EE 475 Smart Plant Caretaker Robot

Nathan Lee, David Luo, Archie Deng, Ryan Nguyen, Henry Do

Dept. of Electrical and Computer Engineering, Box 352500
University of Washington
Seattle, Washington 98185-2500 U.S.A.

Sponsored by Prof. Rania Hussein

March 11, 2024

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY *of* WASHINGTON

Team, Roles, and Responsibilities

Team Member	Roles and Responsibilities
Nathan	Programmed autonomous movement and assembled the movement related hardware of the robot. Project manager of the team.
Ryan	Programmed and tested the AI plant detection program. Project coordinator of the team. Created and proposed idea of smart watering robot to group, where we all agreed upon it
Henry	Programmed and tested the AI plant detection program.
David	Programmed the STM and 3D printed the water pump and STM attachment components for the robot.
Archie	Programmed the Raspberry Pi to connect to the STM and assembled the water pump.

Table 1. Team roles and responsibilities

Product Requirements Document (PRD)

The Smart Plant Caretaker is an automated robot designed to move around and detect plants that it encounters, watering them accordingly. If there is something in the way of the robot, it stops and takes a picture of its surroundings to detect whether the object in front of it is a plant before deciding to dispense water to water it. It repeats a cycle of moving around to find new plants to water once the object detected in front of it has been fully processed and evaluated.

The main goal and need of the Smart Plant Caretaker is to allow homeowners and gardeners to automate the process of watering plants. In the US alone, over 18 million households own gardens. Moreover, the average time spent gardening is 5 hours per week. The Smart Plant Caretaker would allow all of these individuals to save large amounts of time each week while maintaining a healthy fertile garden.

The Smart Plant Caretaker will be used in small and/or medium-sized gardens such as those in a homeowner's backyard. Homeowners and gardeners who strive for convenience would use this robot in order to reduce the amount of labor they would need to put in otherwise into watering plants in their garden.

The Smart Plant Caretaker will be used by initializing a command that activates the robot's autonomous movement, where it would begin moving around the garden looking for plants to water. As mentioned earlier, the robot will stop briefly when it detects an object in front of it to see whether or not it is a plant, where it would water the plant if it is one or do nothing if the object in front of it is not a plant. The only necessary actions by the user would be to fill the

water tank, place the robot near the plants to water, and initialize the starting command. There are no additional components needed to ensure the Smart Plant Caretaker functions correctly other than making sure to charge the battery in between uses. The robot will be used with another device such as a computer to remotely connect to and initialize it in order to begin moving around and performing its intended operations.

Realistic Constraints and Engineering Standards

With the idea of the Smart Plant Caretaker being able to autonomously move around and water plants that it detects, there are some boundaries and constraints that come with it.

1. **Environmental Limitations:** The Smart Plant Caretaker cannot be used in the rain due to its electrical components. This introduces a significant constraint as the device must avoid water from natural sources while being designed to dispense water. Ensuring that the robot is only used under appropriate weather conditions becomes essential.
2. **Camera Resolution and Lighting Conditions:** The robot utilizes a 1080p 8-megapixel camera which requires sufficient light to operate effectively. This constraint affects the robot's performance in dimly lit environments, impacting its ability to accurately detect plants and navigate.
3. **Motor Power and Load Capacity:** The motors, with a power supply between 3-6V, must be robust enough to move the robot and the additional weight from the water. This limitation affects the total weight the robot can carry and still navigate effectively, influencing the design and the operational time between refills.
4. **Water Dispensing Limitations:** The water pump's capacity to dispense water, powered by a 5V connection, and the mechanism's ability to release water accurately when a plant is detected, are crucial. The precision and reliability of this function are vital for the robot's efficiency in watering plants without waste.
5. **Sensor Range and Sensitivity:** The distance sensors, while capable of detecting objects up to 450 cm away, are programmed to react at 25 cm. This proximity constraint ensures safety and accuracy but limits the robot’s early detection capabilities, impacting its navigation and operational speed.
6. **Algorithmic and Movement Constraints:** The robot's mobility and decision-making algorithms are confined to the capabilities of the Raspberry Pi and STM32 board. The efficiency and effectiveness of these algorithms directly influence the robot's performance in plant detection and area coverage.
7. **Physical Design and Weight Distribution:** The design must accommodate the weight of the water, batteries, and electronic components while maintaining stability and mobility. This constraint impacts the overall size, shape, and functionality of the robot. We came to a consensus to create 3D-printed parts for the robot that would hold the STM32 board in place while utilizing the other free space around it to store the Raspberry Pi.
8. **Plant Detection Limitations:** The robot cannot detect newly planted seeds or very small plants, limiting its usefulness in certain stages of plant growth and requiring manual intervention for young plantings.
9. **Water Capacity:** The decision to use a 20oz water bottle balances between operational duration and mobility but limits the number of plants that can be watered in one cycle, affecting the robot's efficiency and user convenience.

10. **Wireless Operation and Energy Requirements:** The need for wireless operation necessitates battery packs for both motors and microcontrollers, influencing the robot's weight, size, and operational time.
11. **Size and Portability:** We decided on a robot chassis that is 10.8 x 6.6 x 2 inches, which is designed to navigate small/medium-sized gardens, imposing limitations on its applicability in larger or more complex garden layouts.

The constraints provide a foundation from which we derive our compliance and performance benchmarks. As we move towards detailing these engineering standards, it's crucial to understand that they are not merely technical formalities but are deeply intertwined with the practical limitations identified. These standards ensure the Smart Plant Caretaker not only meets but exceeds the expectations for safety, functionality, and environmental sustainability, turning challenges into opportunities for innovation and improvement. Below are some specific engineering standards that will guide the development and implementation of the Smart Plant Caretaker.

1. **Electrical Safety and Waterproofing:** Standards related to electrical safety and water resistance, such as IP ratings for electronic components, must be adhered to, ensuring the robot's safe operation in its intended outdoor environment.
2. **Wireless Communication:** The robot must comply with standards for wireless communication, ensuring secure and reliable operation when controlled remotely.
3. **Material Safety and Sustainability:** Materials used in the robot's construction must meet environmental and safety standards, such as RoHS-2 and Pb-free compliance, ensuring the device is non-toxic and environmentally friendly.
4. **Mechanical Integrity:** Standards related to mechanical design and load-bearing components must be followed to ensure the robot's physical durability and operational reliability.
5. **Software and Cybersecurity:** Given the robot's remote operation capabilities, software standards, including those for cybersecurity, must be considered to protect against unauthorized access and ensure user data privacy.

By addressing these realistic constraints and adhering to relevant engineering standards, the Smart Plant Caretaker's design can be optimized for safety, efficiency, and user satisfaction, while navigating the complexities inherent to robotic gardening solutions.

System Requirements Document (SRD)

Parameter	Test Conditions	Description
Precision/Accuracy Requirements	- Variety of plants in various conditions including indoor and outdoor with different backgrounds	- Test various plants with the object detection program to ensure the AI can recognize a variety of plants

Control Requirements	-Indoors	-Remote control of Pi from PC via Wifi
Functional Requirements	-Indoor flat surface with picture of plant on phone -Outdoors with uneven natural surfaces with real plants	- Ensuring the robot can recognize and water plants using the object detection program and water pump - Testing the robot to make sure it can recognize and avoid obstacles
Size / Cost	- Run the movement program while we have 19fl. oz of water installed on the robot - Monitor the budget and prices of materials	- Budget no more than $200-300, robot should hold 19 fl.oz water bottle

Table 2. System Requirements Document

Project Schedule

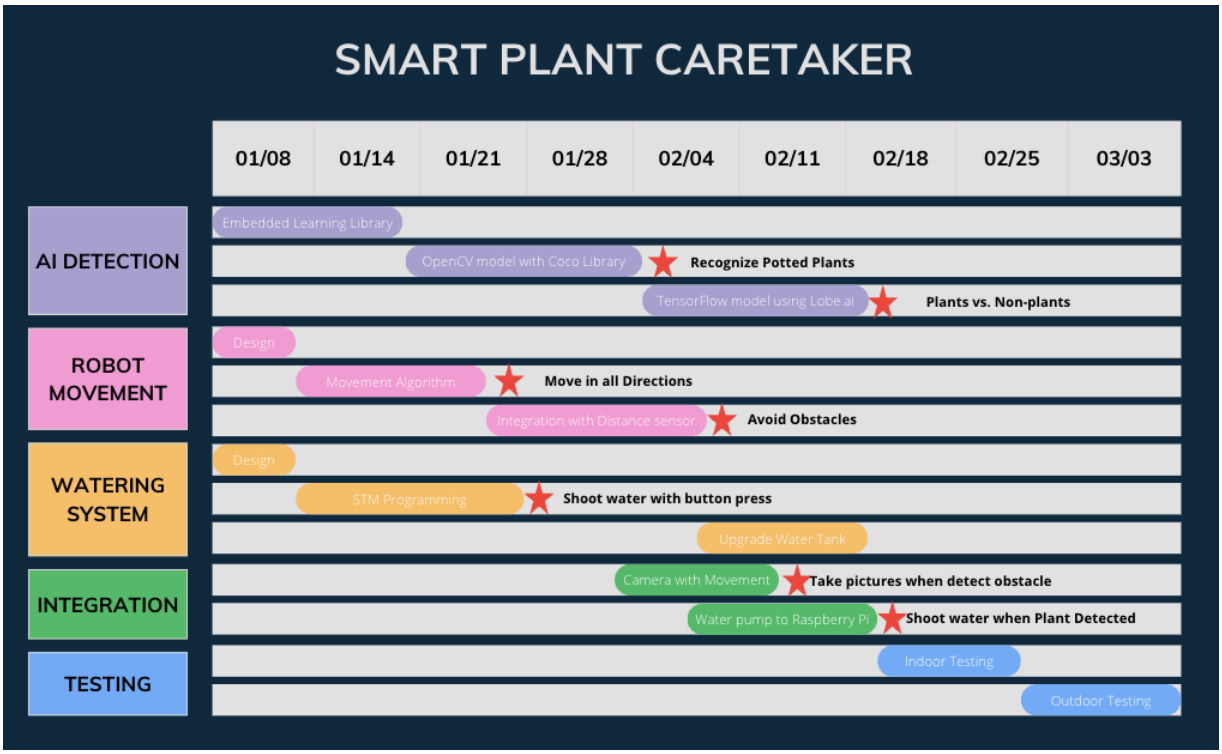

Figure 1. Gantt chart of the project

Project Resources

Hardware Components:

- **Raspberry Pi:** The main controller for the robot, handling processing and decision-making tasks.
- **STM32 Microcontroller:** Secondary controller, possibly for managing real-time tasks or specific sensors.
- Robot Frame: The physical structure of the robot, which houses all components.
- **Raspberry Pi Camera:** For capturing images for plant detection.
- **Water Pump:** For dispensing water to the plants.
- **Water Bottle:** To store water for the watering mechanism.
- **H Bridge:** To control the motor directions.
- **Relay:** To switch higher power circuits with Raspberry Pi.
- **Distance Sensors:** For obstacle detection and navigation.
- **Batteries and Portable Power Source:** To power the Raspberry Pi and other components.
- **Edimax WiFi Adapter:** To provide wireless connectivity to the Raspberry Pi.

Software and Libraries:

- **STMCube:** Software development platform for the STM32 microcontroller.
- **TensorFlow, Lobe.ai:** For developing and implementing the plant detection model.
- **Other Raspberry Pi Libraries:** Additional libraries needed for controlling hardware components or data processing.

Fabrication and Assembly Tools:

- **3D Printer:** For creating custom parts or enclosures.
- **Glue Gun and Electrical Tape:** For assembly and wiring tasks.

Miscellaneous:

- **Work Location:** CSE building lab, where the project development and assembly will take place.
- **Budget:** Approximately $200, to cover the costs of all listed resources.

Smart Plant Caretaker Bill of Materials (BOM)					
Item #	**Description**	**Vendor**	**Part Number**	**Quantity**	**Cost**
1	Raspberry Pi 4	Provided by EE 475	Raspberry Pi 4	1	$35
2	STM32 board	Provided by EE 475	STM32F407G	1	$21.17
3	Mini water pump	Amazon	B0BDSRNXX8	1	$3.00
4	Edimax wifi adapter	Amazon	B008TCWPNG	1	$10.00
5	Raspberry Pi camera	Amazon	B01ER2SKFS	1	$20.00
6	3D printed design	The Mill at UW	N/A	1	$2.00

7	portable power source	Amazon	B0C5ZP3XHS	1	$10.00
8	Gatorade bottle (20 oz)	UW HUB Vending Machine	N/A	1	$2.00
9	AA Batteries	Amazon	B0BFFNZLYY	8	$7.00
10	Relay module watering system	Amazon	B0BWQ6RD95	1	$10.00
11	distance sensor	Amazon	B0B1MJJLJP	2	$6.00
12	Robot set: Motors (4) + Frame (1)	Amazon	B07DNXBFQN	1	$22.00
13	H bridge	Amazon	B07BK1QL5T	2	$12.00

Total Cost	$160.17

Table 3. Bill of Materials of the project

Outline of Experiments

The following section outlines the series of experiments conducted to develop and refine the Smart Plant Caretaker. These experiments are crucial for testing the functionality of each component and their integration into the overall system. The experiments have been categorized based on the primary functions of the robot: AI detection, watering system, and robot movement

AI detection

The first experiment involved attempting to utilize the Embedded Learning Library for object detection. The objective was to evaluate its suitability and effectiveness in recognizing plants. However, due to difficulties in downloading and implementing this library, this approach was abandoned.

Subsequently, the experiment shifted to using a pre-trained Mobilenet model from OpenCV that utilizes the Coco library. The goal was to test the robot's ability to recognize common objects, particularly potted plants, to validate the basic functionality of object detection.
The third experiment involved changing the image recognition code to a binary classification model, distinguishing between plants and non-plants. This was aimed at refining the AI's ability to accurately identify target objects for the purpose of watering.

Due to time constraints, we transitioned to using Lobe.ai for a faster solution in developing a binary classification model. The experiment focused on training a TensorFlow Lite model using a dataset composed of plant images and random non-plant images from the Coco library. The efficiency of model training and integration with the Raspberry Pi hardware was assessed.

Watering system

The first experiment in this category involved creating a motor-based mechanism to control the water flow from a bottle. This method's feasibility, reliability, and effectiveness in regulating water were evaluated.

Following the initial test, the system was upgraded to utilize a water pump controlled by the STM board. The experiment aimed to assess the reliability of using a relay module for turning the water pump on and off based on signals from the AI detection system.

In this experiment, the capacity of the water tank (19fl. Oz) and its ability to water approximately 35 plants were tested. The efficiency of the water distribution system and the adequacy of the water tank's capacity were the main focuses.

Robot movement

This experiment involved setting up the robot's movement mechanisms, including the wiring of an H bridge connected to motors and distance sensors. The objective was to evaluate the basic forward, backward, left, and right movements controlled by the Raspberry Pi.
Integrating distance sensors, this experiment aimed to test the robot's ability to detect obstacles within a 25 cm range and execute appropriate maneuvers to avoid them.
The final experiment in this series tested the implementation of random movement patterns. This involved using a while loop to alternate between obstacle detection and initiating random directional movements to simulate autonomous navigation in a garden environment.

Integration

The overarching logic governing the Smart Plant Caretaker is designed for efficiency and effectiveness, ensuring that the robot autonomously navigates the garden space while fulfilling its primary task of watering plants.

Upon activation, the robot enters into a state of random locomotion, allowing it to explore the garden terrain freely. This phase is crucial for covering as much ground as possible and for identifying potential plants in need of watering.

As the robot moves, its integrated distance sensors continuously scan the immediate vicinity for objects. Should these sensors detect an object within a 25 cm range, the robot will come to a halt. This preventive measure ensures the robot avoids collisions and assesses whether the object detected warrants further action.

Once an object is detected and the robot has stopped, the Pi camera is activated to capture an image of the object directly in front of the robot. This image is then processed by the plant recognition program, a sophisticated algorithm trained to discern plants from non-plant objects. If the recognition program identifies the object as a plant, the robot activates the water pump. This triggers a controlled release of water, adequately hydrating the plant without wastage. Following this, the robot resumes its random navigation to continue its search for other plants. In instances where the object is determined not to be a plant, the robot will engage in a brief retreat,

moving backward for a set distance, before continuing its random movement pattern. This ensures the robot does not remain fixated on non-target objects and covers more area for potential watering opportunities.

To avoid situations where the robot might be trapped or stuck, a counter mechanism is employed. If the robot encounters an object three consecutive times without a clear path forward, it will enter a halted state. This safety measure prevents the robot from continuous ineffective movements in tightly confined spaces or in scenarios where no clear navigable path is present.

Below is the UML chart and sequence diagram for the top level logic of the robot.

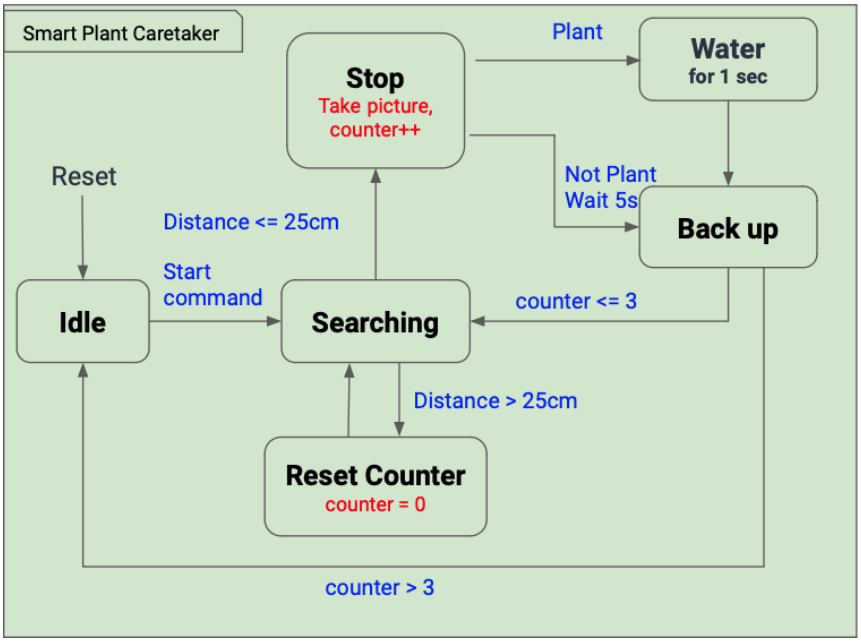

Figure 2. UML chart of the project

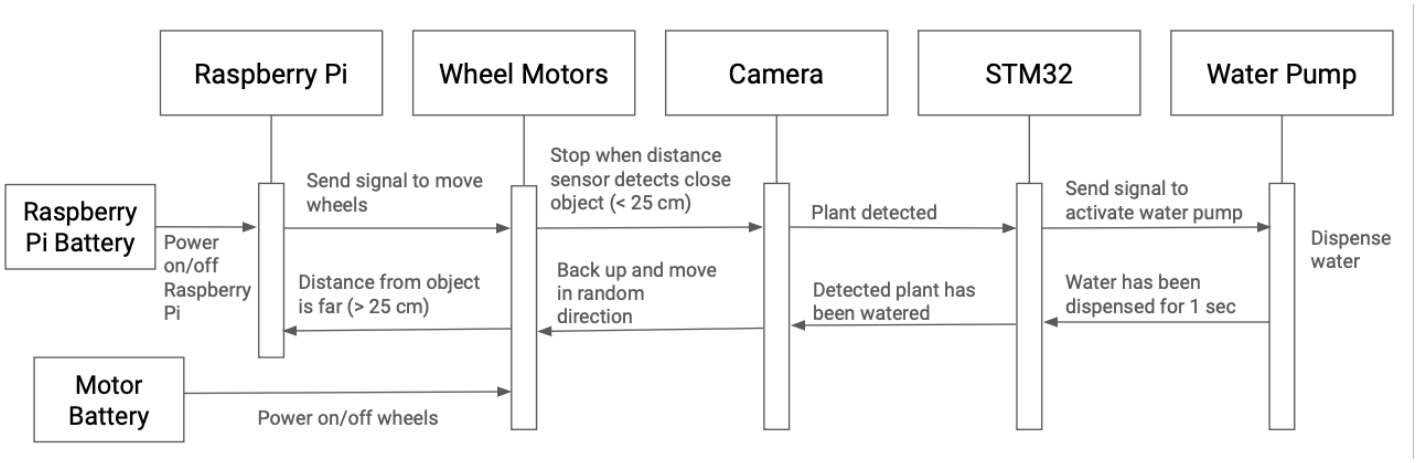

Figure 3. Sequencing Chart of the project

Trial Designs

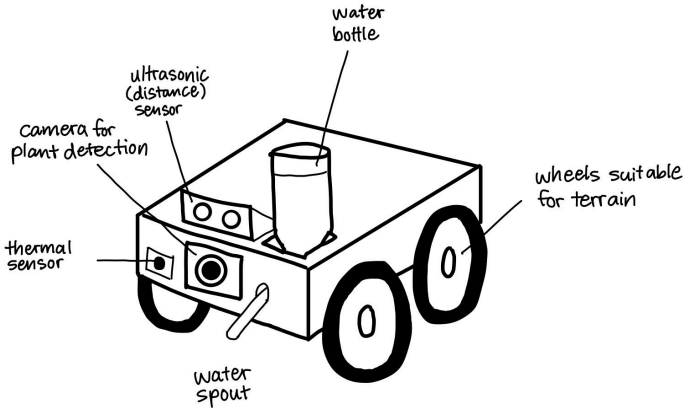

Figure 4. Initial diagram of the robot.

When thinking of a design for the Smart Plant Caretaker, we thought of a robot that would be suitable for outdoor terrain and contain the necessary devices and sensors to be able to detect plants. We also needed a water bottle to store the water needed to water the plants along with a way to release the water so that the plant can be watered when it is detected using a camera. Our first design consisted of the rough sketch above, where it uses a camera for plant detection, a distance sensor for sensing objects and plants, a thermal sensor to detect priority plants, a water bottle and spout for dispensing water, wheels suitable for outdoor terrain, and a small chassis for the robot.

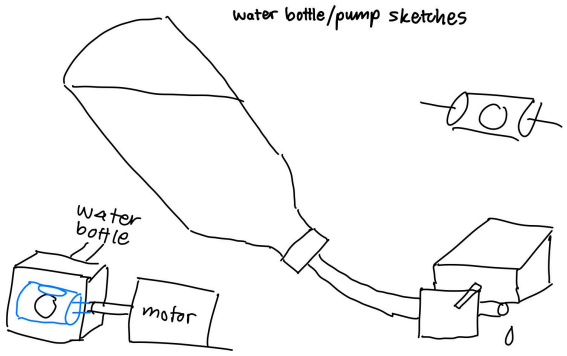

Figure 5. Initial design of the watering system.

We also needed to think of a way to store the water on the robot and how it will release the water when a plant is detected. This rough sketch involves the use of a water bottle turned upside down and a motor connected to a rotating stopper to stop and release the water when appropriate. However, we did not use this design due to its complexity and additional materials needed. We instead decided on purchasing a water pump that we would then place inside of the bottle and dispense water out when the appropriate electric signal is sent to it.

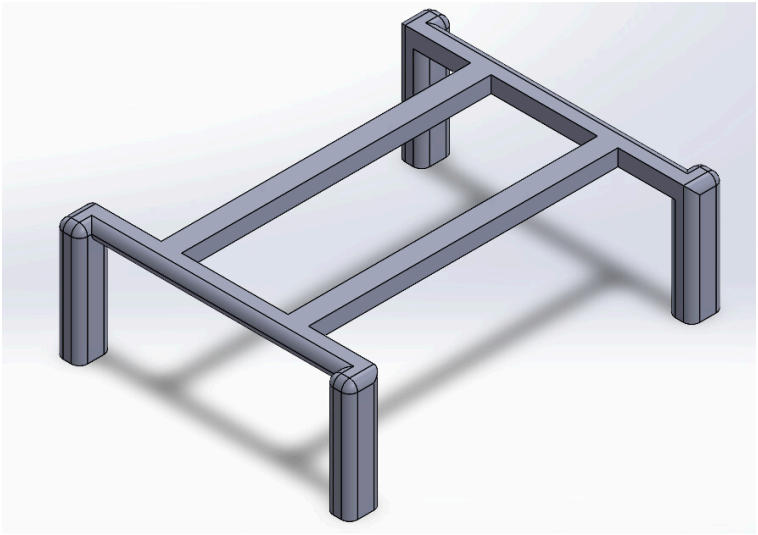

Figure 6. Initial design of the support for the STM board.

This CAD model is a bridge and is one of the two 3D-printed pieces used in our design. This design in particular is used to hold the STM32 board in place when fitted to the robot while providing easy access to its ports and wire connections. There were many adjustments regarding the measurements made with this design due to numerous decisions we made about where it would fit on our robot and what parts can fit under the bridge piece given the height.

Despite measurement differences, the overall design of the bridge piece has been kept the same, and the model shown above represents our final design. We would then use this bridge design to hold the STM32 board on top of it and store the battery used to power the motors for the wheels underneath, placing it towards the back of the robot to make room for necessary sensors in the front.

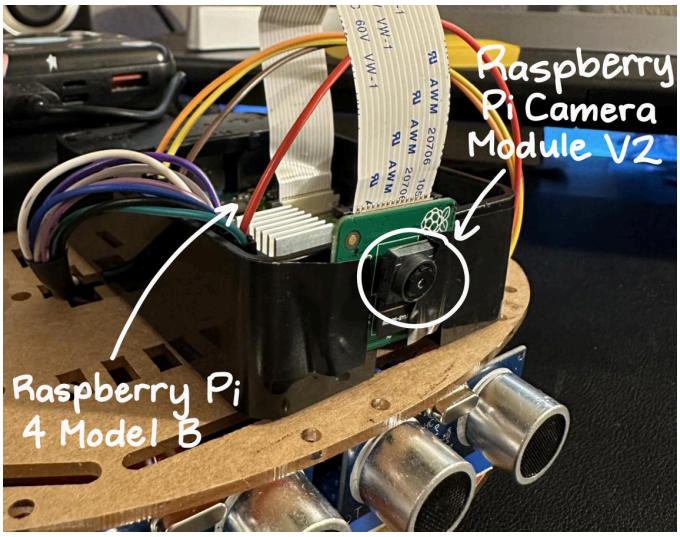

Figure 7. Pi camera connection.

This picture outlines the layout of the Raspberry Pi and the camera module that will be used for plant detection. With the Raspberry Pi, we decided to place the board at the front of the robot due

to the camera module needed for detecting plants. The camera needs to be able to capture as much of its surroundings as possible, and placing it in the back would not be a good choice due to the components that may get in the camera's view. We then secured the Raspberry Pi camera to the side of the Raspberry Pi using electrical tape to stabilize it and to prevent it from making contact with other electrical components.

Figure 8. Support for the water bottle.

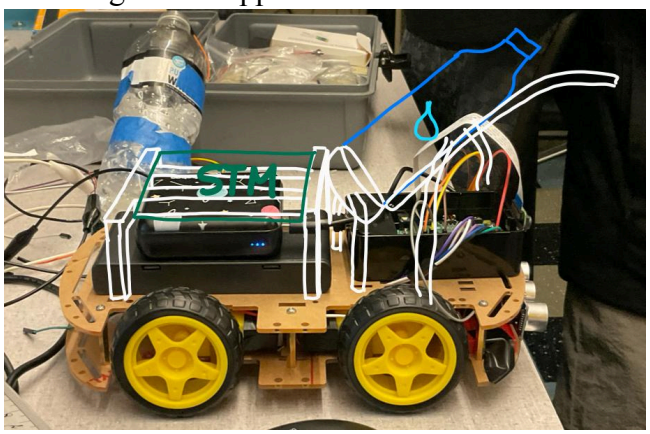

Figure 9. Design of integration of parts

Figure 8 above shows the CAD model of the piece used to hold the water bottle assembly on our robot. The sketch in Figure 9 shows an outline of where to put the STM32 board, mentioned above, and the water bottle assembly in order to connect it with the rest of the robot.

We previously used a plastic water bottle (as shown in the left picture above) to hold water for watering the plants, but we then realized that the water would run out quickly with a smaller size. We also miscalculated the dimensions for the radius of the part used to hold the bottle in place, which was too big for that plastic water bottle. We then did some calculations and found that a 20 oz Gatorade bottle fits very snugly into that spot. Due to the Gatorade bottle being more sturdy,

larger, and less prone to being warped compared to the plastic bottle that we used previously, we decided to switch to using it instead.

We decided to place the water bottle structure towards the front of the robot to allow water to reach the plant sooner, and it also provides easy access to connecting the wires together to the STM32 board as well as providing space for the Raspberry Pi board and camera module right below it.

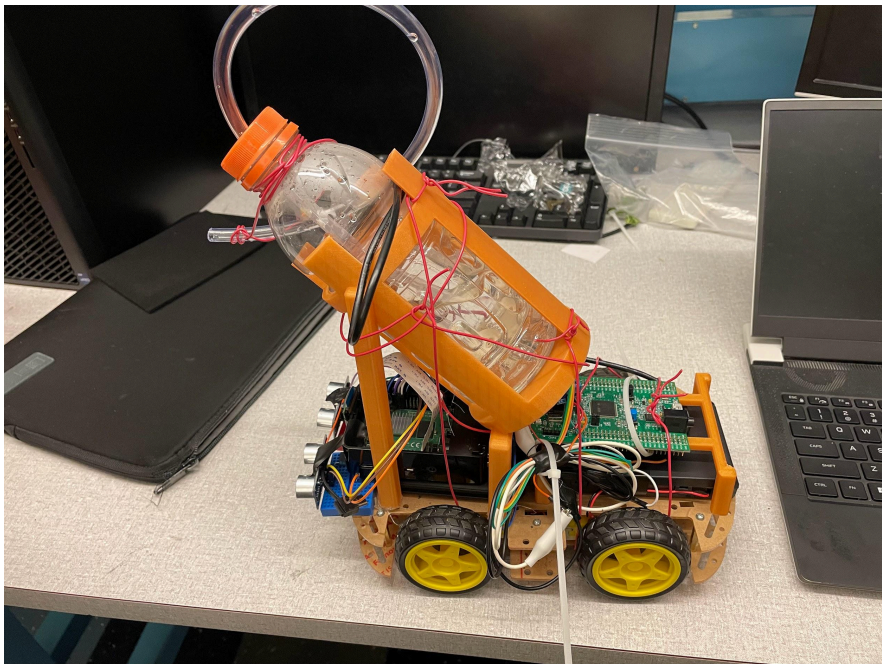

Figure 10. Finished design.

Above is a picture of the finished design. After successful assembling the robot, we performed a series of indoor and outdoor trials to test the functionality and accuracy of the robot.

The robot was tested in a controlled indoor environment to assess its navigation capabilities and obstacle detection accuracy. Utilizing the Raspberry Pi and camera module, the indoor trials focused on the robot's ability to correctly identify plants using the pre-trained AI model. Different types of plants, varying in size, color, and shape were placed in the robot's path to evaluate the detection algorithm's accuracy and sensitivity. The efficiency and precision of the water dispensing mechanism were tested by positioning the robot in the front of various plants to observe if the water was adequately released and targeted correctly. Adjustments were made to the water flow rate and the positioning of the spout to ensure optimal watering.

After initial indoor testing, we moved to outdoor experiments to further test the robot. The robot was subjected to various outdoor terrains, including grass, soil, and uneven surfaces, to test its mobility and the effectiveness of its wheels in outdoor conditions. The endurance of the robot was tested in prolonged outdoor sessions to ensure it could operate effectively over extended periods. This involved evaluating battery life, water storage capacity, and overall stability of the robot's systems over time.

Experimental Outcomes

Currently, we have achieved the main goal of this project which was to design an autonomous robot to auto detect and water plants. The robot can move around randomly without user input while avoiding obstacles utilizing the distance sensors installed in the front of the robot. When the robot approaches within 25 cm of an object, the robot will stop and run the object detection program in the Raspberry Pi using the Raspberry Pi camera to determine if the target object is a plant. If the object is a plant, the robot will activate the water pump to water the plant and if the object is not a plant, the robot will reverse, turn, and then continue its prior movement.
The distance sensors constantly triggered the robot to halt when approaching an obstacle. Although the intended stop distance was 25cm, variations were observed, attributed to the robot's speed and sensor response times. This deviation suggests a need for further cabiration or software adjustment to stabilize stopping distance irrespective of the robot's velocity.

The AI-driven plant recognition system achieves an 87% success rate in correctly identifying standard plants. However, instances of false positives were noted, with the system occasionally green-colored non-plants objects as plants, such as grass in the background. This issue indicates room for improvement in the AI's discriminatory capabilities, requiring further training or adjustment to recognition algorithms.

Durability of the robot also meets the requirements. The water pump's functionality aligned with the expectations, dispensing 50mL of water per activation. With a full tank, the robot successfully watered between 35 to 40 plants. The robot demonstrated a battery life spanning between 3-5 days without the need for recharging.

The robot exhibited optimal performance on smooth surfaces. This finding highlights the limitations in terrain versatility, pointing towards potential enhancements in wheel design.

Impact and Consequences

The Smart Plant Caretaker Robot project introduces an innovative approach to gardening and plant care. This autonomous robot aims to simplify the process of watering plants, offering significant benefits and potential impacts on the gardening industry, as well as on individual homeowners and garden enthusiasts.

For many, gardening is a time consuming activity, requiring regular attention and care. The Smart Plant Caretaker automates the watering process, saving homeowners and gardeners countless hours of manual labor. By ensuring plants are watered accurately and consistently, the robot can help improve plant health and growth. The AI-driven detection system ensures that each plant receives the right amount of water, reducing the risks of over or under-watering. Additionally, this project showcases the potential for advanced technology to be integrated into traditional gardening practices. It could pave the way for further innovations in smart gardening tools, contributing to more efficient and sustainable gardening practices.

Conclusions and Recommendations

The original proposal was for a robot that could move autonomously and water plants. So far our project has fulfilled these basic goals, but the execution is different from our original plan. One planned feature was having the robot movement integrated with plant detection, so the robot could move toward plants. Instead, the current design has random robot movement that stops when any object is detected. Additional features such as weed killer / fertilizer are unlikely due to weight and space constraints. Some of the potential improvements for the robot include sophisticated pathfinding algorithms to enable the robot to remember the positions of plants, upgrading the AI model to differentiate between types of plant species and tailor its watering strategy, and designing a base station where the robot could recharge and refill water.

A major issue with the product was that sometimes the code ran without error, and other random times errors appeared that stopped the code from running. These errors were a greater concern than the actual behavior of the code. Moreover, logistical issues with downloading and running software, hardware, robot assembly, and others played a larger role than expected. Fortunately these errors were resolved, but future efforts on the project could be more efficient with small scale tests that discover and resolve similar issues earlier.

This capstone project has been an invaluable journey, providing us with comprehensive experience in project management over an extended period. We learned to effectively divide tasks, set progressive goals, and navigate through the various phases of the project lifecycle. From a technical standpoint, we gained significant insights and skills in several areas: we became familiar with the Raspberry Pi ecosystem, delved into the application of AI and computer vision technologies, explored the intricacies of GPIO pins for controlling basic robotic functions and motors, and embraced the challenges of assembling a product in a more open-ended manner. This experience has not only enhanced our technical capabilities but also strengthened our problem-solving and teamwork skills, laying a solid foundation for future projects.

References, Acknowledgements, and Intellectual Property

We would like to thank Professor Rania Hussein, the TAs Francisco Luquin Monroy and Benjamin Chang, and anonymous student reviewers for their suggestions. If commercialized, the original overall combination of the watering system, vehicle movement, and AI detection model (each based on some open source tutorial) would likely be copyrighted.

[1] Adafruit Industries. “Make a Pi Trash Classifier with Machine Learning and Lobe.” Digikey. https://www.digikey.com/en/maker/projects/make-a-pi-trash-classifier-with-machine-learning-and-lobe/9029e053c23845e98bea70c5cabfb555 (accessed Jan. 31, 2024).

[2] COCO. “Common Objects in Context.” https://cocodataset.org/#download.

[3] Core Electronics. Object Identification & Animal Recognition With Raspberry Pi + OpenCV + Python. (Aug 23, 2021). Accessed: Jan 29, 2024. [Online video]. Available: https://www.youtube.com/watch?v=iOTWZI4RHA8

[4] Matthew Dunn. Flashing the STM32F103 with a Raspberry Pi / Getting started with the STM32F103. (Mar 20, 2017). Accessed: Feb 7, 2024. [Online video]. Available: https://www.youtube.com/watch?v=tCcxFMU1OFE

[5] MicroPeta by Nizar Mohideen. STM32CubeIDE L298N Motor with STM32F103C8T6. (July 11, 2021). Accessed: Jan 31, 2024. [Online video]. Available: https://www.youtube.com/watch?v=0rbKmkNEdAU

[6] PlantNet. “A Pl@ntNet dataset for machine learning researchers.” PlantNet. https://plantnet.org/en/2021/03/30/a-plntnet-dataset-for-machine-learning-researchers/ (accessed Feb. 5, 2024).

[7] sentdex. Raspberry pi with Python for Robotics - Video Streaming Pi RC Car. (Jun 5, 2014). Accessed: Jan 19, 2024. [Online video]. Available: https://www.youtube.com/watch?v=ZEmKlxxITI8

[8] T. Mariotti. “Gardening Statistics (2024).” RubyHome. https://www.rubyhome.com/blog/gardening-stats/ (accessed Feb. 4, 2024).

[9] Tim. “Object and Animal Recognition With Raspberry Pi and OpenCV.” Core Electronics. https://core-electronics.com.au/guides/object-identify-raspberry-pi/ (accessed Jan. 29, 2024).