CSCI 370 Final Report

Muck Busters

Andrew Bernklau
Rhys Fernando
Julia Luoto
Lindsey Shanahan

Kathy Dean

Revised December 6, 2024

CSCI 370 Fall 2024

Donna Bodeau

Table 1: Revision history

Revision	Date	Comments
New	August 23, 2024	Started the document and made initial drafts for the sections due on 9/1/2024
Rev – 2	August 28, 2024	Revised sections I-V for better coherence and clarity
Rev – 3	September 11, 2024	Added content to Section IV System Architecture and Team Profile
Rev – 4	September 15, 2024	Revised Team Profile and System Architecture sections
Rev - 5	October 11, 2024	Added content to section VII Software Test and Quality and VIII Project Ethical Considerations
Rev - 6	October 18, 2024	Revised sections VII and VIII, added in figure numbers
Rev - 7	October 28, 2024	Added content to sections IX, X, and XI
Rev - 8	November 11, 2024	Added content to sections XII, XIII, References, and Appendix
Rev - 9	November 20, 2024	Revised all sections to begin preparation for final submission
Rev - 10	December 6, 2024	Completed final revisions based on peer feedback

Table of Contents

I. Introduction

The goal of this project is to develop software for an autonomous paddock mucking robot for the Longhopes Donkey Shelter. The shelter is in Bennett, Colorado, founded in August 1999. The shelter's purpose is to rescue and rehome donkeys that would otherwise be slaughtered. The shelter then cares for the donkeys for the remainder of their lives. The shelter is run by staff and volunteers that care for a substantial number of donkeys, and one of the tasks they have is to clean out the paddocks daily. To improve efficiency, reduce costs, and allow the workers to be able to focus on other tasks, the client requested an autonomous paddock mucking robot that would take over part of the responsibility of mucking the paddocks.

The robot needs to navigate through the paddocks and be able to avoid obstacles, such as food and water troughs, toys, donkeys, poles, bushes, and fences. The robot scans the paddock for manure, determines a path to get to the manure and travels there, picks up the manure, and deposits the manure at a predetermined drop location inside the paddock. This project was started by the past summer 2024 field session and was continued alongside the EDNS 490 capstone team. The focus is on developing the robot's software, which will be implemented by the capstone team in a subsequent phase.

The client of this project is the Longhopes Donkey Sanctuary. There is also the possibility of future clients if the robot is manufactured for use by other shelters to improve their efficiency and cut their costs. Overall, the software created by the team throughout this project needs to be sufficient to improve the efficiency of the shelter and others like it for the sake of all these stakeholders.

II. Functional Requirements

The functional requirements are based on the design features that must be included in the robot. The robot must be able to move around the donkey paddock without running into anything, including the donkeys and other obstacles. This can be broken down into identifying obstacles and then physically avoiding them. Additionally, the robot must find the manure, correctly pick it up, and then move it to the drop-off location. This step is possible after the robot can correctly identify the boundaries of the paddock, traverse most of the paddock, detect manure and the drop-off location, and know when the robot has finished or needs staff assistance. A benchmark of 95% was chosen to ensure that the robot would be operating effectively and to avoid any potential problems.

1. The robot must be able to correctly traverse and path through 95% of the paddock.
2. The robot computer vision model must be able to identify manure with 95% accuracy.
3. The robot must be able to stay within the defined bounds of each paddock.
4. The robot must be able to notify staff when it is working and when it is finished with its task (i.e., must have an implemented "finished" state).
5. The robot must be able to correctly avoid obstacles with a 95% accuracy rate.

III. Non-Functional Requirements

The non-functional requirements are constraints placed on the design and implementation of the robot to maintain the functional requirements. To create a robot that can complete the functional requirements listed above, the robot must have hardware and software to support its needs. All while keeping the cost under the budget. The robot operates outdoors and indoors, and needs hardware that allows it to see its surroundings, know where it is, and identify the manure drop-off location. The required software must identify manure and obstacles based on information from the sensors.

1. This robots' hardware needs to be of reasonable capacity to handle these requirements:

    a. It must have enough memory capacity to control the robot and run the computer vision model and navigation software.
    b. It must be able to cool itself, as it operates indoors and outdoors.
2. The budget for the entire robot is $5000.
3. This robot needs to employ a LiDAR system to map its surroundings.
4. The model needs the ability to identify manure based off information from a camera.
5. A unique identifier is necessary to identify each paddock, ensuring the robot operates within the bounds of that specific paddock.

IV. Risks

There are several risks associated with this project, particularly due to the safety needs for the animals involved. One significant concern is the potential for interaction between the donkeys and the robot. This presents a dual challenge: on one hand, the donkeys may play with or interfere with the robot, potentially damaging it; on the other hand, if the robot fails to accurately detect and avoid the donkeys, there is a risk of causing harm to the animals.

Secondly, there are also operational and software risks with the project. A software crash rendering the project immobile necessitates fixing the robot which takes valuable time away from the client. Furthermore, operational hazards like mud, rain, terrain differences, and obstacles could all cause potential harm to the robot.

In addition to the previously mentioned, virtual testing and real-world environments also play a role. The virtual environment which is used for testing might not perfectly simulate the real-world. This could lead to inaccuracies in testing which are hard to account for without physical hardware to test with.

Another possible risk is that the software team's code needs to be integrated with the physical hardware. This can present a challenge because the software team will not be working on the project when this step happens. This might slow the progress of the project and may cause a delay to the final product delivery date.

Lastly, as the client wishes to expand this robot into a commercial product, the ethical concern of job loss is a relevant concern. This robot can take away roles in the animal enclosure clean-up space, which is a risk of this project.

V. Definition of Done

For the Fall 2024 semester, the project is considered complete once the team has identified the necessary sensor hardware, improve the computer vision model that enables the robot to identify donkey manure, devised a solution to determine paddock boundaries, and developed the code required for the robot to navigate the paddocks effectively.

The robot reliably identifies donkey manure within the paddock, as its primary objective is to transport the manure to the designated drop-off location. This requires the ability to distinguish manure from other objects in the paddock. Accurate environmental mapping is critical to allow the robot to make decisions on when to stop, navigate around obstacles, or avoid crossing paddock boundaries.

To optimize pathfinding, the algorithm prioritizes efficiency and simplicity by focusing on the shortest viable path between the robot and manure while avoiding obstacles and remaining within the paddock's limits. To reduce computational complexity, the robot stores GPS coordinates of previously cleaned areas, piles of manure, and the drop-off location. This minimizes the need for full path recalculations.

Obstacles are managed with a proximity-based avoidance system, responding dynamically only when objects are within a set range. Upon completing its cleaning task, the pathfinding algorithm guides the robot to the nearest fence to ensure it remains out of the way before shutting down.

To test these functionalities, the team created a virtual environment with multiple simulated paddocks, exposing the robot to a variety of terrain shapes, obstacle arrangements, and scenarios.

VI. System Architecture

The system architecture for the robot is made up of two key components: the pathfinding algorithm and the software architecture. The pathfinding algorithm contains all necessary steps for pathfinding, and the architecture displays how the sensor data input flows.

Pathfinding Algorithm:

The steps and flow of the pathfinding algorithm are shown in Figure 1.

The first step in the algorithm is to localize the robot by using the RTK GPS to find its geographical position in the world. After the robot is localized, it uses a feature called geofencing to map the boundary lines, creating a polygonal boundary to ensure it stays within the bounds of the paddock. Afterward, the robot identifies the location for a drop site based on previous GPS mapping.

Once the robot has completely oriented itself within the paddock and geolocated the major points of interest, the robot then maneuvers around the paddock identifying nearby manure piles. The robot prioritizes efficiency by adjusting its movement to reduce travel distance and conserve power. Instead of randomly searching for manure, the robot employs a strategy that minimizes driving time by focusing on manure piles closest to its current location. Each manure pile has its GPS coordinates identified then stored on a manure map. This map allows the robot to path find effectively.

Then the robot calculates the optimal path to each nearby manure pile, using a path-finding algorithm to compute the shortest route between the pile. This ensures it covers the minimal distance needed to collect and transport manure to the drop site. If new piles are found during collection, they are immediately added to the map in a way that minimally disrupts the path.

After clearing the manure, the robot checks its battery level. If the battery is sufficient, it continues to the next closest manure and repeats the process. If the battery is low, it returns to the drop site or a charging station before resuming its task, further optimizing to reduce unnecessary travel. And if 95% of the paddock has been traversed for manure removal, the robot stops and emits a flashing light indicating it has finished.

This approach reduces driving time and power consumption by prioritizing routes that require the least movement while ensuring the robot completes its tasks efficiently.

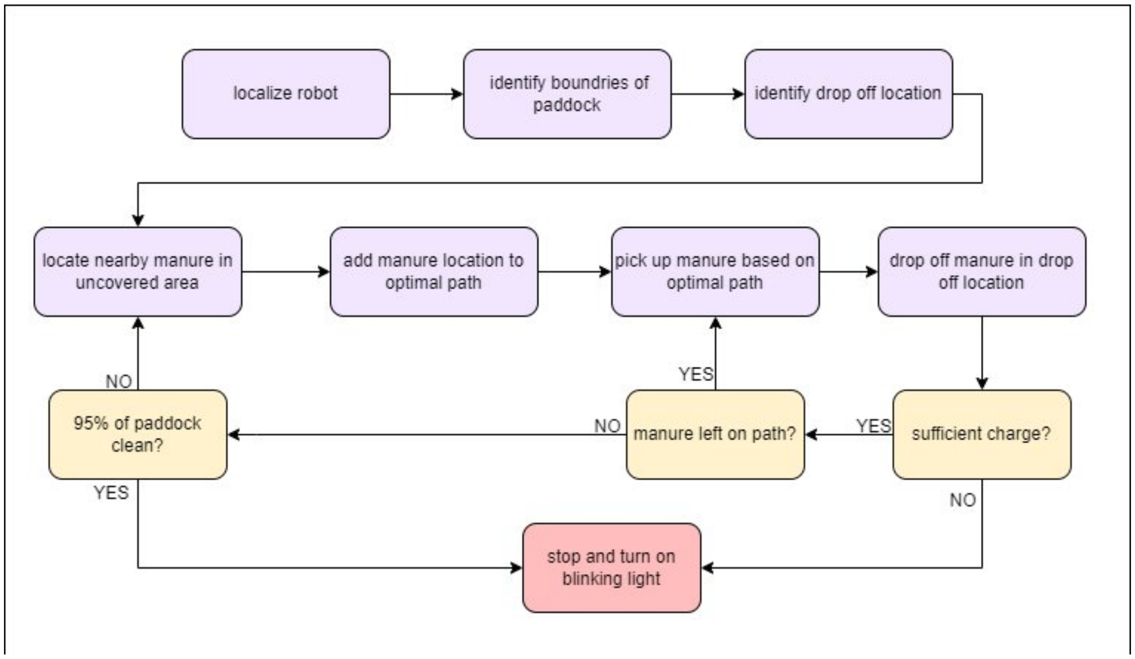

localize robot
identify boundries of paddock
identify drop off location
locate nearby manure in uncovered area
add manure location to optimal path
pick up manure based on optimal path
drop off manure in drop off location
NO
95% of paddock clean?
YES
YES
NO
manure left on path?
YES
sufficient charge?
NO
stop and turn on blinking light

Figure 1: Algorithm for Robot Movement

Sensor Data/Software Architecture:

The software architecture, as seen in Figure 2, has many different components that work together to operate the robot. These components are:

ROS: The system was designed within a Robot Operating System (ROS) framework. ROS is an open-source software that has a collection of packages and utilities run on a Linux system which facilitates data transfer between the modules of the robot system. ROS is also the standard in the robotics industry, so it is easy to modify and understand for future users. In Figure 2 below you can see how we utilized ROS nodes, services and topics to build our software architecture.

Testing: For testing, Gazebo is used since there is not a physical model to test on yet. Gazebo is a software utility that works with ROS to create a virtual environment where the software can control a digital robot. It comes already equipped with a physics engine and emulators for the Camera and LIDAR systems, so success in Gazebo can certify the software for use in a real-world environment.

Computer Vision: The object recognition is completed using YOLOv5 and Roboflow. YOLO (You Only Look Once) is a software built on PyTorch that delivers object recognition in high-speed, high-accuracy results in real time. Along with Roboflow, YOLO is used to identify donkey manure. Roboflow is a software that allows users to upload images, annotate those images, and create a data set. A model is then created using the Roboflow dataset in the necessary YOLOv5 PyTorch format. This model is used for the identification of donkey manure and integrated within the rest of the robot's software.

Microprocessor: The microprocessor chosen needs to satisfy CPU and GPU requirements. The robot requires CPU to take in and process sensor data, such as the LiDAR, and to map and travel the paddock. The robot requires GPU to use computer vision and run the machine learning model to identify objects and donkey manure. A Jetson Orin NX is the microprocessor chosen for this robot.

GPS: GPS (Global Positioning System) gives real-time information about an object's current position. GPS is used to identify where the robot is in the paddock and where the robot has been. Markers are also placed, letting the robot identify where the manure's drop-off location is and where the paddock's boundaries are.

Camera: The camera is used to enable the computer vision model to identify the manure in the paddock. It is necessary to be able to see the manure so that we can detect it and navigate to it using the pathfinding algorithm.

LiDAR: LiDAR (Light Detection and Ranging) sends a laser light from a transmitter and receives the reflection of that light to identify where objects are. LiDAR is used to identify where obstacles are in the paddock and ensure that the robot does not run into them.

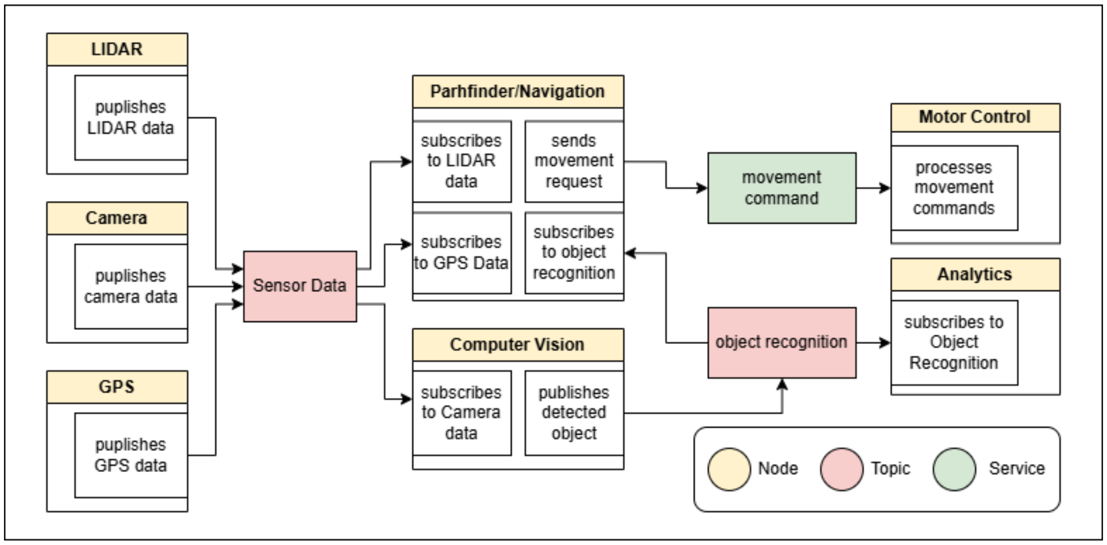


Figure 2: Software Architecture Design

Design Challenges:

The development of an autonomous paddock-cleaning robot presents several significant design challenges that have influenced the development process. These challenges span from accurately identifying boundaries, detecting various types of obstacles, implementing efficient pathfinding algorithms, and compensating for the absence of a physical robot during the testing phase. Each challenge introduces unique complexities that must be carefully considered and addressed to ensure the robot's effectiveness in its environment. The following sections explore these challenges in detail and the potential solutions and approaches being explored to overcome them.

Boundary Recognition

The robot's task is to autonomously navigate paddocks while maintaining its location inside the paddock. This involves recognizing the boundaries in which it should reside. The boundaries in this case are paddock fences of many varieties (wood, chicken wire, metal). Each different type of fence adds complexity to the issue. Yet, it is crucial the robot understands its boundaries for safety and proper functionality. This specific functionality requires the robot to accurately detect boundaries regardless of type, avoid misinterpreting other objects as fences, and detect them – all in real time.

In approaching this problem, a few avenues have been explored.

One approach is to use LiDAR, which is a common technology used in robotics for detecting robot surroundings. It emits light and detects when the light reflects to it. Due to certain fencing features, several fences at the donkey shelter cannot be detected well since some fences are made up of wire and the probability of light reflecting off the wire is low.

Another approach is to use computer vision and depth data, having the robot recognize the different fences it meets and use the depth data to keep a distance from the fences. The disadvantages of this approach are that an incredible number of images are needed for training the model to recognize all the different types of paddock fences, and the camera alone has difficulty detecting objects in dim light or areas of shade.

The final approach considered was using GPS data. GPS can be used to set the coordinates of the boundaries when the robot enters a paddock. There are two options for different accuracies: regular GPS or RTK GPS. Regular GPS would provide 1-3m of accuracy, which if accounted for, could be sufficient. RTK GPS would require a base station which would add to the price and complexity of the system but would bring in an accuracy within 1-2cm. When considering this problem, it is best to also note that GPS needs a Wi-Fi or cellular connection. Thankfully, Longhopes Donkey Shelter has Wi-Fi that can serve as the connection point.

GPS with RTK is the final solution the team has landed on.

Obstacle Detection

The various obstacles within the paddock include donkeys, donkey toys, volunteers, troughs, and barns. The robot must detect an object that is not manure and avoid it. This proves challenging as the robot needs to process the various environments and avoid the many different objects put in its path. Moving obstacles in the paddocks is especially difficult for the image model to process, as movement can be distracting to a computer vision implementation causing stricter identification criteria.

To address this issue, the model is only used for detecting manure and everything else is an object to avoid. This simplifies the challenge significantly by providing the model with a single target to focus on, reducing the complexity of the classification task. It also greatly reduces the amount of training data necessary since we only need to detect manure. This also means a detailed dataset for manure is needed. LiDAR is used to detect where all other obstacles are in the paddock.

Pathfinding Algorithm

When gathering manure, the robot must be capable of autonomously identifying the best route to the manure drop-off location and then the next pile of manure. Since the donkey paddocks are not uniform in shape or terrain and are typically filled with obstacles – some roaming and others stationary, a robust and dynamic algorithm is necessary to keep the robot from bumping into them. This algorithm may become complex and hard to maintain the more it must consider when calculating a path.

To simplify the process, the robot begins by calculating its full path to traverse the whole paddock and when moving, track the areas it has already cleaned. Key points within paddock, such as the drop-off location and piles of manure it has come across, are stored as GPS coordinates to reduce full path recalculations. These coordinates are just simply detours that the robot has to take before returning to its path. Lastly, a simple avoidance system is implemented that responds to obstacles only when they are within a certain range.

Physical Robot

Throughout the development process the software team has been without a physical robot to test. This is a massive complication. A physical robot to test would provide the team with valuable information on how the algorithms and model work in real-world environments, where they may fall short when detecting an object, or where it might do well in recognizing boundaries. Thus, testing requires a virtual environment to supplement the lack of a physical robot.

The virtual environment is set up in Gazebo. It provides a highly realistic simulation environment where robots can interact with objects, obstacles, and sensors, allowing developers to test robot behavior without the physical hardware. This is incredibly helpful, yet developing more realistic environments is time consuming and requires extensive customization, making a complex virtual environment of the paddocks an unrealistic goal. We have created testing environments that best simulate the challenges the robot will encounter in real life, although the world doesn't necessarily look exactly like the paddock.

VII. Hardware and Software Test and Quality

Ensuring the quality of the hardware and software for this project, as well as being able to test both, encompasses many topics. To accomplish this, the team mainly focused on the hardware compatibility, computational power of components and resource management, using a simulation environment, creating a software quality plan, ensuring code quality practices, going through static and dynamic program analysis, and using the agile development process.

1. Hardware Compatibility

Due to the fact that this project involves a physically implemented robot, hardware compatibility is a significant concern. This project involves multiple sensors and computing hardware, such as:

- 3D LiDAR: Used for mapping and obstacle detection in three dimensions.
- Nvidia Orin Model: This acts as the central computing system, handling the processing of data from the various sensors.
- RTK GPS: Provides precise location information, useful for accurate navigation in outdoor environments.
- Camera: Used to identify donkey manure

It is imperative that these components are all compatible with each other. They need the ability to "talk" to each other efficiently. This involves making sure the hardware:

- Fits together physically: The devices like cameras and LiDAR must be mounted properly and work within the design of the robot to not block the sensors or interfere with each other.
- Handles environmental challenges: The robot must be designed for outdoor use, and the challenges that come with that, by ensuring that the sensors are waterproof and durable. Harsh environments can damage sensitive electronics if they're not protected.
- Communicates properly: Data from these sensors must be sent to the Nvidia Orin for processing. The sensors use different data formats and communication protocols, so the team must ensure compatibility. This often involves working with drivers and middleware (like ROS).
- Consumes power efficiently: Each hardware component should be designed to use energy efficiently, ensuring that the battery lasts as long as possible without compromising performance or functionality.

2. Computational Power and Resource Management

The robot must handle large amounts of data due to the point cloud of the 3D LiDAR and the video feed from the cameras. The Nvidia Orin is a powerful processor designed for such high-demand tasks, but even with this power, resource management is important.

- **LiDAR Data Processing**: The 3D LiDAR sensor produces point clouds, which are dense sets of data points representing the precise geometry of objects and surfaces in 3D space. Real-time processing of these point clouds involves intensive computations, including filtering, segmentation, and potential object recognition. It is essential to ensure that the Nvidia Orin's processing capabilities can manage these computational loads efficiently, without causing performance bottlenecks or delays.

- **Balancing Workloads**: In addition to LiDAR, the robot needs to process camera data for manure detection and GPS data for navigation. Ensuring that the Orin can handle this workload is a key consideration for the robot's overall performance. This is achieved by managing resources efficiently through optimized algorithms or by using hardware accelerators, such as GPUs, within the Nvidia Orin to speed up tasks.

3. Simulation Environment: ROS and Gazebo

Before deploying the robot in the real world, the team is using ROS and Gazebo for simulation. This allows the team to test how the robot behaves under different conditions without risking damage to the hardware.

- **ROS:** This is a framework for writing complex robot software. It helps integrate sensors and provides tools for communication between different components. For example, the data from LiDAR, manure detection camera, and GPS can be passed between software nodes in ROS for processing, control, and decision-making.
- **Gazebo**: This is a 3D simulation tool that allows the team to simulate how the robot operates in different environments. For this project, darker areas, ramps, and manure spots are example testing scenarios. Gazebo can simulate these real-world challenges, so the team can see how well the robot's sensors and algorithms perform in various lighting and terrain conditions before testing in real life.

4. Software Quality Plan

The software quality plan ensures that the robot's software is reliable, efficient, and meets its intended purpose. Some key practices to ensure this are:

- **Unit Testing**: This involves testing individual components of the software in isolation. For example, tests were written for the computer vision model, navigation, and LIDAR data separately from the Gazebo world to ensure the code works as expected.
- **Integration Testing**: Since the robot has many interconnected systems (sensors, GPS, cameras), integration testing ensures that all components work together as a whole. For example, making sure the manure detection camera's data can be processed alongside the LiDAR data for obstacle detection.
- **User Acceptance Testing (UAT)**: Once the robot is fully functional, testing it with actual users or in real-world conditions validates whether it meets the needs of its intended operators (e.g., farmers). This helps ensure that the robot is intuitive, effective, and performs as expected in the field.

5. Code Quality Practices

Code quality is important in a project that is handled by multiple people at different periods of time. Some good code writing practices are:

- **Code Reviews**: Code reviews were conducted to ensure that the written code is readable, understandable to other group members, and runs smoothly. Additionally, since this project is going to be handed off to the capstone team at the end of the semester, it is necessary to ensure that documentation done for the project is well written and understandable to outside sources.
- **Refactoring**: Refactoring of code was also conducted to ensure that the team produces clean code, and that the code is easy to build onto for future use.

6. Agile Development Process

The project adopts Agile methodologies to ensure flexibility, rapid iteration, and continuous enhancement throughout the development lifecycle of the paddock mucking robot. Here's how Agile methodologies are integrated into the development process:

1. **Sprint-Based Development**
   - **Short Development Cycles (Sprints)**: Work is organized into two-week sprints, allowing proper focus to specific features or components within manageable time frames.
   - **Clear Objectives**: Each sprint begins with a planning session where tasks are prioritized based on project needs and advisor/client feedback.
   - **Regular Reviews**: At the end of each sprint, review meetings are conducted to assess progress and gather feedback for future iterations.

2. **Continuous Testing and Feedback**
   - **Frequent Testing**: Throughout each sprint, both software and hardware components are continuously tested. Utilizing Gazebo for simulation allows proper identification and rectification of issues early in a controlled environment.
   - **Iterative Refinement**: Feedback from testing phases is promptly integrated into subsequent sprints, ensuring that each iteration brings the team closer to an optimal solution.
   - **Client Input**: Regular demonstrations and updates keep the client informed and involved, providing valuable insights that shape the project's direction.

VIII. Project Ethical Considerations

In the development of a donkey mucking robot, ethical considerations are paramount to ensure that the team's engineering practices not only achieve technical excellence but also align with professional standards and societal values. The complexity of building such a robot from scratch involves numerous decisions that can have significant impacts on safety, privacy, and stakeholder trust. Adhering to the ethical principles outlined by the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) is essential for guiding the team's actions and responsibilities as engineers.

Institute of Electrical and Electronic Engineers (IEEE) Ethical Considerations:

CLIENT AND EMPLOYER (2.01, 2.05):

- An important code of ethics from Client and Employer section 2.01 ensures a software engineer provides service in their areas of competence, being honest and forthright about any limitations of their experience and education. It is especially important for the team to keep in mind that its members are all undergraduate students and by no means experts in their project. The team needs to be transparent about this and strive to provide professional-level work while learning as much as necessary to satisfy their client.
- Another code of ethics from this section is 2.05 which covers privacy. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law. This is a concern for the project as all members have signed an Intellectual Property Agreement. The team needs to respect the confidentiality and privacy that the client requires.

Association for Computing Machinery (ACM) Ethical Considerations:

Avoid Harm (1.2):

- One important ethical consideration is staying compliant with ACM guideline 1.2. Harm is defined as negative consequences, which encompasses physical injuries and property damage. It is engineers' responsibility to minimize any harm done, including unintentionally. Since the robot traverses the paddock, it has the possibility of running into people, dogs, or donkeys and causing personal injuries to them. Additionally, the robot could end up running into different obstacles scattered throughout the paddock, which could cause property damage. It is the team's job to ensure that the robot does not run into any living or nonliving obstacles in the paddock.

Design and implement systems that are robustly and usably secure (2.9):

- Another vital ethical consideration for the successful implementation of the robot is ensuring compliance with the ACM guideline above. Since the robot processes sensitive data like LiDAR, camera, and GPS information, it is crucial to implement security measures to protect against breaches or misuse. Doing this prevents unauthorizes access, safeguard privacy, and ensures that the robot operates securely and reliably in its environment, which aligns with the ethical standards and helps build trust in its performance.

Most Likely to be Violated:

While it is essential for engineers to strive to adhere to all ethical principles, certain principles may take precedence over others depending on the context, and some may be more susceptible to compromise in specific situations. In this project, each team member is navigating the numerous steps and extensive work involved in building a full-fledged donkey mucking robot. Additionally, the limited timeframe allocated for this development necessitates making significant compromises, which sometimes involves prioritizing working software over comprehensive documentation practices. Due to these circumstances, certain ACM/IEEE ethics principles are more likely to be challenged or inadvertently violated.

Ensure proper and achievable goals and objectives for any project on which they work or propose (3.02)

- During development, this principle can be compromised through scope creep. Scope creep occurs when the project's objectives gradually expand beyond the original plan without corresponding adjustments to time, resources, or budget. For instance, initially focusing on basic manure collection may lead to the addition of advanced features like enhanced obstacle detection or increased autonomy as the project progresses. This unplanned expansion can make the project goals unrealistic and unattainable within the given constraints. As a result, the team may struggle to meet deadlines, resources may become overstretched, and the overall quality of the robot could suffer due to rushed or incomplete implementations.

Ensure adequate documentation, including significant problems discovered and solutions adopted, for any project on which they work (3.11)

- Building upon the concern about scope creep violating the previous principle, the above principle may also be compromised. When a team is intensely focused on meeting expanded objectives and tight deadlines, especially in a complex project like developing a donkey mucking robot from scratch, documentation tasks may be deprioritized or neglected. This oversight can lead to incomplete or insufficient records of design decisions, challenges encountered, and solutions implemented.

The team is effectively communicating with the design team, advisor, and client to reduce the impact violating these principles may have.

Michael Davis Tests:

Michael Davis is a philosopher who specializes in professional ethics. Davis's approach to ethical decision making is to first state the problem you are attempting to solve, then check the facts of the problem to ensure that all parts of the problem are considered. Next, identify the stakeholders involved in the problem, those stakeholders' interests, and their responsibilities. Then, develop alternative solutions to the problem and their potential consequences. Finally, evaluate the problem according to his seven tests. In following this process, the reversibility test and publicity test were identified as tests that could possibly fail.

Reversibility Test: Would this choice still look good if I traded places? (i.e., if I were one of those adversely affected by it?)

- While this robot has the potential to increase efficiency and reduce costs for the client, it could take jobs away from the personnel currently doing those tasks. With that in mind, this robot would not be a good option for increasing job count in the local economy.

Publicity Test: How would this choice look on the front page of a newspaper?

- Similar to the reversibility test, while developing a robot to decrease costs and increase efficiency looks impressive to the client, there is still the drawback of potential job loss. If this robot caused people to lose their jobs, it could generate a negative public reaction to it.

IX. Project Completion Status

The team has made considerable progress since the work of the summer team; however, there is still work to be done. Currently, the project has a strong foundation for future teams to build upon. The next steps involve handling LiDAR and GPS data and integrating all systems to complete a cohesive virtual robot environment.

Virtual Testing World:

Under current development status a comprehensive virtual environment has been created in Gazebo Fortress, providing a range of realistic testing conditions. These include a variety of obstacles and features like bumps, turns, fences, hills, ramps, humans, trees, bushes, toys, and stables. These obstacles are present in the testing worlds showcased in Figure 3 and Figure 4 below. Additionally, the environment allows for other environmental factors, such as wind or lighting conditions, to be tested. This virtual world runs on Ubuntu 22.04 with ROS2 Humble Hawksbill, enabling a flexible and stable platform for experimentation and future use.

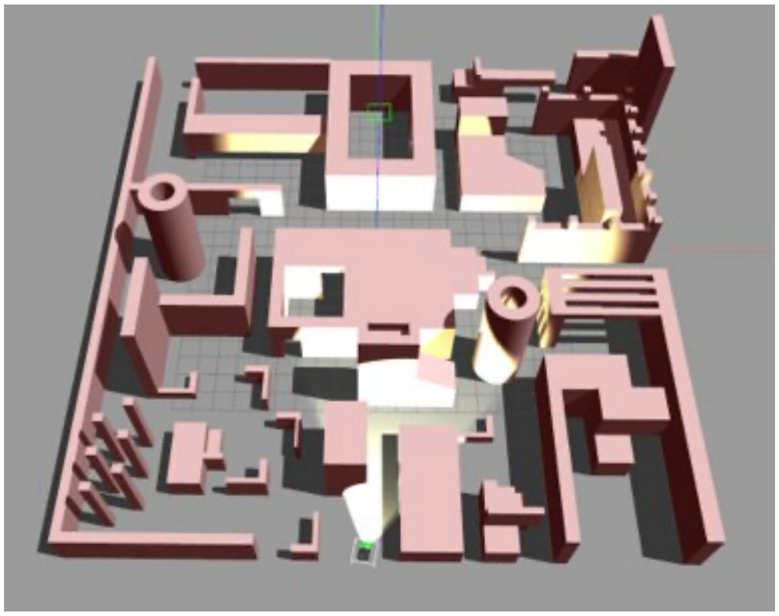


Figure 3: Robot Gym Testing World in Gazebo

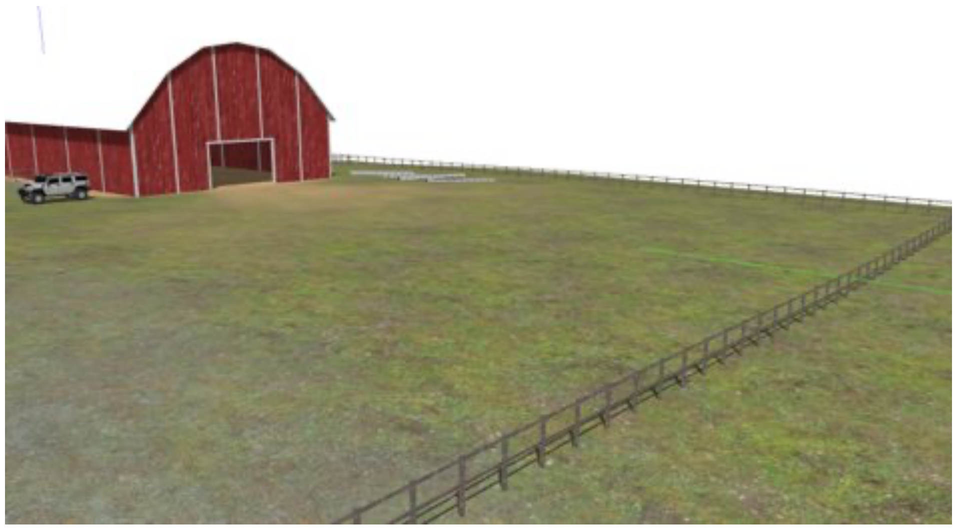


Figure 4: Donkey Paddock Testing World in Gazebo

Navigation:

The navigation component of the project has been created using a 2D lidar-based simulation through the Nav2 framework. This allows for the testing of foundational pathfinding and obstacle detection capabilities within the virtual world. Nav2 creates a 2D cost map of the 3D world, this cost map prevents the robot from entering areas with obstacles or makes it more costly for the robot to navigate close to obstacles, which can be seen in Figure 5. Nav2 navigates the robot to a given goal pose, we have created a ROS node to control Nav2, as seen in Figure 6. This is to make sure we can integrate our computer vision model and other sensor input into the pathfinding algorithm. In the future, transitioning this to a 3D LiDAR would improve the robot's spatial awareness and allow it to interact with its environment with a higher degree of depth and accuracy. This upgrade is critical for accurately simulating real-world conditions where 3D perception is imperative for correct navigation.

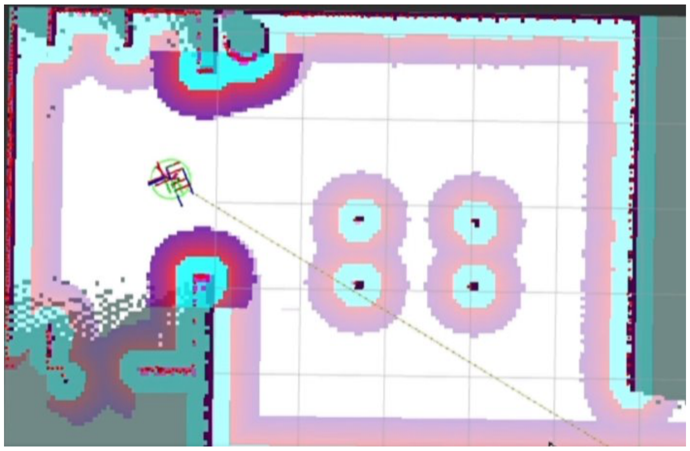


Figure 5: Navigation in RViz2

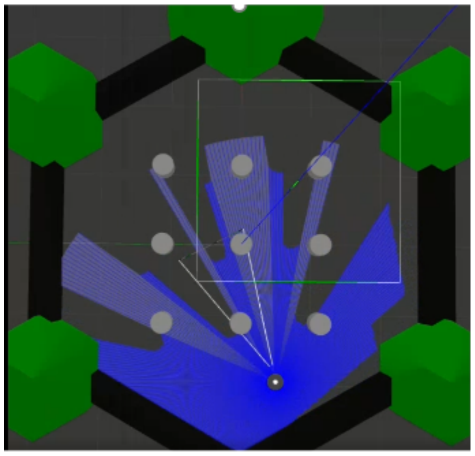


Figure 6: 2D LiDAR in Gazebo

Manure Detection Model:

Building upon the summer team's work, a significant enhancement has been made in object detection, where the manure detection model can now run offline. Roboflow presented challenges because it required an internet connection to function and imposed a mandatory monthly subscription fee. Due to this, the team was forced to build its own model which no longer relies on Roboflow. This model is designed to operate entirely offline and thus has overcome this key limitation. By refining the model to work without an internet connection the team has created a robust, self-sufficient detection system that can achieve more reliable performance. This offline model marks a key advancement, achieving a mean Average Precision (mAP) score of 70%, a substantial improvement over the summer team's model, which had a mAP score of about 50%. While the summer team prioritized achieving high confidence scores (above 80%) for individual detections, their model's overall accuracy across multiple images was around 61%. This is documented in their Roboflow project which can be found here. The new model not only surpasses these metrics but also ensures more reliable performance without dependency on external platforms, representing a major milestone in our implementation. The visualization of how the new model works can be seen in Figures 7 and 8.

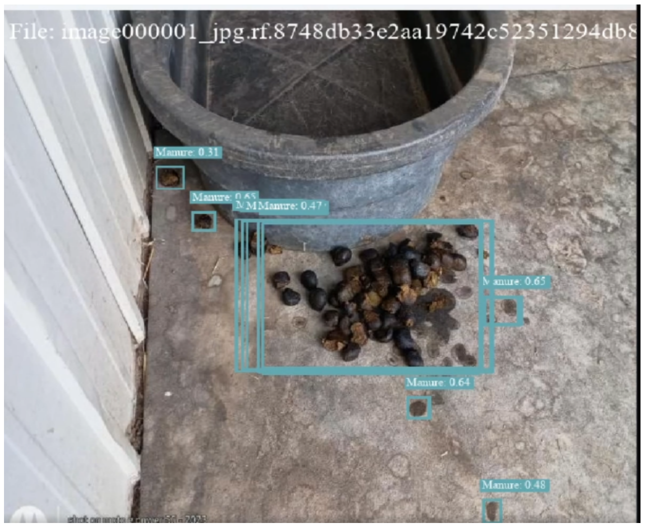


Figure 7: Visualization of Manure Detection

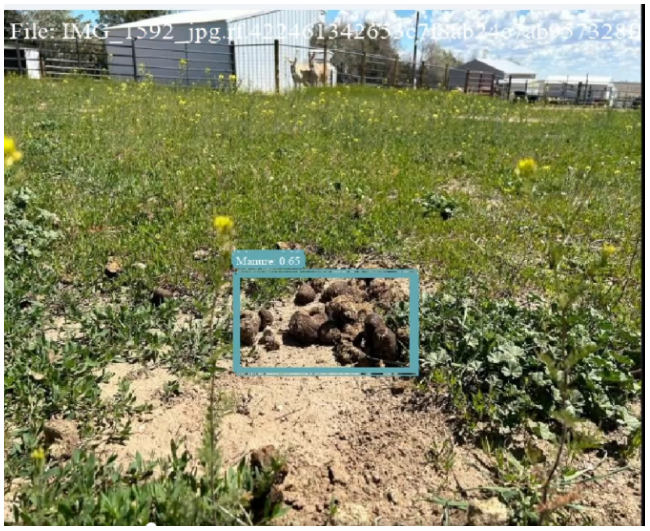


Figure 8: Visualization of Manure Detection

Robot Model Simulation:

A digital replica of the physical robot has been created which matches the Capstone team's current real-world specifications, as seen in Figure 9, and design of the robot. The digital replica, seen in Figure 10, features a rectangular body and four wheels, with a shovel arm attached to the head. Sitting on top of the body are the various sensors, camera, LiDAR, and a GPS antenna. The virtual replica may look primitive, but the sensors are intended to work virtually.

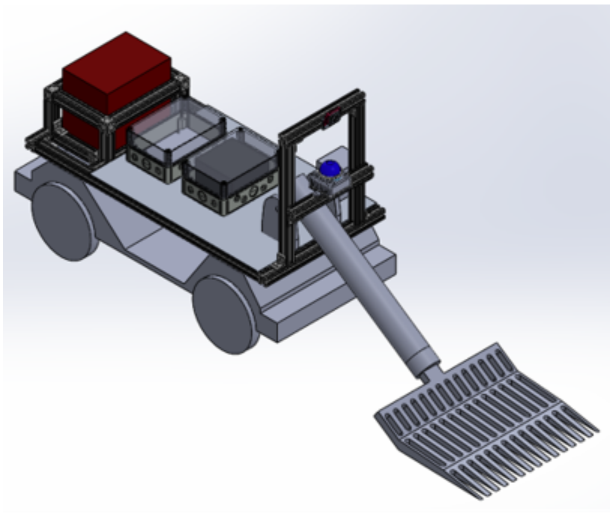


Figure 9: CAD Design of Robot by Sean McNair

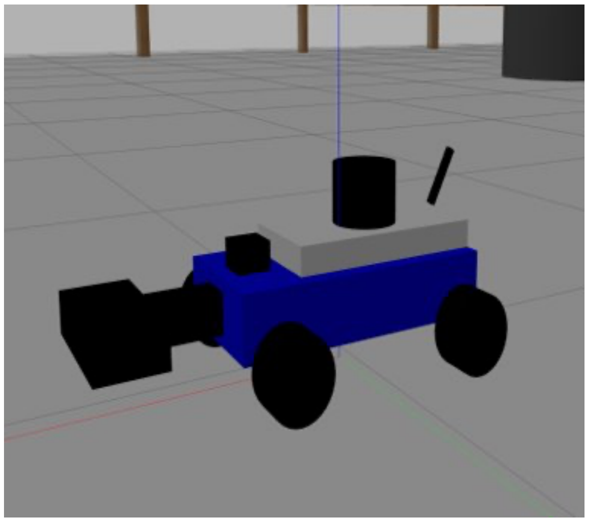


Figure 10: Gazebo Robot Model

System Integration:

This is where most of the future work is focused as the team created multiple pieces working independently. The focus for this issue is about specifically integrating all individual components—virtual environment, navigation system, detection model, and hardware—into a cohesive, fully operational system. The process of integration is currently where the project sits, and future teams will face this challenge. This culminates each aspect into a comprehensive robotic environment that enables autonomous navigation, object detection, and environmental interaction.

X. Future Work

The next challenge of this project is moving from simulation to the real world. Right now, the robot works in a virtual environment, but to use it in real life, many improvements need to be made, and all edge cases need to be considered. It is also expected that the robot is a scalable product that could be used on multiple farms with easy setup and durable hardware to minimize maintenance, so optimization is necessary.

Hardware Development and Integration:

Currently the robot is running in simulation as mentioned above, in the future the robot needs to be brought outside of simulation and into the real world. This means integrating the software with real world hardware – the microcontroller, sensors, and motors. These hardware components need to be thoroughly tested to ensure they perform adequately in real life scenarios. The sensor side of hardware has already been chosen, but processing of sensor data from the physical hardware and testing is still needed and necessary to ensure reliable detection, navigation, and collection of manure in real-world conditions.

Improved Manure Detection:

Accurately detecting manure is essential to this project, therefore the model can always be improved on. A model has been trained with an accuracy of 70%, which is adequate for now and great for testing purposes, but in the future, this accuracy should be improved and brought to roughly 95%. The good news is that the team created a dataset of over 4,000 images for this new model located here. With that amount of label data at hand, this should be low hanging fruit for a future team to improve on. The whole robot navigation revolves around correctly identifying manure, so it is necessary that this part works accurately.

Handling Edge Cases:

There are certain edge cases that are extremely difficult to create in a simulation environment. These include things like uneven terrain, dust, climate conditions like mud, snow, and extreme temperatures. Currently, there is no way of knowing how the robot will react to these edge cases. In the future, testing these edge cases is crucial for the robot's performance outside of an ideal environment. If any of these cases proves to be detrimental to the robot's performance, sensors could be added to detect bad terrain or weather, and it could have an algorithm to solve the issue or ask humans for assistance.

Scalability

There is also a hope to expand this project into a commercial product. For this to occur, the robot needs to be tested extremely well to have the confidence to deploy it in multiple farms with paying customers. Additionally, there needs to be a straightforward way to give the robot the bounds of a paddock. Right now, they are set manually in the code, but in the future, an app where you could draw the bounds would be useful. This way operating and deploying a robot in a new farm would not require anyone to touch the code. Lastly, the robot needs to be made very robust to withstand the wear and tear of time. It would not be ideal to have to frequently fix and maintain robots deployed all around the country.

XI. Lessons Learned

With this project there were many lessons learned by the team. From hardware component analysis to robotics software, there was a necessity to justify the hardware components of the robot, and an extensive list of robotics software used in development.

Hardware:

One of the requirements the team was tasked with was identifying the hardware components needed on the software side to operate the robot. Throughout the process of researching and identifying possible hardware components, the team learned which specifications to look for, such as the CPU and GPU (clock rate) for the microprocessor and the field of view (FOV) and frames per second (fps) for the camera. Although the team initially encountered budget constraints, they were able to compile several different options consisting of various hardware combinations and decided on the most suitable one based on budget limitations, complexity of software implementation, and compatibility of the hardware components.

Roboflow:

While Roboflow is a great platform for annotating images and running computer vision models with internet access, the team discovered that its offline option would not be suitable for running the computer vision model on the robot. Consequently, the team learned how to train their own YOLOv5 model using Ultralytics' YOLO documentation. By utilizing the trained model generated from that process, the team was able to implement the computer vision model on the robot without relying on the internet.

Robot Operating System 2 (ROS2):

ROS2 is an excellent choice for creating the code for this project since it allows for a cohesive framework to work in, utilizing topics, services and nodes, and integration with gazebo for virtual testing. However, it can be hard to use and learn. Since ROS2 works best with a Linux operating system, the team members installed a virtual machine or used a Linux based operating system for developing, which proved to have its own challenges. The ROS2 documentation and tutorials proved to be useful in learning how ROS2 works and in setting up needed dependencies to use certain included packages, such as gazebo.

Gazebo:

Gazebo was very useful in modeling the robot in a virtual environment that simulated a paddock. The ROS2 tutorial for Gazebo and the information available online from experienced developers has been useful in creating the virtual environment and the robot and obstacles inside of the paddock. Using Gazebo allowed the team to test how the robot would work without having the physical robot.

XII. Acknowledgments

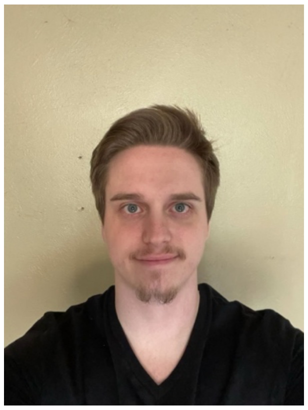
We would like to thank and acknowledge the work and support of our client, Longhopes Donkey Shelter, for providing this project and their guidance. Kathy Dean, the shelter's president and founder, has been particularly involved and helpful in the progression of this project this semester. We would also like to thank the staff at Longhopes for supplying us with photos for our computer vision model and guidance around the shelter.

Next, we want to thank our advisor, Donna Bodeau, for continuous support and encouragement throughout this project. We also acknowledge the work of all the advisors, especially Kathleen Kelly for great coordination of this course.

As we are working on this project with a senior design team, we would like to recognize their work and collaboration throughout this semester and project. Also, the summer team provided us with valuable information and great directions to start with.

Lastly, we would like to acknowledge Kaveh Fathian and the ARIA Robotics lab for their great technical support and help in the early stages of the project when everything was very new to us. We are very thankful for the direction and expertise that he gave us.

XIII. Team Profile



Andrew Bernklau

Undergraduate Computer Science: General

Born in Lakewood, Colorado

Favorite Longhopes Donkey: Milo

Rhys Fernando

Undergraduate Computer Science: Robotics and Intelligence Systems

Born in Parker, Colorado

Favorite Longhopes Donkey: David

Julia Luoto

Undergraduate Computer Science: Robotics and intelligence systems

Born in Tampere, Finland

Favorite Longhopes donkey: Otis

Lindsey Shanahan

Undergraduate Computer Science: General

Born in Wheaton, Illinois

Favorite Longhopes donkey: Luna

References

Appendix A – Key Terms

This appendix provides a concise glossary of key terms and concepts used throughout the project documentation. These terms are integral to understanding the technical and functional aspects of the autonomous paddock mucking robot, including its navigation, obstacle avoidance, and environmental mapping systems.

Term	Definition
LiDAR	*A technology employed by the robot for mapping and navigating its surroundings through light detection and ranging.*
Pathfinding Algorithm	*Refers to the logic and methods used by the robot to determine the most efficient and obstacle-free routes within the paddock.*
Paddock Mapping	*The process of identifying and understanding the boundaries and layout of the paddocks for navigation.*
Obstacle Avoidance	*The capability of the robot to dynamically detect and navigate around stationary and moving obstacles.*
2D LiDAR	*2D lidar emits laser pulses in a single horizontal plane to measure distances, creating a flat representation of the surrounding area. It is commonly used for obstacle detection and mapping in two dimensions.*
3D LiDAR	*3D lidar uses multiple laser pulses across different angles to capture a full three-dimensional point cloud, offering a detailed spatial understanding of the environment.*
Machine Learning (ML)	*Machine learning is a subset of artificial intelligence that enables systems to learn from data, identify patterns, and make decisions with minimal human intervention. It involves training algorithms on large datasets to improve their performance over time without being explicitly programmed for specific tasks.*
Computer Vision	*Machine learning for computer vision involves training algorithms to interpret and understand visual data, such as images or videos, by recognizing patterns,*

	objects, and features. This approach enables computers to automate tasks like image classification, object detection, and image segmentation with increasing accuracy.
Gazebo	*Gazebo is a software program that allows one to simulate a robot in a virtual environment. In ROS we can interact with gazebo as if we were interacting with a real-life robot.*
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.*
Mean Average Precision (mAP)	*The mAP metric for the computer vision model is primarily based on the model's precision and recall metrics. This looks at where the computer vision model identified donkey manure in the validation images, then compares that identification to where manure was determined to be from the original annotated image. The overlap between the bounding boxes drawn from both images is what produces this metric, and the metric is the closest to resembling an accuracy metric.*