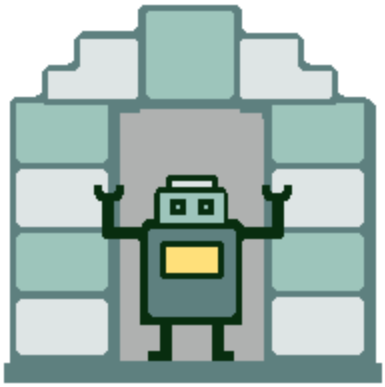
Software Testing Plan

4/5/2019

Version 1.0



Keystone Robotics
Robot Assisted Tours

Project Sponsor:

Michael Leverington

Faculty Mentor:

Austin Sanders

Team Members:

Hailey Ginther, Shannon Washburn, Gabrielle Halopka, Falon Ortega

Table of Contents

1.0 - Introduction

Northern Arizona University's Engineering building is the most important stop of campus tours for future engineering and computer science students. The labs, project rooms, and lecture halls of the building are where they will be studying for the next 4 or more years. This means the impression groups get of the facilities during their time on a tour of the building is an essential contributing factor to attracting and retaining new students. Attracting the attention of these students with physical evidence of the work accomplished by seniors in the department will help persuade these new individuals to enroll.

Our client, Dr. Michael Leverington, is a professor at Northern Arizona University's School of Informatics and Computing. The professor has a Ph.D. in Education, Masters of Computer Science and Psychology, and a Bachelors in Physics. Additionally, he is an avid follower of developments in robotics and is eager to bring the multidisciplinary topic to his department. He has tasked our team with the initial planning, assembly, and programming of a robot potentially capable of giving tours of NAU's engineering.

The ultimate purpose of this project is to create a robot capable of autonomously giving tours of the engineering building, fulfilling the need for a captivating introduction to the projects NAU students can accomplish thanks to or in conjunction with their coursework. Our goal as the starting force behind this project is to put what we have identified as the basic level of hardware and software this robot needs into practice. For the scope of this capstone project, the client's goal is to have a fully assembled robot that can be moved with user inputs. Other major goals during this project for this team are to perform tests that ensure the safety and stability of hardware and to leave behind detailed documentation for future students' use during follow-up projects.

Testing of this project begins on the smallest scale with the unit tests described in the next section, followed by a section on the integration tests between our units and other project components. This is followed by a section on verifying the useability of our project. See our conclusion following the section on useability testing for a summary of our methods and results.

2.0 - Unit Testing

Unit tests help ensure that the base components of our project function as intended, especially for edge case and unexpected input. The goal of these tests is to ensure that methods written by our team for the Arduino and Robot Operating System will operate as expected. Output from these methods being expected values is essential to ensure integration testing and other tests further into development are not working from inherently flawed base code, which in the case of our project could result in improper robot functions or hardware misuse.

For our project, the software units being tested are sorted into two categories: Arduino code and ROS package code.

2.1 - Arduino Code

The Arduino code of this project is a short but essential program that receives directional input from a ROS package that it uses to control the robot's two motors.
In order to test the code within an Arduino efficiently we have recreated the program in the closely-related language C and run tests on this analogous code. This method of testing ensures any potential errors are from the code rather than the hardware it is to run on. There is a single method, the test of which is described below:

Methods set_right_motor() and set_left_motor() are essentially the same methods using different pinouts. The equivalence partitions are negative integer values from -255 to -1, positive integer values from 1 to 255, and 0. The boundary values are -255, -1, 0, 1, and 255. Currently, we have incorrect values of -256, 256, 100.1, 'd', and "string" to ensure robustness of the method.

2.2 - ROS Packages

A small ROS package written for this project controls the speed for our robot given a velocity vector input from a USB joystick. The speed of our robot must be moderated carefully to avoid physical damage to components and surroundings, so the testing of this unit is especially essential. There is just one method within the C++ code of this program, the test of which is described below:

Method speedCallback(). The equivalence partitions are float values between -.5 and .5 and the boundary values are -.5 and .5. These values are being passed via a ROS message

from the teleop_twist_joy node which is part of a built-in ROS package, therefore we feel confident that the message will contain only valid values.

3.0 - Integration Testing

Once our software is tested and our hardware components are connected and ready, integration testing possible. Some of our electronic components rely on the presence of other components to allow for any useful tests to be performed. Therefore, we must assemble these main components circuit together completely to enact an initial integration test. Then we split further integration tests down into a set of steps where integration between all current parts are confirmed to behave as expected before addition of more parts, essentially proceeding through integration testing in a bottom-up approach.

For each module listed, assume the parts and steps of testing listed in any previous models are applied in addition to the parts introduced for that module.

3.1 - Module 1: Base Circuit

Tests the basic circuit needed to power all additions to integration testing process.

Components tested: Batteries, Motor Drivers, Wiring, Power Switch
Method(s) of Integration Verification:

1. No unwanted feedback occurs when the circuit is powered on.
2. All components remain within a temperature that is safe for direct human contact.

3.2 - Module 2: Wheel & Motors, Base

Tests the integration of the basic power circuit with the wheels and motors, plus the stability of the motors once they are attached to the base.

Components tested: Wheels, Motors
Method(s) of Integration Verification:

1. The motors & wheels spin when the circuit is powered on.
2. The motors exhibit no visible shaking loose or wobbling in the base frame when bolted fully onto mount.

3.3 - Module 3: Arduino

Tests the integration of the Arduino & its respective code with the hardware of the previous module.

Components tested: Arduino Uno
Method(s) of Integration Verification:

1. A basic shell code uploaded to the Arduino causes the motors to start spinning once powered.
2. Powering off the Arduino or overwriting its current file halts all motor movement.

3.4 - Module 4: Raspberry Pi & USB Controller

The Raspberry Pi is added to the circuit and is tested for integration with the previous parts, then the integration between the ROS code and the circuit is tested.

Components tested: Raspberry Pi 3, USB controller, Laptop interface
Method(s) of Integration Verification:

1. Raspberry Pi connects to Arduino and powers Arduino.
2. Laptop successfully creates SSH connection to Raspberry Pi via Ethernet cable.
3. Relevant ROS packages successfully loaded and launched from laptop.
4. USB controller input successful transmission exhibited by wheel movement corresponding to direction and speed of joystick press.

3.5 - Module 5: Full Hardware Integration

The circuit and software of previous module is installed inside the barrel housing, secured and insulated as needed.

Components tested: Barrel & mount (overall stability), insulation
Method(s) of Integration Verification:

1. All previous integration steps confirmed to remain successful after installation.
2. Wiring does not shake loose during robot movement.
3. Components do not fall from their respective shelves during robot movement.
4. Ending movement input from USB controller halts all robot movement.

4.0 - Useability Testing

Testing the useability of our project with a group of users is important to make sure our end product will be useful to our client after our team has completed it. The end users identified during the planning of this project are students and faculty of NAU's engineering building. This userbase is expected to possess a level of understanding of basic programming and computer skills needed to power on electronics and perform command line processes with minimal instruction. Any level of familiarity with the particulars of the robot's inner mechanisms is not expected of most of our users. Because our users are expected to be somewhat knowledgeable already, we are holding tests that demand a certain level of experience.

The tests identified in the following subsections are to be performed by a group of NAU engineering students and faculty asked to participate randomly from around the Engineering building (After confirmation and recording of their age, year in school, and major. Non-engineering majors and faculty will not be included in testing for safety purposes).

4.1 - Setup Test

Purpose: To confirm that our client and future students using our project components can adhere to team safety guidelines given a manual of basic instructions and warnings.

Setup:
User is shown where the power switch is located and provided a brief manual listing
(a) Components that need to be plugged in before operations can be started
(b) Wiring and hardware that should be inspected before each power-on
(c) Commands that must be passed at command line to begin operations

The test is considered successful if the user:

- Connects the laptop and game controller to the Raspberry Pi via Ethernet and USB respectively
- Checks that all 4 fuses are present and undamaged in circuit
- Confirms that the Raspberry Pi and Arduino are receiving power from respective battery pack, and the battery packs for the relays are switched on
- Turns on power switch only after all previous tasks have been completed for this test
- Correctly passes given command line instructions to start robot operation

After the test is completed the user will be asked to rate on a scale of 1-10, 1 being simple and 10 being impossible, the difficulty of:
(a) Checking the hardware for faults
(b) Connecting the components
(c) Powering the robot on

4.2 - Operation Test

Purpose: To confirm our client and future users of our project understand how to manually control the robot and read our user manual.

Setup:
User completes test described in 4.1, and is directed to section 2 of user manual containing:
(a) Warning on the speed and danger presented by misuse of user handling of robot
(b) Instruction on how to move robot with joystick and button input

The test is considered successful if the user:

- Reads the section of the manual provided
- Is able to control and move the robot without striking nearby people or obstacles with robot

After the test is completed the user will be asked to rate on a scale of 1-10, 1 being simple and 10 being impossible, the difficulty of:
(a) Reading and understanding the manual explanation
(b) Controlling the robot manually with a USB controller

4.3 - Power-off and charge test

Purpose: To make sure clients and other future users understand the power-down procedure and how to charge the robot.

Setup:
User completes previous tests and is directed to charging information portion of manual which contains:
(a) Warning about improper handling of charging circuit
(b) Instructions on switching circuit to charging from power

The test is considered successful if the user:

- Successfully switches robot to charging mode

- Connects charger to robot and outlet, with charger displaying correctly colored “charging” LED indicator

After the test is completed the user will be asked to rate on a scale of 1-10, 1 being simple and 10 being impossible, the difficulty of:
(a) Switching the circuit from power to charging

5.0 - Conclusion

Tours of the Engineering building are time-consuming and largely uninformative. Professors must take time out of their day to assist with these tours, and the students who give them are often not engineering students. Our client, a computer science professor with a physics and engineering background, wants a device that can take over this role. By automating this process with a robot-assisted tour, faculty can save time, and visitors to the building can be shown the capability of NAU's engineering and computer science students.

Testing is essential at all levels of our robotics project to make sure components work safety and correctly. Unit tests of our programs are simple but important - the testing of the Arduino code significantly lessens the chance of bad input from ROS making the motors to spin uncontrollably and cause accidental collisions or other issues. Similarly, the unit test for the ROS package will help avoid speeds outside of our controlled range that could cause the robot to move much faster than intended. Integration testing at each level of our project helps keep our project modular and lets us identify any problems found through testing by narrowing down potential errors to one or two parts at a time. Finally, our user testing is formatted in a way that resembles actual use of the robot as closely as possible to ensure the results are accurate as possible.

Overall, our team is extremely confident in our work and we believe it will hold up to each of these tests with minimal error. The dozens of hours of work assembling this project and the research that prefaced it have given us the experience we need to develop the set of comprehensive tests presented in this document.