Ceres

A Motorized System for Plant Root Research

Jimmy He Nachiket Joshi Nathaniel Keri

miaoh2 nsjoshi4 keri2

May 1, 2019

Abstract

Imaging plant roots at ACES is a required part of their research process and is currently a done manually, requiring 10 minutes of work for each plant by a talented individual. We seek to remove the manual work from this process to allow the researchers to direct their efforts towards analysis of the acquired data instead of wasting valuable time collecting the data. This document describes a motorized imaging system that will take photos of a tall plant's root system. This will be done by transporting a camera down a standard transparent observation tube into to the soil, taking pictures as it goes. These photographs will then be used for scientific research in the field of agriculture. The imaging device is comprised of a base station on the tube's end resembling a hoist, and a suspended camera placed in the observation tube. A central control server serves as controller, collecting images from a fleet of devices, and presents a GUI to the user with live progress and diagnostics data from each device.

Contents

1 Introduction

1.1 Background

Our senior design project was the creation of a device that automates a current process used in research. The SoyFACE Farm has a cornfield they use to conduct research on the health of plants by imaging their roots [1]. There are thousands of plants in the field, and those plants are accompanied with thousands of clear tubes that penetrate the earth and are angled such that the plants' roots grow around the clear tube. When the roots are imaged, the current process is to send a camera down the tube with a pole, and manually pull up the camera a few centimeters at a time as it takes images of the roots.

Our device instead automatically lowers a camera into the tube, takes pictures on its way, and then compiles and sends the images to a database so the researchers can then analyze them as needed. This way, an extremely large amount of time and effort is saved on behalf of the researchers, freeing them up to do more important tasks instead of what is essentially manual labor. To create this system, nicknamed Ceres, engineering efforts from multiple disciplines were required. The physical assembly needed to be mechanically designed, including the design of the component housings and the mechanisms for camera transport and movement. The electronics needed to be mapped out, linking the camera and motor to the wireless module to be able to communicate the image information. Finally, the software backend that collects, sorts, and stores the images had to be created, as well as the software front end that the users interface with to instruct the device to commence and end operation.

1.2 Block Diagram

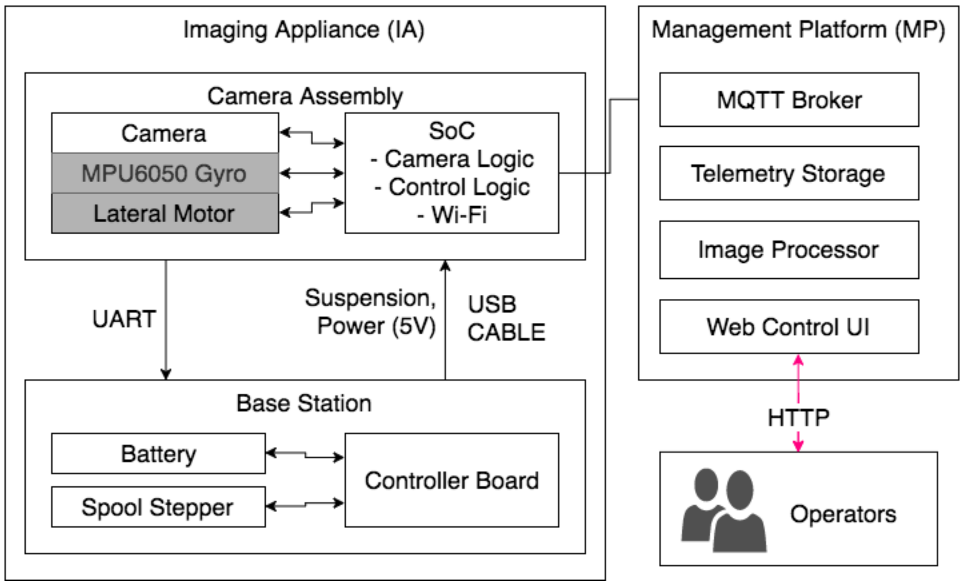

Figure 1: Block Diagram

1.3 Functional Overview

Ceres can be split into two components, a Management Platform (MP) and an Imaging Appliance (IA). The IA can be further split into the Camera Assembly and the Base Station. The following discussion about the components is centered around the block diagram.

The main logic of IA runs on top of an ESP32 SoC inside the camera assembly. It accepts control signal from the operator from the MP, through a MQTT connection over its built-in Wi-Fi. It also has a serial (RS-485) link to the control board in the Base Station, in order to receive battery condition data and transmit signals to control the hoist motor, which in turn manipulates the physical depth of the camera in the tube. The ESP32 SoC is connected to a OV2640 camera, that captures a 2 mega-pixel image at every run. It is fixed-focus since the distance from the camera to the plant root is known. When imaging, the camera first travels up by a fixed distance (2cm) such that the view overlaps with the previous, and takes a image. It then compresses the image into JPEG and transmits it to the MP.

All these functions will be performed autonomously after the system is placed in the observation tube. Additionally, the nature of the MP allows many Ceres systems to communicate with the Management Platform, making the system scalable enough to allow large-scale experiments to be run using the devices. The easily swappable batteries will also contribute to scaling, as the decreased time to reset the devices will add up to less overall time spent maintaining the devices in the field. Finally, because the device is constructed using 3D printing techniques and low cost electrical components, its cost is very low, especially when considering acquiring units in bulk, which would further lower the price.

1.4 Block Requirements

The base station (BS) of the IA is placed above ground, mounted on top of the observation tube. It contains a battery to power the entire IA, and a stepper motor that hoists the camera assembly (CA, hereafter) into the tube. The CA sends a control signal in terms of distance to move, and the ATmega328 MCU in the base station translates it into angular motion before commanding the stepper motor. It also has an endstop switch to indicate the home ($Z = 0$) point. The MCU is also connected to a battery charging / monitoring IC that can charge the battery and report battery level to the camera assembly, such that it can home itself and refuse imaging when the battery level drops too low.

The MP runs on a physical machine of any platform (such as an x86 Linux server) and optionally also act as the Wi-Fi AP for each IA to connect to. The heart of the MP is a MQTT broker collecting telemetry and images from and emitting control signals to one or more IA'ds. Upon reception of images, it processes the image set and stores a panoramic image into its image storage. Upon reception of telemetry, it stores the telemetry in a volatile database for tracking. It exposes a Web UI to the frontend users, such that the users can view and download the images, as well as monitor and control the IA's. When used, the operator enters the name of the image (matching the current date and ID of the observation tube) and presses "Start", starting the automatic imaging sequence of the selected IA. The operator then monitors the Z depth of the camera in real time as an indicator of the imaging process. When done, the operator may download the set of images taken by the camera.

1.5 Performance Requirements

Throughout the creation of Ceres, we followed the guidelines of three main high level requirements:

- The system must be autonomous, performing all main functions without human intervention.
- The system must be scalable, allowing a minimum of 5 imaging appliances to communicate with the management platform in parallel, while being managed by an operator.
- The system must be as low cost as possible, with a target cost of $100 maximum for each imaging appliance.

2 Design

2.1 Procedure

This project adopts the Divide and Conquor strategy in the design of the system and individual components. As the overall block diagram has been established, decisions are made on the inter-communication protocol between each component. As such, each module of the system is individually designed, and the choice of technology and development technique does not impact that of the rest.

2.2 Details

2.2.1 Camera Assembly (CA)

The Camera Assembly serves the purpose of controlling the image-taking logic, coordinating the mechanical movement via the Base Station, as well as capturing and transmitting images from itself. Its PCB was designed around the width constraint of being able to fit in a tube as well as thermal efficiency required for continuous operation in a closed environment.

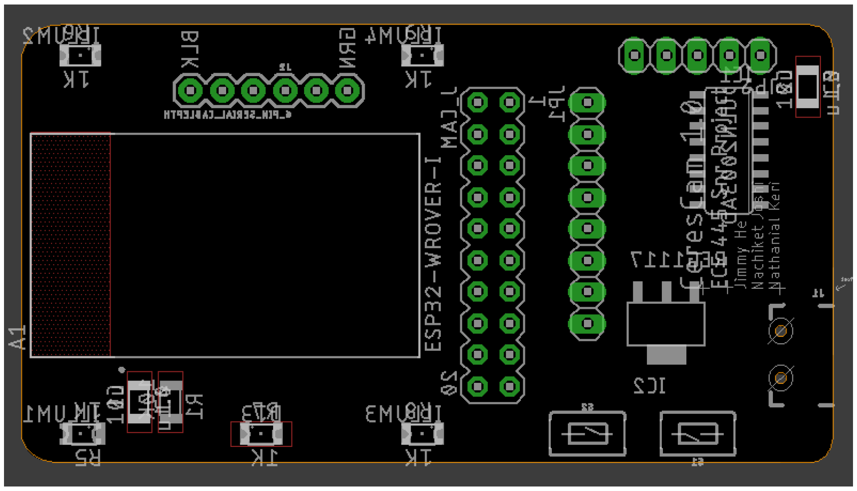

Figure 2: Camera Assembly PCB Drawing

As shown in the PCB Design, the camera assembly uses an ESP32 System-on-Module (SoM) as its primary logic controller. The ESP32 platform was chosen because it is the one of the only available SoM's with built-in Wi-Fi radio, as well as with the GPIO pins and processing power to handle real-time processing of digital images. The choice of the camera module is OV2640, a 200 megapixel sensor coupled with a manual focus lens tuned to a 5cm focal length (the diameter of the tube). The CA communicates with the BS using Serial protocol over a Micro USB connector, leveraging the widely available section of USB cables off the shelf. The UART communication reuses the D+ and D- pins of the cable, as defined in the following pinout table.

USB Pin	Pin Definition	Usage
1	VCC	5V Power
2	BRCT	UART from CA to BS (Tx on CA, Rx on BS)
3	BTCR	UART from BS to CA (Tx on BS, Rx on CA)
4	GND	Ground

Table 1: Pinout Table for the USB Port

The CA firmware is developed around FreeRTOS as the scheduler for multitasking, with ESP-IDF, a vendor-supplied Software Development Kit (SDK) as the Board Support Package. Main features of the firmware includes sending system status packets, receiving control commands from the operator, and orchestrating the imaging sequence as follows:

- Home the Camera
- Do the following until reaching the bottom or user issues cancel command

    Command the BS to lower the camera by 1cm

    Capture an image and signal to the server

    Wait for the server to download the image, and repeat

- After completion or cancellation, home the camera

2.2.2 Base Station (BS)

The Base Station is an passive, auxiliary fixture to the Camera Assembly that provides battery and drive power as a motorized hoist sitting on the top of the observation tube. It adheres to a instruction set defined alongside the CA to move the motor and report battery conditions.

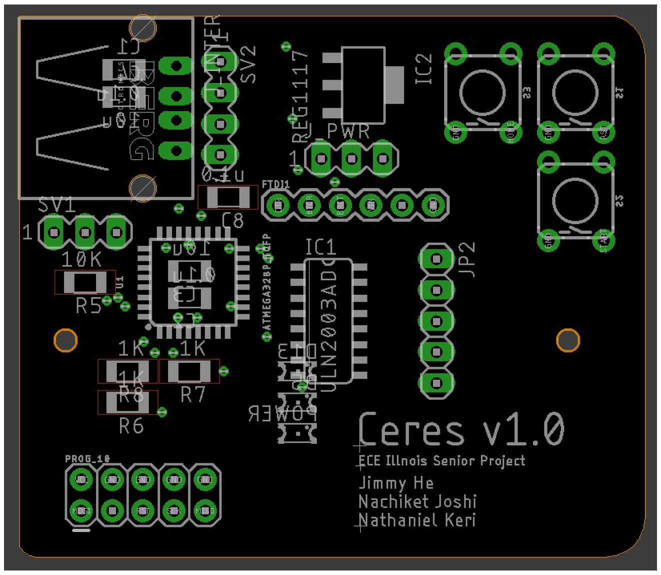

Figure 3: Base Station PCB Drawing

As shown in the schematics drawing, the Base Station is powered by an ATmega328P Microcontroller. It relies on an off-the-shelf powerbank charging circuit to supply the required 5V power, and uses an AMS1117 linear regulator to step it down to 3.3V logic voltage of the microcontroller. The battery voltage is passed through into the Base Station's analog input pin so the battery voltage can be measured. A ULN2003 Darlington Array serves the purpose of a stepper driver, and a 3-pin connector interfaces the microcontroller with the endstop switch, which signals whether the camera asembly is properly homed.

2.2.3 Mechanical Construction

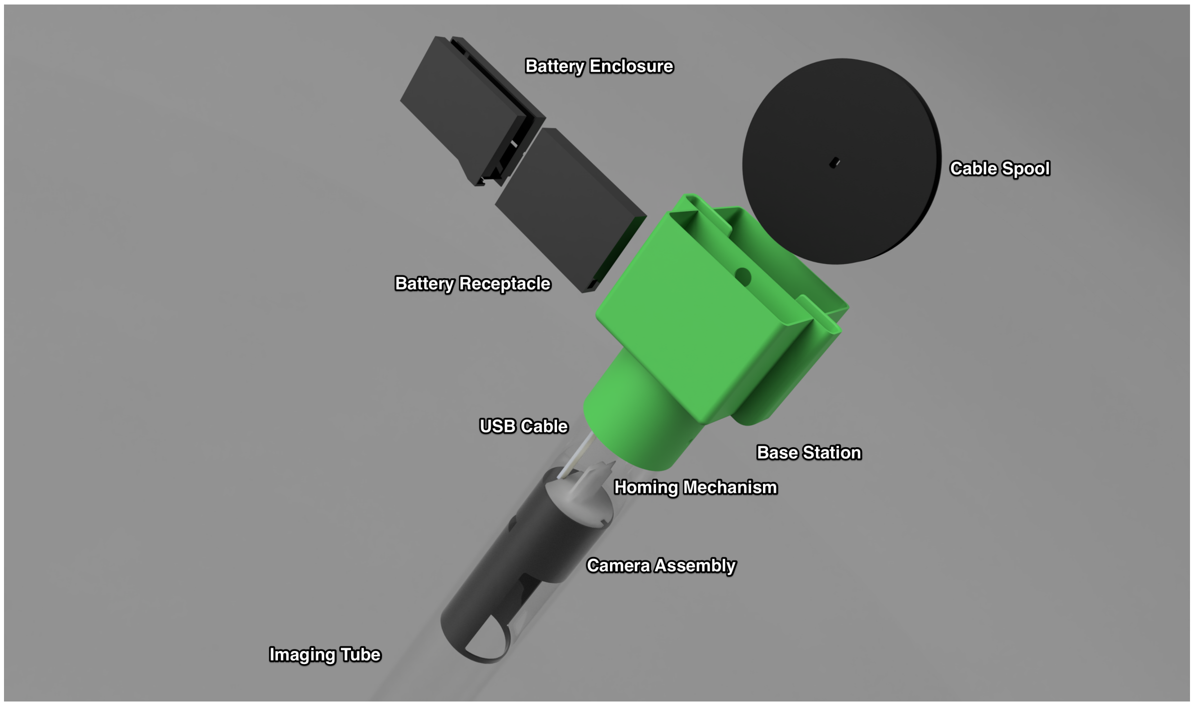

Figure 4: Mechanical Assembly Diagram

The assembly of the Imaging Appliance can be broken down as shown in the figure. The BS is mounted on top of the observation tube. A spooled USB cable connects between the CA and BS, and provides mechanical suspension of the BS. At standby, the CA is lifted up into the receptacle of the BS. In operation, the BS rotates the motor to unwind its USB cabel, lowering the CA into the tube. A removable battery shown on the left, which consists of a 2000mAh lithium ion batter fit in a pair of plastic clamshell-shaped enclosure, can be plugged into the Base Station along a plastic guide rail.

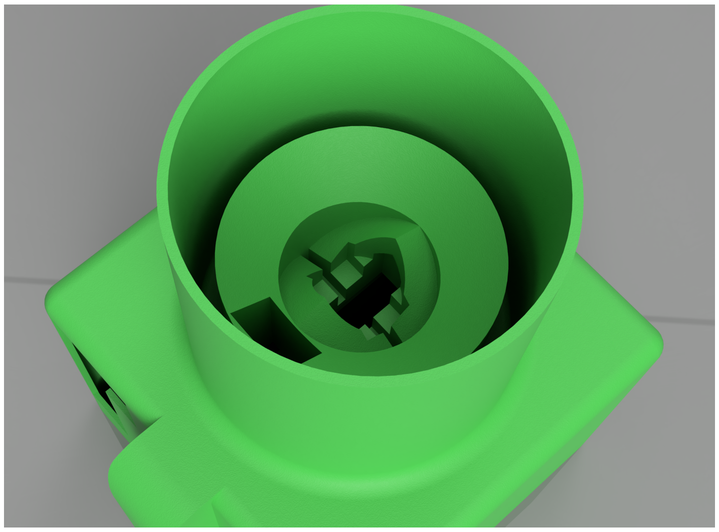

Figure 5: Base Station Homing Mechanism

Most importantly, the receptable contains a mating mechanism (as shown in the figure) that corrects the CA's orientation during homing, allowing the CA to enter even with a large deviation. Such is achieved with a circular ramp on the BS and a flat fin on the CA, which slides along the ramp into the final mating slot and triggers the endstop switch.

2.2.4 Image-Stitching algorithm

The Image Stitching algorithm, as a separately designed component that compiles all images taken along the plant root into a single panorama, is developed using Python and OpenCV. The algorithm invokes OpenCV's Stitching API against a list of images, and outputs a complete image as the result. If OpenCV fails to stitch, the stitching algorithm returns a status code and no image, to the program that requests stitching.

2.2.5 Server-side Software

The Server-side software is a collection of software applications consisting of the HTTP server, MQTT server, MQTT broker and the database. Ceres uses already-built Open Source MQTT broker Eclipse and the PostgreSQL database. Together, the server-side software controls and monitors Ceres cameras, as well as receives, stores and manages the images transmitted by the cameras. In accordance to the protocol in the appendix, the MQTT server receives telemetry and images from the camera, and invokes REST API calls to the HTTP Server. The HTTP server manages the metadata of all images and the status of all cameras in accordance to its database. The images are stored in the file system and the The aforementioned interactions are modeled in the following dataflow diagram.

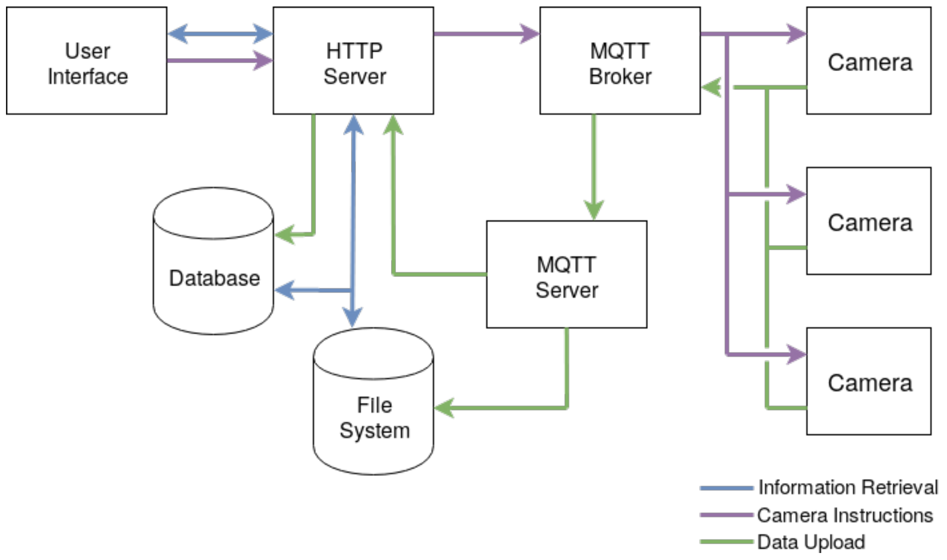

Figure 6: Server-side Software Datapath

In production, the Server-side software is packaged and delivered using Docker, a containerized platform that guarantees identical runtime behavior across different computer environments. This greatly streamlines the deployment of the system, as a complete instance of the software can be launched from a single automated command.

3 Verification

3.1 Verification Overview

During the initial design of the project, we developed the requirements and verification table in appendix A. Each unit can be individually tested in isolation as well as tested underneath a mock testing framework. This mock testing topology can be seen in figure 7.

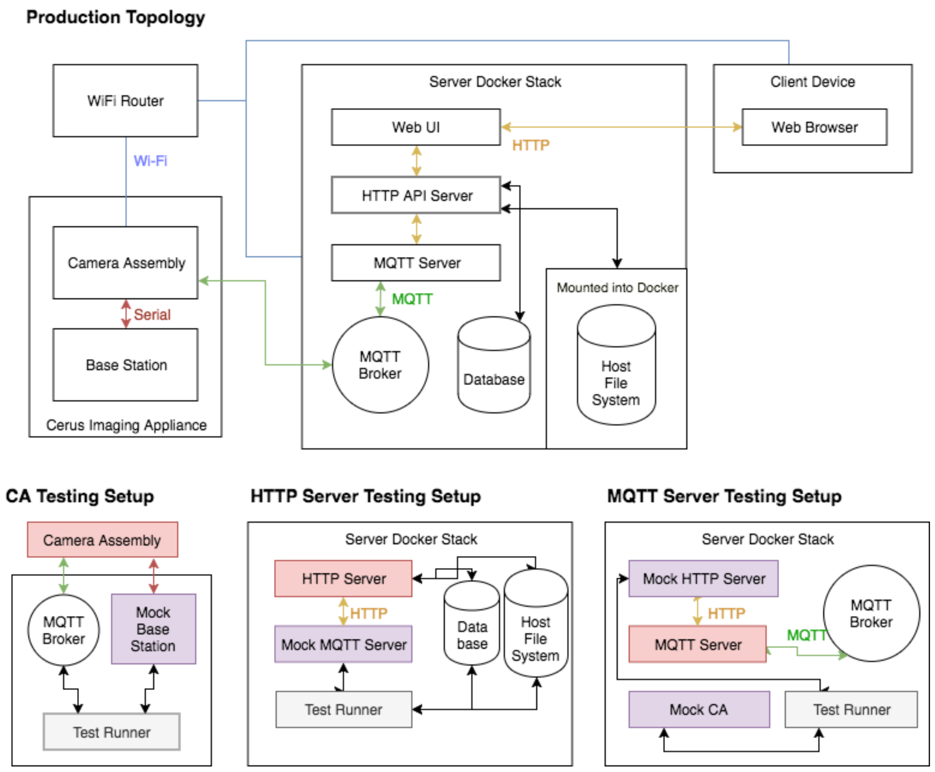

Figure 7: Test Topology Architecture

By using this testing topology, we were able to iterate through the design independently as well as test each component as it was produced. Appendix A also contains the explicit testing steps for each major component of Ceres.

3.2 Camera Assembly PCB

The verification of the camera assembly consisted of unit testing each of the commands in isolation to ensure that each base instruction was consistent. Each unit test ensured that the camera would receive and send the appropriate data to and from the MQTT broker as well as invoke motor-driving functions by sending instructions to the Base Station PCB.

3.3 Base Station PCB

The base station was also tested in isolation. The primary concerns of the verification was to make sure that the base station operated within the requirements laid out in Appendix A.1. Other concerns were the motor controls. We had to ensure that the Base Station PCB did not run the Base Station motor too hot, otherwise the glue that we had placed would melt and the motor would come loose from its position. This is done by manually asserting the motor pins to a high logic level when the driver is not in control of themotor, so there is no static current through the motor.

3.4 Mechanical Design

The mechanical design was tested primarily through trial and error. We first tested the design of the Camera Assembly and Base Station with respect to the homing function. Once the homing function had been finalized, we did not need to alter that portion of the design any further. As we started assembling the physical components together, as we encountered problems with our design, we would alter either the Base Station or the Camera Assembly as needed. Given that the majority of our mechanical components were primarily 3-D printed, which gave us a relatively quick iteration time. This allowed us to be able to test a variety of solutions as the situation required.

3.5 MQTT and HTTP Servers

Testing of the Management Platform, of which the MQTT server and HTTP server are an integral part, was primarily done via mock testing.

The MQTT server verification consisted of manually testing how the MQTT passed image data and various metadata to the appropriate destinations. The metadata was manually inspected to see if they were being posted to the appropriate HTTP server API endpoints for further processing. We inspected the file system manually as the image data was being stored to make sure that the image was being stored appropriately and correctly.

As for the HTTP server, we tested each endpoint manually. This primarily consisted of checking the database retrieval functions, ensuring that each endpoint acted appropriately given a variety of starting conditions. Once the database functions were considered to be correct, we then had to make sure that the file serving was serving both the correct contents, as well as the correct file type as well. The instruction API of the HTTP server was tested via inspection as well, making sure that the appropriate instruction was posted onto the MQTT broker.

3.6 Integration

Integration testing consisted of automated runs of the Management Platform and Imaging Assembly in concert. This testing primarily focused on the functions that would be used most often in the field, and therefore focused on repeatability of the whole process and durability of the individual components through repeated use. Through the verification process, we were able to identify areas we could either simplify or leave for future work on this project, as well as issues in the testing environment itself.

4 Costs

Table 2: BOM Per Set of Equipment - Camera Assembly and Base Station

Name	Quantity	Unit Cost	Total Cost	Large Batch Price
40-pin Header	3	$0.17	$0.51	$0.51
Blue LED	9	$0.34	$3.02	$1.00
Tactile Switches (PTH_6.0MM)	3	$0.10	$0.30	$0.30
Tactile Switches (SMD_6.0×3.5MM)	2	$0.30	$0.60	$0.60
USB Type A Connector	1	$1.11	$1.11	$0.30
0.1u C-USC1206	5	$0.66	$3.30	$1.00
10K R-USC1206	1	$0.09	$0.09	$0.05
10u C-USC1206	4	$0.71	$2.84	$0.92
1K Resistor	9	$0.05	$0.45	$0.24
ATmega328P Microcontroller	1	$2.04	$2.04	$1.50
AMS1117 Regulator	2	$0.44	$0.88	$0.20
ULN20003AD Darlington Array	1	$0.56	$0.56	$0.30
MicroUSB Connector	1	$0.70	$0.70	$0.40
OV2640 Module	1	$10.49	$10.49	$10.49
MPU6050 Module	1	$5.79	$5.79	$1.20
ESP32-WROVER-B Module	1	$5.00	$5.00	$5.00
2000mAh Adafruit Li-Ion Battery	1	$12.50	$12.50	$9.50
3D Printing Material Cost	1	$5.00	$5.00	$4.30
Total Cost			$55.18	$37.81

Large Batch Pricing is the per-unit price when manufactured in a batch of 500 units

Table 3: Labor Cost Estimate

Item	Hours	Hourly Pay	Gross Pay
Development Cost	80 hours	$25/hr	$2000
Testing Cost	30 hours	$25/hr	$750
Assembly Cost	2 hours	$15/hr	$30

5 Conclusions

5.1 Summary

Our group was able to successfully construct Ceres, a device that automatically images plant roots and sends pictures back to a database for further analysis. Using our group's knowledge in disciplines such as mechanical design, software engineering, and integration engineering, we assembled the device and carried out a demo of its functions.

5.2 Successes

The successful completion of our goal can be attributed to many successes and decisions, but the most potent characteristic of our group that led to success was our ability to work orthogonally with each other. Because of the independent nature of the unit tests employed during the testing phase, each group member was able to accomplish their work almost in its entirety before needing to use another's completed work. This resulted in quick and smooth testing, maximizing the efficiency of our time this semester.

5.3 Challenges and Further Work

On the path to success, our group encountered many challenges that we overcame to make sure Ceres was functional. A few non-vital challenges remain that we plan to conquer using methods presented henceforth. One such challenge was the failure of our stepper motor to produce enough torque to trigger the mechanical endstop switch used in the Base Station. This leads to the Base Station not recieving a completed homing signal when imaging is completed. To fix this issue, we'll simply replace the mechanical switch with an optical one that would require no additional force to trigger. These optical switches are the same price if not cheaper than mechanical switches, so no additional problems are foreseen.

Another challenge we ran into was the failure of our image stitching algorithm to assemble the taken images into a panorama. Although the researchers at SoyFACE don't currently use panoramas in their process, we'd like to get this feature to work in case it helps them. To do this, we will conduct image calibration on the taken images to remove distortion and stitch the resulting images together.

While we were testing our completed device, we noticed that the imaging process took longer than theoretically calculated, and diagnosed the issue to be due to network congestion of our testing environment. In case this will be an issue in real world scenarios, we will adjust our imaging process to first cache all the image data as the pictures are taken, and then send everything at once. This will remove imaging delays due to network speed, and quicken the process.

Lastly, since some features and functions of our PCBs were deemed unnecessary (such as our lateral motion motor), there is room for improvement in terms of simplification. Once the PCB designs are simplified they can be built to be smaller and lighter, leading to performance improvements for the hoisting mechanism.

5.4 Ethics

According to the ACM Code of Ethics [2], "A computing professional should... Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing." Our project contributes to the research that the SoyFACE farm is conducting, safely contributing to society. This code of ethics also states that a computing professional should "Be honest and trustworthy." We have made sure our system works well and so gives the researchers correct data as they are conducting analysis on the plants. Finally, the ethics state that we should "Strive to achieve high quality in both the processes and products of professional work." We have done our utmost to create the best possible product for the researchers to be able to use, and I believe in many respects we've either achieved that goal or have concrete plans to succeed.

A Design Document Requirements and Verifications

A.1 Imaging Appliance - Base Station

Requirement	Verification
The Base Station performs HOME.	1. Send H over serial. 2. Check that Base Station begins motion and stops when Camera Assembly is fully homed. 3. Expect K received over serial.
The Base Station knows that a device is already HOME'ed.	1. Send H over serial. 2. Expect K received over serial immediately, and no motion has occurred over the motor.
The Base Station performs a STEP as per the spec.	1. Send S over serial. 2. Expect Base Station to begin motor motion downwards. 3. Expect Base Station motor motion to eventually terminate and receive K when it happens.
The Base Station performs STEP repeatably, with a distance error of at most 30%.	1. Place a ruler along the tube the camera travels down. 2. Send multiple consecutive S commands, each waiting for the motion to complete and a K return to be issued.
The Base Station returns E for STEP when step count exceeds limit.	1. Set a hypothetical limit (10) in software. 2. Send 11S's over serial. 3. Expect only the first 10 commands to result in motion and a K return. 4. Expect the 11th command to result in no motion and an E return.

Requirement	Verification
The Base Station reports the correct number of STEPs when received COUNT	1. Randomly assign an integer N ¡= 10. 2. Issue N S commands, each waiting for the previous motion to complete. 3. At the completion of the Nth command, issue a C command. 4. Expect to receive an integer over serial, and the integer is equal to N
The Base Station clears COUNT when HOME is issued.	1. Perform the above verification, and issue an H command. 2. Wait for motion to complete. 3. Issue a C command. 4. Expect to receive 0 over serial
The Base Station reports the correct voltage for POWER.	1. Disconnect the battery, and place a voltage-dividing potentiometer that ranges from 0.0V to 3.3V. 2. Set the potentiometer to a fixed value, and read the voltage from a voltmeter. 3. Issue a P command and expect a floating-point integer to be returned on serial with a newline character. 4. Expect this number to match the voltmeter reading.
The Base Station ignores commands when in motion, except the X command.	1. Issue multiple S commands to Base Station in rapid succession, before the motion resulted from the first S command had stopped. 2. Record the distance moved as D1. 3. Issue a second S command, wait for motion to end and record the distance moved as D2. 4. Expect D1 to be within 20% of D2. 5. Issue an S command again to start the motion. 6. Issue an X command right after. 7. The motion should stop immediately, and a K will be received.

Requirement	**Verification**
After getting the X command, the Base Station ignores all other commands until successfully homed.	1. Issue an S command. 2. While in motion, issue X. 3. Expect to receive a K and motion stops. 4. Now issue S multiple times, expect E return and no motion each time. 5. Issue H and expect the Base Station to home the Camera Assembly. 6. Once fully homed and K received, issue S commands and the Base Station should generate motions accordingly.
The power supply provides a stable 5V voltage rail with an accuracy of 0.3V.	1. Measure the 5V rail using a voltmeter, from the VCC and GND lines of the interconnect between the Base Station and the Camera Assembly. 2. Expect the reading to be between 5.3V and 4.7V. 3. Repeat the measurement with lithium battery cells measured at 3.0V, 3.7V and 4.2V.

A.2 Imaging Appliance - Camera Assembly

Requirement	**Verification**
Accept an IMAGE command over MQTT and start executing a sequence to collect images.	1. Send an IMAGE command over MQTT to the SoC and observe the following sequence. 2. The verification of the sequence can be done by hooking the Camera Assembly directly to a computer over a serial connection.
During image collection, periodically transmit status information over MQTT. The status information contains battery voltage level, that it is imaging (not idle) and the number of steps so far traversed.	1. After the "Image" command, observe from the MQTT broker that voltage level, the "imaging" status and number of steps are correctly reported.
Before imaging, issues a HOME command to the Base Station at the beginning of the collection sequence and waits for a K response for confirmation.	1. Observe the outputs over the CA's serial port. 2. Expect an H from this port, and no further outputs should be expected until a K response is manually sent to the CA.
Before each image is taken, receive a K response.	1. Observe the outputs on the serial port. 2. Expect an S.
After an image is taken, send the image via MQTT and then receive the next command.	1. The image should be transmitted over MQTT. 2. Receive the next instruction on the serial port.
After imaging is complete, issue a HOME command again to be transported back to home.	1. Observe the serial port, expect an H command. 2. When a K response is manually given, expect the status information sent over MQTT to denote that the unit is idle.

A.3 Management Platform - Telemetry Processor

Requirement	Verification
Track the current status of the Imaging Appliance.	1. Connect a mock Imaging Appliance to the MQTT Broker. 2. Periodically send status packets with mock data. 3. Check from the Management Platform that each change is correctly indicated. 4. Expect that the Management Platform indicates the Imaging Appliance is online. 5. Disconnect the mock Imaging Appliance from the MQTT broker. 6. Expect that after 30 seconds, the Management Platform indicates the Imaging Appliance is offline.
Display the current status of the Imaging Appliance.	1. Present the information received from the MQTT broker on a webpage. 2. Expect that the data received from the webpage is the same as the underlying status packets.
Send imaging and cancellation command to the Imaging Appliance.	1. Connect a mock Imaging Appliance to the MQTT Broker. 2. Periodically send out status packets to indicate to the Management Platform that the Imaging Appliance is online. 3. Expect that the Start button for the Imaging Appliance on the Management Platform is now operable. 4. Click the Start button and expect that the Imaging Appliance receives a command over MQTT to start imaging. 5. Periodically send out status packets to indicate to the Management Platform that the Imaging Appliance is imaging. 6. Expect that the Management Platform indicates the progress of the Imaging Appliance. 7. Click on the Stop button for the Imaging Appliance on the Management Platform and expect the Imaging Appliance receives the Cancel command from the MQTT broker.

A.4 Management Platform - Image Processor

Requirement	**Verification**
Receive images fro the Imaging Appliance over MQTT.	1. Connect a mock Imaging Appliance to the MQTT broker. 2. Send over an image with the same format taken by a real Imaging Appliance by the same protocol. 3. Check that the Management Platform's log indicate a successful receiving of the image. 4. Check that the image appears under the file system that the Management Platform runs on.
Stitch a batch of images into a panorama.	1. Prepare a suite of images captured from a long object and pre-place them in the file system and database. 2. Begin the stitching process. 3. Expect that the returned image is a valid stitched image, and it contains the unique geometry of each test image in the batch.
Serve the stitched image or zip of the raw images over HTTP as a file download.	1. Complete the above verifications. 2. Expect the test run conducted appears in the Web UI of the Management Platform. 3. Click on the corresponding buttons to start downloading the zipped file and the stitched image. 4. Expect that the files are downloaded from the server.

References

[1] Sharon B. Gray, Reid S. Strellner, Kannan K. Puthuval, et al. “Minirhizotron imaging reveals that nodulation of field-grown soybean is enhanced by free-air CO2 enrichment only when combined with drought stress”. In: *Functional Plant Biology* 40.2 (2013), p. 137. DOI: `10.1071/fp12044`.

[2] ACM. *ACM Code of Ethics and Professional Conduct*. URL: `https://www.acm.org/code-of-ethics`.