

McMASTER UNIVERSITY

SMARTSERVE

SOFTWARE & MECHATRONICS CAPSTONE

Verification and Validation

Authors:

Victor Velechovsky - 001305263

Professor:

Dr. Alan Wassyng

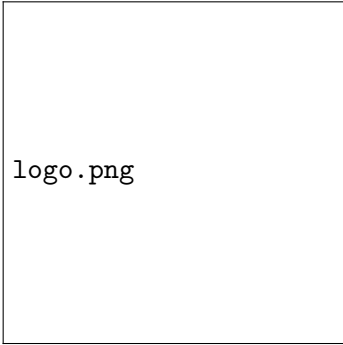
Teaching Assistants:

Bennett Mackenzie

Nicholas Annable

Stephen Wynn-Williams

Viktor Smirnov



logo.png

Last compiled on February 17, 2019

Contents

List of Figures

List of Tables

Date	Revision	Comments	Author(s)
Feb 1, 2018	1.0	Document structure and Headings	Christopher McDonald
Feb 10, 2018	1.1	Philosophy and Intro section	Christopher McDonald
Feb 13, 2018	1.2	UI and DS Tests	Sharon Platkin
Feb 15, 2018	1.3	SM, SMC, SS, CV and SR	Christopher McDonald & Harit Patel & Sam Hamel
Feb 16-17, 2018	1.4	Executive Summary and Testing	Christopher McDonald & Sharon Platkin
Apr 6, 2018	2.0	Added integration tests, fixed grammar and added test instructions	Christopher McDonald
Apr 10, 2018	2.0	Performed tests and wrote executive summary	Christopher McDonald & Sharon Platkin & Sam Hamel

Figure 1: Revision History

1 Executive Summary of Testing

This testing phase completed on April 10th, 2018 and has yielded 56 passes and 7 failures. The failures occurred predominately in the Computer Vision and Shooting categories. The Computer Vision tests did not reach the desired accuracy levels and the shooting mechanism did not hit the desired zones during some of the tests.

The team is confident in the coverage of the functional requirements based on the traceability matrices shown in Tables ?? and ?. In these tables, every requirement has at least one test case which covers it. This is less true for non-functional requirements since 3 of the 18 have no test cases as shown in Tables ?? and ?. These requirements were too difficult to test in an objective way and are thus omitted from this document.

These results accurately represent the priorities of the team to make the system more reliable, responsive and accurate. The team will continue developing the system until April 26th, after which the final presentation will take place on the 27th.

2 Introduction

2.1 Project Overview

is a swarm of autonomous robotic boats meant to carry out measurements over large bodies of water. For our project, these boats will be measuring water temperature, but the idea can be expanded to any number of other quantifiable measurements. Central to our work will be two major components. First, a small motorized boat, attached with a water temperature sensor, as well as a control unit that allows it to communicate with – and be controlled by – a centralized control unit. Second, a software package that can control a large group (swarm) of these boats, with an algorithm focused on producing reliable, accurate, and fast measurements.

Our swarm will aim to cover large areas more quickly and cost-effectively than traditional products. To test the applicability of our project, we will demo it on a small scale body of water, such as a swimming pool, as well as develop a simulation to hypothetically prove the efficacy of the system on a larger scale.

Our project will be conducted between Fall 2018 – Winter 2019 for our Engineering Capstone project at McMaster University, under the guidance of Dr. Alan Wassing. We have four Software Engineering students, and one Mechatronics Engineering student.

2.2 Document Overview

This document will provide details of all formal testing methods and results performed on the SmartServe system. The first part of testing includes detailing how it will be performed and the details of the system on which it is run. This matters due to details which can affect the testing outcomes like operating system, lighting or performance of hardware. The schedule for the test will also be detailed alongside the major deliverables to have clear outcomes to explain to stakeholders. The testing will be performed off of the *master* branch as it stands during the beginning of the testing phase.

The actual testing will then be detailed as test cases based on what subsystem they are testing. As needed, the communication will be tested in between the subsystems to ensure communication is working as intended. Lastly, a test case-requirement matrix will be provided which maps what test cases test which requirement. This will give the reader a simple way to check if a requirement is fully satisfied. Supporting documents include the requirements which can be found [here](#).

2.3 Naming Conventions and Terminology

- **System:** The entire software and hardware package - including the boats, boat hardware, control software, and server running the control software
- **Swarm:** A large group of objects (in our case, motorized boats) that can communicate and perform acts as a group
- **Insect:** A member of the swarm (in our case, a single motorized boat)
- **Simulation:** The simulation will be used for demo purposes, mainly to show that the system is valid with a large number of insects.
- **Researcher:** A user that is interested in the data that is returned from the survey.
- **System Administrator:** A user that controls the parameters of the **swarm**.
- **G.P.S.:** Global Positioning System

3 Testing Philosophy

3.1 Approach

Unit tests for Python modules will be written in PyUnit. All hardware units will be tested manually, according to the guidelines laid out in this document.

Tests will be performed upon completion of this document, and before the final presentation of April, 2019. All results will be included in the 'test cases' section of the document.

Each hardware test will be run three times, if possible, for a greater sense of confidence in the reliability of the product, though of course in a professional environment, more thorough testing would be required.

3.2 Schedule

Task	Date	Notes
Write Test Cases	February 17, 2019	N/A
Run Initial Test Plan	February 18, 2019	NA
Fix Issues Discovered in Testing	February 18-19, 2019	NA
Perform Second Round of Tests and Publish Final Results	February 20, 2019	NA

Table 1: Testing Schedule

3.3 Environment

In order to support OpenCV object detection, the server will run on a high end laptop (2018 15" Macbook pro) with Mac OS Mojave (10.14.3) and a quad-core Intel Core i7.

The object detection might require a certain environment to be used, in order to remove problematic images. Our Revision 0 testing was performed on the floor in Thode Library, though for our final presentation we will likely chose a different location, such as one of the McMaster Basketball courts.

3.4 Setup Instructions

- Assemble insects
- Attach a colored spherical identifier to each insect, each with a clearly distinct color (for our testing, we used 'Red' 'Green' and 'Blue' balls)
- Upload the insect code to each insect arduino
- Upload the master code to the master arduino
- Check out the server code
- Install OpenCV 3
- Install all python dependencies
- Turn on every arduino unit

- Place the insects in their starting locations
- Configure Parameters.py to match your setup (especially the SERIAL and COLOR-RANGE parameters)
- Start the server code with *pythonMain.py*

4 Test Cases

4.1 Area Coverage Algorithm

The area coverage algorithm is written in Python, so pyTest is used to write automated test cases, wherever applicable.

Test ID: ALG1	Run Performance 1	Status: PASS
Description: Simulation runtime performance with a 10x10 grid		
Pass/Fail Condition: Algorithm should run in simulation mode, on the test Macbook, with a 100 x 100 grid, in under 10 seconds.		
Input: 100x100 grid with 3 robots, started in random locations		
Expected Results: Algorithm terminates within 10 seconds	Actual Results: Algorithm terminated within 10 seconds	

Table 2: User Interface Sign up Test

Test ID: ALG2	Area Coverage 1	Status: PASS
Description: Algorithm covers every location in the coverage area		
Pass/Fail Condition: Every region in a 100x100 coverage area is covered		
Input: 100x100 grid with 3 insects, started in random location		
Expected Results: Every region is covered by the end of the simulation	Actual Results: Every region was covered by the end of the simulation	

Table 3: User Interface Sign up Test

Test ID: ALG3	Start Locations 1	Status: PASS
Description: Algorithm works from any starting points		
Pass/Fail Condition: Algorithm covers area with randomly generated starting points		
Input: 10x10 grid with (random) 1-5 robots placed at random locations		
Expected Results: Every region is covered by the end of the simulation		Actual Results: Every region was covered by the end of the simulation

Table 4: User Interface Sign up Test

Test ID: ALG4	Arbitrary Number of Insects 1	Status: PASS
Description: Algorithm works with arbitrary insects		
Pass/Fail Condition:		
Input: 10x10 grid with (random) 1-50 robots placed in the bottom left corner of the region		
Expected Results: Every region is covered by the end of the simulation		Actual Results: Every region was covered by the end of the simulation

Table 5: User Interface Sign up Test

Test ID: ALG5	Arbitrary Region Shapes	Status: FAIL
Description: Algorithm works with arbitrarily shaped coverage areas		
Pass/Fail Condition:		
Input: Rectangle region, circular region, and octahedral region		
Expected Results: Every region is covered by the end of the simulation		Actual Results: We don't currently support non-rectangular coverage areas

Table 6: User Interface Sign up Test

5 Test Case-Requirement Traceability Matrix

Table 7: Matrix to Match Tests to Functional Requirements [1]

Functional Requirement-Test Matrix																		
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18
SMC1	X	X	X	X	X													X
SMC2	X	X			X													X
SMC3	X			X														
SMC4	X																	
SMC5	X		X															
SMC6																		
CV1					X													X
CV2					X													
CV3					X													
CV4					X													X
CV5					X													X
CV6						X												X
CV7					X													
CV8					X													
CV9					X													
CV10						X												
CV11					X													
CV12					X													
SR1														X	X			
SR2							X							X	X			
SR3																		
SR4														X				
SR5														X				
SM1	X		X		X													
SM2	X	X			X													X
SM3																		
DS1								X										
DS2	X	X	X	X	X	X												X
DS3						X												
DS4									X									
DS5					X	X	X	X					X					

Table 8: Matrix to Match Tests to Functional Requirements [2]

Functional Requirement-Test Matrix																		
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18
UI1								X	X	X	X	X	X			X	X	
UI2								X	X	X	X	X	X			X	X	
UI3																X		
UI4								X										
UI5								X	X									
UI6					X	X	X						X	X				
SS1							X							X	X			
SS2							X							X	X			
SS3																		
SS4														X				
SS5															X			
SS6														X				
SS7														X				
SS8					X	X												X
SS9					X	X												X
SS10					X	X												X
SS11					X	X												X
SS12						X		X	X				X					X
SS13						X		X	X				X					X
SS14								X										
SS15						X												
SS16									X									
SS17	X	X	X	X	X													X
SS18	X	X	X	X	X													X
SS19	X		X															
SS20	X		X															
SS21	X	X			X													X
SS22	X	X			X													X
HI1	X	X	X	X											X			
HI2	X	X	X	X											X			
HI3	X	X	X	X											X			
HI4	X	X	X	X											X			

Table 9: Matrix to Match Tests to Non-Functional Requirements [1]

Non-Functional Requirement-Test Matrix																		
	LF1	UH1	UH2	P1	P2	P4	P5	OE2	MS2	S1	S2	P1	LC1	HS1	HS2	HS3	HS4	HS5
SMC1														X	X			
SMC2														X	X			
SMC3														X				
SMC4														X				
SMC5														X				
SMC6																		X
CV1								X										
CV2																		
CV3																		
CV4																		
CV5																		
CV6																		
CV7																		
CV8																		
CV9																		
CV10																		
CV11																		
CV12																		
SR1														X	X			
SR2														X				
SR3														X				
SR4																		
SR5																		
SM1														X				
SM2														X	X			
SM3														X				
DS1						X				X								
DS2														X	X			
DS3					X													
DS4																		
DS5									X		X							

Table 10: Matrix to Match Tests to Non-Functional Requirements [2]

Non-Functional Requirement-Test Matrix																		
	LF1	UH1	UH2	P1	P2	P4	P5	OE2	MS2	S1	S2	P1	LC1	HS1	HS2	HS3	HS4	HS5
UI1	X	X	X		X	X	X		X	X		X						
UI2	X	X	X	X														
UI3			X	X														
UI4			X			X				X								
UI5	X	X	X	X														
UI6	X	X	X	X					X									
SS1															X			
SS2															X			
SS3															X			
SS4															X			
SS5															X			
SS6																		
SS7																		
SS8																		
SS9																		
SS10																		
SS11																		
SS12					X	X				X								
SS13					X	X				X				X	X			
SS14						X				X				X	X			
SS15					X													
SS16																		
SS17														X	X			
SS18														X	X			
SS19														X				
SS20														X				
SS21														X	X			
SS22														X	X			
HI1														X	X			
HI2														X	X			
HI3														X	X			
HI4														X	X			

6 Appendix

Shot ID	Zone ID	Roll (degrees, x-axis)	Pitch (degrees, y-axis)
48	17	0	20
91	12	90	10
148	5	180	10
193	2	270	0

Figure 2: Shot details for tested shots