re.flex Bluetooth protocol

Author:	**Andrei Kluger**	**2021-02-09**
Approved by:	**Camil Moldoveanu**	**2021-02-09**
Revision:	**1**	
Category:	**Public**	

1. Goal

This document describes the communication protocol between the re.flex Bluetooth sensors and the DiGA. It also describes the data collected from the sensors and how it's used.

2. Data

The re.flex application connects via Bluetooth 4.2 to two enabled motion sensors. The sensors do not communicate one with the other, only with the application.

No information storage

The individual sensors do not store any information, everything is sent directly via Bluetooth. When the sensors are not connected to the application via Bluetooth, no data is preserved on the sensors. There is no configuration saved on the sensors at any point in time, and no personal data related in any way to the patient. Sensor data has health or personal significance only once it's combined and processed on the application.

Non reconstructible health data

re.flex sensors use a three axis accelerometer and a three axis gyroscope but do not use a compass as it can have lots of magnetic field interference from equipment or metal parts. This means the positions reported by the sensors are not arranged by the true north and cannot be recomposed by themselves or recombined to create a physical joint angle.

Each sensor sends the app a position in a space which is defined for each sensor. Those two positions, being in two different spaces which have no connection between them, cannot be correlated or interpreted by themselves to generate a world space position. To generate this world space position, the app needs to take the positions from each space and then use the calibration data determined at runtime in the mobile application.

The calibration data contains the relation between the two sensor spaces and is dependent on the placing of the sensors on the patient's leg, the muscle configuration of the patient and other external factors. This changes whenever the patient puts the sensors on the leg as no two placements will be the same.

Using the calibration data, the two positions sent by the sensors are combined and transformed into a common space, linked to the leg, where angles between joints can be determined. Without the calibration data, the angles between the sensors cannot be reconstructed and the position data from the sensors is not usable.

This further strengthens that there's no health data sent via Bluetooth from the sensors to the app. The health data is generated only when the app applies the previously determined calibration.

Binary format

The app shows a 3D live representation of the patient's leg as it moves and, in order to achieve that, the sensors need to send data with a certain speed to the mobile device which in turn needs to receive, process the data and generate the representation. Taking into account the speed limitations of the Bluetooth protocol and the battery consumption when sending packages over Bluetooth, the format of the data was kept to a minimum.

The minimum frequency with which the sensors need to be able to transmit data is 25hz.

Because the app relies on instantaneous feedback to the user, no data is stored on the sensors and the sensors do not transmit if the app is not connected. More, when the app is not connected to the sensors, they enter a sleep phase where motion capture and non-essential features are stopped completely to save power.

3. Interface

The mobile device on which the DiGA is installed scans for compatible Bluetooth 4.2 devices and, if such a device is found, the user is prompted to confirm the connection by pressing a button in the application. The Bluetooth device must be in advertising mode and, during a service scan, it must show the services from the Protocol section below, to be used by the DiGA.

The Bluetooth sensors also use the standard GATT 16 bit UUID Battery service which is implemented according to the Bluetooth SIG.

The sensor communication is done using a 128 bit UUID custom service as described below in the Protocol section.

To achieve a low power consumption, the Bluetooth protocol has a low data transfer rate which means the communication protocol must be as optimized as it can, to be able to conserve power and offer a longer battery life. Also the protocol eliminates almost all overhead to ensure the data transfer can be made even when the signal is not very strong.

4. Protocol

4.1. re.flex Communication service

This service which allows data transfer and sending commands to the re.flex device is defined by the following UDID: **94901830-4562-46D2-911E-998DD6E99F2E**

The re.flex communication service has the following characteristics

- Command characteristic - **94901831-4562-46D2-911E-998DD6E99F2E**
- Packet sending characteristic - **94901832-4562-46D2-911E-998DD6E99F2E**
- Command response characteristic - **94901834-4562-46D2-911E-998DD6E99F2E**

4.2. Command characteristic (write permission with indication)

This characteristic allows the mobile app to send commands to the sensor. The following commands may be sent:

- **Get status** - command id 0x0A. Used to return the status of the sensor

  Command format: 0x0A

  Response format (sent on the response characteristic):

  - byte - the command id - 0x0A
  - int16 - error type - if there is an error detected in the sensor this value is not zero
  - int16 - motion sensor initialization error - if there's an error with the motion sensor this value is not zero
  - int32 - motion sensor reserved bytes
  - int16 - software version - two bytes software version
  - int16 - warming up indicator - if this value is 1 then the sensor is still warming up after having entered a sleep cycle and the app needs to wait for 60 seconds or until the warming up indicator is zero

- **Initialize calibration** - command id 15. This command signals the sensor that a calibration process has started on the mobile app.

  Command format: 0x0F

  Response format (sent on the response characteristic):

  - byte - the command id 0x0F
  - int32 - reserved bytes

- **Sensor notification** - command id 13. This command starts the vibration or lights the leds for a configurable amount of time.

  Command format:

  - byte - the command id - 0x0D
  - byte - reserved byte
  - byte - enable vibration - if this value is 0x01 the vibration is enabled
  - int16 - vibration time - this value sets the number of milliseconds that the vibrator should be activated. If this number is higher than 600 it is capped at this value.
  - byte - LEDs enabled - this value sets which LEDs are lit.

    The bits look like this: 0 0 0 0 0 L3 L2 L1, if L1 is 1 LED1 is enabled, if L2 is 1, LED2 is enabled and so on. More than one led can be enabled at once

  - byte - LED colors - Each LED has two color and this value sets which color is the LED lit with.

The bits look like this: 0 0 0 0 0 L3 L2 L1, if L1 is 1 LED1 is blue, otherwise is red, and so on. In the current app only Blue LEDs are used, the red ones are used for factory testing the features

- int16 - LED1 vibration time - this value sets the number of milliseconds that the first LED should be lit. If this number is higher than 1000, it is capped at this value.
- int16 - LED2 vibration time - this value sets the number of milliseconds that the second LED should be lit. If this number is higher than 1000, it is capped at this value.
- int16 - LED3 vibration time - this value sets the number of milliseconds that the third LED should be lit. If this number is higher than 1000, it is capped at this value.
- byte - reserved byte

Only acknowledgement is sent back to Bluetooth Central as response to this command.

4.3. Command response characteristic (read permission with indication)

This characteristic is used for responses to the queries above. The queries are sent by the mobile device via the command characteristic and the responses via this characteristic.

4.4. Packet sending characteristic (read permission with notification)

This characteristic is used for sending motion data from the sensor to the mobile application. The data is always sent as soon as it's available from the motion sensor, without confirmation. The sensor sends 35 bytes of data at 25Hz rate. Due to backward Bluetooth compatibility, the maximum data package is 20 bytes so the data package is split in two, one package of 17 bytes and one of 18 bytes. They should be reassembled at destination to make the full 35 bytes data package.

Package format:

- byte - command type 0x01
- byte - vector size 0x23
- float - quaternion W - 4 byte floating point value
- float - quaternion X - 4 byte floating point value
- float - quaternion Y - 4 byte floating point value
- float - quaternion Z - 4 byte floating point value
- int16 - raw accelerometer X - 2 byte signed value
- int16 - raw accelerometer Y - 2 byte signed value
- int16 - raw accelerometer Z - 2 byte signed value
- int16 - raw gyroscope X - 2 byte signed value
- int16 - raw gyroscope Y - 2 byte signed value
- int16 - raw gyroscope Z - 2 byte signed value
- int32 - internal sensor timestamp - 4 byte unsigned value
- byte - CRC calculated on one byte as the sum of all the characters in the message