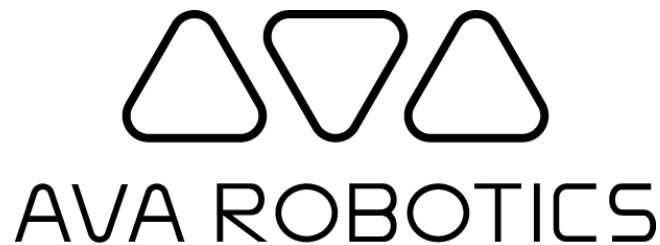




Robot Tour Guide

In collaboration with Ava Robotics



A Major Qualifying Project Submitted to the Faculty of WORCESTER POLYTECHNIC INSTITUTE in partial fulfillment of the requirements for the Degree of Bachelor of Science by:

Henry Dunphy, Zoraver Kang, and William Mosby
April 2019

Submitted to
Professor Jing Xiao, Worcester Polytechnic Institute
Professor Gregory Fischer, Worcester Polytechnic Institute

This report represents the work of three WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review. For more information about the projects program at WPI, please see: <http://www.wpi.edu/Academics/Projects>.

Table of Contents

1.1 Human-Robot Interaction.....	1
1.2 Robot Tour Guides.....	3
1.2.1 Pre-2000s Tour Guide Robots.....	3
1.2.2 Modern Tour Guide Robots.....	3
1.3 Ava Robotics.....	4
2. Methodology.....	5
2.1 Goals.....	5
2.2 User Experience.....	5
2.2.1 User Interface.....	6
2.2.2 Embedded Computer.....	8
2.2.2.1 Security.....	8
2.2.2.2 Power.....	8
2.2.2.3 Kiosk Mode.....	9
2.2.2.4 Networking.....	9
2.2.3 Touch Screen.....	10
2.2.4 Speech Recognition.....	10
2.2.5 Vision.....	10
2.2.5.1 OpenCV.....	11
2.2.5.2 Darknet.....	11
2.2.6 Mount.....	11
2.3 Features.....	13
2.3.1 Drive to Destination.....	14
2.3.2 Tour of Multiple Destinations.....	14
2.3.3 Multi-Floor Traversing.....	14
2.3.4 Obstacle Handling.....	15
2.4 Code Structure.....	16
2.5 Testing Process.....	17

3 Results.....	18
3.1 User Interface.....	18
3.2 Features.....	18
3.2.1 Drive to Destination.....	18
3.2.2 Tour of Multiple Destinations.....	18
3.2.3 Multi-Floor Traversing.....	19
3.2.4 Speech Recognition.....	19
3.2.5 Object Handling.....	19
4 Discussion and Conclusion.....	21
4.1 Conclusion.....	21
4.2 Future Work.....	21
4.2.1 Testing.....	22
4.2.2 Improve Tour Guide.....	22
4.2.3 Improve Speech Recognition.....	22
4.2.4 Elevators.....	23
Appendix.....	24
Appendix A: Python Libraries.....	24
Appendix B: Cost Analysis.....	24
Appendix C: User Interface Images.....	25
Appendix D: User Manual.....	25
Appendix E: Tour Sequence Planning Code.....	29
Appendix F: Tour Guide Sequence Diagram.....	29
Appendix G: Image Recognition.....	30
Appendix H: Connect to Ava Base Platform.....	30
Set up.....	30
Editing Wi-Fi.....	33
Citations.....	34

Table of Figures

Figure 1 - Home page.....	6
Figure 2 - PAT statuses.....	7
Figure 3 - User Interface.....	7
Figure 4 - Overview of Network Configuration and Usage.....	9
Figure 5 - Render of the Frame for the Touchscreen.....	12
Figure 6 - Tab-Slot-Nut-Bolt Construction Strategy.....	12
Figure 7 - Mount Assembly on PAT.....	13
Figure 8 - Render of Mount Assembly.....	13
Figure 9 - Multiple Floor Traversal.....	15
Figure 10 - Code Structure.....	16
Figure 11 - User Interface with Callouts.....	25
Figure 12 - Ava API Home Page.....	26
Figure 13 - Ava API Database Page.....	26
Figure 14 - Ava API Mapping Internals Page.....	27
Figure 15 - Tour Guide Sequence Diagram.....	29
Figure 16 - Chair Classification.....	30
Figure 17 - Person Classification.....	30
Figure 18 - Service Compartment.....	31
Figure 19 - Adapter Settings.....	31
Figure 20 - Local Area Connection.....	31
Figure 21 - Local Area Connection Properties.....	32
Figure 22 - IPv4 Properties.....	32
Figure 23 - wpa-pak Changes.....	33

Acknowledgements

This research was supported by Worcester Polytechnic Institute and Ava Robotics. We would like to specifically thank Professor Jing Xiao and Professor Greg Fischer. We would also like to thank Banuprathap Anandan, Sienna Mayer, Gopal Paripally, and Amit Patel from Ava Robotics for providing assistance and sharing their experience throughout this project.

Abstract

Our goal was to create a friendly robot that could safely guide a user through a building. On our way towards creating an end product, we explored a number of fields including human-robot interaction and robot navigation. Thanks to Ava Robotics, we used one of their first-generation drive bases as the core of our robot, PAT. We mounted a “head,” which contains an NVIDIA Jetson TX2 module, a camera, and a touch screen for user interaction. PAT can efficiently take users through a set of destinations specified by voice commands or its graphical interface. If PAT is stuck, it can detect obstacles and ask for help to remove an obstacle in order to continue guiding the user. We have produced a basic tour guide system using Ava Robotics’ drive base, which created a stepping stone for future development.

1. Introduction and Background

Robots are increasingly being used in the workforce. According to Forbes, North America saw a 7% increase of robots being shipped, raising the total number from 33,530 robots to 35,880 robots (Bach 2019). People are finding new jobs for robots at such an alarming rate, to the point where a new study from the global consultancy McKinsey predicts that one-third of the United States workforce could be displaced (Paquette 2017). These factors may be a cause for concern, but they can also be a reason to rejoice. While robots may replace jobs, they will also create new opportunities for the humans that they replace. They will also allow for the humans they work with to produce many times more than what a human alone could produce (Qureshi 2014).

However, with all these new robots entering the workforce it is important to understand the effect they have on humans. This effect on humans must be carefully considered when designing robots. On the surface, this may seem intuitive, but a lot of research has already begun on how best to approach these design choices. Human-Robot Interaction, HRI, is a new field that has been gaining steam over the last few decades and there is much to be said about robots working cohesively with their human counterparts.

An interesting example of HRI is robotic tour guides. These robots offer a unique way to experiment with different theories developed in the HRI field. There are two challenges the tour guide must be able to accomplish to be an efficient tour guide. The robot first must be able to interact with its users either through a user interface or through verbal communication. Secondly, the robot must be able to navigate in an area populated by humans that can be the cause of many random, unpredictable events. Together these conditions offer a way to test different cognitive and physical interactions between humans and robots.

For our project, we worked with Ava Robotics who supplied us with the base of their Ava 500 telepresence robot. The robot comes with fully functioning object detection and avoidance capabilities as well as support for navigation between two points. We used the features it came with, wrapped a python program around the robot's API, and attached a head to allow it to interact with users in order to turn it into our tour guide robot named, PAT (Prescott Ava Tourbot).

1.1 Human-Robot Interaction

According to humanrobotinteraction.org, "Human-Robot Interaction (HRI) is a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans" (Kanda 2012). HRI has seen an increasing amount of attention along with the rise of autonomous robots. The field currently faces many challenging problems, but it has the potential to positively impact society. HRI is a broad topic and covers many other disciplines, therefore it is hard to generalize (Kanda 2012).

HRI as a field did not gain much interest until a huge technological advancement led to the development of autonomous robots in the 1980s. In contrast to earlier robots that required detailed information about its current environment, autonomous robots used, “Distributed sense-response loops to generate appropriate responses to external stimuli” (Kanda 2012). Autonomous robots were more robust when dealing with changing environments since they were reacting to their sensor data. This changed allowed NASA to adopt autonomous robots for mapping and exploring space which gave more exposure to the field of robotics and in turn, HRI. Scientific meetings such as the IEEE International Symposium on Robot & Human Interactive Communication were key in spreading knowledge about robots. In 2001 the US government funded a workshop on HRI which some consider being the official event that gave way to HRI as its own field. The early 2000s also led to robotic competitions such as AAI Robotic Competition and Exhibition or the Robocop Search and Research Competition. These events also help gain notoriety for the field (Kanda 2012). Seeing as HRI is such a new field, there is not much material on it currently available. Many articles try to make sense of the field itself, but as of now, there are no agreed-upon standards.

According to Bo Xing and Tshilidzi Marwala in their book, *Introduction to Human-Robot Interaction*, the study of Human-Robot Interaction can be split up into two subcategories: physical versus cognitive interaction (Xing 2018). Physical HRI is the study of how robots move throughout their environment and how they sense, react, and plan their actions keeping humans in mind. Robots in industry or service-oriented robots are an example of robots that are studied for their physical interactions with humans (Xing 2018). When looking at physical interactions, human safety must be the top priority, leading to a need for human-friendly designs. For example, human-friendly designs could entail lightweight materials with a slender structure that includes many sensors and flexible actuators. When in physical proximity to a human the robots must take extra care due to the unpredictability of human movements. Physical HRI can also be split into three subdivisions: coordinated, collaborated, and cooperated interaction depending on how much involvement a human has with the robot's task (Xing 2018). Coordinated interaction occurs when a robot assists a human in order to optimize a task. When a human and robot work side by side on the same task, but the robot is handling all the unfavorable duties, this is collaborated interaction. Cooperated interaction occurs when a robot does not rely on a human and can perform entire tasks on its own. On the other hand, cognitive HRI is the study of how humans and robots can combine their cognitive capabilities. The goal of cognitive HRI is to understand how humans perceive robots and allow the robot to anticipate this perception. The robot should be able to react in a familiar way that does not surprise humans but allows them to work together seamlessly. These two subcategories help to organize the broad category that is HRI (Xing 2018).

Human-Robot Interaction is still young, so far researchers have not been able to agree upon any set of standard methodologies for studying the field. The field itself is such a wide space that often one study's conclusion does not fit with another's. It is made especially difficult by the fact that few of the official studies have given reproducible results, which is important

when proving a claim (Dautenhahn 2007). As more work is done in this area, it will lead to more answers, not only about Robots but about humans and intelligence itself.

1.2 Robot Tour Guides

Robotic tour guides have been in use since the late 1990s. In 1998, the Smithsonian briefly used a robotic tour guide named Minerva. Since then, robotic tour guides have become an interesting academic challenge regarding navigation, reliability, and user interaction. A robotic tour guide is also a spectacle and many robotics companies use this task to demo or showcase their new robot. While robot tour guides cannot yet replace their human counterparts, they allow researchers to test human-robot interaction theories and provide a unique attraction that can draw interested parties to a location.

1.2.1 Pre-2000s Tour Guide Robots

There are two robots that are seen as the predecessors of the entire field of robotic tour guides: Minerva and RHINO. Both of these robots were academic projects. Minerva was developed by a joint team of the Carnegie Mellon University and the University of Bonn in 1998. The purpose was for Minerva to tour groups around the Smithsonian while verbally communicate with the crowd. RHINO is another tour guide robot that was developed around the same time as Minerva; it also created by the University of Bonn. However, while Moravia communicates using speech, RHINO used a screen to take user inputs. Another difference between the two is how they learn their environments. RHINO needed to be given a map of the environment, Minerva, on the other hand, creates the map. Like most other robotic tour guides, we were heavily inspired by both. Ava uses a similar mapping and our obstacle response systems was heavily inspired by Minerva. We use both verbal communication and a touchscreen to interact with our user (Al-Wazzan, 2016 | Thrun, 1999).

1.2.2 Modern Tour Guide Robots

The main difference between the robotic tour guides of the late 1990s and early 2000s and their modern equivalents are the people building them. The early examples of modern tour guide robots were built by academics as a way to experiment with HRI. The new generation of robots are used as a showcase for new technology by the companies that built them. The two best examples of this are Tawabo and the Toyota Tour Robot. Tawabo was one of the first tour guide robots we looked at that had a digital face on a screen. Because of this Tawabo can display other information on its screen, instead of relying on its voice to communicate. We emulated this because we believed that a digital face is a great way to make people feel comfortable around PAT. The Toyota Tour Robot was the newest tour-guide robot that we studied. It was developed by Toyota to show visitors around the Toyota Kaikan Exhibition Hall in Toyota City, Japan. It has arms that can interact with the environment, and make human-like gestures. Another feature of the Toyota was that it could recognize name tags and give specialized instructions. The

Toyota robot can be seen as the next level of tour guide robots, combining the jobs of both tour guides and receptionists (Al-Wazzan, 2016 | López, 2013 | Mogg, 2012).

1.3 Ava Robotics

For our tour guide, we use the base of an Ava 500 telepresence robot that was provided to us by Ava robotics. Ava Robotics is a spin-off of iRobot that was started in 2016. IRobot created the first Ava robot called the Ava 500 in 2013. When Ava spun off they released the new Ava robot simply called AVA. The new model had improved object detection and a new top monitor. The functionality of a robot designed for telepresence is similar to the functionality of a tour guide robot. Both involve path finding and object-detection, however, the user of a telepresence robot will have more controls over the exact destination and how to get there (ex. speed and route). The user of a tour guide robot, on the other hand, will have much less control and knowledge of the system. Ava has provided us with support on integrating our software with their web API and connection out mount to their base (Ackerman, 2018).

Working with Ava robotics, they allowed us to use an Ava 500 robotic base which can autonomously navigate a mapped environment safely. The robot uses omnidirectional wheels to move in any direction without rotating or rotate in place about its center. When deployed, using Ava robotics' web-based API you can map out an environment and save locations as a tag which the robot will then use to path plan. The robot has extremely effective obstacle detection and obstacle avoidance while moving. This is due to all the sensors housed in the base. The Ava Base has a planar LIDAR sensor, an array of 3D sensors, motor encoders, and an inertial measurement unit. Using these sensors and its given map, the robot can plan a path in real time around almost any obstacle. These details are all very important for a robot tour guide since it will be operating in a place with a lot of humans and other unpredictable activity. The robot has to be able to adapt to its changing environment and safely arrive at its destination.

Along with the robot, Ava robotics supplied us with a Web Services Application Program Interface used to communicate with the Ava Base robot. This API is a 'RESTful' API that sends URI's to the robot as a way to send commands and interact with the robot. A RESTful API means, "Among other things, that only one URI is published, and the rest of the service URIs are discovered dynamically by client code at runtime" (*Quick Tip for AVA500 Platform*).

2. Methodology

2.1 Goals

The goal of our project was to create an easy to use, friendly robotic tour guide capable of taking a user from a starting point to one or more destinations spanning multiple floors within a building.

To accomplish this goal, the following objectives must be met:

1. Simple user experience
2. Speech recognition capabilities
3. Drive to a destination
4. Take a tour of multiple destinations
5. Multi-floor traversing
6. Ability to interact with the user to get out of trouble

2.2 User Experience

For our user experience, we wanted PAT to function seamlessly in a human environment. We wanted any interaction with PAT to be intuitive and the easiest way to make a robot familiar is to give it human like qualities. We decided that PAT should have a 'head' that would include the senses of sight and sound so that PAT could act in a similar manner to its human tour guide counter parts.

The first sense we wanted to implement was PAT's face. We considered adding physical fixtures that would emulate eyes and a smile, but decided it would be beneficial to be able to show PAT's status through human-like emotions. We wanted a screen that would display a cartoonish face and could change depending on PAT's current state. It also made sense that users could interact with this screen to command PAT. With this in mind, we settled on purchasing a touch screen due to the simplicity and the fact that PAT would not come with a mouse or keyboard.

When working with a human tour guide you communicate with speech, this is something we also wanted to implement into our 'head' design. In order to communicate verbally, PAT would require ears and a mouth. We found a cheap webcam that already came with a microphone which could act as both the eyes and ears for PAT. To give PAT a mouth and a voice we looked for a small robust speaker that had a 3.5mm audio aux input and Bluetooth so it could be easily integrated. Finally with all of our senses accounted for we need a 'brain' to be gather information from each sensor and to run our software. As discussed later in section 2.2.2 we chose the NVidia Jetson TX2 board which would fit all of our software requirements and even came with its own camera that would act as the eyes of our head model.

2.2.1 User Interface

For our user interface, we wanted a minimalistic design that was intuitive as well as easy to use. We also had to keep in mind that most of the development from would be on computers, but the final product would not have a mouse and keyboard instead would use a touch screen. This led to a design choice where we tried to only have a few buttons or other interactive elements per page. The user should be able to immediately understand what each scene is intended for and be able to select an element without causing something unintentionally.



Figure 1 - Home page

The home page is used to start the application and is the return point after an action is completed. The cartoon face was added to give PAT a little life and it will change emotions depending upon PAT's status. As seen in this figure, PAT is happy and therefore everything is all good. When PAT has a sad face, this means PAT got stuck and could not arrive at its destination. When an error occurs within the software, PAT will display a confused face. All of these statuses can be seen in Figure 2 below.

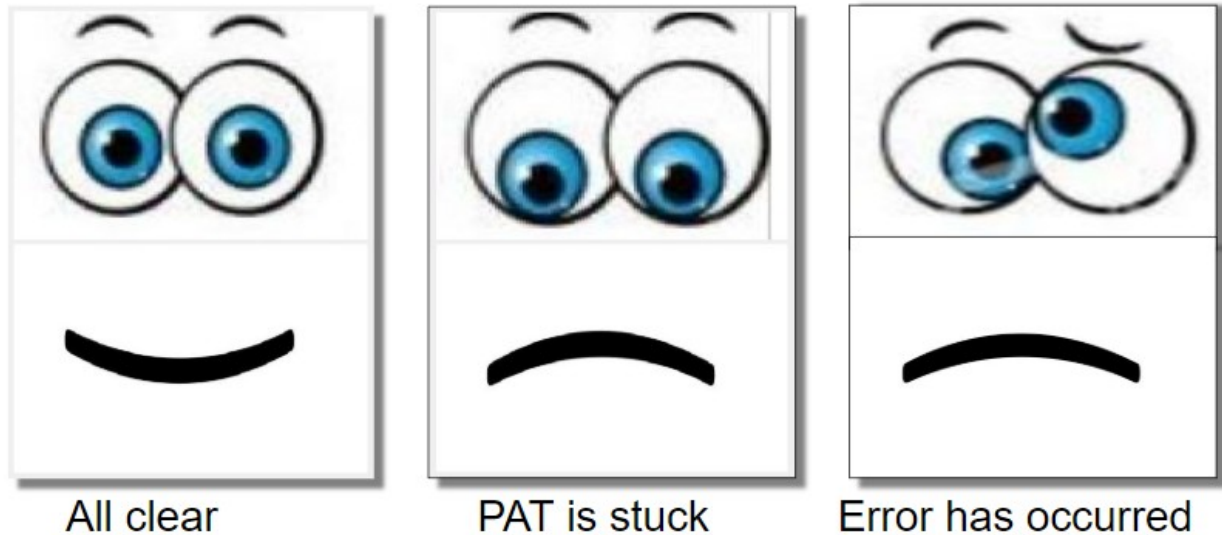


Figure 2 - PAT statuses

Pressing start on the homepage will bring the user to the navigation screen as seen below in . The top white bar is a search bar that allows the user to search for a specific destination. To the right of that is the speech button that allows the user to give verbal commands to PAT. The large buttons in the middle are some of the available destinations for the current floor. Pressing one of these buttons will highlight it in green. After the user is done adding destinations, pressing the 'GO' button will send PAT to all of the selected destinations. If there are more than one selected, PAT will go on a tour of those destinations. The user can use the dropdown button to switch between the different available floors. The next and previous buttons rotate between the available destinations on the current floor. Finally, the home button takes the user back to the start page. Another screenshot with all of the buttons labeled can be found in Appendix C.

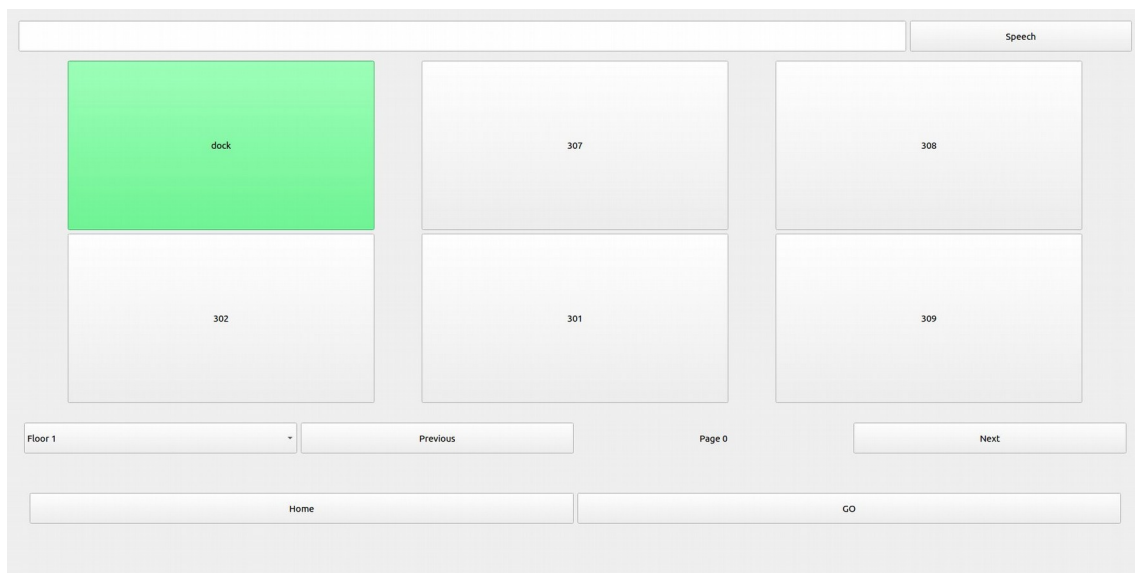


Figure 3 - User Interface

2.2.2 Embedded Computer

At the core of PAT's head is an embedded computer that serves as its "brain," running our tour guide software and all of its dependencies. Our software is written in Python 3 and is designed to run on Linux. It depends on PyQt5 to create a user interface and Darknet for vision processing. Darknet is optimized to be run using CUDA, so one requirement for our robot controller is that it needs to have NVidia graphics onboard. Once this requirement was established, we had a few options to choose from. The maximum power that the Ava 500 can provide to peripherals connected to the base interface is 38 volts * 10 amps = 380 watts, but we aimed to use as little power as possible to maximize the battery life of our system. Our options for a control board with NVidia graphics were the NVidia Jetson TX2, which costs \$300.00 with a student discount and draws up to 21 W in the worst case (12 to 15 W on average) and building a small microITX PC with an NVidia GPU to mount to the base. Of these choices, we selected the TX2 due to its low cost and low power consumption.

2.2.2.1 Security

Since PAT will be operating in a public space and be exposed to the public internet, it is critical that its attack surface is reduced as much as possible. We focused on hardening the SSH server running on the TX2 and the TX2's USB configuration in an effort to prevent the system from being hijacked.

For debugging and remote management, an SSH server was installed on the TX2. If poorly configured, SSH can be a relatively easy entry point for a remote attacker. By default, SSH uses password authentication without rate limiting, which can be brute-forced by a remote attacker, given enough time. In order to mitigate this, several steps were taken.

In order to disable password authentication and root user login, the following lines were added (existing and contradictory lines were modified as needed) to `/etc/ssh/sshd_config`:

- `PermitRootLogin no`
- `PubkeyAuthentication yes`
- `PasswordAuthentication no`

Additionally, Fail2Ban was installed and configured to limit the max number of SSH connection attempts to 5 attempts per every ten minutes. This, combined with strong public-key authentication significantly reduces the chances of the TX2 of being compromised over SSH.

Completely protecting our system against compromise by a local attacker is not possible, but we made it significantly harder by configuring USBGuard to prevent the TX2 from allowing unauthorized USB devices to be used. Only the touch screen and the reverse webcam are whitelisted; all other USB devices are ignored. This means that compromising the system is not as simple as plugging in a USB rubber ducky with a malicious payload.

2.2.2.2 Power

In order to power the TX2, we had to utilize the power port located at the top of the Ava 500 drive base. This power supply provides unregulated battery voltage from 24 to 38 volts and

up to 10 amps of current. On the other hand, the TX2 board only requires a voltage between 5.5 volts and 19.76 volts. The wall adapter included with the development kit is rated to output 19 volts and 4.74 amps. After discussing this problem with Amit Patel from Ava Robotics, we decided to use a step-down converter rated to convert 36 volts to 19 volts to power the TX2.

2.2.2.3 Kiosk Mode

Currently, the TX2 on PAT is configured to boot directly into our tour guide user interface. In order to do this, we created a custom xsession by placing a file in the `/usr/share/xsessions/` directory. This file specified that the `tg_start` script from our project would provide a graphical user interface on login. The `tg_start` script is a simple bash script that starts the python virtual environment for our project, launches the user interface, and re-launches it if it closes. We also edited `/etc/lightdm/lightdm.conf` to configure the TX2 to automatically login to our custom xsession on boot.

2.2.2.4 Networking

The TX2 needed to be connected to both the global internet to enable cloud-based speech to text and the Ava 500 local area network (LAN) to allow the TX2 to control the Ava 500 via the Ava Web Services API. In order to do this, the TX2 was configured to connect to the WPI-Wireless network over Wi-Fi and to the Ava 500 LAN via Ethernet. Since the Ava 500 LAN did not have a DHCP server, the TX2 was configured with a static IP address (172.18.0.10) for Ethernet on the same subnet as the Ava 500, which has an IP address of 172.18.0.1, by a file placed in the `/etc/network/interfaces.d/` directory. A visual overview of the networking configuration is shown in Figure 4.

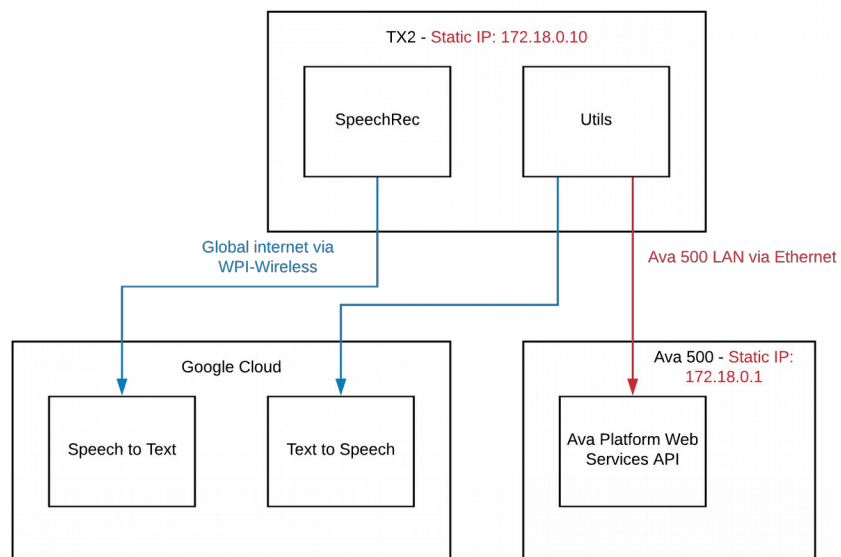


Figure 4 - Overview of Network Configuration and Usage

2.2.3 Touch Screen

A touch screen serves as the front of PAT's head and can display a face that represents PAT's status. Additionally, the touch screen allows users to control PAT through a graphical user interface. Since the screen is mounted on the Ava 500 base, we want to be able to power it over USB using the TX2 rather than adding an additional power supply. It also needs to be supported by Linux. One display that met all of our requirements was the 10" universal touch display from Chalkboard Electronics. This display provided support for 10-finger multi-touch, HDMI input, and HDMI audio breakout, which allowed us to connect the speaker directly to the display.

2.2.4 Speech Recognition

We use speech recognition as an alternate way to take user input. We believe that one of the advantages to this is that it provides the user with more flexibility on how they interact with PAT. Almost every tour robot that we drew inspiration from had some form of speech recognition because it brings our robot one step closer to a human tour guide. To achieve this we use the Python Speech Recognition API. We choose this over using Google's or Amazon's APIs because it gave us increased choices on what engine we would use. This allows us to make changes without impacting other classes. To activate the microphone the user presses the 'Speech' button as seen in , PAT will listen for the user's voice command. After that, we run the user's command through the web API. We then parse the output from the API and compare it against a list of actions ('take', 'directions', 'where', and 'drive') and save the first one that we find. After that, we compare the output against a list of the names from all the nodes from the current database and save all the ones that match. We then run the appropriate action using the saved locations as input. If there is more than one location saved, PAT will start a tour, instead of driving to a single location.

2.2.5 Vision

PAT's vision system serves as its eyes, allowing it to identify people and potential obstacles. We investigated two main approaches for the vision subsystem: OpenCV and Darknet YOLO. Our goals for the system were for it to be able to detect humans, detect a variety of common obstacles, such as chairs, and be able to process more than one frame per second to allow PAT to react to changes in the environment reasonably quickly.

PAT has two cameras, one directly mounted to the TX2 which faces forward and a USB webcam that faces backward, that provide the unprocessed image data for the vision system. The forward-facing camera is used to take pictures of obstacles to classify them, and the rear facing camera was going to be used to track users following PAT to make sure that they don't get left behind, but we were unable to finish this feature due to time constraints.

2.2.5.1 OpenCV

Our first approach to image recognition was based on the venerable OpenCV library. It utilized the included support vector machine detector for human detection and the included cascade classifier for face detection. Although it was able to run at around 20 frames per second, it had trouble detecting humans if their entire body was not fully in the frame, which could be problematic depending on the position of the human relative to the camera mounted on PAT.

2.2.5.2 Darknet

We settled on using Yolo v3 created by Joseph Redmon for our image recognition. This allows us to have real-time image recognition, which we use to detect what is blocking PAT. We take the output from Yolo and cross check it with a list of common objects. If the obstacle is not human, then PAT will ask if for the object to be moved. If the obstacle is human then we ask the human to move. Yolo works by applying a single neural network to the whole image and divided it into regions and predicts bounding boxes and probabilities of the objects in the image. The advantage of this approach is that it is much quicker without sacrificing accuracy. Yolo is also optimized for CUDA, allowing us to run it on our TX2 locally at around 2 to 3 frames per second (Redmon. N.d). Examples of the output of YOLO can be found in Appendix G.

2.2.6 Mount

In order to package the TX2, the touchscreen, the webcam, and the speaker together into a unified head, we designed a mount. We wanted to keep the design as modular as possible so that additional components could be added without hassle. We also wanted to have the emergency stop button easy to access and to leave enough space for wire routing.

To make fabrication easy, we decided to construct the mount out of 0.25” thick Baltic birch plywood, which can be laser cut using one of the several laser cutters on campus. Every piece, except for one, was cut on the laser cutter in Washburn Labs on the WPI campus. The one exception, the frame for the touchscreen, was fabricated using a CNC router because it required pockets of multiple different depths to support the screen properly. A CAD rendering of the frame for the touchscreen can be seen in Figure 5.

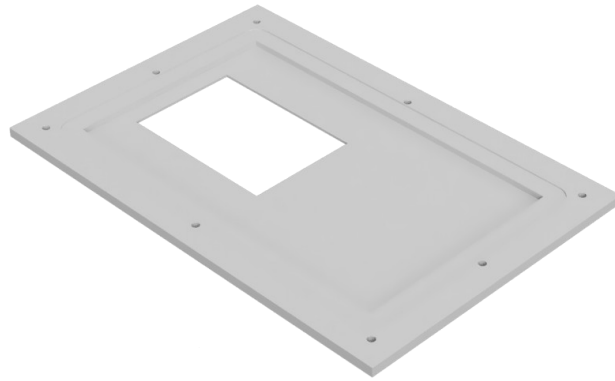


Figure 5 - Render of the Frame for the Touchscreen

Each laser cut piece was designed to fit together using a “tab-slot-nut-bolt” strategy, as shown in Figure 6. In this strategy, the tabs on piece A insert into the slots in piece B, allowing the locknuts held by piece A to align with the bolts inserted through piece B. This strategy worked well for us, but we did have to use a calibration piece to calibrate the friction fit for the locknuts when laser cutting.



Figure 6 - Tab-Slot-Nut-Bolt Construction Strategy

The final design met our requirements well. The modular nature of the design came in handy, as the speaker and rear webcam were selected and added to the design after the majority of the mount had been fabricated and assembled. A CAD render of the mount is shown in Figure 8, and a picture of the assembled mount attached to the Ava 500 base is shown in Figure 7.



Figure 7 - Mount Assembly on PAT

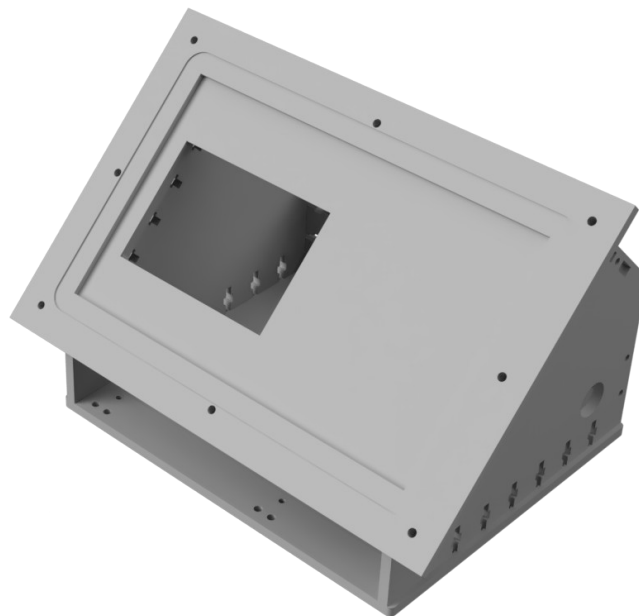


Figure 8 - Render of Mount Assembly

2.3 Features

The main function of a tour guide robot is to navigate the user to a destination or destinations. Driving was the first feature we implemented, and thanks to Ava Robotics' design it was a seamless addition. The first requirement to implement our features is PAT needs a map database to be created from Ava's web-based API. The area PAT operated in would have to be mapped out and all destinations given a tag. The process for this can be found in Appendix __. Using the map database, we were able to program PAT to drive to one destination or to drive to multiple different destinations as a tour of an area. Next, we added support for traveling between

map databases. Finally, we added another layer in our program to detect when PAT is stuck and what to do when no path is available.

2.3.1 Drive to Destination

The first and simplest feature was to have PAT drive to a destination. Ava Robotics had already given us Python code that was capable of sending a robot to a tagged destination on a map database. We just had to add a front end user interface to their program and stored the dictionary of tags instead of querying PAT every single time we wanted to drive somewhere.

2.3.2 Tour of Multiple Destinations

The tour guide feature is the backbone of our project; the system that all other systems will interact with. We first load all the points from PAT's database, if multi-floor is enabled it will load them as well. The user will then select their desired locations. We take these locations then and separate them by floor. We then run a greedy path finding algorithm to find the shortest path for each floor (see Appendix E). We then stitch these plans together with the portals that lead to the other floors. At each location, PAT had the ability to stop and give a brief description of the location. The problem with the descriptions are that we implemented the ability for the user to input the definitions. All the current descriptions are hard coded.

2.3.3 Multi-Floor Traversing

In order to add support for multiple floors, we had to refactor our code. The Floor class was added which on initialization creates a Floor object for each mapping database. The floor object stores its map database, all accessible destinations, and any portal destinations with their coordinates. Portal destinations are tags in the map database that overlap with another map database, these tags have the attribute 'Portal' and its value is the map database that it overlaps with. When PAT needs to travel to a destination that is not in its current map, it will look for a portal destination that can overlap with the map database that contains the destination PAT is looking for. Once that portal's location is determined, PAT will drive to that portal then switch its internal map database and re-localize itself using the stored coordinates and then continue on to the final destination. We were never able to fully test this with an elevator between floors since PAT needs an internet connection for some of its features. Figure 9 is an image to help visualize how the multiple floor traversal works.

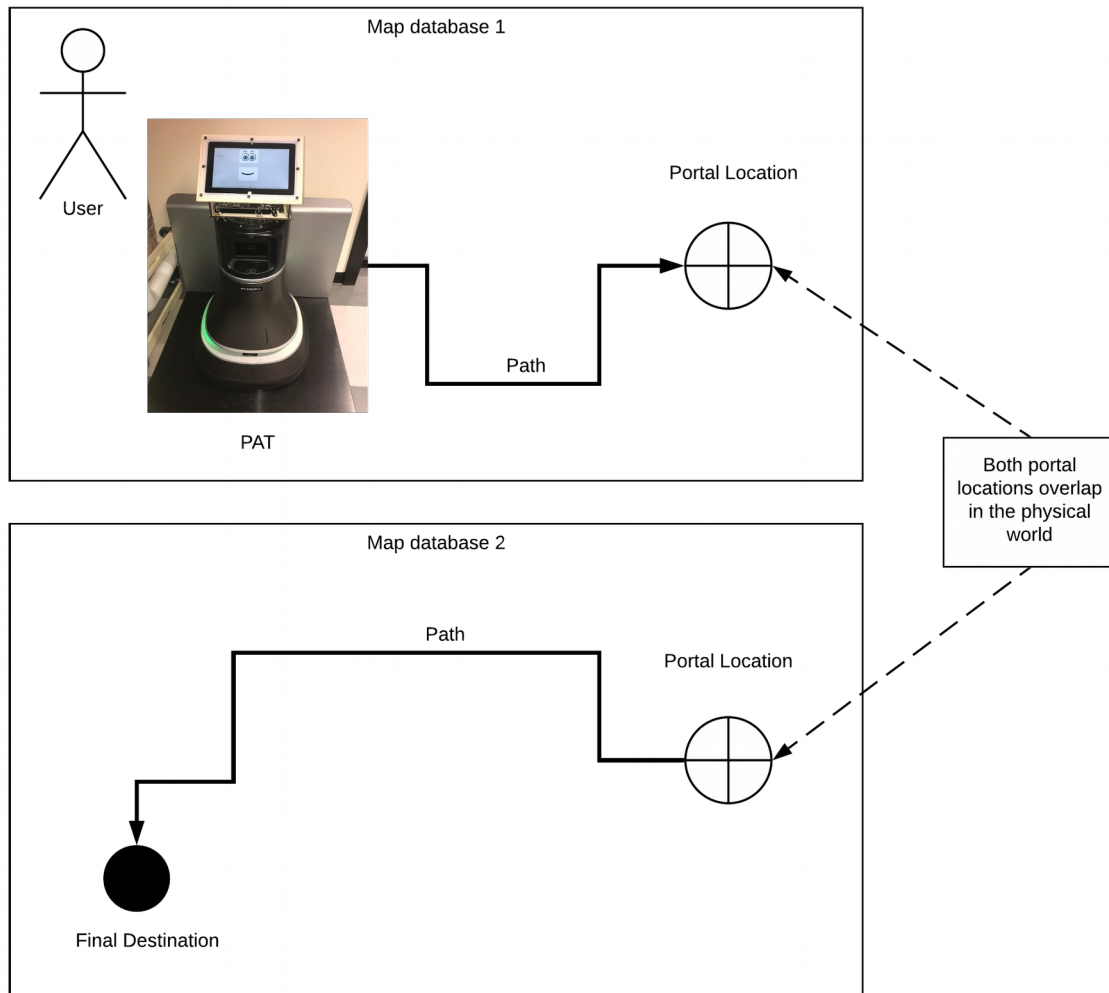


Figure 9 - Multiple Floor Traversal

2.3.4 Obstacle Handling

After some initial testing, we noticed that PAT did not have a very good response to getting stuck. If there was no available path, PAT would fidget back and forth waiting for an obstacle to be moved so it could continue. While this behavior might be acceptable for a telepresence robot, it would make more sense if PAT stopped moving and waited for the user or another person to move a said object out of the way. We added in this feature by checking PAT's status periodically for the status, "waiting for obstacles to be moved". If PAT ever encountered a situation where there was no available path and the status was waiting for obstacles to be moved three times in a row, PAT would stop and ask its user for assistance. This allowed us to come to detect much earlier if PAT was stuck. Later when we added a front-facing webcam, we had PAT take a picture of what is in front of it so that it could alert the user to what exactly was stuck in front of it. This way any nearby people can help move the object or themselves get out of its way. If it was stuck behind a chair or another stationary object, PAT would ask its user to

“Please move the chair”. If PAT was stuck behind a person, PAT would say to the person in front of them, “Excuse me, may I pass by” or something similar.

2.4 Code Structure

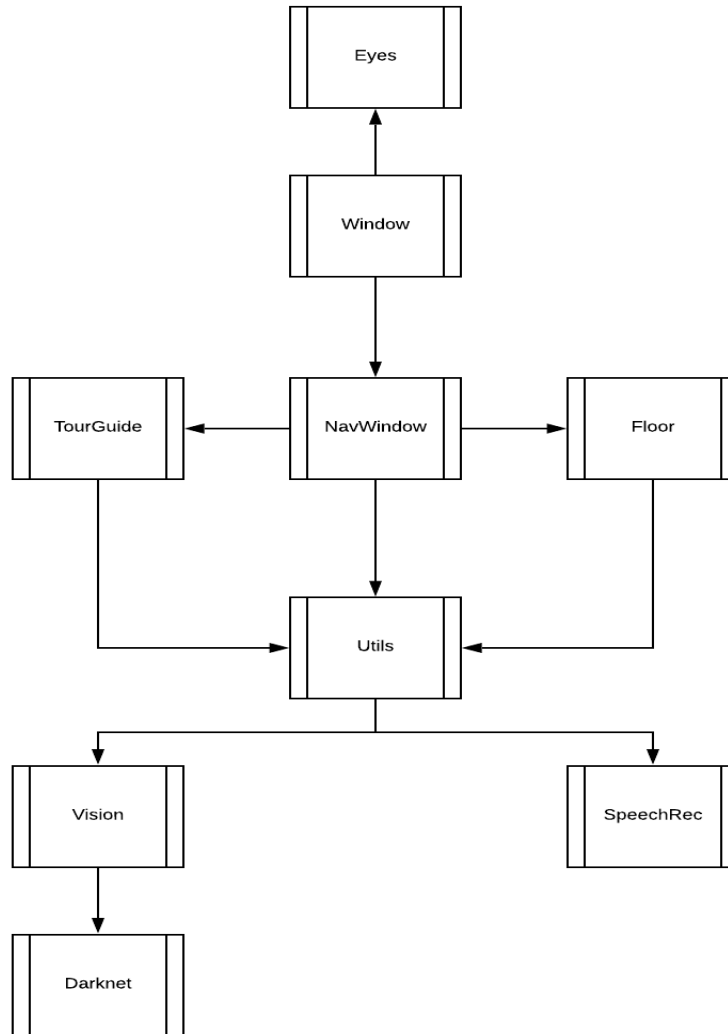


Figure 10 - Code Structure

We loosely followed the Model View Controller pattern for our code structure. The Window class acts as the main controller for switching between all of our views which are the NavWindow, Eyes, and TourGuide classes. The views still contain some controller and model logic, but we tried to limit that as much as possible. The remaining classes, Floor, Utils, Vision, Darknet, and SpeechRec acted as the Model classes. The class Utils is a Singleton in order to insure the integrity of our data and lower our coupling. This is because the Utils class represents the robot and sends all of the commands from our code to the robot.

2.5 Testing Process

Testing is one of the areas that we were not as prepared for and wished that we had spent more time planning out, specifically how we were going to approach testing our robot. The majority of our testing came from trial and error with small changes in between each trial. To test the obstacle handling we blocked a door by different objects, some that we handle and others that we do not. We also used different human as obstacles to ensure that we got the correct response. To test the tour guide we set multiple tours around our lab. To test the object avoidance, we placed an obstacle around the lab as we tested the tour functionality. To test multiple floors we assigned a destination outside of our lab to be a 'floor portal'. We would then run a multi-floor tour.

3 Results

3.1 User Interface

We did not have the ability to do user testing on our user interface due to time constraints, however we believe that it follows the human computer interaction design principles. We use large buttons to make them easier press on a touch screen. All the buttons are also clearly labeled and we believe that the process to complete actions is logical. One of the areas that we could improve is our error handling and user feedback. If we communicate better with the user, then users would not need any prior experience to operate PAT.

3.2 Features

In this section, we will discuss the final version of each feature we implemented in our robotic tour guide. All these features assume that PAT has been set up properly for its environment. This means that any accessible areas have been mapped out and all destinations have been given a tag with the correct attributes.

3.2.1 Drive to Destination

Driving to a destination is PAT's most basic feature and is capable of taking a user from its current location to any mapped destination. To access this feature, the user must have one destination selected as seen in Figure . Once a destination is selected the user presses the 'GO' button and PAT will drive to the selected destination. While driving PAT is able to update its planned path as it encounters obstacles or people in its changing environment. If ever in a situation where PAT is stuck and cannot find a path to the selected destination, it will enact the behavior as described in section 3.2.5.

3.2.2 Tour of Multiple Destinations

The tour guide path finding feature works effectively, however, because we use Manhattan distance as our heuristic to find the closest point if there is no direct path between points PAT may skip a point that is practically closer however it was theoretically farther. This was not a problem when the points are close together however if they are spaced out there is a higher margin for error. There is also redundancy built into the driving of the path to ensure all the points are reached. This came at a trade-off to speed and efficiency. If PAT gets stuck and cannot reach a destination in its tour, it will enact the behavior as described in section 3.2.5.

The multiple floor tour functionality of PAT works as expected. It finds the 'best' path for each floor and links them together through portals. It is hampered by the problems that have

been discussed above and in the Multi-Floor Traversing section (3.2.3). If the end of the tour is on a different floor than the start, PAT will return to the appropriate floor correctly.

3.2.3 Multi-Floor Traversing

This feature is not fully implemented but can be called by both the Drive to Destination and the Tour of Multiple Destinations features. It combines elements from Ava's Web API and our program. Any tag on the web API's map database with the portal attribute is remembered by our program as a location where two map databases overlapped. PAT then will drive to this portal location that connects the two floors and switch to the map database of the next floor. Using the stored information about the portal on the new floor, PAT will localize itself within that floor before continuing onto the final destination. In its current state, this feature has not been tested using an elevator and would require more testing until it was fully functional. The feature could not be tested in an elevator due to the fact that PAT requires an internet connection and would lose connection in the elevator, causing PAT to crash. We had a few solutions in mind that we were unable to implement due to time constraints, but will be discussed in section 4.2.4. Currently PAT is only capable of traveling to a destination that is one portal switch away. Our multi-floor function will not make a path that drives through two portals to arrive at a destination.

3.2.4 Speech Recognition

PAT's Speech Recognition functionality works, however there are a few issues that need to be addressed. The most severe is that there is no feedback to the user indicating when they can start talking. This causes the user to start talking before the system is listening, which leads to the first section of what the user said to be cut off. We could also add more feedback for when the speech recognition fails, letting the user know exactly what went wrong.

3.2.5 Object Handling

If PAT is in a situation where it is stuck and cannot find a path to a certain destination it will attempt to ask its user or a nearby person for help. The Ava drive base came with object detection and avoidance behavior already preprogrammed, but for our needs, PAT was too slow in realizing it was stuck. If our program determines that PAT is stuck it will stop and take a picture of what is in front of it. Then using Darknet's Yolo image recognition algorithm, PAT will categorize what is in front of it. An example of this can be seen in Appendix F. PAT will then ask if the object can be moved, or if the person can give them space to pass, wait 10 seconds to allow its path to clear up before trying again. If PAT cannot successfully arrive after trying 3 times, it returns to either the start of its tour or the dock. This feature works as intended, however, if PAT cannot reach the start it will enter into an endless loop of trying to return to the

start. There are also a few edge cases where PAT will never realize it is stuck and trigger our program's response.

4 Discussion and Conclusion

4.1 Conclusion

In conclusion, PAT is capable of doing speech recognition, obstacle handling and recognition, and tour guide functionality that can operate on multiple floors. While none of the features were incredibly complex, the strength of this project resides around the integration of so many different parts. Each of these basic features are fairly reliable, they are designed to handle many different environments and situations. PAT was designed as a general robot tour guide not just for 85 Prescott, but for any building that it is set up in. This combined with the breadth of features we developed created a product that has been effectively used in demonstrations for actual users to try. Taking ideas from past robot tour guides such as Minerva, PAT builds upon what each of those did successfully and implements those features into a generic robotic tour guide.

While developing each feature we tried to keep in mind all of the human-robot interaction design principles. PAT will have to cooperate with humans in order to give a tour not only in their physical space, but cooperate cognitively as well. PAT will have to navigate areas with high human traffic and behave in a predictable manner. We tried to use this for our advantage by communicating what PAT would need help with clearly and including redundancy in case PAT is ignored.

4.2 Future Work

With a functioning product, there is always room for improvement and we feel like our tour guide still has plenty of areas that could use improvement. We were only able to finish all of our basic features about a week before presentation and therefore did not have time to conduct a user group test to get feedback on our design. There is also room for optimization within our software; specifically the tour guide with more information could be much improved. While our speech recognition feature works, it is rudimentary and is not exactly intuitive. There are more commands it could accept and we would have preferred to use a wake word to start giving a command instead of pressing a button first. Finally, we were not able to get PAT working within elevators and traveling between floors was meant to be part of our basic functions. We had a few solutions in mind for working with elevators, but ran out of time to implement and test anything. Since we were unable to accomplish these additional features we hope that another team will carry on our work and improve upon our design.

4.2.1 Testing

If we had had that time we first would have conducted multiple user group testing sessions in order to get outside opinions on our design. Hopefully, there would be plenty of volunteers, but 20 individual test subjects should provide enough feedback. If at all possible, the test subjects should be different age groups and backgrounds. We do not want all college students with engineering majors since they may have a better understanding of robots than your average person. First, the test subjects should be given a pre-test survey to get their background information such as age and familiarity with robots. Next, the tests would need to focus on the human-robot interaction and the best way to do this would be to give our test subjects tasks to complete with limited guidance. As designers, we will not always be around to help users when they are stuck. PAT should be intuitive enough that the user can complete a task such as go on a tour between two locations. Other tasks we would want users to complete would be: get PAT to take you to another floor, ask PAT to take you to one destination, and ask PAT to take you on a tour of every room in the immediate area. Ideally, the tests would each be filmed so that it can be referred to later and used to make improvements. After the interaction with PAT, the test subject would be given additional questions about how their experience went and what things we can improve upon. After making improvements from the feedback, we would bring in a few of the original test subjects to see if our improvements matched with what they wanted. After that, we would make additional changes if necessary before finding a new set of test subjects to repeat the process again.

4.2.2 Improve Tour Guide

We also would have liked to improve our algorithm for planning the sequence of a tour. Right now we order the tour by the shortest Euclidean distance between each destination. That shortest distance might not always be the case since we do not account for walls or other obstacles. If we could incorporate some of Ava's lower level path finding to estimate distances that would greatly improve our tours. We could also save premade tours to improve the user experience. Therefore a user with no knowledge of the building would not have to guess about which destination they would like to visit. Another feature that would improve user experience would be to have PAT read a description of important locations on a tour. This could be accompanied by having a privileged user tell PAT the description while mapping. PAT would then convert that into a text file that it would save to be read aloud anytime it arrives at that destination again.

4.2.3 Improve Speech Recognition

It is not very intuitive to have to press a button to talk when all of the other buttons are right next to it. Using a wake word such as 'Alexa' would be better than using a button on the graphical interface. We were unable to implement a wake because PyQt5 did not work well with multi-threading. You could not change, remove, or add a new QWidget to a displayed QWidget

if that QWidget was created in another thread. So if we wanted a wake word listener running in a loop in another thread we would not be able to give an indication to our user interface that it was called. There was also an issue when running the google speech listening for too long, it would begin to pick up background noise that was not intended. Our fix was to only run the listening for five seconds, but this would be a little more involved when you want something to constantly wait. One way would just to run our fixed length listener in a while loop and add something to catch if any speech was picked up do not continue in the while loop. There may be a solution that allows this feature to work with PyQt5, for example, using the observer pattern and semaphores to let the user interface update itself through other threads. This is something we looked at, but since the feature was not a priority we moved on. If that would not be a solution, a wake word feature might only be possible with a different user interface library than PyQt5.

Also, we would like to add more command words so that users will not get stuck when trying to talk with PAT. We would like to add functionality that allows us to call destinations by different names. For example, you can go to a room by its room number or by the professor whose office it is. A really effective command would be if the user could tell PAT to 'Stop' or 'Wait' while it is driving, but this would require a similar solution to our wake word.

4.2.4 Elevators

Finally, we would want to expand the multi-floor functionality by adding a more robust and clean way to enter an elevator, wait, and then exit the elevator. Right now elevators are treated as a regular destination with no time for waiting. The solution could be as simple as adding sleep statements to have PAT wait for each stage. However that solution is not very smart and there are much better solutions. Another simple solution would be before entering and exiting the elevator, PAT could display a continue button that the user would press when ready. Ideally PAT would be able to recognize a closed or open elevator door, but the image classifying API we use does not offer classification for elevator doors. Also, PAT does not tell the user which floor number to press which would need to be implemented when arriving at the elevator.

In regards to the issue of losing connection to the internet and crashing when in an elevator we had a few solutions. A good solution could be, before entering an elevator, PAT would download and save locally any information it needed from the internet. Then it can just use the saved information while in the elevator before exiting. After leaving the elevator, PAT would function normally again.

Appendix

Appendix A: Python Libraries

1. Click 7.0
2. Pillow 5.3.0
3. PyQt5 5.11.3
4. PyQt5-sip 4.19.13
5. PyQt5-stubs 5.11.3.3
6. SpeechRecognition 3.8.1
7. gTTS 2.01
8. gTTS-token 1.1.2
9. Numpy 1.16.1
10. python-mpv 0.3.9

Appendix B: Cost Analysis

ASUS RT-N12 N300 Wi-Fi Router 2T2R MIMO Technology	\$29.99
1,000 lb. Capacity Furniture Dolly	\$21.98
NVidia Jetson TX2 Module	\$300.00
Logitech C270 Webcam	\$19.99
Insignia Rugged Black Speaker	\$9.99
Insignia 3.5mm to 3.5mm AUX cord	\$14.99
3.5mm Female to 2.5mm Male Headphone Adapter	\$3.99
Mount Materials	\$25.00
Total:	\$425.93

Appendix C: User Interface Images

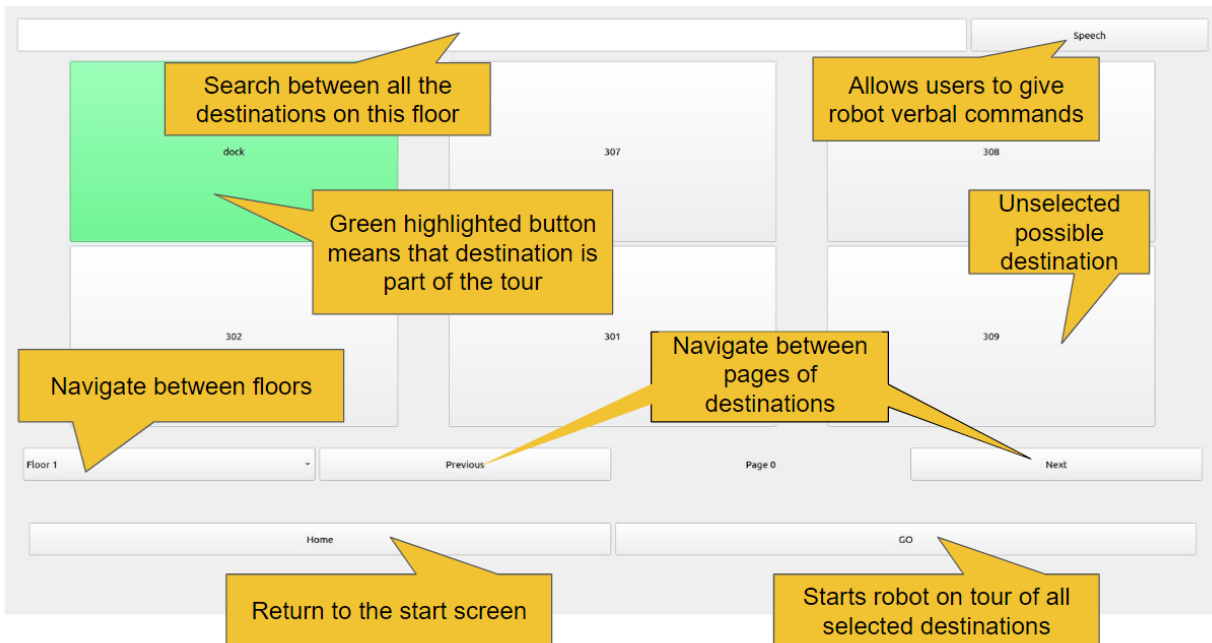


Figure 11 - User Interface with Callouts

Appendix D: User Manual

This is a manual on how to set up PAT in a new building and then use PAT as a tour guide as well as some tips for when things go wrong.

Set up Map Database

1. Connect to Ava drive base (see Appendix H).
2. Plug docking station into desired location (this location should not change).
3. Turn on Ava drive base while charging on dock station. Turn the base on by first flipping the switch in the back just under the service compartment. Then open the service compartment and press the power button for about 10 seconds.
4. Connect to Ava drive base either through a Wi-Fi router or Ethernet cable.
5. Type Ava drive base IP address into an internet browser.
6. Select 'Databases' from Menu

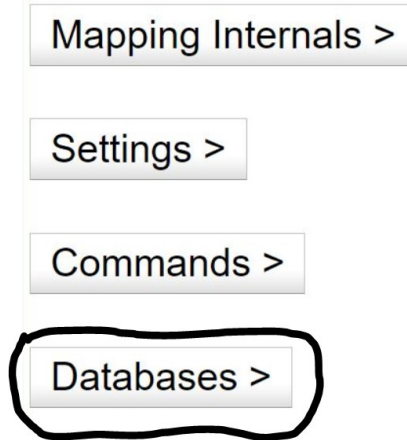


Figure 12 - Ava API Home Page

7. Select create new database, give the map a name, and turn off the sonar option.

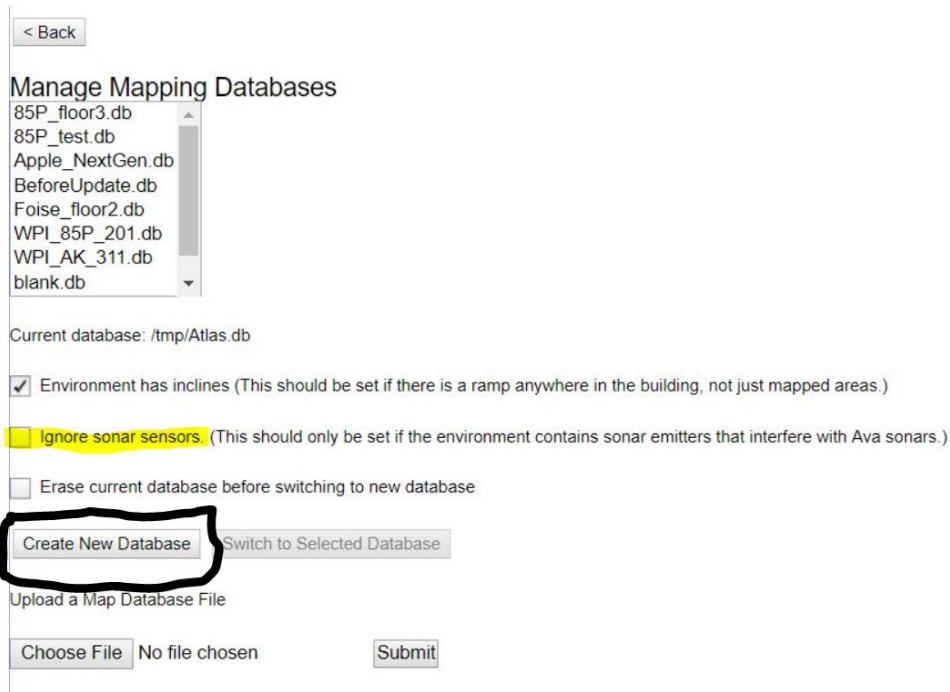


Figure 13 - Ava API Database Page

8. Then go back to the main page and select Mapping Internals.
9. Clicking the 'create map' button will put the Ava drive base into mapping mode.



Figure 14 - Ava API Mapping Internals Page

10. Before doing anything else, click the ‘Tag Current Location’ button to save the location of the dock. It is extremely important that a tag for the dock is created so the robot can localize itself. If a tag for the dock is not created it can ruin the whole map and may require the user to restart the entire process.
11. The administrator can start driving the robot around by click on points in the map. It is recommended to drive around the perimeter of rooms or areas as this helps the mapping process get a good sense of the overall picture.
12. Once the map is satisfactory, press the ‘Localize in Map’ button to save changes to the map.
13. It is recommended that the administrator go into cell clean up mode by selecting ‘cell cleanup’ from the drop down. Then by selecting four points on the map the user can change the color of the points inside that bounded area.
14. Finally, add in all the tags for any desired destinations either by selecting the ‘Create Tag’ mode from the drop down and clicking on points in the map or by driving to a spot in the map and using the ‘Tag Current Location’ button.

Multiple Floors

1. Create a new map following the same steps as before, but make sure there is an overlap between the two maps. Ideally this would be an elevator*.
2. At the overlap location, add a tag in both maps. The robot should be physically in the same two dimensional space so two different tags would be directly on top of each other.

3. On both map databases**, switch to the ‘Edit Tag Attributes’ mode from the dropdown, edit the overlapping tags, and add the attribute “Portal”. The value of this attribute should be the map database name of the map it connects to (include the .db file extension so it looks like, “map1.db”).
4. Then in the software edit the file Window.py to include a list of all available map databases. This list can be found on line 32 as seen in the figure below.

```
30
31     # Hardcoded values for the robot
32     IP = "172.18.0.1:8800" if self.is_tx2 else "192.168.1.44"
33     self.db_names = ["85P_floor3.db", "85P_test.db"] # map databases the robot can reach
34
```

Code Snippet 1 - Database Names List

*In PAT’s current state, our software will crash when PAT enters an elevator and loses connection to the internet. Therefore, multi-floor traversal does not actually work, but it can be used as a way to switch between different map databases.

**If there is trouble switching between two map databases with only one dock, the file LocalizeRobot.py can be used to find a tag and localize the robot at that position.

Running PAT

1. Put PAT on the charging dock.
2. Turn on the Ava drive base.
3. Turn on the TX2 board by pressing on the S4 button (the button closest to the front of the head).
4. If the confused face is displayed on PAT’s face after the TX2 has finished booting, the TX2 is not connected to the Ava 500 LAN. This means that either the Ethernet cable connecting the TX2 to the Ava 500 is unplugged or the TX2 has lost its static IP address for the eth0 network interface. In the case that the Ethernet cable was unplugged, simply plug Ethernet cable into whichever port it slipped out of and reboot the TX2 by holding the S4 button until it shuts down and then pressing the S4 button to boot it back up. In the case that the TX2 has lost its static IP address (this happens fairly often, unfortunately), do the following:
 - a. Connect to the TX2 with the nvidia account using ssh or mosh (mosh is more reliable).
 - b. Use the command “ifconfig eth0” to verify that the TX2 has lost its static IP address. If the output does not contain an IPv4 address, the TX2 has lost its static IP address.
 - c. Use the commands “sudo ip addr flush eth0” and “sudo systemctl restart networking” to restart the networking system.
 - d. Repeat step b. to see if the TX2 has a static IP address of 172.18.0.10 for eth0. If it doesn’t, repeat part c.
 - e. Use the command “sudo killall python” to kill the python process that is running the tour guide software. The tg_start script will automatically restart the program.

5. Wait until the happy face is displayed.
6. If multi-floors have been set up as described above, selecting the checkbox will let PAT travel between all the floors added to the db_names list.
7. By selecting one or more destinations and then the 'GO' button, PAT will lead a user from its current location to the tag(s) selected. All of the other functions as described in section 2.3 will be working as well.

Appendix E: Tour Sequence Planning Code

```

queue.append(startPath) # adds the first path onto the queue
rout = None
while queue: # continues until the queue is empty
    curPath = queue.pop(0) # pops the first path out of the queue
    curPos = curPath.current # finds the last node of that path
    if len(curPath.pathList) == len(locations) + 1: # checks to see if the current path contains all the locations that the user input
        rout = curPath # make the path the final path we will use for the tour
        break # exit the loop because we have found the 'best' path
    for loc in locations: # go through all the location that the user selected
        if curPath.isNotIn(loc): # check to insure that the current location are already not in path
            newPath = path(curPath) # create a new path from the current path
            newPath.addPoint(loc, curPos, distTo(loc)) # add the current lactation to the new path
            queue.append(newPath) # add the new path to the queue

queue.sort(key=lambda path: path.totalCost) # sort the queue by cost

```

Code Snippet 2 – Tour Sequence Planning

Appendix F: Tour Guide Sequence Diagram

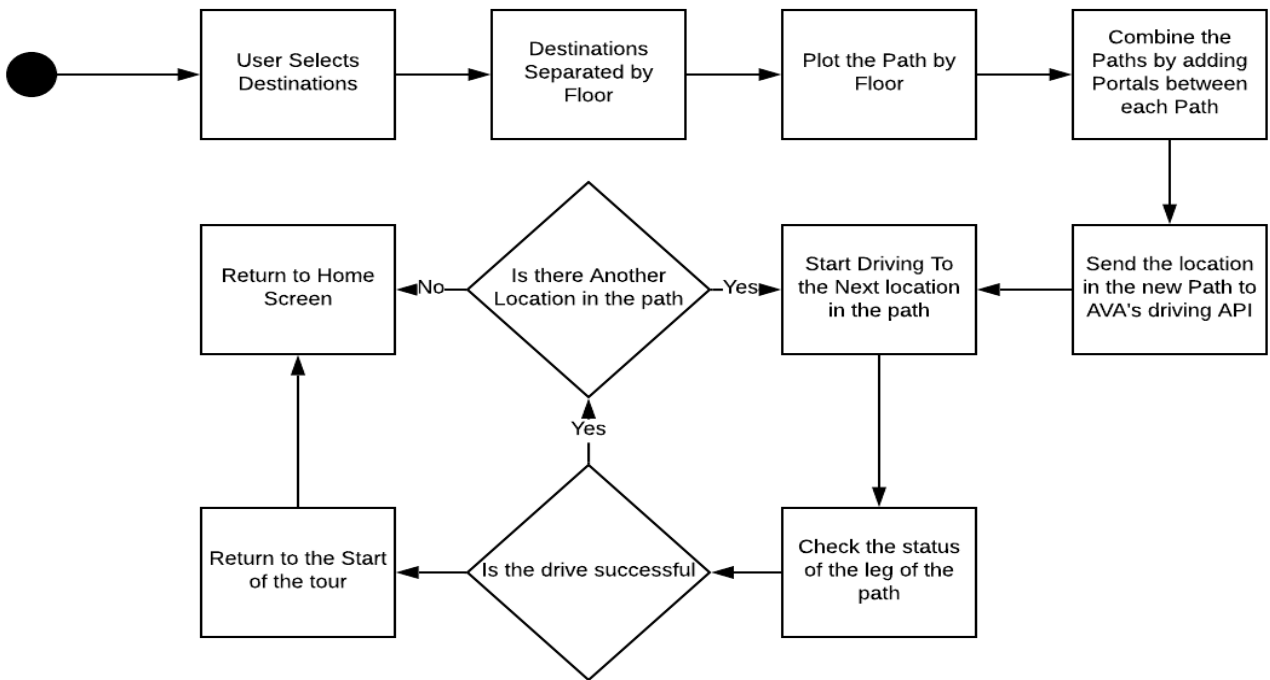
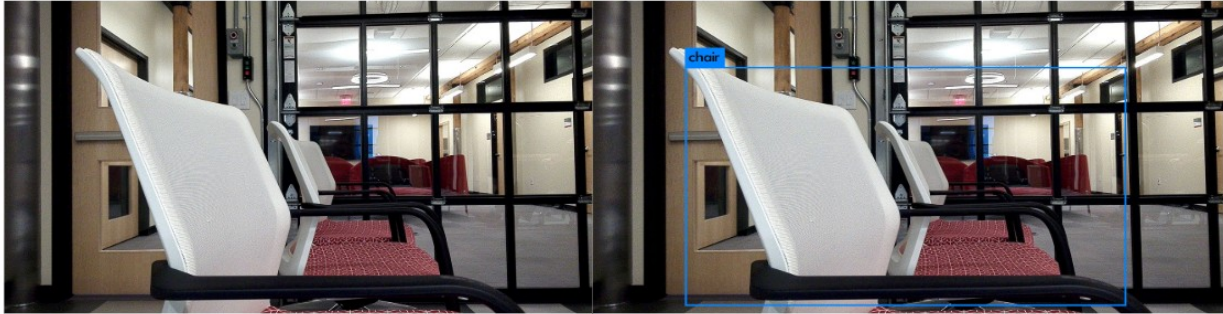


Figure 15 - Tour Guide Sequence Diagram

Appendix G: Image Recognition



This is a picture that PAT took when it was stuck

This is the output after running the image through YOLO

Figure 16 - Chair Classification

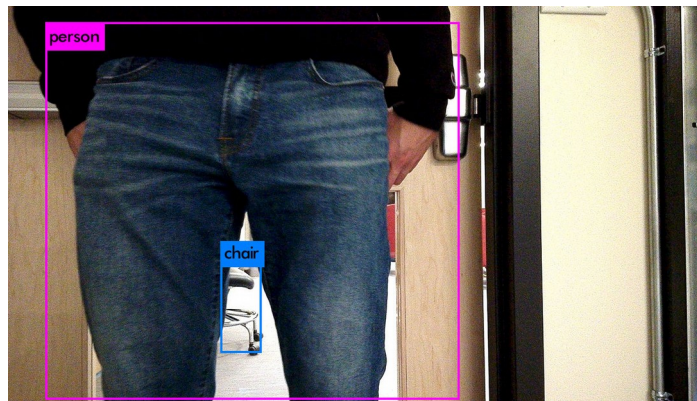


Figure 17 - Person Classification

Appendix H: Connect to Ava Base Platform

Disclaimer: This section comes directly from documentation that Ava robotics gave to us. This is not our work and is simply rewritten and copied from their PDF (*Quick Tip for AVA500 Platform*).

Set up

1. **Disable Emergency Button:** Connect M-Stop Button to P18 Payload Data Cable and Engage the M-Stop Button to override the emergency button so the robot can work in base platform mode – even though there is no emergency button. This is a mandatory step for Dell/Butler /Apple /Avabase platform to work. This step 2 is key for the baseboard firmware update to be successful.

2. **Power On the Robot** make sure robot is Docked and you see the Green LED are blinking
3. **Network Configuration for Ava Base Platform:** Using an Ethernet cable, connect your laptop to the access port in the Ava 500's service compartment.



Figure 18 - Service Compartment

4. Open the Network and Sharing Center on your laptop.
5. Click on Change Adapter Settings



Figure 19 - Adapter Settings

6. In the resulting window, double-click Local Area Connection. You will probably have more than one Local Area Connection. Open the first one in the list.

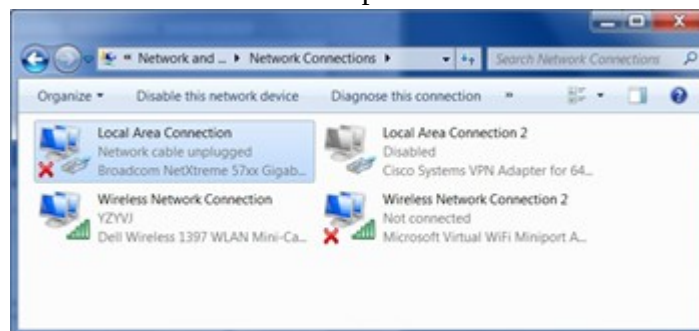


Figure 20 - Local Area Connection

7. In the Local Area Connection window, click Internet Protocol Version 4 (TCP/IPv4) and then click Properties

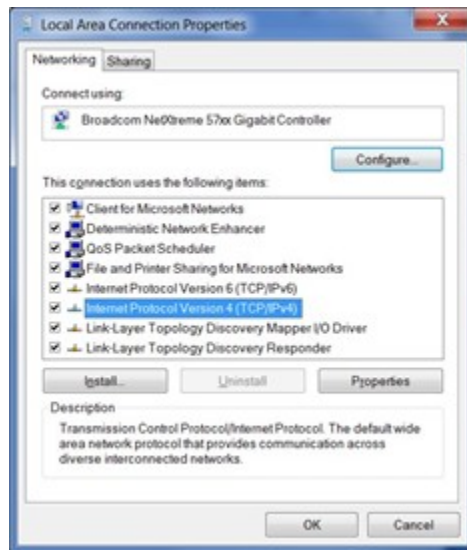


Figure 21 - Local Area Connection Properties

8. In the Properties window, click Use the following IP address: and enter 172.18.0.2. Click on the subnet mask field. Change the value to 255.255.255.0. Click OK.

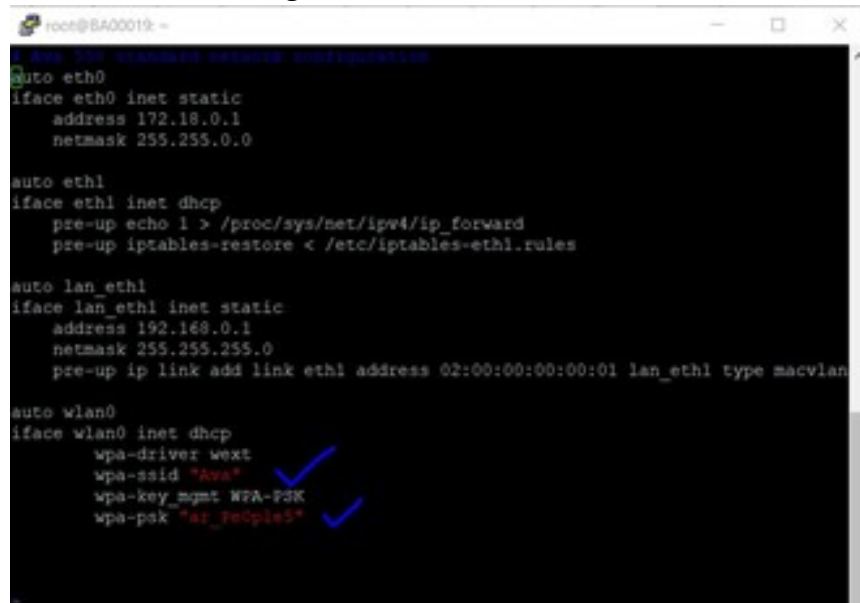


Figure 22 - IPv4 Properties

9. Click OK in the Local Area Connection window.
10. Open Command Prompt, ping the Ethernet port using “ping 172.18.0.1” command and make sure there is a connection between Ava 500 and your PC.
11. Open putty or any SSH & Telnet Client you have on your laptop and enter the Host IP 172.18.0.1 with the username and password provided in Ava’s documentation
12. Once you logged in with admin and enter su – (super user logged in as root) and the password provided in Ava’s documentation

Editing Wi-Fi

1. Once you logged in with super user on putty, issue the following command “vi /etc/network/interfaces”
2. On Vi Editor use ‘INSERT’ key in the edit interface and make the changes on SSID and wpa-pak as shown in the image below.



```
root@BA00019: ~  
# vi /etc/network/interfaces  
auto eth0  
iface eth0 inet static  
    address 172.18.0.1  
    netmask 255.255.0.0  
  
auto eth1  
iface eth1 inet dhcp  
    pre-up echo 1 > /proc/sys/net/ipv4/ip_forward  
    pre-up iptables-restore < /etc/iptables-eth1.rules  
  
auto lan_eth1  
iface lan_eth1 inet static  
    address 192.168.0.1  
    netmask 255.255.255.0  
    pre-up ip link add link eth1 address 02:00:00:00:00:01 lan_eth1 type macvlan  
  
auto wlan0  
iface wlan0 inet dhcp  
    wpa-driver wext  
    wpa-ssid "Ava"  
    wpa-key_mgmt WPA-PSK  
    wpa-psk "a! 7e0pl3"
```

Figure 23 - wpa-pak Changes

3. Hit ‘ESCAPE’ to get out of insert mode, then type: “: wq” to write and quit
4. Verify the changes in the file by opening the file
5. Reboot the robot

Citations

Ackerman, E. (2018, March 13). Ava Robotics Introduces Autonomous Telepresence Robot.

Retrieved February 13, 2019, from IEEE Spectrum: Technology, Engineering, and Science News website:

<https://spectrum.ieee.org/automaton/robotics/industrial-robots/irobot-spinoff-ava-robotics-introduces-autonomous-telepresence-robot>

Al-Wazzan, A., Al-Farhan, R., Al-Ali, F., & El-Abd, M. (2016). Tour-guide robot. *2016*

International Conference on Industrial Informatics and Computer Systems (CIICS), 1–5.

<https://doi.org/10.1109/ICCSII.2016.7462397>

Bash, N. (2019, February 28). A Record Number of Robots Were Put to Work in the U.S. in

2018. Retrieved from <http://fortune.com/2019/02/28/record-number-robots-north-america-2018/>

Burton, B. (n.d.). Meet Smithsonian's new robot docent. Retrieved December 31, 2018, from

CNET website: <https://www.cnet.com/news/smithsonian-museum-new-tour-guide-is-a-pepper-robot-from-softbank/>

Dautenhahn, K. (2007). Methodology & Themes of Human-Robot Interaction: A Growing

Research Field. *International Journal of Advanced Robotic Systems*.

<https://doi.org/10.5772/5702>

Kanda. (2012, February 8). Human-Robot Interaction. Retrieved March 5, 2019, from

<http://humanrobotinteraction.org/1-introduction/>

- LeBlanc, N. (2012). Tour Guide Robot Interactions. *Interactive Qualifying Projects (All Years)*. Retrieved from <https://digitalcommons.wpi.edu/iqp-all/1842>
- Lemaignan, S., Warnier, M., Sisbot, E. A., Clodic, A., & Alami, R. (2017). *Artificial cognition for social human–robot interaction: An implementation* (Vol. 247). doi: <https://doi.org/10.1016/j.artint.2016.07.002>
- López, J., Pérez, D., Santos, M., & Cacho, M. (2013). GuideBot. A Tour Guide System Based on Mobile Robots. *International Journal of Advanced Robotic Systems*, 10(11), 381. <https://doi.org/10.5772/56901>
- Mogg, T. (2012, August 1). Meet Tawabo, a new quadrilingual tour guide robot at Tokyo Tower. Retrieved January 4, 2019, from Digital Trends website: <https://www.digitaltrends.com/cool-tech/tokyo-tower-employs-new-quadrilingual-tour-guide-robot/>
- Paquette, D. (2017, November 30). Robots could replace nearly a third of the U.S. workforce by 2030. Retrieved from https://www.washingtonpost.com/news/wonk/wp/2017/11/30/robots-could-soon-replace-nearly-a-third-of-the-u-s-workforce/?noredirect=on&utm_term=.fa06cceb9784
- Patel, K., & Rashid, S. (2012). Guest Orientation, Assistance, and Telepresence Robot. *Major Qualifying Projects (All Years)*. Retrieved from <https://digitalcommons.wpi.edu/mqp-all/2798>

Qureshi, M. O., & Syed, R. S. (2014, December). The Impact of Robotics on Employment and Motivation of Employees in the Service Sector, with Special Reference to Health Care. *ScienceDirect*, 5(4), 198-202. doi:<https://doi.org/10.1016/j.shaw.2014.07.003>

Quick Tip for AVA500 Platform [PDF]. (n.d.). Cambridge: Ava Robotics.

Redmon, J., & Farhadi, A. (n.d.). *YOLOv3: An Incremental Improvement*. 6.

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., ... Schulz, D. (1999). MINERVA: A Tour-Guide Robot that Learns. In W. Burgard, A. B. Cremers, & T. Crisaller (Eds.), *KI-99: Advances in Artificial Intelligence* (Vol. 1701, pp. 14–26). https://doi.org/10.1007/3-540-48238-5_2

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., ... Schulz, D. (1999). MINERVA: a second-generation museum tour-guide robot. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 3, 1999–2005. <https://doi.org/10.1109/ROBOT.1999.770401>

Wang, S., & Christensen, H. I. (2018). TritonBot: First Lessons Learned from Deployment of a Long-Term Autonomy Tour Guide Robot. 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 158–165. <https://doi.org/10.1109/ROMAN.2018.8525845>

Xing, B., & Marwala, T. (2018). *Introduction to Human Robot Interaction*. In: *Smart Maintenance for Human–Robot Interaction. Studies in Systems, Decision and Control* (Vol. 129). Cham: Springer International Publishing. doi:https://doi.org/10.1007/978-3-319-67480-3_1