COLORADOSCHOOLOFMINES
EARTH • ENERGY • ENVIRONMENT

CSCI 370 Final Report

The Dungineers

Maggie Acheson
Ryan Manley
Grace Van Gorder
Luke Vo

Revised June 9th , 2025

CSCI 370 Summer 2025

Professor Donna Bodeau

Table 1: Revision history

Revision	Date	Comments
New	5/12/2025	Document Creation and Initial Formatting / Setup
Revision 2	5/14/2025	Introduction, Requirements, Risks, DoD, Team Profile *(sections I – V, XIII)*
Revision 3	5/24/2025	System Architecture *(section VI)*
Revision 4	6/1/20205	Testing and Quality, Ethical Considerations *(sections VII, VIII)*
Revision 5	6/8/2025	Project Completion Status, Future Work, Lessons Learned (sections IX – XI)
Revision 6	6/9/2025	Acknowledgements, References, Key Terms *(sections XII, XIII and Appendix A)*
Revision 7	6/10/2025	Peer Review
Revision 8	6/11/2025	Finalized Paper

Table of Contents

I. Introduction

Our product is a prototype of an **Autonomous Paddock Mucking Robot**, which is being developed for **Longhopes Donkey Shelter**. Three previous teams have contributed to this project; one for vision; one for pathfinding; and one for hardware. Our team is responsible for implementing the previous projects' software and integrating it into the hardware for the prototype given to bring it closer to a functional product. The client, **Longhopes Donkey Shelter**, aims to rescue donkeys and rehabilitate them until suitable homes are found, and this robot would reduce their operational costs, allowing them to function more effectively as a non-profit.

Figure 1: Team at Longhopes Donkey Shelter

Two previous CSCI 370 projects and a previous Capstone Design project each tackled various tasks. The two previous software projects worked on a pathfinding algorithm for the prototype and a machine learning algorithm trained on identifying manure.

The current hardware is a Capstone Project. It has the ability to drive around the paddocks and uses a rake on an arm, driven by a CAM, in order to pick up the manure. A LiDAR sensor is included for spatial awareness to avoid obstacles, a GPS sensor is included to define paddock boundaries, while a camera is used for the detection of manure.

Longhopes Donkey Shelter initially sought to reduce the manual labor involved with cleaning their paddocks, but the long-term goal shifted to the prospect of eventually commercializing the paddock cleaner as a product to fund the shelter's operations. This means we must consider the possibility of future clients for this product as well. Additionally, the animals that will be near and interacting with this machine should be considered primary stakeholders, as they will spend the most time with it.

We may have future groups coming in after us to inherit the project, so we should build our software with maintainability in mind, allowing future teams to adjust and adapt it for future prototypes as needed.

II. Functional Requirements

We were given lots of previous materials for our final product. As a result, we need to fulfill the previous functional requirements.

1. **Ease of Use:** The robot interface must be simple enough for a non-technologically inclined individual to use.
2. **Navigation:** It must be able to navigate the paddock autonomously.

3. **Obstacle Avoidance:** It must have a method to avoid collisions with both stationary obstacles (such as toys and trees) and non-stationary obstacles (volunteers and donkeys).
4. **Paddock Profiling:** The unit must be able to save unique paddock boundaries and identify which paddock it is in based on GPS location.
5. **Manure Detection:** The unit must be able to identify manure.
6. **Integration:** We need to test the previous software on the physical model and implement it successfully.
7. **Effectiveness:** The robot must be able to traverse a sufficient 95% of the paddock and pick up 80% of the manure therein.

III. Non-Functional Requirements

The funds spent by the capstone team were $6583.41 with the original project budget being $5000, we worked to keep any additional costs low.

1. **Hardware Upgrades:** Dustproofing, waterproofing to IP-67, and other environmental protection requirements:
   a. Waterproofing through using cable bladders
   b. Covering all non-waterproof or fragile components
2. **Documentation:** Code must be easily readable and maintainable by future teams:
   a. Comments
   b. Documentation on various languages and frameworks used
   c. Well-organized code
3. **Vision Node Tolerances:** Coded tolerances for the robot:
   a. Tolerances for identifying obstacles
   b. Tolerance for what is identified as manure
   c. Tolerance for components of the robot that are in the field of vision
4. **Location Constraints:** No reliance on Wi-Fi or internet connectivity or service connectivity
5. **Compatibility:** Software and hardware should be compatible with each other

IV. Risks

Key risks for the project include:

1. Safety when operating near animals.
2. Software and hardware operate harmoniously with minimal human intervention.
3. Virtual environments accurately reflect the real world.
4. Non-technologically inclined individuals must be able to operate the machine.
5. We inherited this project from previous individuals which means we carry the flaws of their implementations and had to fix them as much as we could during our project.
6. This project already has required significant funding and time investment from previous teams, so there is a larger punishment for failing.

V. Definition of Done

The project will be considered complete after this five-week session if the prototype can be placed in a paddock, started, and autonomously collect manure with minimal human intervention. To do this, the pathfinding, vision, GPS, and controls pipelines must all be functional and communicating.

Tests we want the client to run include:

1. The robot successfully cleans a paddock without manual intervention.
2. The robot demonstrates the ability to avoid basic obstacles and the donkeys.
3. Clients are able to operate the robot.

VI. System Architecture

The system architecture for the prototype robot is composed of two primary controllers: an Arduino and an NVIDIA Jetson Orin (the Orin). Arduino is responsible for hardware control, sending PWM signals in response to I2C movement commands. The NVIDIA Jetson Orin runs ROS2 and handles high-level decision making and planning; it implements pathfinding, vision, and obstacle detection/avoidance algorithms.

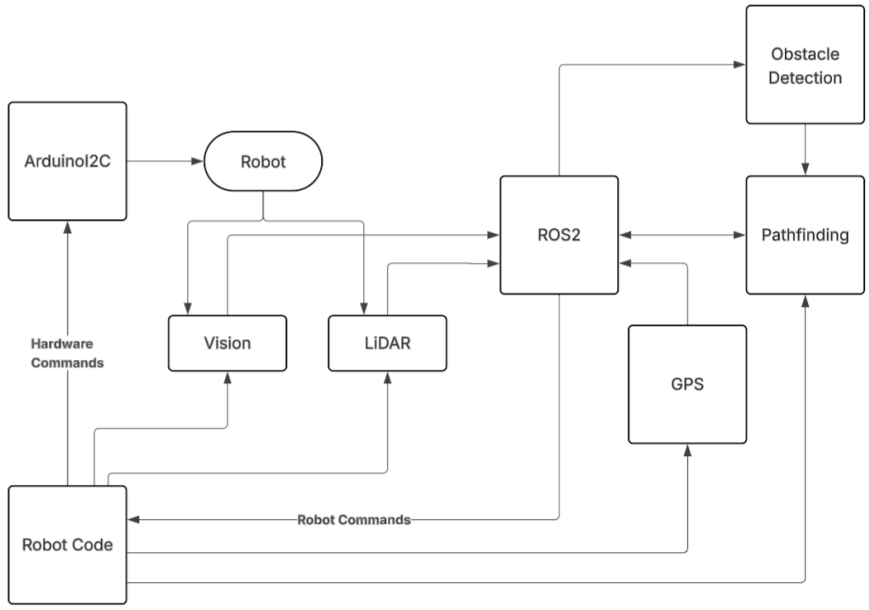


Figure 1: Robot system architecture diagram

Software Framework

As demonstrated by the chart above, the project utilizes ROS2 (Robot Operating System 2) to facilitate communication between software modules, sensors, and hardware. ROS2 is an open-source framework designed for robotics, which allows the creation of packages, nodes, and topics to connect each module of software together. ROS2 has a large community and a lot of resources for learning and building out packages.

The robot makes use of three sensors (GPS, camera, and LiDAR) to implement three main algorithms (pathfinding, vision, obstacle avoidance) and these are all integrated inside of the ROS environment.

Arduino and I2C

Arduino is an open-source electronics platform that aims to integrate hardware and software. It is widely used for building digital devices and embedded systems. The interface between the Arduino and the NVIDIA Jetson Orin uses the I2C protocol, which is often used within Arduino projects to enable communication between the microcontroller and other digital devices.

Movement and Manipulation

Arm: The purpose of this mechanism is to lower and raise a rake which is attached on one end of the arm. It is operated by a CAM to prevent back drive of the motor.

Wrist: This is a motor within the arm which rotates the rake 180 degrees. The function of this motor is to dump any manure that may currently be held within the rake.

Drive Train: This encompasses the tires, steering, suspension, and motors. This allows the robot to move forward, backwards, left, and right, giving it the ability to maneuver around the paddock.

Hardware Implementation: All hardware components are assembled onto a metal chassis that is built into an RC Car Base. The wheels of the RC car were previously upgraded to larger wheels to improve traction in muddy environments. All of the manipulators are controlled by Arduino and limit switches connected to the Orin set the movement boundaries for each axis.

Software Updates

Our challenge is to integrate several components of code into one usable program. We have decided to perform this integration using an Object-Oriented Programming method. This divides the code into individual hardware and software components, combining them in one central class. Each major subsystem (GPS, Vision, etc.) is encapsulated into a separate class or node for us to work in. We will then implement a control class to manage behavior later.

Pathfinding

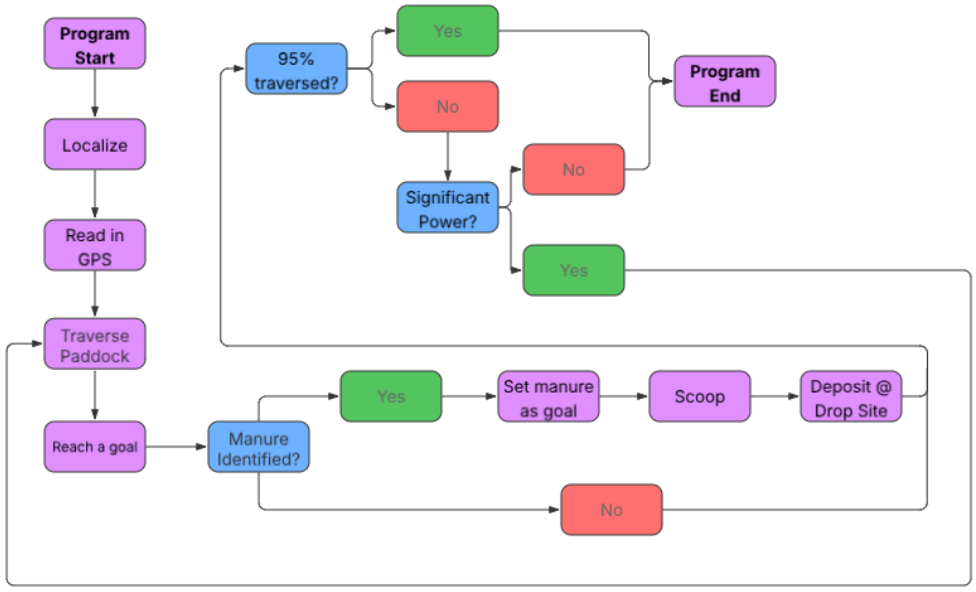


Figure 3: Pathfinding system architecture

Our robot has to find a path through all the obstacles that may be in the paddock. This has to be dynamic to some degree since the environment will likely change in real time. Navigating mobile objects like donkeys inside of the paddock would require the use of a real-time obstacle sensor like LiDAR. Our program, visualized above, defines pathfinding behavior to determine the optimal path to explore 95% of the paddock. These paths will be determined by various factors such as the boundaries and stationary obstacles defined by the GPS, manure detected by the camera, and dynamic obstacles such as staff or animals.

Manure Detection

Our robot must be able to identify manure in various conditions and locales. We used a machine learning trained computer vision model to detect manure using the onboard camera. The previous software team developed an offline model that has stubs for being incorporated into the ROS environment. As of now, the previous code base was only available when feeding in pictures since the camera isn't available as of now. This previous model was trained on YOLOv5 from a Roboflow library of a previously obtained dataset.

Since our robot would need to take in some sort of camera feed, we decided to implement live camera feed processing in order to identify the manure in real time. Figure 4 shows our current approach to how the model should be applied.

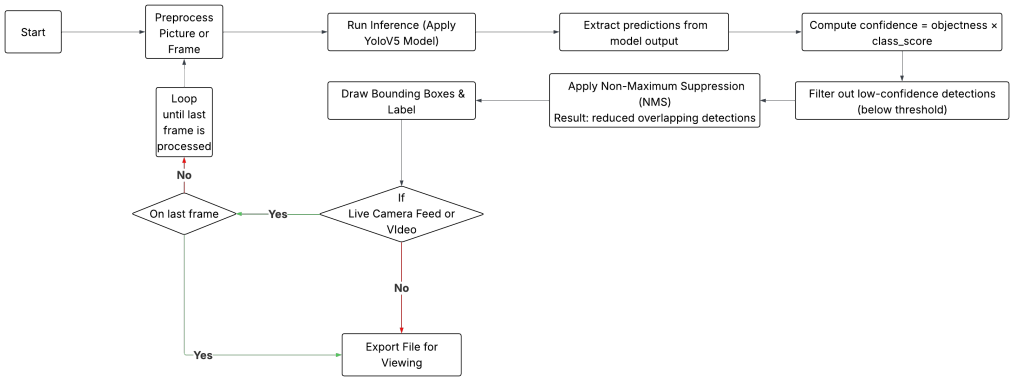


Figure 4: Manure Detection Flowchart

NMS is important for making sure only one pile of manure is detected at a time since without it, multiple bounding boxes could be applied to the same piece of manure. This means our mAP (mean Average Precision) score should be lower since that metric penalizes multiple boxes on the same object. Since we only have one box per object detected now, it should prevent the same pile of manure being detected multiple times in the same frame as well. The downside of this is computational cost, though.

Moving forward, we decided to implement video processing by applying the model per frame to first get a gauge for how live vision would operate. Along with this, we planned to record some videos to test our model as well.

GPS Software & Hardware

The portion of the code that our team is tasked to develop on our own is the GPS software. To define boundaries for our robot's movement, it has been determined GPS is necessary due to the various types of boundaries that a paddock can have. This ranges from things like chicken wire (which LiDAR has complications with) to normal types of wooden fencing. GPS also allows us to define various obstacles as well and associate them with the paddock.

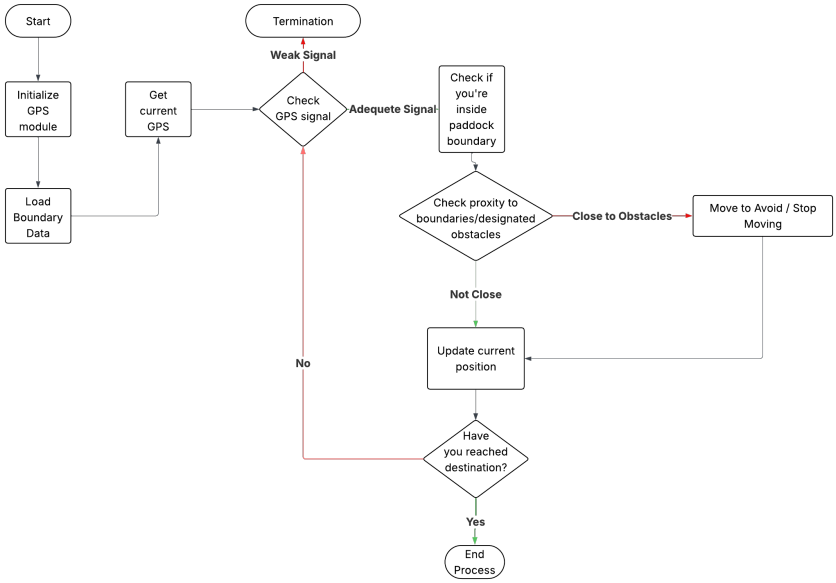


Figure 5: GPS Flowchart

This framework, shown in Figure 5, was developed in **ROS** and will later be incorporated into the previous software team's pathfinding algorithm.

Design Challenges:

Since our project inherits key components from previous projects, lots of the main design obstacles we encountered were centered around adapting to the software work of previous teams. Along with this, we also had to adapt aspects of the physical prototype to be more in line with our functional and nonfunctional requirements.

1. **Robot Architecture**: Our physical prototype came to us with some minor issues that weren't finished and since the previous hardware team didn't have software experience, we had to develop some software to make the prototype's capabilities closer to a final product.
    1.1. Camera is not connected: The camera module attached to the arm tower uses a 15 to 22pin FPC cable, but the cable that came with the camera is not long enough to reach the Orin. We needed to find a solution for the correct length of cable to reach the NVIDIA Jetson Orin.
    1.2. Camera was never integrated in ROS environment
        1.2.1. We need to apply the visual pipeline for this since there isn't node integration yet.
    1.3. Electrical wiring was poorly labelled
    1.4. Electrical was built in a way that the wires easily detached during operation of the robot
    1.5. With the current suspension, the frame collides with the back tires
2. **Previous Software**: Some aspects of the previous software made it hard for our team to inherit, and the unfamiliar code base of previous groups exasperated this to some degree.
    2.1. ROS environment:
        2.1.1. Complicated to access
        2.1.2. Not calibrated to our real-world environment, only the simulated one
    2.2. Gazebo environment is difficult to access

2.3. There are multiple programming languages to integrate

3. **GPS Software:** We had to develop GPS software on our own. This GPS software is required to successfully implement pathfinding since the GPS coordinates will be needed for the boundaries of the paddocks.
    3.1. Integrated from ROS making it difficult to pull information from
    3.2. Developing a new environment and starting over
        or
    3.3. Using previously developed environments which were difficult to navigate and had little documentation

VII. Testing and Quality

Our tests are a mix of advancing the previous team's virtual environment testing whilst implementing their virtual tests into realistic testing with the developed prototype. We verified the previous teams' tests while implementing the software and plan to test the hardware on-site.

Off-Site Testing

1. Establish Connections:
    1.1. With GPS base station through U-Center
    1.2. With GPS rover station through U-Center
    1.3. With Orin through ssh
    1.4. With webcam
2. Camera Manure Detection:
    2.1. Allow robot to detect manure
        2.1.1. We can test this by using preexisting manure photos
        2.1.2. We can test this by using false manure
    2.2. Test different lighting and environment conditions for camera
    2.3. Test different methods of visual feeding for the camera
        2.3.1. Test for pictures
        2.3.2. Test for video
        2.3.3. Test for live camera feed
3. Test Shape for Pathfinding:
    3.1. Manually define GPS boundaries
    3.2. Allow robot to create path in virtual environment
    3.3. Place fake manure somewhere along path
    3.4. Allow robot to run path finding and discover objects
    3.5. Document how all components interact

On-Site Testing

1. Empty Paddock (containing manure, but no donkeys):
    1.1. Manually gather GPS boundaries and stationary obstacles
        1.1.1. Define these coordinates in pathfinding algorithm
        1.1.2. Allow robot to create path in virtual environment
        1.1.3. Allow robot to traverse path in virtual environment
    1.2. Manually traverse paddock:
        1.2.1. Test various terrain
        1.2.2. Test various ground conditions
        1.2.3. Test turning radius
        1.2.4. Test path completion capabilities
    1.3. Find parameters for scooping algorithm
2. Full Paddock (containing manure and donkeys):
    2.1. Observe how the donkeys and the robot interact while device is stationary
    2.2. Observe how the donkeys and the robot interact while device is in motion
3. Full Computer Vision Manure Detection Testing:

3.1. Obtaining more pictures and videos for off-site testing
    3.1.1. Record manure in a variety of locations
    3.1.2. Take videos with changing conditions and different locales
3.2. Live testing camera feed on real instances of manure.
    3.2.1. Test in paddock
    3.2.2. Test in barn/indoor areas
3.3. Testing confidence thresholds for computer vision to find the best range for identification

Functional Requirements Testing

Requirement 1: Ease of Use

When testing on-site, we have the benefit of allowing non-technologically minded individuals to test and operate the robot. By allowing these users to operate the robot, we are able to observe how they expect to interact with it, what aspects of operation are difficult, and any issues they encounter. We'll be able to discuss the user experience with them, taking note of anything that was awkward or difficult for them, and plan to address these problems.

Requirement 2: Navigation

A core requirement for our project is that it should be able to navigate a paddock by itself. We have implemented a previous projects pathfinding algorithm, and our prototype has the ability to detect obstacles and pick a path on its own without human input. We plan to simply start the pathfinding algorithm and compare it to how it would behave in our virtual Gazebo environment and oversee the differences.

Requirement 3: Obstacle Avoidance

By testing within our Gazebo environment, we can approximate how the robot will interact when traversing through various obstacles. Transitioning this into onsite testing, since our environment had a larger number of obstacles and positions it had to navigate through, our hope is that when we traverse through the paddock, there will be less issues that arise.

Requirement 4: Obstacle Additions

Our Gazebo environment easily supports the addition of new obstacles, so we have been able to test obstacles that were in the midst of our paddocks and how they would interact together. When storing this with our paddock profiling, we were planning to store the coordinates of the obstacles and the size of them with whichever relevant paddock profile manually needed these obstacles.

Requirement 5: Paddock Profiling

We were simply planning the ability for each mapped paddock to be saved by recording all of the GPS coordinates involved and simply registering that as a paddock. We will test the prototype's ability to cycle between each paddock and see if it is capable of distinguishing the differences in how it should behave in each paddock.

Requirement 6: Manure Detection

The prototype must be able to recognize manure in various settings and using its own peripherals instead of testing it on our virtual environment. Along with this, it should be initiated to scoop up manure whenever it detects manure. To test this feature, we will use false manure in order to trigger the mechanism and see if it actually will be able to scoop up the manure as spotted.

Requirement 7: Integration

As the robot was developed, we continuously tested our code additions on the hardware, ensuring the changes made didn't impact the operation. When an issue arose, it was quickly addressed to ensure bugs and hardware integration

issues weren't buried over time. The hardware was regularly inspected and tested to ensure issues aren't misattributed to software bugs.

Requirement 8: Effectiveness

As brought up by previous teams before us, the robot needs to be effective to a certain degree. This was determined to be a 95% threshold of effectiveness where it should be able to traverse 95% of the paddock and pick up all the manure it observes. This 95% threshold will be determined by if the hardware prototype can clean up 95% of the paddock.

Non-Functional Requirements Testing

Requirement 1 Dustproofing, Waterproofing, and other environmental protection:

To solve this issue, we have revised certain aspects of the prototype hardware. This includes getting a new container and making sure we apply accurate waterproofing techniques to certain parts of the prototype.

Due to the prototype's need to be durable and traverse various different terrains we have revised the wheel design as well. We improved the previous hardware team's wheel design (which previously involved glueing the wheels as a temporary measure to get them attached) by 3D-printing a new design, shown in Figure 6, mounting the wheel to the base instead of gluing the wheel to the axel. The wheels had larger holes for the axles and accepted a larger hex shaft collar compared to the original wheels, so an adapter was created to accept the smaller hex shaft collar and axle with the new, larger wheels.

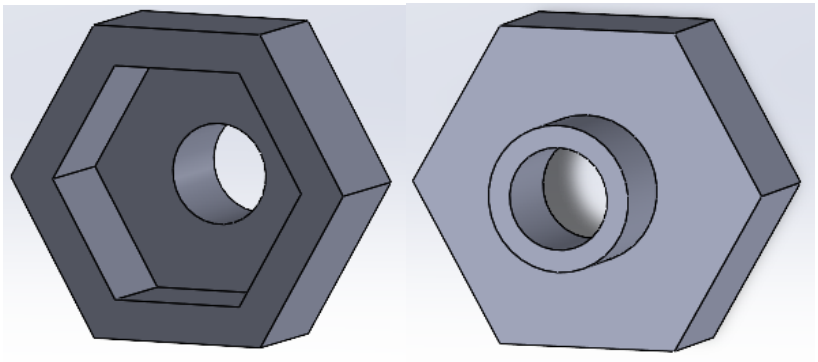


Figure 6: New 3D-Printed Mounting Design

Along with this, we have asked for a new container. The purpose of this container is not only to better waterproof the main electrical components, but to also improve the protection of the components involved.

Requirement 2: Code must be easily readable and maintainable by future teams.

We have developed documentation for all the environments we have worked on and are planning to develop documentation for how ROS will incorporate pathfinding and manure detection as nodes for ROS. This will improve the code base and future teams' ability to interact with the project

Requirement 3: Coded tolerances for the robot

We have tested all the tolerances inside of the Gazebo container and have adjusted them to fit our living environment. We plan to test these tolerances in person to make sure the robot can accurately traverse its locales as needed. These tolerances are the acceptable ranges for the robot's perception and navigation, ensuring that its reliability traverses its environment. We will further test this on-site with real world conditions.

Along with this, we can test the tolerances for our vision. We need to develop a tolerance for what should be considered manure and what components of the robot itself are seen within the field of view of the LiDAR. This means we have to develop some amount of self-occlusion.

Requirement 4: No reliance on Wi-Fi or internet connectivity or service connectivity

Previous teams have implemented most of their code in methods that don't outright rely on internet connectivity. There was a large concern with previously developed computer vision at first relying on internet connectivity, but a workaround was found allowing us to utilize the previous code without internet.

Requirement 5: Software and hardware should be compatible with each other

We have taken steps to integrate the previous team's software environments and adapt them to the hardware. Previous teamwork should be developed as needed. Code developed in the virtual environment will be switched into the physical one.

VIII. Ethical Considerations

There are a lot of ethical considerations when it comes to this Autonomous Paddock Mucking Robot. To achieve full implementation of this product for commercial use, one must make sure that it doesn't violate any IEEE or ACM principles.

IEEE Ethical Considerations

1.03. Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be on the public good.

It is imperative to make sure our software works correctly with the prototype to make sure that stakeholders don't get affected by any problems that could arise from our software. Human stakeholders want the product to be efficient to reduce human labor while animal stakeholders don't want to be harmed by the machine whilst they go about their day. Our product is supposed to improve the quality of life for all involved, so our team has to make sure that our product is at least safe to use.

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.

Regardless of the perceived safety of our product, we must make sure that for the deployment of our robot, we must disclose to potential customers the chance of danger when working with our machine. If any person or animal could be harmed during development, we also must inform the appropriate parties involved otherwise accidental harm could be caused as well.

2.01. Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education.

Since the team is a student team, we have to be honest and realistic about what we can provide to our employer while trying to strive for professional-level work. Since we also inherit this from previous student teams, we also have to be understanding of any limitations that could arise from working with the residuals of previous teams as well.

3.08. Ensure that specifications for software on which they work have been well documented, satisfy the users' requirements, and have the appropriate approvals.

Our final product should have good documentation since it will be commercialized in the future, which means a professional team will further develop our prototype. To adequately pass on our project, our team needs to provide

good documentation and needs to provide adequate information on how to implement and improve things we missed. This will also assist our colleagues during this project as well.

ACM Ethical Considerations

1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.

Our final product should serve to solve the outlined problem of mucking paddocks in a way that makes it easier for the user to run their facility. This would contribute to human well-being by simplifying an undesirable task and allowing for human labor resources to be redirected elsewhere. If we violated this principle and created something that detracts from overall human wellbeing, we would have failed at our main objective.

1.2 Avoid harm.

The final product must avoid harming both the user and the animals around it. We do this by identifying obstacles around the robot. These principles could be violated since the robot is autonomous with little supervision and requires the handling of untrained individuals who could hurt themselves when using the robot. If these principles are violated, this means that the Autonomous Paddock Mucking Robot is not safe to be deployed around both workers and animals, and revisions will be required.

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

As we have worked on development, we have been working with both software and hardware from previous teams. It is important that we keep this principle in mind, especially as we navigate challenges with these previous implementations. Failure to respect the work that goes into production of new computing artifacts could lead us to struggle with implementing these previous works. If we violate this principle in our own work, we could underestimate the time and energy required to meet the deliverables set out for us and could end up not reaching our goals.

2.1 Strive to achieve high quality in both the processes and products of professional work.

Our final product should be high quality, as should our work on that product. Creating a low-quality product and violating this principle could lead to frustration and added costs for future consumers. Not performing high-quality work on the development of this product could lead to struggles to troubleshoot and solve issues that may come up on a large scale. On the small scale of these five weeks, failure to perform quality work could affect the development of groups coming after us.

3.6 Use care when modifying or retiring systems.

Our final product must be easy to maintain and retire by an untrained individual. It must be easy to detect when the robot is in need of maintenance or retirement. If this principle is not followed, the robot could malfunction or run at suboptimal levels without the user even knowing. This could lead to frustration in the best case and potential harm to the user and the animals in the worst case.

3.7 Recognize and take special care of systems that become integrated into the infrastructure of society.

Our final product could become integrated into the infrastructure of society, as farming is a large industry in the US. It is important that we respect this principle and take special care when developing and testing this system.

Michael Davis Tests

Reversibility Test:

The Autonomous Paddock Mucking Robot stands to take away jobs from low skill labor on many livestock farms. For Longhopes Sanctuary in particular, their volunteer employees would simply have more time to take care of other

positions in the shelter, but this is not always the case. If this product were commercialized, individuals in charge of mucking could lose their jobs. On the other hand, having to scoop up manure could be seen as such an undesirable job, that it would be an overall benefit to every worker if the pool of these jobs was smaller.

Product Test:

The Autonomous Paddock Mucking Robot may be developed as a commercial product in the future, which means that we should have high-quality development and adequate documentation for teams after us to pick up where we left off.

IX. Project Completion Status

The team predominately focused on implementation of previous groups' materials and establishing a foundation for future teams to work on. Our improvements spanned both software and hardware, including both documentation and improvement on previous work done. There is still further work to be done, both on the hardware and software sides of the project.

Off-Site Test Results

1. Establish connections:
    1.1. The team established a connection between the base station and a local machine by using U-Center software to interpret data. This verified that the base station was configured properly and was able to define its location within an accuracy of approximately 4 feet. Location was verified through Google Maps.
    1.2. The team was able to establish a connection between the rover station and a local machine by using HW Virtual Serial Port software and U-Center software to interpret data. This verified that the rover station was configured properly to read RTCM data from the base station. The team verified this by confirming that the rover's location had an accuracy within approximately 4 inches which continuously updated as the robot changed locations. Location was also verified through Google Maps and U-Center software.
    1.3. The team did establish a connection between the Orin and a local machine by running the command 'ssh paddock-pal@192.168.131.244' and providing the password 'mines'. This connection was verified by running the program './dungbot' which launches the controller commands.
    1.4. This team was able to establish a connection between the Orin and a webcam. To prove that the integration worked, the Orin was connected to a monitor and peripherals. The team was then able to pull up the camera view on the Orin and verify that its output matched the live feed of the webcam.
2. Camera Manure Detection
    2.1. Our team connected a webcam to Orin in order to test the camera's compatibility with the Orin and if it would interact well with the system. We saw little issues outside of incompatibility issues with Cheese, the default Linux camera software, which we quickly adapted to by downloading Kamoso, an alternative piece of software.
        2.1.1. Using previous teams' datasets along with new photos we took during data collection, we were able to test the model and verify its efficacy.
        2.1.2. False manure allowed us to preliminary testing on our live camera feed code for manure testing which allowed us to develop confidence for bringing it to on field testing. We noticed some false positives, but we thought this was due to the fact that we mostly tested this inside.
    2.2. We were predominantly able to test inside a carpeted room with false manure. Previous teams mostly did different locales in their dataset, but we wanted to take a peek at different lighting conditions as well.
    2.3. Test different methods of visual feeding for the camera
        2.3.1. Since we had previous data sets from previous teams, we were easily able to test the efficacy of our model on pictures. Since we also did data collection, we were able to recalibrate our confidence threshold that was being used for our model as well. When taking new data from our preliminary on-site visit

Commented [M(1]: @Luke Vo (Student)

2.3.2. Our off-site testing for videos mostly was built around our preliminary on-site data collection for videos. Previous teams didn't record videos and during this time, we didn't have any false manure so we planned to record videos on-site so that we could do some off-site development.

2.3.3. Most of our off-site testing for live camera feed was dedicated to simply getting it working as a whole, but we were able to get preliminary testing on our false manure to get a gauge of how it would interact with identifying manure.

3. Test Shape for Pathfinding
    3.1. Manually enter the GPS boundaries
    3.2. Program creates a list of goals, 2m apart, spanning the entire paddock defined by the boundaries
    3.3. Allow robot to create a dynamic path to the first goal in virtual environment
        3.3.1. Ensure that as the robot navigates to each goal, obstacles are detected and avoided
    3.4. Allow robot to continue to navigate to nodes until 95% of the paddock is traversed
    3.5. Document how all components interact

Commented [M(2]: @Grace Van Gorder (Student)

On-Site Test Results

1. Empty Paddock (containing manure, but no donkeys)
    1.1. Manually gather GPS boundaries and stationary obstacles
        1.1.1. Define these coordinates in pathfinding algorithm
        1.1.2. Allow robot to create path in virtual environment
        1.1.3. Allow robot to traverse path in virtual environment
    1.2. Manually traverse paddock:
        1.2.1. **Terrain Observations:** The robot was powerful enough to drive through every terrain such as ditches. Additionally, if the ground is not flat, the rake can be caught on rough terrain, stopping the entire robot.
        1.2.2. **Ground Condition Observations:** The robot was driven on dry and slightly muddy ground conditions. This did not alter the performance of the robot's operations.
        1.2.3. **Turning Radius Observations:** This robot has a minimum turn radius of 2.7 meters.
        1.2.4. **Path Completion Observations:** The team found that at maximum speed the drivetrain would often fail after about 2 minutes of constant driving. Excessive turning aggravated this observation, shortening the failure time to about 1 minute. The team was able to deduce that the drivetrain was pulling too much current from the ESC Motor Controller. Once not at max power, the drivetrain did not fail at any point in testing.
    1.3. Find parameters for scooping command
        1.3.1. Timing: If controlled manually (via time delays as opposed to limit switches), the wrist takes about 6.4 seconds to complete a full rotation. The arm takes about 1.5 seconds to go down, and about 8 seconds to go up (when weighed down)
        1.3.2. **Weight:** The arm struggles to lift heavy piles of manure. Smaller scoops or a stronger arm is necessary.
        1.3.3. **Placement:** The arm should be fully down (limit switch pressed) so that the rake is fully depressed against the ground.

Commented [M(3]: @Ryan Manley (Student)

2. This team made several observations regarding donkey and robot interactions:
    2.1. **Observation 1:** Most donkeys feared the robot while the device was stationary. They would come near the device but not get close to it
    2.2. **Observation 2:** Most donkeys feared the robot while the device was in motion. They would actively run away from the device or leave the paddock entirely.
    2.3. **Observation 3:** The was one notable exception to Observation 1 and 2. A donkey named Gillie was extremely intrigued by the robot, as shown in Figure 7. Any chance he got, he would approach the device, sniff around, bite various components, and lick the battery. Due to the drooling condition this donkey suffers from, the team had to be very diligent to ensure that all electrical components remained dry. This reinforced the need for waterproofing. This also highlighted a requirement that should be

added to the final product. This team now believes that **all individual components should be easily sourceable and replaceable in case of damage**.

Figure 7: Gillie vs. The Robot

3. Full Computer Vision Manure Detection Testing
    3.1. Within bright conditions in the sandy paddocks, most instances of manure could be identified at relatively high thresholds. Some smaller nuggets of manure were not detected until we allowed lower thresholds, but lower thresholds meant more inaccuracies which lead to false positives such as aspects of the donkey being detected (mostly eyes and hooves in the corner of the camera's view).
    3.2. Within low-light conditions like that of a barn's interior, without setting the threshold to be low, the number of detections massively decreased. Along with this, false positives occurred whenever dark hay or similar items were on the ground.'
    3.3. When transitioning between bright conditions and low-light conditions, the camera we are currently using basically can't detect anything at all since, the camera's view is over exposed with light or darkness which means that we may have to adopt a better camera or develop behavior in the robot to adapt to this.
    3.4. An idea of false positives that should be exclusion trained by future teams was developed.

Hardware/ Movement

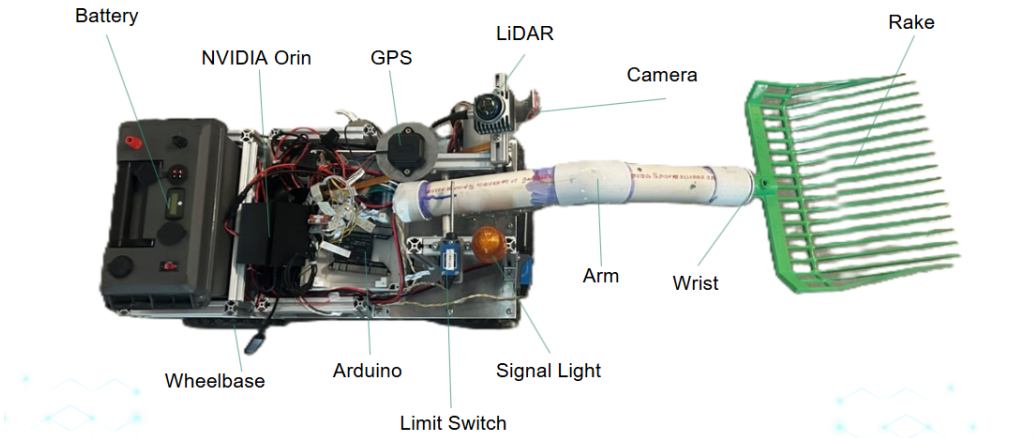


Figure 8: Labeled picture of the robot

We have had success fixing a few of the hardware issues, and our current prototype is shown above (Figure 8). The main issue we identified was that the wheels that were used to replace the wheels of the original RC car were slipping off the chassis. Upon further inspection of the connection between the wheels and the axis, we discovered a 3D printed adapter that had been reinforced with superglue and was not working to hold the wheel in place as it twisted and turned during normal operation. We 3D printed a new adapter to properly connect the wheels to the base.

Additionally, we rewired the interface between the Arduino and the Orin, utilizing a ribbon connector and cage clamps to avoid single pin connectors, which are pulled out easily.

Another issue that was immediately identified was the lack of connection between the camera and Orin. The ribbon connector that came with the camera was nowhere near long enough to connect the box holding the Orin at the center of the base and the camera, which was strategically mounted very far from the other electrical components so as not to obscure the view. We ordered and attached a longer ribbon cable, but we discovered that we would still require an adapter to attach the ribbon cable to the Orin. At this point our work on manure detection had also led us to realize we might want a higher quality camera anyways, and so the issue was never fully resolved. This led us to using a team member's old webcam and using it was a suitable replacement for the camera.

There were a few additional issues that we found with the hardware that caused struggles with our non-functional requirements. Another big issue we were made aware of during our initial hardware inspection was that the robot was not thoroughly waterproof. There was a waterproof box to hold the Orin and the Arduino, and the wire connections between the two, but that box was missing the lid, and we were never able to locate it. We ended up purchasing a larger box to hold those elements, but we did not have sufficient time to install this box and will leave it to future teams. The other issue with waterproofing that we discovered was that cable bladders had not been used for all connections outside of the box, which is necessary to waterproof the robot. We did not have time to install these bladders, although we have a few in storage.

One of the main issues we discovered through on-site testing was that the new wheels were too large for the suspension and were rubbing against both a plastic rudder and the 80/20 base of the robot itself. We were able to remove the rudder which helped with this issue, and in turn with the robot's mobility, but permanent hardware changes will have to be made to resolve these issues more thoroughly.

Pathfinding

Pathfinding algorithms were developed by previous teams, but only inside of a Gazebo environment. This environment was difficult to access and required X11 forwarding for any GUI, which caused it to be difficult to access and run on many types of machines. We documented these issues and produced commands and instructions to access these various environments. Once accessed, we had access to an algorithm that would run from a start point to an end point and avoid obstacles along its way. We were able to modify this algorithm to traverse the entire paddock in a graph of targets, two meters apart, which allows for full visual traversal of the paddock, while also avoiding obstacles. The robot can run this in any virtual world. There was code present in the previous repos for a computer vision node, but it was implemented using textures such as April tags, which do not render in Docker environments. After discovering this we decided to focus on implementing GPS first and successfully read manually entered GPS coordinates. Manure detection has yet to be implemented in the virtual environment, and all pathfinding code still exists in this virtual environment and has not yet been pushed to I2C or tested on the robot. The Gazebo virtual environment and corresponding visualization in RViz are shown below.

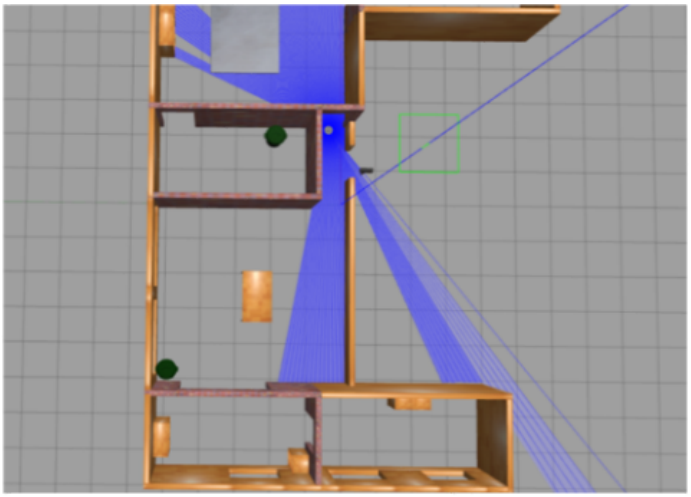


Figure 9: Simulated turtlebot3 house inside of Gazebo environment

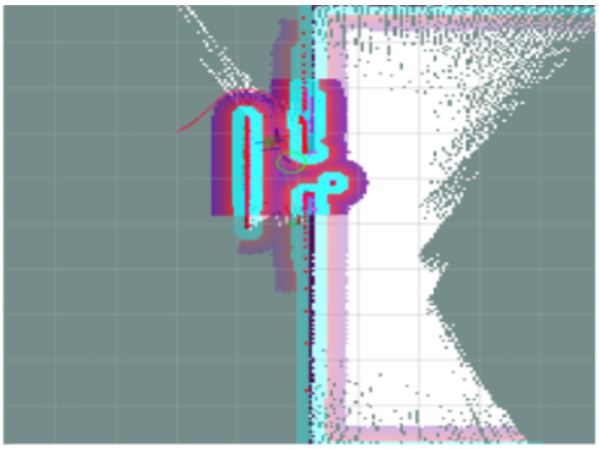


Figure 10: LiDAR visualization of the above Gazebo environment inside of RViz

GPS & LiDAR

This team was able to develop code and successfully configure a GPS system to provide latitude and longitude coordinates within a required level of accuracy of approximately 4 inches. It is configured by two subsystems: a base station and a rover station.

The base station is set up regionally (somewhere on the property) and is composed of two main components. The first of which is a main antenna that connects to as many satellites as possible and reads in raw locational data. This must connect to at last 10 satellites before the data is considered accurate, but with a clear view of the sky should connect to approximately 40 satellites. This raw data is then fed directly into the second component, a Sparkfun Reference Station. The reference station then runs a minute long survey, simply gathering as many data points as possible. After this survey is completed, it averages the gathered data points to formulate the antenna's current location. The latitude and longitude coordinates generated in this step have an accuracy within 5 decimal places or approximately 4 feet. Then the reference station transmits RTCM data (correction data) across a TCP port to be received by the rover station.

The rover station is set up on-board the robot and composed of a local antenna, a data receiver, and the Orin. The local antenna operates similarly to the main antenna; however it transfers its data directly to the receiver. The receiver then averages and interprets this data before transferring it directly to the Orin. The Orin then reads in the RTCM data transmitted from the reference station to increase its coordinate precision to 6 decimal places or approximately 4 inches of accuracy. These coordinates have been published too ROS.

This accuracy was verified through U-Center software. This configuration is illustrated in Figure 11.

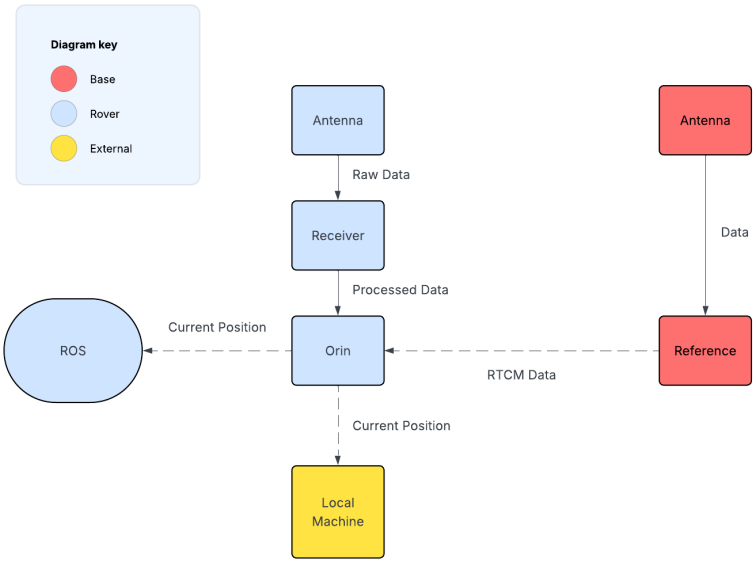


Figure 11: GPS Configuration

Furthermore, a display of this architecture in action is shown above, and its real-life setup is shown below. Additionally, this team has created extensive documentation regarding setup for this configuration including helpful information that may be needed if this configuration ever needed to be updated.

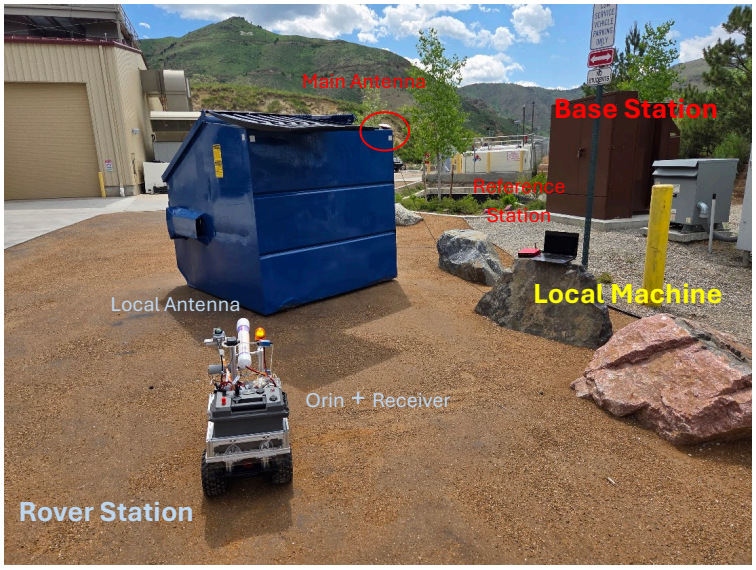


Figure 12: Real world application of full GPS configuration

This team was able to configure the LiDAR for polling; however, data has not been read from the device to prove accuracy yet.

Manure Detection:

Much of the previous team's application of the model was dedicated to an ROS environment, which meant that developing a method suitable for running the model was needed, especially for demo purposes since as a team, we had issues interacting with previous team's environments. Since the previous team's environment required a specific environment, but the environment could not interact correctly with any camera, it was decided that new code should be written and then later adapted to the ROS nodes.

No new models were trained since the original training was set from a Roboflow library with the offline improvements that have been implemented by previous teams. Many issues in the computer vision remain as it may falsely identify specific objects (orange objects or donkey eyes) and has issues with identifying things in certain lightings without adjusting the confidence threshold of the model so we may

We have added the option to apply NMS and live camera feed detection, which makes it easier to demonstrate to our clients while also allowing us to incorporate it more easily into the ROS nodes. NMS removes duplicate bounding boxes on our model, which also reduces false positives in our object detection results. Video detection and live video detection were developed for demonstration purposes for our clients. Video detection is much more computationally heavy than the live camera feed detection that will be implemented into our final product. Live camera feed detection might be too computationally intensive, but the alternative is simply continually taking photos which was not preferred since the previous hardware team had to update the Orin due to storage constraints at a previous point. A photograph with our detection program running is shown below.

Figure 13: Screenshot of Live Camera Manure Detection

X. Future Work

Hardware/ Movement

We have left the issue of implementing the camera hardware to a future team. If given more time, we would have chosen a waterproof webcam with a higher resolution than the camera currently attached to the robot. It will be up to a future team to find a webcam with these specs and properly connect it to the Orin.

We also ran out of time to rewire our Orin and Arduino connections inside of our new waterproof case and recommend a future team to do this work. This will require careful deconstruction of the current wiring and reconstruction of it in the new box as well as drilling holes and implementing cable bladders for wires that connect to other hardware components. Any other connections outside of the box that are not already waterproofed by cable bladders will also need to be carefully drilled in order to install them.

The wheels of the robot are currently rubbing on the frame due to their size, and changes will have to be made to the frame, the suspension, or the wheels themselves in order to mitigate this issue. We recommend a future team purchase new wheels for the robot that will be tougher, filled with dense foam instead of air (to mitigate issues with contact with sharp objects such as burrs), but smaller so they will fit underneath the frame. These could be purchased from the RC car company itself. Additionally, a rebuild of the suspension is highly recommended, to improve mobility in the paddock and reduce rubbing.

Pathfinding

Pathfinding only exists inside of the virtual Gazebo environment at this point. The code will need to be converted into a series of instructions that the Orin will push to the Arduino. We also recommend that future teams implement manure detection as a factor in pathfinding by identifying all manure as goals inside the code. All this functionality should be tested extensively before being placed inside the paddocks, to ensure safety.

GPS and LiDAR:

We need to finish implementing the GPS and LiDAR's hardware with pathfinding. The hardware works and the reference station and Orin are interacting with each other to obtain coordinates, but the pathfinding code needs to fully integrate its submodules. Future teams will also have to map over all the paddocks GPS boundaries with the documentation that we have developed for the GPS. Additionally, the GPS coordinates have been published to ROS, but the LiDAR data has not.

Manure Detection:

The model could be improved when it comes to visual edge cases. While it's fairly accurate as of now, it either isn't sensitive enough to detect manure in certain lighting conditions or misidentifying certain objects like that of the donkey's eyes or hoofs or orange play toys at Longhopes. This also applies to objects not fully in the camera's field of view as well. This means retraining a new model, making sure to supervise the model's learning closer to making sure it can exclude these objects. While LiDAR and obstacle detection may make these false positives less impactful, in a finalized product, it would probably be good to remove these. Transitioning lighting conditions is also a major issue, as the model can't identify manure when adapting to lighting conditions. Along with the issues of detection becoming poorer under certain lighting conditions, future teams will have to consider how they will tackle these issues in conjunction with the lighting shifts. Along with this, a higher resolution camera may be able to remedy these issues, especially issues that arise from different lighting conditions.

As seen in the two Figures 14 and 15 below, we can see that cutoff portions of objects (such as shadows) or circular objects that are orange in color *and* circular in shape can also be identified as manure. These have similarities to manure that our model can see, but to human eyes, can't be identified. This could be the model identifying the shape or seeing orange as a lighter shade of brown, but this is ultimately conjecture.

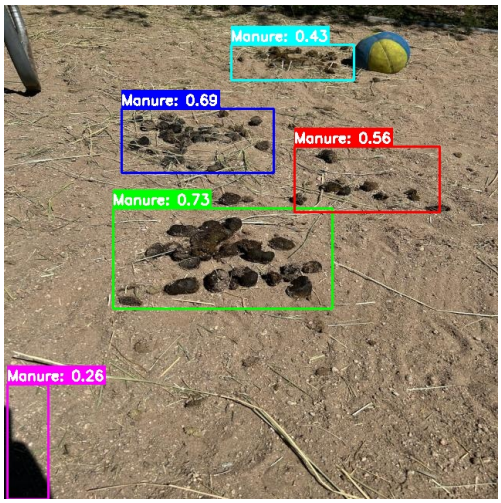


Figure 14: Misidentified shadow.

In addition to this, the computer vision needs to be integrated into the pathfinding decision making still as we have not fully integrated how we will use manure identification to designate all the goals. A robust method for turning detections into actional goals or states for the pathfinding logic is a major decision point for the next team. We can develop coordinate-based goals determined by the camera or line of sight navigation where the robot simply goes to the closest or more recent detection.

Our model doesn't distinguish between each piece of manure as of now. We could achieve this with deep sorting which would assign an ID to each piece of manure but ultimately, since our robot will be scanning the paddock, assigning a value to each value of manure would likely be a waste of computational power but it is important to consider going forward since future teams will have to further implement our components together. Each piece of manure is being detected without overlapping bounding boxes (due to NMS) which probably would make things like deep sorting easier to implement.

Figure 15: Falsely Detected Object Next to Manure Dump Spot

The initial ideas around this idea were either to calibrate the camera and drop a "goal pin" for the manure or to simply drive to the first pile of manure in line of sight. Future teams need to also decide between having manure detection be implemented through a live camera feed or by continually taking photos over time as well. Goal pins could be developed by calibrating the camera to a 3D environment and allowing it to support the ability to detect distance of some sort.

Some of the previously mentioned items would most likely be easiest to implement by training a new model. YOLOv5 was the easiest form of YOLO to use at the time period but newer things like YOLOv8 and YOLOv11 have better support with the Ultralytics Python Library which is an open-source library that would help improve implementation and integration with our model which would ease development in the long run.

XI. Lessons Learned

Throughout the project we were all introduced to new systems and technologies that we had not interacted with before. Many people on our team were introduced to ROS or Docker for the first time and having to learn these as we worked presented unique challenges and opportunities to learn. Working with new software libraries and learning how to work with them regardless of your native platform was generally pretty fulfilling as well. Notably GPS and LiDAR stood out in how interacting with them went.

We also had to adapt to a large code base, so the team learned a lot about managing and maintaining an existing system, as well as integrating multiple separate systems. We realized that to maintain a project's workflow, adequate documentation was massively important for introducing new teams. To alleviate the team's future struggles, we decided to have a larger focus on documentation as a core aspect of our project. When implementing previous ideas from the other team, we realized their ideas, since they didn't have the hardware prototype at the time, were a little incongruent with what our protype was capable of, meaning that we had to learn to adapt based on what was available.

Being able to work on a hardware focused project was new for some of our team members as well and working on the hardware given to the team was an enlightening experience. Getting all the components of the hardware to coordinate with the software and the complexities involved with doing so highlighted the importance of testing and taught us the aspects of system integration as well.

Computational power was something we had to consider at times, especially for some of the more software focused aspects of the project. This was especially in our team's minds since we heard that the Capstone Team had decided to purchase another Orin due to similar concerns. When it came to manure detection, we decided to apply NMS on top of the model previously given to us. Since we initially thought, we were going to use manure detection to drop a "pin" or goal for pathfinding. Reducing the overlapping detections would've made this much easier but since we didn't get to defining goals based on the visual detections from our model and planned to consider pivot towards something else, this ended up being additional computational cost for little reason.

XII. Acknowledgments

Our team would like to express our gratitude to our advisor, Donna Bodeau, who has provided guidance, feedback, and support throughout the project's lifetime. She was very understanding of a lot of the issues we faced throughout this project and helped encourage us to carry on.

Additionally, we would like to thank all the members of the previous three teams, who provided a basis of knowledge, technology, and work for us to build upon. They were frankly the foundation of the project, developing for us the starting software and hardware for us to integrate together.

We would also like to thank our client, Kathy Dean, founder of Longhopes Donkey Shelter, for allowing us to come to the shelter and do testing this semester. The onsite testing gave us helpful, invaluable feedback on our project, and she was very understanding of the issues that we faced.

Finally, our team owes an immense debt to the lovely employees at Book and Brew, the McDonald's in Bennett, and the vending machines in Labriola for keeping us fed and energized this summer.

XIII. Team Profile

Maggie Acheson

Undergraduate in Computer Science: Space

Graduate in Engineering and Technology Management

From Creede, Colorado

Favorite Donkey: Sweetie Pie

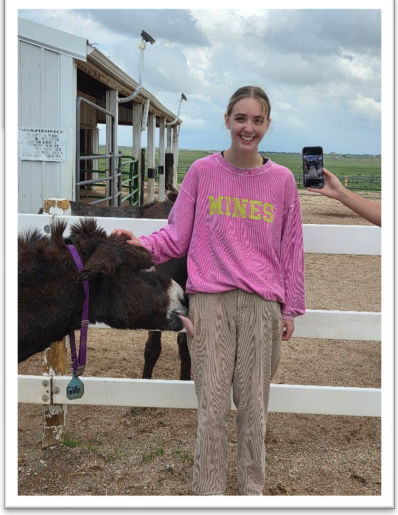
Ryan Manley

Undergraduate in Computer Science: AI and Robotics

Graduate in Robotics

From Fort Collins, Colorado

Favorite Donkey: Molly Brown

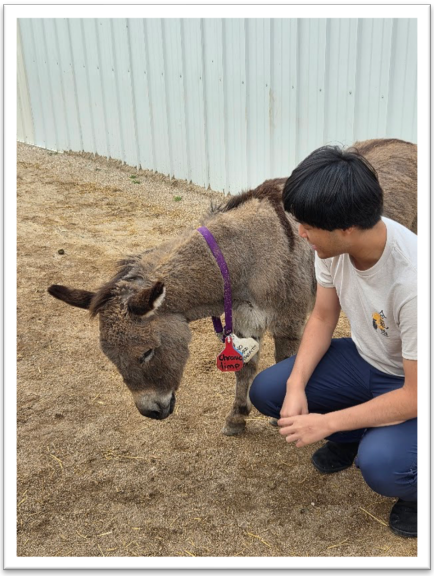


Grace Van Gorder

Undergraduate in Computer Science: AI and Robotics

From Austin, Texas

Favorite Donkey: Gillie



Luke Vo

Undergraduate in Computer Science: General

From Denver, Colorado

Favorite Donkey: Schroeder

References

Bernklau, A., Fernando, R., Luoto, J., & Shanahan, L. (2024). (rep.). *CSCI 370 Final Report - MuckBusters*. Golden, Colorado.

Coshiaosue, J., Coster, S., Doud, B., Henderson, N., Horsfall, L., Houghton, M., McNair, S., Parks, S., & Zindel, O. (2025). (rep.). *Paddock Pal Final Design Report*. Golden, Colorado.

Schuler, K., Collins, D., & Gordon, S. (2024). (rep.). *CSCI 370 Final Report - Blaster's Bucket*. Golden, Colorado.

Key Terms

Term	Definition
Arduino	*An open-source electronics platform based on being a "easy-to-use hardware and software platform".*
Base Station	*A regionally hosted GPS system location somewhere stationary on site. This is composed on an antenna and Sparkfun Reference Station*
Bounding Box	*A rectangle drawn over detected objects (manure in this case) in an image or video.*
Cage Clamp	*A type of electrical terminal block connection that uses spring pressure to secure wires, eliminating the need for screws or crimping*
CAM	*A mechanical mechanism converts rotation into linear motion. It is used in the arm in order to control the lifting mechanism of our hardware prototype. A CAM was used in order to prevent back driving the motor.*
Computer Vision	*A field of AI that enables systems to interpret visual data to make decisions. Here it is used to identify manure.*
Docker	*A platform used to simulate environments inside containers. It ensures consistent environments across machines for developers. We experienced compatibility issues with these containers in this project.*
ESC Motor Controller	*A device that manages the speed, direction, and braking of an electric motor*
Gazebo	*An opensource simulator that integrates with ROS, providing a virtual environment for us to test our hardware prototype in a simulated environment.*

GPS	*Global Positioning System used to determine latitude and longitude coordinates on Earth.*
HW Virtual Serial Port	*A software used to create a virtual serial port from an open TCP port connection. Older versions of U-Center had this capability, however modern versions do not,. This software performs this action for U-Center.*
I2C	*Inter-Integrated Circuit. A communication protocol that transfers data between our microcontroller (Arduino) and our main platform (the Orin).*
LiDAR	*Light Detection and Ranging. A method that measures technology by illuminating its surroundings with laser light and analyzes reflected pulses to determine the range. Used for obstacle detection and mapping.*
Limit Switch	*A sensor used to detect mechanical movements. Used here to prevent over travel of the robot's arm.*
Machine Learning	*AI that enables systems to learn from datasets and train models without explicitly being programmed. Used here for manure detection.*
mAP	*Mean Average Precision. Used to evaluate the accuracy of models for object detection. A combined evaluation of precision and recall.*
NMS	*Non-Maximum Suppression. A computer vision technique that removes duplicate bounding boxes around an identified object to reduce false positives and visually clear up visual clutter.*
Orin (NVIDIA Jetson Orin)	*The NVIDIA Jetson Orin is the primary onboard computing platform used in our system. It runs ROS2 and is meant to manage computer vision, pathfinding, and sensor data and communicates with the Arduino to control movement. It is essentially the hardware prototype's brain.*
Pathfinding	*A process to determine the optimal route while avoiding obstacles. Here it is used for your robot's ability to take the optimal path to its manure. It is implemented in our Gazebo simulation.*
PWM	*Pulse Width Modulation signals control the amount of power delivered to a device.*
Roboflow	*A platform that hosts image data sets for training computer vision models. Our manure detection model was originally trained on a Roboflow dataset.*
ROS	*Robot Operating System. A framework for developing and managing software components for our hardware prototype. This project specifically uses ROS2 for hardware integration.*
RTCM	*Radio Technical Commission for Maritime Services. The standard protocol is used to transmit data to GPS receivers.*
Rover Station	*The locally hosted GPS system onboard the robot. This is composed of an ANN-MB-00-00 antenna, a receiver, and the Orin.*
Sparkfun Reference Station	*A dedicated station designed for high-precision GPS/GNSS applications using Real-Time Kinematic (RTK) technology*
U-Center	*A software tool that visualizes and interacts with GPS data. We used it for debugging and viewing positional data.*
Ultralytics	*An open source team that has developed YOLOv5 and beyond which is the basis behind the vision here. This also could refer to the Python Library of the same name as well.*
YOLO	*YOLO stands for You Only Look Once, and it aims to recognize objects with high accuracy and speed in real time.*