COLORADOSCHOOLOFMINES
EARTH • ENERGY • ENVIRONMENT

CSCI 370 Final Report

Blaster's Bucket

Kara Schuler
Dan Collins
Shane Gordon

Revised May 31, 2024

CSCI 370 Summer 2024

Professor Donna Bodeau

Table 1: Revision history

Revision	Date	Comments
New	5/17/24	Created the Introduction, Functional and Non-Functional Requirements, Risks, the Definition of Done, Team Profile, and Appendix sections.
Rev – 2	5/24/24	Updated sections II-V and Appendix A. Created System Architecture section.
Rev – 3	5/31/24	Updated section V. Created Software Tests and Quality and Ethical Considerations sections. Added 'The Construct' section to 3rd party software.
Rev - 4	6/7/24	Updated section VII and Appendix A.
Rev - 5	6/11/24	Created Results, Future Work, Lessons Learned, and Acknowledgement sections. Updated III, VII, and Appendix A.
Rev - 6	6/13/24	Revised all sections, added more figures, and made updates based off of the reviews from other students.

Table of Contents

I. Introduction

The product is an Autonomous Paddock Mucking Robot for the Longhopes Donkey Shelter. The sanctuary is located in Bennett, Colorado, and was founded in August 1999. They rescue unwanted donkeys that would otherwise be slaughtered and rehabilitate and rehome them. The shelter may care for some of the donkeys for the rest of their lives as they have health issues or are blind, making them more difficult to adopt out. The staff and volunteers care for a large amount of animals, and part of that care involves cleaning their paddocks daily. To make the shelter's operations more efficient and less labor intensive, the client has requested an autonomous paddock mucking robot that can take over some of the responsibility for cleaning the donkey paddocks. This robot will be used to reduce the amount of manual labor and the cost it takes to keep the paddocks clean. Offloading some of the work to the autonomous mucking robot will allow the employees and volunteers to devote their time to other tasks, improving the efficiency and impact of the sanctuary.

Figure 1: Staff and Volunteers at Longhopes Donkey Shelter

The robot must be able to navigate through donkey paddocks, avoiding obstacles such as food and water troughs, toys, donkeys, and fences. Its task is to scan the paddock for manure, find and traverse a path to the manure, pick up the manure, and deposit it at a predetermined drop location inside the paddock. This project will be continued in future field sessions, split between another CSCI 370 team and an EDNS 490 team. Our goal is to create the software framework for the robot so that it can be expanded upon and physically implemented at a later time.

The clients of this project pertain not only to the staff of the Longhopes Donkey Sanctuary but also to more potential clients around the country who may employ future versions of the robotic unit's services to cut labor costs and ensure animal safety. It also extends to the animals themselves, many of whom are older and need medical care and rely on the staff's tireless efforts to give them a happy and comfortable life. Overall, we must ensure that the software we produce throughout this project is sufficient to improve the efficiency of this center and others like it for the sake of all of these stakeholders.

II. Functional Requirements

1. The robot unit must be able to navigate the paddock independently without a pre-established map

2. The robot unit must be able to scan the paddock for manure and the drop location.
3. The robot must be able to traverse a sufficient 95% of the paddock and pick up all manure therein
4. The robot unit must be able to recognize manure in different settings and surroundings, including dirt and sand paddocks, grassy pastures, and concrete barn structures with rubber matting.
5. The robot unit must avoid collisions with both stationary and non-stationary obstacles.
6. The robot unit must be able to uniquely identify individual paddocks.

III. Non-Functional Requirements

1. This robotic unit's hardware will need to be of reasonable capacity to handle these requirements
    a. It must possess enough memory capacity to store a map of its environment and potentially certain statistical observations
    b. It must be able to cool itself, as it will operate outdoors
    c. It must have enough data ports to support six (6) individual motors for the physical design
2. This software will need to employ a camera system to identify manure
3. The software will need the ability to identify manure from camera intake
4. It will also need a Light Detection and Ranging (LiDAR) system to give a more detailed map of the robot's surroundings to avoid collision with fences, farm equipment, and the donkeys themselves.
5. The software will need strong data transferring ability to ensure these systems can work in tandem to allow the robot to navigate accurately around its surroundings.
6. A unique identifier will be necessary to identify the manure dropoff location and orient the robot's position within the paddock.

IV. Risks

The main risks associated with this product are related to its deployment inside animal enclosures. There is the possibility that the donkeys will be afraid of or hostile towards the robot, which could result in an inability to perform its tasks, damage to the robot, or harm to the donkeys. On the other hand, the donkeys may want to interact with the robot without hostility or fear, but still prevent it from completing its tasks or causing damage to themselves or the robot. Regardless of the donkeys' behavior towards the robot, it must still be properly programmed to avoid obstacles. This is also critical to avoid harming the robot.

If the software crashes, the robot must remain stationary in the field and will likely need to be physically maintained by staff at Longhopes, another reason why thorough documentation is an important part of this project. If the robot were to shut down in the middle of cleaning a paddock, the sanctuary employees may need to manually search for the robot in the paddock or the pasture to repair or restart it.

A developmental risk involves the creation of a virtual environment to test the robot's software. The environment will not be able to exactly mimic the conditions of the donkey shelter and could therefore skew testing results to be more accurate than the physical unit may be in the real environment. We will need to take this into account while creating the environment and attempt to make it as accurate as possible while recognizing its limitations. The initial iteration of the physical robot will need to undergo more testing in the real environment to validate the results from the virtual environment and make sure it is consistent.

V. Definition of Done

For this five-week session, in order for the project to be considered complete, the robot must be able to scan the paddock for obstacles, identify manure with the camera, plot and travel a path to get to the manure, pick it up, and deposit it at the manure's drop location. The functionality requires that the robot can recognize donkey manure, find a path from the robot's current position to the manure, move the robot so it is positioned appropriately to pick up the manure, then locate and navigate to the manure drop location. It must also be able to identify when there is an obstacle, such as the donkeys, donkeys' toys, or fencing, while in operation to avoid collisions and have the capability to navigate through the space safely. The obstacle identification, path navigation, and locating the drop spot will be tested in the virtual environment in several different artificial worlds. The manure recognition will be tested using a collection of different videos and photos that were not used in training the recognition software, which will be quantitatively measured using percent accuracy.

VI. System Architecture

Algorithm

The algorithm for this system is fairly straightforward. The robot must first scan the paddock to locate nearby manure piles as well as the manure drop location. The drop location will be identified using an AprilTag that may also contain information about the paddock such as a paddock number. If the robot has not yet located the manure drop site it will ignore manure until the drop site is found. It can then find a path to the closest manure pile, pick it up, find a path to the manure drop location, and deposit the manure. The robot will then check if it has enough battery charge to continue cleaning and if it has traversed at least 95% of the paddock. The robot will only make one pass of the paddock before shutting down; the 95% traversal ensures that the majority of the manure will be found and collected but also ensures that the robot will not run continuously after all the manure has been picked up. See Figure 2 for the detailed flowchart of the algorithm.

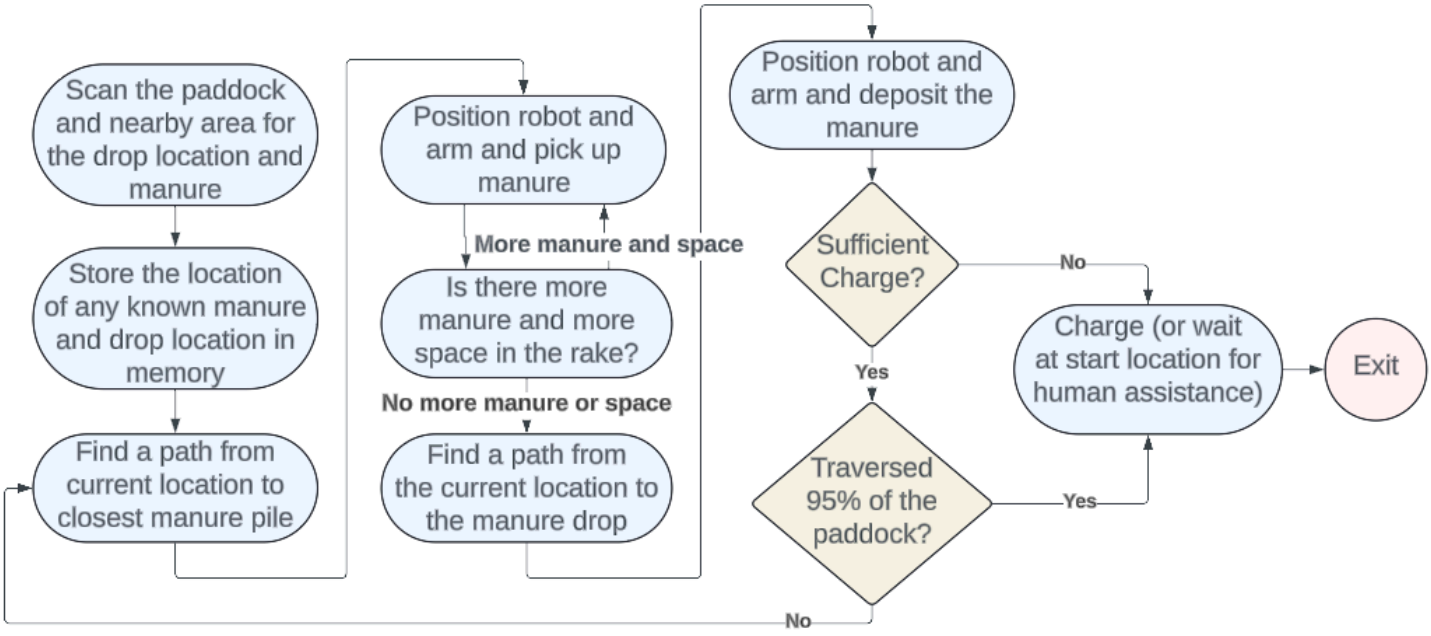

Figure 2: Algorithm Flowchart

Third Party Software

The software for the robot will be run using a Robot Operating System (ROS), specifically ROS2 Humble Hawksbill, an open-source software suite commonly used for robotics, and will be built in ROS Development Studio to be able to use several built-in features. There will also be several existing open-source software utilities incorporated into it. The software will use three core technologies: ROS, Gazebo, and YOLO, each of which works in tandem to form the foundation of the Robot control system. The overall architecture that the system will follow is outlined below in Figure 3. Different functionalities will be separated into Nodes, which will communicate and interact by publishing data to different Topics or Services. Topics have publishers and subscribers while Services have service providers and clients. With Topics, the publisher decides when data is sent out and all subscribers see the data as soon as it is sent out. With Services, the client must send a request to the Service to be able to get data.

For sensor data, the publisher/subscriber schema was more appropriate as it allows the subscriber nodes (Computer Vision and Pathfinding) to receive data at the rate it is produced, rather than having to ask for it as in a service/client schema. This philosophy also transfers to the Object Recognition topic, which should ideally be driven by the node which produces the data, Computer Vision.

For Movement Commands, the Service/Client model was more sensible as the Pathfinding and Navigation node should only have to send instructions when necessary. With the Service/Client model, communication is also two-way as each service request comes with a response, thus the Motor Control node can send back a signal if the desired movement command is impossible/overridden by LiDAR data. It was determined that the LiDAR sensor may potentially need direct access to the movement commands topic in case collision with a moving object is imminent.

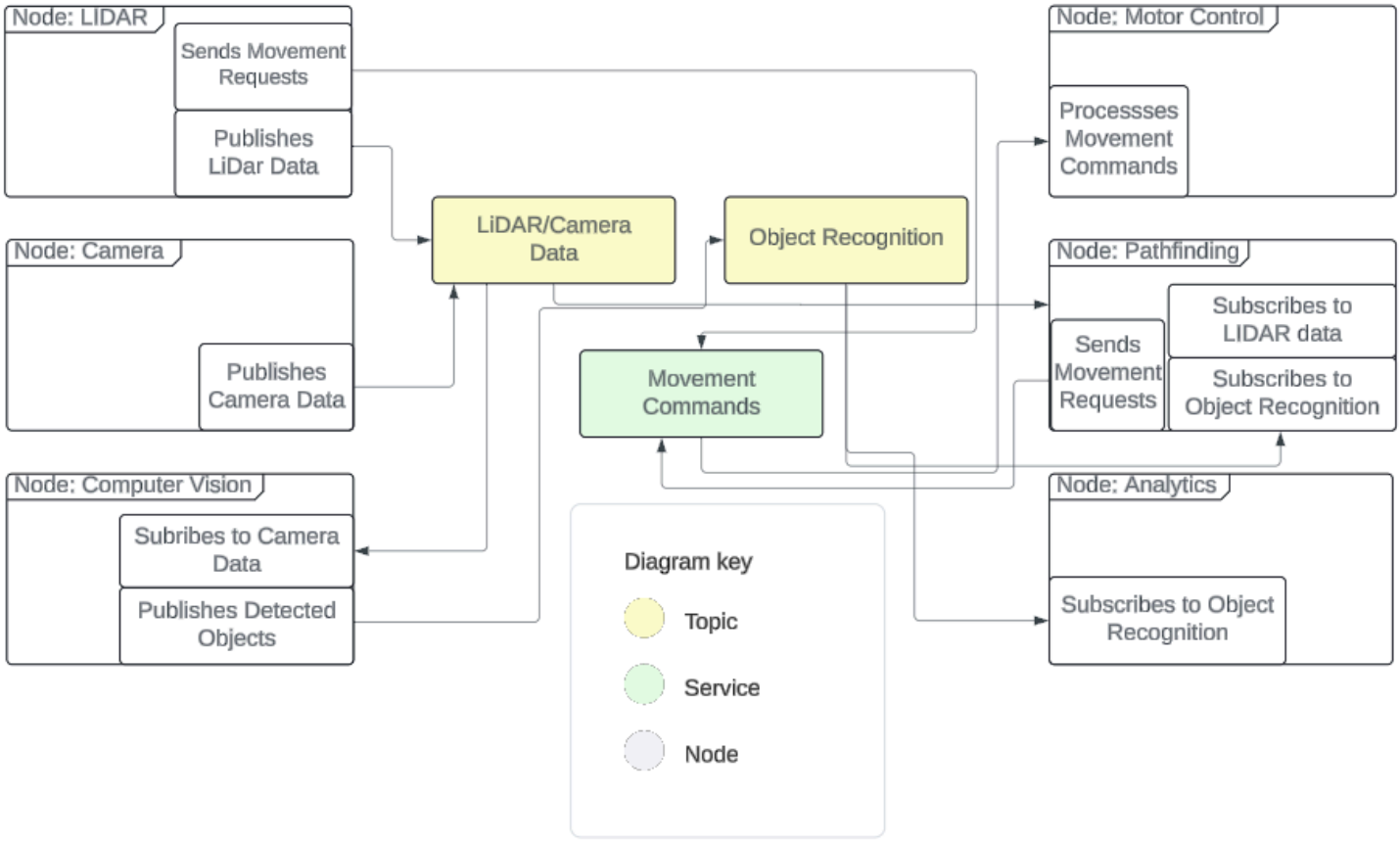

Figure 3: ROS Architecture and Data Flow

ROS: Our system is designed within a Robot Operating System framework. This open-source software is a collection of packages and utilities that are typically built on top of a Linux system to facilitate data transfer and exchange between

different modules of the robot system. ROS standardizes data transfer between Python and C++ programs, allowing them to communicate seamlessly with each other. ROS is the standard across the robotics industry, meaning it will be easier to modify and understand for other maintainers and future field session teams.

Gazebo: To facilitate testing without a physical model, we are employing Gazebo, a software utility that works with ROS to simulate a virtual environment in which our software can control a digital robot. Gazebo comes equipped with a physics engine and emulators for Camera and LIDAR systems, meaning that success within Gazebo can reliably certify our software for use in a real-world environment.

YOLOv5: YOLO (You Only Look Once) is software designed to deliver object recognition in high-speed, high-accuracy results in real-time, that is built on pytorch. This software will be used to recognize donkey manure, with the assistance of Roboflow.

Roboflow: Roboflow is a software designed to allow users to upload images and train machine learning models for object detection. Roboflow will be used to train a model to recognize donkey manure from images and videos. That model will then be used by YOLO, which will be integrated into the rest of the robot's software.

The Construct: The construct is an online application which serves a cloud-based desktop preloaded with Ubuntu Linux and standard ROS2 packages. This software allows our team to use a standard development environment and ensure that our shared repository works at the same capacity across all of our computers. The software is free to use and does not require setting up a local Linux desktop.

Design Challenges

Through the course of development, there have been several design obstacles. The main obstacles have involved the technology and software required for the robot to be able to function in the environment of the donkey shelter. Specifically, the computer vision software and the pathfinding algorithms are the most affected by the challenges of the environment. Another challenge is making the virtual environment similar to the physical environment.

Pathfinding Algorithm: The donkey paddocks are not uniform in shape or terrain, which makes it difficult for the robot to scan the environment, avoid obstacles, and detect manure. There is also a wide variety of obstacles in each paddock, including feeding troughs, toys, rocks, urine spots, and even a golden retriever which the sanctuary employs to greet visitors. Many of the obstacles are moving and the robot will need to constantly avoid them. This will make the pathfinding algorithm and data transfer between sensors more difficult and complex.

Computer Vision Complications: The variation in terrain in the donkey paddocks is especially challenging for computer vision implementations, as object identification has to work consistently in many different environments. Recognizing manure on flat sand and concrete is far easier than recognizing it in tall grass. Moving obstacles in the paddocks will also make image recognition difficult. This movement can be distracting to a computer vision implementation necessitating stricter identification criteria.

Gazebo Implementation: Gazebo is a very complex software that can be difficult to learn. There are also not many pre-made models available, meaning most models for simulating the environment at the donkey shelter needed to be custom built. Simplistic virtual environments can be created without many custom models, but need not simulate the realistic environment. Detailed environments needed to take a lot of time and customization, potentially making complex virtual environments an unrealistic goal.

VII. Software Test and Quality

The majority of our testing has been conducted in the virtual Gazebo environment that we have built. Figure 4 is an overhead view of the most advanced and realistic Gazebo world, which contains realistic fencing (FIgure 5), a model of manure (Figure 6), and a model of an AprilTag (Figure 7). A more basic world with less complex models and four solid walls was also created. The testing for the image recognition software can be done both on-site and with test images on Roboflow. For each functional and non-functional requirement, we have prepared plans to ensure they are tested before being passed to the next team working on the project.

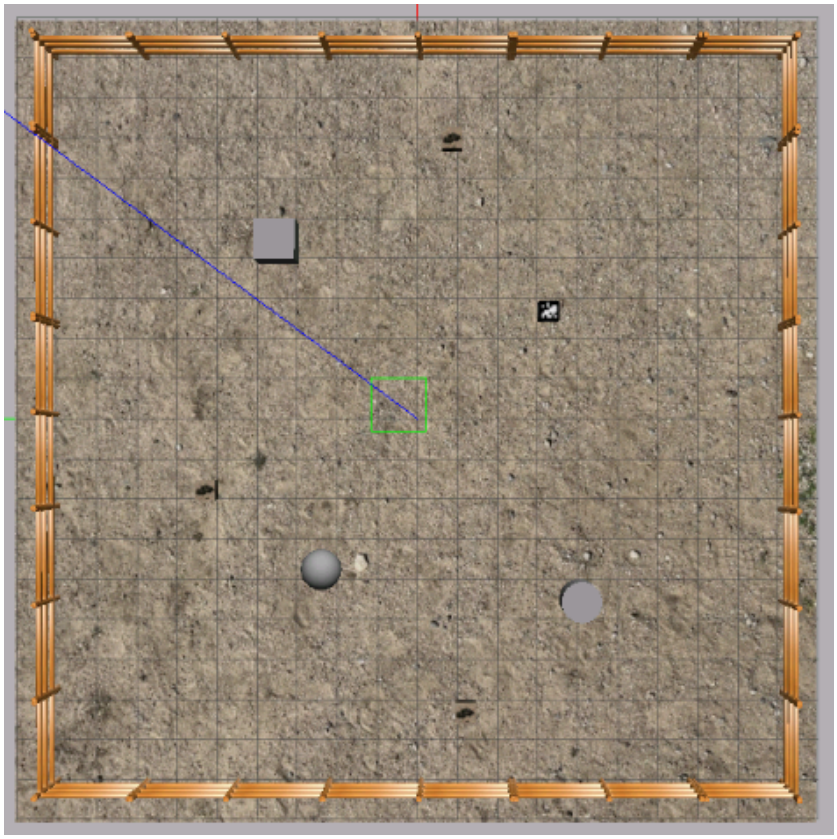

Figure 4: Realistic Gazebo World

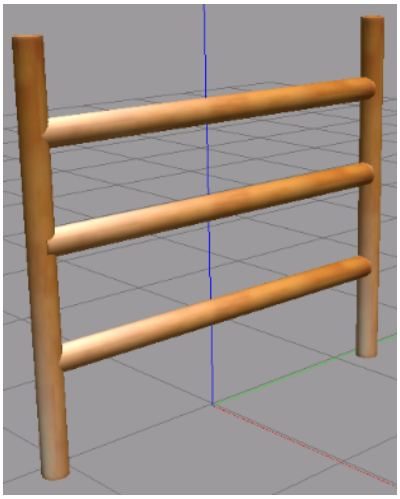

Figure 5: Gazebo Fence Model

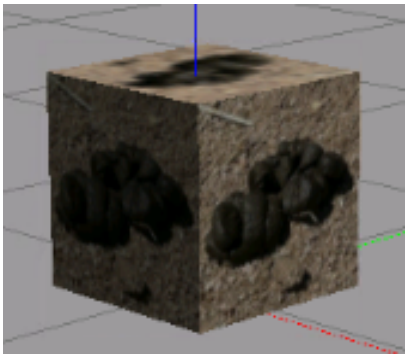

Figure 6: Gazebo Manure Model

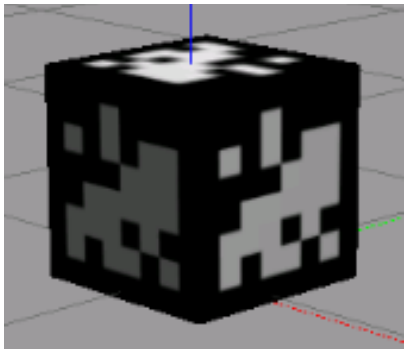

Figure 7: Gazebo AprilTag Model

Functional Requirements

Requirement 1: The robot unit must be able to navigate the paddock independently with minimal guidance.

To test this requirement, we will have the robot set up in the virtual Gazebo environment, but its only impression of this environment will come from its LiDar and camera sensors, rather than a pre-programmed digital map. The robot must be able to use the LiDAR map and camera data to find and traverse a path through the paddock from a start point to an end point without hitting any obstacles. The edge cases will involve moving obstacles and different styles of fencing that may not create a cohesive border. To successfully pass the test, the robot must only be given an endpoint to travel to and be able to navigate any path through the virtual environment without assistance and without hitting a single obstacle.

Robot Independent Navigation Results:

The robot was able to independently navigate through both the basic and the advanced Gazebo virtual simulations when given a destination. It successfully avoided all stationary obstacles. The robot mapped its surroundings using SLAM and navigated using Nav2 only using the map that it generated from LiDAR data. Figure 8 is a screenshot of one of the maps created by the robot as it navigated through one of the basic Gazebo worlds. The red object is the robot, the white lines are empty space where the LiDAR did not hit any obstacles, and the green dots are the edges of obstacles that the LiDAR has detected. The map also updates as obstacles move, also updating the Nav2 navigation instructions so that the robot can attempt to avoid moving obstacles. The robot was not always fast enough to avoid collisions entirely, but would at least stop before a collision.

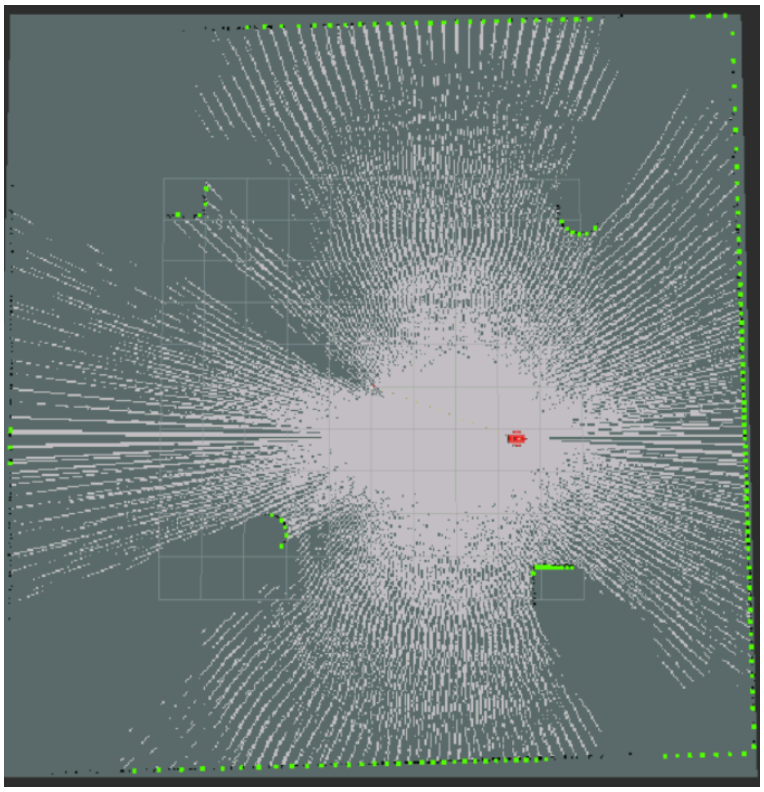

Figure 8: Map Created Using LiDAR Data and SLAM

Requirement 2: The robot unit must be able to scan the paddock for manure piles.

Images of manure were collected in all paddock environments and from several angles to thoroughly train the model. We will also collect images of donkey manure that was not used in the training of the model and use the Gazebo robot's virtual camera feature to test that this feature works properly. This represents a fairly accurate demonstration of the model's abilities, without having to create 3D models of manure. Future testing can be conducted using a simplistic Gazebo model with an image of manure on a cube. The results will be recorded using screenshots and written records. To successfully pass the test, the robot must be able to scan the immediate area using the camera emulator and manure detection and identify the general direction of the closest manure piles.

Manure Scanning Results:

While we were unable to integrate our YOLO model onto the robot with the time constraints, we were able to construct the manure object in Gazebo which can be used in testing by future teams.

Requirement 3: The robot must be able to traverse around 95% of the paddock and pick up all manure therein.

To test this, we can verify after the robot is done scanning how much surface area it covers and how close this value is to the actual surface area of our virtual paddock environment. This will be a nearly complete full run through of the expected behavior of the robot, except to isolate this functionality we will pre-program the manure locations. Edge cases involve obstacles moving while the robot traverses the paddock, checking if the robot will go back to previously blocked locations if needed. The results will be recorded using screenshots and written records. To successfully pass the test, the robot must be able to track how much of the virtual environment it has traversed and reach a minimum of 95% traversal before the end of the test

Paddock Traversal Tracking Results:

Nav2 and SLAM are able to track where in the environment the robot has been and can be used to calculate how much of the paddock has not been traversed. To be sufficiently tested, more software needs to be implemented to monitor the paddock traversal and autonomously control the robot.

Requirement 4: The robot unit must be able to recognize manure in different settings and surroundings including dirt and sand paddocks, grassy pastures, and concrete barn structures with rubber matting.

To test this feature, we first have RoboFlow's built in testing environment which measures how accurately the machine learning model identifies manure in several pre-annotated photos. In the virtual Gazebo environment, we have also replaced the standard gray baseplate with a hay-and-dirt texture to ensure the virtual robot is working in an imperfect natural environment, as it will be in real deployment. We will also conduct another site visit where we can connect our phones to RoboFlow and scan the surrounding area to see the number of correctly identified manure piles and if any were missed or incorrectly identified. Edge cases include manure in grass and poor lighting conditions. The results will be recorded using screenshots and written records. To successfully pass the test, the robot must at least be able to consistently recognize manure on dirt and sand with very little misclassification or failure to identify clear manure piles.

Manure Identification Results:

Virtual tests of the manure image detection using Roboflow on a test set of images went very well. The model identifies manure consistently and with a high level of confidence. Some identifications have a confidence level of 93%. See Figure 9 for one of the test image identifications. Tests were also conducted on-site using phone cameras connected to Roboflow. Manure in different environments and lighting conditions was correctly identified and often identified more than five feet away. It identified manure on both dirt, sand, grass, and hay. See Figure 10 for an example of real time identification. Some weaker areas were discovered: the model struggled with identifying manure in shadows and in the darker corners of the indoor areas. More training photos were taken at any points during the testing where the model struggled to identify manure and were used to strengthen the model in those areas.

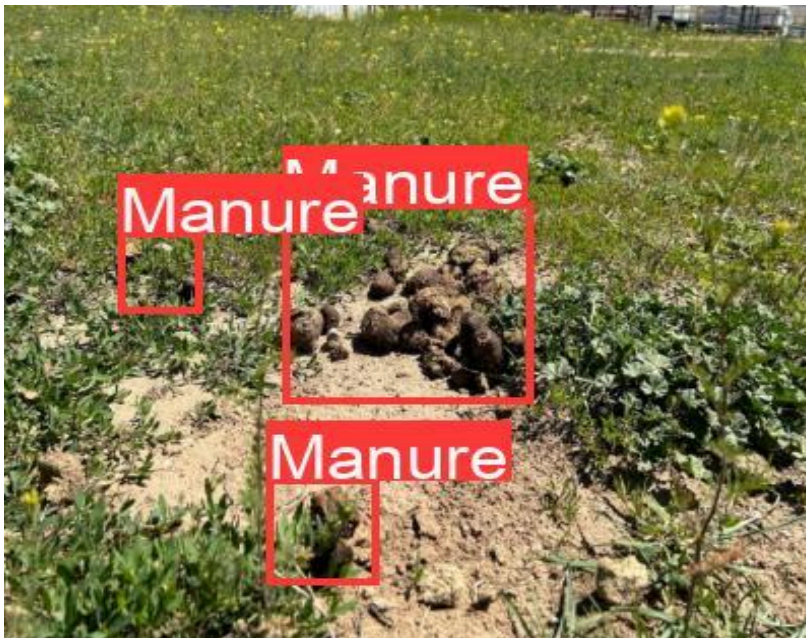

Figure 9: Roboflow Test Image Identification

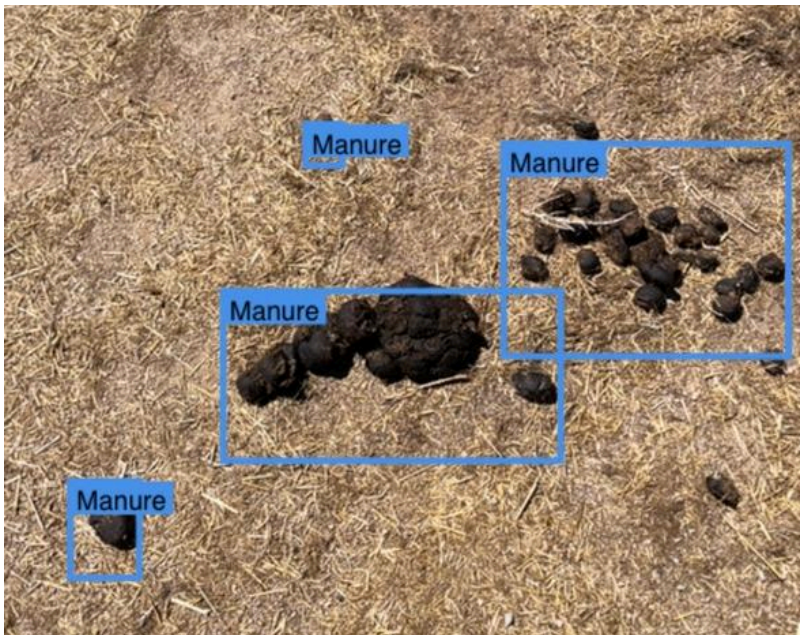

Figure 10: On-site Manure Identification

Requirement 5: The robot unit must avoid collisions with both stationary and non-stationary obstacles.

The ability of the robot to avoid collision will be accurately tested in Gazebo which has an active physics simulation suite. Through visual observation, we can see if the robot is readily avoiding collisions and potentially add scripts to detect and report when collisions with other objects occur. This data will help us sharpen the LiDAR system to avoid future collisions. Edge cases involve several moving objects moving in and out of the robot's initial path plan. The results will be recorded using screenshots and written records. To successfully pass the test, the robot must avoid collision with 95% of obstacles when tested with both stationary and moving obstacles.

Collision Avoidance Results:

The robot was able to successfully navigate around stationary obstacles using SLAM to create a map of the environment and Nav2 to navigate around the known obstacles. For a moving object, the LIDAR system was capable of detecting a fast moving object with an update rate of 20Hz.

Requirement 6: The robot unit must be able to uniquely identify individual paddocks.

Since this feature is mostly a groundwork for future planned statistical analysis systems, testing should only involve the robot's ability to observe an AprilTag and read data on which paddock number is contained within. This will be done in the Gazebo environment, both during other tests and potentially on its own in an isolated testing environment. Edge cases may involve a partially obscured AprilTag such that the robot must move around to get a better view of the tag. The results will be recorded using screenshots and written records. To successfully pass the test, the robot must be able to take in data from an AprilTag that includes a test paddock name or number.

AprilTag Data Intake Results:

Within the AprilTag detection node, the software can also take in a detected AprilTag's ID. This is how information is stored within the AprilTag, but there is currently no defined convention for what kind of information will be stored in the AprilTag. Once the convention is determined, the AprilTag ID can be used to store that data.

Non-Functional Requirements

Requirement 1: This robotic unit's hardware will need to be of reasonable capacity.

The online environment we are using to build this software, The Construct, has features which will allow us to see CPU usage during robot operation. While this is not an entirely accurate reading of how many resources will be consumed by the software in the field, it can help us diagnose spikes and/or potential memory leaks before deployment. We may also run isolated parts of the program or compare the data from The Construct to a Raspberry Pi emulator to validate our choice of hardware. The results will be recorded using screenshots and written records. To successfully pass the test, the software requirements must be compatible with the specifications of an available Raspberry Pi.

Hardware Capacity Results:

With rviz, a joint state publisher, a LiDAR sensor at 60 Hz, and a camera sensor at 60 Hz all running at the same time, the CPU usage, seen using the Linux "top -i" command, in The Construct development studio was 25%. The Construct uses four cores, very similar to many versions of Raspberry Pis. Even while continuing to build more onto the software, Raspberry Pi hardware will be more than sufficient for the software at this time.

Requirement 2: This software will need to employ a camera system.

Gazebo and ROS2 provide plugins for simulating camera input and transferring image data between nodes, allowing us to work camera input realistically into our software and testing before having a physical camera device. The camera system will be tested in a variety of different virtual environments with different lighting conditions. Edge cases would involve very poor lighting conditions and lower quality camera images. The results will be recorded using screenshots and written records. To successfully pass the test, the robot must be able to have a camera emulator in a virtual environment and successfully publish the camera data for other elements to subscribe to.

Camera Visualization Results:

A camera emulator was loaded in Gazebo, connected to a virtual model of the robot, and tested in different conditions. The camera data was sent out correctly as the RGB values of individual pixels and could then be visualized using rviz. The camera received clear images and handled several different lighting conditions.

Requirement 3: The software will need the ability to identify manure from camera intake.

RoboFlow provides a testing suite which automatically judges computer vision models on their ability to match human-provided image annotations. So far, we have reached an identification accuracy of approximately 80.5%, and as we add more data provided by Longhopes we can continue to monitor this number to test how accurate our resulting model will be. RoboFlow will be combined with the virtual camera emulator and environment to test that the accuracy remains the same while using the robot. The results will be recorded using screenshots and written records. To successfully pass the test, the robot must be able to use the image recognition model in conjunction with the camera emulator to identify manure in a virtual environment with an accuracy that is similar to the model when used alone.

Manure Identification in Gazebo Results:

While we were not able to get YOLO integrated onto ROS at this time, this should be the only component needed to achieve our desired test results. The camera's ability to recognize april tags is very promising for this requirement

Requirement 4: It will also need a Light Detection and Ranging (LiDAR) system to give a more detailed map of the robot's surroundings to avoid collision with fences, farm equipment, and the donkeys themselves.

Similar to the camera system, Gazebo and ROS2 include plugins that simulate LiDar input from a virtual robot which will make the job of simulating the real operation of the robot much easier. The accuracy of this LiDar simulation will give us a reasonable outlook on the LiDar system's real-life performance while testing it in the virtual environment. Edge cases involve moving obstacles and different sizes and types of obstacles, as well as different styles of fences while creating the boundary of the paddock. The results will be recorded using screenshots and written records. To successfully pass the test, the robot must be able to run a LiDAR emulator in a virtual environment and successfully publish the LiDAR data for other elements to subscribe to.

LiDAR Visualization Results:

A LiDAR emulator was loaded in Gazebo, as shown in Figure 11, connected to a virtual model of the robot, and tested in different virtual environments. The LiDAR data was sent out correctly as the distance between the sensor and where each individual laser beam hit an obstacle and could then be visualized using rviz. The LiDAR identified the edges of several different objects and had an adjustable range. A visualization from one of the tests is shown in Figure 12.

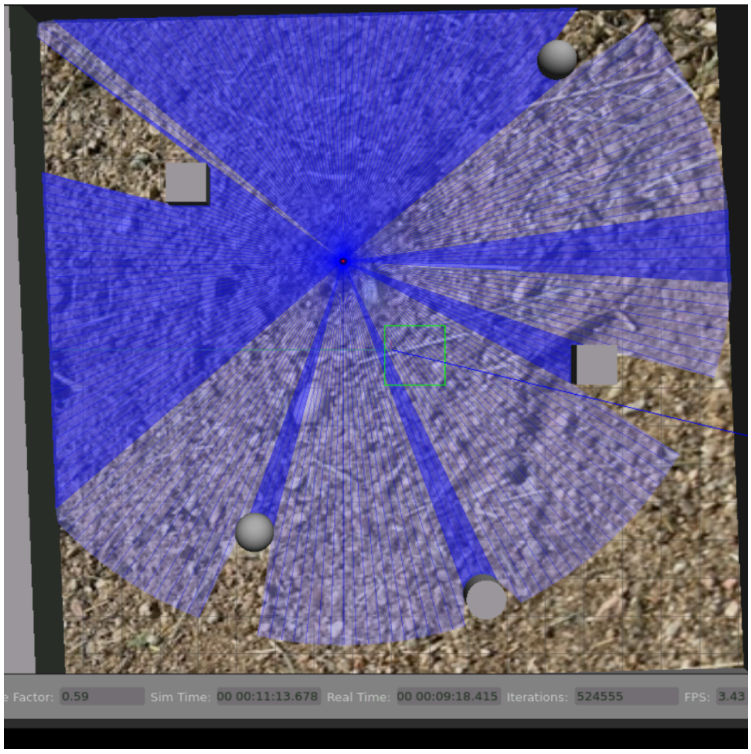

Figure 11: LiDAR Simulation in Gazebo

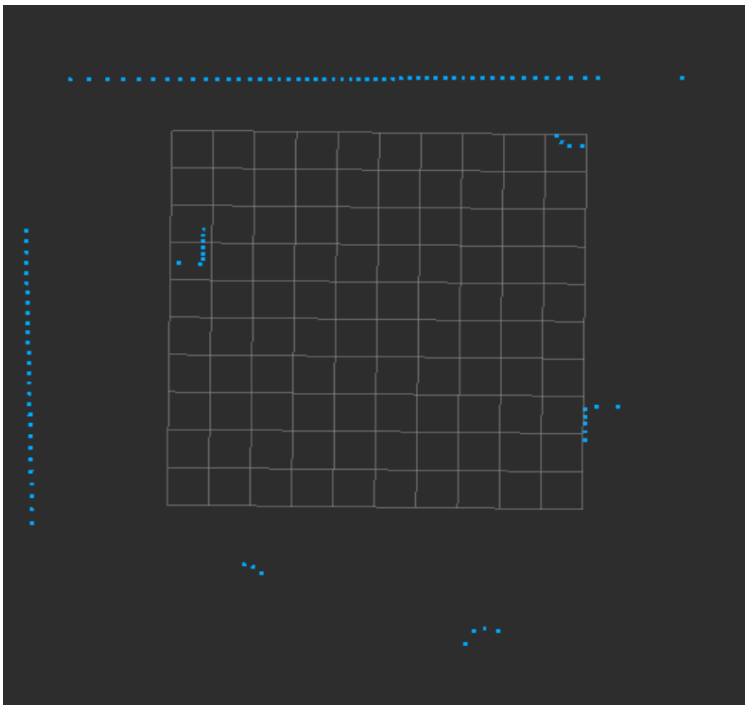

Figure 12: LiDAR Object Detection Visualization

Requirement 5: The software will need strong data transferring ability to ensure these systems can work in tandem to allow the robot to navigate accurately around its surroundings.

To test the system's ability to transfer data quickly and efficiently, we will set the camera in our virtual simulation to have a higher framerate than the planned deployment's camera. This will be a fitting test for ROS2's ability to handle rapid image data transfer, especially since the online interface we use to run this software has strict limitations on CPU usage during runtime. Edge cases would involve all of the systems running at once, even if they wouldn't realistically. The results will be recorded using screenshots and written records. To successfully pass the test, the software functionality must not suffer when multiple nodes and processes are running simultaneously. Software functionality will be measured using run time required to complete tasks and if the tasks are successfully completed.

Data Transference Results:

Tests were done with Gazebo, rviz, a joint state publisher, a LiDAR sensor at 60 Hz, and a camera sensor at 60 Hz all running at the same time. The data transfer between each element was not affected by the amount of processes running and was still efficient and functional.

Requirement 6: A unique identifier will be necessary to identify the manure dropoff location and orient the robot's position within the paddock.

After encoding a unique identifier into the AprilTag used in the Gazebo simulation, we will have to test and ensure that the identifier's paddock ID is consistent across all scans and can be securely stored in the ROS2 system for future use. Edge cases would involve a partially obscured AprilTag and moving the AprilTag to different locations in the same virtual paddock before each test. The results will be recorded using screenshots and written records. To successfully pass the test, the robot must be able to search a virtual environment for an AprilTag and be able to identify

the AprilTag. There is no time limit at this stage, but future testing could require the robot to be able to locate the AprilTag within a certain time frame.

AprilTag Identification Results:

The robot was able to identify an AprilTag in the virtual environment, but only when it was well lit, as shown in Figure 13. The detection system is able to locate the corners of the AprilTag and draw a green box around it as well as locate the center, marked with a red dot. It is unfortunately unable to identify an AprilTag if it is partially obscured, so software will have to be built such that the robot moves around the paddock until the AprilTag for the drop off location is fully visible and able to be detected. The robot is not yet able to search the paddock specifically for an AprilTag, but an AprilTag can be detected directly from the camera system in good lighting conditions.

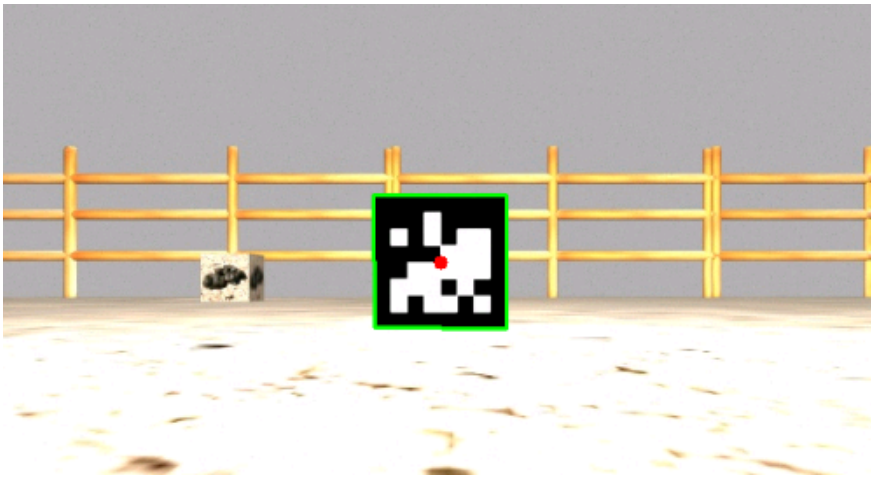

Figure 13: AprilTag Identification

VIII. Project Ethical Considerations

Our project has several ethical considerations involving both the development and deployment of the product. The following are the most pertinent or most likely to violate ACM and IEEE principles:

ACM

1.2 Avoid harm.

The robot must be able to recognize objects that are not manure to avoid collisions. This principle is in danger of being violated because of the complex environment that the robot will be deployed in that includes working around live animals that may be harmed if the robot does not function properly. If this principle is violated, it will mean that the robot is not safe to deploy around animals or potentially even people and would need additional work done before being re-deployed.

1.5 Respect the work required to produce new ideas, inventions, and creative works, and computing artifacts.

This principle is very important during development. Appropriate time must be allotted to be able to do tasks properly and not cut corners, to avoid complications in future work. This principle is important to ensure the best ideas, inventions, creative works, and computing artifacts get heard. If this principle is not followed the end product will likely be of low quality.

2.4 Accept and provide appropriate professional review.

This is another principle that is important during the development process. The group must provide constructive criticism must be able to be given appropriately and respectfully while work is being reviewed. This principle must be followed to ensure that all work done can be appropriately reviewed. If this principle is violated the end product may not be of high quality and would fail to take into account different perspectives and opinions.

2.6 Perform work only in areas of competence.

This principle is the most in danger of being violated as none of the members of the team have previous experience using most of the technology required for this project. The team must be honest with themselves and others and will not attempt to implement any work without gaining knowledge of all the edge cases. If this principle is violated it is likely that the project's software will not be high quality or effective.

IEEE

1.01. Accept full responsibility for their own work.

The principle is important to the project and must be followed during the entire development process so as to avoid issues within the team and make sure that appropriate contributions are being made. Team members are not allowed to blame or give praise to others who haven't had an impact on the work. If this principle is violated it could lead to degradation within the team and a lower quality product.

1.02. Moderate the interests of the software engineer, the employer, the client and the users with the public good.

This principle is very important for the development process. All interests of each teammate and the client will be taken into account before making any further decisions. If this principle is violated, the product will not adhere to the client's requirements.

1.04. Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.

This principle is important for both the development and deployment as well. If any person, animal, or object could be harmed, it is imperative that the appropriate people and organizations are informed of the danger. If this principle is violated, accidental harm could be caused to people, animals, or property.

3.07. Strive to fully understand the specifications for the software on which they work.

This principle is important for the development process. The team must make sure that any software used in the projects works correctly, and ensure that the software is understood fully. If this principle is not followed the end product will be lower quality and difficult to be expanded upon or inherited by future developers.

Micheal Davis' Tests

Michael Davis is a philosopher who focused on ethics in professional environments. During his career, he developed a set of seven analytical tests to determine whether or not an ethical decision was made correctly. Among these, two tests stand out as relevant to our project and the decisions we have had to make during its completion.

1: Reversibility Test. Davis' reversibility test involves considering those affected adversely by an ethical decision and considering if they would make the same decision regardless. In our case, the robot we are creating stands to take jobs away from those who may otherwise be employed. However, the employees of the sanctuary have other tasks that they

can complete and will likely not be out of a job entirely. If members of our team were employees at the donkey shelter, the autonomous paddock mucking robot would still be a good option and ultimately be beneficial to the shelter.

2: Colleague Test: Davis's colleague test requires consideration of colleague opinions on the project and the decisions made therein. In our case, we have been working with several classmates and professors on this project and all seem enthusiastic about the prospect of our robot and its ability to aid the Longhopes Donkey Shelter's charitable mission. They have not raised any immediate ethical concerns and support our planned solution.

IX. Results

The software is able to successfully identify manure from test photographs, detect AprilTags in photographs and from the robot's camera system, and create a map of the environment using a combination of SLAM and Nav2. The robot can also be given a desired destination on the map and travel to it independently, avoiding stationary obstacles. The robot attempts to avoid moving obstacles, but if the obstacle is moving faster than the robot, collision may not always be avoided entirely. Several different virtual environments have been created in Gazebo to facilitate the testing of the different features, including a more realistic world with accurate fencing, simplistic manure models, an AprilTag model, and accurate ground terrain.

Functional Requirements

Requirement 1: The robot unit must be able to navigate the paddock independently with minimal guidance.

Using SLAM and Nav2, the robot is able to be given a destination point and independently navigate around obstacles and travel to the given location.

Requirement 2: The robot unit must be able to scan the paddock for manure piles.

With the robot's camera, image recognition is able to successfully identify manure in good lighting conditions. As the robot traverses the paddock, it is able to use the camera to scan for manure.

Requirement 3: The robot must be able to traverse around 95% of the paddock and pick up all manure therein.

Using SLAM, the robot can remember where in the paddock it has already been and ensure that it is covering at least 95% of the paddock. As the robot moves, it is also constantly using the camera to look for manure piles, so any piles in its path will be detected and picked up.

Requirement 4: The robot unit must be able to recognize manure in different settings and surroundings including dirt and sand paddocks, grassy pastures, and concrete barn structures with rubber matting.

The image recognition system works on a variety of terrains and has been successful during testing on dirt, sand, short grass, and well lit areas of the barn on rubber mats.

Requirement 5: The robot unit must avoid collisions with both stationary and non-stationary obstacles.

The robot successfully avoids collisions with stationary objects. With moving objects, if the robot is moving slower than the moving object, they may collide, but the robot will detect the obstacle and stop moving and attempt to change directions. This functionality may be improved by increasing the robot's speed, but at this stage in development simply stopping to avoid collisions is sufficient.

Requirement 6: The robot unit must be able to uniquely identify individual paddocks.

The software for detecting AprilTags can also read the AprilTag's ID. The ID can encode information about the paddock as desired, including a paddock number, so the robot will be able to uniquely identify paddocks.

Non-Functional Requirements

Requirement 1: This robotic unit's hardware will need to be of reasonable capacity.

The software is still lightweight enough that it would be able to be hosted on a Raspberry Pi. A Raspberry Pi is the ideal hardware for this robot as it is inexpensive and has enough ports for the robot's motors without being excessive.

Requirement 2: This software will need to employ a camera system.

The camera system has been created using ROS and emulated by Gazebo and rviz. It should be fairly simple to connect a real camera to the software and retain all the functionality.

Requirement 3: The software will need the ability to identify manure from camera intake.

The image recognition software is able to successfully identify manure in several different environments and software to take in the data from the camera system and transform it into a jpeg image is completed. Roboflow is currently not implemented in ROS..

Requirement 4: It will also need a Light Detection and Ranging (LiDAR) system to give a more detailed map of the robot's surroundings to avoid collision with fences, farm equipment, and the donkeys themselves.

The LiDAR system has been created using ROS and emulated by Gazebo and rviz. It should be fairly simple to connect a real LiDAR system to the software and retain all functionality. Using SLAM in conjunction with the LiDAR data, the robot is able to create maps of its surroundings.

Requirement 5: The software will need strong data transferring ability to ensure these systems can work in tandem to allow the robot to navigate accurately around its surroundings.

ROS is designed to allow for easily transferable data and having multiple nodes running and communicating at once has not affected the run time or performance of the system.

Requirement 6: A unique identifier will be necessary to identify the manure dropoff location and orient the robot's position within the paddock.

Using the camera system, the robot is able to detect AprilTags which can then be used to mark the drop off location. The robot will find where the drop location is and mark it on the map it creates using SLAM to be able to return to the drop location with manure.

X. Future Work

The future Fall Field Session team will be able to continue to refine the software and build on more features and control of the robot. Some features of the robot that work independently still need to be connected in order to create the desired functionality as well. The Fall Capstone Design team must design and construct the physical robot and work with the Fall Field Session team to integrate the physical robot with the software. The following is the known future work that needs to be done by either one of both of the Fall teams:

- The Nav2 and SLAM mapping system needs to be expanded to be able to create a cohesive boundary around the edge of the paddock even if the LiDAR cannot see the horizontal slats of the fence and only detects the vertical posts.
- The image recognition software that detects manure must communicate with Nav2 to update the desired destination of the robot.
- The AprilTag detection software must communicate with Nav2 and SLAM so that the robot can independently identify the coordinates of the drop location and store that location in memory.
- A new node must be created that communicates with Nav2 to track the robot's progress in the paddock and ensure that the robot covers at least 95% of the paddock and tells the robot to shutdown when enough progress has been made.
- A new node must be created that communicates with the manure detection node and the camera to appropriately position the robot to pick up a pile of manure.
- A new node must be created that controls the robot's arm to pick up and dump the manure.
- A new node must be created that communicates with Nav2 once a pile of manure has been picked up to tell the robot to go to the drop location.
- A new node must be created that communicates with the AprilTag detection node in order to scan the data from the AprilTag and store it in memory as appropriate.
- The image recognition model can be further refined with more training images, should the future teams deem that necessary.
- The virtual model of the robot can be further refined and altered to more closely resemble the physical robot as the future Capstone Design team plans out the specifications.
- The manure detection model from Roboflow must be integrated into the ROS system to run independently without being connected to the internet.

XI. Lessons Learned

We have learned a lot over the course of this project. Aside from the technical knowledge that we have gained involving ROS and similar systems, we have also learned several lessons that can be applied outside of this project. One such lesson is that doing proper research, especially for tasks that you are unfamiliar with, can save several hours of work and increase the end product's quality and stability. Finding experts that are willing to give advice is also incredibly helpful and can be a good starting point for further research. The first steps of a new project should always be to reach out to experts and mentors to look for starting points and advice, then sitting down to do thorough research and planning before actually starting to work on the project. This way, when the work actually begins, there will be a clear direction and the technology will be more familiar and easier to work with. We also learned a lot about using open source software and that it can be difficult for beginners that are not familiar with the software as the documentation can often be fragmented or unclear. Because of this, we made sure to thoroughly document our work and created easy to follow guides for future teams so that they can hit the ground running without needing to dig through fragmented documentation. Relying on certain software or online applications can also sometimes lead to bottlenecks during development due to the limitations of the software or applications. We used The Construct's virtual desktop development studio, but this meant that the simulations often ran quite slowly as the server is hosted overseas. We also learned that plans can change quite quickly at the beginning of projects, especially when working with unfamiliar software. Our initial impressions on how quickly we could accomplish tasks and how far we could get on the project changed quite a bit once we started doing more research and began working. However, we are still very proud with the work we were able to accomplish and how far we have taken this project.

XII. Acknowledgments

We would like to thank our client, Longhopes Donkey Shelter, for providing this project and their guidance. Kathy Dean, the shelter's president and founder, and Victoria Schroeder, the shelter's executive director, were particularly involved and very helpful. We would also like to thank our advisor, Donna Bodeau, as well as all of the advisors, for being a sounding board and providing ideas and guidance.

XIII. Team Profile

Kara Schuler

Undergraduate student Computer Science: Robotics and Intelligence Systems

Born in Franktown, Colorado

Favorite Longhopes donkey: David

Dan Collins

Undergraduate student in Computer Science: Robotics and Intelligent Systems

Born in Denver, Colorado

Favorite Longhopes donkey: Stuart

Shane Gordon

Undergraduate Student in Computer Science: Robotics and Intelligent Systems

Born in Walnut Creek, California

Favorite Longhopes donkey: Luna

References

Appendix A – Key Terms

Include descriptions of technical terms, abbreviations, and acronyms

Term	Definition
LiDAR	*Light Detection and Ranging - A device that casts lasers in an arc around itself to generate data on surrounding obstacles and their distances from the unit.*
Machine Learning (ML)	*A software subsystem that examines images and data to identify objects and patterns in them automatically. These systems must be trained beforehand to identify such objects*
Computer Vision	*A subject of computing that addresses a computer's ability to observe and draw conclusions about the physical environment around itself.*
Gazebo	*Gazebo is a software program that allows one to simulate a robot in a virtual environment. Several packages allow the Gazebo 'world', or virtual environment, to interface with the ROS data network allowing our software to interact with a virtual robot exactly as it would with a potential physical model.*
rviz	*rviz is a software program that is used to visualize robot models, camera data, LiDAR data, and other types of data. It is commonly used in conjunction with Gazebo to visualize data from and about the virtual robot model.*
ROS	*ROS stands for Robot Operating System*, a series of open-source packages that facilitate data transfer and synchronization across several robot components. In particular, our software uses ROS2 Humble for stability and modernity in our construction.
YOLOv5	*YOLOv5 stands for You Only Look Once, and it can recognize objects in images/videos in high-speed, high-accuracy results in real-time.*
Roboflow	*Roboflow is a software that allows the training of objects from images to be recognized.*
SLAM	*SLAM (simultaneous localization and mapping) is a method for robots to construct a map of their environment while tracking the robot's own location at the same time. Our implementation uses slam_toolbox, a standard base for setting up SLAM systems on any ROS2 robot.*

Nav2	*Nav2 is ROS robotic navigation software that supports independent navigation of a robot in complex environments.*
AprilTag	*A unique identifier that is similar to a QR code but stores less data.*