Team Tech Support

2505 Hayward St.
Ann Arbor, MI 48109

Jackson Eggerd
David Grover
Sara Lopez
Connor McKinley
Atharv Relekar
Sarah Rodriguez-Medina

Product Engineers

April 19, 2022

Table of Contents

Executive Summary

Hospitals are currently experiencing a shortage of nurses due to the impact from the coronavirus pandemic. In order to improve hospital efficiency and maintain patient care, Team Tech Support had the idea to develop a robot that will be able to complete tasks around the hospital and allow nurses to focus on more important responsibilities. Our team has designed the BedBot, a scaled-down robot that transports hospital beds from room to room and delivers supplies throughout the hospital.

Our team had to create a design that would be able to maneuver efficiently in a hospital setting and fit the needs of both patients and nurses. Since hospitals use different bed types and designs, we created unique mounting plates for each hospital bed. This universal attachment allows BedBot to work with any type of bed, for the mounting board contains a rectangular colored strip that is detected by the robot. Once the strip is detected at a certain distance, the robot will stop moving and drop the hitch to attach to the mounting plate. Once secured, the BedBot will be able to tow the hospital bed to another location using line navigation and obstacle detection. Our team decided on towing the bed because it allows for easier turns and the implementation of line navigation.

For line following, three infrared (IR) sensors are located at the front of the robot along the center of the bottom plate and pointing down to the ground. The two outer sensors are able to detect the ground surrounding the black line. When it detects a line, the robot is able to determine how tight the correction turn needs to be, thus maintaining its path along the black line. A more detailed description can be found in the line following section below.

To implement obstacle detection, an IR-range sensor is located at the front of the robot on the bottom plate and pointing straight across. The sensors reflect a beam of infrared light and then measure the distance of the reflected light. When the distance between our sensor and an object is 15 cm, the robot stops moving until it no longer detects the object.

To implement the vision system, a PixyCam is located on the back of the robot on the bottom plate. As previously mentioned, there is a colored strip on the mounting board that is detected by the vision sensor, which is the PixyCam, to drop the hitch onto the mount. A PixyCam uses representations of a trained object for the object detection algorithm. Our trained object is the bright rectangular strip, for we have to choose an object with a distinct shape and color in order to be properly detected by a PixyCam.

Once the PixyCam detects the colored strip, it moves directly towards it, thus orienting the robot in front of the hospital bed. As soon as the PixyCam detects that it is 8 cm away from the colored strip, it stops moving and drops the hitch.

The hitch is located on the top plate and is attached to the micro servo. The micro servo has an integrated shaft that is controlled to rotate, thus allowing for the drop-down motion of the hitch.

During testing, the robot was able to effectively detect objects that were in its line of vision. However, the sensors are not able to detect objects that are not directly in front of the bot, which may cause collisions. Another sensor may need to be incorporated in order to resolve this issue before it is scaled up and in the market. We tested different types of turns that the robot could make, and found that a ninety degree curve is possible only if the corner is curved. Hospitals would need to create a pathway of black lines going from room to room with the desired curve when necessary.

The scaled down BedBot is 16.1 cm x 18.5 cm x 9.7 cm. The full scale BedBot will be scaled up by a factor of 6. Once scaled up, the BedBot is very cost effective. Scaling up the design and manufacturing costs, five models of the BedBot would cost around $68,000. We can compare our design to TUG, a robot that has been implemented in hospitals. TUG autonomously delivers food and drugs and costs roughly $100,000. Our estimated cost for BedBot is much lower than our robot is designed to conduct more tasks, thus being more efficient for hospitals. Besides delivering hospital beds, BedBot could also deliver food and supplies throughout the hospital. We mentioned that mounting boards with rectangular strips would be attached to each bed, allowing for any hospital to use this robot for the beds they own. The same can be said for delivery compartments. If a mounting board is attached to a delivery compartment, BedBot will be able to follow the same steps to navigate throughout the hospital and deliver supplies as it does so.

Introduction

Due to the coronavirus pandemic, hospitals are currently experiencing a shortage in nurses. Doctors have had to help with tasks that nurses usually do, taking time away from their own responsibilities. Hospitals have had to offer a considerable amount of employment bonuses in order to hire more people. Incorporating a robot into the hospital that helps with these tasks would greatly benefit hospitals, nurses, and patients.

Our task is to design a robot that transports hospital beds from room to room as well as delivers supplies throughout the hospital. Our prototype must be able to detect obstacles and avoid collisions, properly attach to either the bed or the delivery compartment properly, and follow the correct path to correctly drop off the hospital bed or compartment. This prototype will later be scaled up to fully operate in a hospital setting, taking into account the size of the hallways, the weight of the attachment, and the speed of the robot.

Our design of the BedBot is cost effective for hospitals as they no longer need to worry about offering extreme bonuses when hiring all while incorporating a service that focuses on the needs of nurses and patients. The following sections cover the specifics on the overall design of the robot and how each system functions. These include the hitch-system, the vision system, and the line navigation system. The last section covers the future implementation of the BedBod as well as potential issues that should be recognized.

Design Overview

Given the M-Bot base, our team began brainstorming the best way to utilize its features to create our prototype. The preliminary design was a custom bed that the robot could drive under and then raise up via a vertical linear actuator, but this was scrapped in favor of a design that hitched onto the front of a bed, as it would be easier to adapt to hospitals that used different brands of beds, and eliminated the need to create a bed from scratch. We decided the wheels should be towards the front of the robot in order to counteract any instability while towing. We put the arduino on the lower face plate, along with the PixyCam facing backwards. We modeled our hitching system on the common tow hitch for a trailer, with a hitch arm attached to the servo on our robot, and a mounting plate that could be adaptable for multiple beds. Finally, the infrared sensors for line following and collision avoidance were mounted at the front of the bot on the lower face plate in order to have the most accurate readings of their surroundings.

3D Printed Parts

As many parts of our robot are specific to our prototype and have complicated geometry, the team decided to utilize 3D printing to create them.. Shown in Figure 1, a mount was designed for the servo motor on the rear of the robot. Design of this mount used exact measurements of the particular servo we used, along with slight tolerances for installation. It was printed using a resin printer, as resin printing creates a stronger end part than traditional filament in Fused Deposition Modeling(FDM)

printing. We used resin to print all of our parts, to be consistent and this servo mount was incredibly useful, never had to be replaced or redesigned during testing, and is on the final prototype of the robot.

The hitch system was entirely 3D printed, and was made up of two main parts: the mounting plate and the hitch itself. The hitch itself was increased in thickness and length as testing progressed and the team's preference for the position of the hitch changed. The mounting plate also changed as testing progressed and shortcomings in the design were discovered, as shown by Figure 2. The first design broke while first testing the towing capabilities of the robot, and was made thicker, along with extra support, in order to withstand the forces generated while towing the bed. Finally, when testing PixyCam, our team realized the accuracy was not as high as originally theorized. Therefore, we decided to update the mounting plate to include a larger area for the hitch, along with a small divot that the hitch would fall into while towing. This final design worked incredibly well in final tests, and is what we stuck with for the prototype, shown in Figure 3.

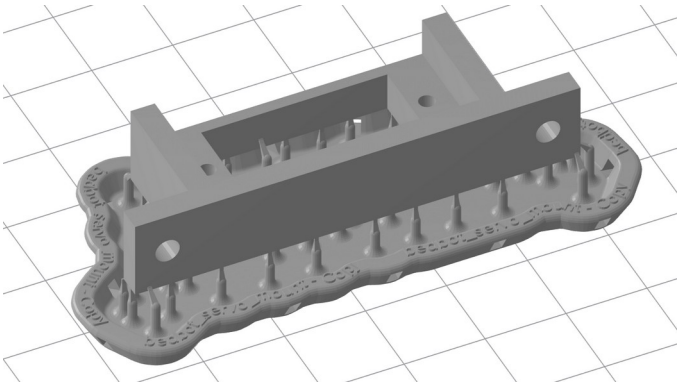

Figure 1: Servo mount displayed in 3-D printing software with supports for print

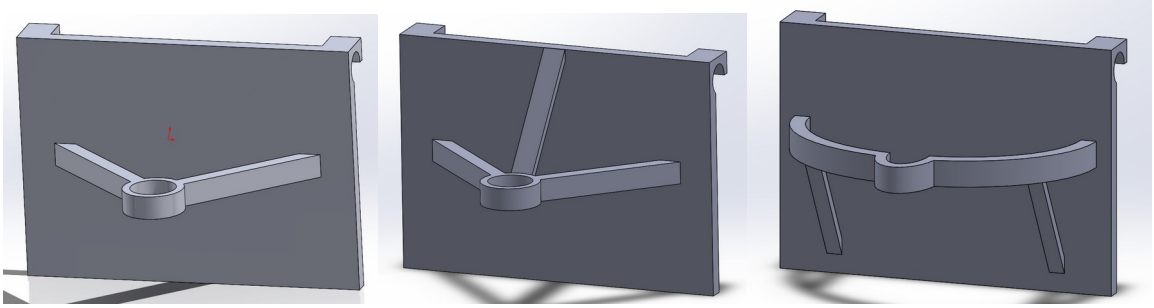

Figure 2: Iterations of the mounting plate

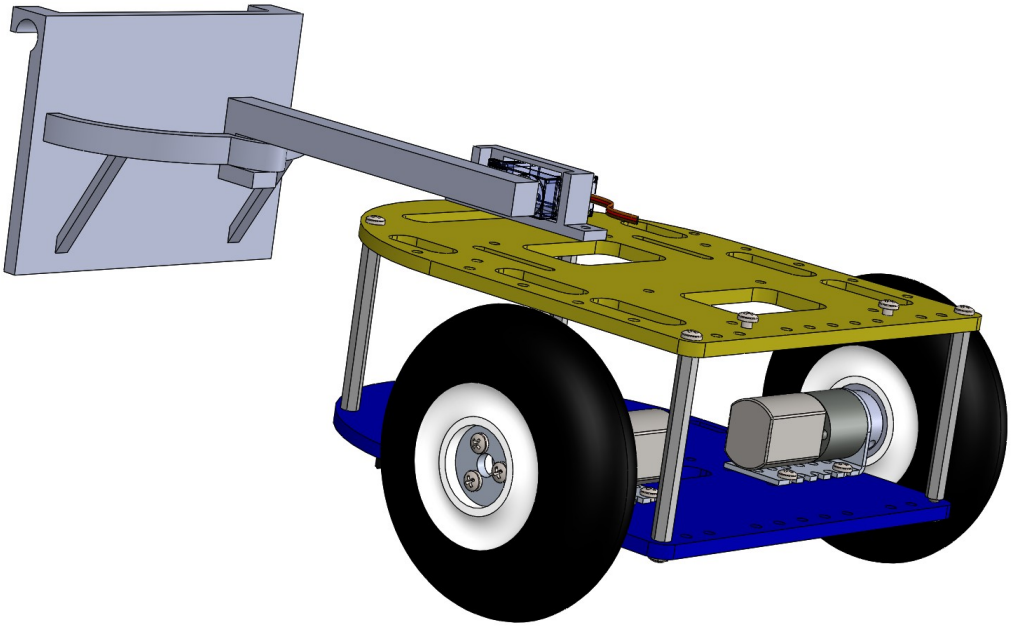

Figure 3: Final hitch, mounting plate, and servo mount shown in Solidworks

Vision Tracking

In planning for the robot to hitch to a hospital bed, we knew there would be several complications. The first was finding the bed, as we didn't want to rely on all hospital rooms to be organized the same. Secondly, when the robot did find the bed, it would need to drive up to it in a way that positioned the hitch inline with the hook on the robot so that it would be able to drop down without missing the hitch. The team came to the conclusion that the most realistic solution to these constraints would be a vision tracking system capable of detecting the bed and guiding the robot through the hitching process. The solution we settled on is a PixyCam camera due to its all-in-one package of object detection, software visualization, and provided code library for interfacing with the robot.

Object Detection

A key strength of the PixyCam is the software paired with it to visualize object detection. PixyCam uses a system of object "signatures" which are representations of a trained object saved to the camera that can be used for the object detection algorithm. Using an application called PixyMon, the camera can be put into an object capturing mode where the software will visualize a distinct object it picks out of the video frame. Pressing a button on the camera confirms the object, saving its signature to the camera. The ability to save objects to the camera allowed us to train the camera

separately with a computer and then plug it into the robot when it was ready to be used. However, in this process, we discovered an imperfection in the detection, where the camera finds several small objects which it incorrectly labels as smaller versions of the object signatures we saved (see Figure 5). This “noise” was something we decided was best to handle through code covered in the section below.

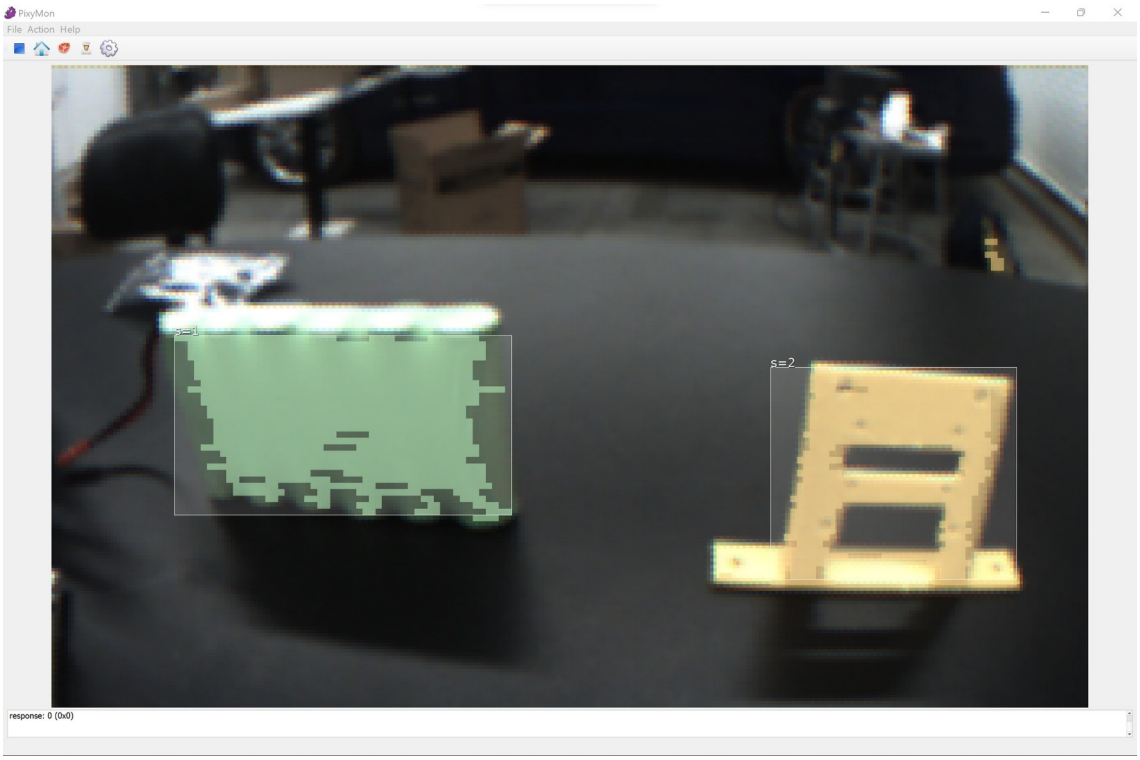

Figure 4: PixyMon software, multiple object signatures visualized

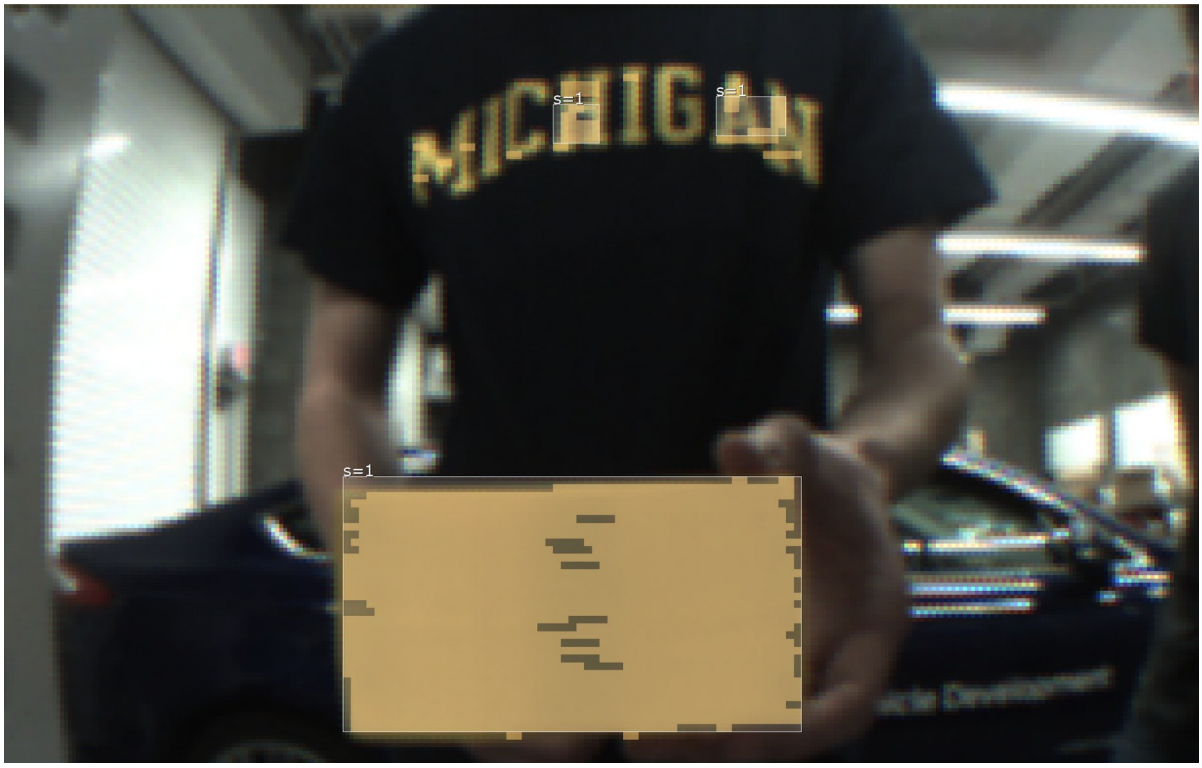

Figure 5: Text on t-shirt creates undesired background objects

Handling Objects With Code

The PixyCam itself is unable to make decisions about the objects it detects or the data it sends to the robot. Instead, we used a code library made for the PixyCam that allows Arduino code running on our Raspberry Pi to communicate with the camera. The code is able to access a collection of all objects the PixyCam is detecting. To handle the "noise" found previously, the code was designed to search through all objects and determine the correct vision target. By default, the PixyCam returns only objects matching the programmed signatures. Our chosen setup relies on only one object signature, so the only factor left to consider is the size of each object. By testing the PixyCam in several areas with different vision targets, we came to the conclusion that it is reasonable to rely on all background objects being smaller than the desired vision target. Using this conclusion, the final coded solution to finding the vision target checks the width of each object detected and keeps track of which is largest at any given moment. If the width of the largest object is above a 30 pixel threshold in the camera's view, the rest of the code continues to perform actions using this target. Otherwise no object is large enough to be used for tracking and the robot enters an error state where it must be redirected to another target by a person. We found this solution reliably eliminated the chance that the robot began tracking false objects and was able to stop the robot from performing actions when it no longer had a view of its target.

Autonomous Control

For the broader algorithm controlling the robot, vision tracking is engaged when the robot first enters a room looking for the bed it was assigned. Thus, our vision tracking algorithm was designed to begin at the entryway to a room. With the current iteration of our design, the robot will be manually switched to the vision tracking process once it drives to the correct room using different code. Once vision tracking is engaged, all controls are handled by the data provided by the PixyCam. To make sure the robot moves to the correct location in the room, the robot first pivots towards the vision target it sees on the bed. To judge this movement, the code measures the target's position on screen (particularly the horizontal position or "x" coordinate). If the position exceeds a threshold to the left or right, the robot turns in that direction. Once the bed is centered in the camera's view, it drives towards the bed, pivoting again if it loses alignment. To determine when to stop in front of the bed, the robot checks if the target's width on screen is above a threshold. Once it stops, the robot checks the alignment of the bed with a narrower threshold, pivoting so that the hook is within the ideal range of 1-2 inches from the hitch. Finally, the servo-mounted hook is dropped down, connecting the robot and the bed. Now, when the robot drives out of the room, it tows the bed, following the same path it took in.

Line Following

To navigate from room to room in the hospital, the team decided to use line following in order to keep a majority of the robot's navigation simple. We determined the most reliable setup was three infrared line following sensors mounted on the front of the chassis. These sensors detect the amount of infrared light reflected by the surface directly underneath it. The sensors can be calibrated through the use of a dial on each sensor. This allows the line following to work on different surfaces and lines as long as the line reflects less light than the surface.

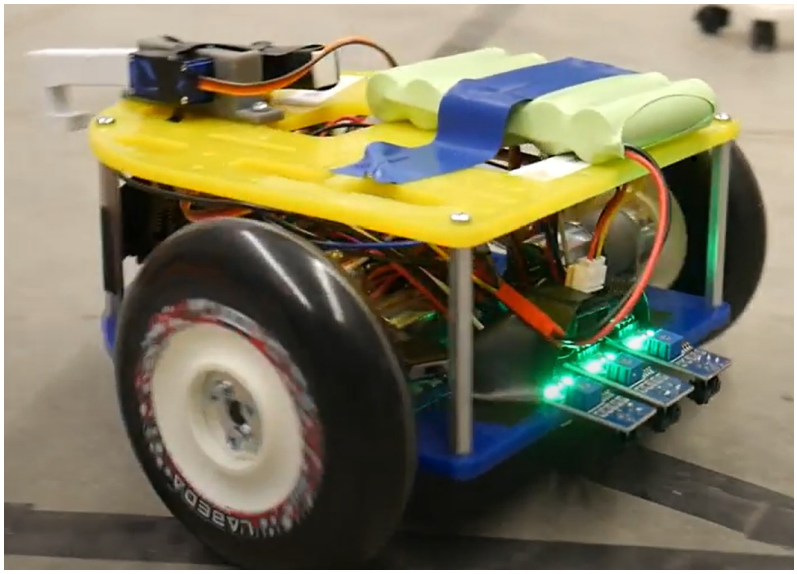

Figure 6: Three IR sensors pictured on the front of the robot

Steering

The collective data provided by the left and right sensors can be used to determine which side of the line the robot is on. The data provided by the middle sensor can be used to determine whether the robot is still centered on the line. For example, if only the left sensor detects the line, we know that the robot is to the right and is not centered on the line. However, if the middle and left sensor detect the line, we know that the robot is almost on the line but is still to the right. This can be used to determine how tight the correction turn needs to be.

Present Issues

The simplicity of the setup can lead to problems in a dynamic environment. For example, differences in lighting on the robot's path can result in the sensors reading a false positive. This can ultimately lead the robot to lose the line it was following. Once the robot no longer detects a line, it does not have sufficient sensors or logic to return to the line. However, when deployed in a controlled environment, the robot often behaves as intended.

Cost

We wanted to ensure that our robot was cost-efficient, and would not require too many expensive materials to develop. Thus, our goal was to keep the costs for our prototype under $150.

One of the major items that we needed was a PixyCam to detect objects and help the robot hitch itself to the bed. This was our most expensive item at $50 and there was not any cheaper alternative to use which would simultaneously not require extensive knowledge about computer vision and object detection.

Another major required object that we needed was a Raspberry Pi. This was required to implement the code for the object to follow the lines laid out and detect and move towards the object needed. This was our second most expensive item at $41. A cheaper alternative would have been an arduino, however the Raspberry Pi had better input/output signals and port connections and thus, we justified the higher cost.
The rest of our items were much cheaper. The three IR sensors, which were used for detecting the lines for our robot to follow, cost a total of $14. The two DC motors that we needed to make our robot's wheels turn cost only $4. Accompanying the motors, we also had speed controllers to ensure that our robot could appropriately speed up and slow down as needed. The two speed controllers cost a combined $20, which is quite a bit more expensive than the motors but not as much as our two most expensive items.

The micro servo needed to drop down the hitch to latch onto the hospital bed cost us $6. Finally, the 3D printed parts cost us only a total of about $15. Since we decided to 3D print our hitch and our mount, we were able to save a lot of money on these parts whereas it would have been much more expensive to buy them.

Item	Quantity	Total Cost
PixyCam	x1	$50
Raspberry Pi	x1	$41
Provided MBot Base	x1	$75
Micro Servo	x1	$6
3D Printed Parts	x3	$15
IR Sensor	x2	$14
	Total:	$136

Table 1: Cost of prototype

Performance

Hitch Testing

When testing the hitch, our team set out to ensure that the robot could successfully pull just the bed, and then the bed with a simulated passenger. Our criteria was simply that the bot must successfully pull the bed and simulate passing around corners and on straights in a controlled manner, with no failure of parts. During testing of the original hitch mounting plate, the ring snapped off of the plate, forcing a redesign with more support for the ring on the mounting plate, and no more fatigue failures occurred in subsequent testing. When PixyCam was used to hitch the bot onto the mounting plate, we quickly realized we had overestimated its accuracy, as it could only successfully hitch itself 10% of the time. We redesigned the mounting plate one last time to include a larger area for the hitch to drop into, shown in Figure 2. This increased the percentage of successful hitches by the robot to 90% of the time. With this final mounting plate design, the BedBot was able to meet our criteria for towing the bed and passenger in a controlled manner, with no failure of parts.

Line Following Testing

Our criteria for the BedBot's ability to follow lines changed as testing went on. At first, we wanted the robot to be able to take 90° sharp turns and smooth turns with a radius of 1 inch, along with successfully staying on a straight. When we first started testing, the bot could navigate sharp corners of about 45° and smooth corners of radius 6 inches. We updated the line following code to help the bot make sharper turns of around 60° and of radius 3 inches, but this sacrificed the smoothness of how the robot drove on the straights, as the more dramatic updating it did caused it to zigzag instead of driving straight. We tuned the robot to find a balance between these two aspects, finding that the robot could still drive smoothly on straights while making 50° sharp turns and smooth turns of around a radius of 4.5 inches. Our group agreed that the line following system was successful with these values, as it achieved a good balance of our original criteria, and since we could design the turns it would take in a real-world setting, we could work around these values, we just wanted to optimize them as much as possible.

Future Improvements and Potential Issues

Limited time for this project means that not everything that we envisioned for this robot was possible. We had to simplify our designs to a version that could be completed in just a few months. However, the current iteration of our robot was created with the idea in mind that this was a proof of concept. It has been sized down from its real world application as well. If given the time needed to perfect our vision, we would be able to make a multitude of improvements.

The current system we use for navigation works well but could be improved upon relatively easily. The IR sensors that we use for the line following bring with them some limitations. By being restricted to following a particular line, the options for where the robot can go is greatly limited. This method also creates an issue at corners and intersections. The robot is not capable of the typical 90 degree turns as it can't see them coming until the robot is already off course. Additionally the robot can only ever go straight at cross intersections. To fix these issues we would replace the IR sensors with a second PixyCam. The PixyCam will be able to detect colors quickly and much more accurately than the current design. One Pixie alone does the job of three IR sensors, greatly reducing the amount of complexity in the robot. Along the lines we would be able to put colored stripes to indicate when a turn was approaching. This would allow the robot time to slow down and switch to code that completes 90 degree turns. The colors can also be used to indicate an intersection so that the mapping system can tell the robot which way to go.

More sensors would be added to improve our obstacle detection. The current system can only really detect objects directly in front of it. If something is coming from the side then the robot wouldn't be able to respond until it was too late. This is fixed quite easily by adding a sensor on each side of the robot. They could all run on the exact same code and wouldn't take up a great amount of processing power.

A feature that we weren't able to show in the given timeframe is that our robot can carry or tow much more than just beds. The hitch plate we designed can easily be modified to fit many attachments. The robot would be able to pull a food cart and deliver orders to patients. It could tow a supply cart to help nurses restock their rooms. The robot could also be able to pull a cleaning attachment that would vacuum or mop the floor as it went along to help with sanitation. One of our main goals was to create a robot that would be able to help hospitals in many places, taking up the small tasks that take up too much time for the limited number of nurses.

One of the ways that we plan to help the nurses is by having our robot guide patients. This would be a far future goal but realistic. We could add a screen display that could be interacted with or display instructions. The robot would be able to show patients which room to go to completely autonomously.

To help the robot appear appealing to the general public we would make a custom chassis. The robot would look much more approachable without wires hanging everywhere and a custom chassis would allow for the above improvements to easily be implemented into the design as we see fit. Without the limitation of the M-Bot, the cameras and sensors could be placed in areas to allow for better scanning.

A long term plan that we always had in mind was an internal mapping system. The current iteration of the robot can get from place to place but only if there is a direct line to it. An internal mapping system would let the robot travel to any room no matter where it was situated in the hospital. We could use the color coded floor tape if needed to mark out intersections but if the technology becomes advanced enough, we could take out the need for line following altogether. This would require a lot of testing and time but is essential if our design was to be used in the real world.

There are a few possible issues we foresee with our robot. One that often comes up in discussion is the topic of having no nurse supervising the patient during travel. This makes complete sense and we have come up with a few solutions. First is to somehow implement the robot with a way to monitor vitals. The robot could send out an alert if it notices the patient is in danger or distress. A second option was to only use this method for the lower priority patients. If the patient is the type that is already in recovery or has a simple ailment then the possibility of something going wrong on the journey is greatly reduced. Finally the robot could simply be used to move empty beds. Empty beds are still moved dozens of times a day as hospitals come and go. This is the perfect task for a robot to work on while the doctors and nurses focus on the patients. Another issue that needs to be considered is how comfortable nurses and patients would be with the idea in the first place. It's hard to put the same trust into a robot that you would a living being. To solve this problem we would have a trial run with the robot. We would start by just having one robot in the hospital doing simple jobs. As it performs we could survey nurses and patients on how to improve the design. Our goal is to make sure that everyone would feel safe with our robot. Our robot is designed to improve the lives of those in the hospital and we will do our best to achieve this.

Conclusion

Team Tech Support has put extensive time and energy into developing a robot capable of having a real world impact for hospitals. Every component of our design has gone through a series of iterations and refinements to increase our design's effectiveness and create a realistic prototype of our vision for the robot. Analyzing both performance and cost, our team is confident that the BedBot can be competitive in the developing market of hospital robots. While there are a few remaining issues with the current design, we have devised a number of improvements and new methods to bring the robot closer to a complete product. Overall our team is satisfied with the progress we've made and believe strongly in the product we've developed.

References

Boyle, P. (2021, September 7). Hospitals innovate amid dire nursing shortages. In *Association of*

American Medical Colleges. Retrieved from https://www.aamc.org/news-insights/hospitals-innovate-amid-dire-nursing-shortages

Robots Making Rounds (2008, March). In *MachineDesign*. Retrieved from https://www.machinedesign.com/archive/article/21826528/robots-making-rounds

Simon, M. (2017, November 10). Tug, the Busy Little Robot Nurse, Will See You Now. In

Wired. Retrieved from https://www.wired.com/story/tug-the-busy-little-robot-nurse-will-see-you-now/