Python programming and data visualization
Warsaw University of Technology
Faculty of Electrical Engineering

PyFeet

Bazyli Reps
Aleksei Haidukevich

Task Description

PyFeet serves the goal of visualization the data, gathered from six pressure sensors, placed on a special shoes. The experiment is meant to reveal anomalies when compared to a healthy foot.

Technology

PyFeet was created using **Plotly Dash** library, with **Flask** as a framework. The database used is **SQLite3**.

Data

The main source of the data is the server of the Elektryczny Faculty on **WUT**. In order to access it, one needs to establish a **VPN** connection. See **README** for more details.

The **API** of the **Tesla** server returns the data in a form that can be treated as **JSON**. The workflow of the application consists of visualization the data and saving it in the database for the further fetching.

Look and Feel

Since **Dash** library is used, **PyFeet** needs to be thought of as a single-page application. The main components of the visualizations are:

- the time series, which depict value changes read from sensors with time;
- feet icon showing the position and the value on the sensors using colors;
- the intractable list of revealed anomalies and their distribution over time

All of the data can be acquired for each of six patients.

Implementation

The effect is achieved through the communication of the two Python modules: **main_app.py** and **DashDBmanager.py**

DashDBmanager.py is designed to be a singleton class of a database manager. Using the **parseJSON** method, it turns acquired data into a record in a database. The columns are:

- ID
- DATE
- USERNAME
- DESCRIPTION
- S1_VALUE, ..., S6_VALUE
- S1_ANOMALY, ..., S6_ANOMALY
- IS_ANOMALY

The other key functionalities are:

- **list_anomalies** method, which finds all records of the given patient with the anomaly on any sensor.
- **select_area** method, which returns a record with the anomaly, along with the records from 5 seconds before and after it, for the reason and the aftermath of the anomaly occurrence to be illustrated.

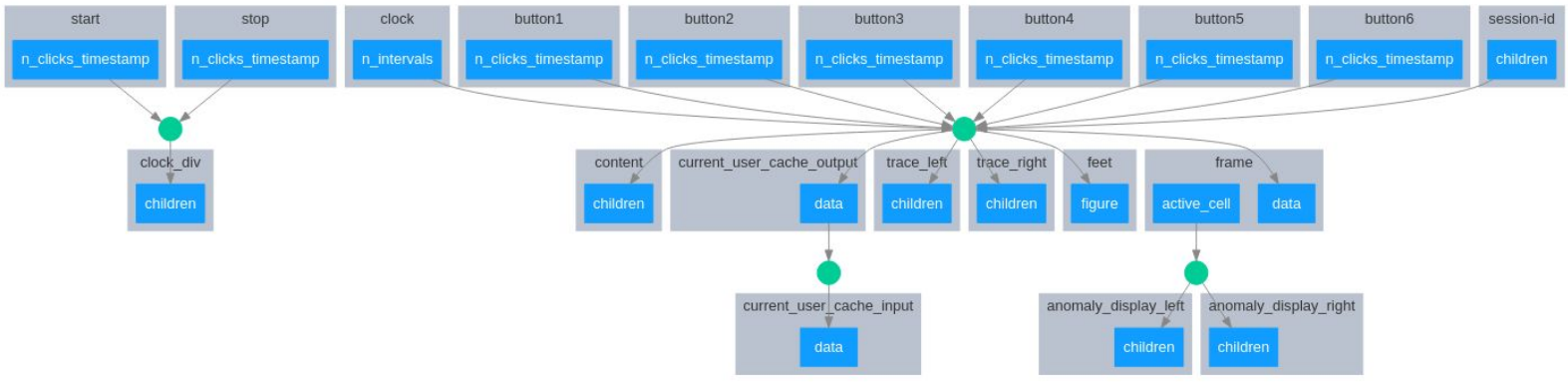
In **main_app.py** the database manager instance is created for each session of **PyFeet**, and is stored in a dictionary. A single database is used for every session, which is represented by **walktraceDB.db** file.
API returns the data no more frequently than one **JSON** per second, thus the refreshing rate of the **graphs** is similar. The graphs themselves are **Graph** objects from

dash_core_components. The values on the graphs is rotated and redrawn every time new data appears. The maximum length of the graph is 600 seconds. The buttons are designed to **stop** and **continue** the illustrating process.

The value of the sensor is mapped to the color and represented in a **feet graph**, which is essentially a **scatter plot.**

Whenever an anomaly is revealed, a corresponding record is added to the **interactable table**. The performance is controlled by the **display_anomaly** method. Each record in the anomalies table can be pressed, which results in **two** new **graphs** appearing, which illustrate the **anomaly occurrence**, along with the values from preceding and succeeding seconds. What happens here is the request to the database (**select_area** method). The anomaly displays are then updated with the fetched data.

The **callback diagram** looks like this:



Release

The whole application is open-sourced and can be found under [this address](#).