Denys Pedan #23011350

Assignment 1

Overview:

The project is a Flask web application designed to collect, store and visualize data from SenseHat. Collecting and storing data in a database. Analyzing temperature trends and displaying historical temperature data.

Data Collection and Storage:

The data from the sensor is written to the database. Before writing the data to the database, the data is validated first. If there is no .db file, then the program will create it itself at the first launch. And it will write data to it.

Web Interface:

Simple and clear interface for tracking data from the sensor in real time. The user does not need to refresh the page for new data, as the site itself dynamically updates the data. Warning messages are highlighted in red and are located under the data itself on the site. The user can also customize the limits when warnings appear on the Setting page.

Data Analysis and Reporting:

A simplified version of data analysis for spikes and trends was implemented. Temperature jumps are determined by the last record and the current one. If the difference is 5 units, then the jump will be registered. The trend is determined by the last 5 records in the database table. If the records grow or fall every time, this will be a trend. That is, the trend is determined over 5 records. But the trend is not determined over years or months.

Why not AI:

It would be a good idea to implement AI for trend analysis, but for AI to correctly predict trends, a large amount of data is needed to train the AI and it would analyze the data using cloud services and report the trend through an API, for example. I looked at ready-made AI solutions, but they need to be trained and I don't have the right data on which I could train the AI. Therefore, I used a more simplified implementation of data analysis.

System Reliability and Error Handling:
I use validation of data from the user and from the sensor. I also have several try blocks that can catch errors and so that the program itself continues to work without interruption.

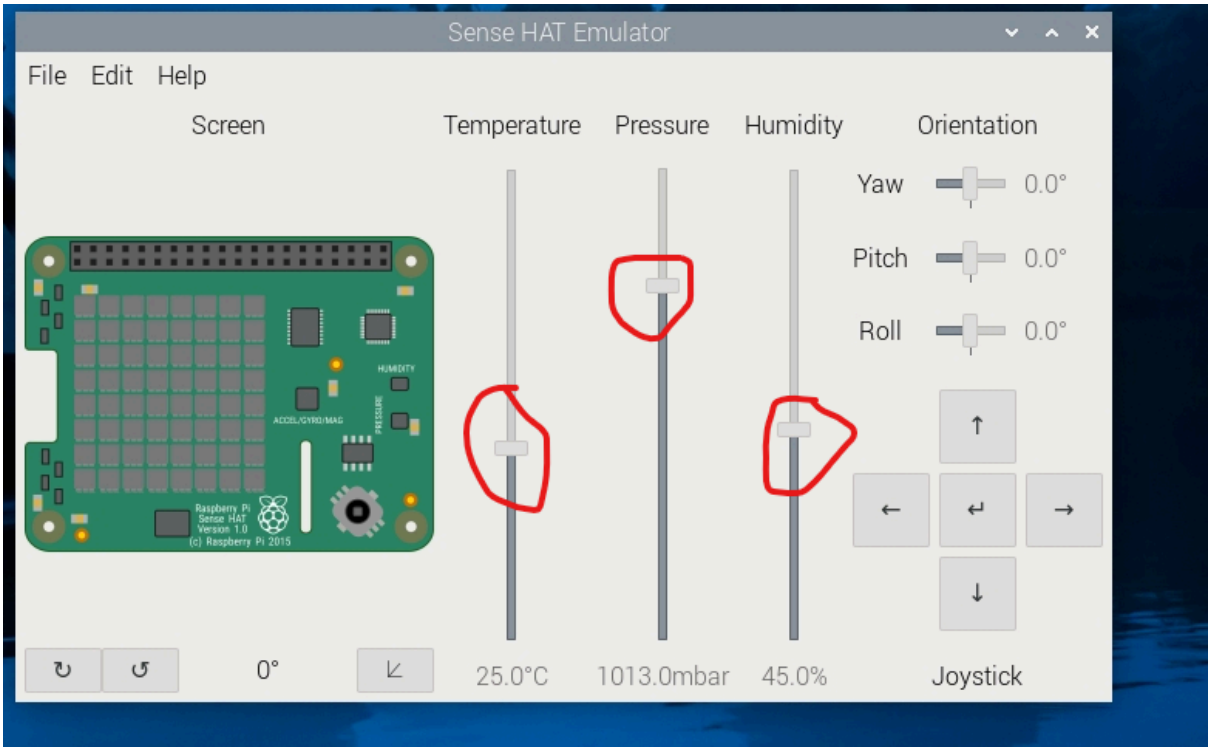
Code Quality and Maintainability:
The code itself is small, so I did not split it into files. The code itself is documented and easy to understand. In my opinion, if this code is split into files according to the principle of clean architecture, the project will become more difficult to understand. Since the project itself is too small.

Security:
Data validation, which improves security. Also, for production, you need to remove the Debug label. To improve security. Since the task itself did not say how many users the site is designed for. For many or for one. I left the settings tab public. Therefore, any user can change the warning label and this will affect all users. So if the site is designed for many users, there must be registration and authentication of each user, so that each user has a warning setting. If the site is designed for one user, you need to create an admin panel or simple authentication. So that the API is not public. But it seems to me that this is a little beyond the scope of the assessment. In the current implementation, public APIs do not violate security, and the post request is validated.

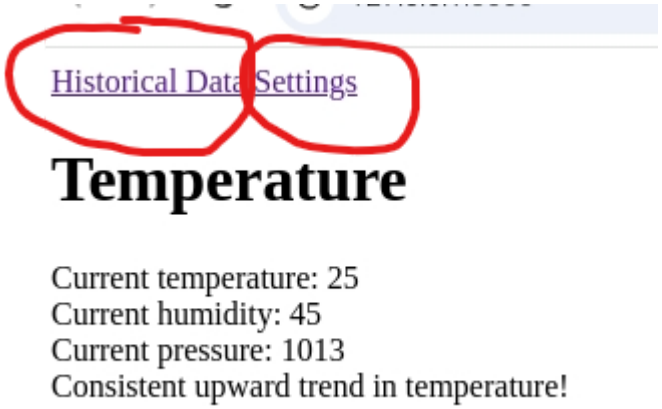
Setup and configuration instructions:
If your raspberry pie allows you to install these modules this way, then it is better to do so. Otherwise, it is better to create a virtual environment and download the modules there and configure the IDE or code editor for your virtual environment. Another way is to install modules into the global environment via the administrator command

pip install Flask
https://sense-emu.readthedocs.io/en/v1.0/install.html
pip install SQLAlchemy

User manual:
Run the main.py script. SenseHat Emu will start itself if it was not opened before. On the site navigation Settings, Historical data, and Home page.

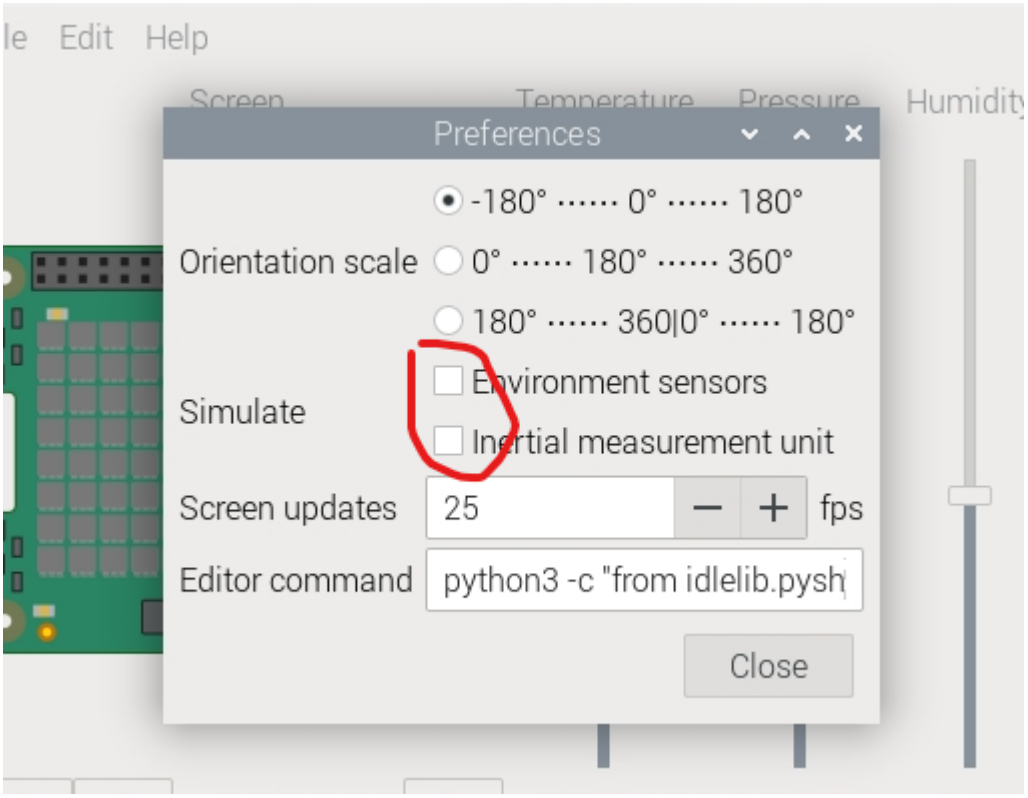


Change data to show on the website.



Simple navigation.

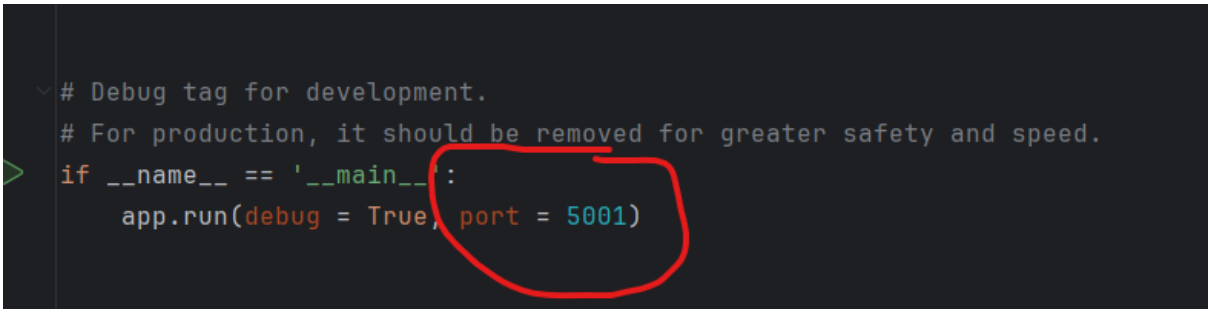
Troubleshooting guide for common issues:



Uncheck these two boxes to make the trend analysis function available. These emulator functions create noise in the data and the temperature can "jump" by 0.001. Since I do not have a noise smoothing function implemented and there is no AI for data analysis.

```
<head>
    <meta charset="UTF-8">
    <title>Historical Temperature</title>
    <!-- Chart.js library for creating charts -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <!-- Moment.js library for date manipulation -->
    <script src="https://cdn.jsdelivr.net/npm/moment@2.29.1"></script>
    <!-- Chart.js adapter to use Moment.js with Chart.js -->
    <script src="https://cdn.jsdelivr.net/npm/chartjs-adapter-moment@1.0.0"></script>
</head>
<body>
```

The libraries must be added in this order or there will be an error.



When I tested my project on raspberry pi. I encountered a problem that when I finished or stopped the project. But I had to change the ports, because it gave an error that they were busy. So you can just increase this number by 1