**Sprint 1 - Endurance Design Document**

**March 24, 2021**

Armand Valentino, Kristina Good

**Table of Contents**

**1.** **EXECUTIVE SUMMARY**

1.1 Project Overview

1.2 Purpose and Scope of this Specification

**2.** **PRODUCT/SERVICE DESCRIPTION**

2.1 Product Context

2.2 User Characteristics

2.3 Assumptions

2.4 Constraints

2.5 Dependencies

**3.** **REQUIREMENTS**

3.1 Functional Requirements

3.2 Security

*3.2.1* *Protection*

*3.2.2* *Authorization and Authentication*

3.3 Portability

**4.** **REQUIREMENTS CONFIRMATION/STAKEHOLDER SIGN-OFF**

**5.** **SYSTEM DESIGN**

5.1 Algorithm

5.2 System Flow

5.3 Software

5.4 Hardware

5.5 Test Plan

5.6 Task List/Gantt Chart

5.7 Staffing Plan

# 1. Executive Summary

## ***1.1*** ***Project Overview***

The goal of this project was to create a program that would make a Sphero SPRK robot complete 1 lap of a rectangular course.

## ***1.2*** ***Purpose and Scope of this Specification***

This project is to show that our team can create a program that shows how steady this robot can follow a course. This is only part 1 of the final project. For example:

**In scope**

This document addresses requirements related to Sprint 1: Endurance of Project A:

· Sprint 2 and 3 will be completed at a later date

**Out of Scope**

The following items in Sprint 1 of Project A are out of scope:

· Sprint 2 and 3 will test other elements of the robot, not just ‘endurance’

(Phase 3 will be considered in the development of the requirements for Phase 2, but the Phase 3 requirements will be documented separately.)

# 2. Product/Service Description

In this section, describe the general factors that affect the product and its requirements. This section should contain background information, not state specific requirements (provide the reasons why certain specific requirements are later specified).

## ***2.1*** ***Product Context***

How does this product relate to other products? Is it independent and self-contained? Does it interface with a variety of related systems? Describe these relationships or use a diagram to show the major components of the larger system, interconnections, and external interfaces.

## ***2.2*** ***User Characteristics***

Anyone can use this software and product, including younger people. For the most part they are also used for/by:

· Student/faculty/staff/other

· experience

· technical expertise

## ***2.3*** ***Assumptions***

List any assumptions that affect the requirements, for example, equipment availability, user expertise, etc. For example, a specific operating system is assumed to be available; if the operating system is not available, the Requirements Specification would then have to change accordingly.

## ***2.4*** ***Constraints***

Describe any items that will constrain the design options, including

· parallel operation with an old system

· audit functions (audit trail, log files, etc.)

· access, management and security

· criticality of the application

· system resource constraints (e.g., limits on disk space or other hardware limitations)

· other design constraints (e.g., design or other standards, such as programming language or framework)

## ***2.5*** ***Dependencies***

List dependencies that affect the requirements. Examples:

· This new product will require a daily download of data from X,

· Module X needs to be completed before this module can be built.

# 3. Requirements

· Describe all system requirements in enough detail for designers to design a system satisfying the requirements and testers to verify that the system satisfies requirements.

· Organize these requirements in a way that works best for your project. See **Error! Reference source not found.Error! Reference source not found.**, **Error! Reference source not found.** for different ways to organize these requirements.

· Describe every input into the system, every output from the system, and every function performed by the system in response to an input or in support of an output. (Specify what functions are to be performed on what data to produce what results at what location for whom.)

· Each requirement should be numbered (or uniquely identifiable) and prioritized.

See the sample requirements in Functional Requirements, and **Error! Reference source not found.**, as well as these example priority definitions:

**Priority Definitions**

The following definitions are intended as a guideline to prioritize requirements.

· Priority 1 – The requirement is a “must have” as outlined by policy/law

· Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits

· Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS **must be** either A or I" or "It **would be nice** if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

· A good requirement is:

· Correct

· Unambiguous (all statements have exactly one interpretation)

· Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)

· Consistent

· Ranked for importance and/or stability

· Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms and specify measurable quantities)

· Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)

· Does not specify any particular design

· Traceable (cross-reference with source documents and spawned documents).

## ***3.1*** ***Functional Requirements***

In the example below, the requirement numbering has a scheme - BR\_LR\_0## (BR for Business Requirement, LR for Labor Relations). For small projects simply BR-## would suffice. Keep in mind that if no prefix is used, the traceability matrix may be difficult to create (e.g., no differentiation between '02' as a business requirement vs. a test case)

The following table is an example format for requirements. Choose whatever format works best for your project.

For Example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req#** | **Requirement** | **Comments** | **Priority** | **Date Rvwd** | **SME Reviewed / Approved** |
| ENDUR\_01 | Follow the rectangle course, stop at the starting point |  | Important | March 31st | Completed |
| ENDUR\_02 | Make a figure eight, repeating the program 5 times |  | important | April 12 | In progress |
| ENDUR\_03 | Complete the agility course, making sharp turns, jumping the ramp, and hitting the ‘pins’ down |  | important | April 21 | In progress |

## ***3.2*** ***Security***

### **3.2.1** **Protection**

Certain factors must be ensured before the robot performs its activities. For example:

* flat surface
* no obstruction or obstacles (Sprint1)

### **3.2.2** **Authorization and Authentication**

Our team and progress will be overlooked by Professor Gil Eckert

## ***3.3*** ***Portability***

The Sphero company offers many products, including the SPRK. Our program can run on any SPrk device. Furthermore,

* Sphero SPRK can be operated anywhere, so long as the Sphero EDU program is running on a computer or laptop capable

# 4. Requirements Confirmation/Stakeholder sign-off

|  |  |  |
| --- | --- | --- |
| **Meeting Date** | **Attendees** | **Comments** |
| 03/24/2021 | Armand Valentno, Krstina Good | Completed part of Sprint 1 Project |

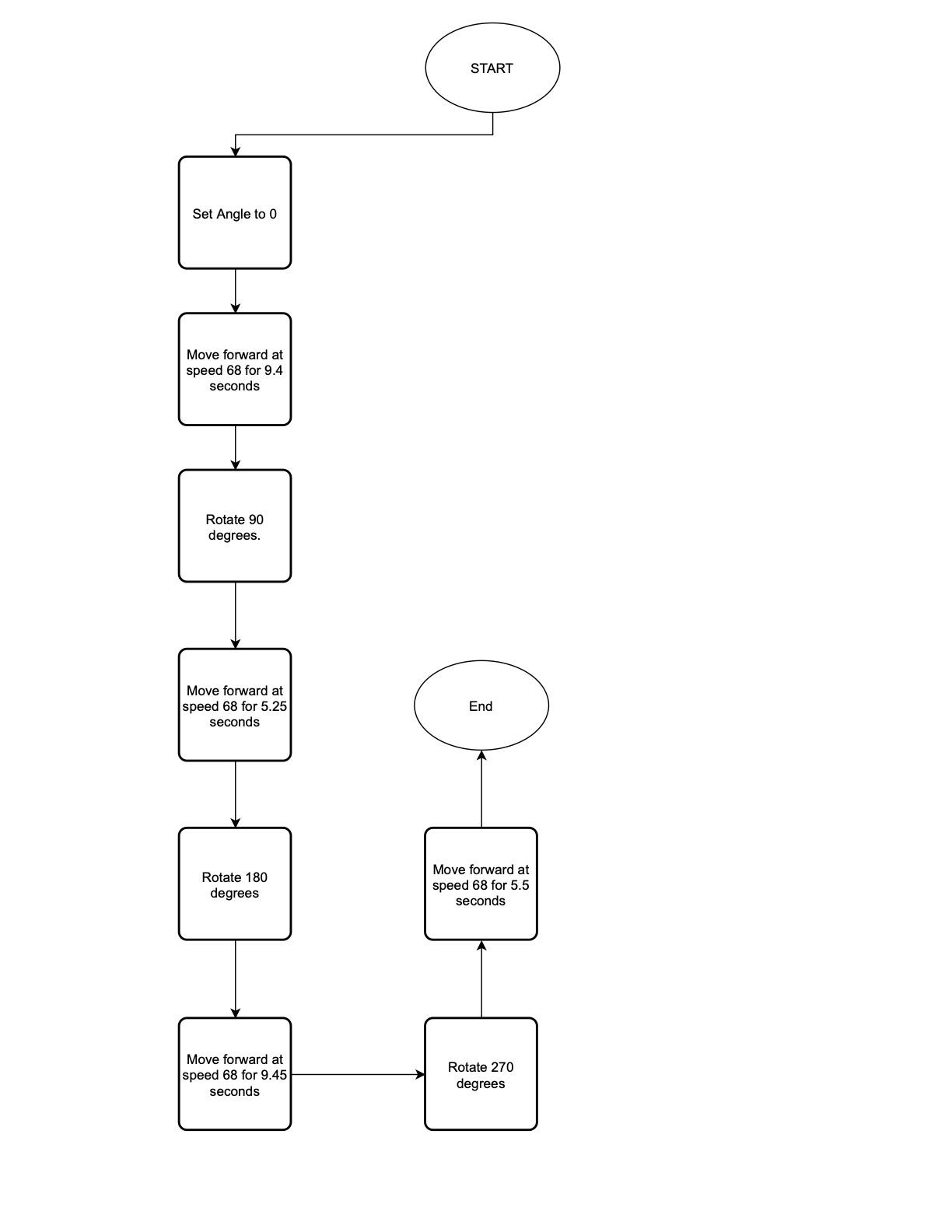
# 5. System Design

## ***5.1*** ***Algorithm***

The algorithm for Sprint 1 must complete a circuit that is shaped as a rectangle. The product must follow the rectangle lines at a steady pace. The coding will involve inputs controlling the moving and turning mechanisms of the product, in order for it to output 1 lap completing the circuit.

## ***5.2*** ***System Flow***

Below is the system flow chart mapping out what our block code for Sprint 1 should look like:



## The actual block coding program that was created and used on the robot is below, along with the sensor data showing the completion of 1 rectangular lap around the Endurance sprint court:

## ***5.3*** ***Software***

The software used to create and run this program is called Sphero EDU. Within the application, block coding is an option to create programs for the Sphero SPRK robot, which is what our team used.

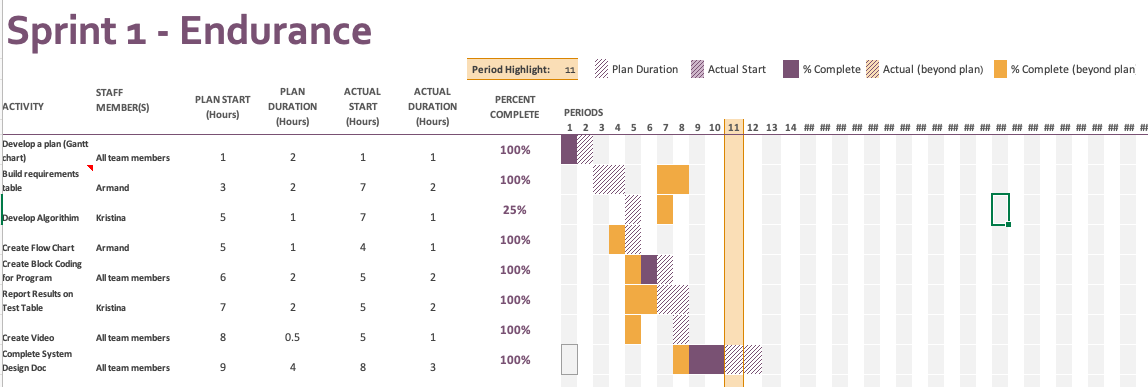
## ***5.4*** ***Hardware***

Our team used a MacBook to test our product and its coding design.

## ***5.5*** ***Test Plan***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Reason for Test Case** | **Test Date** | **Expected Output** | **Observed Output** | **Staff Name** | **Pass/Fail** |
| First long straight (running 10 seconds) | 3/24 | Reach first end point | Overshot the endpoint | Kristina, Armand | Fail |
| First long straight (shortened running time) | 3/24 | Reach the first end point, and rotate for second leg. | Stopped right at the first end point to turn. | Kristina, Armand | Pass |
| First short straight (running 5 seconds) | 3/24 | Reach the second end point, and rotate for third leg. | Stopped too short of the second end point | Kristina, Armand | Fail |
| First Short straight (shortened running time) | 3/24 | Reach the second end point, and rotate for the third leg. | Stopped at the second end point, ready to turn. | Kristina, Armand | Pass |
| Second long straight & second short straight | 3/24 | Complete the third and fourth legs of the circuit using the same running times as first long, first short | Did not exactly follow the second half of the course correctly | Kristina, Armand | Fail |

## ***5.6*** ***Task List/Gantt Chart***



## ***5.7*** ***Staffing Plan***

Below lists a table of contributors to the project, and what roles and responsibility

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Responsibility | Reports To |
| Kristina Good | Coder/contributor | Gantt Chart, Develop algorithm, videographer, completing system design | Professor Gil Eckert |
| Armand Valentino | Coder/coder contributor | Gantt Chart, Create code and flowchart, completing system design | Professor Gil Eckert |