Technological Feasibility

11/10/2022

Floor Explorer Algorithms Team

Sponsored by: Michael Leverington

Mentored by: Rudhira Talla

Members:

Jacob Doyle, Armando Martinez, Luke Domby, Aidan Halili, Vincent Machado

Table of Contents

1. Introduction

As the years have gone by we have seen an increase in the use of robotics in many different fields, but unfortunately, classrooms have not been able to keep up with this rising demand. We have seen non-trivial simulators and limited functionality robots in the classroom setting, but what if we had a higher standard for robotics software? What if there was a way to integrate more of this at an “affordable” cost? When we say affordable, it’s important to denote that this means affordable in terms of the average household, not that of an academic organization.

Our sponsor, Michael E. Leverington, has been attempting for years to create both; a robot capable of being modular in its use and programmability; and modular software implementing a basic navigational component. He recently came up with the idea of using the IRobot Create 3, to help test the navigational software components. The Create 3 is a small circular robot with similar sensors and capabilities to that of a Roomba vacuum, a common robot found in society today. The Create 3 is a relatively cheap (under $500) solution, and easily accessible to those with access to computers. It comes standard with basic mobility actuators and a variety of sensors, all which can be programmed to complete user decided tasks. Team F.E.A.T. is dedicated to creating a robotics platform that can be programmed as needed to a variety of similarly functioning robots. If it seems vague when we say, ‘similarly functioning robots' ', that’s because it is, we will discuss this in more detail later. Aside from creating a modular platform, it will show proof of concept and function as a tour guide for the 2023-2024 academic year at Northern Arizona University.

We begin this challenge by looking at the previous versions of the project, which have, intuitively, ended in failure. Of course with the addition of the Create 3 robot, the project does vary in multiple aspects, but the major milestone continues to be the navigational component. One previous team, in fact, did not actually have access to a functioning robot. Furthermore, it is quite ironic that this was the team that seemed to have actually gotten the closest to the goal. This change, while useful for F.E.A.T., is not the only problem the project has encountered in past iterations. Multiple teams' failures have stemmed from multiple areas, but the biggest and most glaring issue seems to be with their original feasibility reports. More specifically it was and their focus on one specific aspect of their robots sensor capabilities to control navigation. For example, one team chose to focus solely on self localization via router triangulation. This proved to be far too slow to accomplish the minimum viable product. Another team chose to focus on a virtual map creation which called for a high dependency on the robots hardware. As you can see, there are many things to consider when attempting to overcome this challenge.

Our solution is a more practical and modular approach to programming the robot. We aim to combine the different attempts that previous projects attempted, in tandem with the newly provided resources. This includes the robust yet simple nature of the Create 3 to create a program capable of self navigation that could be moved to other robot platforms with minimal amounts of configuration. The goal being to use the robots wifi capabilities to help it self locate after a short length of time when it believes it has reached certain checkpoints in its digital map. More specifically, it will use its various physical sensors and odometry mechanisms to keep track of its location and movements in real time. This combination of approaches should allow the robot to successfully navigate around unforeseen obstacles, while maintaining a backup system, should it move off course. Our code will also maintain modularity by containing multiple changeable constants that can be calibrated to fit sensors and actuators of those of which are not provided or compatible to our Create 3. We believe these changes from previous attempts will be enough to successfully solve the programming problem with robots, and in turn, help push their use in various classroom environments. With that said, in the exploration of the Create 3 capabilities and have come upon a few different technical challenges.

2. Technological Challenges

When we began looking into Create 3, a few issues immediately arose. The most obvious problem we faced was the robot's method of connecting to wifi. By default the robot only accepted a single password for connecting to wifi so that we could communicate with it and issue it various commands. This led to us having to use a hotspot to initially connect to the robot. We are looking into how to overcome this design flaw through a variety of methods, with the most current direction being the use of its MAC address to facilitate a connection to wifi manually. If necessary, we will write additional code that will allow us to directly connect the robot to a school's network. Another more prominent issue that we are facing is the robot's sensors. While being highly accurate, they are significantly worse than initially thought. Our preliminary tests indicate that the IR sensors have a very short range of only about a foot for strong/usable feedback. We will have to adjust our approach and make design decisions for our code that will take into account the IR sensors shortcomings. This means, the implementation of code that will be compatible/compliant with varying sensor types, such as LIDAR and SONAR.

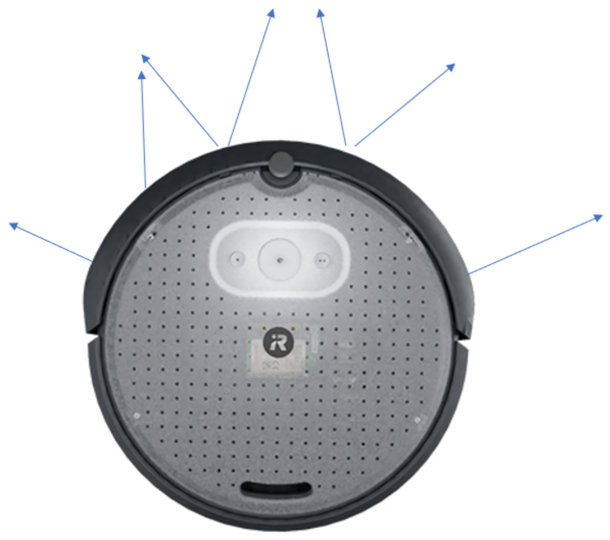
Image 1.1

The next hurdle we have to overcome is the coding language that is used for the robot itself. The Create 3 uses ROS2 Galactic which is the most stable on the outdated version of Ubuntu 20.04. This has led those in the team who do not use linux to search for various solutions to the problem. Virtual machines have been our solution so far but come with a few problems of their own, such as the simulated computer not counting itself as being on the same network as the robot. We've acquired a project specific linux machine running that specific version of Ubuntu that will accompany the robot wherever it goes to help lessen the complexity of connecting to the robot. Another temporary solution is the use of the Create 3 Web Playground in Python. This has given us access via bluetooth to the robot, in which we are able to send limited commands that were provided by IRobot Education. We used this to thoroughly test various mechanisms of the standard Create 3 robot when ROS2 access was not available.

Our approach to programming the robot includes the ability to create a digital map using the robot and its sensors to map its surroundings. The idea is to allow the robot to hypothetically navigate any area if it is given the time to prepare. We will also have to be sure of the robots ability to navigate on any surface and that our code will not experience significant interruptions due to different flooring. This also includes the robots ability to avoid unforeseen obstacles appearing in its path (including humans) and be able to stop at a distance that would give the robot and the person enough time to properly avoid each other. To be able to successfully accomplish these goals and overcome the challenges that lay before us we have begun running tests on different alternatives and the sensor capabilities the robot offers.

3. Technological Analysis

Our goal is to program a robot that can navigate from one position to another without colliding with any obstacles or parts of its environment. On top of this our code should be able to function in different robotic platforms both to expand upon this in the future but also to allow it to be used as an educational tool in various classes and projects. The code also needs to be able to function with robots that use differing sensors from ours, for example changing from Infrared sensors to sonar based sensors. Both of these should be able to accomplish a similar job but the sonar sensors may be able to get more accurate reading from longer distances both allowing for more delicate responses and for planning future actions further ahead than our IR sensors would allow. On top of this the Create 3 is not set up to connect to networks that require both a password and username and as such is going to be a major hurdle for us to overcome. Finally our robot seems to have trouble with differing materials, some of the sensors seem to perform slightly worse than normal on reflective materials and the robot may not be as accurate on carpet as it is on tile.

Our ideal solution would be to create a program that works on any platform with only changes to constants that control the distances at which the robot would react and how the calculations would be done. For example our robot only has a reliable sensing range of one foot ahead of it but future robots may have different sensors that allow for more range and return different numbers as its output. Our goal is to have our code set up in a way that allows for easy modifications as well. We also need our code to use as little to no proprietary code that comes with the Create 3 as that would not work on non-Create 3 platforms.

For our first problem of creating code that is capable of moving the robot from place to place without hitting any obstacles we have come up with a two pronged approach. We have gone over what we could from the previous groups that attempted this and found that their approach was too focused on one singular aspect of the robot and in one case they had the unfortunate case of lacking a robot entirely. However, we learned that their approaches had their own merits, the group that used wifi to help localize the robot were able to do so accurately but took too long. Conversely the group that used sensors to track the robots position were able to make digital maps for the robot but they struggled to successfully keep the robot going where it needed to. So our approach will take advantage of both of those ideas, using wifi to get the robot back on track to its proper location at certain “checkpoints” and using the sensors with a digital map for real time calculations and movement. We plan on using the inbuilt cliff detection sensors to make sure the robot avoids any dangerous cliffs.

Issue:

We are looking for a platform that would let us cover both of the functions we are looking for: mapping the second floor using the IR sensors and reacting to obstacles in real time.

Desired Characteristics:

To accomplish this goal, we need a platform that offers libraries that not only add to our work, but that are as modular as possible in order for this project to be applied to other robots from a different manufacturer (or even self-made!). It also needs to be highly configurable in order to tackle edge cases or specific requests from our client in the future.

3.1 Alternatives:

Upon research, we found two alternatives:

- Python Playground Implementation: This alternative was recommended as a beginner's experience to the Create 3 Robot by iRobot Education. It allows FEAT to connect to the robot via Bluetooth and run a series of commands through Python and irobot’s educational package.
- ROS2 libraries + iCreate custom messages (Python or C++): A much more robust interface and implementation, recommended by the client. ROS2 is a series of libraries widely used in the development of robot related applications. The set up that iRobot Education recommends includes custom “messages” (commands sent to the robot) developed by them, that allow even more interaction between FEAT and the Robot.

3.2 Analysis:

Python Playground Implementation: While developed by the same company that designed the iRobot Create 3 robot, this alternative might not be effective in the long run, due to the several limitations it could bring to the project:

- Even though the functions that the education package provides are useful, they are limited by the scope of “educational”. These libraries are designed to work around the concept of “teaching” and not a practical implementation.
- To use the Python Playground application, it assumes that the robot is one designed by the company, which goes against one of the main desired characteristics mentioned above (modularity).

Even though the Python Playground is not effective enough to satisfy our desired characteristics, we found it highly useful to test the capabilities of the Robot.

ROS2 libraries + iCreate custom messages (Python or C++): In contrast to the previous alternative, the ROS2 alternative provided by the client satisfies all the desired characteristics that we look for in the project.

- It provides modular functions and libraries that can be combined with the Icreate custom messages, but do not rely entirely on them.
- After running several tests, we found that the implementation of ROS2 libraries is slightly faster than the first alternative.
- Through our research, we found many examples of both libraries and applications that deal with similar goals to the project’s goals.

Even though the ROS2 alternative could be considered harder to be set up (compared to the Python Playground), the variety of functions and libraries it provides outclass the ones from the first alternative.

3.3 Chosen approach:

After testing both alternatives intensively, we have decided that the ROS2 implementation is the most adequate alternative to support our project. While the Python Playground provided easement of access and interaction, ROS2 showed to be much more modular and effective to tackle the task at hand for this project.

Alternative	Modularity	Ease of use	Reliance on Third party services
Python Playground	Low	High	High
ROS2 + Custom Messages (C++ / Python)	High	Medium	Low

Table 3.1

3.4 Proving feasibility:

Our plan is to try several implementations of ROS2: some that implement the most basic functions available in the ROS2 libraries, and another that mixes the use of ROS2 libraries and the custom messages provided by the API of the Robot. The next problem is modularity, our code must be able to work with other robots in the future and as such we must make our program in consideration for future changes. Any code that works with the IR sensors must be configured in a way that will allow for quick and easy changes that will allow for them to be calibrated as needed for differing sensors. It is also important to note the short range of our IR sensors on our Create 3, the code must be able to work with our current sensors but be ready for improved sensors in the future.

The final problem is a technical one rather than a problem with physical sensors. The Create 3 cannot connect to wifi unless it only uses a password, something which NAU wifi does not offer. The bright side is, that when we connect using a hotspot the robot is responsive and does not seem to require too much bandwidth for its operation.

We have completed a series of tests on the aforementioned aspects of the Create 3 and have received some interesting data from them. That should help us with our implementation/creation of code. For example our IR sensors are unable to give feedback that is notable beyond about 1 foot away.

Actuators	Tile Flooring	Carpet Flooring
Expected Distance	6’	6’
Actual Distance	10’	10’

Table 3.2

Cliff Sensors	Responsivity
Touching the ground	Responding and Detecting
1 Foot above the ground	Responding but not detecting distance

Table 3.3

	Non-reflective material	Reflective material
Responsivity of IR sensors	Normal Signal Strength	Weaker Signal Strength

Table 3.4

IR Sensors	Distance of 1 Foot	Distance of 3 Feet	Distance of 5+ Feet
Sensor 1	Notable Response	No Notable Response	No Notable Response
Sensor 2	Notable Response	No Notable Response	No Notable Response
Sensor 3	Notable Response	No Notable Response	No Notable Response
Sensor 4	Notable Response	No Notable Response	No Notable Response
Sensor 5	Notable Response	No Notable Response	No Notable Response
Sensor 6	Notable Response	No Notable Response	No Notable Response
Sensor 7	Notable Response	No Notable Response	No Notable Response

Table 3.5

Rotating and Move	Accurate on Tile	Accurate on Carpet
90 Degrees	Yes	Yes
.3 Radians	Yes	Yes

Table 3.6

We conducted the above tests by creating small snippets of code that specifically affect only the tested aspect of the robot and can record the data at various times throughout the tests. These gave us accurate readings of what the robot was detecting and how it was responding to different commands. For distance and rotation measurements we used physical markers and measuring tape to ensure that our tests were accurate. From these tests we learned that we could not solely depend on the IR sensors for our robots traversal and that we would need to take advantage of the router triangulation method that a previous group attempted to ensure that its movements are accurate and that it is able to properly position itself. As such we have weighted the following on importance.

Criteria	Alternative
1. Able to navigate to a location	Digital mapping and Wifi triangulation
2. Able to avoid ledges	Use of both IR sensors and cliff sensors
3. Able to avoid obstacles	Code that takes into account the sensors range
4. Able to accurately move between locations	The use of code that takes advantage of the robots accurate rotation and accounts for its inaccuracy in movement

Table 3.7

Further testing will need to be done to ensure that we are on the right path. Our current course of action is to create more complex code that will allow us to test the parts in use together. From creating a small obstacle course for testing the actuators in combination with sensors to creating a small map that the robot can navigate to test its ability to recognize checkpoints and reorient itself. If we can get these aspects working together then the robot should be able to function in a specified environment.

4. Technological Integration

With all of the previous problems/challenges combined we have had to take a few steps back to analyze our available options and how we can combine them together into a robust and modular system. We are planning to combine the sensors on the robot with digital mapping to allow whatever robot that our code is used on to be able to navigate its surroundings with only minor modifications to the code. The plan consists of 5 interconnecting parts: the ability for the robot to follow a nearby wall to help it navigate its surroundings, the robot being able to create a digital map of its environment and follow instructions to move to different points, the ability to self localize independent of its IR sensors using router distance to triangulate its position, the use of its downward facing sensors to ensure the robot has no problems with cliffs, and the ability to avoid obstacles in its path.

The most basic task for the robot is to be able to move from point A to point B and our plan to accomplish this is for the robot to use the walls as guides for it to navigate between those two points. A problem that came up with our testing of the Create 3 is that it is made by the same company that makes the Roomba. That robot uses a mixture of IR and bumper detection to help it follow nearby walls and as such any pre-existing programs would use a similar system for following walls. We cannot use the bumpers due to the requirement that the robot does not hit anything during its navigation. As such our program must be able to follow the wall at a safe distance without losing track of it during corners or any gaps. On top of that we must be sure that our actuators and odometry are entirely accurate to mitigate any problems that may occur during transit. Something that our testing has shown to be accurate on all flooring for its turning but struggling to move the specified distance on most types of flooring. On top of this we have to be aware of the robots inbuilt coordinate system that uses quaternions and has its positive x coordinates facing forwards while the positive y is directed left.

To be able to follow walls and be sure of where it is going the robot must have a digital map that it is able to both reference and use to help track its location as it moves. We hope to make the map making process automatic and allow the robot to be able to map out its environment with minimal outside interference. This would allow for the robot to function almost anywhere if given enough time and preparation, something that will be important in the future.

However, a big problem with letting the robot control itself to map out the room is the danger of environmental hazards. That's where the cliff sensors and IR sensors will be used, they both have incredibly short range and as such it may be necessary to take advantage of both to successfully achieve our task.

The combination of all the previously mentioned aspects will create a system that is adaptable to most environments and is easy to implement.

5. Conclusion

Robots are becoming more and more common throughout the world and our lives but educational institutions are falling behind in having accessible ways for students to learn and experiment with them. As such, our goal with this project is to create a base line that allows for different robot platforms to be used by schools to help educate students on how robots function and allow for expansion upon that code. Our project will have usable features that can be experimented with from navigation through digital maps and wifi triangulation to using the sensors to navigate from one location to another without colliding with any obstacles. All the while the code will be modular and easy to implement into any future robot that would continue to use ROS2. As we continue to test how each sensor interacts with the others and continue to work on implementing the ability to connect to more secure wifi networks we will continue to improve both our knowledge and our implementation. We are aiming to not only encourage the learning process through our coding but to inspire creativity in future students that follow.