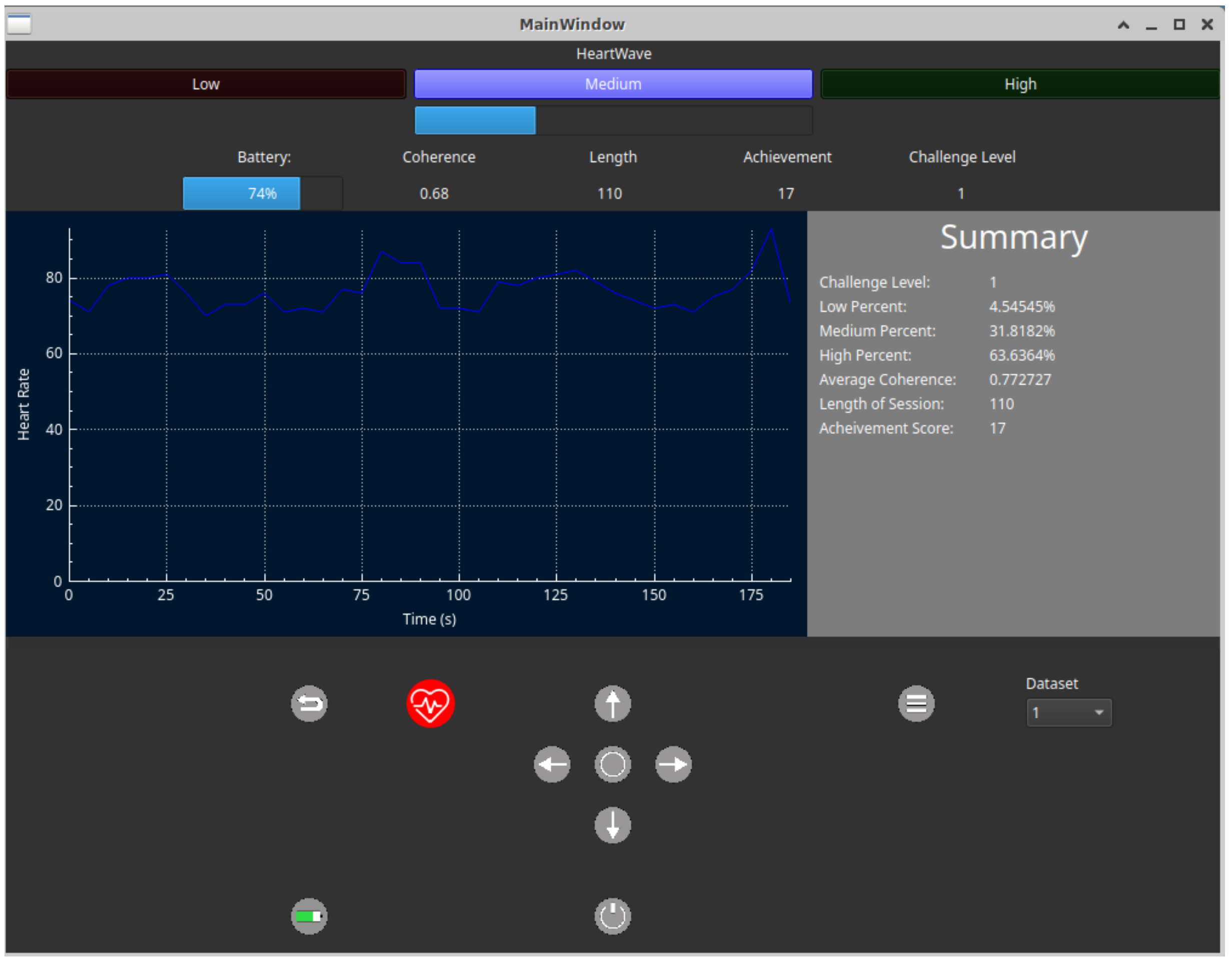
Design Explanation

Our implementation of the HeartWave device had to accommodate many features. Our design goals were based on the provided specification and functionality. We had to take into account the user interface, functionality, following a recognized architecture and testing that ensures the device is safe and functions as intended.

1.1 User Interface

HeartWave is designed to be a device for the average consumer. This requires a UI that is easy to use, understand, and navigate. Our final UI (user interface) design incorporates a clear layout that is easy on the eyes and simple to use controls. All the options in the Menu are clear and easy to understand. It is also very easy to navigate through the menu and the user is able to hide/show the menu so that there is less information displayed on the screen.

HeartWave User Interface



Battery:	Coherence	Length	Achievement	Challenge Level
74%	0.68	110	17	1

Summary

Challenge Level:	1
Low Percent:	4.54545%
Medium Percent:	31.8182%
High Percent:	63.6364%
Average Coherence:	0.772727
Length of Session:	110
Acheivement Score:	17

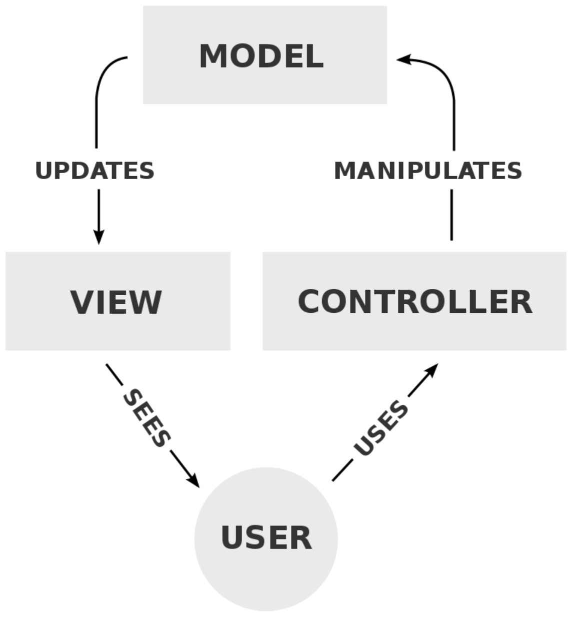
Dataset: 1

1.2 Functionality

HeartWave is designed to measure and analyze HRV (Heart Rate variability) patterns using a heart rate sensor. The information is analyzed and provides the user with feedback on their coherence levels. This value provides insight on the health and well being and ultimately helps the user reduce their stress. There are multiple elements to calculating this value, a user is able to synchronize their breathing with the onboard breathpacer. Pairing breathing and the heart rate collected by the sensor, we are able to calculate the HRV value and provide a coherence score through analysis. This score is then displayed on the devices screen along with the HRV graph. All the information from the sensors are collected, stored, and processed all within the device.

1.3 Design Architecture

There are many layers to the architecture of HeartWave, we have hardware (sensor), the layer responsible for storing data, the layer in which we analyze the data and the interface layer that will display the information to the user. Taking into account all these different aspects had lead us to adopt the Model View Controller (MVC) design architecture.



Frey, R. (2010). *Model-view-controller*. Wikipedia. https://w.wiki/znd

This model view controller architecture follows a very clear design flow. The user is presented a view that they are able to navigate and control which manipulates a model on the backend. The information is then processed in the backend and updates the view for the user. This architecture if followed thoroughly can help separate the “behind the scenes” workings of the device into their respective sections. Some of the many benefits of MVC is modular, scalable, and maintainable code. Adopting this design will also help reduce complexity and makes it easier to troubleshoot issues. This also makes it easier to assign different sections to team members, we were able to assign members to work exclusively on the front-end (view) and back-end (model). Collaboration between the two groups was required when implementing the controller aspect, as it requires the back-end components to connect to the front-end components.

We also have adopted an object oriented design to help implement our device, this means our code is modular and capable of scaling. The mainwindow (view + controller), interacts with the menu object (controller) that helps manipulate the device in order to carry out use cases and working functionality. The device object initializes and works directly with three other objects, the sessions (session also works with the breathpacer exclusively), history, and settings. When a new session is initialized through the device, it grabs the settings from the settings object and passes it into the session so that it can begin analyzing the heart variability input. During this time, the device also helps return related tuples of data that are displayed onto the screen for the user to see. At the end of the session, the device stores the session into the history objects vector member for it to be accessed at a later time. Any form of information that is displayed on the screen is sent up by relevant objects.

1.4 Conclusion

In conclusion, our implementation of the HeartWave device has been designed to meet standards of the consumer. With an easy-to-use interface and implementation of specified functionality that follows a widely recognized architecture (MVC). We were able to seamlessly assign tasks and carry them out with minimal issues to create the HeartWave device.