Sprint 3: Agility - System Design Document

December 4, 2023

Table of Contents

[1. Executive Summary 3](#_Toc21616852)

[1.1 Project Overview 3](#_Toc21616853)

[1.2 Purpose and Scope of this Specification 3](#_Toc21616854)

[2. Product/Service Description 3](#_Toc21616855)

[2.1 Product Context 3](#_Toc21616856)

[2.2 User Characteristics 3](#_Toc21616857)

[2.3 Assumptions 3](#_Toc21616858)

[2.4 Constraints 3](#_Toc21616859)

[2.5 Dependencies 4](#_Toc21616860)

[3. Requirements 4](#_Toc21616861)

[3.1 Functional Requirements 5](#_Toc21616862)

[3.2 Security 5](#_Toc21616863)

[3.2.1 Protection 5](#_Toc21616864)

[3.2.2 Authorization and Authentication 6](#_Toc21616865)

[3.3 Portability 6](#_Toc21616866)

[4. Requirements Confirmation/Stakeholder sign-off 6](#_Toc21616867)

[5. System Design 6](#_Toc21616868)

[5.1 Algorithm 6](#_Toc21616869)

[5.2 System Flow 6](#_Toc21616870)

[5.3 Software 6](#_Toc21616871)

[5.4 Hardware 6](#_Toc21616872)

[5.5 Test Plan 7](#_Toc21616873)

[5.6 Task List/Gantt Chart 7](#_Toc21616874)

[5.7 Staffing Plan 7](#_Toc21616875)

# Executive Summary

## Project Overview

The project involves coding a robot to complete different tasks while traveling around a set course. We hope to show a less tech-savvy audience the capabilities of coding and how far coding has come.

## Purpose and Scope of this Specification

The purpose of this specification document is to outline the project requirements and constraints for coding a robot. It serves as a comprehensive guide for the development team, stakeholders, and other interested parties involved in the project. The primary objective is to ensure a clear understanding of what the robot coding project entails, including what falls within the project's scope and what is excluded from it.

Intended Audience: The intended audience for this specification includes:

1. Development Team: Software engineers, hardware engineers, and other technical staff responsible for coding and assembling the robot.
2. Project Stakeholders: This may include project managers, product owners, and anyone responsible for overseeing the project's progress and ensuring it aligns with organizational goals.
3. Quality Assurance Team: Testers and quality assurance professionals responsible for validating that the robot meets the specified requirements.
4. Business Analysts: Individuals responsible for gathering and documenting business requirements and ensuring alignment with the project's objectives.
5. Management: Senior management or executives who need an overview of the project's scope, constraints, and goals.

In Scope: This document addresses requirements related to the initial phase of the Robot Coding Project:

1. Robot Hardware Development: Design and assembly of the physical robot, including sensors, actuators, and other components.
2. Robot Software Development: Coding the software that controls the robot's behavior, including its movement, navigation, and interaction with the environment.
3. Integration with External Systems: If necessary, any integration with external systems, such as remote control interfaces or data exchange.
4. Testing and Quality Assurance: Validation of the robot's functionality and performance.
5. Documentation: Creating user manuals, technical documentation, and maintenance guides.

Out of Scope: The following items are explicitly excluded from the initial phase of the Robot Coding Project:

1. Advanced Features Development: Any advanced features or capabilities beyond the basic robot functionality.
2. Additional Hardware Enhancements: Any significant hardware upgrades or enhancements not initially specified.
3. Compatibility with Future Technologies: Future-proofing the robot for technologies and standards that may emerge after this project's completion.
4. Phase 2 Development: Any modifications, improvements, or updates that are part of a subsequent phase of the project.
5. Legal Compliance Changes: Changes required to meet legislative or regulatory mandates beyond the scope of the initial phase.
6. Marketing and Sales: Activities related to marketing, sales, or deployment of the robot, such as market research or sales strategy.
7. User Training: Training programs or resources for end-users are not part of the initial coding phase.
8. Third-Party Integrations: Integrations with third-party systems or platforms beyond those initially specified.

It's essential to note that while some items are explicitly outside the scope of this initial phase, they may be considered in future project phases or documented separately to ensure transparency and alignment with the project's long-term goals. The scope and objectives may evolve as the project progresses, and those changes should be documented and communicated to stakeholders accordingly.

# Product/Service Description

The development of the robot coding project is influenced by multiple factors, including its purpose, market conditions, budget constraints, regulatory requirements, available technology, and operational environment. Considerations such as user experience, scalability, maintenance, security, and risk assessment play essential roles. These factors serve as the background and rationale for specifying detailed requirements, ensuring the robot aligns with market expectations, adheres to regulations, and meets user needs while maintaining quality, safety, and future adaptability.

## Product Context

The product, in this case, the robot developed as part of the coding project, may have relationships with other products, systems, or components depending on its intended use and functionality. Whether it's independent and self-contained or interfaces with various related systems will vary based on its design and purpose. Here are two scenarios to consider:

1. Independent and Self-Contained Robot:
   * In some cases, the robot may be designed to be independent and self-contained, performing its tasks without the need for external interfaces. For example, a cleaning robot in a household environment may operate autonomously, without direct connections to other systems or components.
   * Such a robot would have its sensors, control systems, power source, and onboard intelligence, making it self-sufficient for its intended tasks.
2. Interconnected Robot with External Interfaces:
   * In other scenarios, the robot may be designed to interface with a variety of related systems or components. This is common in industrial, healthcare, or research applications.
   * A diagram of the major components and relationships might include:
     + Robot Core: The robot's main hardware and software components, including sensors, actuators, control unit, and communication interfaces.
     + Central Control System: An external system that manages and coordinates multiple robots in a fleet.
     + User Interface: A human-machine interface that allows operators or users to control and monitor the robot's activities.
     + Data Communication: Interfaces with external data networks or cloud services for data exchange and updates.
     + Integration with Other Machines: The robot may need to interact with other machines or systems in a production line, sharing data or collaborating on tasks.
     + Environmental Sensors: Sensors within the robot that monitor the environment and gather data for navigation and decision-making.
     + Remote Control: A remote control system for operators to take over in case of emergencies or for specific tasks.
     + Cloud Services: Access to cloud-based services for software updates, data storage, and analytics.
   * The interconnections among these components may include wired and wireless data links, control signals, and power supplies.

The exact nature of these relationships and interfaces depends on the robot's design specifications and intended application. Some robots may operate independently, while others may be part of a larger network of interconnected devices and systems, creating a more complex ecosystem. The diagram can provide a visual representation of these relationships and help in understanding the robot's place within a larger system.

## User Characteristics

1. Student User Profile:
   * Type: Student
   * Experience: Limited experience with robotics and coding, primarily focused on educational activities.
   * Technical Expertise: Basic understanding of coding concepts, such as block-based programming, and some exposure to robotics in classroom settings.
   * General Characteristics: Typically younger users with a strong interest in learning about robotics. They may have varying levels of motivation and may use the robot for educational projects, competitions, or STEM-related activities. User-friendly interfaces and educational materials are essential to engage this group.
2. Faculty User Profile:
   * Type: Faculty
   * Experience: Experienced educators with backgrounds in teaching science, technology, engineering, or mathematics (STEM) subjects.
   * Technical Expertise: Proficient in instructional methods, pedagogy, and curriculum development. They may have some technical expertise in coding and robotics.
   * General Characteristics: Faculty members may use the robot as a teaching tool in classrooms or educational programs. They need support for integrating the robot into their curriculum, access to educational resources, and the ability to customize the robot's programming for specific learning objectives.
3. Staff User Profile:
   * Type: Staff
   * Experience: Variable, depending on the role within the organization. Staff members may include administrative, technical support, or operations personnel.
   * Technical Expertise: Ranges from non-technical to moderate technical expertise, depending on job responsibilities.
   * General Characteristics: Staff users may interact with the robot in administrative or operational contexts. They require ease of use, clear interfaces, and minimal technical barriers to perform their tasks. Training and support should be accessible for staff members with varying levels of technical knowledge.
4. Other User Profile:
   * Type: Other
   * Experience: Diverse backgrounds and experiences, including hobbyists, researchers, and individuals with specific application needs.
   * Technical Expertise: Variable, from novice to expert, depending on the individual's goals and interests.
   * General Characteristics: This group may have unique and specialized use cases for the robot. It may include researchers using the robot for experiments, hobbyists exploring robotics as a passion, or professionals seeking solutions for specific tasks. The product should provide flexibility and customization options to accommodate a wide range of user needs.

Understanding these customer profiles is crucial for tailoring the product to meet the needs and expectations of different user groups. By considering the varying levels of experience, technical expertise, and motivations of each user type, the robot product can be designed to be user-friendly, educational, and adaptable to a broad range of applications.

## Assumptions

1. Availability of Compatible Hardware:
   * Assumption: It is assumed that the necessary hardware components, including sensors and actuators, required for the robot's functionality are readily available and can be sourced without significant delays.
   * .: If certain hardware becomes unavailable or experiences delays, it may necessitate reevaluating the robot's design and specifications to accommodate alternative components.
2. Stable and Compatible Operating System:
   * Assumption: The development and operation of the robot assume the availability of a stable and compatible operating system for running the robot's software.
   * Impact: If the selected operating system becomes unstable, incompatible, or undergoes significant changes, the software and interface design may need to be adjusted or rewritten to ensure compatibility.
3. User Training and Expertise:
   * Assumption: Users, particularly in educational or academic settings, are assumed to have access to basic training or guidance on using the robot and have some level of technical expertise.
   * Impact: If users lack the assumed level of expertise or training resources, the product may need to provide more comprehensive user guides, tutorials, or easier-to-use interfaces to accommodate less experienced users.
4. Reliable Network Connectivity:
   * Assumption: The robot may rely on network connectivity for software updates, remote control, or data exchange. Reliable internet access is assumed.
   * Impact: If network connectivity is unreliable or restricted, the product may need to offer alternative update mechanisms or offline capabilities.
5. Availability of Development Tools and Environments:
   * Assumption: Developers and programmers are assumed to have access to the required software development tools, IDEs, and environments.
   * Impact: If certain tools or environments are not accessible or experience significant changes, the product may need to adapt to alternative development platforms.
6. Compliance with Regulatory Requirements:
   * Assumption: It is assumed that the robot complies with existing regulatory requirements and safety standards during the development phase.
   * Impact: If new regulations or standards emerge or if the robot does not meet expected compliance, adjustments to the product may be necessary to ensure legal and safety requirements are met.
7. Availability of Third-Party Integrations:
   * Assumption: The robot may integrate with third-party systems or platforms for certain functionalities. These integrations are assumed to be available and stable.
   * Impact: If third-party integrations become unreliable or change significantly, the product may need to adapt to alternative integration options or develop in-house solutions.

Understanding these assumptions is vital for risk assessment and contingency planning. If any of these assumptions prove to be incorrect or change during the project's lifecycle, the product requirements may need to be adjusted to ensure the product's successful development and usability.

## Constraints

1. Parallel Operation with an Old System:
   * Constraint: The robot may need to operate in parallel with existing legacy systems, and its design must accommodate integration with these older systems.
   * Impact: Design considerations should include compatibility, data exchange protocols, and possibly the need for middleware or translators to bridge the gap between the new robot and the old systems.
2. Audit Functions (Audit Trail, Log Files, etc.):
   * Constraint: The project may require the implementation of robust audit functions, including audit trails and log files, to track system activities, user actions, and data modifications.
   * Impact: The design must include features for generating and storing audit data, as well as user interfaces for reviewing audit logs and performing necessary forensic analysis.
3. Access, Management, and Security:
   * Constraint: The robot's design must adhere to stringent security standards and ensure controlled access to its functions and data.
   * Impact: The design must incorporate authentication, authorization, encryption, and other security mechanisms to protect against unauthorized access and data breaches. This includes secure user management and access control.
4. Criticality of the Application:
   * Constraint: The robot's application may be mission-critical, and its design must prioritize reliability, redundancy, and failover mechanisms to ensure continuous operation.
   * Impact: The design must consider high availability, disaster recovery, and fault-tolerant features to minimize downtime and mitigate the impact of system failures.
5. System Resource Constraints:
   * Constraint: The robot may have limitations on system resources, such as disk space, memory, or processing power.
   * Impact: The design should be resource-efficient, optimizing code and data storage to operate within the defined resource constraints without performance degradation.
6. Design Standards:
   * Constraint: The project may be subject to specific design or industry standards, such as programming languages or frameworks.
   * Impact: The design must adhere to these standards, ensuring compliance with regulations or industry best practices. This may include using particular coding languages, development methodologies, or software frameworks.
7. Hardware Compatibility:
   * Constraint: The design must consider compatibility with specific hardware components and peripherals required for the robot's operation.
   * Impact: The design should specify and support the necessary hardware interfaces and peripherals, ensuring seamless integration and functionality with compatible components.
8. Environmental Constraints:
   * Constraint: The robot may need to operate in specific environmental conditions, such as extreme temperatures or humidity levels.
   * Impact: The design should account for environmental constraints by incorporating suitable protective measures and component selection to ensure the robot's reliable operation under these conditions.
9. Safety and Regulatory Standards:
   * Constraint: The robot may need to adhere to safety and regulatory standards that influence its design.
   * Impact: The design should include features, materials, and control systems necessary to meet safety and compliance requirements, ensuring the robot's legal and safe operation.
10. Scalability and Upgradability:
    * Constraint: The robot's design should consider the potential for scalability and upgradability to accommodate future enhancements.
    * Impact: The design should incorporate modularity, flexibility, and compatibility with future hardware and software components, making it easier to expand and upgrade the robot's capabilities.

Understanding and addressing these design constraints is essential to ensure that the robot project meets its objectives, complies with relevant standards, and operates effectively and securely within its operational environment.

## Dependencies

1. Availability of Sensor Data:
   * Dependency: The robot's functionality may depend on real-time or periodic data from external sensors or sources.
   * Impact: The requirements should specify the data sources, data formats, and data update frequencies that the robot relies on. If the data is unavailable or delayed, it may affect the robot's performance and response.
2. Completion of Firmware Development:
   * Dependency: Certain firmware or hardware components may need to be developed or integrated into the robot before the software can be finalized.
   * Impact: The product requirements should define the sequencing of firmware and software development and the dependencies between these components to ensure a cohesive and functional system.
3. Availability of External APIs:
   * Dependency: The robot may rely on external application programming interfaces (APIs) to access data or perform specific tasks.
   * Impact: The requirements should specify the APIs used, including authentication and access requirements. Any changes or unavailability of these APIs may require adjustments to the software.
4. Regulatory Approvals:
   * Dependency: Regulatory approvals or certifications may be necessary for the robot to be legally sold or operated in specific markets.
   * Impact: The product requirements should address the need for regulatory compliance, and the design and development should align with these requirements. Delays or complications in obtaining approvals can impact the product's release.
5. Integration with Third-Party Systems:
   * Dependency: The robot may need to integrate with third-party systems, such as inventory management software or automation systems.
   * Impact: The requirements should detail the integration points, data exchange formats, and compatibility with these third-party systems. Changes or delays in these systems may affect the integration process.
6. Availability of Training Resources:
   * Dependency: Users may require training to effectively use the robot.
   * Impact: The product requirements should include user training and documentation. The availability and effectiveness of training resources will influence the user-friendliness and adoption of the product.
7. Prototype Testing and Feedback:
   * Dependency: Testing prototypes of the robot may provide valuable insights and feedback for refinement.
   * Impact: The requirements should include provisions for prototype testing and user feedback. Feedback from testing may lead to changes in the product's design and functionality.
8. Availability of Key Personnel:
   * Dependency: The project may rely on key personnel, such as domain experts or software developers.
   * Impact: The project plan should consider the availability of these key personnel and allocate resources accordingly. Delays or resource unavailability may impact the project timeline.
9. Market Research Findings:
   * Dependency: Market research may uncover specific user needs or preferences that influence the product's features and design.
   * Impact: The requirements should incorporate the findings from market research to align the product with market demands and customer expectations.
10. Hardware Procurement and Assembly:
    * Dependency: The availability of hardware components and the assembly process may affect the overall timeline.
    * Impact: The project plan and requirements should consider the procurement and assembly processes to ensure timely completion and delivery of the robot.

Understanding and managing these dependencies is essential for planning and executing the robot project successfully, as they can significantly impact the project's timeline, functionality, and user satisfaction.

## Functional Requirements

| Req# | Requirement | Comments | Priority | Date Rvwd | SME Reviewed / Approved |
| --- | --- | --- | --- | --- | --- |
| ENDUR\_01 |  |  |  |  |  |
| ENDUR\_02 |  |  |  |  |  |
| ENDUR\_03 |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| ENDUR\_XX |  |  |  |  |  |

## Security

### Protection

* Password Protection: Make sure only authorized users can access the robot by using strong passwords and only giving access to the right people.
* Keep Secrets Safe: Use encryption to protect information from being seen by people who should not.
* Guard the Doors: Use firewalls to control who can come in and use tools that can spot if something terrible is happening.
* Write It Down: Record what the robot does and who uses it so you can check it for problems later.
* Check the Information: Make sure the robot's information is not changed by mistake or on purpose. Use special checks to confirm it is still okay.

1. **Authentication and Authorization**:
   * Implement robust authentication to ensure that only authorized users or components can access the system.
   * Employ role-based access control (RBAC) to manage permissions and access rights effectively.
2. **Encryption**:
   * Encrypt sensitive data at rest and in transit to prevent unauthorized access and data breaches.
   * Use secure encryption protocols and algorithms to protect data confidentiality.
3. **Activity Logging and Audit Trails**:
   * Maintain detailed logs of system activities, including user actions and critical events.
   * Retain historical data sets for auditing and forensic analysis, enabling the detection of unusual or malicious activities.
4. **Code Review and Security Testing**:
   * Conduct regular code reviews to identify and address vulnerabilities in the robot's software.
   * Perform security testing, including penetration testing and vulnerability scanning, to proactively find and fix security weaknesses.
5. **Patch Management and Updates**:
   * Keep the robot's software and firmware up to date with the latest security patches and updates.
   * Regularly apply security updates to minimize vulnerabilities and exposure to potential threats.

These five factors represent the core aspects of securing a robot's system, encompassing access control, data protection, monitoring, code quality, and the maintenance of a secure and up-to-date environment.

### Authorization and Authentication

Authentication: Authentication is like proving who you are to access a system.

1. Username and Password: You need a username and a secret password to show you are the right person.
2. Fingerprint or Face Scan: Some systems use your unique fingerprint or face to confirm it is you.
3. Two-Factor Authentication (2FA): You need two ways to prove your identity, like a password and a code sent to your phone.

Authorization: Authorization is like saying what you can do once you've proven yourself.

1. User Roles: You might have different jobs or roles in a system, and each role can do different things.
2. Permissions: The system decides what actions you can take, like viewing, editing, or deleting stuff.
3. Access Control Lists (ACL): A list that says who can do what in a system.

1. **Strong Authentication**:
   * Enforce strong password policies.
   * Implement Multi-Factor Authentication (MFA).
   * Consider biometric authentication and certificate-based authentication.
2. **Authorization Control**:
   * Utilize Role-Based Access Control (RBAC).
   * Apply the Principle of Least Privilege.
   * Use Access Control Lists (ACLs) for resource-specific access control.
   * Implement dynamic authorization policies and Attribute-Based Access Control (ABAC).

By combining these practices, you can establish a robust security framework that ensures only authorized entities gain access to the robot's system while maintaining precise control over what they can do within the system.

## Portability

* Less Host-Specific Parts: Try to have fewer parts in your system that work only on a specific computer or operating system.
* Not Tied to One Place: Make sure your code only relies a little on things that only work on one type of computer or system.
* Use a Universal Language: Choose a programming language that works well on many different systems, like Java or Python.
* Work the Same Everywhere: Make sure your system works the same, no matter what kind of computer, network, or place you use it.
* Split Code into Pieces: Keep your code organized into different parts, and ensure they talk to each other. This helps to change parts that only work on one system.
* Test on Many Computers: Check that your system works on different types of computers and systems. Fix any problems that pop up.

1. **Percentage of Code that is Host Dependent**:
   * Minimize host-dependent code to ensure the system's core functionality remains independent of the host environment.
2. **Use of a Proven Portable Language**:
   * Select a widely portable programming language to simplify cross-platform compatibility.
3. **Environment Independence**:
   * Design the system to be independent of specific operating systems, networks, and development/production environments.
4. **Modularization and Abstraction**:
   * Organize the system into modular components with well-defined interfaces to facilitate component-level porting.
5. **Use of Cross-Platform Libraries and Frameworks**:
   * Incorporate libraries and frameworks that are designed for cross-platform compatibility, reducing the need for host-specific code.

# Requirements

* The course will start off by the robot going in a square
* The robot will face 3 objects that it has to avoid
* The robot will then go over a ramp
* Robot will knock down the pins

## Functional Requirements

| Req# | Requirement | Comments | Priority | Date Rvwd | SME Reviewed / Approved |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Security

### Protection

Secure Access: Ensure that only authorized individuals can operate the robot by implementing robust password protection and restricting access to the appropriate personnel.

Safeguard Confidentiality: Employ encryption techniques to shield sensitive information, preventing unauthorized individuals from accessing it.

Control Entry Points: Utilize firewalls to manage access points and deploy tools capable of detecting any abnormal or potentially harmful activities.

Documentation: Maintain a comprehensive record of the robot's activities and its users, facilitating the identification and resolution of issues at a later stage.

Verify Data Integrity: Regularly inspect the robot's information to ensure it remains unchanged, implementing specialized checks to confirm its integrity.

### Authorization and Authentication

Identity Verification: Authentication serves as the process of confirming one's identity to gain access to a system.

Credentials: To establish your identity, you typically require a username paired with a confidential password.

Biometric Verification: Certain systems employ unique features like fingerprints or facial scans to validate your identity.

Two-Step Verification (2SV): This involves using two distinct methods, such as a password and a code sent to your mobile device, to authenticate your identity.

Permission Granting: Authorization dictates the actions you can perform after successfully authenticating yourself.

User Roles: Within a system, different roles or positions may exist, each with distinct capabilities and responsibilities.

Access Permissions: The system determines the specific actions you are allowed to undertake, such as viewing, editing, or deleting content.

Access Control Lists (ACL): A roster specifying who possesses the authority to execute particular actions within a system.

## Portability

When we talk about making a system portable, we mean it can easily work on different computers and systems. Here are some simple things to consider:

Avoid Using Complicated Code for Specific Computers:

Try not to use code that only works on certain types of computers.

Keep the code simple and general so that it can be used on different machines.

Use Languages That Work Everywhere:

Choose programming languages that are known to work well on many different computers, like Java or Python.

Don't Depend Too Much on Specific Tools:

Make sure the system doesn't rely too much on specific tools or programs that only work on certain computers.

Make Sure It Works Everywhere:

Test the system to ensure it works the same way on different computers, networks, and environments.

Avoid making the system too picky about where it's used.

By doing these things, you make it easier for the system to be used on different computers without much trouble.

# Requirements Confirmation/Stakeholder sign-off

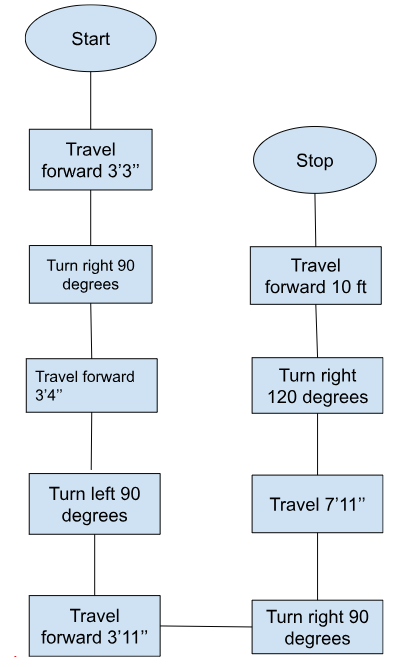
|  |  |  |
| --- | --- | --- |
| Meeting Date | Attendees (name and role) | Comments |
|  |  |  |
|  |  |  |

# System Design

## Algorithm

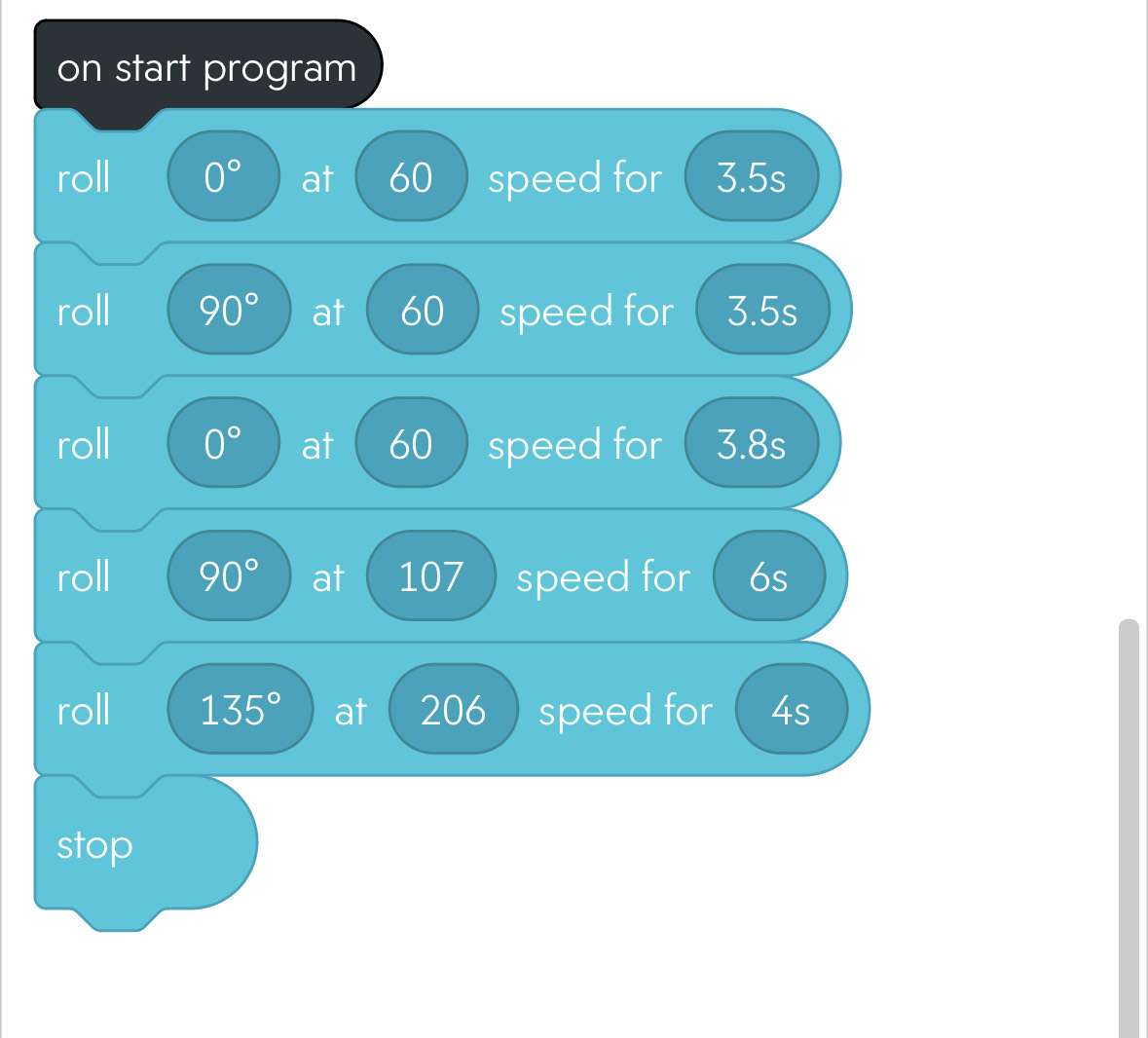
* Travel forward 3’3’’
* Turn right 90 degrees
* Travel forward 3’4’’
* Turn left 90 degrees
* Travel forward 3’11’’
* Turn right 90 degrees
* Travel 7’3’’ at high speed
* Turn right 120 degrees
* Travel forward 10 ft at high speed

## System Flow



## Software

We used the sphero.edu software to code our sphero SPRK+



## Hardware

We used a sphero SPRK+

## Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Reason for Test Case** | **Test Date** | **Expected Output** | **Observed Output** | **Staff Name** | **Pass/Fail** |
| To test if the sphero travel the set distance | 12/2 | Sphero travels set distance | Sphero traveled in circle | Trevor Nolan | pass |
| To test if sphero will travel the set course | 12/2 | Sphero travels set course | Sphero did not travel set course | Trevor Nolan | fail |
| Test corrections | 12/2 | Sphero travels set course | Sphero traveled set course | Trevor  Nolan | pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Task List/Gantt Chart

## Staffing Plan

| Name | Role | Responsibility | Reports To |
| --- | --- | --- | --- |
| David | Recorder/Typer | GitHub, Editing SDD |  |
| Trevor | Coder | Coding, Editing SDD |  |
| James | Typer | Editing SDD |  |