F1/10 YellowTails

Technological Feasibility Assessment

Bowen Boyd
Hanyue Wang
Kyle Watson
Jordan Wright

Faculty Mentor:
Isaac Shaffer

Project Sponsor:
Truong X. Nghiem, Assistant Professor, SICCS, NAU
Trong-Doan Nguyen, PhD Student, SICCS, NAU

11/08/19

Table of Contents

1. Introduction

F1/10 Yellowtails is a team that was formed with the goal to improve access to the F1/10 autonomous racing platform. The members of F1/10 Yellowtails are Bowen Boyd, Hanyue Wang, Kyle Watson, and Jordan Wright. We are creating a new autonomous racing interface system called RosConnect. The project is sponsored by Dr. Truong Nghiem and his graduate research assistant Doan Nguyen. Dr. Nghiem is the Director of the Intelligent Control System Lab, or ICONS lab, at Northern Arizona University. At the ICONS lab our clients are creating new theories and algorithms for intelligent and high performance control systems. This includes developing solutions for the transportation industry. Our clients are now focused on improving algorithms for self driving cars.

Self-driving cars have proven to be the future of the automobile industry. The ever-growing need for greater road safety, reduced road congestion, and environmental stability means that vehicle autonomy advancements are particularly vital for society. This new, ever-growing industry needs more engineers to help extend outreach to the general population and solve the problems that are holding it back. However, there are currently no high-school programs that provide autonomous technology education to students, especially those with limited coding experience. This lack of access, to the emerging field of autonomous technology, results in potential engineers who choose other fields and under-educated leaders of tomorrow.

Dr. Nghiem and Doan are creating a summer camp for high school students that focuses on autonomous vehicles that lowers the barrier of entry to this technology. Through this new learning opportunity they hope to get high school students interested in STEM and recruit them to study at Northern Arizona University. The F1/10 autonomous racing platform is an RC car that is one-tenth the size of a formula one race car. It is powered by an Nvidia Jetson computer onboard the RC car and has sensors that you would find in a real self driving car. Our clients want high school students to receive hands on experience with said RC cars to understand how cars work.

The problem and premise for the creation of this project is that the F1/10 autonomous platform, in its current state is complicated to operate. The Nivida Jetson runs the Ubuntu Operating System which is then used to run the Robotic Operating System (ROS), in order to control the car. Everything must be run from the command-line and ROS adds additional hurdles like the need to source your workspace directory and use the Catkin compiler to compile your C code which needs two other files to be correctly set up syntactically, with dependencies and the correct parameters for the package you are trying to make. Changing one setting might also require you to change multiple files in different directories.

Dr. Truong Nghiem, Doan Nguyen, and F1/10 Yellowtails aim to make ROS more accessible and easier to use with RosConnect, a Graphical User Interface (GUI) for driving autonomous F1/10 vehicles. With it's intuitive design, this new interface system gives access to autonomous racing to every high-school student, irrespective of her or his coding competency. Shortcuts included in RosConnect such as node switches, drag-and-drop file fields, and kill switch controls will not only provide a learning experience to any high-school student but also any researcher interested in the F1/10 autonomous racing platform.

This document details RosConnect's technological feasibility by analysing and discussing several technologies that are key to its development. We begin in Section 2 by reviewing our environment variableS and the primary obstacles we need to overcome to create RosConnect. These obstacles include: GUI Framework, Auto Documentation, Testing Framework and Communication Methods between pur software and the F1/10 car. In Section 3 we will take a look at each obstacle individually and discuss technologies that can help us overcome each technical challenge. Since our challenges are highly coupled, we will then explore, in Section 4, a solution that connects each of the challenges in a sensible way. Lastly, we will present a summary of our findings, a brief description of remaining open questions, and an explanation of how we plan to address said questions, and can be found in Section 5.

2. Technical Challenges

Successfully building RosConnect requires overcoming several technological challenges. We will begin by introducing a list of environmental constraints with a brief explanation. These environmental variables demand no further analysis because of their fixed nature in relation to our system design. We will then discuss the four primary obstacles that will drive the design decisions necessary to create RosConnect.

2.1 Environmental Constraints

Raspberry Pi

The host machine to be used at the summer camp that will run ROS and RosConnect is a Raspberry Pi. The Raspberry Pi will have an Ubuntu Linux OS on board and will be used to communicate with the Nvidia Jetson board on the vehicles.

Ubuntu

Running ROS (our platform for robotics operations) requires a user to be running an Ubuntu Linux Operating System.

Robot Operating System (ROS)

The F1/10 racing platform foundationally relies on the Robotic Operating System. This system works as a framework using the concept of an OS. In general, ROS consists of code and tools that help make the F1/10 vehicles run, including the infrastructure for running it, similar to messages passing between processes.

ROS is designed to be a loosely coupled system where a process is called a node and every node should be responsible for one task. Node communication is achieved via message passing through logical channels called topics. Each node can send or get data from other nodes using the publish/subscribe model. The primary goal of ROS is to support code reuse in robotics research and development, and hence is our primary means of operation.

Node Communication

Nodes are simply instances inside ROS, which implies that any means of communication between them is fixed to the ROS platform.

F1/10 Autonomous Platform

The RC car platform is a predetermined platform developed by a group at Penn University. The platform is well documented so that all cars have the same components making races dependent on a teams software.

2.2 Design Factors

2.2.1 Graphical User Interface

The current work flow of the F1/10 racing platform is using the command line to start and navigate through ROS. We want the user to focus on understanding how ROS works and not how to use the command line. High school students will be able to control the car by using our software's Graphical User Interface (GUI). Our GUI needs to guide high school students through ROS so that they can use it to control the autonomous racing car. Our GUI also needs a kill switch in case of an emergency situation such as losing the connection between our software and the car or when the car goes out of control.

2.2.2 Auto Documentation

Documentation is one of the key tools that allows anyone to understand your codebase. Since our project is for a high school summer program we want to document our project so that the students can see how our code works. We are researching auto documentation tools to improve our workflow and remove save time. In the end we will not have to worry about writing all the documentation by hand and get more time to focus on our project.

2.2.3 Unit Test Libraries

After refactoring code, we as programmers must ensure that changes in our project's system do not break the design contract of said system. Unit Testing allows for this assurance. Since our goal is to make RosConnect scalable, we will need to develop various combinations of files into a working system of nodes to achieve autonomy. This means that we will have many unique cases and implementing/testing each one separately is near impossible without a unit test library.

2.2.4 Communication Method

One of the core features of our software is sending data to the F1/10 RC Car and maintaining a connection. Through this connection we need to be able to send a kill command and transfer data from the car to our software. If we do not maintain some form of connection our kill switch will no longer be able to send the kill command. Therefore, in case of emergencies we will have a script that auto runs on the car checking connections and if it finds there are none, will activate the kill command locally.

3. Technology Analysis

In this section, we will present a detailed analysis of the technologies necessary to create RosConnect. For each category we will explain why we need or want that particular tool for the creation of our system and then give metrics for how we will compare them. Lastly, we will explain our choice and discuss how we are going to prove the feasibility of said choice in the scope of our system.

3.1 Graphical User Interface

3.1.1 Overview of Problem

A GUI will help lower the barrier of entry for the high school students and is the backbone of our project. By replacing the command line high school students will be able to work with ROS to control the car. We also want to keep the system scalable so we want a rich feature set for our GUI. Our client has requested that our GUI implements a kill switch for situations where we lose control or connection in an autonomous robot race. The GUI will allow high school students to use the F1/10 platform, and therefore should be clear and easy to read. A good GUI framework will help us solve these problems.

3.1.2 Metrics

- **Documentation**
    - Multiple references of documentation will help better our understanding of a given candidate framework by either providing more information than any single reference or presenting the same information differently.
    - This metric will be based on the following references of documentation: Tutorials, Books, Wiki, Repository/Git, and Open Source.
    - Scoring Method: 0-5 where 1 point is assigned per provided reference from above, and 0 is assigned for a reference that is not provided.
- **Testing Suite**
    - The existence of a builtin testing suite will allow us to test our code faster, and improve our work efficiency.
    - Scoring Method: 1 for YES, 0 for NO.
- **Designer**
    - The existence of a graphical interface designer will provide us with a tool for easily creating a GUI layout.

    - Scoring Method: 1 for YES, 0 for NO.

- **Familiarity**
    - We want to take into account our team's current experience working with a given candidate framework to give us a sense of where our team's starting point of understanding is.
    - If more team members have experience with a framework, then we can theoretically use and implement said framework more quickly and more easily.
    - Scoring Method: 0-4 with each point representing a team member who has experience with the given framework.

3.1.2.1 Kivy

About

Kivy is a completely free-to-use cross platform Python GUI framework. Kivy is business friendly and is under the MIT license. Kivy has the ability to handle animations, caching, gestures, graphics and more. It also has built-in user interface controls such as buttons, cameras, tables, slides, and tree controls. In regards to our project Kivy runs on Ubuntu. Kivy is an open source toolkit that allows programs created with the same source code to run across platforms. It focuses on innovative user interface development such as multi-touch applications. Kivy also offers a multi-touch mouse simulator.

Analysis Overview

As can be seen in Table 1, the forms of documentation found for Kivy include tutorials, books, a wiki page, and a Github repository which is open source. The testing suite for Kivy is a unit testing platform which is located in the kivy/tests folder, and is separated into two parts: Non graphical unit test and Graphical unit test. Additionally, Kivy does provide a graphical interface designer called Kivy Designer. Lastly, only one of our teammates has experience with Kivvy.

Documentation Analysis						
	Tutorial	Books	Wiki	Repo/Git	Open Source	Total
Kivy	1	1	1	1	1	6

Table 1: Documentation Analysis for Kivy.

Results

- **Documentation:** 6

- **Testing suite:** 1
- **Designer:** 1
- **Familiarity:** 1

3.1.2.2 PyQt

About

PyQt is a cross-platform toolkit for creating GUI applications. It combines Python with the Qt library. The Qt library is one of the most powerful libraries at present. You can generate your GUI with Python scripts and package them into exe and other software platforms running on windows/mac/linux. Python's wrapper for the cross-platform GUI toolkit Qt implements 440 classes and a combination of 6000 functions and methods. PyQt is implemented as a plugin for Python.

Analysis Overview

As can be seen in Table 2, the forms of documentation found for PyQt include tutorials, books, a wiki page, and a Github repository which is not open source. PyQt provides a plugin option called pytest-qt for utilizing Pytest as a unit testing platform. In terms of this analysis, we shall consider pytest-qt a builtin testing suite for PyQt. Additionally, PyQt does provide a graphical interface designer called Qt Designer. Lastly, only three of our teammates has experience with PyQt.

Documentation Analysis						
	Tutorial	Books	Wiki	Repo/Git	Open Source	Total
PyQt	1	1	1	1	0	5

Table 2: Documentation Analysis for PyQt.

Results

- **Documentation:** 5
- **Testing suite:** 1
- **Designer:** 1
- **Familiarity:** 3

3.1.2.3 Tkinter

About

Tkinter is Python's default GUI library, and it is also the oldest Python GUI framework. The standard interface for using the TK GUI tools in Python is already included in the standard Python installation. In the installation, the well known IDLE is a simple GUI for creating GUIs using Tkinter. It is easy to learn and use.

Analysis Overview

The Tk GUI toolset bundled with Python is the Tcl code wrapped in Python. It is implemented by the Tcl interpreter embedded inside the Python interpreter. The Tkinter call is converted to a Tcl command and then passed to the Tcl interpreter for interpretation to create the GUI interface. In contrast to Tk and other language bindings, such as PerlTk, it is implemented directly from the C library in Tk. As can be seen in Table 3, the forms of documentation found for Tkinter include tutorials, books, a wiki page, and a Github repository which is open source. Tkinter comes with PyUnit as its builtin unit testing library. Additionally, Tkinter does not provide a graphical interface designer. Lastly, only one of our teammates has experience with Tkinter.

Documentation Analysis						
	Tutorial	Books	Wiki	Repo/Git	Open Source	Total
Tkinter	1	1	1	1	1	6

Table 3: Documentation Analysis forTkinter.

Results

- **Documentation:** 6
- **Testing suite:** 1
- **Designer:** 1
- **Familiarity:** 1

3.1.3 Chosen Approach

As is shown in Table 4, PyQt is our best choice with a score of 10. PyQt has its own Qt designer which will make creating the layout of the page easy. PyQt can develop a more beautiful interface and saves us time. It also supports Cross-platform. The Qt library is very powerful and PyQt allows us to call APIs in the Qt library using Python. The familiarity of these frameworks are the same across all candidates. Despite PyQt not being open source, the documentation of PyQt is detailed and specific.

Grading Summary: Graphical User Interface Framework					
	Documentation	Testing Suite	Designer	Familiarity	Overall Total
Kivy	6	1	1	1	9
PyQt	5	1	1	3	10
Tkinter	6	1	1	1	9

Table 4: Graphical User Interface Framework Results

3.1.4 Proving Feasibility

To prove feasibility, we will make a simple GUI prototype to test whether PyQt will work for our project. We will run the GUI on a condensed set of F1/10 racing platform files from our client to test the correctness of our choice. If all the requirements are met, then we can further develop the GUI by adding everything we need. We will use this prototype to determine if PyQt is a good fit for our project and is easy for high school students to use.

3.2 Automatic Documentation

3.2.1 Overview of Problem

Since reading documentation is such a big part of helping someone understand of what is going on with someones code, we need to figure out a good documentation tool to help produce the best documentation possible. This tool needs to be clean and very readable since we do not want the user to be confused on what is going on. This tool will help us focus more on creating a better product instead of writing all of the documentation.

3.2.2 Metrics

- **Implementation difficulty**
    - For the implementation difficulty we will see how hard it is going to create the auto documentation. This will be done by tracking how long it took to install the product. Once installed we will look at how many more steps we have to take to completely have it installed.
    - Scoring Method: 0-5. The score of 0 will mean it was hard to install and a lot of steps. A score of 5 will be it was easy to install and anyone would be able to handle this task easily.
- **Guidance Level**
    - For guidance level we will look at about six different factors that will help provide strong evidence of guidance if people need it. Each one of these factors will get a score of 0 for no information and 1 if there was any information at all.
    - One of the factors we are looking at is the release date and we will score this one different because if it has a date past 5 years ago it will be a zero and anything newer than 5 years gets a score of 1.
    - We are not looking at how much information we can find in each we just want to know if there is information out there for people to look for if they need it. Once we go through and look at all of these factors we will add them up and that will be the overall total score for Guidance Level.
- **Level of Organization**
    - For the level of organization we will look at five different factors that will help provide strong evidence of organization. Each of these factors gets a score of 0 or 1 where 0 means that it is lacking this factor and one being it is in the html document.
    - The factors are multiple pages, having an index page, clean layout, parameters that stand out and finally easy to read text. For the multiple pages we are looking to see if the information is all on one page or if it is across multiple.
    - Another thing that will help us is if there is an index or an index page that will let the reader go to a specific method that they want to look at.
    - We will look at the layout of the base document to see how easy it is to follow, which is a big thing to have for people looking at the documentation.Additionally, we will determine if the parameters stand out easily.
    - Lastly, we will determine the level of ease to read the text on the page.

3.2.2.1 Sphinx

About

In 2008 Georg Brandl a developer first released sphinx. Sphinx is an auto documentation that was written in python and mostly used by python but you can use it for other languages also. Since 2008 and the initial publication of sphinx it now is on version 2.2.0 and that was released in August of this year.

Analysis Overview

While researching the Sphinx it was really easy to install since all you had to do is just had to use the environment pip install. After you are done and you want to actually use the tool you have to run sphinx quickstart. Running the run sphinx quickstart in the terminal is probably the longest time to get your project set up. What this command does is it prompts the user with questions on how to set up the settings for the documentation. Since these prompts are only done one, when you are setting up the settings for the project it may take a little time to get used to. When producing the documentation it can be created two ways, html page or a pdf. While conducting the research we got the html part of the display down. The pdf version is a little more complicated so we did not go down that path too much. When looking at the html version of the documentation it organizes them by class There are plenty of videos out there and other documentation that you can learn so much from.

Documentation Analysis							
	Tutorial	Books	Wiki	Repo/Git	Open Source	Last Release Date	Total
Sphinx	1	1	1	1	1	1	6

Table 5: Documentation Analysis for Sphinx .

Organization Analysis						
	Multiple Pages	Index	Clean Layout	Parameters clear	Readable text	Total
Sphinx	1	1	1	1	1	5

Table 6: Level of Organization for Sphinx .

Results

- **Implementation difficulty:** 4
- **Guidance Level:** 6 taken from Table 5.

- **Level of Organization:** 5 taken from Table 6

3.2.2.2 PyDocs

About

PyDocs is a built-in library in python. Pydoc is a very simple and easy tool to use for creating documentation. The current version of pydoc is 3.7 which is actually the current version of python. Pydoc works with both in version 2 and 3. Pydoc is also a helper for specific method . If you want information about what all goes in with pydoc time for example will give you all the information you need to know about the method-time.

Analysis Overview

Since PyDocs is built into Python, so there is no installation process unless python isn't already installed on the computer. The output for PyDocs is either in the terminal or on a website or a text file. Printing it to the terminal is not the best since whoever is wanting to read our documentation would have to run it in the terminal. For our project we are trying to go away from the terminal. So having this option is not good. Printing to the website would be easier since as long as you have internet and it was already built you can just look at it. The third way to produce the documentation is in a text file. This option is another good one but the organization and readability might be a little harder to read. When researching we just focused on the html output for PyDocs so we could compare the websites with all the other options.

Documentation Analysis						
	Tutorial	Books	Wiki	Repo/Git	Last Release Date	Total
PyDocs	1	1	1	1	1	5

Table 7: Documentation Analysis for PyDocs .

Organization Analysis						
	Multiple Pages	Index	Clean Layout	Parameters clear	Readable text	Total
PyDocs	0	0	1	1	1	3

Table 8: Level of Organization for for PyDocs

- **Implementation difficulty:** 5
- **Guidance Level:** 6, taken from Table 7.
- **Level of Organization:** 3, from Table 8

3.2.2.3 pDoc

About

In 2013 Andrew Gallant created the documentation tool for people to use. This tool is compatible with both Python 2 and Python 3. The current Python 2 is using 0.3.2 with Python 3 it is 0.7.1.

Analysis Overview

Pdoc is another very simple auto documentation tool. The installation process was super easy since you all you have to do is pip install it. From there you just put pDoc and the class name that you want documented. Again like Sphinx you have to have it all documented in your class but you just call pdoc with your class name it creates an html for that class. So you still need to put up some work upfront but the website that it creates is super easy to read. With pdoc it includes the part of the code that it is documenting. I think that is a great functionality because if they want to see what it actually does they have it right there.

Documentation Analysis						
	Tutorial	Books	Wiki	Repo/Git	Last Release Date	Total
pDoc	1	1	1	1	0	4

Table 9: Documentation Analysis for pDoc

Organization Analysis						
	Multiple Pages	Index	Clean Layout	Parameters clear	Readable text	Total
pdoc	0	1	1	1	1	4

Table 10: Level of Organization for pdoc .

- **Implementation difficulty:** 5
- **Guidance Level:** 5 taken from Table 9.
- **Level of Organization:** 4 taken from Table 10.

3.2.3 Chosen Approach

In conclusion, we decided to choose Sphinx as our Automatic Documentation. Table 11 shows that all the total score of 14, 13 and 13 for Sphinx, PyDoc and pDoc respectively. First one is that you can have the output go to either an html page or even a pdf which is nice if people don't have internet, they still can read documentation. Another reason is when looking at pyDoc some of the websites with tutorials that we looked at were using sphinx to create it when it could of been the one which they were describing.

Automatic Documentation				
	Implementation difficulty	Level of Guidance	Level of Organized	Overall Total
Sphinx	4	5	5	14
PyDoc	5	5	3	13
pDoc	5	4	4	13

Table 11: Automatic Documentation Results

3.2.4 Proving Feasibility

To prove feasibility we are going to test it with our prototype that we are going to create for the tech demo for the end of this semester. This will let us know for sure that the choice of Sphinx for our auto documentation is actually going to work. Testing it this semester will give us more time to research more over break if there is anything that doesn't work. During this time of testing we will continue to look into creating the pdf for the documentation.

3.3 Unit Test Library

3.3.1 Overview of Problem

Autonomous racing, within the F1/10 platform, is achieved through a complex communication system within a network of nodes built from the Robotic Operating System. This means that coherent communication between nodes is vital for successful systematic operations. Furthermore, since RosConnect will build nodes determined by user input, it is easy to see that a platform to test and ensure the successful construction of nodes is necessary.

Unit test libraries enable the required assurances. This leads to the question of which libraries will be most effective in achieving our goal of testing variation from user input in accordance to node construction and communication. Furthermore, due to the Python language's vast libraries and easy maintainability, we will choose to focus the attention of our analysis to unit test libraries specifically tailored to the Python programming language. Specifically, we will analyze the following two unit testing libraries: PyUnit and Pytest.

3.3.2 Metrics

- **Examples**
  - Unit testing examples will give us an estimate of the thoroughness of documentation for a given unit testing library. Referring to documented examples will help expedite the process of producing various unit tests for our system.
  - Scoring Method: 0-5 denoting the number of testing examples provided by documentation for a given candidate.
- **Guidance**
  - Multiple references of guidance will help better our understanding of a given candidate framework by either providing more information than any single reference or presenting the same information differently.
  - This metric will be based on the following references of guidance: Tutorials, Books, Wiki, Repository/Git, and Open Source.
  - Scoring Method: 0-5 where 1 point is assigned per provided reference from above, and 0 is assigned for a reference that is not provided.
- **Auto-generation**
  - The capability of auto generating tests will be useful in conserving time spent on producing unit tests.
  - Scoring Method: 0 or 1 refer to "not provided" and "provided", respectively.
- **IDE**

- Since RosConnect will build nodes via bash commands that call Python scripts, we must determine if the unit testing framework requires an IDE to run.
- Scoring Method: 0 or 1 refer to "required" and "not required", respectively.

3.3.2.1 PyUnit

About

PyUnit is a Unit Testing framework that operates as the Python language version of JUnit. PyUnit is incorporated in all versions of Python (after the year 2000) as the keyword unittest. Since PyUnit is the standard unit test framework that comes with Python, it is a natural candidate to consider.

Analysis Overview

The implementation of testing examples was relatively easy given that PyUnit is built into Python and Python had been previously installed on the local testing machine. Since PyUnit comes with all the latest versions of Python and Python is well documented, there exists documentation to implement at least a dozen simple example tests. In terms of this analysis, we implemented five tests in total. The first three tests, provided explicitly by documentation, involved checking for all uppercase, all lowercase, and the existence of a space between the substrings "hello" and "world" for the well-known HelloWorld program. Additionally, we implemented two tests that check if an array of integers is completely fill with even integers and if an array of integers is completely filled with odd integers. Furthermore, as can be seen in Table 9 below, we found tutorials, books, and a wiki page for PyUnit, but did not find a reliable repository or open source resource for PyUnit. Lastly, PyUnit allows for automation of test generation through a context manager called subTest and an IDE is not required to make/run tests.

Guidance Overview						
	Tutorial	Books	Wiki	Repo/Git	Open Source	Total
PyUnit	1	1	1	0	0	3

Table 12: Guidance Overview Results For PyUnit

Results

- **Examples:** 5
- **Guidance:** 3
- **Auto-generation:** 1

- **IDE:** 1

3.3.2.2 Pytest

About

Pytest is an open source Unit Testing framework that operates for the Python programming language. Pytest 4.6 series is the last to support Python 2.7 and 3.4 while pytest 5.0, the latest version, will support only Python 3.5+. Since Pytest is used by bigshots such as Dropbox and Mozilla, it is a natural candidate to consider.

Analysis Overview

The implementation of testing examples for Pytest was extremely easy given that Python had been previously installed coupled with the fact that there exists easy to read documentation providing well written examples that was found with a single google search. The examples that we implemented for this analysis included an uppercase test, lowercase test, and the existence of a space between the subwords “hello” and “world” for the well-known HelloWorld program. Additionally, we were able to implement two other tests which included checking for the first seven primes and checking if each number in an array is prime. Therefore, the total number of implemented examples was 5. Furthermore, as can be seen in Table 10 below, we found tutorials, books, a wiki page, and a Github repository which is open source for Pytest. Lastly, Pytest auto generates tests through a feature that allows parametrizing any fixture and an IDE is not required to make/run tests.

Guidance Overview						
	Tutorial	Books	Wiki	Repo/Git	Open Source	Total
Pytest	1	1	1	1	1	5

Table 13: Guidance Overview Results For Pytest

Results

- **Examples:** 5
- **Guidance:** 5
- **Auto-generation:** 1
- **IDE:** 1

3.3.3 Chosen Approach

Grading Summary: Unit Test Library					
	Examples	Guidance	Auto-generates	IDE	Total
PyUnit	5	3	1	1	10
Pytest	5	5	1	1	12

Table 14: Unit Test Library Results

In conclusion, we decided to choose Pytest as our Unit Testing Framework. We found that although the documentation for PyUnit provided 5 well-guided examples (as can be seen in Table 14), the capability of navigating the documentation was only moderately easy as compared to Pytest's documentation. The documentation for Pytest not only provided 5 well-guided examples but also proved to have a high ease of navigation. For the guidance metric of this analysis we found that the overall score of 5 for Pytest outweighs the score of 3 for PyUnit. Furthermore, as denoted in Table 14 with the scores of 3 and 5 for PyUnit and Pytest, respectively, we contend that the existence of a Github repository and the open-source nature of Pytest make for a more suitable Unit Testing Library. Our reasoning is that the Github repository and open-source nature of Pytest simply provide more information to reference as compared to PyUnit. More information will help expedite the process of understanding how to use a given candidate framework. Lastly, from Table 14, we see that both candidates scored a 1 for Auto-generation and IDE giving total scores of 10 and 12 for PyUnit and Pytest, respectively.

3.3.4 Proving Feasibility

A simple way to demonstrate the feasibility of Pytest will be to create a single test node within the network of nodes built from RosConnect. By inserting a <test> tag in the roslaunch file we can generate a test node corresponding to said tag. We will then use Pytest to generate unit tests inside said test node. After roslaunch has been exited, a wrapper for roslaunch named rostest will intercept all output from Pytest and produce a single Integration test report in XML.

3.4 Communication Method

3.4.1 Overview of Problem

We need to choose a connection type that will let us transfer the ROS nodes quickly and reliably. We then need a connection that is constantly handshaking with the car to make sure we are in control and can stop the car of our own volition. Our client wants a kill switch implemented as the F1/10 cars are expensive and can reach speeds that can total the hardware. The car will be connected to the same network as the Raspberry Pis that are ruining our software. If possible we might even implement connection redundancies to assure a constant connection. Lastly we need a way for the ROS nodes on the car to send data back to our software

.

The Robotic Operating System, or ROS, is at the core of our project. The F1/10 car uses ROS on top of Ubuntu 16 to do everything from driving to using the on board sensor to determine how to drive itself. Our GUI will send code to the car and allow users to change parameters in these nodes. The RosCore, or master node, has a variable called the ROS_MASTER_URI. This variable tells all nodes where they can find the master. We can set the ROS_MASTER_URI variable to another computers RosCore. This technique is used to run ROS “headless”. For example, if a robot has a lot of sensors but the onboard computer is not strong enough to visualize the data it is receiving, you can set the Master Node on the robot to the RosCore running on your desktop.

The most common way to set the ROS_MASTER_URI is to set it to the “heads” IP address. This requires that both the head and body be on the same network. Alternatively you can set the ROS_IP to the IP address of the ad-hoc bluetooth network. The ROS_MASTER_URI environmental variable is the ROS way to handle communication between two different machines. Its range depends on what backbone it is placed weather that is Wi-Fi or Bluetooth. Because our software needs to access the data on the RC car we will have to use the ROS_MASTER_URI in our project. We will compare BlueTooth and Wi-Fi to determine which communication method will be best for our project.

3.4.2 Metrics

- **Range**
    - When considering what forms of communication to use we need to look into the range of the communication standard. In our case the range is hardware bound to what is on the Nividia Jetson and the Raspberry Pi. We will rank them based on

which is longer with 0 having the shortest distance and 1 having the longest distance.

- **Experience**
  - We also want to take into account our teams experience working with the communication method. For simplicity sake we will use a scale from 0 to 4 with each point representing a team member who has experience with the given technology.
- **Reliability**
  - The reliability of the connection is important. If the connection fails the car should stop. This could become frustrating as the end users will not have control over the connections working in the background. We want to consider if there are multiple timeouts. Do we have to manually reconnect the two devices on a regular basis? Whats is the communications methods bandwidth? We will use the following table to determine a reliability metric:

Reliability Analysis			
Frequent Disconnects/Time outs	Have to set up multiple times	Transfer Speed	Total Score
Yes/No	Yes/No	1 for slowest 2 for fastest	Noes + Transfer Speed Rank

Table 15 : The scoring system for determining the reliability of a communication method

- **Abstraction level**
  - Lastly we want to consider on what hardware level the connection runs on. If the communication method is in the hardware (i.e BlueTooth receiver for the RC car) it would be better for emergency signaling. We will use a point system where:
    - Runs on the OS +1
    - Runs directly on the hardware +2

3.4.2.1 Wi-FI - SSH

About

SSH was created in the 1990's to replace older methods of connections and communication protocols. Tatu Ylonen who was a researcher at Helsinki University of Technology in Finland when there was a hacker who used a password-sniffing attack to get usernames and passwords from hundreds of students and staff. In response Ylonen started working on Secure Shell which encrypted all network traffic which was years ahead of the then current plain text traffic. Ylonen make his work open source and is used all over the word today to provide secure connection to remote machines.

Analysis Overview

SSH requires that both machines be connected to a network. In our case the car and Raspberry Pi will be on the same wifi network and all we will need is the ip address of both. If the car and Desktop where on different networks port forwarding would have to be set up. The Nvidia Jetson TX2 wifi specs are 802.11a/b/g/n/ac 2×2 867Mbps which on average has a range of 50 feet indoors. All the members of our team have experience using SSH and understand how to use it. One of the downsides to SSH is that the connection can timeout randomly. This varies depending on the host and remote machine settings as well as the network. In addition if the IP address of the Nvidia Jetson changes the new IP address has to replace the old IP listed on the Machine that is remoting in. SSH also runs on the OS level and only works if the Nvidia Jetson is connected to a network.

SSH is a powerful tool that can allow us to transfer files to the car and run command line arguments. It requires that both the car and the Raspberry PI be connected to a network and for static ip address or a way for updating the ip address on both the cars OS and our GUI. SSH would allow us to easily push files from our GUI to the car but might not be the best solution for keeping a constant handshake.

Results

- **Range:** 0
- **Experience:** 4
- **Reliability:** 2
- **Abstraction Level:** 1

3.4.2.2 Bluetooth

About

Bluetooth was originally designed to replace RS-232 cables which are used for serial port connections in 1989 and more specifically Nils Rydbeck, the original mind behind the new form of communication, wanted to develop a wireless headset. Originally named "short-link" until an employee at Intel proposed the name Bluetooth after a 10th centry Danish king who united all of the Danish tribes into one kingdom. This embodied the concept of new communication system that united a variety of communication protocols. In fact even the Bluetooth logo is the bind rune merging King Harald Bluetooth's initials. Bluetooth has come a long way since its inception with more than 4 billion devices with Bluetooth technology expected to ship in 2019.

Analysis Overview

The Raspberry Pi 4 has the newest version of the Bluetooth protocol, Bluetooth 5. Bluetooth 5 is rated to cover distance from 100ft to 1000ft depending on the environment which is around 3 times the previous protocols range. Files can be transferred over Bluetooth with speeds up to 2Mbps making it slower than the average WiFi based options but 2 times faster than Bluetooth 4.2. On the upside is that bluetooth is meant to maintain a connection with one device being able to potentially wake another. No one on our team has implemented Bluetooth into software but we all have experience using it as consumers. Bluetooth has the potential to be accessed from a hardware level but the F1/10 cars bluetooth module is on the Nvidia Jetson making it OS bound for our application. It is also important to note that ROS can send and receive data over Bluetooth.

Bluetooth is a truly dynamic communication method that can allow multiple devices to connect to one another directly with no middleman like Wi-Fi. While its data transfer speed is not as fast as Wi-FI it does have a range advantage. There is also no need to work about changing IP address or identifiers. Implementing Bluetooth would give us a reliable connection to the car at all times and can be used to trigger the kill switch that the client has requested. We can also have nodes that post topics on a Bluetooth connection and nodes that subscribe to that topic on another Bluetooth connected device.

Results

- **Range:** 1
- **Experience:** 0
- **Reliability:** 3
- **Hardware Level:** 1

3.4.3 Chosen Approach

Due to the nature of ROS we will have to use the ROS Master URI communication, the question is whether to use Wi-Fi or Bluetooth for the underlying network. Bluetooth offers farther range but at the cost of speed and doesn't require keeping track of IP addresses. On the other hand, Wi-Fi is faster but has less range in addition to having to deal with changing IP addresses. For the backend of the ROS communication we will go with Wi-Fi. This allows us to SSH to send the files needed to the car and run commands on the F1/10 car. We need to SSH into the car to execute bash scripts that will run the roslaunch command with all previously sent files. This script will also set the ROS_MASTER_URI variable to our desktops current IP address. A script on board of the car will keep track of which connections are active. Should all communication channels close this script will initiate a kill command.

GUI F1/10 Car Communication Methods					
	Range of connectivity	Familiarity	Reliability	Abstraction Level	Overall Total
Wi-FI SSH	0	4	2	1	7
Blue- tooth	1	0	3	1	5

Table 16: GUI F1/10 Communication Methods Results

3.4.4 Proving Feasibility

Now that we know how we are going to communicate with the car our next step is a working prototype where we can send files to the car, maintain a connection to the car and get data back. Using multiple forms of communication allows us to have redundancy in case one of the connections fails. Our prototype needs to be able to send a kill command to the car as it is one of the few specific features our client has requested.

4. Technology Integration

4.1 Overview

We need to provide a layer of abstraction between ROS and the F1/10 platform so that high school students can work with autonomous racing technology. While creating this abstraction between ROS and the car we have to make sure everything still runs smoothly. Our software also needs a communication method to connect to the F1/10 car. We want to make sure our code is reliable and bug free. In addition, we want extensive documentation for our code so that it can be understood and used by many developers after us.

4.2 Integration Challenges

When we are integrating our GUI and the car we will address the various hardware connections involved. On the car in the raspberry pi there is going to be a couple of USB ports that we can use if we need to connect a keyboard and mouse. The Raspberry Pi also has an HDMI cable for direct connection and so we are able to see the output on the screen and work directly with the hardware. The other option is using the computer/GUI to connect to the car using wifi to SSH into the car to start everything that way.

One of the main challenges is connecting to the car when the lidar is on already. If this is the case right now we have to disconnect the lidar and then connect via SSH and then restart it so it will be able to use it. Another integration challenge we face is lost connection with wifi on the car. If the car loses wifi we need to make sure that it will be able to still stop since we may not have control over the car anymore. The GUI that we create will hold functionality with ROS. So, our GUI will have to communicate with the Raspberry Pi and Jetson Nividan board so it knows when something has started. Table 17 below shows a flow chart of hardware connections in the system and our communication method's role in this system.

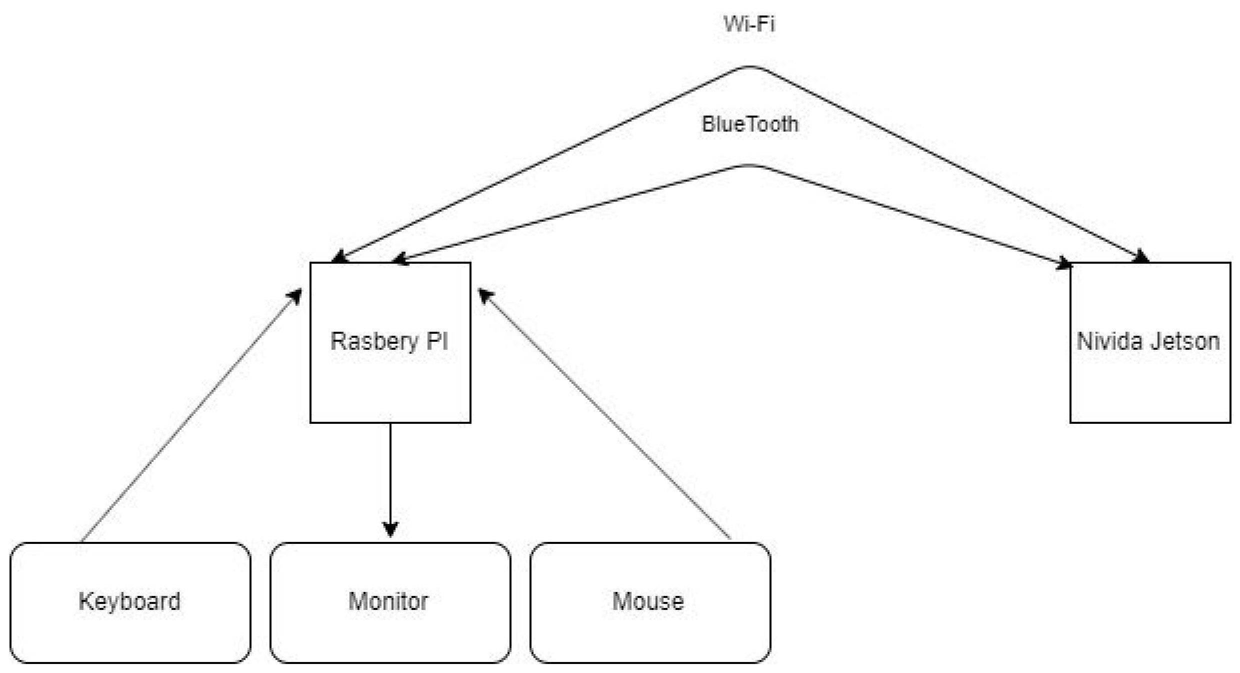

Table 17: Hardware integration diagram.

5. Conclusion

RosConnect will correct the need for high school student exposure to autonomous technology by providing a new GUI with intuitive design, and we are confident in our ability to build the RosConnect application by the end of this project. In Table 18, we list the challenges, solutions, and our confidence level that a given solution will perform as expected.

Challenges and Solutions			
Tech Challenges	Solution	Confidence Level	Backup Solution
GUI	PyQt	High	Kivy
Auto Documentation	Sphinx	High	pDoc
Unit Test Library	Pytest	High	PyUnit
Communication Methods	Wi-Fi	Medium	Add Bluetooth

Table 18: Results of our Challenges with backup solutions and how confident we are about them.

Our presented solutions are straight-forward and robust. PyQt is the go-to graphical user interface for the Python programming language. Sphinx provides an easy-to-use platform for implementing readable documentation for future users, and Pytest is considered the run-anything, no-required-api testing framework. Our major hurdle will be utilizing Wi-Fi as the communication method between RosConnect and the F1/10 vehicle. Immediate next steps in overcoming said hurdle will be to create a simple prototype that maintains connection to the car, for some predetermined amount of time, while we transfer data and receive data from the car. In conclusion, we are excited and eager to provide this great application called RosConnect that will make high school student's hopes and dreams come true!