Development Plan
Software Engineering

Team #6, Six Sense
Omar Alam
Sathurshan Arulmohan
Nirmal Chaudhari
Kalp Shah
Jay Sharma

Table 1: Revision History

Date	**Developer(s)**	**Change**
2025-09-22	Nirmal, Sathurshan, Omar, Kalp, Jay	Initial Write-up

This document describes the development plan for the spatial awareness device *audio360*. It contains internal logistics that our team will adhere to, and measures in place to ensure the confidentiality of our device.

1 Confidential Information?

There is no confidential information to protect.

2 IP to Protect

There is no IP to protect.

3 Copyright License

Our team will be adopting the MIT License.

4 Team Meeting Plan

The team will meet twice a week in-person at a predetermined location. Each meetings will limited to a maximum of 30 minutes. Additional meetings may be scheduled as needed during lecture times if course instructor has no content to share. Team meetings will be organized by the the meeting chair and follow the agenda similar to below:

- Individual blockers/updates since last time met (10 minutes)
- Individual next steps discussions (3 minutes)
- Pre-determined questions (7 minutes)
- Additional discussions (10 minutes)

The team will meet with the project advisor in-person once every two weeks, on Mondays from 12:30 pm to 1:00 pm. If the team finds that an in-person meeting with the advisor is not necessary, the team will send a concise update through email.

Meetings with advisor will be organized by the meeting chair and follow an agenda similar to below:

- Team updates from the past 2 weeks (5 minutes)
- Pre-determined questions (15 minutes)
- Additional discussions (10 minutes)

5 Team Communication Plan

The team will use the following communication methods for each of the scenarios listed:

- **General team communication:** Microsoft Teams Group Chat. Every team member is expected to send general queries and updates to the group chat. Try to avoid sending individual messages to team members regarding the project unless necessary.
- **Urgent team communication:** Microsoft Teams Group Chat and individual text messages. If a message is urgent, it should be marked as urgent in the group chat. If no response is received, individual text messages (SMS) can be sent to team members.
- **Communication with the instructor, TA, and supervisor:** University email with Outlook. Every team member is expected to be cc'd on emails sent to the instructor, TA, or supervisor.
- **Sharing Documents:** Github repository and Microsoft OneDrive. All project-critical documents need to be stored in the Github Repository. Large files that are not project-critical (e.g datasets, SDKs, etc.) can be stored in the university hosted OneDrive.
- **Development communication:** Github Issues and Merge Requests. All development-related communication that needs to be tracked for traceability should be done through Github Issues and Merge Requests. This ensures that critical communication and decisions are documented and can be referenced later if needed.
- **Meeting Scheduling:** If a meeting needs to be scheduled outside of regular team meetings and involves multiple team members, we will be using the Outlook Calendar to find a suitable time for everyone. It is expected that all team members will export their calendars to Outlook to reflect accurate availability. For one-on-one meetings, direct communication through MS Teams is encouraged. If a meeting needs to be scheduled with the instructor, TA, or supervisor, the team will use email to coordinate a suitable time.

6 Team Member Roles

- Leader (Sathurshan Arulmohan)
  - Point of contact for professor and TAs.
  - Ensures that the team is on track to meet deadlines.
  - Project planning and scheduling.
- Meeting chair (Kalp Shah)

  - Creates the agenda for meetings.
  - Facilitates the meetings.
  - Ensures that the team sticks to the agenda.

- Reviewer (Jay Sharma)
  - Reviews all deliverables before deadline to ensure all sections are completed.
  - Reach out to the reviewers of each section to ensure they approve it before the deadline.

- Note taker (Nirmal Chaudhari)
  - Takes notes during meetings.
  - Create meeting notes summary after the meeting ends to grasp meeting content.
  - Coordinates with project leader after meeting ends for action items.

- System Specialist (Omar Alam)
  - Manages and validates system design at a high level.
  - Responsible for hardware & software interfaces.
  - Handles budgeting and expenses, ensures total budget remains under $500.

7 Workflow Plan

Git will be used for version control and collaboration. Each branch will correspond to one or more issue in our Kanban board, with the following naming convention `issue/#<issue-number1>-#<issue-number2>-...`. If an issue has multiple sub-issues, then the sub-issues shouldn't be included in the branch name, only the parent issue number(s). Moreover, each commit should include what is being worked on (doc, fix, feature, test), the issue number that is being worked on, and a one line description of what was done. For traceability, each commit made to the branch should be specific to one issue.

When an issue is ready for review, a pull request must be created and assigned to at least one member of our team. For traceability reasons, all issues addressed in a PR must be linked in the `development` section. Github automatically makes this mapping if the issue number is in the commit message, which is why it's a rule. The repository settings have been configured so that each PR requires at least 1 approval before being merged. A default template for PRs has also been created so the assignee can provide information of what exactly is done in that PR, and for all PRs to follow a consistent format. A PR

should never be force merged into main if it contains pipeline errors.

Issues will be managed using the Kanban board projects linked to our github repository. We will have one general Kanban board project for miscellaneous tasks like adding more github workflows, and then one Kanban board project for every major deliverable. Each Kanban board project will have the same structure, with columns for Backlog, To Do, In Progress, In Review, and Done. Each issue will be assigned to a team member before it is moved to the "In Progress" column. Issues should only be moved to "In Review" when the work is complete and a PR has been created. Once the PR has been approved and merged, the issue can be moved to "Done". We will be using labels to classify the type of issue that was created. The following labels will be used: documentation, bug, Development, enhancement, meeting and Research.

Label	**Description**
Documentation	Tasks related to updating any of the files in the `/doc` folder of this repository, or any readme files for the project.
Bug	Issues or errors in the system that need to be fixed.
Development	Active system implementation work for new features and/or core functionality.
Enhancement	Improvements to existing features. Note: Development refers to new features that are introduced. This label is used if improvements are made to existing features.
Meeting	Scheduled discussions, planning sessions, or syncs between team members.
Research	Exploration, investigation, or prototyping to guide future development

Table 2: CLassification Labels

8 Project Decomposition and Scheduling

We will be using Github Projects, more specifically Kanban boards, to decompose and schedule our main milestones. Our Kanban boards will be available in the projects tab of our github organization.

We will have one general Kanban board for the entire project, for miscellaneous tasks that don't directly related to any main milestone. In addition, each main milestone will have its own Kanban board to track progress and decompose the deliverables into smaller tasks specific to that milestone. Listed below are a list of the main milestones (with deadlines) for which we will create Kanban boards for. Note, the dates listed in each milestone are subject to change.

1. Problem Statement, POC Plan, Development Plan (2025-09-22)

2. Req. Doc. and Hazard Analysis Revision 0 (2025-10-6)
3. V&V Plan Revision 0 (2025-10-27)
4. Design Document Revision -1 (2025-11-10)
5. Proof of Concept Demonstration (2025-11-11 to 2025-11-28)
6. Design Document Revision 0 (2026-01-19)
7. Revision 0 Demonstration (2026-02-02 to 2026-02-13)
8. V&V Report and Extras Revision 0 (2026-03-09)
9. Final Demonstration Revision 1 (2026-03-23 to 2026-03-29)
10. Final Documentation Revision 1 (2026-04-06)
11. EXPO Demonstration (TBD)

9 Proof of Concept Demonstration Plan

[What is the main risk, or risks, for the success of your project? What will you demonstrate during your proof of concept demonstration to convince yourself that you will be able to overcome this risk? —SS]

The biggest risk for the success of the project is the risk of not having a synchronized microphone array that can recognize the directionality of an audio source. This is the fundamental technology that the rest of the project and strech goals are build upon (excluding the audio classification pipeline).

The goal of the proof of concept demonstration is to showcase the functionality of a microphone array on a controlled environment - a table top lab surface. The microphone array should be synchronized and capable of recognizing the directionality of a single audio source (sine frequency sound) with a maximum error of 45 degrees on the 2D plane of the table. Though this demonstration doesn't showcase the ability of recognizing multiple audio sources with the directional microphone array on a smart glasses frame (which is the final goal), it showcases the feasibility of recoginizing the directionality of an audio source anywhere on a 2D plane with adequete accuracy. If successful, this would give us confidence to extend the project with the audio classification pipeline, development of recognizing more than one audio source, extending the audio analysis pipeline onto the smart glasses, and audio filtering on direction.

10 Expected Technology

10.1 Programming Languages

The following programming languages will be used throughout the project.

- **C/C++**: For developing embedded software running on the microcontroller.
- **Python**: For prototyping features and visualizing collected data for analytics.

All C/C++ header files and Pyhon code will include docstrings describing every file, class, and function to aid contributors. Docstring format will follow the convention outlined in the project's Coding Standard.

10.2 Libraries

C/C++ Libraries

Library	Description	Usage
FFTW	Fast Fourier Transform Library	Used to translate signals from time to frequency domain efficiently.

Table 3: C/C++ libraries that will be used.

Python Libraries

Library	Description	Usage
pyroomaudio	Room Acoustics Simulation Library	Used for prototyping audio array processing algorithms.
Numpy	Numerical Computing Library	Used during prototyping for handling larger computations.
Matplotlib	Plot Visualization Library	Used for plotting data.

Table 4: Python libraries that will be used.

10.3 Linter and Formatting Tools

ClangFormat will be used for autoformatting C/C++ code, ensuring compliance with the Coding Standard. Additionally, Microsoft's C/C++ extension in Visual Studio Code will be used as the C/C++ linter. pylint will be used as the Python linter.

10.4 Testing Frameworks

GoogleTest will be used as the main testing framework for C/C++ code. Since Python will primarily be used for prototyping and visualization, unit tests will generally not be written. However, if correctness becomes critical, pytest will be used.

Code coverage for C/C++ will be measured using gcov and visualized with lcov. Coverage metrics will not be collected for Python tests if any are written.

10.5 CI/CD

CI

1. Run all unit tests (*PR blocking*).
2. Build source code using a compiler compatible with the target microcontroller (*PR blocking*).

CD

1. Generate and publish code coverage reports.

10.6 Performance Measuring Tools

Due to hardware constraints of running on a microcontroller, performance profiling will rely on the profiler available within the microcontroller's IDE.

10.7 Tools

Tool	**Description**
GitHub Projects (Kanban board)	Track issue progress and status.
GitHub Milestones	Organize and group GitHub issues.

Table 5: Tools that will be used.

10.8 Hardware Components

- Four microphones
- Analog to Digital Converter (ADC) modules, one per microphone
- Microcontroller
- Screen (monitor / smart glasses)
- Glasses frame

11 Coding Standard

Coding language	Coding Standard
C	GNU C coding standard
C++	Google C++
Python	PEP8

Table 6: Team adopted coding standards.

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. Why is it important to create a development plan prior to starting the project?

   Omar Alam: Creating a development plan for a large project is crucial to ensure that the team is aligned on goals and that expectations are understood by everyone. It helps keep everyone accountable since we are all agreeing to the same rules and any disagreements on processes can be resolved by referring back to the plan.

   Kalp Shah: Creating a development plan for projects of this scope is really important as it gives us guidelines to follow during development. With large projects such as these, we're bound to encounter many roadblocks but having a concrete development plan that all members of the team agree on (which is the most important part to me) gives us a good rulebook to look back on and follow. I find thinking about the technologies and workflows we'll use as not as important though, compared to the team communication and roles sections at least. I feel like the technologies and workflows tend to change as we learn more about the nature of the project, but having set defined roles and guidelines between the members that clarify how to deal with conflict, what expectations are, who should be responsible for what, etc.

   Nirmal Chaudhari: Creating a development plan is important because it helps the team establish clear goals and standards. With everyone agreeing on the development plan, it will serve as a point of reference in the future, to ensure consistency among the team's work. It also helps

the team begin thinking about the requirements of the project, and what research needs to be done going forward.

Jay Sharma: Creating a development plan is important because it provides a roadmap for the project. It helps the team stay on track and ensures that everyone is working towards the same goals. It also helps the team identify potential risks and challenges early on, so that they can be addressed before they become major problems.

Sathurshan Arulmohan: A development plan serves as a guidance for the project. It lays out some of the key rules that the team agrees to adhere to, ensuring alignment and minimizing the risk of confusion or conflict later. Developing this plan also helps clarify priorities. For example, we have highlighted the importance of securing hardware components as early as possible and beginning software development on tasks that are independent of hardware availability.

2. In your opinion, what are the advantages and disadvantages of using CI/CD?

   Omar Alam: CI/CD has many advantages, most of which tie back to the concept of iterative development. It allows us to ensure that the code branch that is considered production-ready ("main") is always in a deployable state. It also allows us to catch bugs when multiple developers are working on and merging in the same codebase. The main disadvantage of CI/CD is the intiial setup time and feasability of using it. In some cases, CI/CD may not be feasible which might end up applying to our project. This is mainly due to the embedded nature of our project and the lack of support for CI/CD on embedded systems. Furthermore, setting up unit tests in CI/CD will require us to write platform agnostic code that can also run on the embedded micro-controller and our development machines. This will require extra effort and guidelines to get right.

   Kalp Shah: CI/CD is really good, espeically for larger pojects with a team such as these. In my eyes, they help the most with maintainability - having a test suite and various workflows that execute after each small iteration of the project. This just helps us ensure that we're not going to break something that we didn't intend to break. Espectially as you continue to add many features on existing ones over the course of many months, where you've probably already forgotten what worked how they work or they were just implemented by another team member entirely. CI/CD really helps to act as the second pair of managing eyes to make sure that everyone's code is always compatible with the past and with each others, keeping a functional version of the project available at all times.The main disadvantage, though, is that it's a lot of setup, and has it's own amount of overhead for maintenance. For example, always ensuring that the test suite is up-to-date after each feature iteration. Although those are just good software practices, so maybe they are beneficial in that way too, who knows. But I'm sure that if we're not careful, we might end up

with a lot of unnecessary workflows that just add to the complexity of the project.

Nirmal Chaudhari: Using CI/CD enables our team to develop code in an agile context, thus allowing iterative improvement of key features in our project. Moreover, CI/CD also pushes us to validate our code more frequently, using automated pipelines. This ensures the quality of our code, and if configured correctly, can help ensure that our code meets the requirements we initially set. Despite these advantages, CI/CD may be limited when it comes to developing software or a product within a small time frame (like we are doing for our capstone). For projects with small deadlines, it may not be feasible to go with the iterative approach, since we may be forced to fully implement the project anyways. Also, when considering things outside of software, like embedded systems, iterative development doesn't really work if the hardware we are using is just a single component.

Jay Sharma: In my opinion, the main advantage of CI/CD is the reliability it brings to the development process - we can catch issues early, keep code deployable, and deliver features faster. The downside is that setting up and maintenance can be complex, and if not done well, they can slow things down instead of helping.

Sathurshan Arulmohan: The main advantage of CI is automation of build checks and tests, ensuring that the software is always in a working state. This reduces the risk of introducing broken code into the main branch and provides developers with immediate feedback. In our project, this is especially useful since development occurs on different OS, and we may not have access to the same microcontroller compiler locally.

The advantage of CD is that updates are automatically delivered to the target environment whenever new changes are merged. This ensures that the deployed system always contains the latest code, reducing manual effort.

On the other hand, CI can sometimes hinder progress. For instance, if unrelated tests fail, developers may spend significant time debugging issues that are not caused by their own changes in a PR, slowing down feature delivery. Similarly, CD can introduce challenges when automatic deployment is not desirable. In some cases, it is preferable to release stable versions on a fixed schedule rather than pushing the latest changes immediately.

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

Omar Alam: Since the team project is exepcted to require embedded hardware, we had disagreements on whether or not we could adequately specify elements needed to complete the "Expected Technology" section. We resolved this by researching the hardware we might end up using and

determining that we could make reasonable assumptions on the technology we would be using.

Kalp Shah: Personally, I don't think we had any disagreements on anything. I think there were just sections, however, that we were just unknowledgable about. For example, the "Expected Technology" section (hardware specifically) was one that sparked a lot of discussion since none of us had really worked with much hardware before - and definitely not in the field of audio processing. But after conducting some research and getting in touch with our supervisor (who gave us really good insight), we were able to move past that as well.

Nirmal Chaudhari: Our team didn't really have any disagreements while working on this deliverable. The only disagreement related to the work I was doing was naming convention for branches, and whether more than one issue should be implemented in a single branch. We resolved this by meeting up in person, and agreeing that a branch can contain multiple features.

Jay Sharma: In this deliverable, we experienced conflicting opinions on meeting times, branch names, and tools. However, we discussed options briefly, and voted on the best option. This kept us moving forward.

Sathurshan Arulmohan: During this deliverable, our team faced challenges in defining roles and responsibilities. While everyone agreed that each member should have a clear role, there were disagreements about the significance and scope of certain roles, as some appeared to overlap with others. After discussion and careful consideration, we resolved this by introducing the System Specialist role, responsible for both hardware and software integration. Since our team is primarily software focused, having a dedicated member specializing in hardware provided balance and ensured smoother integration across the project.

Appendix — Glossary

Audio Classification Pipeline: Software pipeline aiming to classify the sound sources. For example, the audio can be recognized as a car, person speaking, alarm, etc.

Appendix — Team Charter

External Goals

The team's primary goal is to gain expertise in embedded software development while contributing a novel technique to the broader scientific and engieering community. The main focus is that each member strengthens or acquires technical skills that will be directly valuable in their future careers. As a stetch goal, the team aims to rank within the top five projects at the Capstone Expo, showcasing both technical innovation and professional presentation.

Attendance

Expectations

The team expects all members to attend all team meetings and to be punctual. Every team member must indicate whether they will be attending by responding to the Outlook meeting invites through their university emails. Team meetings will begin at most 5 minutes after the scheduled start time. In-person meetings will be preferred over virtual meetings. Virtual meetings will be held only when more than 3 team members are unable to attend in person. If a team member is unable to attend a meeting in-person, they may be able to attend virtually if it is deemed viable by the rest of the team on a per meeting basis.

Acceptable Excuse

Unfortunate circumstances may arise that prevent a team member from attending a meeting. These circumstances include are encompassed in the following list:

- Illness.
- Family emergency.
- Medical or academic related appointment.
- Religious holiday.
- Other circumstances agreed upon by all team members.

In Case of Emergency

Try to predict absences beforehand, but emergencies still arise despite best efforts. In the case of an emergency, the absent team member must notify the team as soon as possible through Microsoft Teams. If the emergency is expected to last for a prolonged period of time, please refer to instructions provided by the course instructor to communicate with the Associate Dean.

Accountability and Teamwork

Quality

Our team expects all members to come to meetings fully prepared with completed work, clear progress updates, and constructive contributions to discussions. All deliverables must meet our established quality standards including proper code review, testing, and thorough documentation. We emphasize proactive communication, early problem identification, and mutual support to ensure the team's success and maintain high professional standards throughout the project.

Attitude

We expect all team members to maintain a positive, collaborative attitude and treat each other with respect and professionalism. We encourage open communication, constructive feedback, and diverse perspectives while ensuring all interactions remain productive and inclusive. Our team adopts a conflict resolution approach that prioritizes direct communication, active listening, and finding mutually beneficial solutions to maintain a supportive working environment.

Stay on Track

We will track team performance through regular check-ins, commit frequency, and meeting attendance to ensure everyone contributes their fair share. For members who exceed expectations and consistently deliver quality work, we will reward them with treats like Tim Hortons coffee or donuts to recognize their efforts. For those not contributing their fair share, we will initially cover for them and provide support, but after repeated instances, we will issue formal warnings and escalate to our TA or instructor if necessary to maintain team accountability and project success.

Team Building

We will build team cohesion by hosting social gatherings at our off-campus house, where we can work together on projects in a relaxed environment and spend time hanging out to strengthen our relationships. These informal settings will help us bond as a team while still maintaining productivity and collaboration on our project goals.

Decision Making

We will make decisions through consensus when possible, seeking common ground and agreement among all team members. However, when there is a clear divide or disagreement that cannot be resolved through discussion, we will hold a formal vote to reach a decision and move forward with the project. This approach ensures we respect everyone's input while maintaining progress when consensus cannot be achieved.