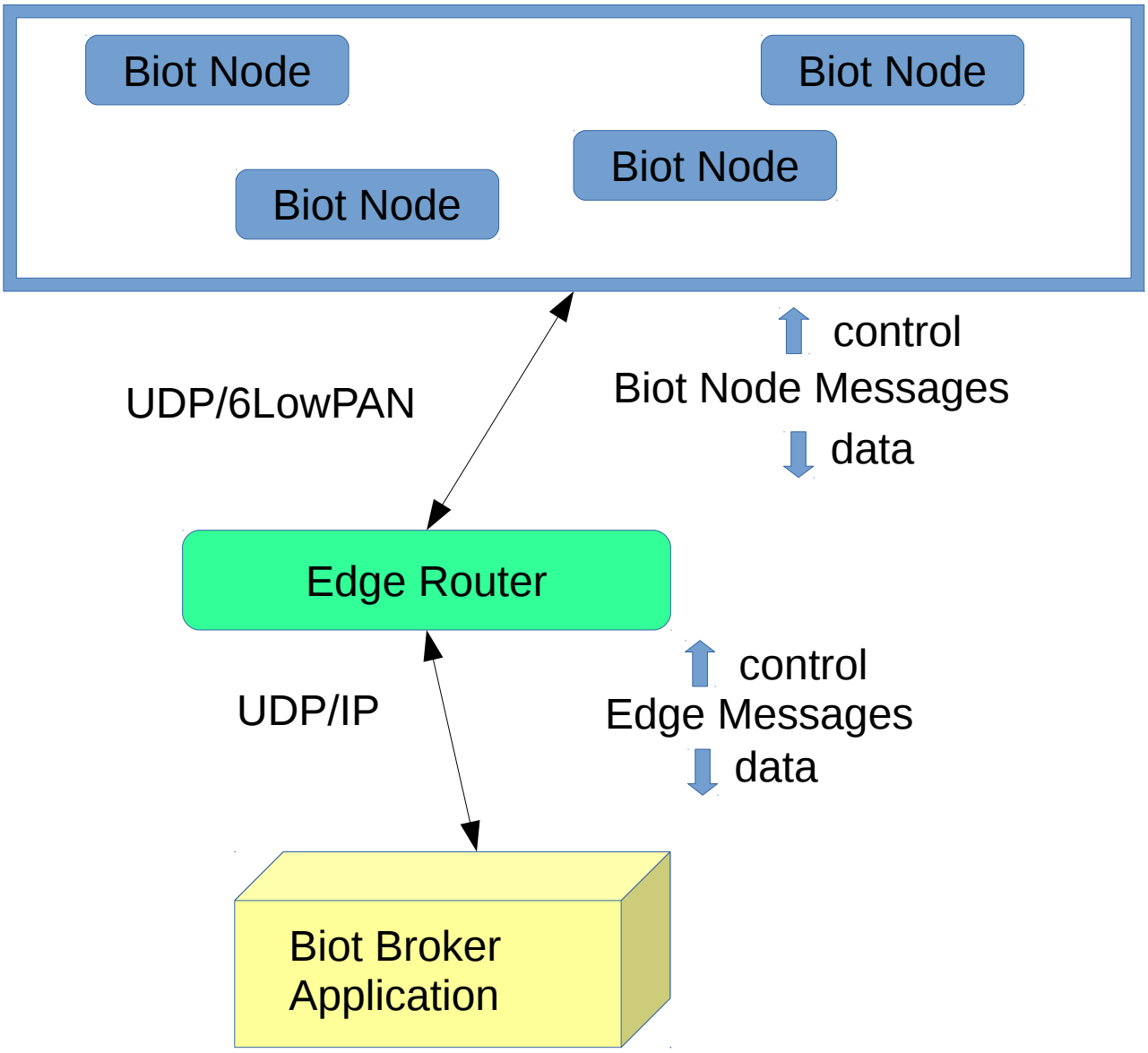
Examples of Biot Node Control Messages:

Control Message	example	description
LED display	`cled#2`	turn LED off(0), on(1), normal blink(2) or rapid blink(3)
System Time	`ctim#653483`	set node system time to given value
Set IMU DOF	`cdof#111`	set node gyros, accelerometers, magnetometers on(1) or off(0)
Set Magnetometer Calibration Values	`ccav#-89:-86:-82:88:3:54`	Set the node's calibration values
Set Magnetometer Calibration Mode	`cmcm#1`	Change auto calibration mode 0:off, 1:on 2:reset and on, 3: reset and off
Set unsolicited data update interval	`cdup#200`	set how long a node waits between sending unsolicited node updates (0 = do not send any)
Reboot	`creb#`	Node should reboot itself

Edge Router communications:

The Edge Router communicates with both Biot Nodes (using 6LowPAN UDP) and the Broker Application (using UDP/IP).

Drawing 3: Edge Router Communications



Biot Node Messages (UDP/6LowPAN) between the Edge Router and Biot Nodes are described in the previous section (Biot Node Messages).

Communication between the Broker Application and the Edge Router are called "Edge Messages" and consist of text strings sent via UDP/IP. They are similar in format to Biot messages however they have an extra '#' separated field to record the associated Biot IPv6 network address (if applicable).

Types of Edge Messages

As with Biot Node Messages, there are 2 categories of **Edge Messages:**

Data Messages and **Control Messages**.

Data messages are generated by the Edge Router and directed to the Broker Application.

Control messages come from the Broker Application and are directed to the Edge Router.

Each message consists of 3 '#' separated fields, the first is the message type, the second the message value and the third an IPv6 network address. Some messages have empty values for the value or address fields.

Edge Data Messages

Data messages contain either Orientation, Magnetometer Calibration or Status data.

Data Messages are constructed and sent autonomously from the Edge Router to the Broker, typically in response to the Edge Router receiving a Node Data Message from a node.

Examples of Edge Data messages.

Data Message	example	description
Orientation	`do#12472:-.9:0.8:1.1:0.2#affe::584b:1`	contains a node's IPv6 address and its timestamp and orientation w,x,y,z quaternion values
Calibration	`dc#-89:-86:-82:88:3:54#affe::584b:1`	contains a node's address and that node's magnetometer calibration values
Status	`ds#111:200:1#affe::584b:1`	contains a node's address and its IMU DOF settings, its current unsolicited data update time interval and its auto calibration mode

Edge Control Messages

Control Messages are used to change the configuration of one or more Biot Nodes belonging to the Edge Router. They are sent from the Biot Broker to the Edge Router.

Examples of Edge Control Messages:

Control Message	example	description
Identify a Node	`cled#2#affe::584b:3763:a0ca:1`	turn the LED belonging to the Biot Node with the address '`affe::584b:3763:a0ca:1`' off(0), on(1), to normal blink(2) or set to rapid blink(3)
Synchronise all Nodes	`sync##`	ask edge router to send a ctim message to all its nodes
Set IMU DOF	`cdof#111#affe::584b:3763:a0ca:1`	set a node's Gyros, Accelerometers, Magnetometers to on(1) or off(0) (the 3 bits are in the order GAM)
Set Magnetometer Calibration Values	`ccav#affe::584b:3763:a0ca:1#-89:-86:-82:88:3:54`	Set a node's calibration values to the ones provided
Set Magnetometer Calibration Mode	`cmcm#1#affe::584b:3763:a0ca:1`	Change a node's auto calibration mode to: 0:off, 1:on 2:reset and on, 3: reset and off
Set unsolicited data update interval	`cdup#200#affe::584b:3763:a0ca:16"`	set how long a node waits between sending unsolicited node updates (0 = do not send any)
Reboot	creb##`affe::584b:3763:a0ca:1`	Tell a node to reboot itself

Biot Broker communications:

The Biot Broker provides a TCP/IP REST API to User Applications.

The Biot Broker talks to the Edge Router using UDP/IP Edge Messages.

The Edge messages are described in the previous section. (Edge Router communications).

Biot Broker API commands:

The REST API commands allow user applications to get data about the system (including node orientations) and to adjust the behaviour of the system.

Biot Broker API commands:

Responses will be of the Mime-Type application/json.

Examples of API commands

Method	Example	Description
Root	`GET '/'`	describes the REST Interface
All Biot Data	`GET '/biotz'`	get a summary of all active Biots the system has data on
Get Count	`GET '/biotz/count'`	return a count of known active Biots
Addresses	`GET '/biotz/addresses'`	return a list of active Biot IPv6 addresses
get all Node data	`GET '/biotz/addresses/IPv6_ADDRESS`	get details about the Biot with the given address
get Node orientation data	`GET '/biotz/addresses/IPv6_ADDRESS/data'`	get the orientation quaternion for a given Biot
get current magnetometer calibration	`GET '/biotz/addresses/IPv6_ADDRESS/calibration'`	get current magnetometer calibration data for a particular Biot
Send a Calibration to a Biot	`PUT '/biotz/addresses/IPv6_ADDRESS/calibration'`	Sets the calibration of a node to the parameters passed in the request.
Set the Biot LED	`PUT '/biotz/addresses/IPv6_ADDRESS/led'`	Sets the Biot's LED display to the mode in the passed parameters

Detailed API and Messaging specifications

Biot Message Protocol

The Biot messaging protocol uses UDP/6LowPAN and consists of small text messages.

Each message consists of 2 fields, a short text identifier and a string of associated data.

The field separator is the '#' symbol.

Biot messages can be divided into 2 types, data or control messages. Data messages are sent from Biots, control messages are sent to Biots.

Data message identifiers are 2 characters starting with 'd'. Control message identifiers are 4 characters starting with 'c'.

Being UDP you cannot rely on Biot messages to always arrive at their destination and you will not receive any confirmation of their reception. You also cannot rely on Biot messages to appear at a destination in the sequence they were sent.

For this reason, a stream of Biot data messages are sent regularly by the Biots without being asked.

To determine if a control message was received by a node or may need to be resent, the unsolicited status data messages that are sent by the Biot at regular intervals can be inspected to see how the Biot is currently configured and if the requested changes have been made.

Biot Messages:

List of all Biot commands

Message	Type	Identifier
Orientation	Data	do
Calibration	Data	dc
Status	Data	ds
LED Status	Control	cled
Time	Control	ctim
DOF	Control	dof
Set Calibration Values	Control	ccav
Set Magnetometer Calibration Mode	Control	cmcm
Set Data Update Interval	Control	cdup
Reboot	Control	creb

Orientation: do

		Notes
Message Type	Data	
Description	an orientation message is used to describe a spatial orientation. It consists of a system time stamp (to allow the orientation to be correlated with a time) and a Quaternion.	
Identifier	`do`	
Format	`do#TS:QW:QX:QY:QZ`	TS = node system time QW = node orientation quaternion w component QX = node orientation quaternion x component QY = node orientation quaternion y component QZ = node orientation quaternion z component
Example	`do#653472:-2.987:0.88:1.1000:0.289`	

Comments

A Biot sends orientation messages to the ER on a regular basis. The frequency of sending is controlled by the Biots current Data Update Interval setting.

Calibration: dc

		Notes
Message Type	Data	
Description	a calibration message is used to describe the 3 magnetometers envelope of readings. These values indicate how the 3 magnetometers readings deviate from an ideal spherical distribution and can be used to reduce hard and soft iron biases that otherwise distort the magnetometer readings.	
Identifier	`dc`	
Format	`dc#X0:Y0:Z0:X1:Y1:Z1`	X0 = minimum x-axis value Y0 = minimum y-axis value Y0 = minimum z-axis value X1 = maximum x-axis value Y1 = maximum x-axis value Z1 = maximum x-axis value
Example	`dc#-89:-86:-82:88:3:54`	

Comments

A Biot should send calibration messages to the ER on a regular basis. The frequency of sending is controlled by the Biots current Data Update Interval setting. This information can be saved elsewhere in the system and later used to allow the quick set up of the magnetometer calibration of a node after it reboots.

Calibration data is typically sent at 1/100 the frequency of data orientation messages.

Status: ds

		Notes
Message Type	Data	
Description	Describes the status of a Biot. It contains the IMU DOF settings, the node's unsolicited data update time interval and the magnetometer auto calibration mode	
Identifier	`ds`	
Format	`ds#GAM:UI:CM`	GAM = the Gyro, Accelerometer, Magnetomer on/off status (0:off, 1:on) UI = the update interval in system time units CM = magnetometer calibration mode (0:off, 1:on)
Example	`ds#111:200:1`	Gyro, Accelerometer all on, wait 200 time units between sending data updates and auto-calibration is on
Example	`ds#010:200:0`	Gyro off, Accelerometer on, Magnetometer off, no data updates to be sent and auto-calibration is off

Comments

A Biot should send status messages to the ER on a regular basis. The frequency of sending is controlled by the Biots current Data Update Interval setting. This information can be used by other parts of the system to verify that the node is in the state requested or expected (particularly as UDP control messages sent by the ER to a node may arrive out of sequence or may be missed altogether by the Biot)

Status data is typically sent at 1/100 the frequency of data orientation messages.

LED Status: cled

		Notes
Message Type	Control	
Description	Request to set the LED status of a Node	
Identifier	`cled`	
Format	`cled#MODE`	MODE = 0:off 1:on, 2: standard blink (the default), 3: rapid blink
Example	`cled#2`	The node should rapidly blink its LED
Example	`cled#0`	The node should turn its LED permanently off

Comment:
If sent a MODE=3 signal (flash rapidly), the node will flash rapidly for several seconds before resuming the default slow flash behaviour (MODE=2). Modes 0 and 1 are intended for development and diagnostic use and should not normally be required or used.

Time: ctim

		Notes
Message Type	Control	
Description	Request to set a Biot's system time to the given time value	
Identifier	`ctim`	
Format	`ctim#VALUE`	VALUE = time
Example	`ctim#12345`	set the Biot's system time to 12345

Comment

Normally this command is sent by the ER to keep the nodes synchronised to a common time. Biot's on receiving this message adjust their system time to match the given value.

DOF: cdof

		Notes
Message Type	Control	
Description	Request to turn the various IMU components on or off.	
Identifier	`cdof`	
Format	`cdof#GAM`	G = gyros off:0, on:1 A = accelerometers off:0, on:1 M = magnetometers off:0, on:1
Example	`cdof#100`	turn the Biot's gyros on and accelerometers and magnetometers off

Comment:
The DOF command is designed for development and diagnostic use, it should not normally be required or used as it stops the orientation data from being properly calculated.

Set Calibration Values: ccav

		Notes
Message Type	Control	
Description	Request to set a node's magnetometer calibration envelope.	
Identifier	`ccav`	
Format	`ccav#X0:Y0:Z0:X1:Y1:Z1`	X0 = minimum x-axis value Y0 = minimum y-axis value Y0 = minimum z-axis value X1 = maximum x-axis value Y1 = maximum x-axis value Z1 = maximum x-axis value
Example	`ccav#-89:-86:-82:88:3:54`	

Comments

If a magnetometer operated ideally the *magnitude* of the North Vector measured by the magnetometers would be constant irrespective of the Biot's orientation. The end of the North Vector should always lie on the surface of a sphere centered on the origin.

In real life, magnetometer readings are distorted by soft and hard iron errors due to local magnetic field variations caused by the device's construction, nearby objects, other electronic/electric/magnetised devices etc. The result is the surface the vector traces out is displaced from the origin and the shape is deformed from a perfect sphere.

The envelope values indicate the actual maximum and minimum values encountered in the x, y and z directions allowing the system to calculate how far from the origin and how far from a sphere the readings appear to trace and so correct for these biases.

Set Magnetometer Calibration Mode: cmcm

		Notes
Message Type	Control	
Description	Request to turn a node's auto-calibration on/off	
Identifier	`cmcm`	
Format	`cmcm#MODE`	MODE: 0 = stop auto calibration 1 = start or restart auto calibration 2 = reset the calibration envelope and start auto calibration 3 = reset the calibration envelope and stop auto calibration
Example	`cmcm#0`	stop auto calibration (eg if you wish the node to stop calculating the calibration envelope).
Example	`cmcm#1`	start (or restart) auto calibration (eg if you wish the node to keep adjusting the calibration values as it takes readings).

Comments

In auto calibration mode, the system will update the magnetometer calibration envelope if the system detects any new extreme x, y and z values. If auto calibration is turned off, the envelope will stop being updated. Modes 2 and 3 will reset the envelope values to their un-calibrated state and turn auto calibration on (mode 2) or off (mode 3).

Set Data Update Interval: cdup

		Notes
Message Type	Control	
Description	set a Biot's time interval between data updates	
Identifier	`cdup`	
Format	`cdup#TIME`	TIME = time interval (in system time units) to wait between sending the unsolicited data messages. 0 = do not send unsolicited data messages
Example	`Cdup#0`	In this example the Biot should stop sending unsolicited data messages
Example	`cdup#200`	In this example the Biot should attempt to send data messages every 200 time units

Comments
Unsolicited orientation messages will be sent every update interval. Calibration and Status data messages will be sent less often.
Currently the calibration and status data time interval is set to be 100 times as long as the Data Update Interval meaning there will be 100 orientation messages sent for every calibration or status message sent.

Reboot: creb

		Notes
Message Type	Control	
Description	The Biot should rebbot itself and resume its normal boot up state.	
Identifier	`creb`	
Format	`creb#`	No data should be sent, just the identifier
Example	`creb#`	

Comments
On receiving this message, the RIOT operating system that is running on the Biot microprocessor should reboot itself. This will set the Biot back to its default configuration, including wiping magnetometer calibrations, resetting the system time and restarting the network system etc

After rebooting, it may take several seconds before the Biot becomes appropriately registered on the 6LowPAN network again and thus before it is available to participate in the system again.

Edge Messaging Protocol

The Edge messaging Protocol uses UDP/IP.

In a similar way to Biot messages, Edge messages can be divided into 2 types, data or control messages. Data messages are sent from the ER, control messages are sent to the ER.

Most Edge messages have a matching message in the Biot Message API.

The main difference between Biot and Edge messages is that Edge messages usually need to include a network address to indicate either: the Biot node the message comes from or the Biot the data should be sent to.

Edge messages are therefore of a similar format to Biot messages but with an address field added to them (they use the same '#" symbol as the field separator).
eg

```
ds#111:200:1#affe::584b:3763:a0ca:19b6
```

which is a message that has an identifier 'ds', a value of 111:200:1 and a node address of affe::584b:3763:a0ca:19b6

List of all Edge commands

Message	Type	Identifier
Orientation	Data	do
Calibration	Data	dc
Status	Data	ds
Identify	Control	cled
Synchronise	Control	csym
DOF	Control	dof
Set Calibration Values	Control	ccav
Set Magnetometer Calibration Mode	Control	cmcm
Set Data Update Interval	Control	cdup
Reboot	Control	creb

Edge Messages:

Orientation: do

		Notes
Message Type	Data	
Description	Contains orientation data from a specified Biot	
Identifier	`do`	
Format	`do#ORIENTATION#IPv6-ADDRESS`	The ORIENTATION value will be in the same form as in the corresponding Biot API Orientation message.
Example	`do#6532:-.987:0.88:1.1000:0.289#affe::584b:3763:a0ca:1`	
Behaviour	This message is sent from the ER to the Broker whenever the ER receives an orientation Biot message from a Biot.	

Comments

The network address is the address of the Biot that the orientation data came from.

Calibration: dc

		Notes
Message Type	Data	
Description	Contains calibration data from a specified Biot	
Identifier	`dc`	
Format	`dc#CALIBRATION#IPv6-ADDRESS`	The CALIBRATION value will be in the same form as the corresponding Biot API Calibration message.
Example	`dc#6532:-.987:0.88:1.1000:0.289#affe::584b:3763:a0ca:1`	
Behaviour	This message is sent from the ER to the Broker whenever the ER receives a calibration Biot message from a Biot.	

Comments
The address is the network address of the Biot Node that the calibration data came from.

Status: ds

		Notes
Message Type	Data	
Description	Contains status data from a specified Biot	
Identifier	`dc`	
Other properties	`dc#STATUS#IPv6-ADDRESS`	The STATUS value will be in the same form as the corresponding Biot API Status message.
Example	`ds:111:200:1#affe::584b:3763:a0ca:1`	
Behaviour	This message is sent from the ER to the Broker whenever the ER receives a status Biot message from a Biot.	

Comments
The network address is that of the Biot Node that the status data came from.

Identify: cled

		Notes
Message Type	Control	
Description	Set the behaviour of a Biot Status LED	
Identifier	`cled`	
Format	`cledc#MODE#IPv6-ADDRESS`	MODE 0 = off 1= on 2 = normal blink (default) 3 = rapid blink
Example	`cled:2#affe::584b:3763:a0ca:1`	
Behaviour	When this message is received by the ER, it should send a “cled” Biot message to the specified Biot.	

Comments

If sent a MODE=3 signal (flash rapidly), the node will flash rapidly for several seconds before resuming the default slow flash behaviour (MODE=2).

Modes 0 and 1 are intended for development and diagnostic use and should not normally be required or used.

A control message to set a Biots LED status to MODE=3 can be used to assist in visually identifying that Biot because of its rapid flash rate.

Synchronise: csyn

		Notes
Message Type	Control	
Description	Ask the ER to sent a time control messsage to all its Biot nodes. This should cause them to adopt that system time as their own.	
Identifier	`csyn`	
Format	`csyn##`	
Example	`csyn##`	
Behaviour	When this message is received by the ER, it will send a “csyn” Biot message to all the Biots the ER knows.	

Comments

Once all nodes are synchronised with the ER's system time, the LED status “heartbeat” blink should occur at the same time for all nodes, indicating their active participation in the ER's DODAG network.

An out of sync “heartbeat” from a Biot's LED indicates the Biot is not properly participating in the system.

Set DOF: cdof

		Notes
Message Type	Control	
Description	Tell a Biot to adjust the status of its Gyroscopes, Accelerometers and or Magnetometers. Similar to the Biot Command defined above but requires a Node address so the ER knows where to send the command	
Identifier	`cdof`	
Format	`cdof#GAM#IPv6_ADDRESS`	G = gyros off:0, on:1 A = accelerometers off:0, on:1 M = magnetometers off:0, on:1
Example	`cdof#111#affe::584b:3763:a0ca:1`	
Behaviour	When the ER receives this message it should send a “cdof” Biot message to the appropriate Biot.	

Comments
The DOF command is designed for development and diagnostic use, it should not normally be required or used as disabling any DOF will stop the orientation data from being properly calculated.

Normally all DOFs are active (ie mode 111).

Set Magnetomer Calibration Values: ccav

		Notes
Message Type	Control	
Description	Tell a Biot to use the given magnetometer calibration envelope. Similar to the Biot “ccav” command defined above but it requires a Node address so the ER knows where to send the command	
Identifier	`ccav`	
Format	`ccav#X0:Y0:Z0:X1:Y1:Z1#IPv6_ADDRESS`	X0 = minimum x-axis value Y0 = minimum y-axis value Y0 = minimum z-axis value X1 = maximum x-axis value Y1 = maximum x-axis value Z1 = maximum x-axis value
Example	`ccav#-89:-86:-82:88:3:54#affe::584b:3763:a0ca:1`	
Behaviour	When the ER receives this message it should send a “ccav” Biot message to the specified Biot.	

Comments

If a magnetometer operated ideally the *magnitude* of the North Vector measured by the magnetometers would be constant irrespective of the Biot's orientation. The end of the North Vector should always lie on the surface of a sphere centered on the origin.

In real life, magnetometer readings are distorted by soft and hard iron errors due to local magnetic field variations caused by the device's construction, nearby objects, other electronic/electric/magnetised devices etc. The result is the surface the vector traces out is displaced from the origin and the shape is deformed from a perfect sphere.

The envelope values indicate the actual maximum and minimum values encountered in the x, y and z directions allowing the system to calculate how far from the origin and how far from a sphere the readings appear to trace and so correct for these biases.

Set Calibration Mode:cmcm

		Notes
Message Type	Control	
Description	Request to turn a node's auto-calibration on/off	
Identifier	`cmcm`	
Format	`cmcm#MODE#IPv6_ADDRESS`	MODE: 0 = stop auto calibration 1 = start or restart auto calibration 2 = reset the calibration envelope and start auto calibration 3 = reset the calibration envelope and stop auto calibration
Example	`cmcm#0#affe::584b:3763:a0ca:1`	stop auto calibration (eg if you wish the node to stop calculating the calibration envelope).
Example	`cmcm#1#affe::584b:3763:a0ca:1`	start (or restart) auto calibration (eg if you wish the node to keep adjusting the calibration values as it takes readings).
Behaviour	When the ER receives this message it should send a “cmcm” Biot message to the specified Biot.	

Comments

In auto calibration mode, a Biot will update the magnetometer calibration envelope with any new extreme x, y and z values. If auto calibration is turned off, the envelope will stop being updated. Modes 2 and 3 will actually reset the envelope values to their un-calibrated state and then turn auto calibration on or off as indicated in the table.

Set Data Update Interval: cdup

		Notes
Message Type	Control	
Description	Request to set the time interval between data updates	
Identifier	`cdup`	
Format	`cdup#TIME#IPv6_ADDRESS`	TIME = time interval (in system time units) to wait between sending the unsolicited data messages. 0 = do not send unsolicited data messages
Example	`cdup#0#affe::584b:3763:a0ca:1`	tell node it node should stop sending unsolicited data messages
Example	`cdup#200#affe::584b:3763:a0ca:1`	tell node it should attempt to send data messages every 200 time units
Behaviour	When the ER receives this message it should send a “cdup” Biot message to the specified Biot.	

Comments
A Biot's unsolicited orientation messages will be sent every update interval. Calibration and Status data messages will be sent less often.
Currently the calibration and status data time interval is set to be 100 times as long as the Data Update Interval meaning there will be 100 orientation messages sent for every calibration or status message sent.

Reboot: creb

		Notes
Message Type	Control	
Description	The Biot should rebbot itself and resume its normal boot up state.	
Identifier	`creb`	
Format	`creb##IPv6_IDENTIFIER`	No data should be sent, just the identifier and node address
Example	`creb##affe::584b:3763:a0ca:1`	
Behaviour	When the ER receives this message it should send a “creb” Biot message to the specified Biot.	

Comments
After a Biot node reboots, it may take several seconds before a Biot becomes appropriately registered on the 6LowPAN network again thus before it is available to participate in the system again.

Broker REST API

The Biot Broker allows Applications to interact with the Biot System via a TCP/IP HTTP REST web service.

The Broker listens and responds to REST requests from User Applications and communicates with the Edge Router by sneding and receiving UDP/IP Edge messages.

The REST URL interface is divided into several sets of resources, the major ones being:

1. Biot resources

2. Storage resources

Biot Resources are concerned with the Biots themselves, their configuration and orientation and how they are connected to limbs (or other objects).

Storage resources are concerned with giving applications the ability to store and retrieve data (such as Biot calibrations, details on assemblies of limbs, application settings etc) between sessions.

Broker API methods

Summary of all API Resources

Resource	Description
General Utility resources	
`GET '/'`	Return an overview of Resources available in the API
`GET '/biotz'`	Return a list of active Biots in the system
`GET '/biotz/count'`	Return a count of active Biots
`PUT '/biotz/synchronise'`	Tell the ER to synchronise all Biots
`GET '/biotz/addresses'`	Return a list of active Biot addresses
Specific Biot resources	
`GET '/biotz/addresses/IPv6_ADDRESS`	Return a summary of a particular Biot
`GET '/biotz/addresses/IPv6_ADDRESS/data'`	Return a particular Biot's orientation data
`GET '/biotz/addresses/IPv6_ADDRESS/calibration'`	Return a particular Biot's Calibration envelope
`GET`	Return a particular Biot's

`'/biotz/addresses/IPv6_ADDRESS/status'`	configuration status
`GET '/biotz/addresses/IPv6_ADDRESS/led`	Get a Biot's LED mode
`PUT '/biotz/addresses/IPv6_ADDRESS/led`	Set a particular Biot's LED mode
`GET '/biotz/addresses/IPv6_ADDRESS/dof`	Get a Biot's current DOF settings
`PUT '/biotz/addresses/IPv6_ADDRESS/dof`	Change a Biot's DOF settings
`GET '/biotz/addresses/IPv6_ADDRESS/interval`	Get a Biot's Update Interval
`PUT '/biotz/addresses/IPv6_ADDRESS/interval`	Change a Biot's update interval
`GET '/biotz/addresses/IPv6_ADDRESS/auto`	Get a Biot's calibration mode
`PUT '/biotz/addresses/IPv6_ADDRESS/auto`	Change a Biot's calibration mode
Data Storage	
`GET '/data'`	Return a list of known categories for which data is held
`GET '/data/:category'`	Return a list of known resources that exist in a given category
`GET '/data/:category/:name'`	Return a specific resource
`PUT '/data/:category/:name'`	Save a resource
`DELETE '/data/:category'`	Remove a category
`DELETE '/data/:category/:name'`	Remove an item
Dummy Biot Resources	**(intended for system testing, development and diagnostic use only)**
`GET '/devel/dummybiots`	Get a list of all dummy nodes
`PUT '/devel/dummybiots/IPv6_ADDRESS`	Create a new Dummy Biot node with the given address
`DELETE '/devel/dummybiots`	Delete all dummy Biots
`DELETE '/devel/dummybiots/IPv6_ADDRESS`	Delete a specific dummy Biot

As a general rule, any successful call will return a 200 status code and a response body with some text.

Response bodies will typically be JSON

Root Resource

Root Resource	
Description	Returns an overview of Resources available in the API
Method	GET
URL	/
Example	
Request	`GET / HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`{`
	`  "title": "Biotz Broker REST API",`
	`  "description": "Interface to a network of Biot Orientation Sensors",`
	`  "version": "0.1",`
	`  "links": [`
	`  "http://localhost:8889/",`
	`  "http://localhost:8889/biotz",`
	`  "http://localhost:8889/data"`
	`  ]`
	`}`

Biotz Resource

Biotz Resource	
Description	Returns an summary of Active Biots
Method	GET
URL	/biotz
Example	
Request	`GET /biotz HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`{`
	`  "count":3,`
	`  "addresses":[`
	`  "affe::594a:1455:ff12:f9f2",`
	`  "affe::594c:1c57:5786:21b2",`
	`  "affe::5942:376a:83b:b8d6"`
	`  ]`
	`}`

Biotz Count

Count Biots	
Description	Return a count of Active Biots
Method	GET
URL	/biotz/count
Example	
Request	`GET /biotz/count HTTP/1.1`
Response	`HTTP/1.1 200 OK` `"3"`

Synchronise Network

Synchronise Network	
Description	Attempt to force all Biots to synchronise with the ER
Method	PUT
URL	/biotz/synchronise
Example	
Request	`PUT /biotz/synchronise HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`"OK"`

Biotz Addresses

Biotz Addresses	
Description	Return the IPv6 addresses of known Biots
Method	GET
URL	/biotz/addresses
Example	
Request	`GET /biotz/addresses HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`[`
	`  "affe::594a:1455:ff12:f9f2",`
	`  "affe::594c:1c57:5786:21b2",`
	`  "affe::5942:376a:83b:b8d6"`
	`]`

Biotz Node

Specific Biot Resource	
Description	Return details of a Biot Node
Method	GET
URL	/biotz/addresses/:address
Example	
Request	`GET /biotz/addresses/affe::594a:1455:ff12:f9f2 HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`{`
	`"data":"48400:0.929597:0.306609:-0.019845:-0.203584",`
	`"calibration":"-119:-45:-421:313:275:0"},`
	`"status":"111:200:1",`
	`"interval":"200",`
	`"auto":"1",`
	`"dof":"111",`
	`"led":"2"`
	`}`

Biotz Orientation Data

Biot Orientation	
Description	Return details of the orientation of a Biot Node
Method	GET
URL	/biotz/addresses/:address/data
Example	
Request	`GET /biotz/addresses/affe::594a:1455:ff12:f9f2/data HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`"48400:0.929597:0.306609:-0.019845:-0.203584"`

Biotz Status Data

Biot Status	
Description	Return details of the configuration status of a Biot Node
Method	GET
URL	/biotz/addresses/:address/status
Example	
Request	`GET /biotz/addresses/affe::594a:1455:ff12:f9f2/status HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`"111:200:1"`

Biotz Calibration Data

Biot Calibration	
Description	Get/Set the magnetometer calibration envelope of a Biot Node
Method	GET/PUT
URL	/biotz/addresses/:address/calibration
Example	**Get current value**
Request	`GET /biotz/addresses/affe::594a:1455:ff12:f9f2/calibration HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`"-267:0:-428:72:123:0"`

Example	**Set new value**

Request	`PUT /biotz/addresses/affe::594a:1455:ff12:f9f2/calibration HTTP/1.1`
	
	`-262:10:-401:66:120:24`

Response	`HTTP/1.1 200 OK`
	
	`"OK"`

Biotz Interval Data

Biot Status	
Description	Get/Set the current update interval value
Method	GET/PUT
URL	/biotz/addresses/:address/interval
Example	**Get current value**
Request	`GET /biotz/addresses/affe::594a:1455:ff12:f9f2/interval HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`"321"`

Example	**Set new value**

Request	`PUT /biotz/addresses/affe::594a:1455:ff12:f9f2/interval HTTP/1.1`
	
	`123`

Response	`HTTP/1.1 200 OK`
	
	`"OK"`

Biotz LED Data

Biot Status	
Description	Get/Set the current LED mode
Method	GET/PUT
URL	/biotz/addresses/:address/led
Example	**Change DOF setting**

Request	`PUT /biotz/addresses/affe::594a:1455:ff12:f9f2/led HTTP/1.1`
	
	`3`

Response	`HTTP/1.1 200 OK`
	
	`"OK"`

Example	**Get current DOF setting**
Request	`GET /biotz/addresses/affe::594a:1455:ff12:f9f2/led HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`"2"`

Biotz DOF Data

Biot Status	
Description	Get/Set the current update interval value
Method	GET/PUT
URL	/biotz/addresses/:address/dof
Example	**Change DOF setting**

Request	`PUT /biotz/addresses/affe::594a:1455:ff12:f9f2/dof HTTP/1.1`
	
	`101`

Response	`HTTP/1.1 200 OK`
	
	`"OK"`

Example	**Get current DOF setting**
Request	`GET /biotz/addresses/affe::594a:1455:ff12:f9f2/dof HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`"101"`

Biotz Magnetometer Calibration Mode

Biot Status	
Description	Get/Set the current magnetometer calibration mode
Method	GET/PUT
URL	/biotz/addresses/:address/auto
Example	**Set mode**

Request	`PUT /biotz/addresses/affe::594a:1455:ff12:f9f2/auto HTTP/1.1`
	
	`1`

Response	`HTTP/1.1 200 OK`
	
	`"OK"`

Example	**Get current mode**
Request	`GET /biotz/addresses/affe::594a:1455:ff12:f9f2/auto HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`"1"`

Data Storage Categories

Data Storage	
Description	Get a list of categories that have been used
Method	GET
URL	/data
Example	
Request	`GET /data HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`[`
	`  "calibrations",`
	`  "assemblies",`
	`  "limbs",`
	`  "configurations"`
	`]`

Comments:

The Data Storage Methods allow applications to store and retrieve data between sessions. They act like a simple file storage system. The "category" property is similar to a directory or folder and the "name" property is equivalent to the name of the file. Data to be stored is sent in the body of the PUT request.

Data Storage Items

Data Storage	
Description	Get a list of items that exist in a category
Method	GET/DELETE
URL	/data/:category
Example	
Request	`GET /data/calibrations HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`[`
	`  "affe::594a:1455:ff12:f9f2",`
	`  "affe::594c:1c57:5786:21b2",`
	`  "affe::5942:376a:83b:b8d6"`
	`]`

Request	`GET /data/limbs HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`[`
	`  "femur",`
	`  "tibia",`
	`  "pelvis"`
	`]`

Example	**Delete a Category**
Request	`DELETE /data/calibrations HTTP/1.1`

Response	`HTTP/1.1 409 Conflict`
	
	`"category contains items and cannot be deleted, delete items first"`

Example	**Delete a Category**
Request	`DELETE /data/emptyAndUnwantedCategory HTTP/1.1`

Response	`HTTP/1.1 200`
	
	`"OK"`

Comments:

You cannot DELETE a category if it contains items.

Particular Data Storage Item

Stored Item	
Description	Get/Put a stored item
Method	GET/PUT
URL	/data/:category/:name
Example	**Get saved item**
Request	`GET /data/calibrations/affe::594a:1455:ff12:f9f2 HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`"-262:10:-401:66:120:24"`

Example	**Save new item**

Request	`PUT /data/limbs/femur HTTP/1.1`
	
	`{"length":"24","vertices":"/data/models/femur.json"}`

Response	`HTTP/1.1 200 OK`
	
	`"OK"`

Example	**Get saved item**
Request	`GET /data/limbs/femur HTTP/1.1`

Response	`HTTP/1.1 200 OK`
	
	`{`
	`  "length":"24",`
	`  "vertices":"/data/models/femur.json"`
	`}`

Comments:

if you PUT an item in a category that doesn't exist, the category it will be created.

Dummy Biots

Dummy Nodes	
Description	Create Dummy Biot Nodes for development/testing/diagnostics
Method	PUT/DELETE
URL	/devel/dummybiots/:address
Example	**Create Dummy Biot**
Request	`PUT /devel/dummybiots/affe::002 HTTP/1.1`
Response	`HTTP/1.1 200 OK` `"OK"`
Example	**Delete Dummy Biot**
Request	`DELETE /devel/dummybiots/affe::002 HTTP/1.1`
Response	`HTTP/1.1 200 OK` `"OK"`

Comments:

Dummy Biot Nodes are used for development and testing, they would not normally be used otherwise.

Dummy Biots

Dummy Nodes	
Description	all Dummy Biot Nodes
Method	GET/DELETE
URL	/devel/dummybiots
Example	**List all Dummy Biots**
Request	`GET /devel/dummybiots`

Response	`HTTP/1.1 200 OK`
	
	`[`
	`  "affe::001",`
	`  "affe::002",`
	`  "affe::003",`
	`]`

Example	**Delete all Dummy Biots**
Request	`DELETE /devel/dummybiots`

Response	`HTTP/1.1 200 OK`
	
	`"OK"`

Comments:

Dummy Biot Nodes are used for development and testing, they would not normally be used otherwise.