# Development Process

**Group #12 – Botanica**

**SproutBot**

Arun Mistry

Mina Demian

Nicholas Levantis

Usman Minhas

Table 1: Revision History

| Date | Developer(s) | Change |
|---|---|---|
| October 8, 2023 | Arun Mistry, Mina Demian, Nicholas Levantis, & Usman Minhas | Initial Development Process Plan |

# Table of Contents

# Overview

This document will outline how Botanica will effectively work to produce a working solution for the project SproutBot. Included in this document is an overview of process workflow, as well as details regarding roles and responsibilities, procedures, tools, and version control.

# Process Workflow

Our main method of workflow will be to split up into 2 parallel teams. One team will be responsible for working on the robot navigation and movement system, and the other team will be responsible for working on the plant maintenance system. Once both sides are worked on to completion, we will then focus on integrating both components together. Further details are given below.

## Navigation and Movement

Navigation and Movement refers to the section of the robot responsible for enabling the robot to move from one location to another, including both software and hardware portions of the system. While we must decide both the hardware required for movement and the software methods for navigation together, we will try to focus on the hardware side first.

| Project Step | Inputs | Outputs |
|---|---|---|
| Research | • Generate requirements on the appropriate hardware to use for navigation<br>• Generate requirements on what software methods are needed to navigate from point A to point B | • Appropriate hardware selection<br>  ○ Acceptance Criteria: The hardware components must be meaningful and total cost must be well below budget<br>• Appropriate software methods for navigation<br>  ○ Acceptance Criteria: The navigation techniques must be achievable by the hardware chosen |
| Movement Hardware | • Purchase the required components. This could include, but is not limited to, the motors to be used, battery sources, the wheels, the controller to connect everything together<br>• Select a placeholder for the plant care system, such as a box | • The appropriate hardware is selected for movement capabilities<br>  ○ Acceptance Criteria: All the chosen hardware components are able to interface with each other<br>• The robot is able to move<br>  ○ Acceptance Criteria: Robot can move between different points from different orientations when different inputs are provided to different motors<br>• The robot is able to balance<br>  ○ Acceptance Criteria: Robot does not tip over during normal movement |

| | | |
|---|---|---|
| Navigation Software | • Set up the coding environment<br>• Select a method of navigation. Navigation could include, but is not limited to, computer vision for object recognition, PWM for wheel movement, signal tracking | • The robot is able to navigate between different points<br>  ○ Acceptance Criteria: The robot's navigation software automatically provides different inputs to different motors in order for navigation to any given point<br>• The robot is able to avoid obstacles<br>  ○ Acceptance criteria: During movement, the navigation system steers robot away from colliding with any obstacles<br>• The robot is able to navigate towards a plant<br>  ○ Acceptance criteria: The robot recognizes where a plant that needs to be watered is, and moves towards it when needed |

## Plant Maintenance

Plant Maintenance refers to the aspect of the robot that is responsible for monitoring plant health and delivering nourishment when necessary.

| Project Step | Inputs | Outputs |
|---|---|---|
| Research | • Generate requirements on the appropriate hardware to use for plant monitoring, potentially including location sensors, moisture sensors, water delivery system & pump, and more.<br>• Generate requirements on what software methods are needed to adequately monitor the plant health and ensure the correct amount of water/nourishment is delivered. | • Appropriate hardware selection<br>  ○ Acceptance Criteria: The hardware components must be meaningful and total cost must be well below budget<br>• Appropriate software methods<br>  ○ Acceptance Criteria: The software must be compatible with the selected sensors & boards. |

| | | |
|---|---|---|
| Monitoring | • Purchase the required components. This could include, but is not limited to, sensors to be used, boards, power source, and intermediate components such as wiring.<br>• Develop software to communicate data to the robot | • The system can provide accurate and useful data about the plants health<br>  ○ Acceptance Criteria: information regarding at least one aspect of plant health, for instance moisture, can be extracted from the plant.<br>• The system provides the moving robot with information about the plants' location<br>  ○ Acceptance Criteria: The robot is capable of accurately determining the location of the plant. |
| Delivery | • Purchase components which could include a pump, water reservoir, piping etc.<br>• Develop software to dispense water onto the plant precisely when required | • The system can deliver water to the plant when it is needed<br>  ○ Acceptance Criteria: The plant receives an appropriate amount of water when it is needed. The water is delivered accurately, and spillage is not excessive |

## Integration

Once both individual components of the project are completed, we will then focus on integrating them together, resulting in a completed plant care robot.

| Project Step | Inputs | Outputs |
|---|---|---|
| Integration | • Combine the robot's movement system with the plant maintenance system | • The completed robot is able to move around and water plants. There are multiple acceptance criteria for this output:<br>  ○ Both individual components' software are combined and run on one controller<br>  ○ Both individual components' hardware are integrated into one unified robot<br>  ○ The robot is able to navigate between different points<br>  ○ The completed robot is able to water different plants |

# Roles and Responsibilities

## Roles

There are 2 separate sections of the robot that can be identified, the movement system and the plant care system. As such, there are 2 separate sub teams that will focus on working on each half of the system.

### Navigation and Movement

| | Arun Mistry | Mina Demian |
|---|---|---|
| Navigation Software – Software implementation on navigating between points | Subsystem Lead | Subsystem Co-Lead |
| Movement Hardware – Hardware build on allowing the robot's movement | Subsystem Co-Lead | Subsystem Lead |

### Plant Maintenance

| | Nicholas Levantis | Usman Minhas |
|---|---|---|
| Plant Care Delivery – Designing the water delivery system | Subsystem Lead | Subsystem Co-Lead |
| Plant Care Monitoring – Monitoring a plant's needs | Subsystem Co-Lead | Subsystem Lead |

## Accountability

A common failure point during projects lead by multiple members is the lack of communication, or the lack of accountability. To avoid this pitfall, we have decided to split the responsibility of one task within multiple people. For every task, 2 people will be responsible, the Subsystem Lead and the Subsystem Co-Lead for the specific section the task falls under. The other 2 people will be responsible for general review at a high level, ensuring that the task is completed satisfactorily.

The Subsystem Lead will hold the main responsibility on working on the task, and the Subsystem Co-Lead will be responsible for supporting the Subsystem Lead, as well as reviewing the task and approving its completion. The other team will act as external stakeholders and review progress of the section at a high level. For every section, the Subsystem Lead and the Subsystem Co-Lead are responsible for the successful implementation of their section's product subsystem.

### Documentation

All team members will be responsible for keeping information on tasks and goals up to date. All team members will also be responsible for working on all documentation required, such as Hazard Analysis, System Requirements, System Design, Verification and Validation documents, and any other documentation required.

## Procedures

### Meetings

The team plans to meet a minimum of 2 times per week, with at least 1 in person meeting. Tuesday 12:30PM – 1:00PM and Friday 12:30PM – 1:30PM are weekly tentative times. The mentioned weekly meetings are mandatory for all group members, but other meetings may be held throughout the week for more specific goals. Beyond internal group meetings, we will also have aim to have a weekly meeting with our supervisor, Alan Wassyng, Tuesday 1:00PM – 1:15PM.

For all meetings, an agenda will be set, and each member will give an update on their progress from the previous meeting. Team members will fill a shared and unified Meeting Notes document with any notes or points made by other members from the meeting regarding their work. The meeting will be concluded with a set of goals for each member to achieve.

### Standards

There are a few separate areas where the team believes setting a standard would prove to be beneficial, and they are given in the table below.

| Standard | Description |
|---|---|
| Software Development | When working with any software, it must be set up such that any other stakeholder may easily duplicate the software environment with the specific components required and pick up the project at the level it has reached.<br><br>When selecting any software to work with, a discussion must be held on the applicability of the chosen tool. For any software development to be done, a discussion must be held on what needs to be accomplished. These decisions must be agreed upon by the team responsible for working with the software. |
| Hardware Development | When working with any hardware, it must be set up such that any other stakeholder may easily duplicate the hardware environment with the specific components required and pick up the project at the level it has reached. |

| | |
|---|---|
| | When selecting any hardware to work with, a discussion must be held on the applicability of the chosen tool. For any hardware development to be done, a discussion must be held on what needs to be accomplished. These decisions must be agreed upon by the team responsible for working with the hardware.<br><br>Designs must be validated using datasheets, feasibility and applicability checks, and compared against the actual needs of the project. Designs must be verified once completed against the acceptance criteria for the task it was built for. |
| Coding Standards | There is no specific coding standard we will follow. However, we will be following a few general guidelines:<br>• Prioritize code readability wherever possible<br>• Avoid deep nested loops<br>• Avoid infinite loops other than the main function<br>• Do not use unsafe code such as "**goto**"<br>• Avoid writing long lines of code<br>• Split functions into sub functions to improve reusability<br>• Use meaningful variable and function names<br>• Separate large chunks of code into separate files and modules<br>• Indent code appropriately using tabs, where a tab equals 4 spaces<br>• Include meaningful comments explaining functions and modules<br><br>We will also be using linting tools such as PyLint for our Python code, and ESLint for our JavaScript code, to make sure that our code will follow the above coding standards. |

# Tools & Version Control

## Version Control & Continuous Integration

Our team has decided to use Git for version control, with a dedicated GitHub repository for this capstone project. We will be using a GitHub Flow branching workflow, where we would have our master branch which contains the code that is deployable, and we would be creating branches off the master branch for any features, enhancements, or bug fixes, as well as for documentation.

We would also be using pull requests (PRs) to review any code or documentation changes. Code changes PRs would be reviewed by the team member that did not work on it, according to the Roles & Responsibilities section, while Documentation PRs would be reviewed by all team members before being pushed to the master branch. The naming structure we decided for our branches would be "Type/Description", where the Type would be either Feature, Enhancement, Bug, or Documentation.

We have decided that we will not be using Continuous Integration as our project contains both software and hardware components and we have decided to use linting tools for our software components instead.

### Project Management

We have decided to use GitHub Issues for Project Management and keeping track of our development artefacts. Issues will be created for each task, after they have been identified by the team or any team member. When creating an issue, one of these issue types/labels will be chosen:

- Bug: Task used to report any bugs discovered during testing.
- Feature: Task used to list the major requirements to be implemented for a new feature or new request.
- Enhancement: Task used to list any enhancements or additional requirements that should be added to an existing feature.
- Documentation: Task used to track documentation that needs to be created or edited.

Issues would also contain a label stating what product subsystem that the issue is related to, and they can be related to multiple subsystems. The following labels would be used to identify which part the issue is specifically related to: Movement, Navigation, Plant Care Delivery, and Plant Care Monitoring. Issues would also have Hardware and Software labels, depending on the skills and tools needed to work on this issue.

### Project Board

As part of Project Management, we have decided to use the Project Boards in GitHub as a Kanban Chart. We have also decided to use the following statuses to track progress on the issues.

- To Do: Stories in this stage have been polished and are ready to be worked on.
- In Progress: Stories in this stage are assigned to a team member, and they are currently working on it. Before Story is moved to the Testing stage, the team member that worked on it will test it.
- Testing: Stories in this stage are currently being tested by another team member than the one that worked on it.
- Done: Once the other team member has reviewed and approved the changes, then the changes are pushed, and the Story is moved to this stage.

We have decided that the process for creating any new issues would be that the team's lead or co-lead would create the issue and would add all the needed details to the issue by conferring with their other team member. Once the issue is ready, it will be reviewed by the whole team in the next meeting.

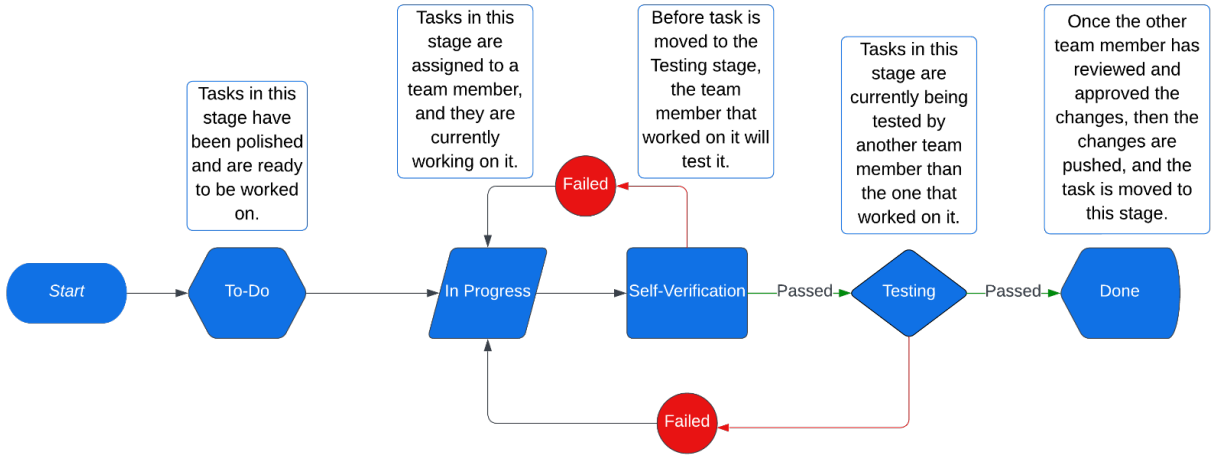


*Figure 1: Workflow of Issues in our Team's Kanban Chart*

## Development Artefacts

Any major changes will be discussed during the meetings and will be included in the meeting notes and tracked on the GitHub issue.

Any issues that are created will contain a description and all the needed files, such as screenshots, data sheets, etc.

Any Feature or Enhancement issues will contain the reason the feature and enhancement are needed for the overall success of the product, as well as the requirements for it.

Any Bug issues will contain the actual result and the expected result with a screenshot or video of the issue.

Any Documentation issues will contain the rubric and suggested content documents provided as attachments and will contain a plan on how and when we are going to create the document.

Since our project is a Mechatronics project and will contain Hardware and Software components, we have decided that issues that do not contain software or hardware languages and will not have code in the GitHub repository such as Hardware labelled issues would instead include the progress done, as well as pictures or videos.

Once issues are created, a testing plan with test cases will be created for them and mentioned in the issue description. These test cases will be used to determine if the issue is closed and should be moved to done or not.

When issues are completed, a message will be written in the commit message and the PR on GitHub mentioning the issue number, type and description of the changes done to the code, in the following format: [Issue #] Type: Description.

## Tools

| Tool | Description |
|---|---|
| Git | Git will be used for version control with its ability to track changes and maintain version history. Git will also be used to work collaboratively on the software. |
| GitHub | A web-based service that increases the usability and management of Git that will primarily be used for project management. GitHub also includes a kanban tool that will be used to track issues and cases. |
| Autodesk Inventor | Inventor will be used for 3D CAD design, modelling and simulations for both individual hardware components and assemblies. These parts and assemblies can be 3D printed for rapid prototyping. |
| KiCAD | KiCAD will be used for electrical and circuit design. KiCAD can be used for schematic design and verification using simulation as well as PCB design. |
| PrusaSlicer | PrusaSlicer will be used to convert CAD files created on Autodesk Inventor into printing instructions for 3D printing. |
| VS Code | VS Code is a code editor that will be used for editing and debugging. It's also useful due to its ability to be customized and features including integration with Git. |
| Python | Python is a programming language that will be used as our primary programming language and will use it to interface with the robot for abstract concepts such as computer vision. |
| PyLint | PyLint is a linting tool that will be used for error analysis and debugging, as well as maintaining good coding style when programming in Python. |
| React Native/ JavaScript | React Native is a JavaScript framework that will be used for creating a companion app for the user to interact with the robot. |
| ESLint | ESLint is a linting tool that will be used for error analysis and debugging, as well as maintaining good coding style when programming in JavaScript. |
| Arduino IDE | Arduino IDE will be used for writing and uploading code to the Arduino hardware. |

For all the listed tools, we will be using the latest version available that is compatible with the hardware and other tools.