Sprint 1 - Endurance Design Document

April 5, 2021

Table of Contents

1. Executive Summary

1.1 Project Overview

The purpose of this project is for the robot to be able to go through a triathlon course by following the algorithm and the audience would be the class. There are three parts to the triathlon; endurance, accuracy, and agility. Each part of the course gets more difficult.

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).

The robot is a sphere and us being used to get around an obstacle course without knocking anything over, light up, and speak. In the first sprint the robot has to be able to go in the shape of a square and also light up and speak.

1.2 Product Context

This product is related to other products because it is a robot and can be managed by different types of coding. For this specific project we are using block code. This robot would be self-contained because everything needed to control it is in the app. The app is what connects it to be able to control it through bluetooth.

1.3 User Characteristics

- Student/faculty/staff/other: students can learn how to use block code, teachers can gain experience by using it to teach.
- experience: gives one experience on computing, especially block code.
- technical expertise
- other general characteristics that may influence the product

1.4 Assumptions

Some assumptions are that this is an easy first part of the triathlon. THe equipment availability is pretty easy (using the robotics classroom) because Christina is very available with the room's hours.

1.5 Constraints

- parallel operation with an old system
- audit functions (audit trail, log files, etc.)
- access, management and security: sign into robotics classroom, figure out time with school
- 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)

1.6 Dependencies

- This new product will require a daily download of data from sphero app
- Endurance needs to be completed before we can start the next
- figuring out time to get to the classroom and test out robot

2. 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).

2.1 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	robot will start with green light and say ‘ready, set, go’ then	starts on yellow square with blue tape			

	move forward to first corner and stop				
ENDUR_02	after stopping robot turns red and says 'I'm done and I need water' then turn right at corner	moves forward then stops			
ENDUR_03	robot will move forward to next corner, stop, and turn right	third corner			
ENDUR_04	robot will stop at the third corner turn 90 degrees and then move forward	robot should end at beginning corner			
ENDUR_05	robot stops at beginning corner	robot ends			
ENDUR_XX					

2.2 Security

2.2.1 Protection

Specify the factors that will protect the system from malicious or accidental access, modification, disclosure, destruction, or misuse. For example:

- encryption
- activity logging, historical data sets
- restrictions on intermodule communications
- data integrity checks

2.2.2 Authorization and Authentication

Specify the Authorization and Authentication factors. Consider using standard tools such as PubCookie.

2.3 Portability

If portability is a requirement, specify attributes of the system that relate to the ease of porting the system to other host machines and/or operating systems. For example,

- Percentage of components with host-dependent code;
- Percentage of code that is host dependent;
- Use of a proven portable language;
- Use of a particular compiler or language subset;
- Use of a particular operating system;
- The need for environment-independence - the product must operate the same regardless of operating systems, networks, development or production environments.

3. Requirements Confirmation/Stakeholder sign-off

Include documentation of the approval or confirmation of the requirements here. For example:

Meeting Date	**Attendees (name and role)**	**Comments**

03/22/2021	Adrianna Lanfranco	confirmed
03/22/2021	Dylan Leray	confirmed…….

4. System Design

This section will provide all details concerning the technical design, staffing, coding, and testing the system

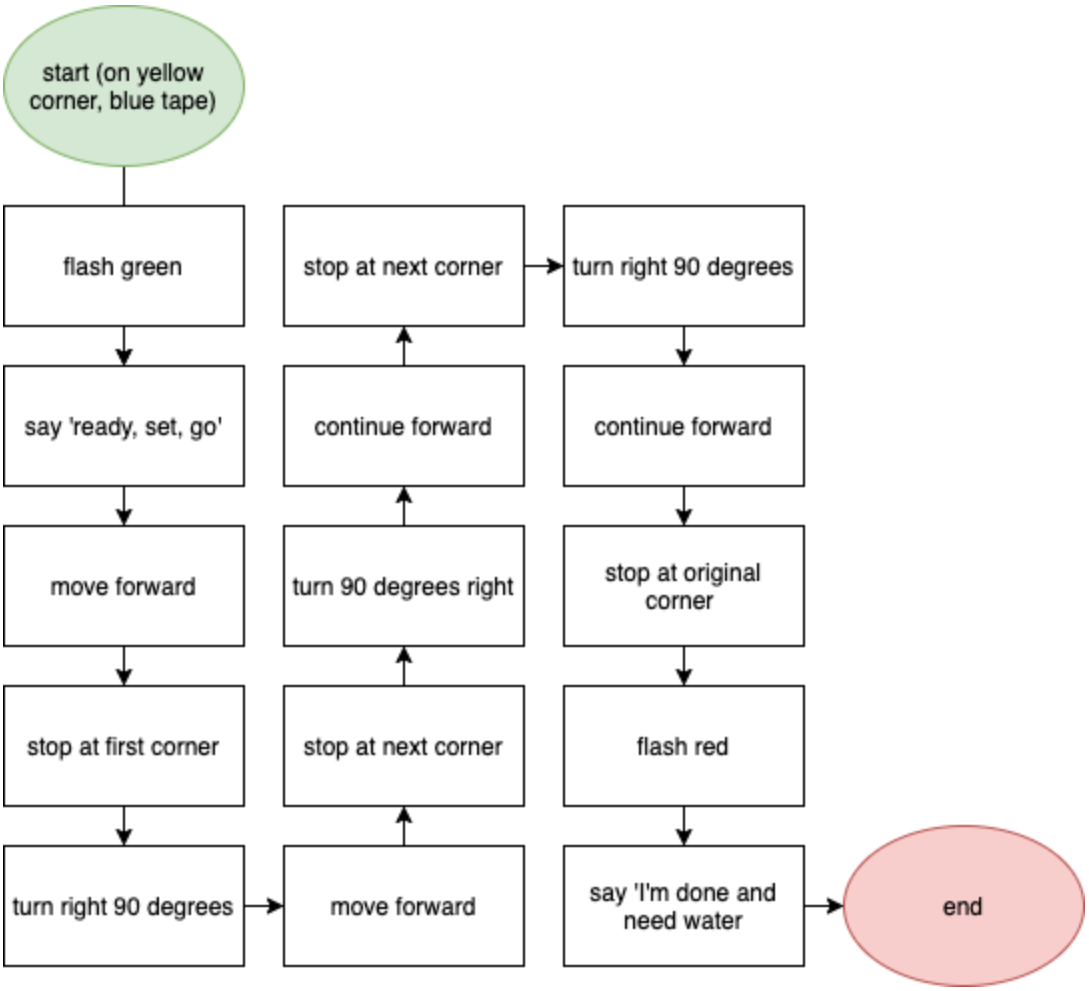
4.1 Algorithm

Develop and describe here the algorithm that will be used to provide the required performance of your software

- robot will begin at starting spot
- robot will flash green
- robot will say ‘ready, set, go’
- robot will move forward
- robot will stop at first corner
- robot will turn 90 degrees
- robot will move forward
- robot will stop at next corner
- robot will turn right
- robot will move forward
- robot will turn right at third corner
- robot will move forward
- robot will stop at original corner
- robot will turn red
- robot will speak ‘i’m done and need water’
- robot will then completely stop

4.2 System Flow

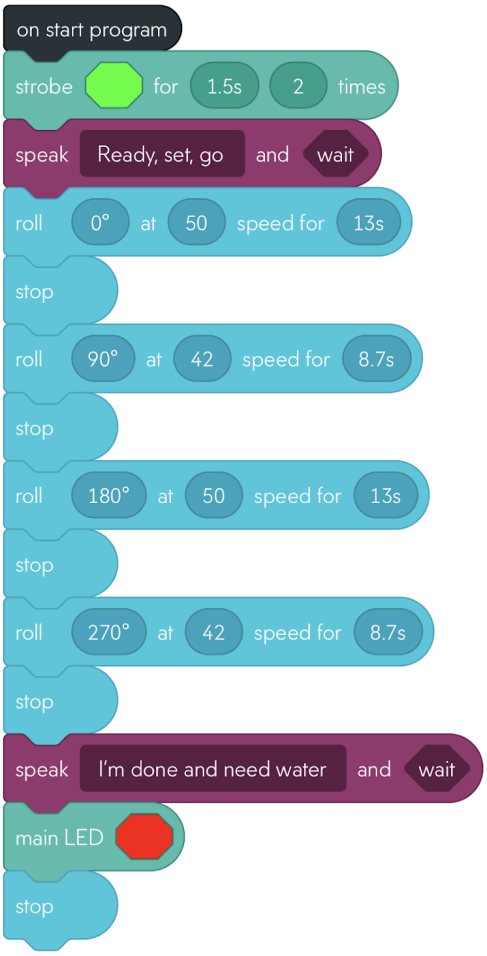
Develop a flowchart (and show here) that accurately depicts how your software application will act to fulfill the algorithm

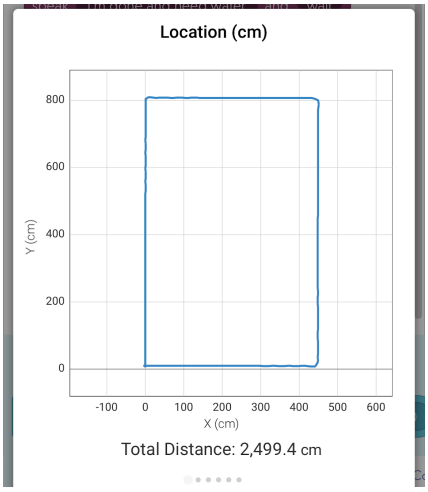


4.3 Software

Describe software languages/platforms/api's used to develop and deploy this application

- Sphero
    - block code





4.4 Hardware

Describe hardware platforms that were used to develop, test and demonstrate this application

- sphero robot
- adrianna: mac

4.5 Test Plan

Include a test plan showing all unit tests performed for this application, Include test rational, test date, staff member, pass/fail status

Reason for Test Case	Test Date	Expected Output	Observed Output	Staff Name	Pass/Fail
create code	3/30/21	figure out code	block code	adrianna lanfranco	P
test the code	3/30/21	code to work	took about an hour, robot kepet favoring the left side	adrianna lanfranco	P
practice the code multiple times	3/30/21	make sure code works more than one time	took about 20 min	adrianna lanfranco	p
record video	3/30/21	record final video	record video after testing	adrianna lanfranco	P
upload work to system design doc	3/30/21	make sure all documents are included	forgot a few(fixed)	adrianna lanfranco	P

4.6 Task List/Gantt Chart

Embed your gantt chart here
https://docs.google.com/spreadsheets/d/1mQ2GAvJUJ4ceKCrHbuEUnojyfPLMvCCb/edit#gid=1479078586

ACTIVITY	STAFF MEMBER(S)	PLAN START (Hours)	PLAN DURATION (Hours)	ACTUAL START (Hours)	ACTUAL DURATION (Hours)	PERCENT COMPLETE
Develop a plan (Gantt chart)	All team members	1	1	1	1	100%
Build requirements table	Adriana and Dylan	1	1	30 min	20 min	100%
Develop an algorithm	Adrianna	30 min	same	March 24	10 min	100%
flowchart	Adrianna	45 min	same	March 24	20 min	100%
block code	Adrianna	1	same	March 30	1	100%
test table	dylan	30 min	same			0%
staffing plan	all team members	30m min	same			0%
systems design document	all team members	1	same			0%
robot video	adrianna	1	same	March 30	10 min	100%
gitthub repository	adrianna	10 min	same	March 30	5 min	100%
Activity 11						0%
						0%

4.7 Staffing Plan

Insert a chart/table that depicts the roles and responsibilities of each team member that worked on this project

Name	Role	Responsibility	Reports To
adrianna lanfranco	member	flow chart, algorithm, code,video,	all members
dylan leray			
