Software Testing Plan

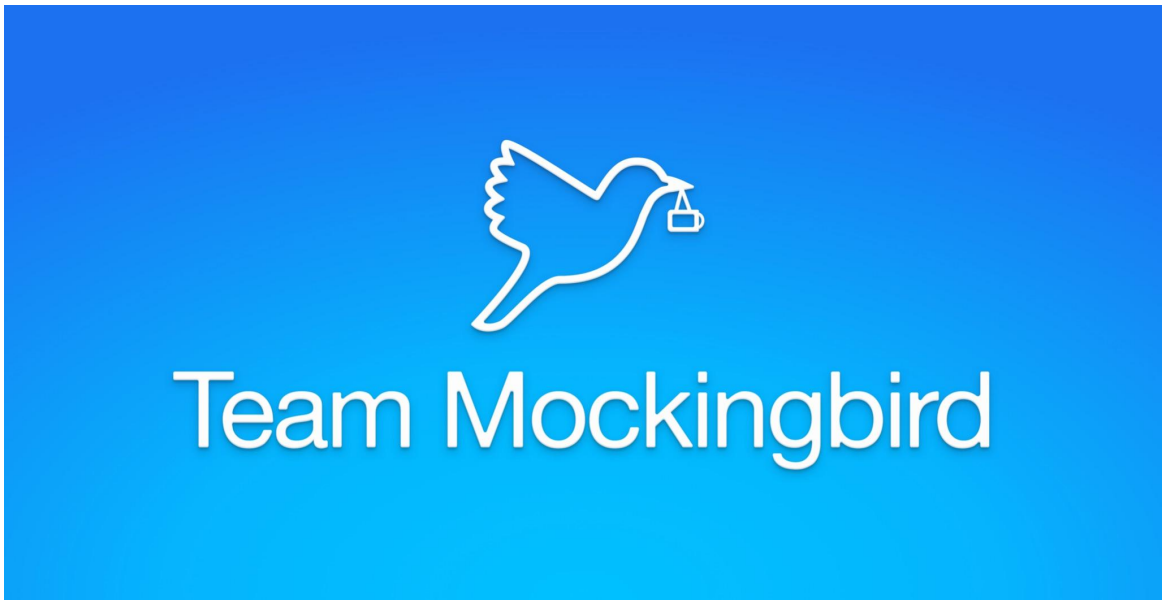
Nov 11, 2022

Team Sponsor: **Alexander (Allie) Shenkin**

Team Mentor: **Daniel Kramer**

Team Instructor: **Michael Leverington**

Document produced by



Austin Malmin, **Conrad Murphy**, **Charles Saluski, and ShanHong “Kyle” Mo**

Version 1.0

Introduction

In the world of automation, normal mundane tasks are being replaced by artificial intelligence that can perform these jobs at higher efficiency and with better accuracy. The world is starting to see a breakthrough in the field of robots and drones that perform tasks autonomously. Today, humans are currently performing tasks that can easily be automated with self-guided drones. Search and rescue missions can prove to be dangerous in places not easily navigated by humans, such as collapsed caves, house fires, war zones, etc., but a drone could make these missions safer and more efficient. Amazon has already taken use of autonomous drones in their delivery of goods throughout the United States and the United Kingdom. Drones can dust crops with greater precision and cheaper cost than planes piloted by humans. Our focus is the use of drones to aid in the study of rainforest ecosystems.

Mapping the upper and lower forest canopy currently requires a researcher to manually move a ground-based sensor through the forest which is time-consuming and difficult. The sensors used are Light Detection and Ranging (LiDAR). LiDAR sensors measure the time for a laser beam to return once fired to determine the distance of the object from the sensor and create a 3D model of the environment, whether on land or on the seas. Ecologists perform research focused on how forests around the world respond to climate change and simulate the effects certain changes in the climate would have on the environment. Team Mockingbird is working with Dr. Allie Shenkin to build an autonomous drone navigation system.

Our vision for Project Glasswing is a system with the end goal of guiding a drone through the rough forest environment quickly. We will not work with a drone ourselves, but future work may be done to mount the system to a drone. The system will use a LiDAR sensor for mapping its surroundings and saving it to permanent storage for future study, while other sensors will serve the role of object detection and avoidance to avoid crashing into branches and vines. A successful project will drastically improve Dr. Shenkin’s research on the effects of climate change on our rainforests.

Unit Testing

Unit testing is the idea of testing each individual component separate from the system to ensure that each component in a system works and just needs integration testing with other components, more on that to come later. Our System has 3 different categories of “Units” to test which are hardware, navigation algorithms and ROS.

Hardware Testing

When testing our hardware, it is important to acknowledge that this will be a never ending process. Over time hardware will depreciate and eventually need to be replaced even with no fault of the user. In our hardware testing, we will be focused on measuring performance of the different components, not necessarily if the hardware is functional or not.

Raspberry Pi

Figure 1: Raspberry Pi

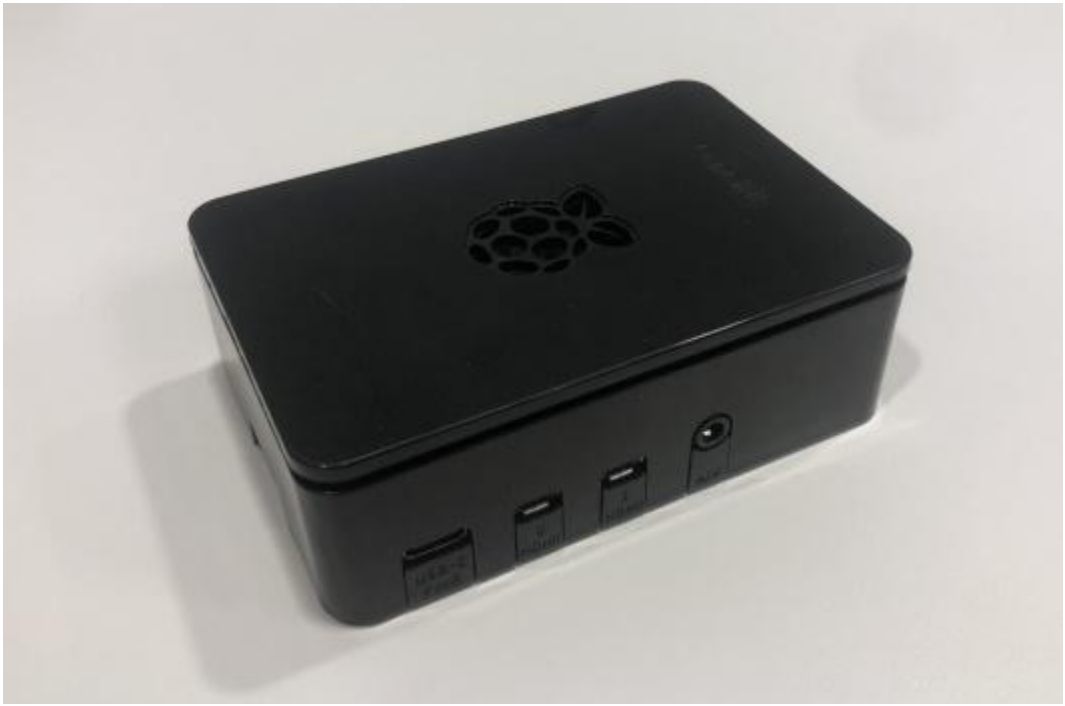


Figure 1: shows an image of the raspberry pi used for this project.

In order to test our Raspberry Pi there are a couple metrics we can measure: computing power and memory speed. We can test the Raspberry Pi's computing power by running various pieces of our code with the *htop* command active and monitoring the CPU utilization. The *htop* command displays all the cores of the raspberry pi and each cores utilization. This will accurately determine what the Raspberry Pi is capable of handling before it throttles speeds down and is unable to compute anymore. The outcome for this test will be a deepened understanding of the actual computing power of our Raspberry Pi. Similarly, we will also need to test the memory write speed of our SD card nested inside the Pi. This can be done with the IOPing package and this package will test the memory speed and display its performance. Utilizing this package will provide us an exact understanding of how much data can be written to the SD card and how quickly.

LiDAR Sensor

Figure 2: RPLiDAR A1

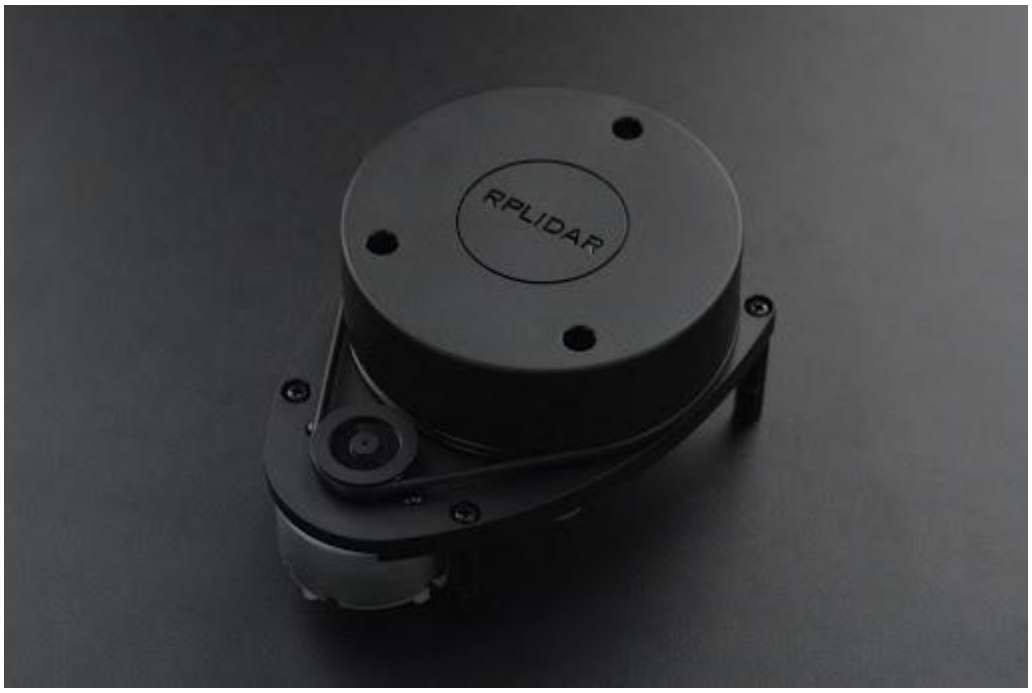


Figure 2: shows an image of the LiDAR device used for this project

The RPLiDAR is a brand of LiDAR sensor that registers objects around it in a two dimensional space. This piece of hardware is unique as we want to measure its efficiency at measuring size of objects and at different distances. Our team will test the device by having different sized objects varying from size of 1 centimeter in diameter to objects of 1 meter in diameter. All of these objects inside the test suite will be measured at distances ranging from 0.5 meters up to 15 meters. We will start with objects 0.5 meters away from the LiDAR and move back 1.5 meters at a time until the object is no longer visible to the device.

Navigation Module

The navigation module takes the map generated by the SLAM process and the robot's current calculated position and a desired end position and calculates a movement plan for the robot. We will test this module by constructing simple scenes in Gazebo, a simulation environment for ROS, and checking that the navigation system is able to navigate through the scene in an expected manner. We will also test that when a goal is given that does not have a reasonable solution, such as being outside of a closed room, that the robot does not erroneously attempt to navigate through the wall.
We will also validate the movement commands that the navigation module generates for the robot, by having a set of expected paths that the robot might take, and validating that the movement commands that the module sends to the robot are within an expected margin of these baseline paths.

ROS Testing

ROS is a framework which supports drivers for sensor hardware and open-source libraries for mapping tools, all of which can be easily run on one central platform. Our hardware and sensor libraries all directly connect with ROS in some way through talker and listener nodes, so we will be testing whether these nodes work as intended. We will also test ROS's compiler, as well as the individual ROS mapping libraries we use, which are RPLiDAR and Hector SLAM.

ROS Talker and Listener Nodes

ROS uses talker and listener nodes to interact with other components and their data. Since these nodes are what hold our entire system together, it is crucial that they can send and receive data properly. Fortunately, ROS provides debug output in the terminal which shows whether communication with other components succeeded or failed. Therefore, we simply need to run each component and see that there are no errors output by ROS in the terminal. This means running the LiDAR sensor and launching the RPLiDAR and HectorSLAM programs in isolation. The tests will succeed if ROS is able to communicate and interact with them without indicating failure.

Compiler

ROS has a built-in compiler which builds executable code from the source libraries in the *src* folder. The compiler is invoked using the command *catkin_make.* To ensure that the compiler works properly, we will clean out our previously built executables, then run *catkin_make* to compile the software again. If the compiler works, we expect that all code will be compiled smoothly without any errors. After the code is compiled, we must run a *source* command to ensure that the code is runnable in ROS. At this point, we can test if the compiled code functions correctly by running our individual ROS libraries.

RPLiDAR Library

The RPLiDAR library is provided in the ROS framework. This library interprets the point cloud data from an RPLiDAR sensor and outputs the LiDAR data on a visual display program built into ROS called RViz. This display updates in real time, so that the detected objects can be seen in real time. To test that this component runs without errors, we simply launch the library through ROS's command line interface and ensure that RViz opens up as required. The program does not open if a LiDAR is not detected in the computer's USB port, so our tests can also ensure that ROS interfaces with the LiDAR as required.

HectorSLAM Library

Our system uses a library provided in the ROS framework called HectorSLAM. This library performs SLAM (Simultaneous Localization and Mapping), a process in which a robot moves while tracking its position in the environment and maps the surrounding space in real time. HectorSLAM takes LiDAR data as input and produces a map on RViz. To test that this component runs without errors, we simply launch the library through ROS's interface and ensure that RViz opens up as required.

Integration Testing

Due to the nature of the team's capstone project, we focus on integrating hardware with each other and with existing open source software. Due to this, a lot of testing requires understanding of the purpose of each hardware and how they integrate with others.

Hardware Testing

For starters, all machinery requires some form of power in order to function, so to make sure of this, the team will need to test if the Raspberry Pi gets enough power to itself, which is easily tested by checking if the cord that the supplier brought to us works as intended. After that, we need to test if the Pi passes enough power to its attachments such as cameras and LiDAR. For this part of the project, we not only need to test if the cord the team is using can provide power, it also needs to check if data can pass through it as well. Throughout the duration of this project, the team tested many different hardwares, excluding the Pi which is the step we already covered, one of which being a 2D LiDAR and different cameras such as the Intel RealSense camera. Due to the nature of the price of LiDAR and the limited budgets, we tested backpack hardware such as cord, only to realize that many cords we possess only pass power and data, thus the cord that was provided is difficult to replace if damaged or lost. We also had to test the memory card, which was simple and difficult as testing it just requires the team to put it in, but unfortunately there are fakes that are difficult to figure out the problem without prior knowledge, due to that the team is required to purchase a new one.

Hardware with Software Testing

After confirming that all the cord can pass power and allows data passage, basically after checking that all hardware is working as intended, the next step is check how software integrates with each other. In order for our project to succeed, the team needs to find a working ROS with a functional SLAM library for the hardwares we are using. After finding a ROS, in our case the teamused both ROS1 and ROS2 with ROS1 being used for the hardware we currently possess and ROS2 is used to work with a virtual robot to test out simulations. The testing for SLAM requires selecting the best fit packages fitted to our needs, which usually can be determined by the hardware we are using as there are recommended SLAM for each type of hardware. In order to test SLAM, the team has to have a working ROS with the proper version as well as it works with RViz (ROS Visualization) which is a program that allows users to see a perception of what the robot (both real or virtual) sees from a bird's eye view. After making sure that all the software can be properly executed, we then tested if the hardware works as well which is done by simply executing the program with a set of instructions, so in case of LiDAR, plug in LiDAR to Raspberry Pi and execute the instructions. After making sure that there are no issues opening up softwares, the team needs to test if the data can pass through such as ROS passes the output to SLAM for mapping, this is true for all hardwares. This can be tested by doing a test run, for a LiDAR, the team will walk around to see if it can detect walls and objects, this is the same for cameras as well.

After making sure all the functions of the hardwares and softwares are working as intended, we need to make sure that the maps and data can be stored properly, usually this isn't a problem due to the fact that all ROS version have a save file function, unless the data size is too large, which is the case for the RealSense camera, as the memory card we possess is limited due to budget constraints.

Scripts and Commands

As we are the spearhead of this project that will be passed down to future groups, we wrote a bash script that allows the team and future teams to execute our program using one line of command which usually requires opening multiple terminals and multiple sets of commands. This bash script code is easily tested by finding the path that we need to take in the terminal and implement it in a script. As for the ROS2 and the virtual robot, it is required to put in commands in the terminal to access certain functions unlike testing LiDAR and cameras in which all functions exist in the ROS interface. Not only that, the team needs to make sure that the programs are able to exit gracefully, which can be done using CTRL-C in the Linux terminal.

Usability Testing

This is a unique project as it does not have an “End User.” Our project is being continued by an electrical engineering capstone team who will take this project and build upon it to bring it to as near to completion as they can. Therefore, they are the end users as they are the ones who will interact with our product. A successful usability test would require them to spend minimal time to get our system up and running on their machines with little headache. In order to accomplish this, we will create 2 documents to help them: an installation guide and a user manual.

Installation Guide

Throughout this project, our team has been finding different ROS packages that can be installed and utilized successfully. Surprisingly, this was quite a challenge to find working packages in combination with our hardware and a compatible version of ROS. In the end, our team found the following items to work together:

- Operating System: Ubuntu 20.04
- ROS: Noetic Ninjemys
- Slam Package: HectorSlam
- LiDAR Node: RPLiDAR

These packages each have their own unique install instructions. In order to ensure the success of the system, our team will create an installation guide with detailed steps and links to helpful articles about how to install and test the installation of each package. With this document, the installation process should be streamlined and much more accessible for the teams to come after us.

User Manual

Now that everything has been installed for future teams, it is critical that they have a clear understanding of how to use the system. Currently, the process to run the system involves 2 command line terminals, an RViz window to view the output of the LiDAR and a plethora of commands. While to our team, this is not a problem as each member has an in-depth understanding of the system and its file structure paired with a background in computer science, to a group of electrical engineers this could prove to be complex and an issue. Team Mockingbird has 2 solutions to this problem: immense documentation and simplifying the process of running the system.

Each node inside of ROS has a launch file. Inside these files are details of exactly what will happen upon the initialization of the specific ROS node. These things can vary from the frame rate at which the LiDAR node scans, to the type of output that is displayed in the Rviz visualization window. With such widespread applications it is important to clearly state what the code is doing inside each portion and what the effects of certain changes are. This is done by adding documentation to the launch files. This way, when problems occur, troubleshooting can be an easier process.

The next task is to simplify the process of running the system. For this, our team takes inspiration from the launch files that exist for each ROS node. Our team has decided to make a bash script to run the commands necessary to get the system up and running. This script needed to be simple to use, effective, and provide appropriate messaging to the user. The bash script performs similar operations to the launch files as it contains the configuration for the system. The best part of the bash script is that it reduces the process of opening multiple terminals and several lines of code to one simple command

with no command line arguments. This should ensure that future teams can access and run the system with ease.

Conclusion

Autonomous drones are a major advancement in many different fields of study and service, including the study of intricate details in the rainforests. Dr. Alexander Shenkin is a researcher and climatologist who is attempting to create detailed maps of plots of forested land. His current workflow is slow and tedious, requiring large investments in time and manpower in order to create a map which omits crucial data. This approach is unwieldy and unsatisfactory, and a better solution is needed.

Our mapping system will greatly improve Dr. Shenkin's workflow in his study of the rainforest, allowing for faster and more efficient data collection from our ecosystems. This data will help us understand our environment, revealing important conclusions such as the effects of climate change and the mitigation of carbon in the atmosphere.

In order for this system to be as useful to Dr. Shenkin as it can be, it needs to be dependable as it will be mounted to expensive drones and the system itself is built with expensive components. This software testing plan ensures that each piece of the hardware not only functions correctly on its own but also when connected with each other component inside the system. We will create a user manual to pass on to future users in order to better use our system and make sure its simple to use and easy to install. Our navigation package will be tested and documented so that all behavior from the navigation instructions can correlate to some behavior described in the user manual.

By testing all 3 of these components, we can ensure that our system will be reliable and predictable as long as the lifespan of the hardware. This will enable Dr. Shenkin to get back to mapping environments and performing cutting-edge research.