Python programming and data visualisation

Walking Habits

Design decisions about the visual presentation of the data and the implementation architecture

1. Design decisions about the visual presentation of the data

To visualize real time data we decided to used feet animation (as it was suggested) and plot being a combination of 2 plot types: line plot and scatter plot.

Feet animation is combination of feet image and sensors location with their name and value. We decided to use here both value and sensor value to make sure that even new users would know the names of the sensors. Color of each sensor is changing in time and depends on sensor value - it can change from nearly white color to completely red. Color change is very useful for user, because at first glance he can verify how high is the pressure of parts of feet on the ground.

To visualize the changing of values received from sensors we decided to use the line plot because it perfectly shows how the sensors values are changing in time. To show anomalies we decided to use a scatter plot displayed above lines corresponding to sensors values. Each anomaly point is displayed above the trace line in point where anomaly occurred and is much thicker than a line. Scatter points and lines have the same colors to simplify the observation of anomalies for specific sensors by the users. Users can disable each of all plot parts - it improves analyses possibilities by rejecting unnecessary data and focusing on the data being analyzed.

In the history section, almost the same plot shows walking history as a trace of sensors values and anomalies for the period of the last 10 minutes. Additionally, we implemented a simple graphic illustrating sensors location on feet to help users to locate them. We added here a histogram to allow users to see data from another perspective. It allows users to see how often device send values belonging to specified ranges (each range covers 64 units, e.g. first one range from 0 to 63 units). It can help users to find some specific walking patterns or habits.

2. The implementation architecture

To implement our application we used Dash, Plot, Cloudant and Pusher.

We based mainly on Dash because it allows creating reactive web apps in pure Python and is easy to connect with Plotly. We used Plotly because this library provides all the necessary plots types for our application and is very easy in use.

As a database, we used Cloudant - distributed database offered by IBM. Two key features that motivated us to use this database were price (it is free with some limitations, but for our usage free plan was enough to perform all actions) and existence of Python library for easy writing and reading data.

To create real-time traces and animation we used Pusher. It provided us the possibility to refresh plot and animation just after receiving value from the sensor by the application. We tried to use timer to provide this feature, but we decided that real “live” refreshing we will create with Pusher. Another advantage of Pusher that made us decide to use it was that it decreases server load because part responsible for sending refresh data to the frontend part of the application was moved to the Pusher server.