NILE

Detail Design Document

Tia McKenzie
Nicodemus Phaklides
Lachlan McManus
Jacob Woodruff
Emmanuel Jefferson
Alexander Hoppe

ME 407
Robotics Preliminary Design
Embry-Riddle Aeronautical University

Table of Contents

1.0 Introduction....5
1.1 Requirements....5
1.2 Conceptual Design....6
1.3 Design Specifications....8
1.3.1 Soil Water Saturation....8
1.3.2 Soil Temperature....8
1.3.3 Application Efficiency....8
1.3.4 Fertilizer Application....8
1.3.5 Weed Extermination....9
1.3.6 Plant Location Determination....9
1.3.7 Ingress Protection....9
1.3.8 Communication....9
2.0 System Design....10
2.1 Mechanical Design....10
2.1.1 Frame Subsystem....11
2.1.2 Trolley Subsystem....12
2.1.3 End Effector Subsystem....13
2.1.4 Fluid Management....13
2.1.5 Cable Management....14
2.1.6 Structural Analysis....15
2.1.6.1 System at Rest....16
2.1.6.2 End Effector Driven into Ground....17
2.1.7 System Kinematics....20
2.1.7.1 Forward Kinematics....21
2.1.7.2 Velocity Kinematics....23
2.1.7.3 Inverse Kinematics....25
2.1.8 Equations of Motion....28
2.2 Electrical Design....29
2.2.1 Power Distribution....30
2.2.1.1 High-Voltage Elimination Circuit....30
2.2.1.3 Power Budget....31
2.2.1.4 Power Box....32
2.2.2 Data Distribution....32

2.2.2.1 Hardware PCB....33
2.2.2.2 Rotational Enclosure Electronics....34
2.2.2.3 Trolley Electronics....34
2.2.2.4 End Effector Electronics....35
2.3 Software Design....35
2.3.1 System Website....37
2.3.1.1 Website Back-End....37
2.3.1.2 Website Front-End....40
2.3.2 Data Pipelines....41
2.3.2.1 MySQL Connector....41
2.3.2.2 ROS Communication....42
2.3.3 Control Algorithms....44
2.3.4 Image Processing....46
2.4 System Integration....50
3.0 Testing....51
3.1 Soil Water Saturation....51
3.1.1 Soil Water Saturation Results....52
3.2 Soil Temperature....53
3.2.1 Soil Temperature Results....54
3.3 Application Efficiency....55
3.3.1 Application Efficiency Results....56
3.4 Fertilizer Application....56
3.4.1 Fertilizer Application Results....57
3.5 Weed Extermination....57
3.5.1 Weed Extermination Results....57
3.6 Plant Location Determination....58
3.6.1 Plant Location Determination Results....59
3.7 Ingress Protection....61
3.7.1 Ingress Protection Results....61
3.8 Communication....62
3.8.1 Communication Results....62
4.0 Project Management....63
4.1 Team Organization....63
4.2 Project Planning....63
4.3 Budget....65

5.0 Conclusion 65
6.0 Citations 66
7.0 Appendix A – Additional Resources........ 67
7.1 CAD and PCB Files........ 67
7.2 Main Software Repository........ 67
7.2 Website Repository........ 67

1.0 Introduction

Conventional agricultural methods overcompensate plant and soil needs through wasteful watering practices and excessive application of pesticides and fertilizers, leading to substantial environmental damage [1]. This damage takes on various forms, including pollution via runoff, soil depletion, and the extinction of local pollinators [2]. The Novel Irrigation and Land Use Efficiency (NILE) system aims to optimize current agricultural practices with a unique robotic approach to precisely monitor the health of various crops.

In recent years there has been an explosion in the number of robotic systems designed to use varying levels of autonomy to approach challenges facing agricultural industries. Concepts like the Greenhouse Partner Robot System [3] seek to augment farmer abilities in the greenhouse through a cooperative approach to automation. Similarly, Agrobot [4] and Vinebot [5] support farmers by providing an autonomous platform upon which sensors can gather information. Both approaches utilize unique methods of locomotion and control but are limited in scope and ultimately require the intervention of farmers or other systems to directly affect crops.

Some more autonomous systems, such as the Farmbot, tend to plants based on internal determination. This approach can monitor, and water plants based on user input and general watering guidelines based on species [6]. Stereoscopic imaging allows the system to find adequate locations to measure moisture content, for which the user can define feedback systems within its web app. Another approach, as taken by AgBotic Inc. And their robotic gantry allows for significantly increased control over the watering and fertilizing of a field when compared to traditional systems [7]. Unfortunately, it requires significant infrastructure investment in greenhouses and supporting equipment which so far prohibits widescale industrial adoption.

Numerous systems gather and relay sensor data to farmers for interpretation, but comparatively few systems can independently infer plant health from the data. One promising approach utilized machine learning paired with computer vision to scan images of plant leaves and identify individual plants, as well as distinguish between healthy and diseased crops based solely on physical appearance [8]. Combining this approach with more conventional sensor data, we plan to develop a fully autonomous plant health assessment system that does not need farmer intervention. The goal of this project is to incorporate concepts from these disparate approaches in a fully autonomous system capable of caring for crops from sow to harvest.

1.1 Requirements

Requirements state what a system must do to solve a problem. NILE defined the following eight requirements for a system designed to nurture crops between the planting harvesting phases.

1) The system shall be capable of measuring the water saturation for the soil at any point within the growing zone.
2) The system shall be capable of measuring the temperature of the growing zone.
3) The system shall be capable of irrigating crops within the growing zone with a field application efficiency greater than 90%.

4) The system shall be capable of supplying the nutrients needed to maintain plant health within the growing zone.
5) The system shall be capable of exterminating weeds from any point in the growing zone.
6) The system shall be capable of determining the location of plant foliage, plant stems, and weeds within the growing zone.
7) The system shall be capable of being certified to an ingress protection level of IP55.
8) The system shall be capable of communicating plant health, soil health, and the detection of weeds to the end user.

For the purposes of this document, the growing zone is defined as the three-dimensional soiled volume from the surface to a depth of 5 centimeters that can be reached by the end effector. In addition, field application efficiency is an industry-defined term describing the ratio of water input to an irrigation system vs the water that reaches the growing zone. It is expressed mathematically as follows.

$$Field\ Application\ Efficiency = \frac{\sum Water\ \ Delivered\ to\ the\ Growing\ Zone}{\sum Water\ \ Input\ to\ the\ System} \times 100\%$$

Given the above requirements, the next phase of the design process was to develop a conceptual design that would be capable of meeting them.

1.2 Conceptual Design

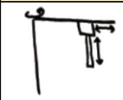
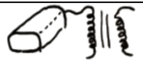
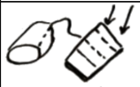
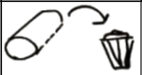
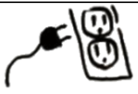
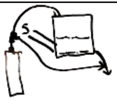
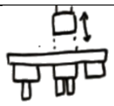
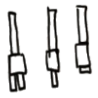
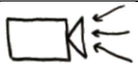
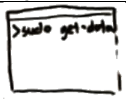
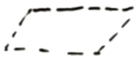
Given our problem statement and using the requirements as a guide, we considered a wide range of possible implementations. To determine the optimal approach, we began with a morphological chart to compile the available options, then used decision matrices to down select to the optimal configuration.

To begin, we listed the primary systems the system would require as follows.

1) *Mode of locomotion:* how the system maneuvers within the growing zone.
2) *Power system:* the method used to provide power to electrical components.
3) *End effector operation:* how the sensors, nozzles, etc. are mounted to the end effector.
4) *Watering end effector:* the tool by which water is delivered to the growing zone.
5) *Parasite removal end effector:* the tool used to eliminate parasites.
6) *Detection hardware:* the physical devices used to determine the location of plants.
7) *Detection software:* how the system will determine plant health data.
8) *Communications:* how the system will communicate its status to the end user.
9) *Growing zone:* the environment in which the system operates, and plants grow.

Given the key functions, we brainstormed possible solutions and generated the following morphological chart.

Table 1. Morphological Chart

Functions	**Concepts**						
	1	2	3	4	5	6	7
Mode of Locomotion	*A-Frame*	 *Polar*	*Cartesian*	*Train*	*Bug*	*Strand-beast*	*Mobile-Arm*
Power System	 *Bat. /Inductive*	 *Bat. /Solar*	*Bat. /Grid*	 *Bat. /Swap*	*Rail*	 *Grid Power*	
Fluid Handling	*Combined Fluid Line*	*Dual Fluid Line*	 *Fertilizer Mixing*				
End Effector Operation	 *Hot-Swap*	*Rotary*	 *Multiple*	*Omnibus*	*Hand*		
Watering End Effector	*Nozzle*	*Needle*	*Water Capsule*				
Parasite Removal End Effector	*Drill*	*Prong*	*Laser*	*Fire*	*Wedge*	*Taser*	
Detection Hardware	 *Camera*	*False NDVI*	*Stereoscopic*	*Infrared*	*LIDAR*		
Detection Software	*CV Segmentation*	*CV and ML*	*Spidering*	*User Input*			
Communication	 *Terminal*	*Web app*	*Screen*				
Growing Zone	*Raised Bed*	*Pots*	 *Field*				

Using these options as starting points, we ranked the viability of each solution based on metrics of cost, ease of use, scalability, and more in decision matrices. This process guided us to a design that combined existing center pivot irrigation technology with robotics in the form of a cylindrical robot that rotates about a central core and translates radially and vertically.

This system would utilize an end effector that combined temperature sensing via a thermistor, moisture determination via a capacitive sensor, parasite elimination via high energy electrical arcing, and the delivery of water and fertilizer via a single nozzle. Furthermore, the locating of plants and weeds would be done with a computer vision algorithm while communication with the user existed through a web application.

1.3 Design Specifications

Specifications define how a design must perform to ensure it meets the requirements. Given this configuration of the system discussed in the previous section, we created the following specifications.

1.3.1 Soil Water Saturation

The system shall be capable of measuring the water saturation for the soil at any point within the growing zone. With this requirement in mind, we defined the following specifications.

1) The moisture sensor shall have a minimum resolution of ±5% field saturation.
2) The moisture sensor shall have an operating range of 0% – 100% field saturation.
3) The moisture sensor shall be capable of taking two independent readings within 10 cm of each plant’s foliage exterior.

1.3.2 Soil Temperature

The system shall be capable of measuring the temperature of the growing zone. With this requirement in mind, we define the following specifications.

1) The temperature sensor shall have a minimum resolution of ±1°C.
2) The temperature sensor shall have an operating range including, but not limited to, -10°C to 50°C.
3) The temperature sensor shall be capable of taking two independent readings within 10 cm of each plant’s foliage exterior.

1.3.3 Application Efficiency

The system shall be capable of irrigating crops within the growing zone with a field application efficiency greater than 90%.

With this requirement in mind, we define the following specifications.

1) The liquid delivery system shall be capable of supplying water to the growing zone with a minimum rate of 40 mL/s.
2) The fluid distribution system shall be capable of measuring the flow rate of liquids input to the liquid delivery system and the flow rate liquids delivered to the end effector with a resolution of ±2 mL/s.

1.3.4 Fertilizer Application

The system shall be capable of supplying the nutrients needed to maintain plant health within the growing zone. With this requirement in mind, we define the following specifications.

1) The liquid delivery system shall be capable of supplying a user defined liquid volume of fertilizer to the growing zone with a minimum rate of 10 mL/s.

1.3.5 Weed Extermination

The system must be capable of exterminating weeds from any point in the growing zone. With this requirement in mind, NILE defines the following specifications.

1) The system shall be capable of positioning the end effector with a resolution of $\Delta e_{max} = \pm 0.5cm$ in the r, and z directions and a $\Delta\theta_{\max} = \pm 0.145°$ on the θ axis with respect to the growing zone origin.
2) The weed elimination system shall be capable of generating at least a 15kV pulse arc with a discharge energy of at least 135 mJ.

1.3.6 Plant Location Determination

The system shall be capable of determining the location of plant foliage, plant stems, and weeds within the growing zone. With this requirement in mind, we define the following specifications.

1) The detection algorithm shall be capable of determining the position of exterior plant and weed foliage with a resolution of +2 cm with respect to the stereoscopic camera in cartesian coordinates.
2) The detection algorithm shall correctly differentiate between plants versus weeds with 99% identification accuracy.
3) The detection algorithm shall inspect the growing zone a minimum of twice per 24-hour period as the soil requires.
4) The detection algorithm shall complete all assessments prior to the start of the next inspection period.

1.3.7 Ingress Protection

The system shall be capable of being certified to an ingress protection level of IP55. With this requirement in mind, we define the following specifications.

1) The system shall maintain satisfactory operation in the presence of dust after exposure to sunlight over the course of 5 day/night cycles.
2) The system shall maintain satisfactory operation when subjected to pressurized water delivered from a nozzle with a minimum diameter of 6.3mm after exposure to sunlight over the course of 5 day/night cycles.

1.3.8 Communication

The system shall be capable of communicating soil health, and the detection of weeds to the end user. With this requirement in mind, we define the following specifications.

1) The system shall send/receive data to and from a server through a Wi-Fi connection, outgoing data will be displayed to the end-user via a web-application.

2) The web-application shall update the sensor readings within one minute of the system completing measurements.

2.0 System Design

The NILE robot was heavily inspired by the center-pivot irrigation systems used widely in farms across the western United States. In a traditional center pivot irrigation system long chains of sprinkler gantries rotate about a central well over the course of multiple days. While significantly more efficient than historical irrigation techniques, this model must overcompensate most plant needs to account for the lowest common denominator.

Our system is designed to replicate the efficient locomotion mechanism of the legacy design while improving plant care with precision end-effectors that travel along the gantry sections. Thus, instead of just wastefully watering, our system would be able to dynamically supply the exact amount of water and nutrients needed by the plants and eliminate weeds. An example of a center-pivot irrigation system and the NILE prototype design are shown below in figure 1.

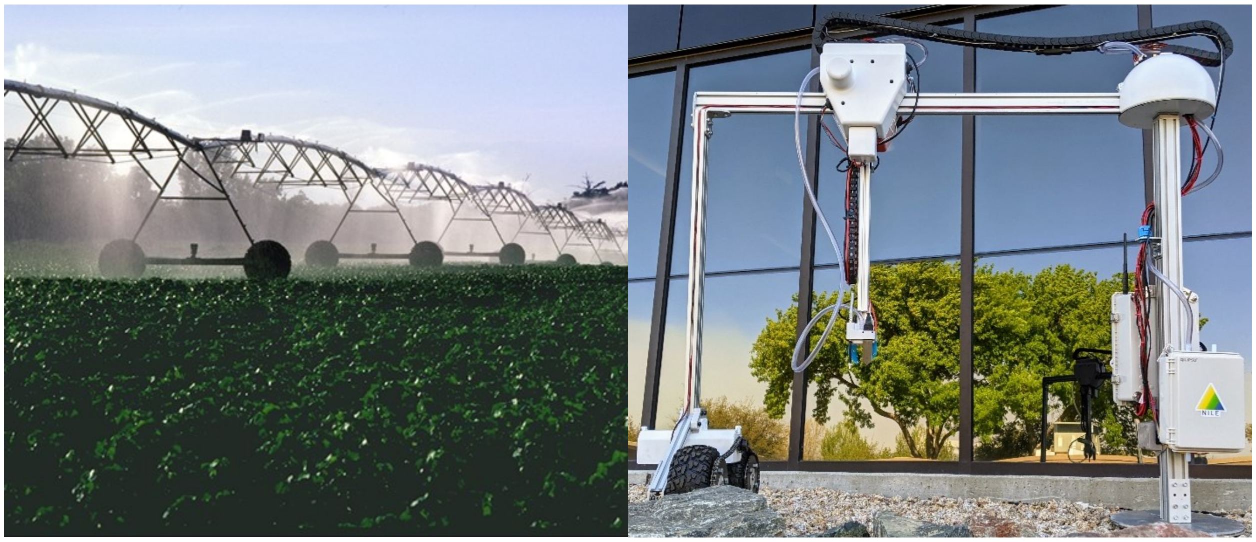


Figure 1. Center Pivot Irrigation System Alongside NILE Prototype

For the purposes of this capstone the team designed, built, and tested a scale model of the envisioned system with a growing zone 2 meters in diameter. While this system is limited in scope, more suitable for backyard gardening than industrial farms, it provides a proof of concept that we hope will inspire larger systems.

2.1 Mechanical Design

To enable rapid development and increase the ease of manufacturing, the mechanical design was split into three modular subsystems in addition to fluid and cable management. First, the frame subsystem is the backbone of the robot and includes the central tower along with the rotational gantry. Second, the trolley sub system translates radially on the gantry and moves the end effector up and down. Finally, the end effector subsystem interacts directly with plants and weeds in the growing zone.

2.1.1 Frame Subsystem

The frame subsystem can be divided into two main parts, the central tower and the rotating gantry. The central tower is mechanically simple but serves as the foundation of the whole robot. Constructed of 50x50 mm 80/20, t-slot extrusion the tower is topped with a stainless-steel central axle and grounded with a heavy steel plate. In addition, it supports the waterproof electronics and hydraulics enclosures. The electronics enclosure contains the power supply and system determination computer while the hydraulics enclosure contains two solenoid valves and a flowmeter used to measure and control the fluid system. All this can be seen below in figure 2.

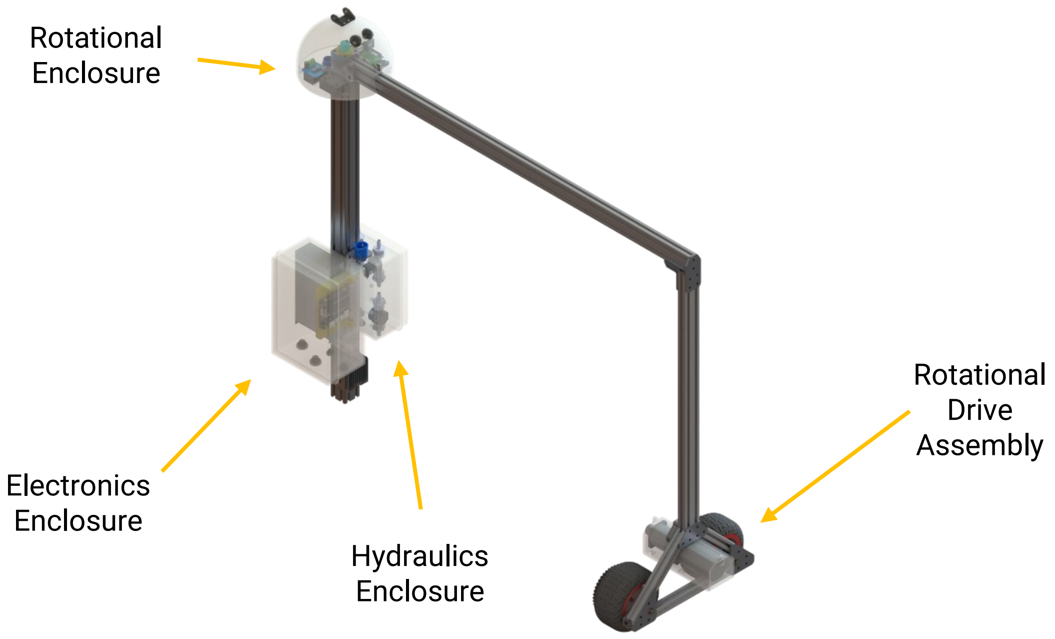


Figure 2. Frame Subassembly

The gantry consists of a 50x25 mm 80/20, t-slot beam supported by the central tower on one end and the rotational drive frame on the other. On the central tower side, bronze bushings rest on the tower and rotate about its axle while being attached to the gantry via custom machined 6061 aluminum mounts. Those mounts also support the absolute encoder used to determine the angle of rotation and the rotational enclosure. The rotational enclosure is a custom printed case that houses the hardware microcontroller and its various connections in addition to the rotational drive motor controller.

While its controller is located above the central tower, the rotational drive motor is located at the other end of the gantry. The motor is mounted to the gantry by means of a custom 6061 aluminum base plate and 50x25 mm 80/20, t-slot extrusion. To rotate the gantry the motor drives two large wheels via a 4:1 roller chain configuration and a 100:1 gearbox. This configuration significantly decreases the power required to achieve stable control of the rotation by increasing the moment arm. Furthermore, it is highly accurate as the absolute encoder can detect any wheel slip and compensate accordingly.

2.1.2 Trolley Subsystem

The trolley subsystem supports the radial and vertical translation of the system while also housing electronics for the end effector. The trolley consists of two 6061 aluminum mounting plates with high friction wheels supported between them. The radial translation is accomplished by driving one of the wheels with a geared DC motor and letting the others rotate freely. One of the freely rotating wheels measures the position of the trolley via an encoder on the shaft and the other is for stabilization. This setup allows the driven wheel to slip without disrupting the positional accuracy. Please see figure 3 below.

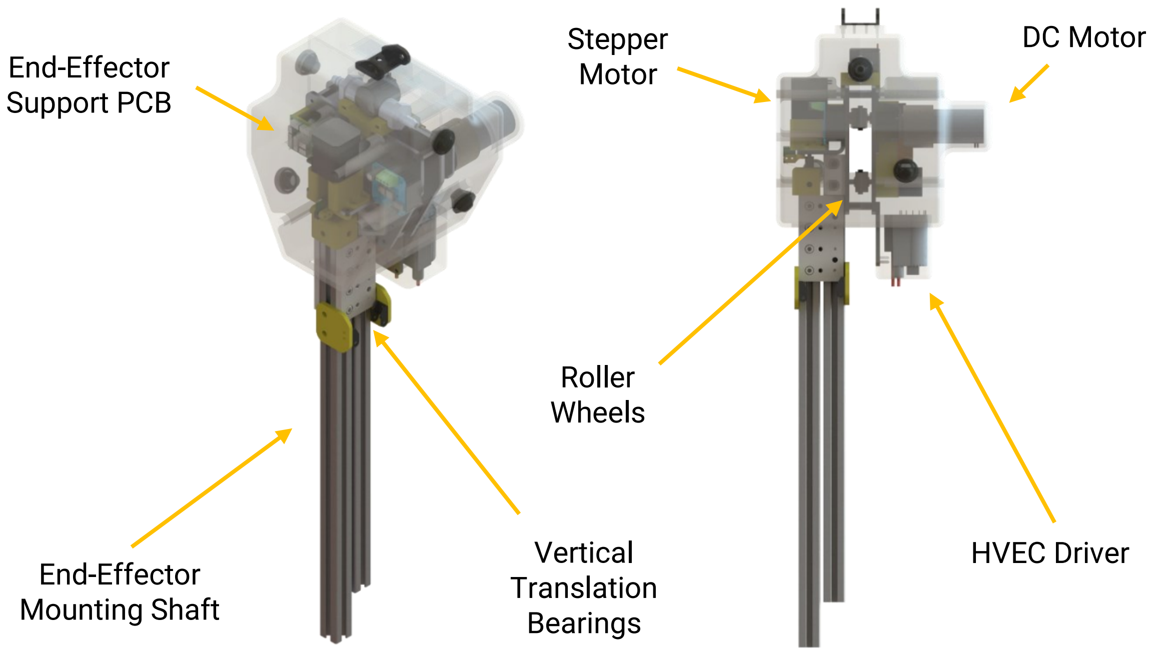


Figure 3. Trolley Assembly

Vertical translation is accomplished via an ultra-precision lead screw driven by a high-torque stepper motor. The end effector mounting shaft rides on bearings slotted into the aluminum extrusion tracks and ensures that it is always secure.

2.1.3 End Effector Subsystem

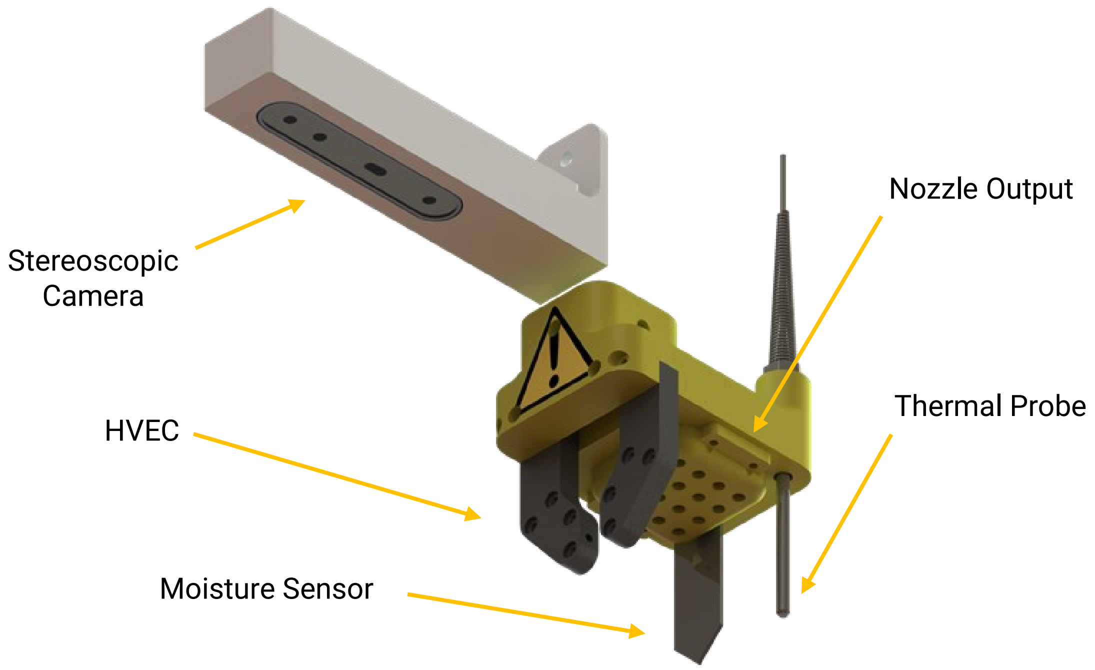


Figure 4.1 End Effector Subassembly

The end effector, as seen in figure 4, is small compared to the other subsystems but is perhaps the most critical as it allows the NILE system to interact with the environment. From top to bottom, the first component is the waterproof camera housing, and the Intel Realsense stereoscopic camera is used for the computer vision and machine learning systems. Then we reach the primary component, the 3D printed omnibus mount which supports the rest of the sensors and the High Voltage Weed Elimination Circuit (HVEC) which will be discussed further in section 2.2.1.1. In addition, the water and fertilizer application nozzle are built directly into the end effector to supply the plants with all their needs.

2.1.4 Fluid Management

The NILE hydraulic system supports the watering and fertilizing of all plants in the growing zone. To decrease the complexity of the system, it is designed to receive two pressurized inputs of water and liquid fertilizer. When deployed this is as simple as a hose connected to mains water with an inline fertilizer douser.

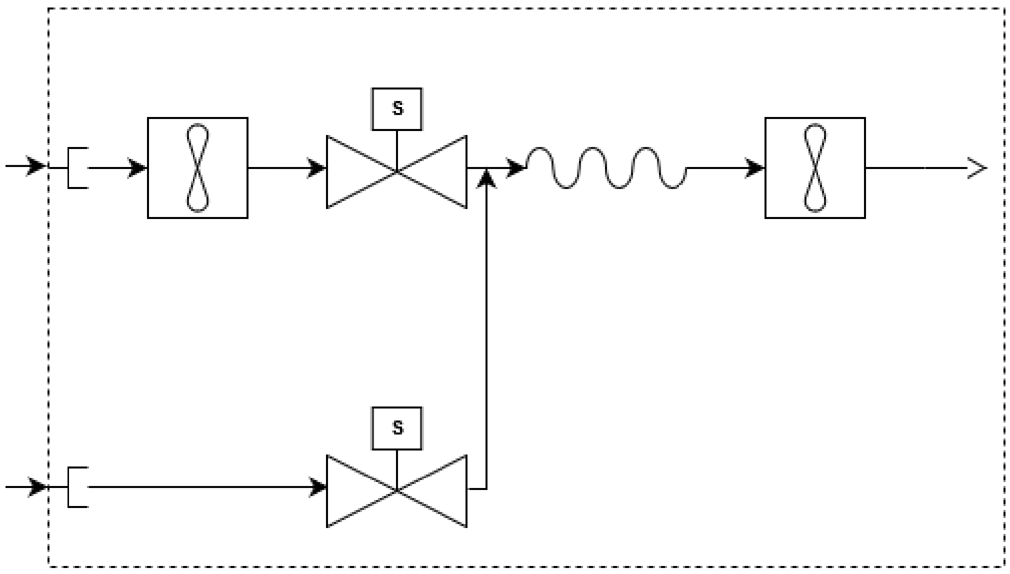


Figure 5. Hydraulic System Diagram

Upon input, the water flow rate will be immediately sampled with a turbine style flow meter before entering a solenoid valve, as seen in the upper left corner of figure 5. The fertilizer input passes through a different solenoid valve, as seen in the lower left corner of figure 5, before rejoining the nozzle line through a Y-fitting. The two solenoids control access to the nozzle based on whether water or liquid fertilizer is desired. Finally, the last flow meter provides feedback used to identify leaks and calculate the field application efficiency before dispensing the fluid on the plants via the nozzle on the end effector.

2.1.5 Cable Management

Cable management is a vital part of any robotic system and is especially important for the NILE system due to the addition of fluid lines. The first step of the process was the development of enclosures capable of meeting the ingress protection specification. For the central tower this was as simple as selecting off the shelf enclosures. However, the complex geometry of the rotational enclosure and trolley necessitated custom 3d printed enclosures that were thoroughly painted to ensure water tightness. Both the off the shelf and custom enclosures were fitted with rubber gromets to allow for wires and tube to pass through without compromising ingress protection, as seen in figure 6.

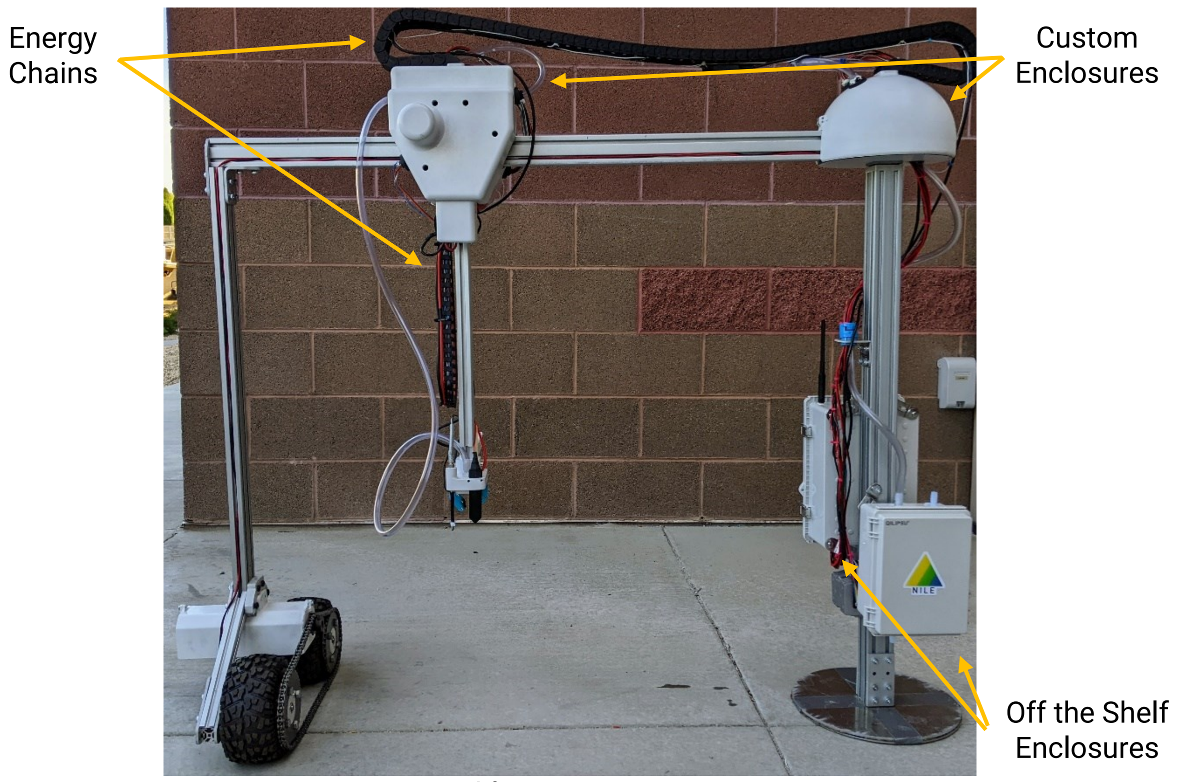


Figure 6. Cable Management Overview

The next step was the management of wires and tubes between the various watertight enclosures. This was relatively simple between the rotational enclosure and trolley in addition to the trolley and end effector because the motion was linear. For these interconnects, energy chains were chosen based on the size of components and the distance of motion.

The connection from the electronics and hydraulics enclosures to the rotation enclosure was significantly more difficult due to the rotational motion. While slip rings were initially considered because they would have allowed infinite rotation, they were eventually rejected due to the cost and complexity. Thus, it was decided the cables would simply wrap around the central tower and rotation would be limited to +/- 180°. However, an industrial scale solution would likely benefit significantly from a slip ring implementation by placing more components inside the rotational enclosure.

2.1.6 Structural Analysis

To ensure the proper operation of the system in nominal and edge case conditions NILE performed structural analysis on all the critical components during the design phase. For this analysis two configurations were identified as having the most critical loading: the system at rest with the trolley in the middle of the beam and the end effector driven into the ground during sensing operations. Loads were calculated based on the material properties of the individual components from the Solidworks Material Library and the data sheets of the actuators. All Finite Element Analysis [FEA] simulations were conducted using the Solidworks Simulation software.

2.1.6.1 System at Rest

Since the robot will be at rest for most of its life cycle, that state is critical to analyze and ensure stability and reliability. The four areas of concern identified for this state include the gantry beam, the various axles, and the stepper motor mount.

To analyze the main gantry beam, NILE performed a beam Finite Element Analysis [FEA]. A beam analysis was ideal for this application as it allows us to simulate the complex but uniform cross section of the aluminum extrusion over a long distance. For the simulation, the central shaft mount was assumed to be a fixed support, the rotational drive assembly as a roller joint, and the weight of the trolley / end effector as a remote load applied to the points where the wheels touch the extrusion.

By applying a uniform gravitational load and running the simulation, the following stress and displacement plots were created. These can be seen below in figures 7 and 8.

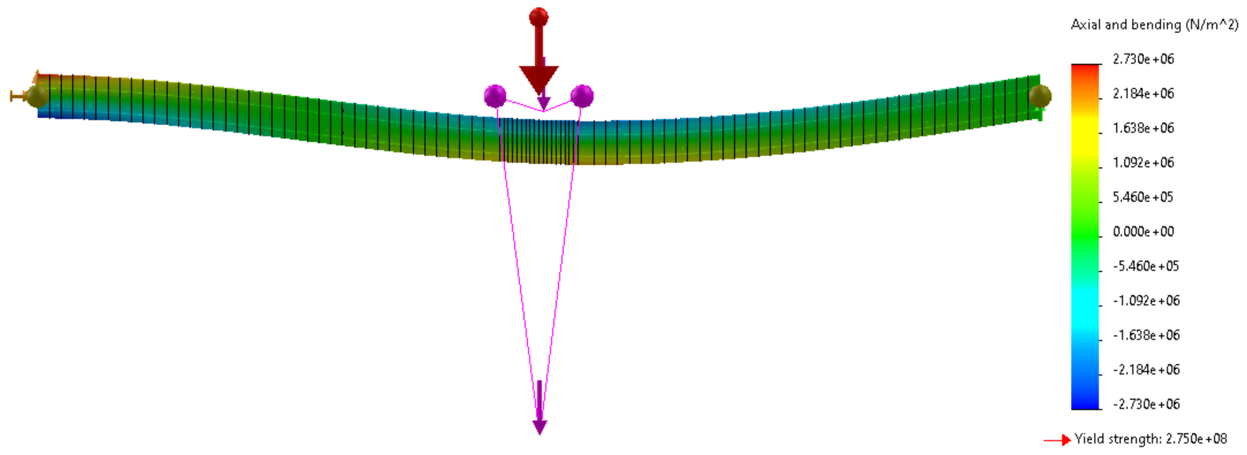


Figure 7. Gantry Beam Stress

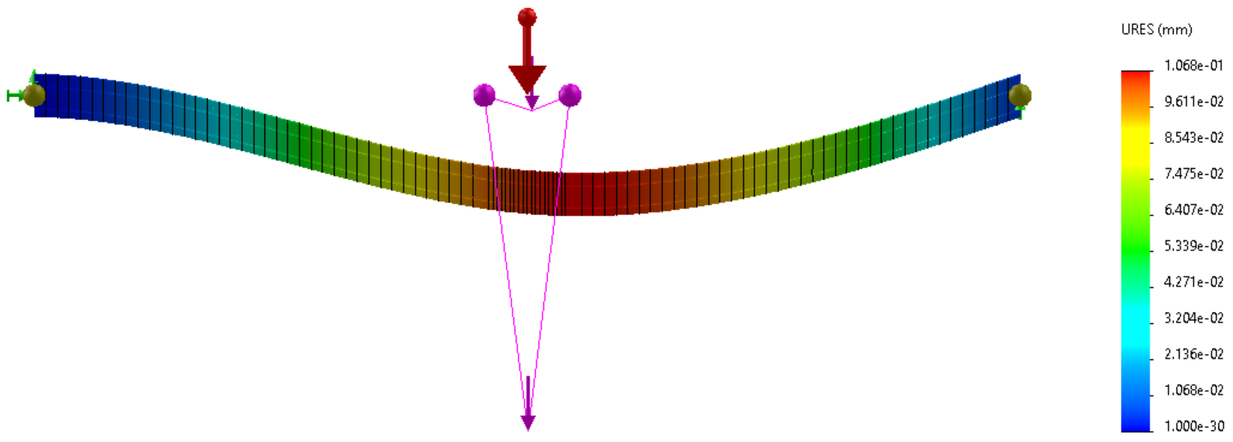


Figure 8. Gantry Beam Displacement

As seen in figure 8, the maximum displacement is on the order of tenths of millimeters which is well within our positional accuracy specifications. Furthermore, the maximum combined axial and bending moment stresses are significantly below the yield strength of the material.

Next, a finite element analysis was performed on the stepper motor mount to ensure it will be able to withstand the weight of the third link. For this analysis, the motor mount and the sheet metal baseplate share a no-interference bond and pre-torqued bolts fixing them together. In addition, the baseplate is fixed with foundation bolts and a virtual wall. Furthermore, the 3D printed mount was meshed as tetrahedral elements and the baseplate was modeled as shell element body.

To simulate the load, a remote distributed mass corresponding to link three was applied to the surface of the motor mount and gravity applied to the whole model. 3D printed parts are difficult to model as the many layers and infill patterns are not easily defined in FEA. For this analysis NILE assumed the motor mount was homogeneous ABS as the loading is in compression and will be printed with near 100% infill.

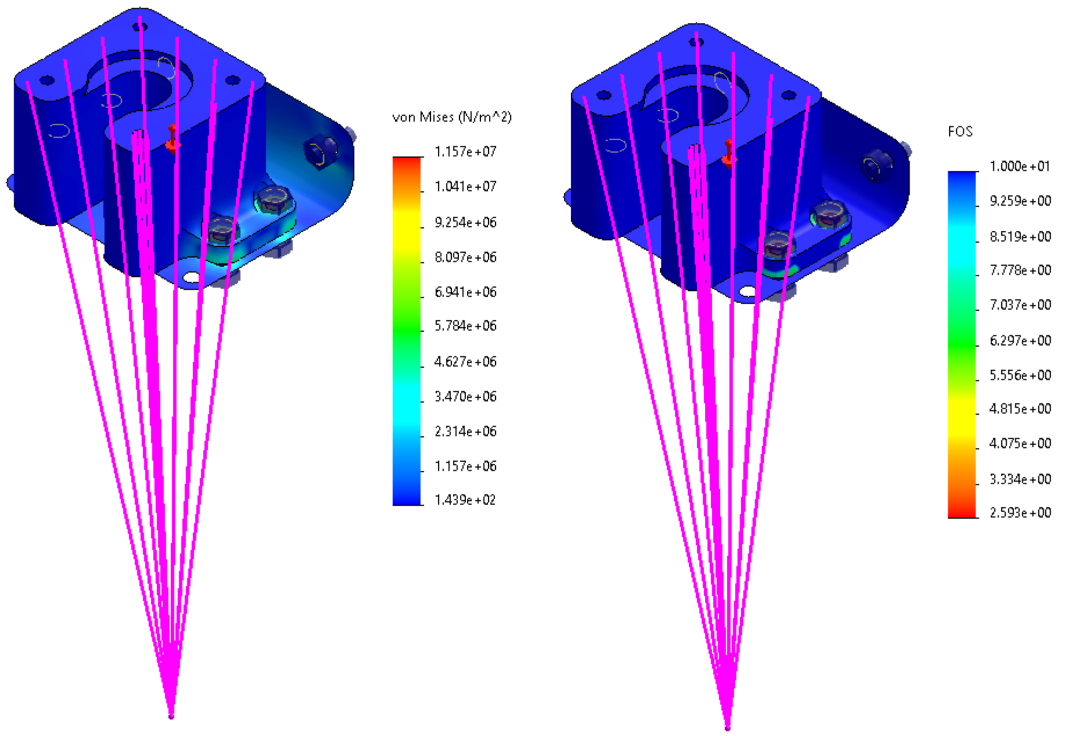


Figure 9. Motor Mount Gravitational Stress and FOS

As can be seen in figure 9 above, the stresses in the parts are minimal with the lowest safety factors being due to the pre-torque as opposed to the load itself.

2.1.6.2 End Effector Driven into Ground

The second load condition NILE analyzed was the end effector being pressed into the ground by the stepper motor/lead screw. The first step of the process was to determine if the maximum force exerted by the stepper motor was greater than the weight of the trolley assembly.

The maximum pushing force that the stepper motor can supply through the lead screw can be derived by solving the power screw mechanics equation as presented in Shigleys [9 page 402].

$$F_{push} = \frac{2 \times T_{lower}}{D_{pitch} \times \left(\frac{\pi \times \mu \times D_{pitch} - L_{\text{Lead}}}{\pi \times D_{pitch} + \mu \times L_{\text{Lead}}}\right)} = 41.35 \text{ N}$$

Then, the force of gravity applied on the gantry by link two is given by the following.

$$F_{grav} = W_{link2} \times 9.81 \frac{m}{S^2} = 52.01 \text{ N}$$

As seen from the equations above $F_{push} < F_{grav}$. Therefore, the sensing operation does not have a significant impact on the loading of the rest of the system. Given that, the next step was to analyze the components it does effect: end effector, translational support beams, lead screw, and motor mount.

To analyze the end effector, NILE did not incorporate the sensors or HVEC in the analysis as those non-3D printed parts would experience relatively low insertion forces. Instead, NILE focused on the 3D printed end effector mounting part and the nozzle.

As the lowest, large flat surface on the end effector the load found above is applied evenly across the nozzle surface. The nozzle is attached to the mounting part by a no-interference bond and pre-torqued bolts and the mounting part is fixed using four foundation screws and five virtual walls representing the aluminum extrusion support beam.

As mentioned previously, 3D printed parts are difficult to model and, as the parts will be experiencing significant cantilever loading, the results of this simulation must be critically examined.

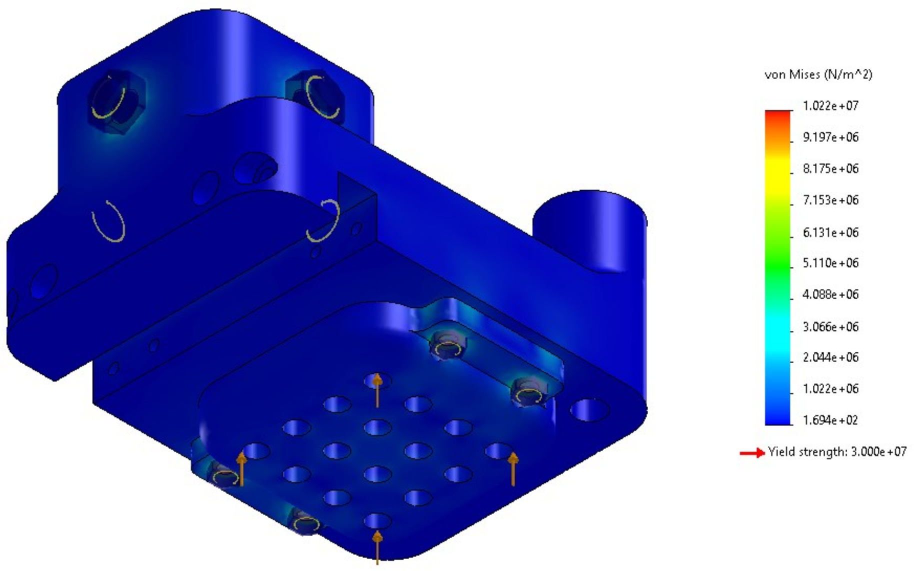


Figure 10. End-Effector Stress

Fortunately for this analysis, the loading is low enough that the components do not experience significant stresses and most of the geometry has a factor of safety greater than 5. Furthermore, the areas with the highest stresses appear to be stress concentrations linked to the screw pre-torques as seen in figure 10 above.

With the end effector confirmed to be strong enough, the next components to model are the two support beams that allow for the vertical translation. The response of these components is somewhat difficult to model because of their large size and the complex interactions of the lead screw and linear bearings.

To simulate these interactions in FEA, both beams were modeled with beam elements. The static beam has one fixed end, representing its mounting point, with the other end rigidly attached to the translating beam, representing the bearing connection. The translating beam will have an addition fixture in the vertical direction representing the lead screw nut.

Modeling the upward pushing force as a rigid, remote load and running the simulation, the following stress and displacement plots were created.

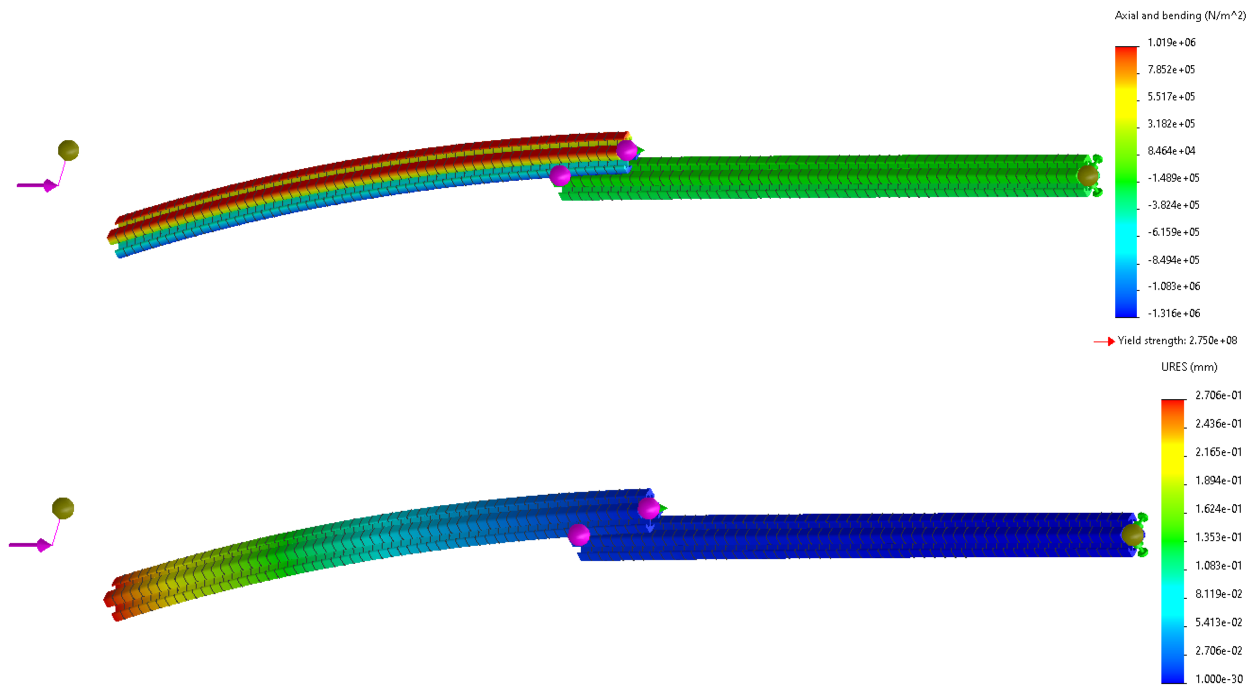


Figure 11. Vertical Translational Beam Stress and Displacements

As can be seen in figure 11 above, the maximum combined axial and bending moment stresses are significantly below the strength of the aluminum and the maximum displacement is roughly 0.1mm, well within our accuracy specifications.

To ensure the lead screw could survive the compressive thrust load one could start from a basic machine design analysis to calculate the maximum stress at the roots of the screw threads. However, the plastic lead nut is more likely component to fail and the stainless-steel lead screw.

The lead nut is rated by the manufacturer to withstand up to 111 N of dynamic thrust load which is far more than the 41N the stepper can apply.

The final assembly to model is the motor mount and base plate discussed previously. For this analysis the components have been meshed and fixed in the same way. Here, the upwards force calculated above is applied to the screw holes that secure the motor to the mount as the force will be transmitted through the lead screws to the motor itself.

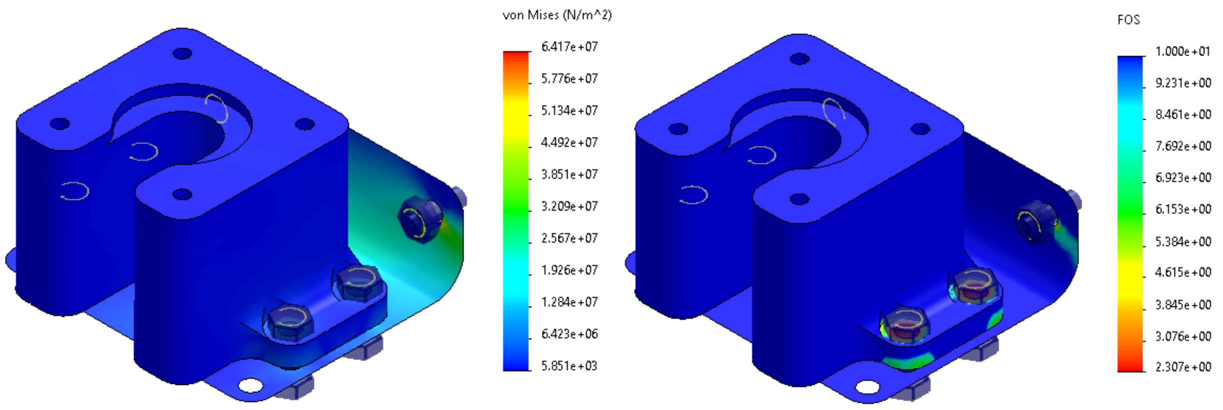


Figure 12. Motor Mount Stress and FOS

As can be seen in figure 12 above the sheet metal base plate is experiencing significantly more loading than the simple gravitational load case but the stresses are still well withing the factors of safety for both materials. Thus, with the design verified the next step was to determine the system kinematics.

2.1.7 System Kinematics

Given the detailed design of the robotic system, the next step was to derive its kinematic properties so that it may be controlled. This section will show the derivation of the forward, velocity, and inverse kinematics of the robot and the control systems that take advantage of the derivation will be discussed further in section 2.3.3.

Throughout the rest of this section and the document, NILE will refer to links one, two, and three on the robot. The links are separate from the major assemblies defined above as they are defined for mechanical and kinematic analysis. From left to right, figure 13 below defines Links one, two, and three from left to right.

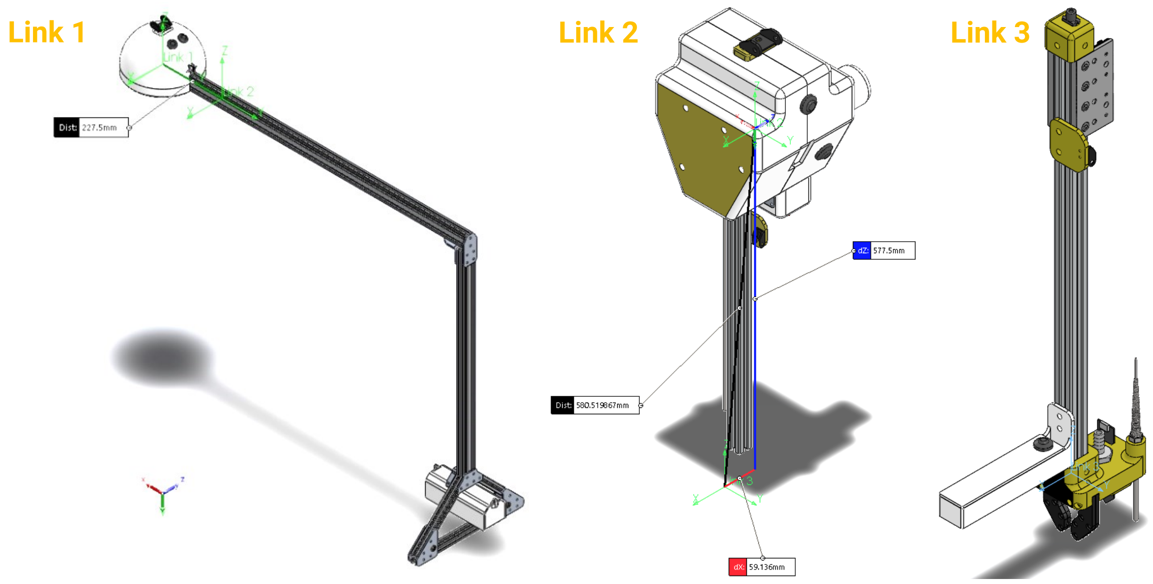


Figure 13. Robot Chassis Links

2.1.7.1 Forward Kinematics

Forward kinematics concern the mapping of joint space coordinates into task space coordinates. Given the linkages defined above in figure 13, we define four generalized, joint space, coordinates as follows.

$$\gamma = [\theta_1 \quad d_2 \quad d_3 \quad \theta_4]^T$$

In the above equation θ_1 is the rotation of link one about the central tower, d_2 is the translation of link two along the gantry, d_3 is the vertical translation of link 3, and θ_4 is the rotation of the wheels. Because θ_1 and θ_4 are not independent we must define the following kinematic relationship to solve. This assumes no slip or deformation in the rotational drive wheels, which is not entirely realistic but will be sufficient for this analysis.

$$\theta_4 = \theta_1 \frac{r_L}{r_W} \frac{N_2}{N_1}$$

Given the fully defined generalized coordinates, the next step was to find the position vectors of each linkage's frame relative to that of the next linkage based on the mechanical model.

$$\underline{{}^{I}_{I}R_{1}} = \begin{bmatrix} 0 \\ 0 \\ 1425e-3 \end{bmatrix} (m)$$

$$\underline{{}^{1}_{1}R_{WheelA}} = \begin{bmatrix} 143.305826e-3 \\ 1095.9432e-3 \\ -944.561165e-3 \end{bmatrix} (m)$$

$$\underline{{}^{1}_{1}R_{WheelB}} = \begin{bmatrix} -143.305826e-3 \\ 1095.9432e-3 \\ -944.561165e-3 \end{bmatrix} (m)$$

$$\underline{{}_1^1R_2} = \begin{bmatrix} 0 \\ 227.5e-3+d_2 \\ 0 \end{bmatrix} (m)$$

$$\underline{{}_2^2R_3} = \begin{bmatrix} 0 \\ 0 \\ -577.5e-3-d_3 \end{bmatrix} (m)$$

We also define the position vectors from frame three to the various end effector components.

$$\underline{{}_3^3R_{Nozzle}} = \begin{bmatrix} -40e-3 \\ 0 \\ -30e-3 \end{bmatrix} (m)$$

$$\underline{{}_3^3R_{HVEC}} = \begin{bmatrix} 0 \\ 0 \\ -65 \end{bmatrix} (m)$$

$$\underline{{}_3^3R_{Temp}} = \begin{bmatrix} -67.5e-3 \\ 25e-3 \\ -80e-3 \end{bmatrix} (m)$$

$$\underline{{}_3^3R_{Camera}} = \begin{bmatrix} 110.35e-3 \\ 0 \\ 25e-3 \end{bmatrix} (m)$$

Next, we define the transformation matrices that describe the orientation of each frame relative to the previous frame. Note, I refers to the identity matrix.

$$T_1^I = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = I_3$$

$$T_3^2 = I_3$$

$$T_{Wheel}^1 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 \\ 0 & 1 & 0 \\ -\sin\theta_1 & 0 & \cos\theta_1 \end{bmatrix}$$

Then, the orientation of frames two and three can be expressed relative to the internal frame as follows.

$$T_2^I = T_1^I T_2^1$$

$$T_3^I = T_2^I T_3^2$$

Finally, using properties of matrix multiplication we can find the forward kinematic equations for each joint and end effector as follows.

$$\underline{{}^I_I R_2} = \underline{{}^I_I R_1} + T^I_1 \begin{bmatrix} 0 \\ 227.5e-3+d_2 \\ 0 \end{bmatrix} (m)$$

$$\underline{{}^I_I R_3} = \underline{{}^I_I R_2} + T^I_2 \begin{bmatrix} 0 \\ 0 \\ -577.5e-3-d_3 \end{bmatrix} (m)$$

$$\underline{{}^I_I R_{Moist}} = \underline{{}^I_I R_3} + T^I_3 \begin{bmatrix} 67.5e-3 \\ 17.5e-3 \\ -80e-3 \end{bmatrix} (m)$$

$$\underline{{}^I_I R_{HVEC}} = \underline{{}^I_I R_3} + T^I_3 \begin{bmatrix} 0 \\ 0 \\ -65e-3 \end{bmatrix} (m)$$

$$\underline{{}^I_I R_{Temp}} = \underline{{}^I_I R_3} + T^I_3 \begin{bmatrix} -67.5e-3 \\ 17.5e-3 \\ -80e-3 \end{bmatrix} (m)$$

$$\underline{{}^I_I R_{Nozzle}} = \underline{{}^I_I R_3} + T^I_3 \begin{bmatrix} -40e-3 \\ 0 \\ 30e-3 \end{bmatrix} (m)$$

$$\underline{{}^I_I R_{Camera}} = \underline{{}^I_I R_3} + T^I_3 \begin{bmatrix} 110.35e-3 \\ 0 \\ 25e-3 \end{bmatrix} (m)$$

$$\underline{{}^I_I R_{WheelA}} = {}^1_1 R_{WheelA} + T^I_1 {}^1_1 R_{WheelA}$$

$$\underline{{}^I_I R_{WheelB}} = {}^1_1 R_{WheelB} + T^I_1 {}^1_1 R_{WheelB}$$

Given the above equations we can map joint space coordinates to task space coordinates and will also be able to calculate the velocity kinematics in the next section.

2.1.7.2 Velocity Kinematics

Velocity kinematics describe how joint velocities of the affect the linear and angular velocity of the end-effector. The NILE robot has a cylindrical configuration thus the velocity kinematics are relatively straight forward. For example, the angular velocity and the rate of change of the orientation of the first frame, second frame, and third frame are all equal because there is only one rotational joint. Furthermore, the only difference between the second and third frame is a component appearing in the z position.

Due to the similarities between the first, second, and third frames we will focus on frame three in our derivation below because it also describes the velocities of the end effector. First, we find the angular velocity of frame three relative to the inertial frame measured in frame three to be as follows.

$$ {}_{3}^{3}\omega_I = \begin{bmatrix} 0 \\ 0 \\ \frac{RW\dot{\theta}_4 \cos\left(\frac{RW\theta_4}{RL}\right)^2}{RL} + \frac{RW\dot{\theta}_4 \sin\left(\frac{RW\theta_4}{RL}\right)^2}{RL} \end{bmatrix} $$

Below we find the vector that describes the velocity of frame three relative to the inertial frame

$$ {}_{I}^{I}\dot{r}_3 = \begin{bmatrix} -\dot{d}_2 \sin\left(\frac{RW\theta_4}{RL}\right) - \frac{RW\dot{\theta}_4 \cos\left(\frac{RW\theta_4}{RL}\right)(d_2 + 0.23)}{RL} \\ -\dot{d}_2 \cos\left(\frac{RW\theta_4}{RL}\right) - \frac{RW\dot{\theta}_4 \sin\left(\frac{RW\theta_4}{RL}\right)(d_2 + 0.23)}{RL} \\ \dot{d}_3 \end{bmatrix} $$

Then, we have the matrix that describes the rate of change of the orientation of frame three below.

$$ {}_{w}^{w}\omega_I = \begin{bmatrix} -\frac{RW\dot{\theta}_4 \sin(\theta_4)}{RL} \\ \dot{\theta}_4 \\ -\frac{RW\dot{\theta}_4 \cos(\theta_4)}{RL} \end{bmatrix} $$

Moving away from link three, we also calculated the angular velocity of the wheel frames relative to the inertial frame measured in the wheel frame below.

$$ {}_{w}^{w}\omega_I = \begin{bmatrix} -\frac{RW\dot{\theta}_4 \sin(\theta_4)}{RL} \\ \dot{\theta}_4 \\ -\frac{RW\dot{\theta}_4 \cos(\theta_4)}{RL} \end{bmatrix} $$

Then, we calculate the vector that describes the velocity of each wheel relative to the inertial frame.

$$ {}_{I}^{I}\dot{r}_{WA} = \begin{bmatrix} -\frac{RW\dot{\theta}_4\,(1.0959)\cos\left(\frac{RW\theta_4}{RL}\right)}{RL} + \frac{(0.1433)RW \sin\left(\frac{RW\theta_4}{RL}\right)}{RL} \\ \frac{RW\dot{\theta}_4\,(0.1433)\cos\left(\frac{RW\theta_4}{RL}\right)}{RL} - \frac{(1.0959)RW \sin\left(\frac{RW\theta_4}{RL}\right)}{RL} \\ 0 \end{bmatrix} $$

$$
{}_{I}^{I}\dot{r}_{WB} = \begin{bmatrix} -\frac{RW\dot{\theta}_4\,(1.0959)\cos\left(\frac{RW\theta_4}{RL}\right)}{RL} - \frac{(0.1433)RW\,\sin\left(\frac{RW\theta_4}{RL}\right)}{RL} \\ -\frac{RW\dot{\theta}_4\,(0.1433)\cos\left(\frac{RW\theta_4}{RL}\right)}{RL} + \frac{(1.0959)RW\,\sin\left(\frac{RW\theta_4}{RL}\right)}{RL} \\ 0 \end{bmatrix}
$$

Finally, we find the matrix that describes the rate of change of the orientation of the wheels below before moving on the inverse kinematics in the following section.

$$
\dot{T}_w^I =
$$

$$
\begin{bmatrix} -\frac{\dot{\theta}_4\cos\left(\frac{RW\theta_4}{RL}\right)}{RL}\sin(\theta_4) + \frac{RW\,\sin\left(\frac{RW\theta_4}{RL}\right)\cos(\theta_4)}{RL} & -\frac{RW\dot{\theta}_4\cos\left(\frac{RW\theta_4}{RL}\right)}{RL} & \frac{\dot{\theta}_4\cos\left(\frac{RW\theta_4}{RL}\right)}{RL}\cos(\theta_4) - \frac{RW\,\sin\left(\frac{RW\theta_4}{RL}\right)\sin(\theta_4)}{RL} \\ -\frac{\dot{\theta}_4\sin\left(\frac{RW\theta_4}{RL}\right)}{RL}\sin(\theta_4 - \frac{RW\,\cos\left(\frac{RW\theta_4}{RL}\right)\cos(\theta_4)}{RL} & -\frac{RW\dot{\theta}_4\sin\left(\frac{RW\theta_4}{RL}\right)}{RL} & 0 \\ -\dot{\theta}_4\cos(\theta_4) & 0 & -\dot{\theta}_4\sin(\theta_4 \end{bmatrix}
$$

2.1.7.3 Inverse Kinematics

Inverse kinematics are the opposite of forward kinematics and concern the mapping of task space coordinates into joint space coordinates. The NILE system has multiple sensors on the end effector so we will describe 5 inverse kinematic solutions below.

First, we calculated the inverse kinematics for the camera and visualized its position in the x-y plane in figure 14 below.

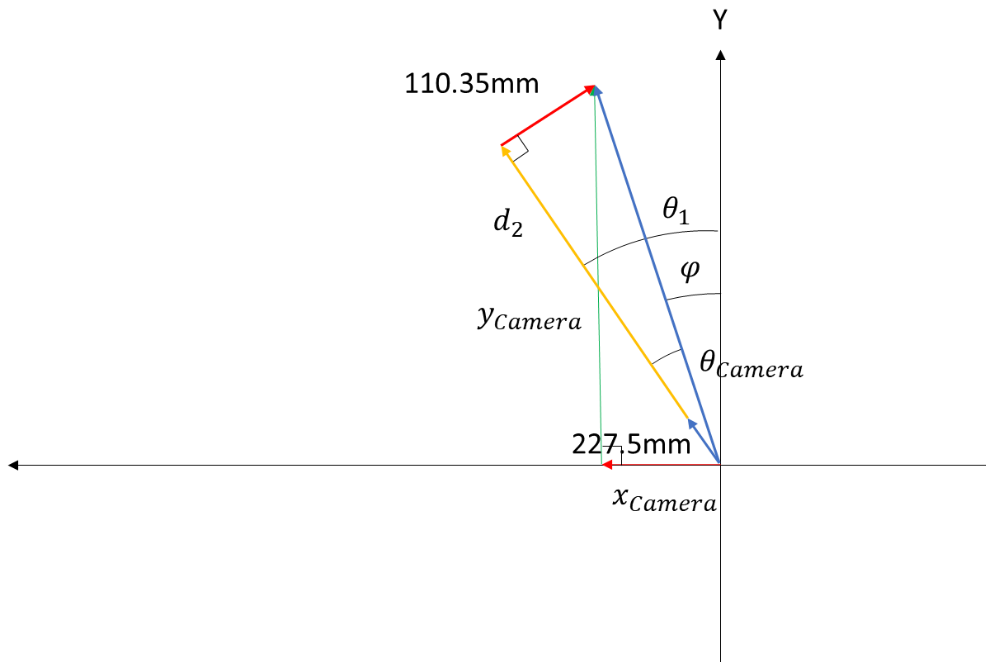


Figure 14. Camera Inverse Kinematics

$$\varphi = -atan2(x_{Camera}, y_{Camera})$$

$$\theta_{Camera} = arcsin\left(\frac{110.35e - 3m}{\sqrt{x^2{}_{Camera} + y^2{}_{Camera}}}\right)$$

$$\theta_1 = \varphi + \theta_{Camera}$$

$$d_2 = \left(\sqrt{x^2{}_{Camera} + y^2{}_{Camera}}\right)\cos\theta_{Camera} - 227.5e - 3m$$

$$d_3 = 872.5e - 3m - z_{Camera}$$

Next, we calculated the inverse kinematic equations for the nozzle which follows the same format as the camera.

$$\varphi = -atan2(x_{Nozzle}, y_{Nozzle})$$

$$\theta_{Nozzle} = arcsin\left(\frac{40e - 3m}{\sqrt{x^2{}_{Nozzle} + y^2{}_{Nozzle}}}\right)$$

$$\theta_1 = \varphi - \theta_{Nozzle}$$

$$d_2 = \left(\sqrt{x^2{}_{Nozzle} + y^2{}_{Nozzle}}\right)\cos\theta_{Nozzle} - 227.5e - 3m$$

$$d_3 = 817.3e - 3m - z_{Nozzle}$$

Then, we find the inverse kinematic equations for the moisture sensor. Again, with the same format.

$$\varphi = -atan2(x_{Moist}, y_{Moist})$$

$$\theta_{Moist} = arcsin\left(\frac{67.5e - 3m}{\sqrt{x^2{}_{Moist} + y^2{}_{Moist}}}\right)$$

$$\theta_1 = \varphi + \theta_{Temp}$$

$$d_2 = \left(\sqrt{x^2{}_{Moist} + y^2{}_{Moist}}\right)\cos\theta_{Moist} - 245e - 3m$$

$$d_3 = 767.5e - 3m - z_{Moist}$$

Then, the temperature sensor inverse kinematics with the same format.

$$\varphi = -atan2(x_{Temp}, y_{Temp})$$

$$\theta_{Temp} = arcsin\left(\frac{67.5e-3m}{\sqrt{x^2{}_{Temp} + y^2{}_{Temp}}}\right)$$

$$\theta_1 = \varphi - \theta_{Temp}$$

$$d_2 = \left(\sqrt{x^2{}_{Temp} + y^2{}_{Temp}}\right)\cos\theta_{Temp} - 252.5e-3m$$

$$d_3 = 767.5e-3m - z_{Temp}$$

Finally, we calculate the HVEC inverse kinematic equations which have a slightly simpler format than the previous examples because it is vertically in line with frame three. It is visualized in figure 15 and described by the equations below.

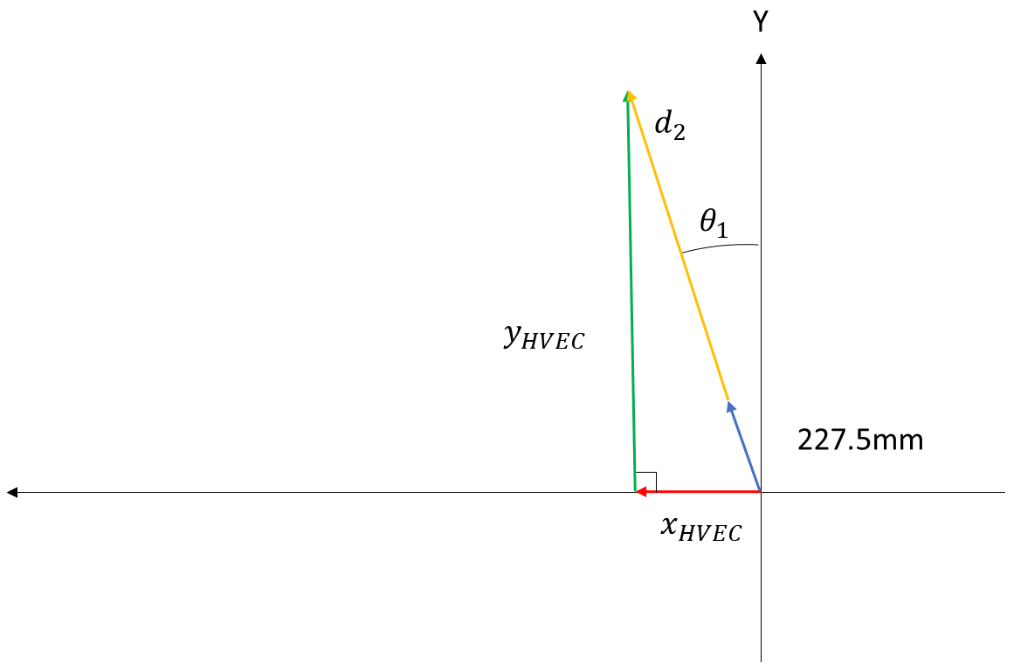


Figure 15. HVEC Inverse Kinematics

$$\theta_1 = -atan2(x_{HVEC}, y_{HVEC})$$

$$d_2 = \left(\sqrt{x^2{}_{HVEC} + y^2{}_{HVEC}}\right) - 227.5e-3m$$

$$d_3 = 782.5e-3m - z_{HVEC}$$

2.1.8 Equations of Motion

Given the system mass and inertial properties from the Solidworks model and the kinematic equations from the previous section we could calculate the robot's equations of motion. To do this we utilized the Newton-Euler method which describes the equations of motion where γ is the set of generalized coordinates.

$$H(\gamma)\ddot{\gamma} + d(\gamma)\dot{\gamma} + G(\gamma) = F_\gamma$$

The first variable, H(γ), is the system mass matrix and is listed below. Note that some large constants and expressions are shown as single variables in the matrix to increase readability.

$$H(\gamma) = \begin{bmatrix} m_1 + m_2 & 0 & \frac{rW(Gam2_x + Gam3_x)}{rL} \\ 0 & m_3 & 0 \\ \frac{rW(Gam2_x + Gam3_x}{rL} & 0 & H_{33} \end{bmatrix}$$

$$H_{33} =$$
$$f_1 + (sin(\theta_4)f_2 - cos(\theta_4)f_2)RL \cdot RW + mWx_6RW^2 + k_5f_3 + GamWy \cdot k_2 + k_1f_4 + k_3f_4 + k_4f_5$$

$$f_1 = (JWyy \cdot x_4 \cdot RL^2)$$
$$f_2 = (GamWx \cdot x_2 - JWxy \cdot x_7)$$
$$f_3 = (GamWz \cdot x_2 - JWxy \cdot x_7)$$
$$f_4 = (m_2 + m_3)$$
$$f_5 = (Gam2y + Gam3y + d_2m_2 + d_2m_3)$$
$$f_6 = \left(J1zz + J2zz + J3zz + {d_2}^2m_2 + {d_2}^2m_3\right)$$
$$f_7 = \left((JWzz - JWxx)(cos(\theta_4))^2 + JWxx + Gam2y \cdot d_2 + Gam3y \cdot d_2 - JWxz\,sin(2\theta_4)\right)$$

$$k_1 = RW^2x_1$$
$$k_2 = RW^2x_2$$
$$k_3 = RW^2x_3$$
$$k_4 = RW^2x_4$$
$$k_5 = RW^2x_5$$

$$x_1 = 28842854106992903572254488031920128000000$$
$$x_2 = 277854611059209293708063984844800000000000$$
$$x_3 = 63382530011411470074835160268800000000000$$
$$x_4 = 126765060022822940149670320537600000000000$$
$$x_5 = 3281307297482047674897531830951393361922$$
$$x_6 = 154859754020833869401932349434295410156 25$$
$$x_7 = 253530120045645880299340641075200000000000$$

The Coriolis and Centripetal vector, d(γ), is the second variable and has been listed below. Again, some large constants and expressions are shown as single variables.

$$d(\gamma) = \begin{bmatrix} \dfrac{r_W{}^2 \dot{\theta}_4{}^2 \left(100000\left(Gam2_y + Gam3_y\right) + 22753(m_2 + m_3) + 100000(d_2 m_2 + d_2 m_3)\right)}{100000 r_L{}^2} \\ 0 \\ d(3) \end{bmatrix}$$

$$d(3) = \frac{k_3(k_1 x_1 f_1 + d_2 k_1 x_1 f_2 + k_2 \cos(\theta_4) f_3 + k_2 \sin(\theta_4) f_4 + k_1 x_2 f_2 - JWxzk_3 x_1 \cos(2\theta_4) + k_3 x_4 f_4 \sin(2\theta_4))}{RL^2 x_4}$$

$$f_1 = Gam2y + Gam3y$$
$$f_2 = (m_2 + m_3)$$
$$f_3 = (GamWx \cdot x3 - JWxy \cdot x1)$$
$$f_4 = (GamWz \cdot x3 - JWyz \cdot x1)$$
$$f_5 = (JWxx - JWzz)$$

$$k_1 = RW\dot{d}_2$$
$$k_2 = RL\dot{\theta}_4$$
$$k_3 = RW\dot{\theta}_4$$

$$x_1 = 140737488355328000000$$
$$x_2 = 320220007254877798 4$$
$$x_3 = 140737488355328000000$$
$$x_4 = 70368744177664000000$$

Finally, the vector of generalized gravitational forces, $G(\gamma)$, is shown below.

$$G(\gamma) = \begin{bmatrix} 0 \\ -9.81m_3 \\ -20 \cdot GamWx\cos(\theta_4) - 20 \cdot GamWz\sin(\theta_4) \end{bmatrix}$$

Thus, with equations of motion fully defined, we conclude the mechanical design section of this document. In the following section we will discuss the electrical design that powers the robot.

2.2 Electrical Design

NILE's electrical system was designed to support all the power and data distribution needs of the robot. From the conceptual design process, we knew that the robot would have a fixed point from which we could access 120 VAC mains power. Furthermore, we would need to power three motors, the combination of two fluid lines, three soil sensors, the HVEC, and whatever other components would be needed to control the robot.

To ease integration and replicate an industrial scale implementation of the NILE system, the electrical design was split into three major subsystems. First, the central tower subsystem contains the main power supply in addition to the system determination computer and the hardware microcontroller. Then, the trolley subsystem contains the HVEC and stepper drivers in addition to the trolley PCB which amalgamates sensor data. Finally, the end effector sub effector contains the soil sensors, HVEC, and stereoscopic camera. In an industrial system the exact configuration would follow a similar pattern to take advantage of the economies of scale from multiple, independent trolleys. In the following sections we will discuss the power regulation and distribution through the system in addition to the data flow through the various sensors and computers.

2.2.1 Power Distribution

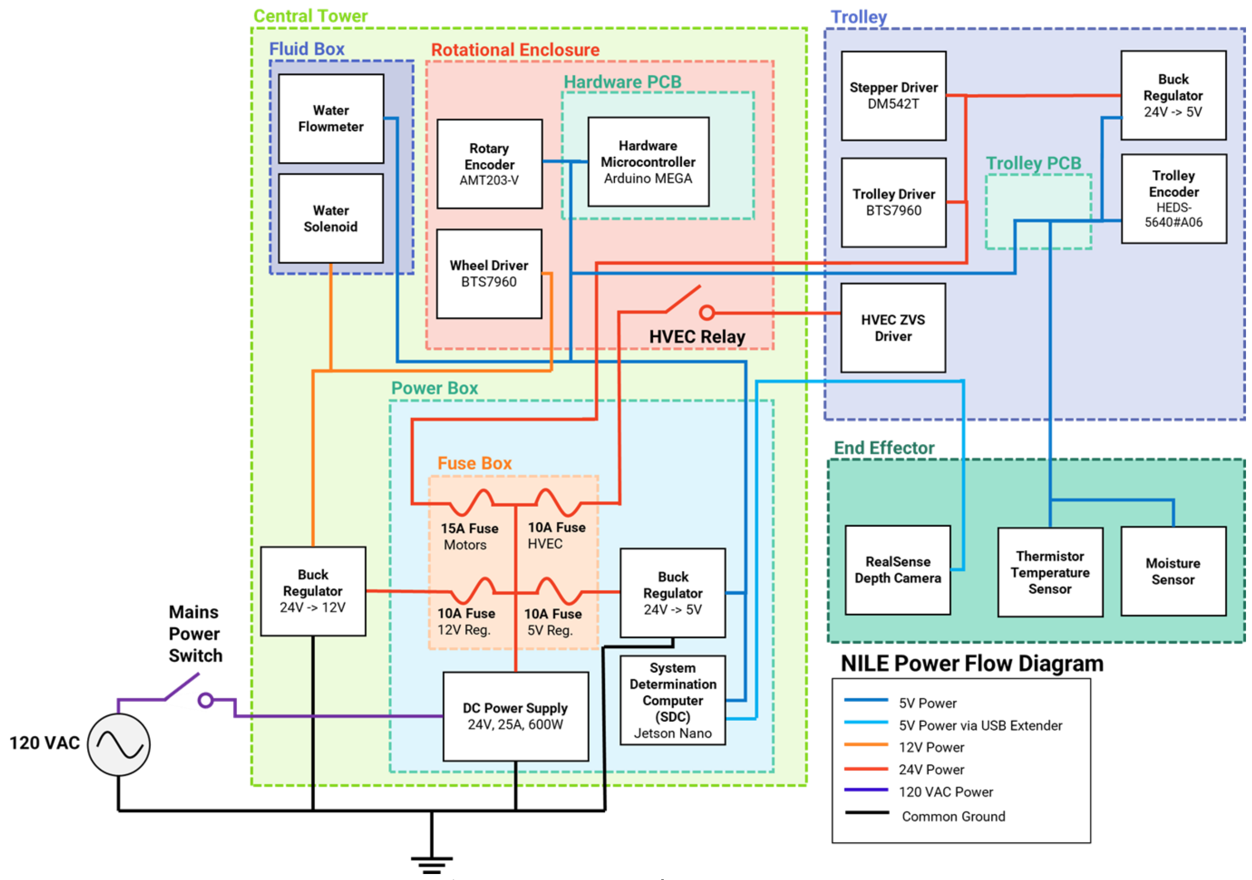


Figure 16. Power Flow Diagram

As depicted in figure 16, the robot primarily operates off a 24V, 25A, 600W DC power supply which is powered by mains. This voltage is split amongst four fuses to safely power the regulators, motors, and HVEC. This ensures that the power supply and components will be kept safe in the event of a short circuit. Three voltage levels are used through the system, 24V, 12V, and 5V. 24V is used to power the regulators, the trolley motor, vertical stepper motor, and HVEC. 12V is used to power the wheel motor and solenoids. Finally, 5V is used for the System Determination Computer (SDC), Hardware Microcontroller (HM), and most miscellaneous digital sensors. All circuits within the robot operate from common ground for safety and electrical simplicity.

2.2.1.1 High-Voltage Elimination Circuit

To eliminate weeds, the NILE Robot generates a high-voltage arc through a flyback transformer. Due to their low impedance at high frequencies, flyback transformers operated at resonance (around 10-100kHz) can produce an incredibly large voltage at their secondaries. The NILE

Robot uses a Zero-Voltage Switching (ZVS) driver to produce such a frequency, where two MOSFETs oscillate in an astable multivibrator configuration. This configuration, displayed in figure 17, drives a large current into the primary coil of the transformer, which is reflected as an incredibly large voltage on the secondary. With our prebuilt driver, we estimated voltages as high as 120kV on the secondary, as we could draw 4cm arcs under air's breakdown voltage of 30kV/cm.

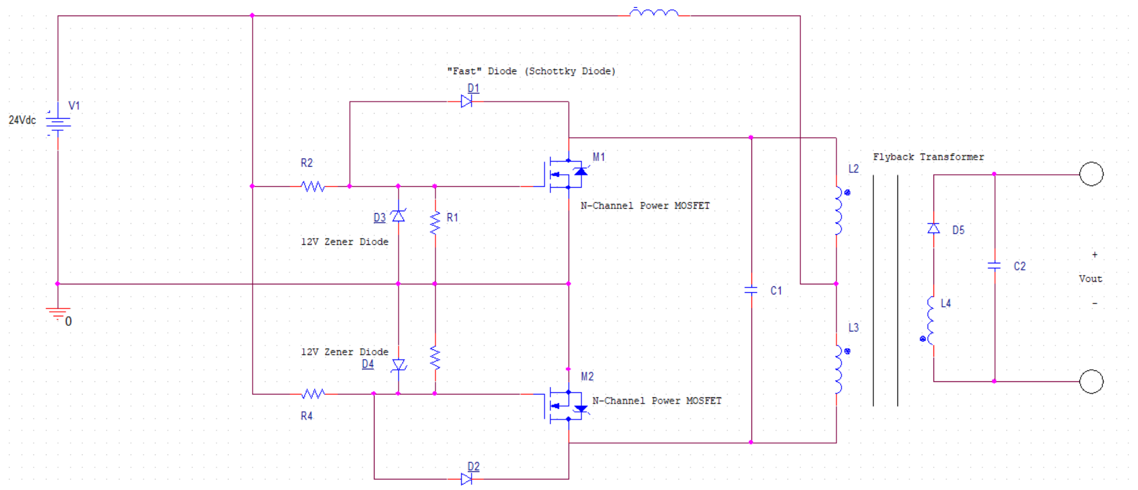


Figure 17. High-Voltage Elimination Circuit

2.2.1.2 Power Budget

The NILE Robot's power supply can provide sufficient power in all cases. In the impossible case that all subsystems are running, only 89% of the supply's maximum rating is used. This is an impossible case as the software only allows one motor/stepper to be running at any time. Additionally, the HVEC can only turn on if no motors are running. Therefore, this comprehensive case demonstrates that the system will always have sufficient power. Figure 18 lists power requirements of each electrical subsystem.

Power Budget | Everything Running			
Device	**Voltage (V)**	**Maximum Current (A)**	**Power (W)**
Main Power Supply	**24.00**	**25.00**	**600.00**
Vertical Stepper Motor/Driver	24.00	3.20	76.80
Trolley Motor/Driver	24.00	5.20	124.80
HVEC	24.00	5.00	120.00
24V-12V Buck Regulator	**12.00**	**30.00**	**360.00**
Wheel Motor/Driver	12.00	15.00	180.00
Water Solenoid	12.00	0.45	5.40
Total Consumption		15.45	185.40
% of Maximum Device Rating		52%	52%
Input Side Consumption @ 95% Efficiency	24.00	8.13	195.16
24V-5V Buck Regulator	**5.00**	**5.00**	**25.00**
NVIDIA Jetson Nano	5.00	2.00	10.00
Arduino Mega	5.00	0.20	1.00
Total Consumption		2.20	11.00
50% Variance from PCB Components		3.30	16.50
% of Maximum Device Rating		66%	66%
Input Side Consumption @ 90% Efficiency	24	0.76	18.33
Total Consumption		**17.30**	**535.09**
% of Maximum Power Supply Rating		**69%**	**89%**

Figure 18. Maximum Potential Power Consumption

2.2.1.3 Power Box

A waterproof enclosure mounted to the side of the central spire contains all critical power supply systems. Inside is the 24V power supply which routes through a fuse box. This splits off supply voltage in four separate lines, one for the HVEC, one for trolley drivers, one for the 12V buck regulator, and one for the 5V regulator. By using fuses, there is an additional safety against short circuits, as high current devices such as the HVEC and motors can get their own fuse. These external lines all route out of the power box through a waterproof insert. Mains power is brought to the power supply through an external waterproof IEC C14 plug. This plug contains a switch which allows for external power to be shut off quickly.

Additionally, the power box shown in figure 19 contains the Jetson Nano. It is powered by a 24V->5V buck regulator which resides next to it. The Jetson is connector to the Arduino and RealSense camera through two USB cables that also route out of a waterproof insert.

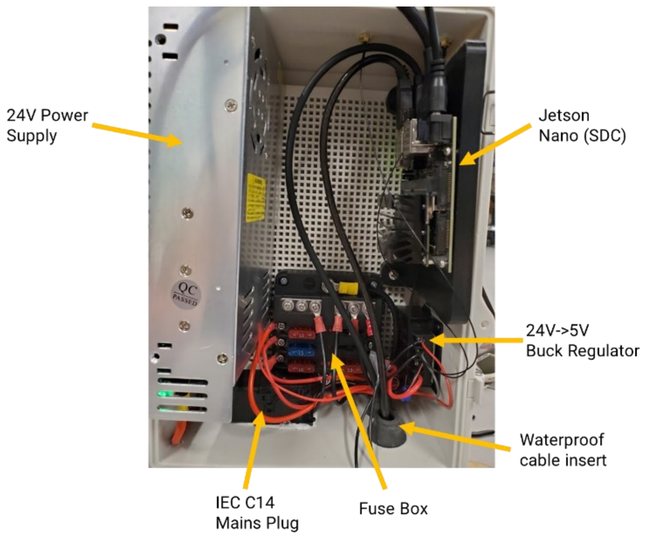


Figure 19. Power Box Layout

2.2.2 Data Distribution

Data interfacing throughout the robot is done primarily through two PCBs, the Hardware and Trolley PCBs, as shown in Figure 20 below. These boards interface through ribbon cable harnesses and pass sensor/driver information to and from the HM. Digital (I^2C, SPI) signal lines are brought into each board through JST-XH connectors. In principle, this was done to keep integration speedy but in hindsight, these connectors broke easily and were time-consuming to crimp. Regardless, this set up allowed for succinct data transfer, allowing the SDC to control the HM (and by proxy, the entire robot) via ROS.

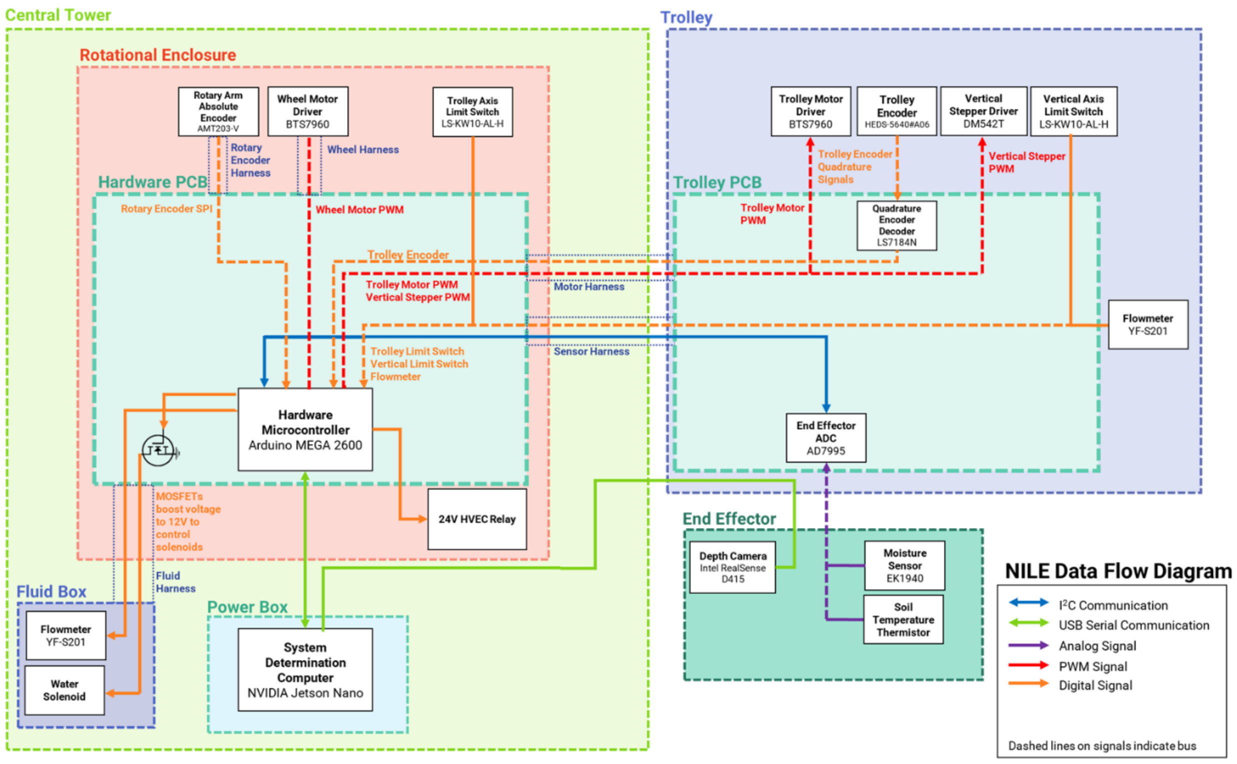


Figure 20. Data Flow Diagram

2.2.2.1 Hardware PCB

To interface with the full system, the Hardware Microcontroller uses a custom PCB shield. It contains several JST-XH connectors to connect to the system's many harnesses in an organized manner. Two low-side MOSFETs are present on the board to drive the fluid solenoids. A screw terminal powers the board with 5V power from the 24->5V buck regulator, reducing the power load on the USB-connected Jetson. Finally, there are three status LED lights forming the colors of the NILE logo as depicted below in figure 21. A yellow light indicates power, green light indicates I^{2}C SDA status, and the blue light is a user definable PWM.

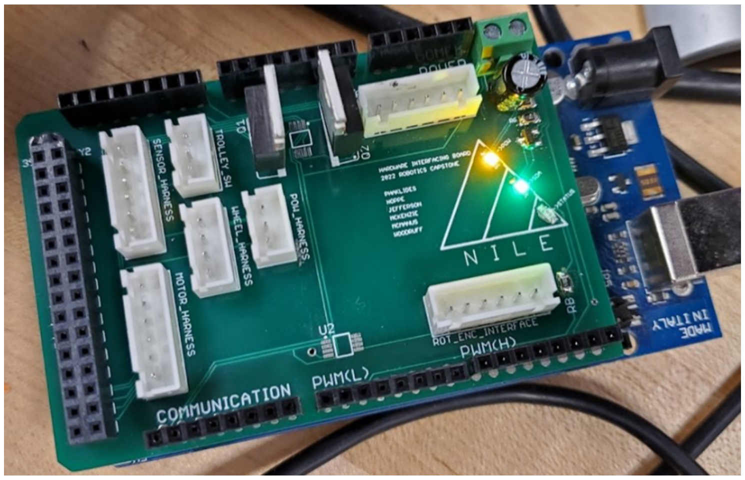


Figure 21. Hardware PCB

2.2.2.2 Rotational Enclosure Electronics

The rotational enclosure provides a waterproof space for the HM and assorted sensors/drivers. Near the central axis of the robot is an AMT203-V absolute encoder which records the gantry's rotational displacement. It connects over SPI into the HM and provides a sufficient 4096-count resolution. The limit switch for aligning the trolley is placed on the exterior of the enclosure and routes to the HM. To reduce the long wires going to the wheel motor to just two, its BTS7960 motor driver is placed here. This driver runs off the 12V power line and takes in PWM from the HM. Finally, an opto-isolated relay allows the HM to control 24V power to the HVEC. This full setup can be seen in figure 22 below.

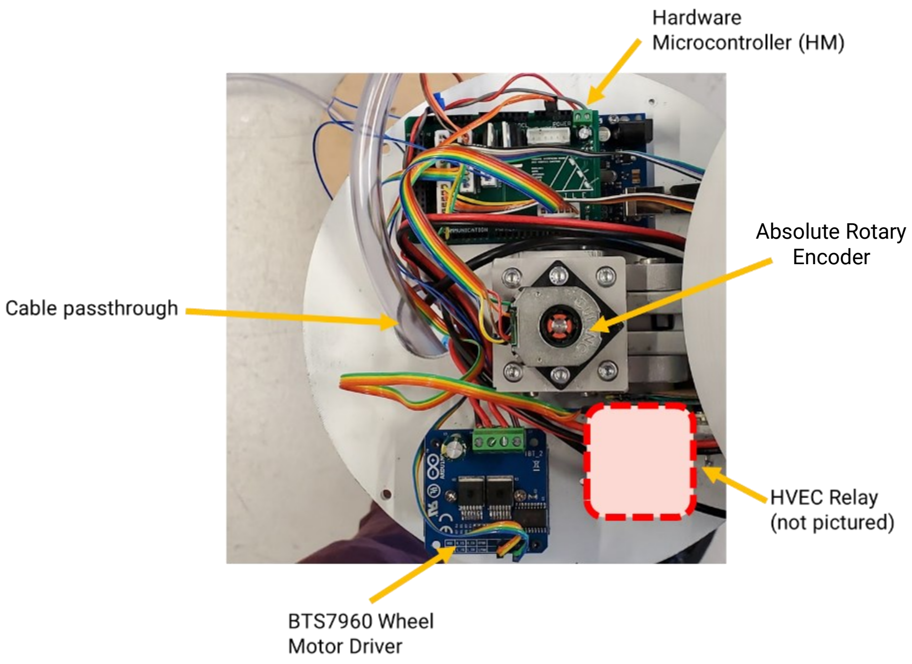

Hardware Microcontroller (HM)
Absolute Rotary Encoder
Cable passthrough
HVEC Relay (not pictured)
BTS7960 Wheel Motor Driver

Figure 22. Rotational Enclosure Electronics

2.2.2.3 Trolley Electronics

The trolley is the most electrically complex section of the robot as it must be mobile while supporting all drivers and sensors. An image of the trolley PCB interfacing with assorted trolley systems is displayed in figure 23. Power and data cables route in through waterproof inserts. This includes the 24V motor line which powers the vertical stepper through a DM452T driver as well as the trolley motor through a BTS7960 driver. The HVEC 24V line is run to the ZVS driver within the trolley, with its ground return path leading back to its relay. All cables from the HM are routed into the Trolley PCB, which acts to conglomerate all trolley peripherals. This board again uses JST-XH connectors for all interfaces with emphasis on keeping long-distance signals as digital transmissions for noise reduction. Leaning into this, an AD7995 10 Bit ADC is used on the board to digitize temperature and moisture data over the I^2C line. Encoder values from the HEDS-5640 are decoded through the LS7184N quadrature clock converter IC for easier reading.

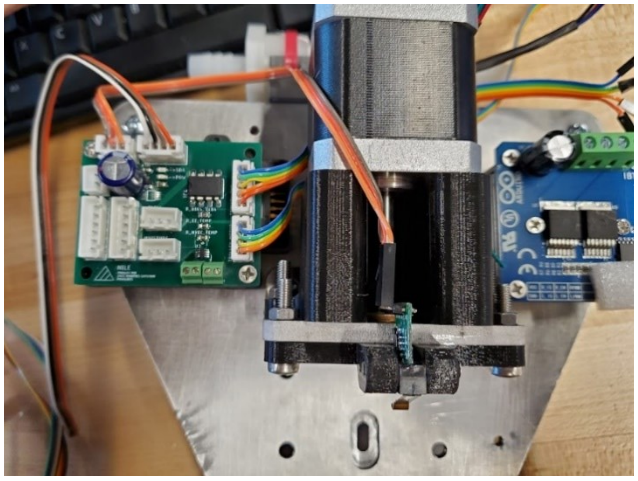


Figure 23. Trolley PCB Interfacing with Assorted Trolley Systems

Throughout integration, this end of the robot received a lot of attention as the long ribbon cables were subject to attenuation and interference. To help this, capacitors were added on select digital lines to filter out noise. Ensuring that the sensor systems receive adequate 5V power, an additional 24V->5V buck regulator is used, running off the 24V motor line, seen in figure 24.

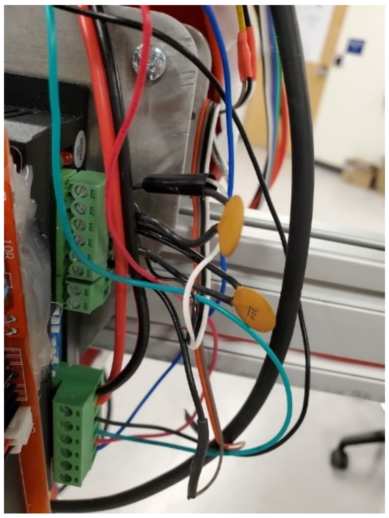


Figure 24. Capacitive Noise Reduction on the Stepper Driver Lines

2.2.2.4 End Effector Electronics

As the main sensor package of the robot, the end-effector contains a thermistor temperature sensor and a capacitive moisture sensor. These route into the Trolley PCB, being digitized via the onboard ADC. High voltage wires from the HVEC route down to their prongs on the end-effector to produce an arc.

2.3 Software Design

Due to our goal of creating a completely autonomous system capable of caring for plants from sow to harvest, the NILE software design is exceptionally complex. First, the is a significant amount of low level I/O operations required to interface with sensors and control drivers.

Second, computer vision and machine learning operations to enable autonomy would require significant computational power. Due to these separate focus areas, we implemented two separate processing units: the Hardware Microcontroller (HM) and the System Determination Computer (SDC).

The HM is an Arduino Mega development board responsible for all the low-level communication needs of the robot. It is then in bidirectional communication with the SCD via the Robotic Operating System (ROS) framework to send data and receive direction. The SDC is a NVIDIA Jetson, an embedded computer development board that has sufficient processing power to make NILE autonomous. It is further equipped with the Ubuntu 18.04 operating system, OpenCV, TensorFlow, and a host of other computer vision and machine learning libraries. A high-level block diagram of the software structure can be seen in Figure 25.

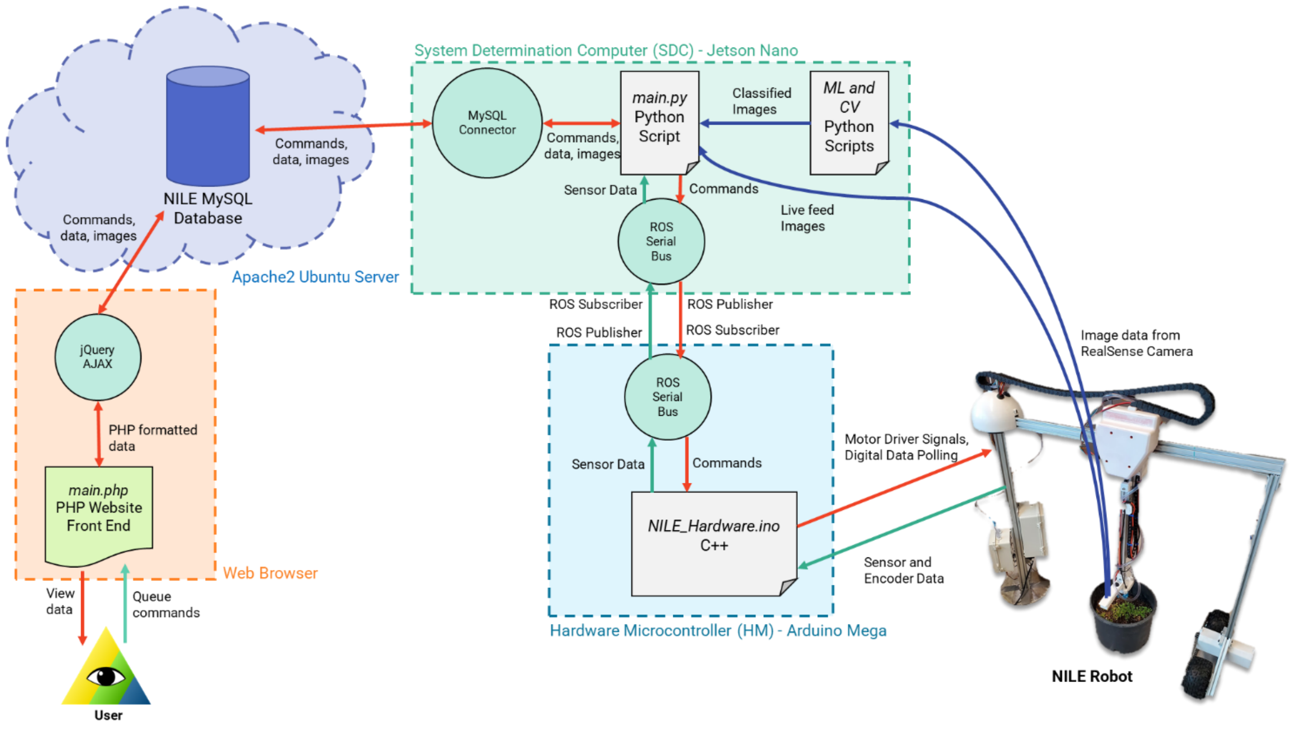


Figure 25. System Software Block Diagram

The current software pipeline starts with user-specified commands on the website. Once a command is entered into the queue, the Apache2 server forwards the instruction to the SDC over the internet. Each command, except for interfacing with the camera, corresponds to data that is passed onto the HM through the ROS protocol. Once received, the HM sends control signals and polls data to and from the appropriate hardware. Data ready for publishing on the website is forwarded from the HM back to the SDC through ROS while commands are still executing. Upon completion of a task, the HM sends status flags back to the SDC. This workflow is illustrated in Figure 26 below.

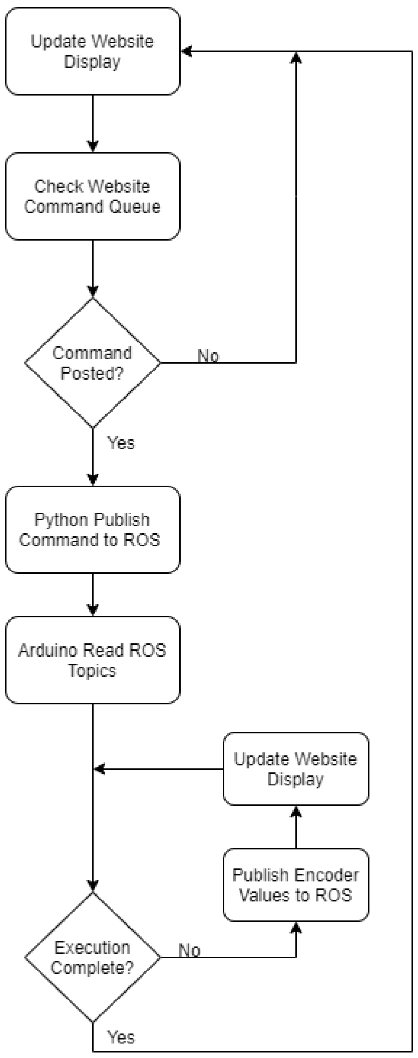


Figure 26. System Software Flowchart

2.3.1 System Website

To control and monitor the robot, a website was chosen due to its vast cross-compatibility and ease of implementation with database services. By using an external site with a database, we could centralize all commands and data, removing the need to directly poll the robot for information. An Ubuntu Apache2 server was hosted on the http://nilerobot.info domain, allowing for basic SSH, HTML, and FTP access. With this, our MySQL database containing all system data can be interfaced by the website and the robot.

2.3.1.1 Website Back-End

The back end of the website contains all supporting elements to keep the website hosted and data organized. As stated previously, we use a MySQL database to store system data, where we create tables with specific columns based on the data's structure. These tables are described in table 2.

Table 2. MySQL Table Definitions

Table Name	Table Columns	Column Description
System Info	ID	Unique row id
	IP Address	Local IP of robot, used for VNC connection
	Last Boot	Timestamp of last bootup
Users	ID	Unique row id
	Username	Username
	Password	User password, hashed
	Role	‘admin’ or ‘guest’
Robot Status	ID	Unique row id
	Timestamp	Timestamp of current position measurement
	Theta	Theta axis measurement (deg)
	R	R axis measurement (m)
	Z	Z axis measurement (m)
Queued Commands	ID	Unique row id
	Timestamp	Timestamp to queue command for
	Command	**Robot Control Commands** “*moveTo*” – Moves to position “*homeTrolley*” - Homes the trolley “*homeVert*” -Homes the vertical stepper “*homeRot*” - Homes to 0 absolute rotation angle “*hydrate*” – D0 passes measured water amount “*hvec*” – D0, D1 pass on/off times in ms, I0 passes pulse # “*takeImage*” – Publishes RealSense image “*goLive*” – Enables live mode on website “*senseSoil*” – Measures soil temperature and moisture “*kill*” – Safely exits ROS serial node
	Theta Queued	Theta to go to, only for “*moveTo*”
	R Queued	R to go to, only for “*moveTo*”
	Z Queued	Z to go to, only for “*moveTo*”
	D0	Additional double-type argument
	D1	Additional double-type argument
	I0	Additional integer-type argument
Completed Commands	ID	Unique row id
	Queued Timestamp	Timestamp that the command was queued for
	Finished Timestamp	Timestamp the command was finished at
	Command	Robot control command *see queued commands
	Theta Queued	Theta to go to, only for “*moveTo*”
	R Queued	R to go to, only for “*moveTo*”
	Z Queued	Z to go to, only for “*moveTo*”
	Theta Actual	Theta measurement at command completion
	R Actual	R measurement at command completion
	Z Actual	Z measurement at command completion

	D0	Additional double-type argument that was queued
	D1	Additional double-type argument that was queued
	I0	Additional integer-type argument that was queued
	Message	Indicates additional information about command status
Soil Samples	ID	Unique row id
	Timestamp	Timestamp of soil measurement
	Theta	Theta of soil measurement
	R	R of soil measurement
	Z	Z of soil measurement
	Moisture	Measured moisture of soil, 0-1024 integer
	Temperature	Measured temperature of soil, Celsius
Camera Images	ID	Unique row id
	Timestamp	Timestamp of image
	Theta	Theta position of image
	R	R position of image
	Z	Z position of image
	Image	Base 64 encoded string with JPEG image information

Integral to the entire control scheme of the robot, this database includes the *Queued* and *Completed Commands* tables. Commands are scheduled via the user on the website by inserting entries into the *Queued Commands* table. Elaborated in *3.2*, The SDC Python scripts for MySQL Connector and ROS parse through the latest entry on this table, executing it at its timestamp. Once a confirmation of success has been received from the HM, the queued command is moved into the *Completed Commands* table to save a comprehensive history of the robot's actions. This structure allows for dynamic and recursive control of the robot, as it can queue up commands for itself following field analysis.

2.3.1.2 Website Front-End

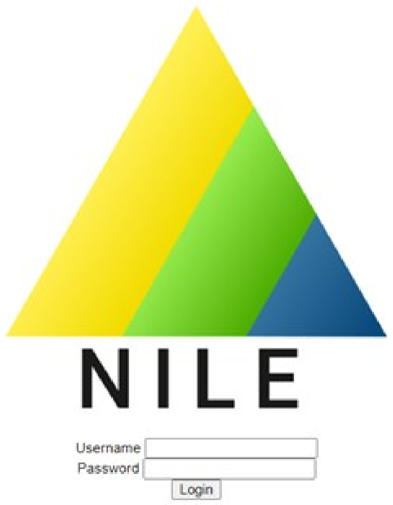
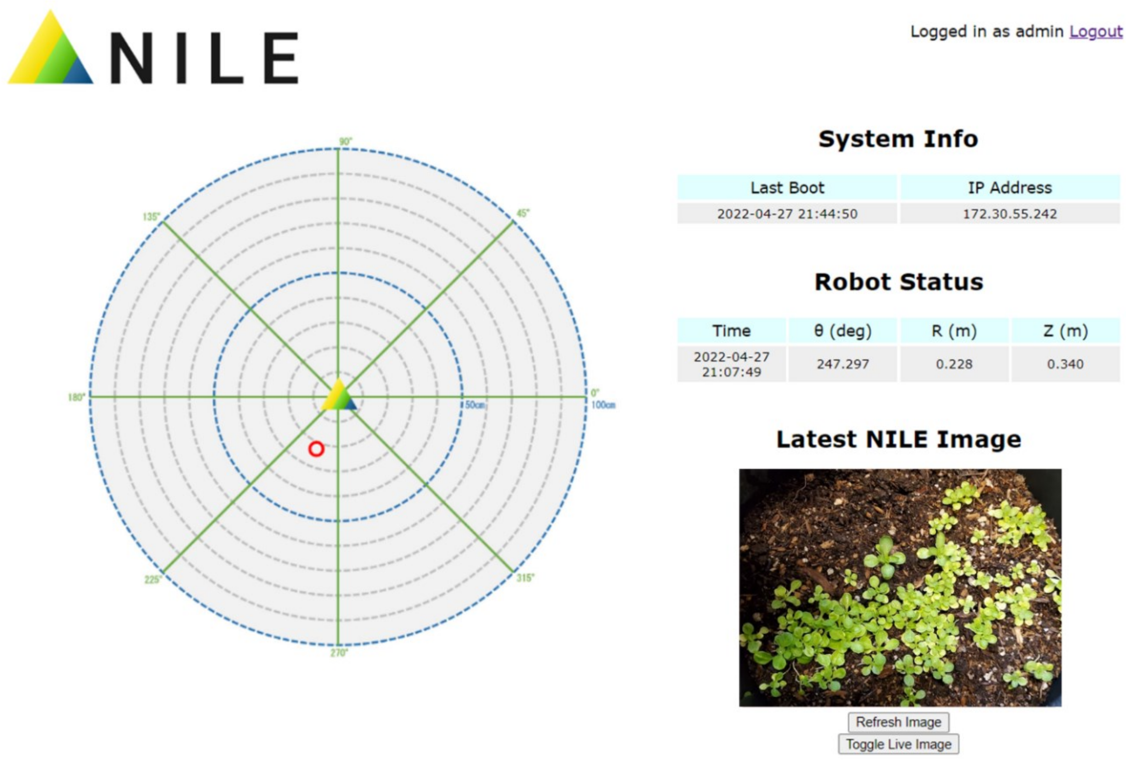


Figure 27. NILE Website Login Page

Before entry into the site, one must log into one of the two user accounts, the website login display shown in figure 27 above. Currently, there exists a guest account with username guest and password guest in addition to an admin account. This distinction is incredibly important, as we only want admins to be capable of commanding the robot. The guest account is limited to only viewing database information. This is done at a band-limited rate as well of one update per 10 seconds, contrasting with the one update per second for the admin account. The main display can be seen in figure 28 below.



System Info

Last Boot	IP Address
2022-04-27 21:44:50	172.30.55.242

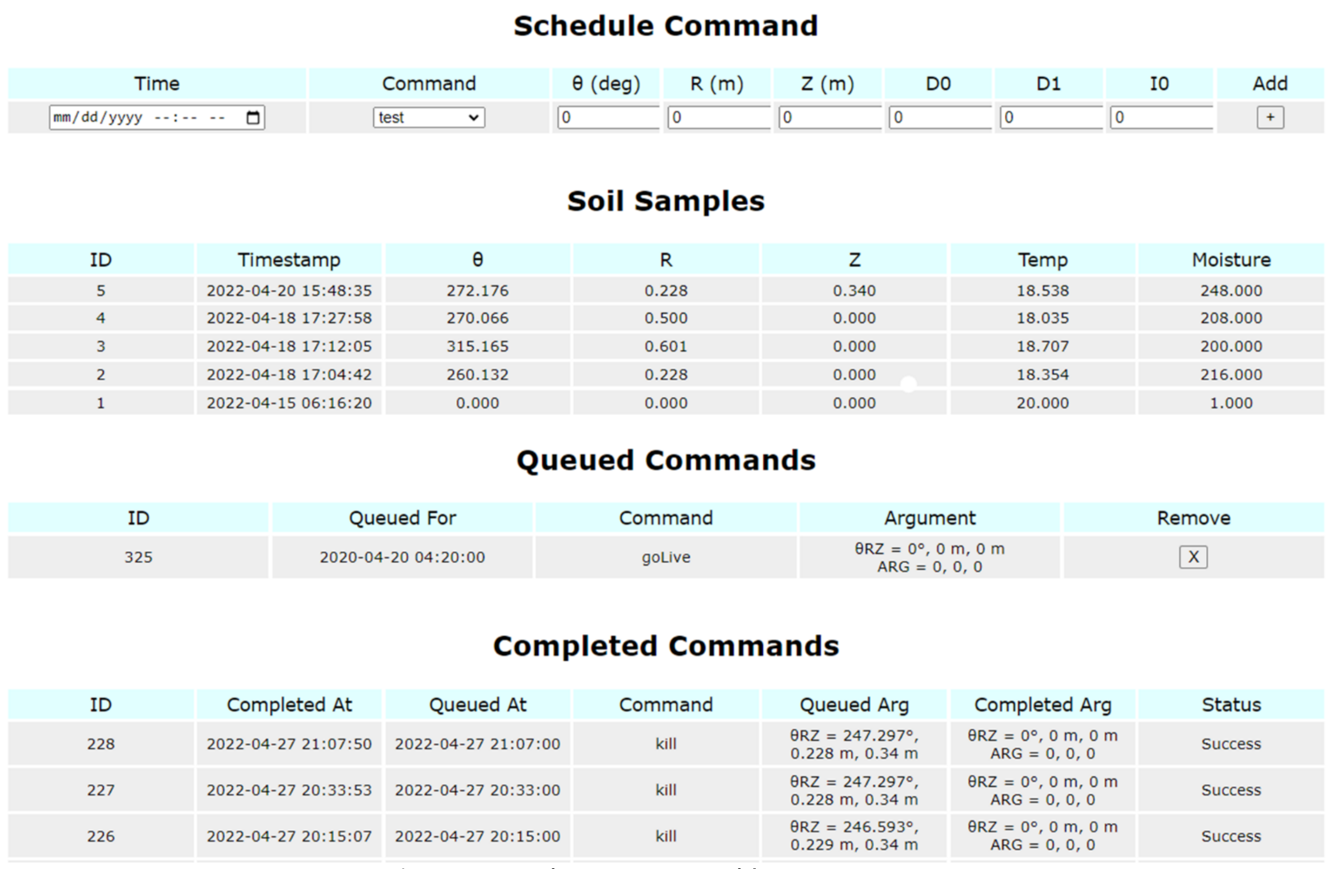
Robot Status

Time	θ (deg)	R (m)	Z (m)
2022-04-27 21:07:49	247.297	0.228	0.340

Latest NILE Image

Figure 28. Website Main Display

The front-end of the website acts as a real time interface from the database to the user. Using the JavaScript module, jQuery AJAX, PHP code can be run on demand to change the site contents without requiring a page refresh. This allows for the data to be updated in near “real-time” (1 second latency, 1 update per second) with zero user interaction required. This appears as the dynamic tables relaying MySQL information as well as the live image feed and live coordinate position map. For this coordinate map, an external JavaScript module was created to draw the robot’s position through an HTML5 canvas element.



Schedule Command

Time	Command	θ (deg)	R (m)	Z (m)	D0	D1	I0	Add
mm/dd/yyyy --:-- --	test	0	0	0	0	0	0	+

Soil Samples

ID	Timestamp	θ	R	Z	Temp	Moisture
5	2022-04-20 15:48:35	272.176	0.228	0.340	18.538	248.000
4	2022-04-18 17:27:58	270.066	0.500	0.000	18.035	208.000
3	2022-04-18 17:12:05	315.165	0.601	0.000	18.707	200.000
2	2022-04-18 17:04:42	260.132	0.228	0.000	18.354	216.000
1	2022-04-15 06:16:20	0.000	0.000	0.000	20.000	1.000

Queued Commands

ID	Queued For	Command	Argument	Remove
325	2020-04-20 04:20:00	goLive	θRZ = 0°, 0 m, 0 m ARG = 0, 0, 0	X

Completed Commands

ID	Completed At	Queued At	Command	Queued Arg	Completed Arg	Status
228	2022-04-27 21:07:50	2022-04-27 21:07:00	kill	θRZ = 247.297°, 0.228 m, 0.34 m	θRZ = 0°, 0 m, 0 m ARG = 0, 0, 0	Success
227	2022-04-27 20:33:53	2022-04-27 20:33:00	kill	θRZ = 247.297°, 0.228 m, 0.34 m	θRZ = 0°, 0 m, 0 m ARG = 0, 0, 0	Success
226	2022-04-27 20:15:07	2022-04-27 20:15:00	kill	θRZ = 246.593°, 0.229 m, 0.34 m	θRZ = 0°, 0 m, 0 m ARG = 0, 0, 0	Success

Figure 29. Website MySQL Table Section

The website allows for a detailed look at all the MySQL tables that pertain to the robot. Users logged in as admin can schedule commands using the top table. Additionally, they can remove queued commands through an X. These use HTML forms that both run PHP code to modify the database as the user requests. A screenshot of the website display is shown in figure 29 above.

2.3.2 Data Pipelines

Part of the job of the SDC is to be able to receive data and control the HM while interchanging data with the online database. For this job, we used two incredibly robust interfaces: MySQL Connector and ROS, running off a singular Python script.

2.3.2.1 MySQL Connector

MySQL Connector was used to handle all communications between the SDC and the MySQL database. It provides a succinct way to issue commands to the database through Python. A library was written by the team to handle common processes, such as: requesting the latest command, requesting the time until the latest command, moving queued commands to complete, publishing coordinates, publishing images, and publishing sensor values. All together these enabled full system control and analytics from the website. A general flow chart for this operation is shown below in Figure 30.

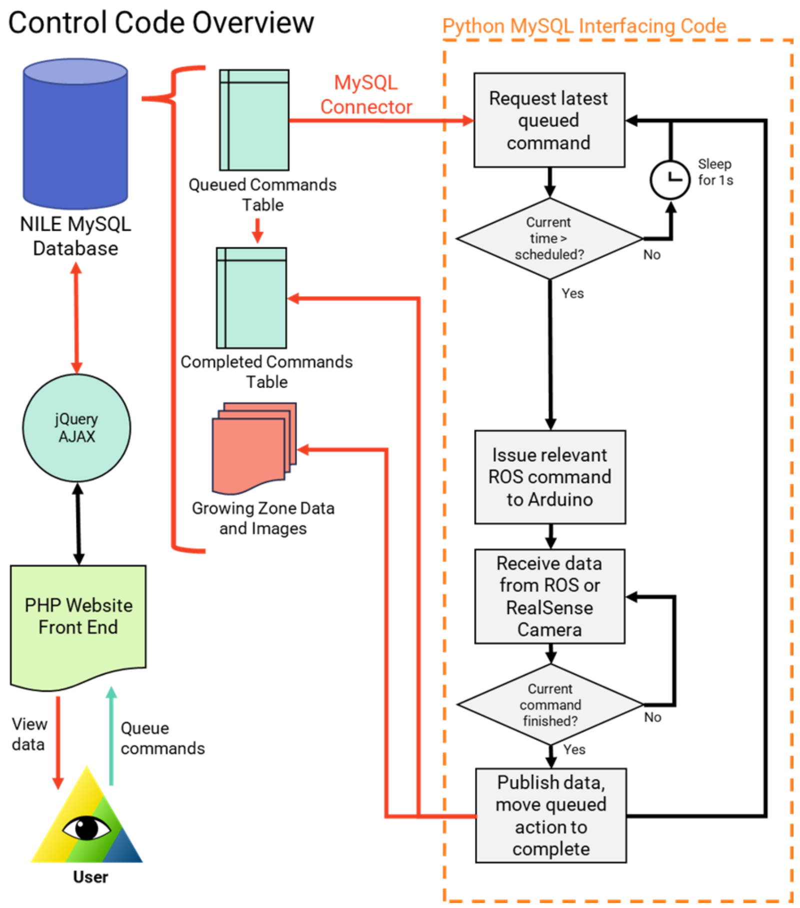

Control Code Overview
Python MySQL Interfacing Code
NILE MySQL Database
MySQL Connector
Queued Commands Table
Completed Commands Table
Growing Zone Data and Images
jQuery AJAX
PHP Website Front End
View data
Queue commands
User
Request latest queued command
Sleep for 1s
Current time > scheduled?
No
Yes
Issue relevant ROS command to Arduino
Receive data from ROS or RealSense Camera
Current command finished?
No
Yes
Publish data, move queued action to complete

Figure 30. MySQL Interfacing Code Diagram

2.3.2.2 ROS Communication

To facilitate efficient data transfer between the Jetson Nano (SDC) and the Arduino Mega (HM), our system makes use of the Robotic Operating System (ROS) protocol. ROS provides streamlined serial communication by way of *topics* and *nodes*. Each ROS topic corresponds to a specific function or peripheral actuated by the Hardware Microcontroller. Figure 31 displays all the topics and nodes employed by our system. Commands from the SDC are sent to the HM through ROS publishers in a Python script. The HM receives the packaged commands via of isolated Arduino ROS subscribers and interacts with the specified hardware. During and after the completion of each command, the HM sends back status flags and sensor readings back to the SDC through ROS publishers on the Arduino side, this data is received by the running Python script and passed to the website via MySQL Connector.

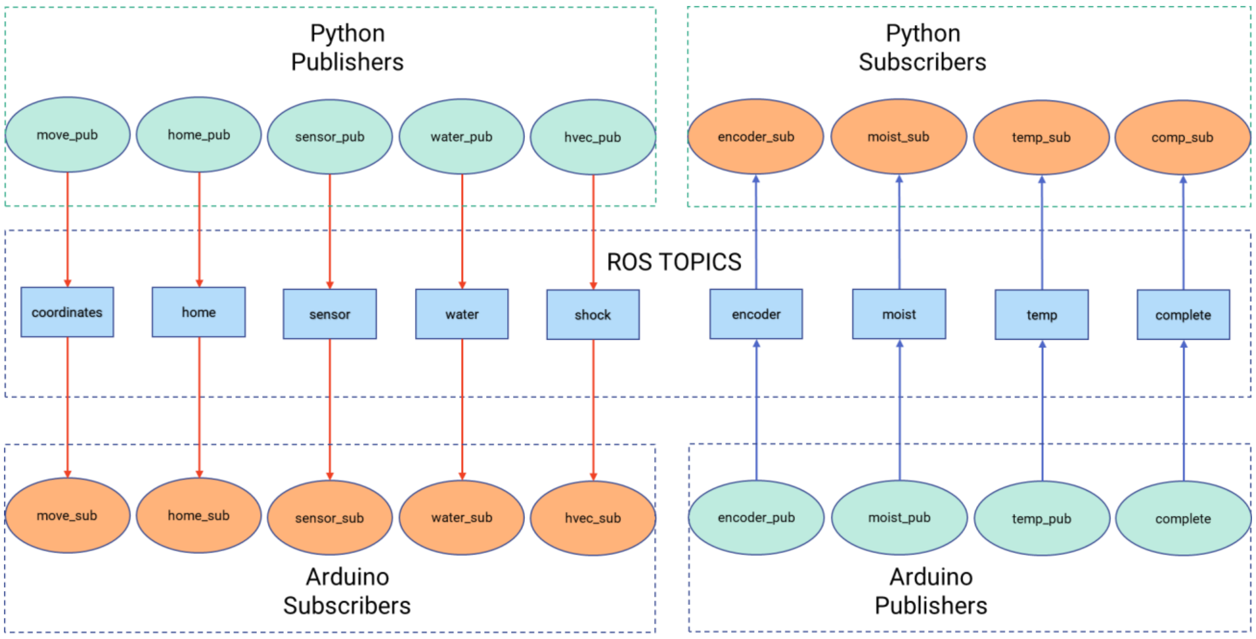
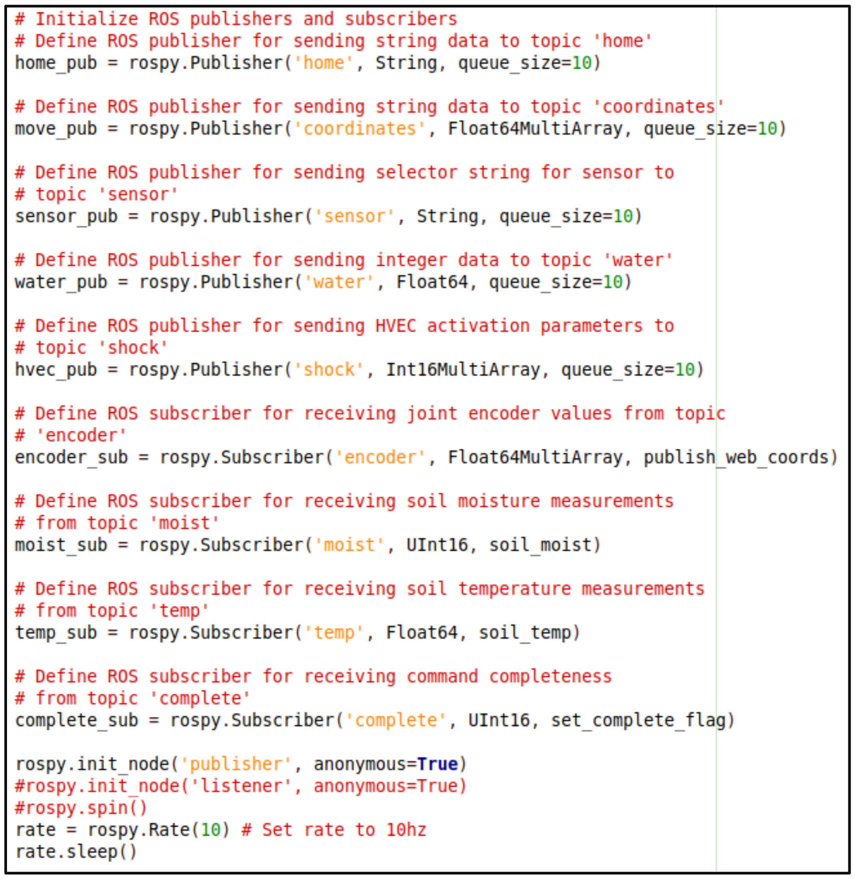


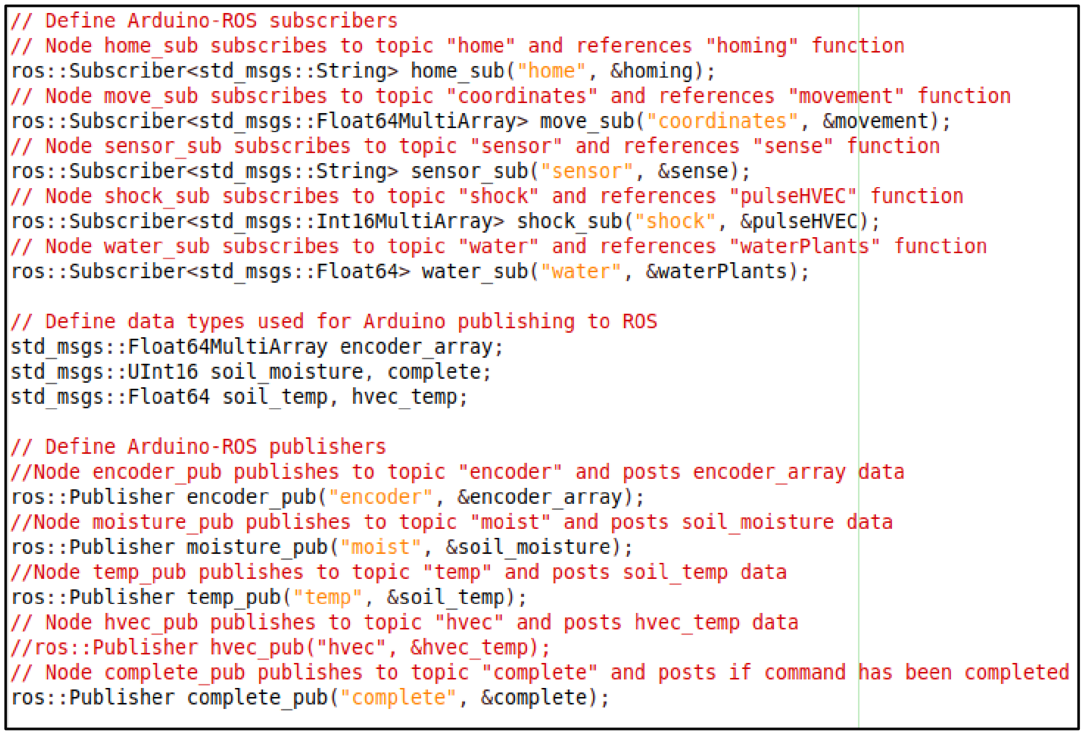
Figure 31. Diagram of Python-ROS-Arduino Interface

Initialization of these ROS nodes required scripting on the Jetson Nano with Python scripting, and separate code on the Arduino with C-programming. This is displayed in Figures 32 and 33.



```
# Initialize ROS publishers and subscribers
# Define ROS publisher for sending string data to topic 'home'
home_pub = rospy.Publisher('home', String, queue_size=10)

# Define ROS publisher for sending string data to topic 'coordinates'
move_pub = rospy.Publisher('coordinates', Float64MultiArray, queue_size=10)

# Define ROS publisher for sending selector string for sensor to
# topic 'sensor'
sensor_pub = rospy.Publisher('sensor', String, queue_size=10)

# Define ROS publisher for sending integer data to topic 'water'
water_pub = rospy.Publisher('water', Float64, queue_size=10)

# Define ROS publisher for sending HVEC activation parameters to
# topic 'shock'
hvec_pub = rospy.Publisher('shock', Int16MultiArray, queue_size=10)

# Define ROS subscriber for receiving joint encoder values from topic
# 'encoder'
encoder_sub = rospy.Subscriber('encoder', Float64MultiArray, publish_web_coords)

# Define ROS subscriber for receiving soil moisture measurements
# from topic 'moist'
moist_sub = rospy.Subscriber('moist', UInt16, soil_moist)

# Define ROS subscriber for receiving soil temperature measurements
# from topic 'temp'
temp_sub = rospy.Subscriber('temp', Float64, soil_temp)

# Define ROS subscriber for receiving command completeness
# from topic 'complete'
complete_sub = rospy.Subscriber('complete', UInt16, set_complete_flag)

rospy.init_node('publisher', anonymous=True)
#rospy.init_node('listener', anonymous=True)
#rospy.spin()
rate = rospy.Rate(10) # Set rate to 10hz
rate.sleep()
```

Figure 32. Python Script for ROS Setup



```
// Define Arduino-ROS subscribers
// Node home_sub subscribes to topic "home" and references "homing" function
ros::Subscriber<std_msgs::String> home_sub("home", &homing);
// Node move_sub subscribes to topic "coordinates" and references "movement" function
ros::Subscriber<std_msgs::Float64MultiArray> move_sub("coordinates", &movement);
// Node sensor_sub subscribes to topic "sensor" and references "sense" function
ros::Subscriber<std_msgs::String> sensor_sub("sensor", &sense);
// Node shock_sub subscribes to topic "shock" and references "pulseHVEC" function
ros::Subscriber<std_msgs::Int16MultiArray> shock_sub("shock", &pulseHVEC);
// Node water_sub subscribes to topic "water" and references "waterPlants" function
ros::Subscriber<std_msgs::Float64> water_sub("water", &waterPlants);

// Define data types used for Arduino publishing to ROS
std_msgs::Float64MultiArray encoder_array;
std_msgs::UInt16 soil_moisture, complete;
std_msgs::Float64 soil_temp, hvec_temp;

// Define Arduino-ROS publishers
//Node encoder_pub publishes to topic "encoder" and posts encoder_array data
ros::Publisher encoder_pub("encoder", &encoder_array);
//Node moisture_pub publishes to topic "moist" and posts soil_moisture data
ros::Publisher moisture_pub("moist", &soil_moisture);
//Node temp_pub publishes to topic "temp" and posts soil_temp data
ros::Publisher temp_pub("temp", &soil_temp);
// Node hvec_pub publishes to topic "hvec" and posts hvec_temp data
//ros::Publisher hvec_pub("hvec", &hvec_temp);
// Node complete_pub publishes to topic "complete" and posts if command has been completed
ros::Publisher complete_pub("complete", &complete);
```

Figure 33. Arduino Script for ROS Setup

2.3.3 Control Algorithms

The control system was implemented as three separate controllers. The controller accepts the desired joint variables which are calculated using the robot's inverse kinematic functions. Each desired joint variable is fed into its respective controller. The following control scheme was implemented to prevent the end effector arm from brushing against plants:

1) The end effector and trolley are zeroed
2) The gantry is driven to the desired position
3) The trolley is driven to the desired position
4) The end effector is driven to the desired position

The control systems for the rotational and radial translations were implemented using PI control laws. The control laws receive the current coordinate and desired coordinate as inputs and then calculate the following error values:

$$e = (x_d - x)$$
$$e_i = e_i + e \cdot \Delta T$$
$$e_d = \frac{e - e_{last}}{\Delta T}$$

These error values are then used with the gains in the following equations to calculate a PWM value:

$$PWM_\theta = -1 \cdot (K_{p,\theta} e_\theta + K_{i,\theta} e_{i,\theta} + K_d e_{d,\theta})$$
$$PWM_d = K_{p,d} e_d + K_i e_{i,d} + K_d e_{d,d}$$

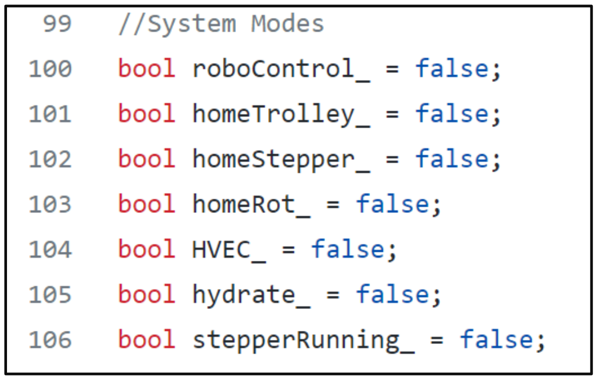
The rotational PWM is made negative due to direction handling of the motor. The gain values in the equations above were found experimentally. A Kp value of 250 was used in both controllers to give the system better speed at the start, even if the initial error was small. A Ki value of 10 was also used in both controllers. The integral control was added to prevent the system from undershooting. Derivative control was implemented because the integral control would cause the system to overshoot when the initial error was large. A Kd value of 10 in the rotational control law and a Kd value of 5 in the horizontal control law were found to prevent the overshoot.

The control system for the vertical translation joint was very simple because an external library handled the control. The AccelStepper library can control the velocity and acceleration of a stepper motor while driving the motor to the desired position. A control function was created using functions from the AccelStepper library. The desired height coordinate is transformed into a step count and passed to the control function using the equation below. The stepper motor is then driven by the AccelStepper functions within the control function until the desired step count is reached.

$$Steps = \frac{Vertical\ Distance(m) \cdot 400\frac{Steps}{Rotation}}{0.01905\frac{m}{Rotation}}$$

2.3.3.2 Arduino Implementation

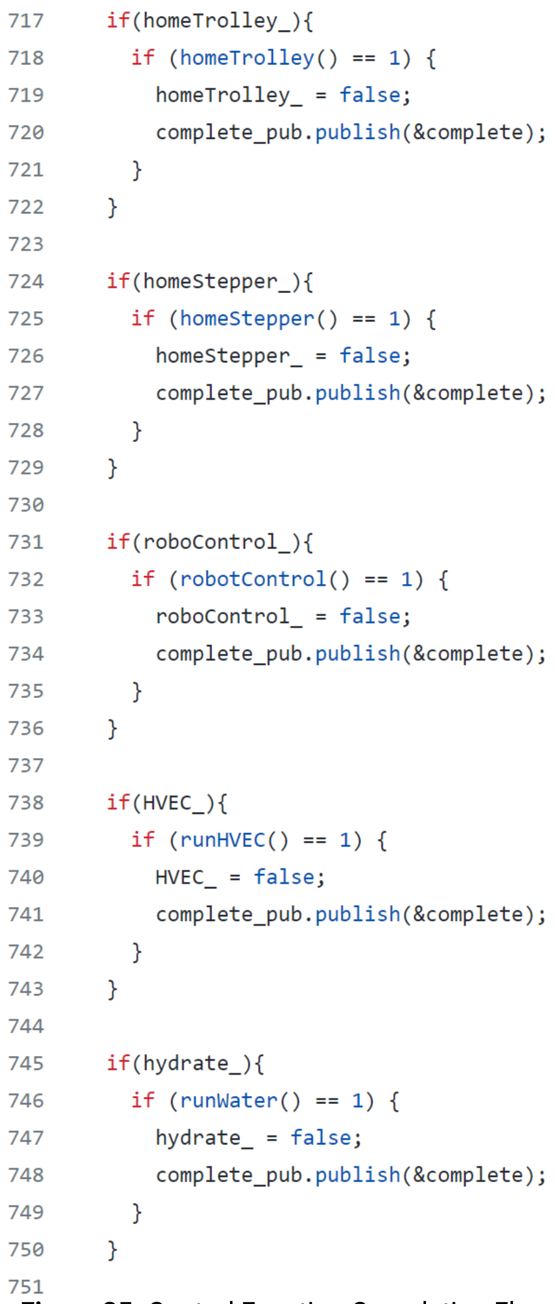
For algorithmic organization, the control laws described above were encoded into Arduino functions using status flags. Each mode of system motion was assigned a Boolean variable to prevent multiple control laws from attempting simultaneous execution. Figures 34 and 35 display this.



```
//System Modes
bool roboControl_ = false;
bool homeTrolley_ = false;
bool homeStepper_ = false;
bool homeRot_ = false;
bool HVEC_ = false;
bool hydrate_ = false;
bool stepperRunning_ = false;
```

Figure 34. Boolean Status Flags

Each Arduino control function is then linked to a status flag to denote whether it is has finished execution. Completion flags are then passed onto ROS.



```
if(homeTrolley_){
  if (homeTrolley() == 1) {
    homeTrolley_ = false;
    complete_pub.publish(&complete);
  }
}

if(homeStepper_){
  if (homeStepper() == 1) {
    homeStepper_ = false;
    complete_pub.publish(&complete);
  }
}

if(roboControl_){
  if (robotControl() == 1) {
    roboControl_ = false;
    complete_pub.publish(&complete);
  }
}

if(HVEC_){
  if (runHVEC() == 1) {
    HVEC_ = false;
    complete_pub.publish(&complete);
  }
}

if(hydrate_){
  if (runWater() == 1) {
    hydrate_ = false;
    complete_pub.publish(&complete);
  }
}
```

Figure 35. Control Function Completion Flags

2.3.4 Image Processing

To make informed decisions in caring for the growing zone, our system needed to be able to detect, label, and localize plants at different stages of development. To this end, the NILE robot was designed to leverage both machine learning and computer vision assessing plants.

2.3.4.1 Machine Learning

Commonly referred to as image classification, the machine learning implemented on the NILE robot was primarily concerned with receiving a raw image in real time and being able to categorize the contents with minimal pre-processing. To achieve this, we made use of supervised learning with Convolutional Neural Networks (CNN). Mathematically, this approach iterates through square segments of an image, performing numerical convolutions along the way. The results of these operations are then fed into a *deep* neural network; an interconnected array of scalar nodes whose individual outputs are related to the values of inputs being received. A neural network familiar with specific patterns in an image will generate a consistent range of outputs when exposed to a new image containing that pattern. The specific neural network adopted by the NILE algorithm is the MiniVGGNet, a relatively basic deep neural network comprised of 23 nodal layers. This architecture was inspired by a case study in which the MiniVGGNet was used to classify 17 common flower species [11].

Teaching our CNN to recognize plants of interests required that a data set of training images be compiled. For proof-of-concept, the NILE robot needed to be able to distinguish between mature mache, generic plant sprouts, and dead leaves. By exposing the CNN to a collection of 1,365 pictures over the course of dozens of training iterations, our system was able to generalize patterns in images containing an item from that training category with 99% accuracy. Figure 36 captures the performance of the CNN against an unfamiliar set of new images.

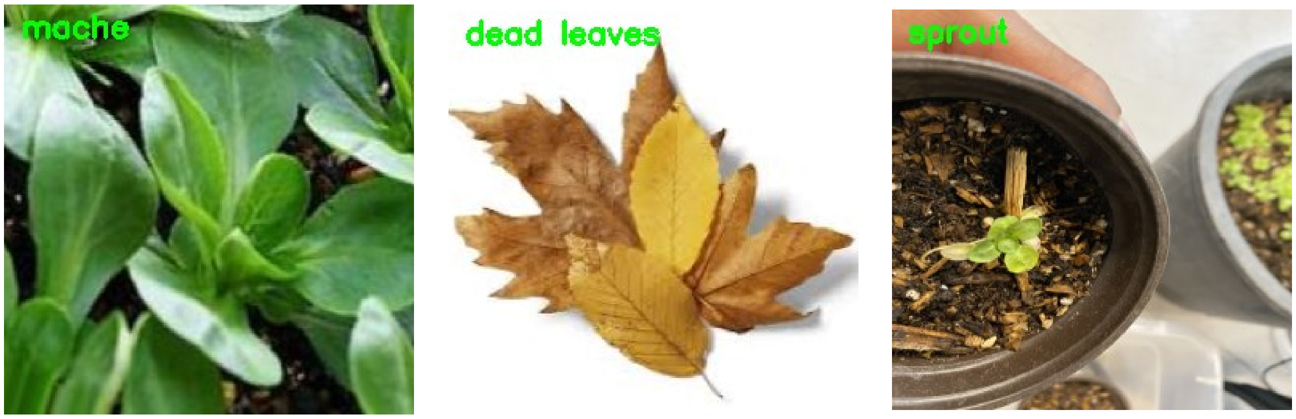


Figure 36. Convolutional Neural Network in Action

As a side note for further development, the intent of our design is to combine this image labeling with object localization, a technique in which the algorithm iterates through regions of interest (ROI) within a target image and performs classification at each step. This would enable our system to generate precise coordinates for the location of each detected entity.

2.3.4.2 Computer Vision

In leu of the neural network being able to localize detected plants, the NILE system implements a series of computer vision techniques to create bounding boxes around plants based on physical appearance. This feature extraction consists of four major steps: Gaussian blurring, RGB-to-HSV conversion, color masking, and contour detection. The first stage, Gaussian blurring, is performed to counteract the data noise and background of the image itself and get rid of very small contours, seen in figure 37.

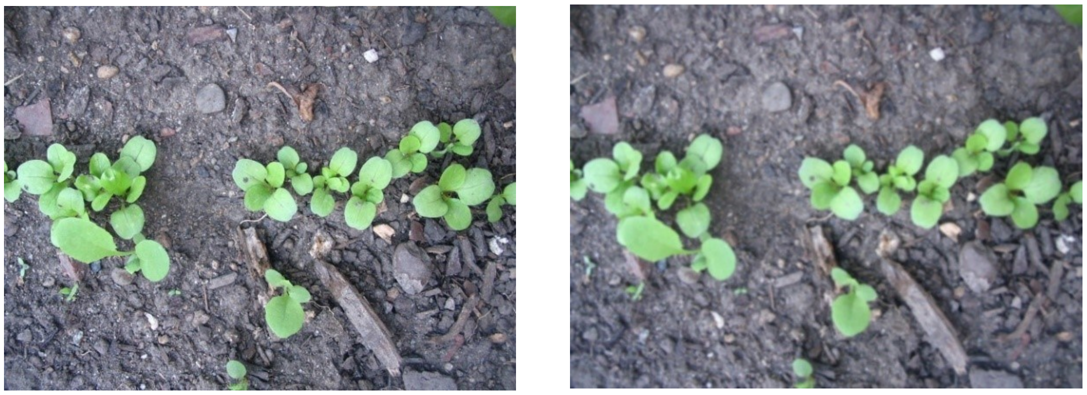


Figure 37. Original Image Versus Gaussian Blur

From there, the image format is transferred from Red-Green-Blue to Hue-Saturation-Value representation, where hue describes the color category, saturation defines how vivid the color is, and value correlates to the brightness. This can be seen in figure 38. This will allow the vision system to properly identify colors even in the presence of shadows and varied lighting conditions.

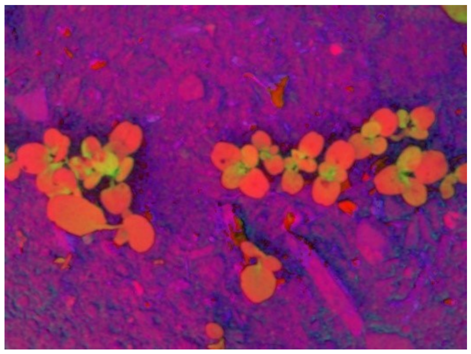


Figure 38. Hue-Saturation-Value Color Space Image

To isolate foliage within the image, the Python script then applies a color mask with ranges that have been determined heuristically. The result of masking is a binary image containing only black and white, where white segments correspond to colors within the desired range, seen in figure 39. This binary image is passed into a contour detection algorithm that groups the blobs into individual objects called contours. These contours define the regions where plant foliage has been detected and is ready for classification by the machine learning inference.

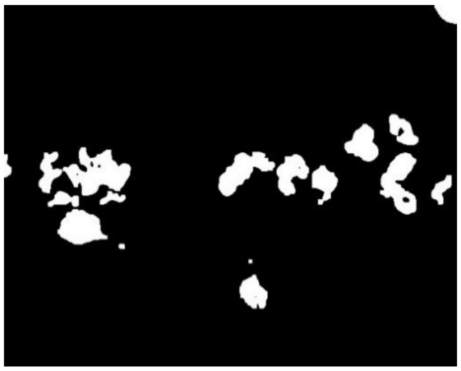
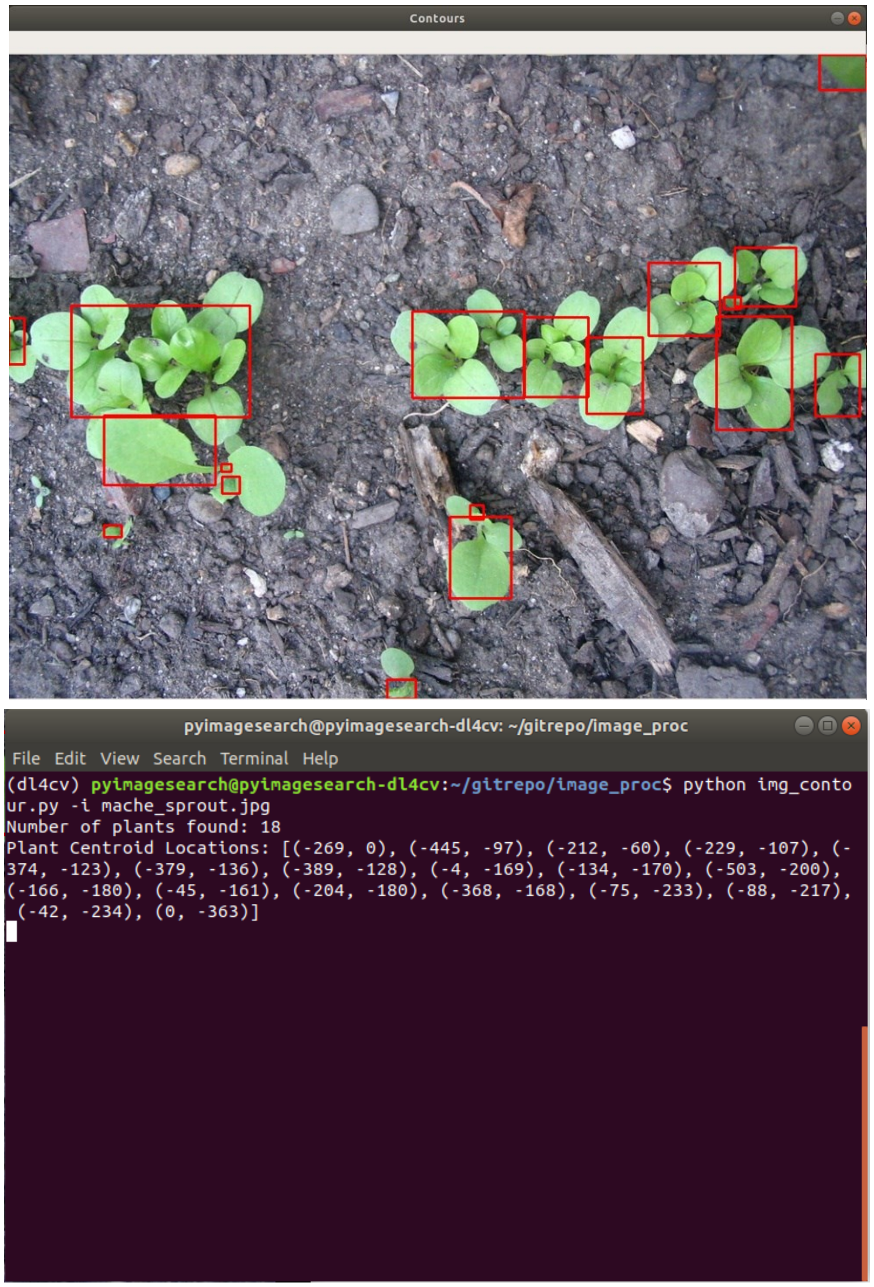


Figure 39. Color Masked Image

From the list of contours, the algorithm computes a bounding box for each entry and calculates its centroid with respect to the center of the frame.


Contours
pyimagesearch@pyimagesearch-dl4cv: ~/gitrepo/image_proc
File Edit View Search Terminal Help

```
(dl4cv) pyimagesearch@pyimagesearch-dl4cv:~/gitrepo/image_proc$ python img_conto
ur.py -i mache_sprout.jpg
Number of plants found: 18
Plant Centroid Locations: [(-269, 0), (-445, -97), (-212, -60), (-229, -107), (-
374, -123), (-379, -136), (-389, -128), (-4, -169), (-134, -170), (-503, -200),
(-166, -180), (-45, -161), (-204, -180), (-368, -168), (-75, -233), (-88, -217),
 (-42, -234), (0, -363)]
```

Figure 40. Computer Vision Plant Localization

Considering the system requirements, this machine learning and computer vision combination represents only an intermediate solution to autonomous assessment of the growing zone, as demonstrated in figure 40 above. However, both approaches demonstrate firm promise for a future endeavor in which the two algorithms are integrated congruently.

2.4 System Integration

Multi-disciplinary system integration of NILE began January 2022 and was completed April 2022. The first step of this process was the acquiring of off-the-shelf components and the manufacturing of custom components. This led into our primary integration challenge, supply chain disruptions stemming from shipping slowdowns and integrated circuit chip shortages. While most of components arrived in reasonable time frames the 80/20 aluminum extrusion components that form the backbone of the NILE mechanical assembly arrived over a month late. While some work could progress during this it significantly delayed integration.

The end frame of our robotic solution drives the angular position of the end effector. The end` effector is attached to a lead screw which controls its vertical positioning. Also, the end effector is attached to the trolley which houses the gantry motors to control the radial position of the end effector.

On the central pivot there are two control boxes, the hydraulic box and the electrical box. The electrical box houses the jetson and the power supply. All the communication and power wires were fed through the energy chain which runs along the top of the gantry. Each wire connects either to components in the trolley house or are then fed through another energy chain down the vertical, lead screw controlled, joint to connect to the end-effector components. Power and communication wires for the end frame motor were fed through the gantry slots and then down the end frame slots to the motor housing. The hydraulic box housed one flowmeter and two solenoid valves. These fluid lines were also fed through the energy chain along the gantry and down the vertical joint to eventually release the water through a mechanical nozzle housed in the end effector.

Power throughout the system is distributed from our 24V power supply to four separate lines through the fuse box. As elaborated previously, this adds additional safety against shorts for high-current devices. Due to the long wires in the system, there was notable attenuation in voltage going from the power box to the trolley. This was acceptable for the 24V lines, but it caused problems for the 5V devices on the trolley. We opted to add an additional 5V regulator on the trolley to fix this, ensuring that the Hardware PCB received adequate voltage.

The flow of information throughout the system required a complex and robust pipeline to go between the robot and the database. Sensor information is amalgamated by the Trolley and Hardware PCBs, with their digital data interpreted by the HM. This data is packaged and sent over a ROS serial publisher to an accompanying ROS subscriber on the SDC. From here, data is uploaded to its appropriate table in the MySQL database via MySQL Connector, centralizing all information in the system. A user can then access that data via the website, which uses PHP to interface with the database. This chain of information goes the other way as well, where commands entered on the website can make it to the SDC, then to the HM to control the mechatronic systems of the robot.

3.0 Testing

After the system was fully assembled and integrated, we performed a variety of tests to verify that the specifications had been met. For each specification, one or more tests were developed and executed. The tests and their results are summarized below in Table 3 and discussed in greater detail in the following subsections.

Table 3. Test Results Summary

Specification	Test	Result
Soil Water Saturation	Resolution and Operating Range	System Error
	Independent Measurements	System Error
Soil Temperature	Resolution and Operating Range	System Error
	Independent Measurements	System Error
Application Efficiency	Flow Rate and Application Efficiency	Pass
Fertilizer Application	Flow Rate	Pass
Weed Extermination	Positional Accuracy	System Error
	Discharge Voltage	Pass
Plant Location Determination	Plant Location Accuracy	Fail
	Plant Location Consistency	Fail
	Plant Location Time	Fail
Ingress Protection	Ingress Protection	Pass
Communication	Communication	Pass

Since conducting the tests in mid-April we solved all the system errors that prevented the successful completion of the moisture and temperature sensing tests in addition to the positional accuracy tests. Unfortunately, there was not sufficient time to reconduct the formal tests or complete the plant location determination test.

3.1 Soil Water Saturation

To verify the resolution and operating range of the soil water saturation sensor, we performed the following test procedure.

1) Fill three equal sized pots with dirt from the same dry source and label them.
2) Measure and record the mass of each pot with dirt.
3) Fill one pot until the water level reaches the dirt level and stays at that level for 1 minute. Record the mass of the water filled pot.
4) Fill one pot with half the mass of water added to the previous pot. Leave one pot completely dry.
5) Utilize the moisture sensor to measure the saturation of the soil in each of the three pots.

To pass the resolution and range test the reading from the fully saturated pot was required to be between 95% to 100%, the half-saturated pot reading between 45% to 55%, and the non-watered pot reading between 0% to 5% saturation.

To verify the ability of the system to take independent measurements, we performed the following test procedure.

1) Initialize the PhaseSpace Motion Tracker System.
2) Install 1 or more PhaseSpace LED trackers securely to the end effector.
3) Home all robot axes then start recording with the PhaseSpace.
4) Randomly generate a point within the growing zone.
5) Command the robot to move to the desired point and record motion with the PhaseSpace.
6) Chose a new point within 10 cm of the initial point.
7) Command the robot to move to the new point.
8) Stop the PhaseSpace Recording.

To pass the independent measurement test, the actual distance between the robot in both positions as recorded by the PhaseSpace was required to be less than 10 cm.

3.1.1 Soil Water Saturation Results

We were unable to complete the resolution and range test due to issues with the capacitive moisture sensor integration that were not resolved by the time of testing. During the test attempt, the system was capable of correctly identifying 0% and 100% soil saturation. However, when attempting to measure the 50% saturated soil the sensor jumped between 0% and 100% reads with no discernable pattern. Initial investigations point to an error in the ADC (analog to digital conversion) chip used to decode the moisture sensor output as opposed to an issue with the sensor itself.

We were unable to complete the independent measurement test due to issues with the stepper motor that drives the vertical axis. For reasons yet to be determined, the motor is locked in the current configuration and does not respond to commands. For the purposes of the independent measurement test we drove the rotational and radial linkages to demonstrate functionality.

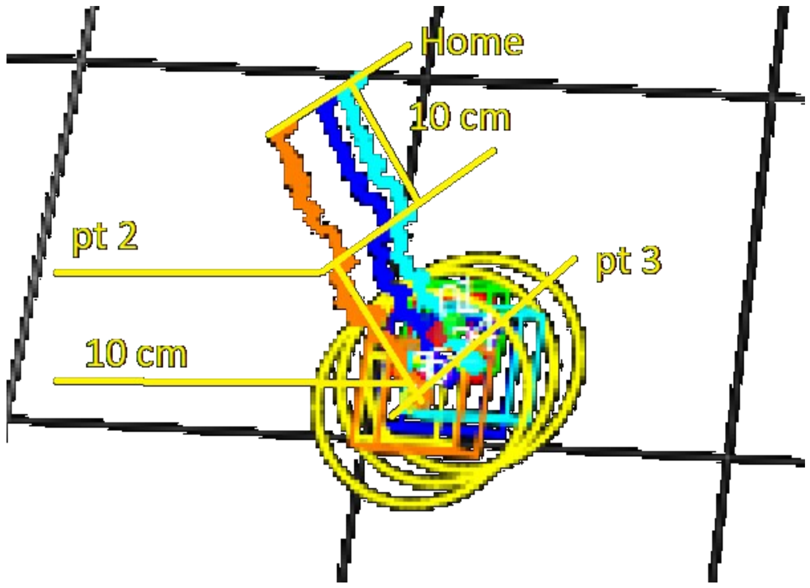
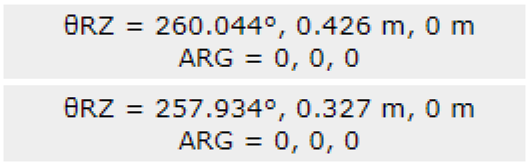


Figure 42. PhaseSpace Positional Results

Figure 42 above shows the output of the PhaseSpace tracking system. The highest points of the figure show the LEDs attached to the end effector zeroed at the home or zeroed position described in the test. Then the robot translates 10 cm radially to point 2 which is in the middle of Figure 42. Finally, the robot is rotated three degrees at point 2 then driven an additional 10 cm radially to reach point 3, completing independent measurement test.

Using the PhaseSpace software and robot's internal position determination we calculated that, between points 2 and 3 in Figure 42, the robot rotated -2.11° and translated 9.9 cm. The positions of points 2 and 3 can be seen below in Figure 43.



```
θRZ = 260.044°, 0.426 m, 0 m
ARG = 0, 0, 0

θRZ = 257.934°, 0.327 m, 0 m
ARG = 0, 0, 0
```

Figure 43. Point 2 and Point 3 Locations

This demonstrates precision control with the two points exactly 9.95 mm apart which would be sufficient to pass the test. Unfortunately, because of the issue with the stepper, neither point was in the growing zone, so it does not pass the test.

3.2 Soil Temperature

To verify the resolution and operating range of the temperature sensor, we performed the following test procedure.

1) Fill three equal sized pots with dirt from the same dry source and label them.
2) Add antifreeze to one pot and place it in a freezer until its temperature is at or below -10°C. Leave one pot at room temperature. Add boiling water to the final pot and ensure that its temperature is at or above 50°C.
3) Measure the temperature of each pot with the temperature sensor in conjunction with a control thermometer. The control thermometer must have a resolution of ±0.5°C or less.

To pass the resolution and range test, the temperature sensor was required to read within ±0.5°C of the control thermometer.

To verify the ability of the system to take independent measurements, we performed the following test procedure.

1) Initialize the PhaseSpace Motion Tracker System.
2) Install 1 or more PhaseSpace LED trackers securely to the end effector.
3) Home all robot axes then start recording with the PhaseSpace.
4) Randomly generate a point within the growing zone.
5) Command the robot to move to the desired point and record motion with the PhaseSpace.
6) Chose a new point within 10 cm of the initial point.
7) Command the robot to move to the new point.

8) Stop the PhaseSpace Recording.

To pass the independent measurement test, the actual distance between the robot in both positions as recorded by the PhaseSpace was required to be less than 10 cm.

3.2.1 Soil Temperature Results

We were unable to complete the resolution and range test due to issues with the thermistor integration that were not resolved by the time of testing. The high temperature sample was the first tested with a verified temperature of 50.5°C. The temperature as recorded by the robot from the point of insertion to 1 minute after are shown below in Figure 44.

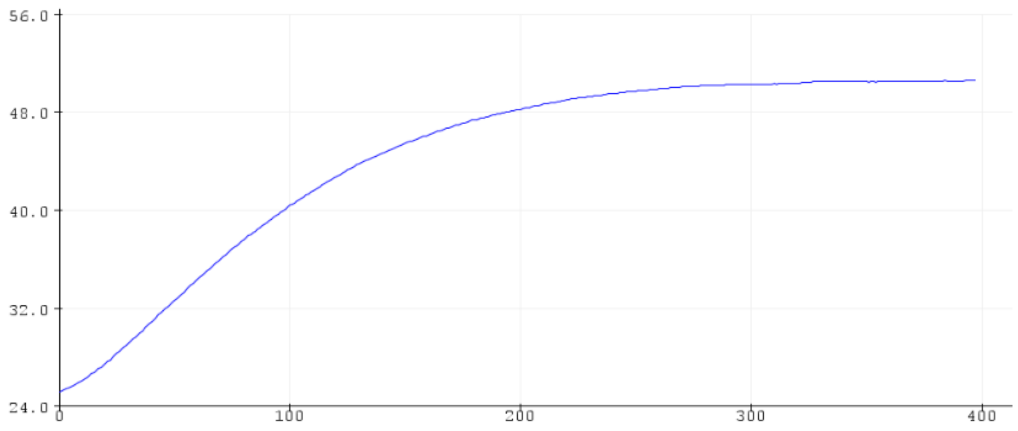


Figure 44. High Temperature Soil Test

In the figure above, the temperature is shown on the y-axis in °C and the number of readings received are in the x-axis. As can be seen, the temperature reads 50°C which is within 0.5°C of the actual temperature required to pass the test.

Next, the room temperature sample was tested and verified to have a temperature of 21.5°C.

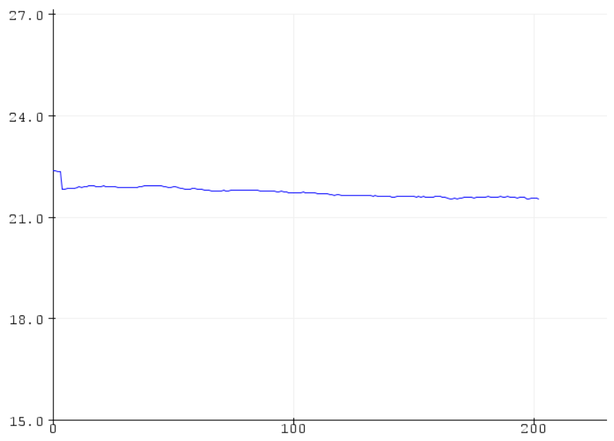


Figure 45. Room Temperature Soil Test

As can be seen in Figure 45, the robot's temperature sensor reading settled on a temperature of 22°C which is inside the 0.5°C of the actual temperature required to pass the test.

Finally, the cold temperature sample was tested and verified to have a temperature of -10°C.

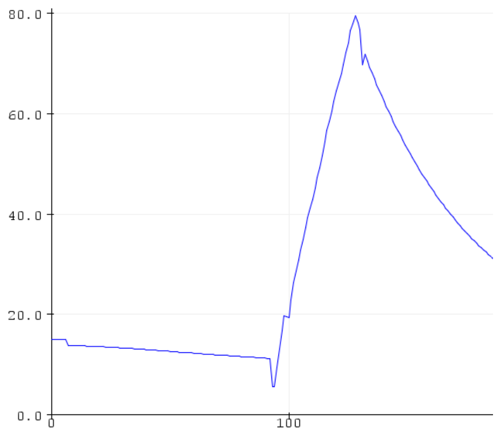


Figure 46. Cold Temperature Soil Test

As can be seen in Figure 46, the output graph does not match the expected result. The readings show an initial but slow drop in the recorded temperature followed by a quick spike to unreasonably high temperatures which only dropped after removal from the sample. This same behavior was repeated on subsequent tests. We are unsure why this occurred as the sensor is rated down to -40°C but we suspect the same ADC chip that caused issues during the moisture tests.

The independent measurement test for the temperature and moisture sensing operations are identical. In summary, the system did not pass the test due to issues with the stepper motor that drives the vertical axis. For a full overview of the results see Section 3.1.1.

3.3 Application Efficiency

To verify the ability of the system to meet the flow rate and field application efficiency specification, we performed the following test procedure.

1) Set a beaker and scale with a resolution of ±1 g or less under the flow outlet.
2) Command the microcontroller to output the water at maximum speed for 10 seconds then shut off the water.
3) Record the mass of the water added to the beaker.

To pass the test, a minimum of 401 g of water was required to be recorded in the beaker at the conclusion of the test. Furthermore, the volume of water recorded by the flow meters was required to be within ±20 mL of the volume calculated from the scale. Finally, the field application efficiency was required to be greater than 90% as calculated by the flow meters.

3.3.1 Application Efficiency Results

We successfully completed the test and verified the ability of the system to deliver water at the required flow rate and resolution. We set up a container and scale under the robot and directed the microcontroller to output the robot for 10 seconds. After the time was up, we recorded 654 g of water in the container as seen below in Figure 47.

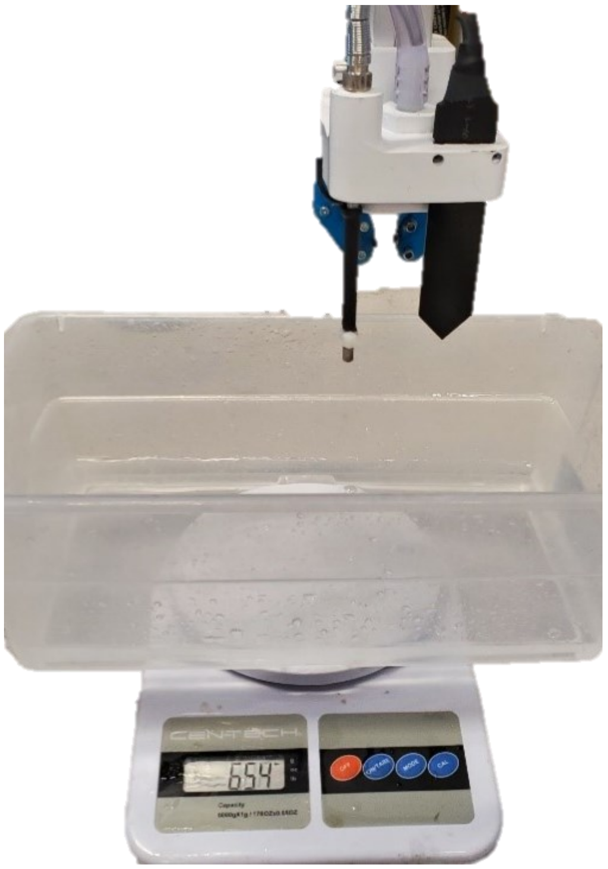


Figure 47. Application Efficient Test

In addition, the robot recorded the delivery of 669 mL of water with the main flow meter and 665 mL of water with the trolley flow meter. This is within the ±20 mL range specified and represents an application efficiency of 99%.

3.4 Fertilizer Application

To verify the ability of the system to meet the flow rate specification, we performed the following test procedure.

1) Set a beaker and scale with a resolution of ±1 g or less under the flow outlet.
2) Command the microcontroller to output the water from the fertilizer delivery system at maximum speed for 10 seconds then shut off the water.
3) Record the mass of the water added to the beaker.

To pass the test, a minimum of 101 g of water was required to be recorded in the beaker at the conclusion of the test.

3.4.1 Fertilizer Application Results

We successfully completed the test and verified the ability of the system to deliver liquid fertilizer at the required flow rate. As directed in the test procedure, we set up a container and scale under the robot and directed the microcontroller to output fertilizer for 10 seconds. After that time, we recorded 653 g of water in the container. This is significantly greater that the 101 g specified.

3.5 Weed Extermination

To verify the ability of the end effector to reach any point in the growing zone with the specified accuracy, we performed the following test procedure.

1) Initialize the PhaseSpace Motion Tracker System.
2) Install 1 or more PhaseSpace LED trackers securely to the end effector.
3) Home all robot axes.
4) Randomly generate a point within the robot's workspace.
5) Command the robot to move to the desired point and record motion with the PhaseSpace.
6) Repeat steps four and five one more time.

To pass the test, the positions recorded by the PhaseSpace were required to be within ±5 mm in the r and z directions and ±.145° in the θ direction.

To verify the specified discharge voltage, we performed the following test procedure.

1) Measure the distance between the HVEC output prongs.
2) Place a plant or weed between the HVEC output prongs.
3) Command the HVEC to activate and verify that arcing occurs between the output prongs.

To pass the test, the HVEC end effector was required to cause arcing between the prongs and the prongs were required to be a minimum of 0.5 cm apart. Given that the breakdown voltage of air is 30 kV/cm this ensures that the required discharge voltage was met.

3.5.1 Weed Extermination Results

We failed the positioning accuracy test and due to insufficient control and issues with the stepper motor that drives the vertical axis. The sensor independent measurement tests and the positioning accuracy test were performed simultaneously, for a full overview of the results see Section 3.1.1.

Perinate to the positioning accuracy test, the robot was commanded to rotate -3° and translate 10 cm radially. An analysis of the internal measurement systems and the PhaseSpace showed that the robot rotated -2.11° and moved 9.9 cm radially. While the radial dimension is well within specification, the rotational accuracy is far outside of spec and the vertical translation was nonfunctional. While both the vertical stepper and radial driver are capable of meeting the specification, they need further integration troubleshooting and further controller development respectively to be brought up to spec.

We successfully tested and verified the system's ability to create the required discharge voltage for the purpose of weed elimination. First, the distance between the innermost surface prongs was recorded to be 0.75 cm. Then, a sacrificial mache was placed between the HVEC output prongs and the HVEC commanded to fire. The HVEC successfully arced and eliminated the plant as shown below in Figure 48.

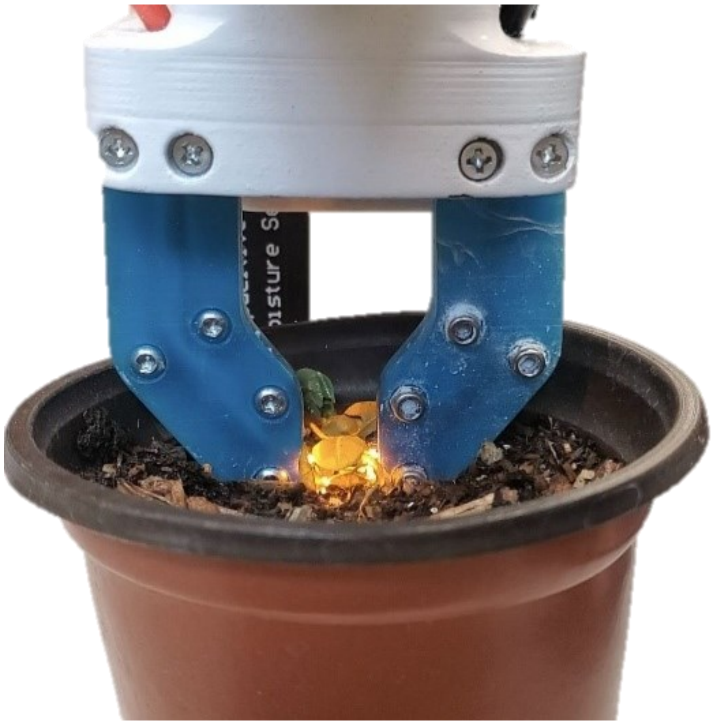


Figure 48. Weed Extermination Test

3.6 Plant Location Determination

To verify the ability of the system to determine the location of plant foliage, plant stems, and weeds with the accuracy specified, we performed the following test procedure.

1) Place three plants in the growing zone.
2) Command the system to determine the location of the plants within the growing zone.
3) Record the number of plants identified and their positions.
4) Install 1 or more PhaseSpace LED trackers to each plant and to the robot end effector.
5) Home all robot axes.
6) Record the positions of the end effector and plants as determined by the PhaseSpace system.

To pass the test, the positions of the plants as determined by the NILE system were required to be within +2 cm of the positions as determined by the PhaseSpace system.

To verify the ability of the system to determine the location of plant foliage, plant stems, and weeds with the consistency specified, we performed the following test procedure.

1) Configure a Python script to import 100 pre-classified images not previously introduced to the NILE machine learning-based texture analysis algorithm.
2) Command the object classification algorithm to make a prediction of the contents of image; each prediction will be compared to the ground-truth classification of each image.

To pass the test, the plant identification algorithm was required to correctly label no less than 99 of the 100 images presented to it.

To verify the ability of the system to perform the test and analyze the results within the specified time, we performed the following test procedure.

1) Command the system to perform a complete scan of the growing zone for plants and weeds. Record the time from the initial command to the publishing of the results on the web-application.

To pass the test, the total time from the initial command to the publishing of results had to be under 12 hours.

3.6.1 Plant Location Determination Results

Due to setbacks with website and kinematics integration, we were unable to perform the first test for plant location precision. Additionally, we failed the plant determination test because the classification accuracy obtained was lower than the benchmark. A total of 100 new images were fed into the trained neural network. Of these 100 images, 87 were classified correctly indicating an 87% identification accuracy. The 13 images incorrectly labelled contained irregular shadows and obscured angles, while all well-formatted image samples presented no issues to the neural network.

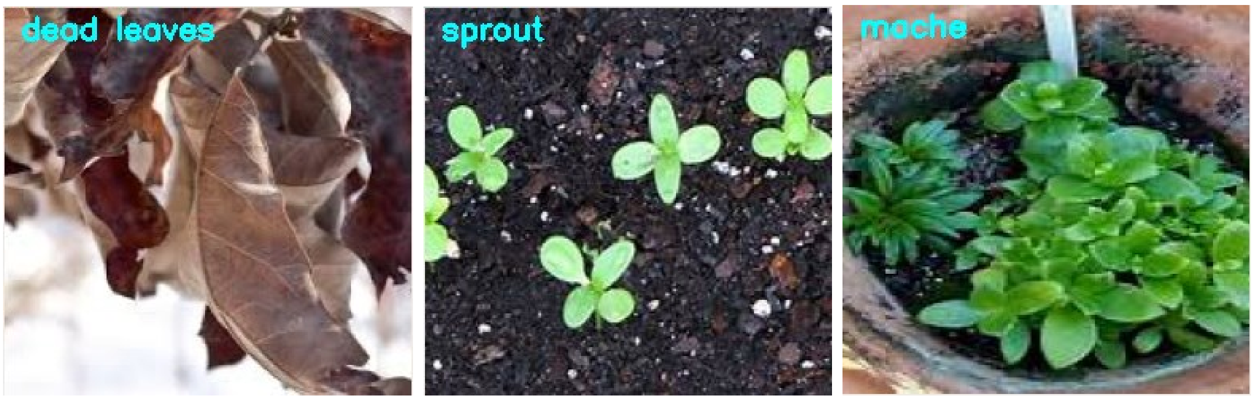


Figure 49. Successful Classifications

To test plant location prediction with purely computer vision, we supplied an image to the image processing software algorithm. In this test, the system was able to identify major plant foliage within the frame and create corresponding bounding boxes for each contour, as depicted below in Figure 50.

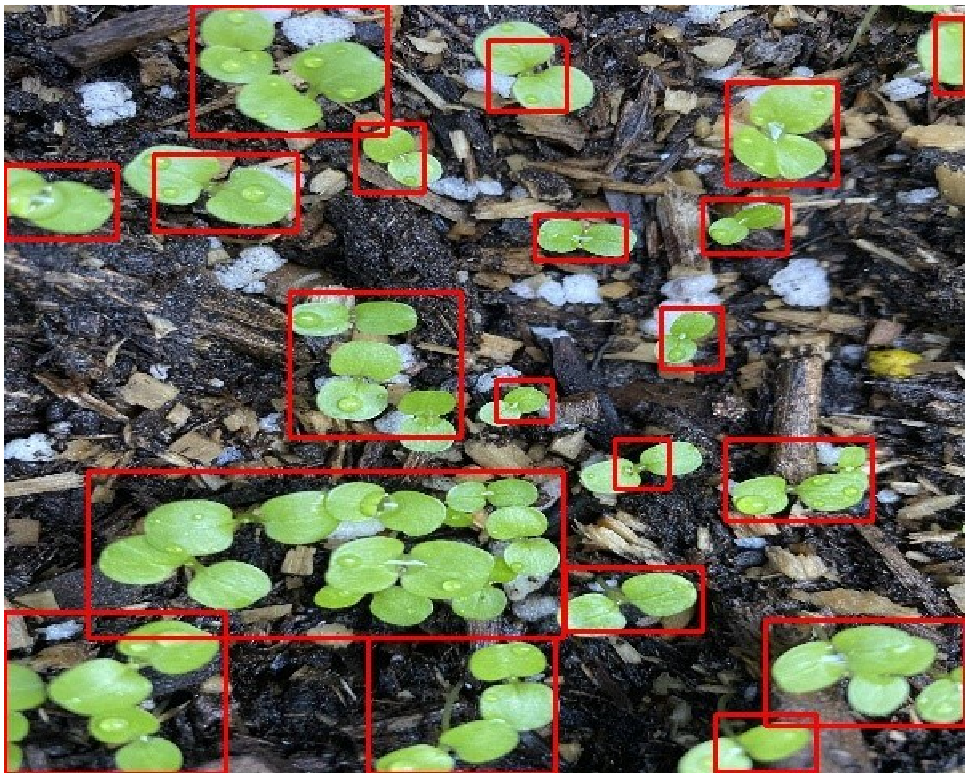
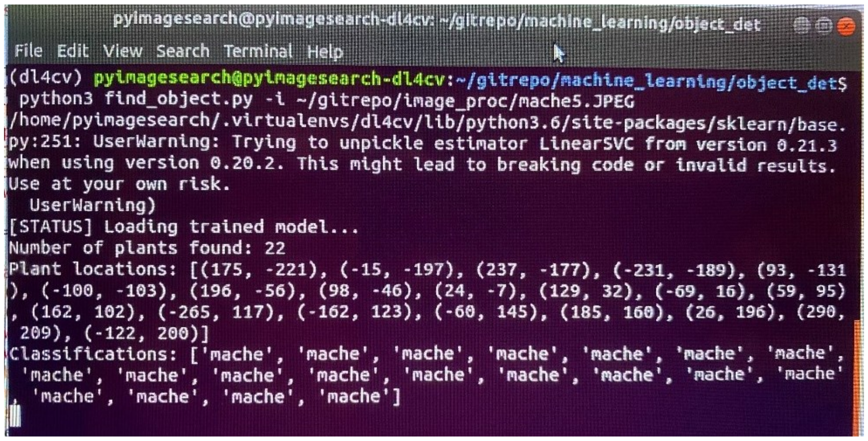


Figure 50. Identification of Plant Foliage

The centroid of each bounding box was computed with respect to the center of the image frame. Combining these values with the forward kinematics of the robot allows us to calculate task-space coordinates for each detected plant.


pyimagesearch@pyimagesearch-dl4cv: ~/gitrepo/machine_learning/object_det
File Edit View Search Terminal Help

```
(dl4cv) pyimagesearch@pyimagesearch-dl4cv:~/gitrepo/machine_learning/object_det$
 python3 find_object.py -i ~/gitrepo/image_proc/mache5.JPEG
/home/pyimagesearch/.virtualenvs/dl4cv/lib/python3.6/site-packages/sklearn/base.
py:251: UserWarning: Trying to unpickle estimator LinearSVC from version 0.21.3
when using version 0.20.2. This might lead to breaking code or invalid results.
Use at your own risk.
  UserWarning)
[STATUS] Loading trained model...
Number of plants found: 22
Plant locations: [(175, -221), (-15, -197), (237, -177), (-231, -189), (93, -131
), (-100, -103), (196, -56), (98, -46), (24, -7), (129, 32), (-69, 16), (59, 95)
, (162, 102), (-265, 117), (-162, 123), (-60, 145), (185, 160), (26, 196), (290,
 209), (-122, 200)]
Classifications: ['mache', 'mache', 'mache', 'mache', 'mache', 'mache', 'mache',
 'mache', 'mache', 'mache', 'mache', 'mache', 'mache', 'mache', 'mache', 'mache'
, 'mache', 'mache', 'mache', 'mache']
```

Figure 51. Localized Coordinates of Detected Plants

Due to complexity of algorithm development, we were not able to test the system's ability to distinguish between healthy plants from weeds. However, the steps performed in this section indicated a high level of promise for further development towards those goals.

3.7 Ingress Protection

To verify the ability of the system to withstand water and dust ingress as defined by IP55, we performed the following test procedure.

1) Let the system run in a powered, or sit in an unpowered state, for a minimum of 5 day/night cycles in an outdoor environment.
2) Spray the system with pressurized water from a nozzle (maximum outlet diameter of 6.3 mm) for a minimum of 5 minutes.
3) Operate the system for an additional day-night cycle.

To pass the test, the system was required remain operable for the duration of the test without cleaning or repairing due to water or dust ingress. During that time, the system was also required to take images of the growing zone, water plants, and reach any arbitrary point in the growing zone.

3.7.1 Ingress Protection Results

We successfully completed the ingress protection tests and verified the robot's ingress protection level. On April 6th the robot was placed outside in the Space Robotics Lab courtyard. As directed by the test, it remained outside while undergoing electrical and software development from April 6th to April 11th. During that time, another group was building a small boat in the courtyard which spread sawdust throughout the area and on the robot.

Then, on April 11th, the robot was sprayed with water from a 6.25 mm nozzle for 5 minutes as directed in the test procedures. See Figures 52 and 53 below which show the test in progress and the nozzle used.

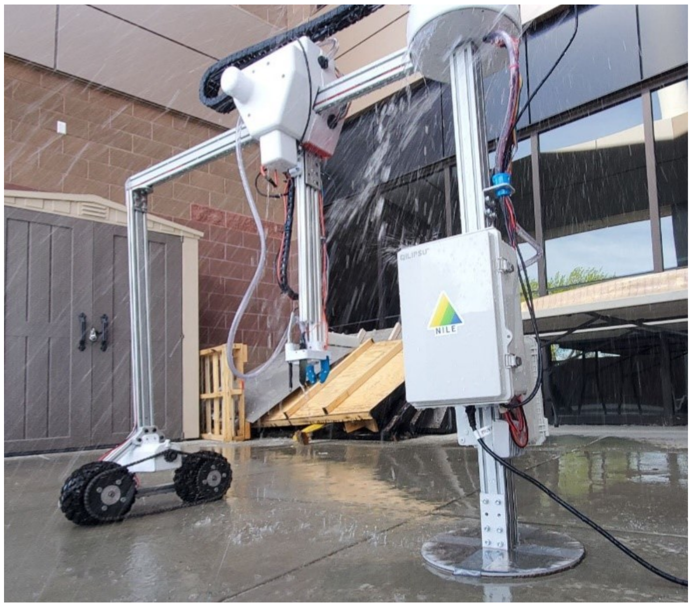


Figure 52. Ingress Protection Test

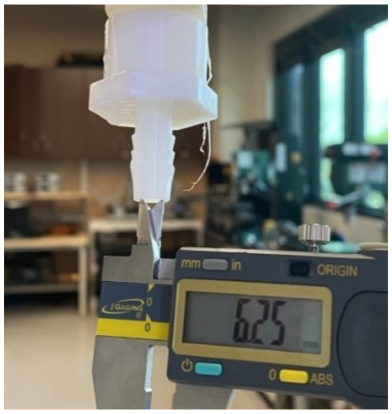


Figure 53. Ingress Protection Test Nozzle

After the robot dried following the test it was inspected for water damage inside the seals and tested for functionality. No signs of ingress or faults were found. Finally, the robot was left outside for an additional two days, after which no problems were found.

3.8 Communication

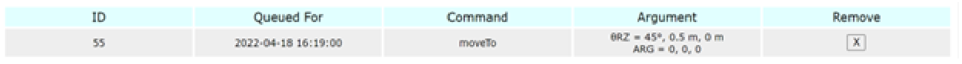
To verify the ability of the system to communicate the specified information to the end user, we performed the following test procedure.

1) Queue up a command for robot to move to a specific location in the growing zone via the website.
2) Queue a command for the robot to take a temperature measurement at that location and time how long it takes for the website to update with the reading with a stopwatch.
3) Create another future command and verify that you can remove the command immediately

To pass the test, the web application was required to display soil health statistics and the movement to a new location with live updates pushed to the website. Furthermore, the time between the temperature sensor reading starting and the update of the web-application had to be less than 60 seconds.

3.8.1 Communication Results

We successfully completed the communication test and verified the ability of the system to pass information and receive commands from the user. Communication with the robot was established with the admin login at the http://nilerobot.info web application. As directed in the test, an arbitrary *moveTo* command was sent via the *Schedule Command* table and subsequently added to a list of queued commands as shown in Figure 54.

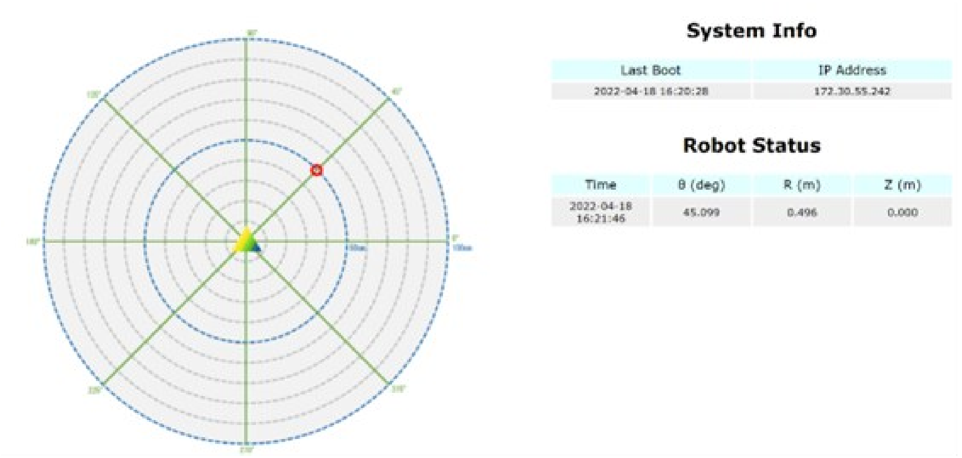


ID	Queued For	Command	Argument	Remove

55	2022-04-18 16:19:00	moveTo	θRZ = 45°, 0.5 m, 0 m
			ARG = 0, 0, 0
				X

Figure 54. Queued Command Table Entry

The robot proceeded to move to the desired coordinate with the robot's position, command, and sensor data updated every one second.



System Info

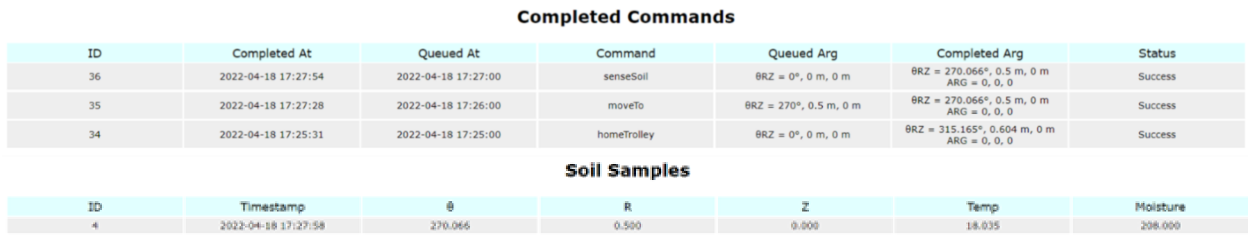
Last Boot	IP Address
2022-04-18 16:20:28	172.30.55.242

Robot Status

Time	θ (deg)	R (m)	Z (m)
2022-04-18 16:21:46	45.099	0.496	0.000

Figure 55. Website Interface Visualizing Movement of Robot.

The *Completed Commands* table shown below in Figure 56, displays the queuing and successful execution of the *homeTrolley*, *moveTo*, and *senseSoil* command sequence. Finally, the *senseSoil* command then added the measurement to the *Soil Samples* table as shown below in Figure 56.



Completed Commands

ID	Completed At	Queued At	Command	Queued Arg	Completed Arg	Status
36	2022-04-18 17:27:54	2022-04-18 17:27:00	senseSoil	θRZ = 0°, 0 m, 0 m	θRZ = 270.066°, 0.5 m, 0 m ARG = 0, 0, 0	Success
35	2022-04-18 17:27:28	2022-04-18 17:26:00	moveTo	θRZ = 270°, 0.5 m, 0 m	θRZ = 270.066°, 0.5 m, 0 m ARG = 0, 0, 0	Success
34	2022-04-18 17:25:31	2022-04-18 17:25:00	homeTrolley	θRZ = 0°, 0 m, 0 m	θRZ = 315.165°, 0.604 m, 0 m ARG = 0, 0, 0	Success

Soil Samples

ID	Timestamp	θ	R	Z	Temp	Moisture
4	2022-04-18 17:27:58	270.066	0.500	0.000	18.035	208.000

Figure 56. Soil Sensing Measurement with Coordinates

4.0 Project Management

Project management is the glue that holds teams together and ensures that all objectives are met. For the NILE project, project management was vital in keeping the program on track when faced with delays and integration difficulties. The following sections will discuss the overall team organization, project planning process, and budgeting.

4.1 Team Organization

To leverage the diverse talent within our team, NILE was split into three sub-teams: mechanical, electrical, and software. The mechanical sub-team focused on the computer-aided design, structural analysis, and physical assembly of the robotic chassis. The electrical team gave attention to the creation of the HVEC, developed the PCBs, and managed the hardware connections needed to interface the various peripherals. Meanwhile the software team was responsible for creating a custom software environment, developing unique computer vision algorithms, and implementing machine learning techniques for image processing.

4.2 Project Planning

To ensure that our initiative remained on target to completion, we created a series of weekly tasks and deadlines and held regular team check-up sessions. Table 4, below, highlights the objectives we outlined at the beginning of the detailed design phase.

Table 4. Detail Design Gantt Chart

Item/Week	1	2	3	4	5	6	7	8	9	10	11	12	13
Website Updates	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Gantt Chart Revisions	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Verify PCBs	🟩												
Verify ROS and CV comm.	🟩												
Purchasing	🟧	🟧	🟧	🟧	🟧	🟧							
Talk to Machine Shop	🟩	🟩											
Verify ROS abilities to IK		🟧	🟧	🟧									
PDR review of feedback + Descisions		🟩											
Web Development Plan		🟩											
Grow more Mache		🟩											
ROS IK trajectory simulation			🟧	🟧	🟧	🟧	🟧						
Custom parts fabrication complete			🟧	🟧	🟧	🟧							
Mechanical assembly			🟧	🟧	🟧	🟧	🟧	🟧	🟧				
Integrate PCBs			🟧	🟧	🟧	🟧	🟧	🟧	🟧				
Mechatronic system testing			🟧	🟧	🟧	🟧	🟧	🟧	🟧				
Test document draft						🟩	🟩						
Controller (close loop)						🟧	🟧	🟧					
Test positional accuracy							🟧	🟧	🟧	🟧			
Web development							🟧	🟧	🟧	🟧	🟧	🟧	
Mechatronics/Controller Tuning							🟧	🟧	🟧	🟧	🟧		
ML texture and shape analysis							🟧	🟧	🟧	🟧	🟧	🟧	🟧
ML plant classification							🟧	🟧	🟧	🟧	🟧	🟧	🟧
Test document										🟩			
Prepare Final Document											🟩	🟩	🟩
Prepare Final Presentation											🟩	🟩	🟩
Poster											🟩	🟩	🟩

The graphic above captures the progress of design about halfway into the semester. Over the course of 13-week period, the NILE team had to shift all mechanical assembly and mechatronic system testing to the right on the timeline. This was caused by increased shipping delays due to inclement weather and the impact of COVID-19 on the national supply chain. As a result, key fabrication tasks and full system integration were not completed until after week 13. This backlog also caused all objectives scheduled for week 10 and beyond to shift to week 13, causing tight time constraints for evaluating system performance. In the meantime, software development for machine learning and the architecture for the system website were accelerated to a status of 3 or more weeks ahead of schedule. This became crucial near the conclusion of the design process when we began integrating the website with system mechatronics and troubleshooting complex artificial intelligence nuances.

4.3 Budget

Budgeting has been a large part of the project management process from the beginning. As a six-person team we were allocated $900 and used slightly over that amount. Table 5 below outlines the major purchase requests submitted over the course of the design process.

Table 5. Updated Purchase Totals

Supplier	Subtotal
8020	$255.56
Adafruit	$33.80
Amazon	$307.03
STEPPERONLINE	$17.90
Digikey	$78.44
McMaster-Carr	$231.83
Total	$924.56

$924.56 nearly in budget but, it must be clear, does not account for the total cost of the system. We were able to take advantage of scavenged components and materials in addition to free machine shop time. However, now the path has been blazed possible future iterations of the system may be able to focus more on cost savings.

5.0 Conclusion

The proof of concept NILE system is on course to address the issues of wasteful watering and excessive fertilization. Our system has been developed with scalability in mind so that the technology we developed could be used to make a significant difference in the world. Over the course of the year our requirements have been adjusted as we gained experience and a better understanding of the problem. We found the capabilities of our design and the level of technology available and scaled our requirements accordingly.

Based on our experience building this prototype we have gained significant insight into what is necessary to develop a complete system and the time required to do so. Based on this, and our ultimate goal of creating a scalable system, if we were to start again, we would focus much more heavily on creating a self-contained trolley. As part of this, greater emphasis would be placed on making enclosures easier to work in, creating more robust electrical connections, and utilizing software engineering best practices during code development.

To any future continuation of the NILE project, never be afraid to pick the novel solution. Our weed elimination circuit may not have been the easiest solution but was highly effective and inspired many who heard about our project. A focus on the most innovative component, the trolley, and foregoing the less scalable aspects would benefit any student continuing the project. In addition, avoid having a homogenous team makeup and try to attract students from other colleges to avoid specialization while increasing diversity of talent.

6.0 Citations

[1] K. Smarsly, "Agricultural ecosystem monitoring based on autonomous sensor systems," 2013 Second International Conference on Agro-Geoinformatics (Agro-Geoinformatics), 2013, pp. 402-407, doi: 10.1109/Argo-Geoinformatics.2013.6621952.

[2] G. Merrington, D. Nfa, R. Parkinson, M. Redman, L. Winder, 2014. Agricultural Pollution. London: CRC Press.

[3] H. Durmus, E. O. Gunes, M. Krc, and B. B. Ustundag, “The design of general purpose autonomous agricultural mobile-robot: Agrobot,” in Agro-Geoinformatics (Agro-geoinformatics), 2015 Fourth International Conference on, July 2015, pp. 49–53.

[4] K. Kashiwazaki, Y. Sugahara, J. Iwasaki, K. Kosuge, S. Kumazawa, and T. Yamashita, “Greenhouse partner robot system,” in Robotics (ISR),2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK), June 2010, pp. 1–8.

[5] J. Tardaguila, “The vinerobot project,” http://www.vinerobot.eu/, 2014, accessed=2021-11-16.

[6] “FarmBot | Open-Source CNC Farming,” FarmBot. https://farm.bot/, accessed=2021-10-1.

[7] J. P. Gaus, “Robotic Gantry Bridge For Farming,” US Patent 9,622,398 B2, 18 Apr., 2017.

[8] K. Mahajan, P. Darunte, B Borase, R. Bodake, “Agricultural Robot for Plant Health Detection”, International Journal of Engineering Applied Sciences and Technology, 2019, pp 177-181, vol 4, issue 3, ISSN No 2455-2143.

[9] Budynas, R. G., Nisbett, J. K., & Shigley, J. E. (2011). Shigley's mechanical engineering design. New York: McGraw-Hill.

[10] L. F. P. S.l., “Dead Leaf Stock Photo. image of tree, nature, fall, cycle - 26824466,” *Dreamstime*. [Online]. Available: https://www.dreamstime.com/royalty-free-stock-image-dead-leaf-image26824466. [Accessed: 28-Apr-2022].

[11] A. Rosebrock, Deep learning for computer vision with python. Philadelphia, PA: Adrian Rosebrock, PyImageSearch.com, 2019.

7.0 Appendix A – Additional Resources

Please see the following sections for links to the GitHub repositories where the NILE documents are stored. All resources, information, and documents created by the NILE project are released into the public domain and are available for free and unencumbered. Go forth and create!

7.1 CAD and PCB Files

https://github.com/NILE-ERAU/NILE_Hardware

7.2 Main Software Repository

https://github.com/NILE-ERAU/gitrepo

7.2 Website Repository

https://github.com/NILE-ERAU/NILE_Website