# Progress Report: Wearable Health

Tomas Opazo
Sen Lu
Jan 10, 2017

## Summary of last meeting

Last meeting, we were asked to work on the following tasks

1. Reading from multiple nodes (boards)
2. Data collection trough API
3. Send data to cloud (most probably AWS -Amazon Web Services-)
4. Posture and other kinetic recognition (free fall detection, stress, and so on)

## Work done on tasks 1 and 2

MbientLab has three API to work with: Android, iOS and C++. In our case Android was chosen, therefore an Android phone plays the role of data collector device.
We set up a GitHub repository for software management [2] using an MbientLab sample-project [1] as a starting point to develop the Android application.
The application was gradually modified to collect the following data (sensors) from every connected node (board).
Data (sensors) that we have successfully retrieved so far

- Acceleration
- Angular velocity (gyroscope),
- Pressure
- Temperature

Data (sensors) that we have NOT successfully retrieved so far

- Humidity => API is available but the board is not currently sending data, it can be a configuration problem

Not attempted yet

- Optical Heart rate (PPG)
- Galvanic Skin Response (GSR)

## Work done on tasks 3 and 4

We have not attempted to send data to the cloud (AWS -Amazon Web Services-). This feature shouldn't be too complicated to implement in the future. A producer-consumer design pattern is the most probable candidate to be implemented.

We are currently saving data to a local file in the phone. Every node (board) writes data to two files, the first such file is intended for acceleration and gyroscopic data only, and the second file is for all the other data (temperature, pressure, humidity, etc).

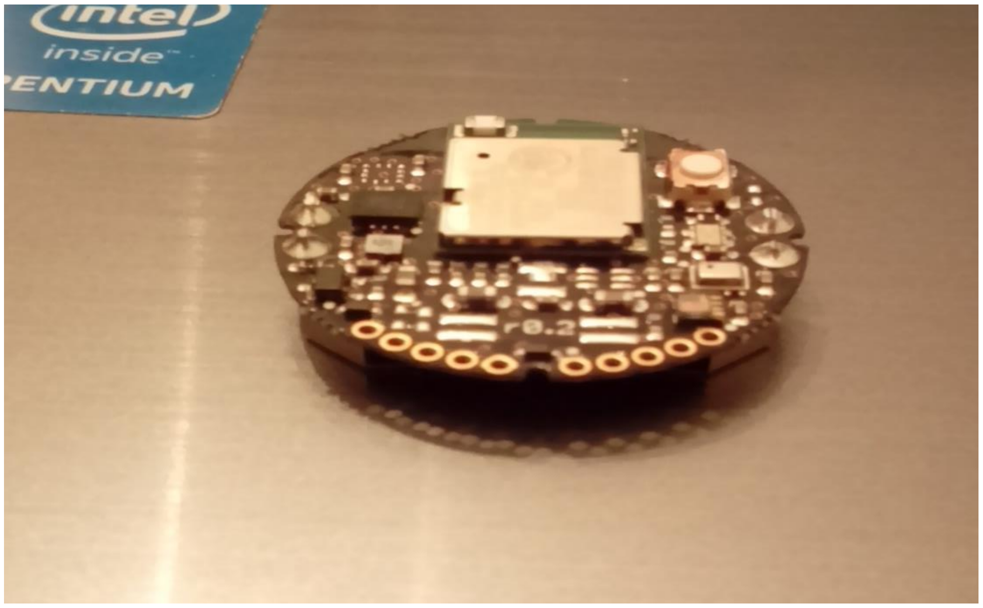
Because we currently don't have a Server to process and graph the collected data, we first attempted to plot acceleration and gyro data in real-time directly on the phone using a graphing library called "opengl", which is the recommended library for Android. Although some progress was made the library was difficult to setup and the idea was dropped. The alternative strategy was to write a “ServeSide” code in python language, which in fact can be later reused when the real Server is available. This code is now part of the repository [2]

## Initial results on tasks 3 and 4

Using the developed Android application [2] we are now capable of:

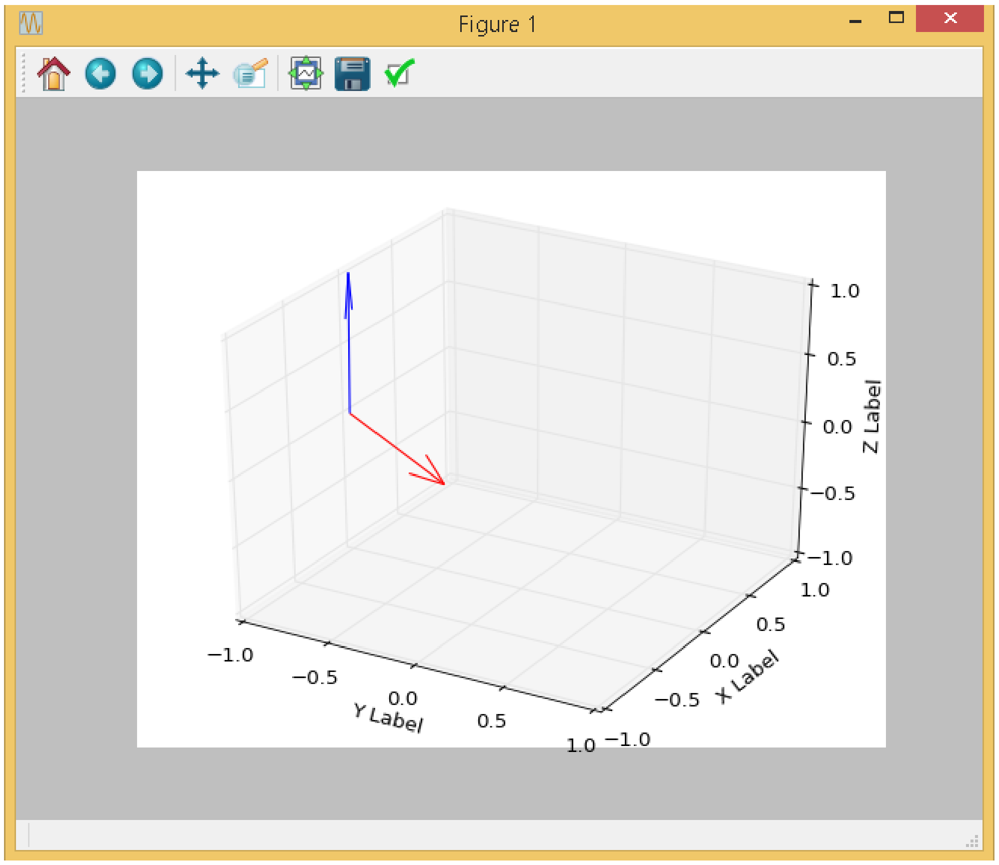
1. Turn on and connect multiple nodes (boards) to an Android phone (currently up to 3 simultaneous nodes have been connected, but there is no software limitation to this number)
2. Start collecting data (the procedure is effective but still rudimentary as the application is not yet user-friendly)
3. Stop collecting data
4. Send data to an email or transfer files over USB (this is the yet to be implemented task 3, which is currently a manual process instead of automatic)
5. Read data and in the case of acceleration and gyroscopic data run a Server Side python script that graph them in 3D
6. All the posture and other kinetic recognition (free fall detection, stress, etc) is yet to be implemented

Two worth-mentioning tests have been made thus far. The idea of the first one was to rotate the board and check if the output data (result of the physical rotation of the board) were aligned with theoretical axes of the datasheets.



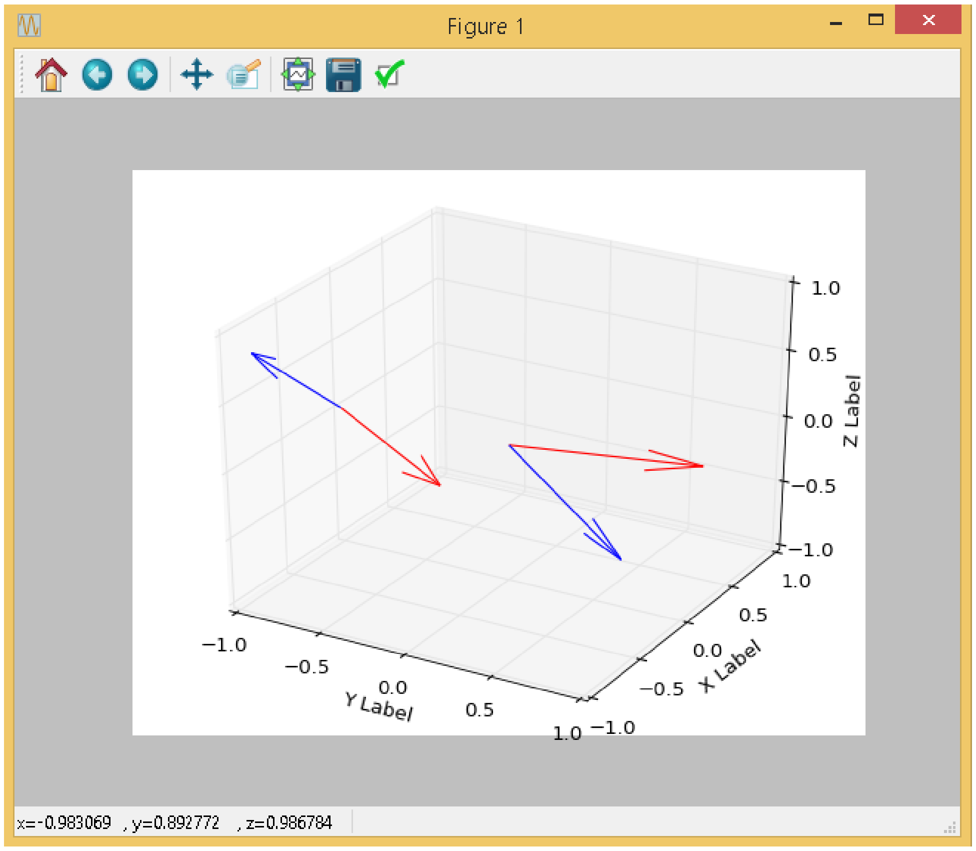
The test started with the sensor as in the picture above (horizontal), and then it was moved according to:

1. horizontally counterclockwise (positive rotation along yaw axis) until it reached +90°
2. horizontally clockwise (negative rotation along yaw axis) until it reached 0° again
3. laterally to the left (negative rotation along roll axis) until it reached 90°
4. laterally to the right (positive rotation along roll axis) until it reached 0° again
5. laterally to the back/observer (negative rotation along pitch axis) until it reached 180° (turned upside down)



The purpose of the second test was to have an initial idea of how the data of person walking would look like. Two boards were attached to the knees of a volunteer (my wife), she was asked to

1. walk in straight line for 5 seconds approx.
2. wait for 5 seconds, turn around and wait 5 seconds more
3. walk in straight line for 5 seconds approx.



## Comments

We think is import to define a certain range of the data-rate necessary for each sensor, as this as effects on the type of programming strategy that the Android application uses. Choosing the lowest rate necessary frees time for more nodes to be connected at the same time, this could be important in order not to overload the phone processor (Bluetooth transmission -> write data to file -> send data over internet)

## References

[1] Mbientalb tutorial https://mbientlab.com/tutorial/android/java/
[2] Github repository for Sw storage and management https://github.com/PsuMobileHealth/DataCollector
[3] Mbientalb Github repository https://github.com/mbientlab-projects
[4] Producer - consumer design pattern http://www.ni.com/white-paper/3023/en/ and https://dzone.com/articles/producer-consumer-pattern