Software Requirements Specification

for

NASA Robotics Competition

Version 1.0 approved

Prepared by:

Jonathan Sahagun
Jose Ambroso
Christopher Portugal
Christian Soltero
Mikey Thong

Advisors:

Dr. Jose Macias
Richard Cross

12/8/2018

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

This document will outline in detail the software and design requirements for the competition. This document will provide several views of the system's design in order to facilitate communication and understanding of the system. It intends to capture and convey the significant architectural and design decisions that have been made for the Swarmies.

1.1 Purpose

The purpose of this document is to present an understandable description of the architecture and design required of the Swarmies in the competition. This is version 1.0 of this document. This document includes descriptions of the hardware components that each Swarmie contains and the interactions between them and the software. It also gives a brief description of what this software will be used for. Additionally, it lists functional requirements for the Swarmies.

1.2 Intended Audience and Reading Suggestions

This document is intended for the project managers as well as the developers working on the Swarmies.

1.3 Product Scope

The software product will be an entry to the Swarmathon Competition.This software will be the core of a Swarmies logic when moving autonomously during the competition. On release the software shall allow Swarmies to move through the arena efficient and return the resource cubes. It should also be able to map the arena as it moves.

1.4 Definitions, Acronyms, and Abbreviations

See Appendix A: Glossary.

1.5 References

NASA Swarmathon Home Page: http://nasaswarmathon.com/

Swarmathon 2018 Guides Web Page: http://nasaswarmathon.com/2018-swarmathon-guides/

Swarmathon GitHub Page: https://github.com/BCLab-UNM/SwarmBaseCode-ROS

Gazebo Home Page: http://gazebosim.org/

Guide to run a Swarmie: https://github.com/BCLab-UNM/SwarmBaseCode-ROS

Guide for Swarmie OS installation: https://github.com/BCLab-UNM/Swarmathon-Docs/blob/master/PhysicalInstallGuide.md

Manual to assemble all physical components of a Swarmie: https://github.com/BCLab-UNM/Swarmathon-Robot/tree/master/AssemblyManual

2. Overall Description

The software being developed will be used to control the Swarmies during the competition.

2.1 Product Perspective

The software for the Swarmies is completely independent of any other product. This software is being developed for a competition. There is base code provided by the competition organizers and it is our liberty to alter certain parts of the code to find the best way to build upon this software. There are over 30 other competitors developing software with the same goal in mind, although implementation and approach may differ. Our final product will be different from any of the other competitors due to differing approaches in solving this problem.

2.2 Product Functions

The behavior of the Swarmie is built on the goal of retrieving april cubes and dropping them off at the collection zone. The Swarmie's initial movements and path are based on a search algorithm. If there is an obstacle detected, then the Swarmie will take the proper procedures to go around the obstruction and continue the path that was previously interrupted. Once an april cube has been detected, the Swarmie will stop and pick up the cube. Once the cube is secured in the gripper, then the Swarmie needs to return back to the base to drop off the cube.

2.3 User Classes and Characteristics

Only the Swarmathon organizers will use this software. As they are organizing the event they have intimate knowledge of how the rovers operate. The organizers also have an understanding of how the participants' code works because they get routine check-ins from the competition participants,

2.4 Operating Environment

The base code, framework, tools provided by the competition organizers was written for the Ubuntu 16.04 operating system. ROS is the framework required to operate the Swarmies. A Swarmie and its hardware components are interfaced to the computer running Ubuntu by a custom microcontroller that is compatible with the Arduino platform.

2.5 Design and Implementation Constraints

The design and implementation is limited to the Swarmathon rules and guides set by the competition organizers. A major constraint is the inability to make physical modifications to the Swarmies and some software packages may not be altered.

2.6 User Documentation

Guide to run Swarmie: https://github.com/BCLab-UNM/SwarmBaseCode-ROS. This online guide provides images and steps run a Swarmie. A small description is given at every step.

Guide to install the Swarmathon III Swarmie Disk Image: https://github.com/BCLab-UNM/Swarmathon-Docs/blob/master/PhysicalInstallGuide.md. This guide provides images and steps to install a disk image onto the Swarmie's onboard computer. The disk image contains all files needed to run a Swarmie and installs them.

Manual to assemble all physical components of a Swarmie: https://github.com/BCLab-UNM/Swarmathon-Robot/tree/master/AssemblyManual. This manual provides steps and ample pictures to put a Swarmie together.

For more information : http://nasaswarmathon.com/2018-swarmathon-guides/ . This webpages contains a list of guides for the Swarmathon 2018 competition and Swarmies.

2.7 Assumptions and Dependencies

We must assume that the Swarmies in Florida will be running the same OS, and will be on par or better maintained the and functional as the test Swarmies we are given.

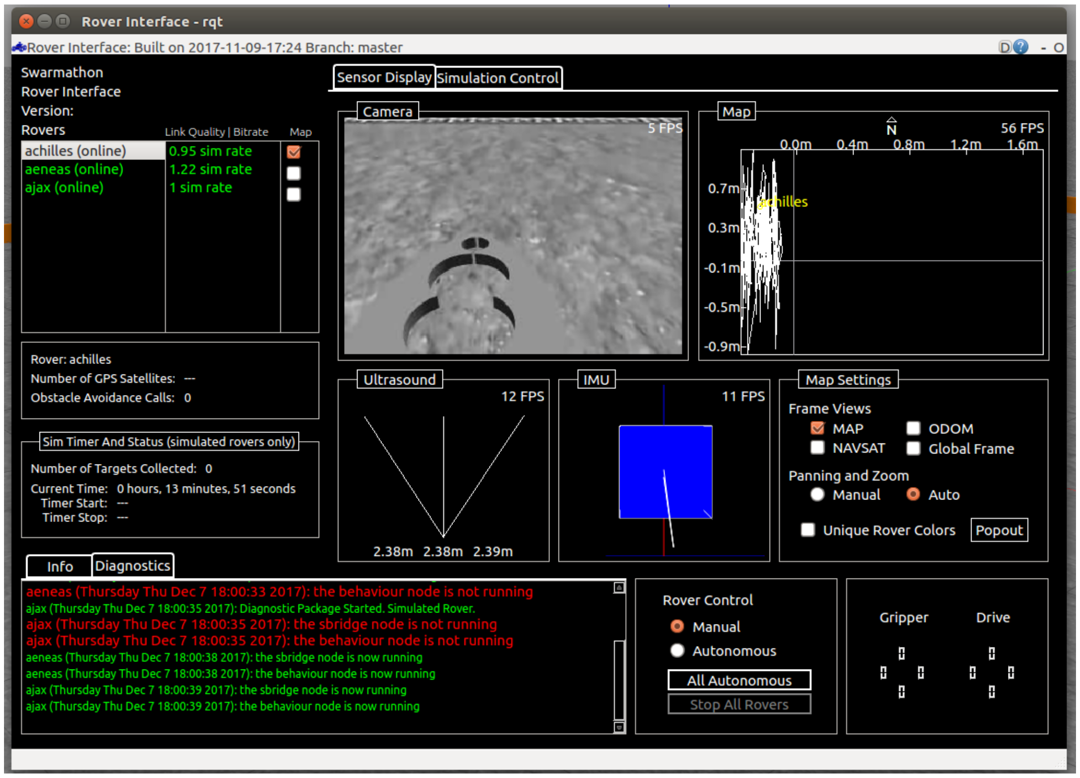
2.8 Apportioning of Requirements

Changes in the competition, such as rules or rovers, will delay future versions of the system. As this is for a competition, this future version system is bound to Swarmathon. Waiting for updates from competition organizers can cause delays in the system.

3. External Interface Requirements

3.1 User Interfaces

There are two user interfaces, Rover Interface which mainly displays Swarmie information, and Gazebo which runs a simulation. To run a Swarmie and initiate the rover interface on the master computer, the master computer must have the Ubuntu 16.04 operating system, ROS, and Gazebo installed. The user can start the Rover Interface program using Ubuntu's terminal. When the master computer is connected to a Swarmie it will display sensor and mapping information and video from the camera in the Rover Interface program. The Rover Interface program also lists the Swarmies that are connected and their hardware status. The Rover Interface program also provides the ability to switch the Swarmie between autonomous mode and manual mode. Finally the Rover Interface program allows the user to modify simulation settings and to launch a simulation of the Swarmies in the Gazebo program. The Gazebo interface displays a simulated Swarmies and arena. The Gazebo interface allows the user to change the user point of view within the simulated arena. Gazebo updates Rover Interface program during the simulation, providing sensor, mapping and other information.



Rover Interface: Built on 2017-11-09-17:24 Branch: master

Swarmathon
Rover Interface
Version:

Rovers	Link Quality | Bitrate	Map
achilles (online)	0.95 sim rate	✓
aeneas (online)	1.22 sim rate	
ajax (online)	1 sim rate	

Rover: achilles
Number of GPS Satellites: ---
Obstacle Avoidance Calls: 0

Sim Timer And Status (simulated rovers only)
Number of Targets Collected: 0
Current Time: 0 hours, 13 minutes, 51 seconds
Timer Start: ---
Timer Stop: ---

Map Settings
Frame Views: [✓ MAP] ODOM NAVSAT Global Frame
Panning and Zoom: Manual [✓ Auto]
Unique Rover Colors

Rover Control: [✓ Manual] Autonomous

```
aeneas (Thursday Thu Dec 7 18:00:33 2017): the behaviour node is not running
ajax (Thursday Thu Dec 7 18:00:35 2017): Diagnostic Package Started. Simulated Rover.
ajax (Thursday Thu Dec 7 18:00:35 2017): the sbridge node is not running
ajax (Thursday Thu Dec 7 18:00:35 2017): the behaviour node is not running
aeneas (Thursday Thu Dec 7 18:00:38 2017): the sbridge node is now running
aeneas (Thursday Thu Dec 7 18:00:38 2017): the behaviour node is now running
ajax (Thursday Thu Dec 7 18:00:39 2017): the sbridge node is now running
ajax (Thursday Thu Dec 7 18:00:39 2017): the behaviour node is now running
```

Figure 1 The Rover Interface program

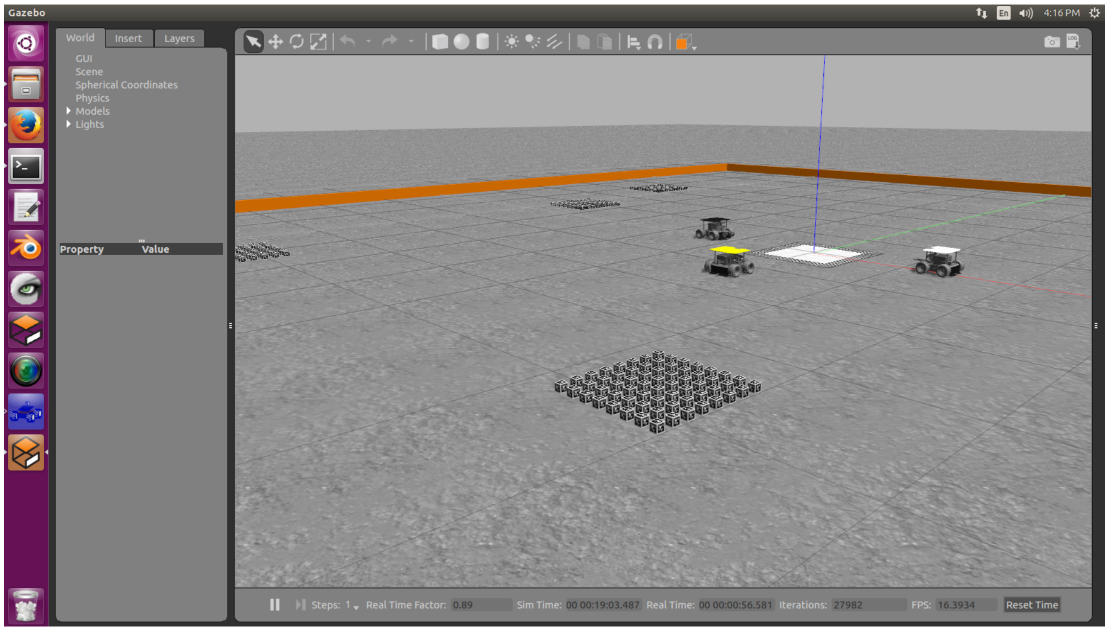


Figure 2 The Gazebo program

3.2 Hardware Interfaces

The Swarmies' code need the following hardware to run:

- Web Camera: The camera is connected to the onboard computer through usb. The camera driver sends camera data to a Swarmie's code. Camera data is used to identify when a rover sees an april tag cube.
- Microcontroller: Each Swarmie has an Arduino compatible microcontroller connected to its' onboard computer, through usb. The Swarmie's code parses and communicates information to and from the Arduino. The microcontroller obtains data from the ultrasound sensors, servo motors, and motor encoders. The microcontroller also translates commands from onboard computer to the encoders and servo motors.
- Encoders: Each Swarmie has 4 encoders, one for each wheel. All encoders are connected to the microcontroller. Each encoder moves a wheel depending on the voltage provided.
- Router: To run a Swarmie over a network, it must be connected to the same router as the master computer.
- Servo Motors: The servo motors are connected to the Swarmie's microcontroller and the Swarmie's claw. Depending on the voltage provided, one of the two servos opens or closes the Swarmie's claw. Another servo is used to move the claw up or down. The claw's main purpose is to pick up april cubes.

- IMU: The IMU is connected to the onboard computer through usb. The Swarmie's code receives gps, orientation, and acceleration data from the IMU using the IMU driver

3.3 Software Interfaces

Arduino IDE - Version 1.85. The arduino IDE is used to compile the arduino source code for the microcontroller and to debug the microcontroller. Debugging is done using the output terminal of the IDE to display data and errors that would otherwise be inaccessible.

Gazebo - Version 7.0.0. Gazebo is provided by http://gazebosim.org/. This software allows for simulation of the Swarmies for testing without the Swarmies. It is used to simulate up to six Swarmies for the competition.

ROS - version Kinetic Kame 1.12.7. ROS(Robot Operating System) is a collection of software frameworks that facilitates software development for the Swarmies

Rover Interface - Built Date 2017-11-09. Rover Interface was provided by the competition organizers. The user will be using the Rover Interface program in order to operate a Swarmie. There will be two options to work with:

- Sensor Display
  - The Sensor Display is used to output the data from a single Swarmie being tested in a physical or simulated environment. The data output to the user includes Sonar readings, orientation,GPS, odometry, and camera view.
- Simulation Control
  - The user will have the ability to simulate the Swarmies in a virtual environment as opposed to physically testing them. All that needs to be done is input values such as number of Swarmies and cubes on the field. In order to run the simulation Gazebo is used.

3.4 Communications Interfaces

The utilization of SSH allows us to remotely login to a Swarmie on the network. Remotely logging in gives access to the Swarmie so building, editing, and running code directly on the Swarmie is possible.

4. Requirements Specification

4.1 Functional Requirements

1.01 The rover hardware shall receive the motor and sensor data from another rover.

1.02 The rover hardware shall send the motor and sensor commands to other rovers.

1.03 The rover hardware shall transfer the rover's data over a network.

1.04 The rover software shall detect the Apriltags.

1.05 The rover software shall map obstacles.

1.06 The rover software shall receive sensor data.

1.07 The rover software shall send grid map data.

1.08 The rover software shall check whether the sensors are running.

1.09 The rover software shall check sensors for connection status.

1.10 The rover software shall avoid obstacles.

1.11 The rover software shall avoid pushing the april cubes out the collection zone.

1.12 The rover software shall keep track of the sensors.

1.13 The rover software shall avoid collision with another rover.

1.14 The rover software shall search for april cubes.

1.15 The rover software shall pick up april cubes.

1.16 The rover software shall drop off april cubes in the collection zone.

1.17 The rover software shall check if the rover is in the collection zone.

1.18 The rovers shall move autonomously.

4.2 External Interface Requirements

Webcam C170

- Description of purpose: The cameras main purpose is to identify april tag cubes.
  Source of input or destination of output: The camera outputs video data to the Swarmie's onboard computer.
- Valid range, accuracy and/or tolerance: Universal Clip Adjustability (range): 71mm

- Video capture: Up to 1024 x 768 pixels
  Timing: 18 fps
- Data formats: yuyv

Ublox LEA-6 GPS Dev Board

- Description of purpose: This obtains gps, orientation, and velocity data.
  Source of input or destination of output: The destination of output is the Swarmie's onboard computer.
- Valid range, accuracy and/or tolerance:
- Velocity Accuracy: 0.1 m/s
- Spherical Error Probability (SEP): <3.5 m
- Horizontal positioning accuracy: without aid: 2.5m, SBAS (Satellite Based Augmentation System): 2.0 m
- Operational limits: Dynamics: 4g, Altitude: 50,000 m, Velocity: 500 m/s Relationships to other inputs/outputs: In the Swarmie's code this data will be displayed in the user interface.

131:1 Metal Gearmotor 37Dx57L mm with 64 CPR Encoder (No End Cap)

- Description of purpose: Motors for wheel movement, and sending wheel rotations to Swarmie computer.
- Source of input or destination of output: The destination of output is the Swarmie's onboard computer.
- Valid range, accuracy and/or tolerance: 80 RPM and 300mA free-run, 250 oz-in (18 kg-cm) and 5 A stall
- Units of measure: Centimeters
- Relationships to other inputs/outputs: In the Swarmie's code this data will be displayed in the user interface.

4.3 Logical Database Requirements

All data will be sent through ROS messages nearly at all times. The data shall be retained for the duration of the ROS core program run time.

4.4 Design Constraints

The Swarmies during the competition are not the same as the Swarmies being developed. There will be differences in the development Swarmies and the competition Swarmies. Development

Swarmies will be tested and worked on in California while the competition Swarmies will be in Florida. The different climates will affect design and implementation.

Our software programming liberties are limited in that we cannot tamper with certain portions of the given code. The ROS frameworks, the Rover Interface program, or how the Swarmie communicates with the interface or other software can not be altered. Packages that have not been whitelisted by the competition organizers cannot be used.

In terms of hardware, we are not allowed to alter the electronic components on the Swarmies. We are not allowed to alter any of these by competition rules and due to the fact competition Swarmies and development Swarmies are different.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Swarmies' performance needs to satisfy the following requirements:

The user shall connect the Swarmies in less than 1 second.

The user shall control the Swarmies in less than 1 second.

The user shall receive the motor and sensor data in less than 1 second.

All Swarmies shall connect in less than 1 second.

All Swarmies shall communicate in less than 1 second.

All Swarmies shall process motor and sensor data in less than 1 second.

All Swarmies shall receive motor and sensor data in less than 1 second.

5.2 Safety Requirements

The Swarmies use a lithium-ion battery that can cause potential fire hazard. All batteries attached to the Swarmies must be in supervision in case of potential fire hazard when charging the batteries. We have a 13.4 Ah, 14.4 V battery that we could charge at 0.1 Ah to 6 Ah with the battery charger. For safety concerns and longevity of the battery, we charge the batteries inside a fire retardant at 3 Ah, fully charging a completely discharged battery in 4.5 hours.

The Swarmies can also cause damages if they flip over when moving too fast during testing. Therefore for safety concerns, the maximum speed for the Swarmies will be set to 1 m/s. Due to there being no sensors attached on the back of the Swarmies, Swarmies can collide with a wall when driving backwards and can cause extensive damages. Driving backwards without any sensors to detect walls is driving blind. Collision can be prevented by manually stopping the Swarmies from moving autonomously or by human intervention when they are close to colliding.

5.3 Security Requirements

The user must connect to Swarmies over some network. We recommend a private wireless network protected using WEP security and without internet access. This minimizes security exposures.

5.4 Software Quality Attributes

This document does not assure quality as it is for a competition and will be worked up until the submission deadline. After the submission deadline, no work on this software is promised until California State University, Los Angeles reconfirmed as participants for a future Swarmathon competition. If reconfirmed as participants for a future Swarmathon competition, continuation of this software is not guaranteed as the rules, base code and software requirements may change.

5.5 Business Rules

This software has is intended for use only by the competition organizers. The organizers shall use this software to operate Swarmies during the competition.

6. Other Requirements

Swarmathon is robotics competition. The goal of competition is to place first. This software's purpose aside from operating an autonomous rover is to win Swarmathon. As participants of Swarmathon, members of this team are required to volunteer in an outreach project, sharing and exposing robotics and the STEM fields to K-12 schools.

Appendix A: Glossary

Definitions

- April tag cube: A cm by cm cube. The Swarmies' main objective is to collect as many of these cubes as possible within an arena.
- Collection Zone: The area in the competition arena where the Swarmies start and where they drop off the april tag cubes that they collect.
- Competition Swarmie: Rovers used in competition with the rover software from the development Swarmie
- Development Swarmie: Rovers used for developing the rover software
- Gazebo: Simulation Software for robots.
- Master Computer: the computer that runs and connects all Swarmies.
- Swarmathon: The NASA Robotics Competition.
- Swarmie: Rover used for the competitions.
- Rover Hardware: A rover and its physical components, including its sensors and motors.
- Rover Software: The software being developed for the Swarmathon competition.
- Rover Interface: A user interface that displays Swarmie information and allows user to create a simulation.
- Swarmathon: The NASA Robotics Competition.

Acronyms

- IMU (Inertial Measurement Unit)
- NASA (National Aeronautics and Space Administration)
- ROS (Robotic Operating System)
- SSH (Secure Shell)

Abbreviations

- April cube: April tag cube

Appendix B: Analysis Models

TBD

Appendix C: To Be Determined List

The To Be Determined List is TBD