Tyler Doupe 2/27/2025

Final Project IT 209

For my project, a problem that my robot will solve is delivering lightweight objects from one location to another. Overtime, a much larger scale of this project would be to make this robot handy for the garage by making it capable of identifying various tools within a storage location and delivering the specified tool to me. However, for this course, I need to start at the building blocks of this project and that would be having the project capable of traveling to a specified tool location, gathering a lightweight object, and then transporting the specified object to a specified destination location. Along the path, there may be obstacles that alter the path on a regular basis, thus the robot must be capable of reacting to these obstacles and efficiently navigate around the obstacles to continue delivering the object to the destination location. By having a robot capable of delivering items in an efficient manner and capable of navigating objects, I can focus my energy on other tasks while the robot makes the travel distance to retrieve a light weight object for me.

When thinking through some of the goals of this project and how I could possibly take advantage of the sensors and actuators found on the Finch 2.0, a few ideas come to mind of what sensors and actuators may be best to accomplish my goal. To start, the Finch 2.0 has a distance sensor on the beak of the device. This distance sensor could be used to help the Finch avoid obstacles in the Finch’s path. By taking advantage of the distance sensor for obstacle avoidance, the Finch can successfully navigate obstacles in its path without colliding into the obstacles. Another sensor that the Finch has that I feel I can take advantage of for prototyping this design are the infrared sensors. By using the infrared sensors, my Finch can navigate along a predefined path on the ground by following the line. This line will act as the optimal path for the Finch to take when navigating between the pick-up and drop-off locations. Regarding actuators, one actuator that will be used for this project are the wheels. The Finch wheels will be used to move the Finch between locations and around obstacles. Another actuator that I will implement is the Finch’s tail lights. The Finch will have a red light turn on when the Finch loses the predetermined path that it is traveling on and when it enters a search mode. The Finch will also have a yellow tail light turn on when the Finch detects an obstacle and enters its obstacle avoidance method.

This robot-based solution is original in a few different ways. Although the Finch 2.0 is designed for educational purposes, I intend to learn how to program this robot to meet a unique real-world solution. I have found myself in a pickle a number of times as I am under a vehicle or in a tight location and wish someone was around to go grab an item for me so I can continue to work in my tight space location or wait for the item to be retrieved so I can continue my work. Often climbing into tight, small spaces is a beast of a task so having someone/something capable of retrieving an item for me could be a life saver. This is where my idea of a small robot, capable of navigating between two points while carrying a lightweight object comes into play. By having a robot that could make the trip to a specified location so I can continue working or remain in the tight space I am located in would be extremely beneficial and convenient to my work. Up to this point in my life, I have not heard of a robot that is capable of retrieving items for a user so they can continue their work on another task. For example, when working on a vehicle, it can be a pain to climb underneath the vehicle to check a bolt size, only to find out that you grabbed the wrong socket size. This then leads to you climbing back out from under the vehicle, walking to the tool location, and retrieving the tool. Or even simpler, lets say the user already had the tool but needed both hands to climb further under the vehicle and shimmy their way to the exact location under the vehicle that they are working on. In the process, they forgot their tools, or left them behind by a few feet that is just out of their reach in the new location. Having a robot that could move those tools closer to the user would be extremely convenient and time-saving!

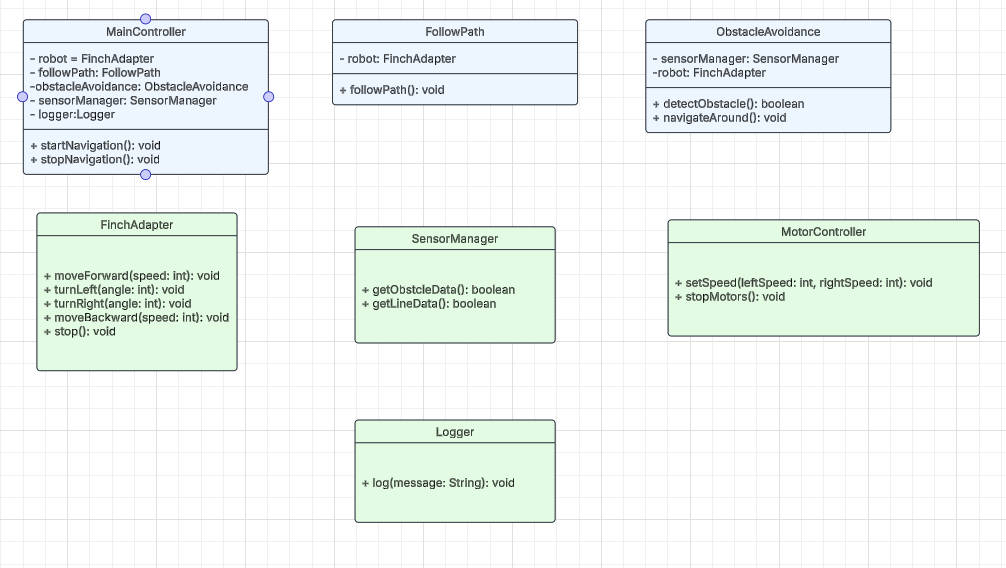
My idea for this robot relates to new trends in robotics as our current state of society is a constant race for the next greatest artificial intelligence (AI) and automated processes. Many businesses in life are creating their own AI to be used by their co-workers and there is a race for the next best automated device to help with everyday tasks. Some of the earlier forms of automated robots I found were devices like the Kitchen Aid that allowed users to mix items while they continued working on other tasks they needed to perform when baking. Then the Roomba came out and people could have their floors vacuumed and mopped without moving a finger. Even more recently I have heard about automated lawn mowers so people no longer have to run a lawn mower and can focus on other tasks in their life. It is honestly incredible to see the rate at which life tasks are becoming automated. A robot capable of retrieving items could come in handy in a number of scenes from hospitals, to libraries, to garages, etc. The concept of this robot can be highly values and thus fits the latest trend in robotics.

This robot-based solution is very basic due to the time to design this robot design as well as the sensor and actuator limitations placed on this device, however, this robot-based solution still holds many similarities and differences when compared to other robotic applications in existence today. For example, one robot application that this Finch robot-based solution could be compared to is the Roomba. The Roomba has quickly become a common household robot due to the convenience it brings its users. The Roomba has proven to be efficient at obstacle avoidance and automated navigation. Unlike my Finch, the Roomba has the capability on board to map out a room and thus can navigate the entire room vacuuming the room. My Finch does not hold the capability to withstand this sort of data to understand the room layout but it does hold line sensors that it can use to navigate along a predefined line. Although these navigation paths differ between the robotic systems, the principle of room navigation is similar between these devices. The Roomba is also much more intelligent in how it handles obstacles in its path to still achieve maximum cleaning of the room while navigating around the object. My Finch design is capable of navigating around objects and returns to its predetermined path after successful obstacle navigation but it does so in a very ridged manner. My Finch design turns to avoid the obstacle, then travels a predefined distance before turning back toward the generalized direction of the original path and then continues navigating until it senses the line. Lastly, my Finch design and the Roomba are both capable of automated navigation. The Roomba is superior in that it can handle wireless communication with a cell phone and or can run based off a predefined schedule set by its user. My Finch design starts when I press the start button but once started, my design will continue to navigate back and forth between the two set locations. The Roomba, once started, will navigate its rooms vacuuming and return to its dock when it has completed its task. Although the Roomba is far superior in how it navigates and avoids obstacles, my Finch design and the Roomba do have similar qualities. Another robotic application that my Finch can be compared to is the Amazon Kiva Robot. The Amazon Kiva robot follows QR codes on the floor in order to follow a predefined path. This quality is similar to my robot design in following a predefined path as my robot follows the black electrical tape on the floor by using its infrared sensors. The Amazon Kiva, similar to the Roomba and my Finch, is capable of automated navigation and obstacle avoidance which makes the three of these robotics similar. The Amazon Kiva is far superior in design than my Finch but shares many similar navigation principles to my Finch as it follows a predefined path via QR codes on the floor, it successfully avoids obstacles in its path then returns to the QR coded path, and the Kiva is capable of autonomous navigation once given a command. A third example of a robotic application in existence today that shares qualities with my Finch design is the Tesla Autopilot. There is a growth in automated vehicles as artificial intelligence continues to grow. Recently Tesla has released their Tesla Autopilot that uses cameras and sensors in order to adequately avoid collisions with other vehicles, remain in the proper road lane and navigate between locations. Although this is at a much more sophisticated level than my Finch design, the overall principle remains the same. My Finch avoids obstacles, navigates between two locations and uses sensors to stay on the predefined path. The Autopilot uses different, more technologically advanced sensors to perform its task but shares similar qualities and a comparable task to what my Finch is designed to perform.

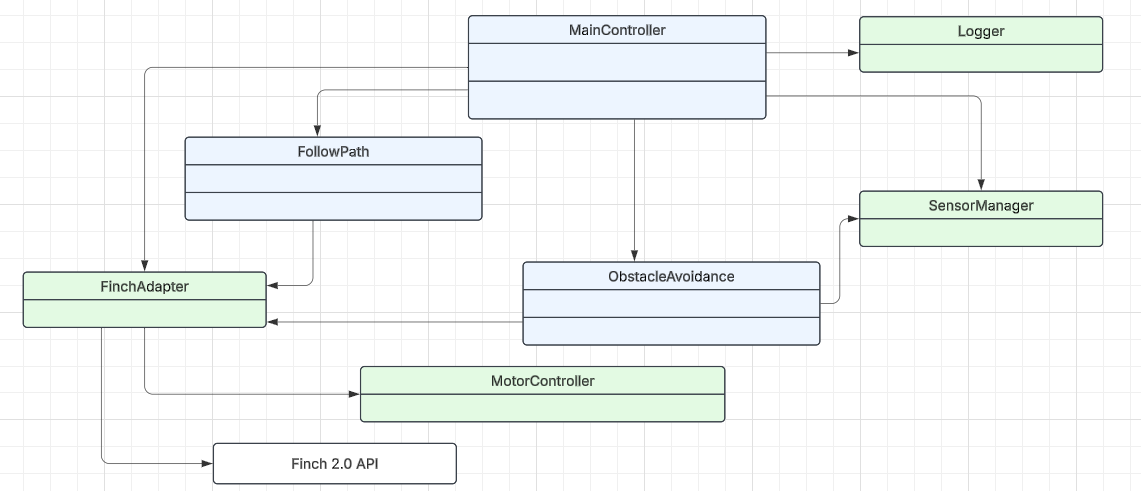
For the first critical behavior that my Finch will exhibit as part of the final project, I chose to build a prototype capable of obstacle handling. For my final project, obstacle avoidance will be a critical feature for the robot to navigate its environment between two set locations. If an obstacle is presented in the predefined path, the robot must be able to identify the obstacle using its IR sensors and effectively navigate around the obstacle. If the robot does not perform this crucial behavior, an obstacle could cause the robot to become useless and would restrict the robot from performing its designed task of delivering objects between the two set locations. Although there are some things to be tinkered with regarding this current prototype, the designed prototype is capable of detecting an object in its path and then makes a decision on a viable path to pursue to navigate around the obstacle. Upon successful navigation around the obstacle, the robot resumes on a straight path heading to show proof on concept that the robot can successfully identify an obstacle in its path and navigate around that obstacle.

For the second critical behavior, I chose to show how the robot would navigate along a pre-defined path. In order to implement this behavior for the prototype, the robot will use its line sensor to follow a single line placed on the floor. The robot will navigate left and right along the line to navigate along the line to show the proof of concept that the robot is capable of navigating along a pre-defined path. In addition to this concept, I added the ability for the robot to come to a stop upon encountering an obstacle set on the predefined path. Implementing this critical behavior shows that the robot will be capable of navigating a predefined path and come to a half upon encountering an obstacle.

1. Below is the UML class diagram that explicitly documents the design of the application I will be building for this project. This UML shows the class attributes and behaviors that each class will have for the final design.



1. Below is the UML class diagram that documents the relationships between user-developed classes.



Following the development and testing of my robot’s design there were many areas in which my design met the expected results. My robot design is capable of navigating between two locations by using its infrared sensors to remain on track with the electrical tape line on the ground. The robot is capable of following the line between these two locations and turning around to return back to the starting point after traveling down and back on this line. If an obstacle is placed in the path of the robot on this line, the robot is able to identify an obstacle in its path and successfully navigate around the object. After navigating around the object, the robot then turns back toward the line and is able to retrack the line to continue on with its navigation between the two set points. By successfully completing these behaviors, this robot design is able to meet the basic expectations of this robot design. However, with all of this being said, there are still areas for improvement with this design. Although this design is capable of returning to the predefined path, it could fall off the path. What I mean by this is that when an obstacle is identified in its path, the Finch performs a predetermined obstacle avoidance plan. This plan, although effective for straight forward obstacles in its path, it does not allow the Finch to dynamically avoid an obstacle and return to the path. Additionally with the method in place to return to the original predefined path, the robot continues at a straight line after avoiding the obstacle. If the robot’s heading was altered due to a bump in the floor or a pebble under a wheel, this could cause the robot to not return to the path and instead it would be placed into an indefinite loop searching for the line. At a basic level, this software design meets the expectations but there is a great level of detail that could be enhanced to improve on this design.

A remediation plan to address these deficiencies would start with a more sophisticated obstacle avoidance method. The current method is predefined and ridged but a more optimal obstacle avoidance method would enable the robot to identify the proper distance that the robot may need to travel in order to clear an obstacle. After traveling a distance to clear the obstacle, the robot could turn to check if this distance would in fact suffice to clear the obstacle. After determining this distance, the Finch would navigate in a manner to travel past the obstacle and then would have an improved manner of returning to the predetermined line. I currently set the Finch to return to the line on an angle so when the line was found, the Finch would be in a direction already that it would realign to the path and continue on its previous trajectory. An improved manner of correcting this behavior may include traveling in the general direction of the line and when the line was detected, the Finch could make use of the compass to return to a heading that was previous in place prior to performing obstacle avoidance.

When assessing my robot’s design, it is clear that my robot’s design implements a subsumption architecture. When designing my robot, I wanted to prioritize obstacle avoidance thus obstacle avoidance was set to the highest-level behavior in the design. Regardless of what the robot is doing, if an obstacle is detected with its distance sensors, the robot will switch to performing its obstacle avoidance method. After the obstacle has successfully been avoided, the robot returns to performing lower-level behaviors. The middle-level behavior in my robot design is the behavior to search for a line. If the line is lost, the robot will not know where to navigate and thus this behavior is set to the middle-level behavior. This allows the robot to begin searching for the line and ultimately turn around at the end of the predefined path to return the other direction along the line. Lastly, the line following behavior is set at the lowest priority. The robot executes line following until an obstacle is detected or the line is no longer detected. If either the line searching behavior or the obstacle avoidance behavior are in effect then the line following behavior is stopped until either of those behaviors are no longer active. This subsumption architecture allows my robot to perform its desired task efficiently and effectively while properly managing the outlined priorities. This also removes the need for complex decision making by my robot as priorities are clearly defined in advance for the robot. Regarding simulation, I did not create any unit tests or anything to that nature when designing this robot design. Instead, I performed numerous trial and error attempts with the Finch. After every respective change to the robot design, I tested that the robot performed how I expected and if it did not, I reworked the code in order to tailor the Finch to operate in an expected manner. This manner of simulation and testing allowed me to see how the behavior prioritization would take effect in a real-world environment. I also was able to see how the robot would operate in the real world thus enabling me to better understand potential errors the robot may experience. For example, when testing the robot for the first few times with the obstacle avoidance behavior, I realized the impact my tile floor in the basement was having on the corrections the robot was taking to return to the line. In several instances, the robot was unable to return to the line because the bumps between tiles disrupted the path the Finch was taking and thus the Finch would become stuck in a loop searching for the line again. This was a great test for the robot as a virtual test environment would likely not be able to simulate such a behavior.

Overall, this project was eye-opening to see the capabilities a small device with only a few sensors could have when working on robot behaviors. It was extremely rewarding to build a project of my desire from start to finish and ultimately create a prototype of an original robot-based solution. Although there are numerous areas for improvement with this design, this design is effective in accomplishing its assigned task and serves as an excellent prototype of a robot-based solution that could be improved on in the future. Improvement to the code, improved sensors, improved data processing and improved actuators could all aid in contributing to a more refined robot-based solution in the future!

References:

BirdBrain Technologies. Finch robot: Programming with Snap! https://learn.birdbraintechnologies.com/finch/snap/program/3-1

BirdBrain Technologies. (2019, December 10). Finch 2.0: Getting started with Snap! YouTube. https://www.youtube.com/watch?v=i9R913Oj-1w

Jameco Electronics. How it works: Tesla Autopilot self-driving automobile technology https://www.jameco.com/Jameco/workshop/Howitworks/how-it-works-tesla-autopilot-self-driving-automobile-technology.html#:~:text=The%20radar%20and%20forward%2Dfacing,which%20both%20cars%20are%20traveling

Lucid Software Inc. Lucidchart [Online diagramming tool]. https://www.lucidchart.com

Microsoft. Create a UML class diagram. Microsoft Support. https://support.microsoft.com/en-us/office/create-a-uml-class-diagram-de6be927-8a7b-4a79-ae63-90da8f1a8a6b

Sparx Systems. (2016, March 7). Logical model tutorial. https://web.archive.org/web/20160307174708/https://sparxsystems.com/resources/tutorial/logical\_model.html