Table 1: Revision History

Date	**Developer(s)**	**Change**
26.09.2022	Udeep Shah	Inital commit
Date2	Name(s)	Description of changes
...	...	...

Development Plan
SE 4G06, TRON 4TB6

Team 26, STRONE
Jordan Bierbrier
Azriel Gingoyon
Taranjit Lotey
Udeep Shah
Abraham Taha

1 Team Meeting Plan

The team will meet in-person on Wednesdays at 4:30pm on a weekly basis. These meetings will typically be done in person at Thode Library or at the Capstone project room. However, additional meetings or daily check ups will be scheduled on a per need basis depending on the status of the project and upcoming deliverables. Additional meetings may either be held in person or virtually through the use of the team's Discord server.

2 Team Communication Plan

The team will have two primary sources of communication which include a Discord server as well as a Facebook messenger group chat. The Discord server is used for hosting virtual meetings as well as sharing files and useful links. Communication will also occur through commit messages and issues when collaborating on deliverables through Github. The purpose of using Github will also be for team collaboration as well as changing history and reverting opportunities.

3 Team Member Roles

Abraham Taha

Application Developer - This position is focused on creating a web application for the end user. This will allow the end user to interface with the hardware and change the sounds to detect.

Jordan Bierbrier

Signal Processing / Embedded Systems Developer -

Taranjit Lotey

Application Developer - Communication between application and hardware / backend development to send physical signals to user

Azriel Gingoyon

Signal Processing / Embedded Systems Developer - Cost-effective component research, wristband design, motor/microcontroller integration

Udeep Shah

Signal Processing / Embedded Systems Developer - Noise filtering, signal isolation resource optimization and power optimization

4 Workflow Plan

- Git will be used for collaborating, sharing, tracking and saving our work (code and documents). We will include a main branch, which will consist of the most updated working code. The main branch will connect sub branches of various hardware components, ideally each hardware component will have their own sub branch which will be connected via code in the development branch.
- Branches will be used when editing or adding code/ documents to the repository. Branches will follow a naming convention for consistency and coherency. They will be created based on issues, for example, IssueNumber-Description-Branch. As a result, allow easy branch linking. Additionally, comments or questions found within an issue can easily be traced to a particular branch.
- When a group member wants to merge a branch, a pull request must be created. Pull requests will consist of a comprehensive description of the incoming changes, and additional comments, such as tests conducted.
- We will also be using a pre product branch giving the capability to test functionality of referencing branches within each other before making a merge request to main.
- Additionally there will be a Gitlab continuous integration (CI) test to see if the incoming changes are acceptable (passing a linter test and unit tests). Furthermore, reviewers will be added to assess the changes and leave any comments which will be documented to review and store thought

processes at that current stage of development. Once the CI test passes and the reviewers are satisfied with the changes, the reviewers can accept the changes and the branch can be merged into main.

- Track issues within Github (Kanban Board) which will reference the issue to designated branch during the pull request. We can tag issues by priority (low, medium, high), by milestone (hardware, application, firmware, etc) and labels. Issues can be assigned to designated group members, and have reviewers to check up on status and work.

5 Proof of Concept Demonstration Plan

The major areas where the project could face issues are

- **Sound recognition** - We might not be able to isolate or classify signals accurately. This is especially hard when there is a lot of background noise.
- **Vibration motor** - There is a risk of the motor not being powerful enough to get the user's attention. This may seem easy to tackle by increasing the size and power of the motor but this issues a new issue of power consumption and size increase.
- **Integration of hardware onto a small footprint** - This is the biggest risk with our project. We might not be able to fit all the components i.e the microcontroller, motor, microphone and battery in the small footprint that we desire. The reason why this is hard to tackle is that the smaller we go with our hardware, especially microcontroller and battery, the more cautious we have to be with power consumption and resource allocation.

The risks associated with our project may change as our development is still ongoing. Our PoC (Proof of Concept) will try to focus on the integration of major features into a small footprint (still bigger than the final prototype). The main features of the PoC will be sound recognition for three different sounds, vibration alerts and app communication. The proof of concept targets the above risks. Sound recognition will be first tested on a computer using the computer microphone and then transferred to our PoC. The vibration motor also will be tested out on a person through the thickness of a wristband. The PoC will also give us an idea of how small we can compress the device without losing functionality. The successful demonstration of the PoC will certainly reduce the likelihood of not being able to deliver on a final product.

6 Technology

- Specific programming language
- Specific linter tool (if appropriate)

- Specific unit testing framework
- Investigation of code coverage measuring tools
- Specific plans for Continuous Integration (CI), or an explanation that CI is not being done
- Specific performance measuring tools (like Valgrind), if appropriate
- Libraries you will likely be using?
- Tools you will likely be using?

7 Coding Standard

- Leave concise and appropriate comments
- Name variables in Camel case
- Implement exception handling wherever possible
- Do not use the same identifier for multiple purposes
- Use proper indentation
- Conform similar pieces of code into a function
- Test the code whenever possible
- Avoid lengthy lines of code
- Abstain from deep nesting (nesting 4+ levels of code in the same block)

8 Project Scheduling