Graduating Gardeners

Author: Hiroko Abe, Sarah Jang, Kanon Kihara:
Electrical and Computer Engineering, Carnegie Mellon University

***Abstract* — An automatic greenhouse system capable of maintaining specific temperature, lighting, and watering conditions, as well as detecting plant growth status and defects with an accuracy of over 90%. The greenhouse is connected to an interactive web application where users can receive alerts if any changes occur to the plants, monitor their plants live, and manually control environmental variables in a timely fashion, where the greenhouse environment will be adjusted to the new settings within an hour. Compared to complex and expensive industrial greenhouse systems, we aim to provide an intuitive, compact, low-cost, and effective greenhouse system for new gardeners**.

***Index Terms* — Automation, Computer vision (CV), Edge Detection, Greenhouse, HSV Color Detection, Microcontroller, NumPy, OpenCV, Web application**

I. Introduction

The demand for gardening supplies has increased significantly and caused frequent supply shortages throughout 2020 as more and more people started gardening for the first time during the Covid-19 pandemic. For many who are new to gardening, creating a gardening environment at home could be an expensive and complicated task. Growing plants that have specific environmental needs is also challenging for beginners. Greenhouse technology has been around for a very long time, and there are many greenhouses on the market that have advanced features to increase environmental sustainability, plant growth, and productivity. However, these systems are intended for large-scale industrial usage, which require a lot of investment in infrastructure and specialized knowledge to operate. We are planning to implement some of these features that help optimize plant growth on a small scale greenhouse system with a user friendly website in order to cater towards normal people who want to enjoy gardening at home.

Our greenhouse system will automatically maintain specific temperature, lighting, and watering conditions, and alert users of plant growth status and defects. The greenhouse will be connected to an interactive web application where users can receive the alerts, monitor their plants live, and manually control environmental variables. With the assumption that this greenhouse system will be used to grow common household plants, our goal is to build a greenhouse that is able to maintain a target soil moisture percentage with a 5% room of error, a target temperature with a 5°F room of error, and provide the plants with a target of 4 hours of light everyday. The system should accurately detect plant growth stages and defects with an accuracy greater than 90%. When a user changes the target temperature, soil moisture, or light duration on the web application, the new target should be met within an hour of the command. The web application UI will be intuitive and user friendly to provide the user with an effortless, yet successful gardening experience.

II. Design Requirements

In order to set specific, quantitative requirements for our project, we have decided to focus the scope of our greenhouse to automatically grow pea shoots. They are relatively easy and only take about 2 to 3 weeks to fully grow, which will give us enough time to test the greenhouse with a full growth cycle of the pea shoots. Based on specific growing conditions of pea shoots, we are requiring our greenhouse to be able to maintain a target soil moisture percentage (volumetric water content) with a 5% room of error, in order to ensure that the soil moisture percentage will stay well in between the field capacity and permanent wilting point. We are also requiring our greenhouse to be able to maintain a minimum target temperature with a 5°F room of error, since pea shoots should ideally be grown in temperatures between 55°F and 65°F. By setting the minimum target temperature to 60°F, we will ensure that the ideal temperature range is maintained within the greenhouse even during the winter. The greenhouse should be able to provide a target amount of light to the plants (in hours/day) if the greenhouse is placed indoors, or if there is no sunlight available outside on a particular day. The greenhouse needs to adjust quickly to manual changes of environmental variables too; when a user changes the target temperature, soil moisture, or light duration on the web application, the new target should be met within an hour of the command.

On the software side, we require that our greenhouse is able to classify certain growth stages like germination, sprouting, and harvesting, with an accuracy of greater than 90%, and detect common defects like bending, wilting, and leaves changing color with a false positive rate of less than 10% and a false negative rate of less than 5%. Different plants have different growth stages, but in the case of pea shoots, growth stages like sprouting and harvesting can be classified based on simple metrics such as height and the number of leaves, so we are requiring our CV algorithm to accurately classify the plants' growth stages. Detecting plant defects are more challenging, but since we do not want to risk ignoring plant defects that lead to serious complications, we are tolerating a higher false positive rate than the false negative rate. Security is also an important aspect, so we will map a single account registered on the web application to a single greenhouse by asking users to create an account with a password and authenticating requests to control the greenhouse. The user's commands to change temperature, soil moisture, or light duration conditions should be sent and received by the greenhouse hardware within a minute to help ensure that the previous requirement of adjusting conditions within an hour of the command is met. We plan to live stream the plant on the web application, which will require a camera with day and night vision, a wide field of view, and a reliable connection to keep the video streaming available 24/7. Finally, the UI of the web application should be intuitive and easy to navigate to provide the user with a pleasant experience.

Fig. 1. Overall system design of the greenhouse (Images are from Amazon product description pages)

III. Architecture and/or Principle of Operation

In our project, we have 3 main crucial components which are

- Hardware system
- CV analysis
- Web application

The key to the success of this project will be the hardware system setup. In Figure 1, we have overall system design of the greenhouse, indicating the location of each of the components. Because the greenhouse has shelves, we can place a water tank and plant grow light on the top shelf. On the middle shelf, we will have two plant trays with moisture sensors, Raspberry Pi that holds our IR-cut camera, and ESP32. We will also have two water pumps for each of the plant trays. Lastly, on the bottom shelf, we have a mini heater, 5V relay, and extension cord. Unlike what we indicated in our initial design report, we decided to get rid of the fan or cooling system because we were not able to find any compact AC within our budget. Therefore, we are only focusing on the heating system.

On ESP32, two soil moisture sensors, light intensity sensor, and temperature sensor will be connected. The data collected from these sensors will be processed within ESP32 and then sent to our web application backend by using AWS IoT and AWS DynamoDB. ESP32 will also send signals to the 5V relay so that it can turn on/off the mini heater, water pumps, and LED plant grow light.

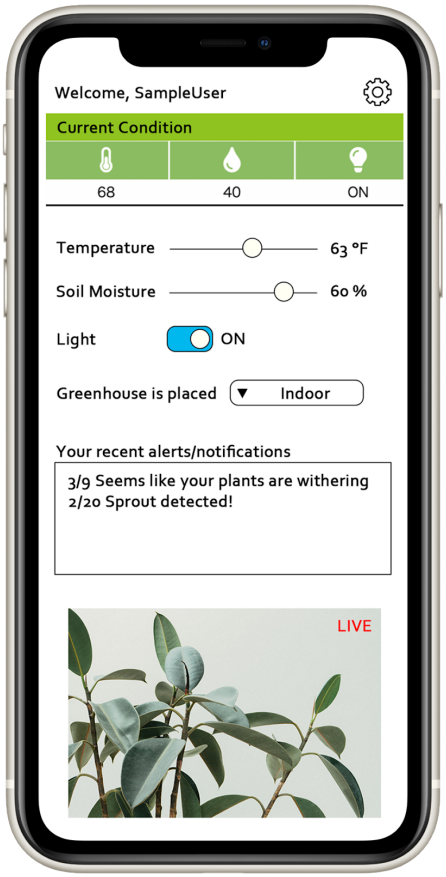


Fig. 2. Web application UI

The plants will be always monitored by an IR-cut camera that has day and night vision. The captured video will be sent to a web application via WI-FI and with a live streaming script. This video or image will also be sent to our CV analysis system which is operated by OpenCV. We will have 3 detections within our CV algorithm: leaf shape detection, HSV color detection, and stem detection. OpenCV will be able to achieve these detections swiftly as it already has HSV color detection, edge detection, size/shape detection functions within its library. Our 3 main detections will then be used for growth stage classifier, defect detector, and stem/vine bending measurer. CV analysis information will be sent to the web application backend for further use.

Our users will interact most with our web application. The user will first register or login to our web application by creating a new account or logging in with Google OAuth2. Then the user will be led to the main user page that shows the information about their greenhouse, shown in Figure 2. While the user can monitor the live video of the greenhouse, the user page will also indicate the current temperature and soil moisture, and whether the light is turned on/off. The user can change the value of these parameters manually within the web application. The user can also indicate if the greenhouse is placed indoors or outdoors. This information is necessary since regular light intensity sensors cannot differentiate between indoor lighting and sunlight, but indoor lighting does not contain the necessary wavelengths for optimum plant

growth. Therefore, we assume that all light measured by the light intensity sensor is indoor lighting if the user indicates that the greenhouse is located indoors, while we assume that all light measured by the light intensity sensor is sunlight if the user indicates that the greenhouse is located outdoors. By using data from our hardware and CV analysis, we will be sending out notifications to users through our web application. For example, when the temperature gets too high or if some plants start to wither, the user will receive a SMS notification. The user can also specify what kind of notifications or how often they want to receive them.

IV. Design Trade Studies

A. ESP32 vs Arduino vs. Raspberry Pi

TABLE I. Microcontroller Comparison

Board	Comparison Aspect		
	Required Parts	***Price (Sparkfun)***	***Max Temp***
ESP32S	Micro USB Cable	$20.95	125°C
Arduino Uno	USB-B Cable	$22.95	85°C
Raspberry Pi 3 B+	Micro USB Cable, SD Card, SD Card Reader	$35.00	85°C

We will be using an ESP32 board to gather sensor data, send sensor data to AWS, and control the 5V relays. There are other microcontrollers like the Raspberry Pi and Arduino that could also perform the same functions. All 3 of these boards support WiFi and Bluetooth connectivity, and can be connected to temperature, soil moisture, and light intensity sensors. However, there are some key differences between them that helped us decide to use the ESP32 board. Table I. above shows that the ESP32 board is the cheapest option, and can operate at a higher temperature than the Raspberry Pi or the Arduino Uno. While it is true that the Raspberry Pi is more expensive because it has more memory and processing power than the other 2 boards, our greenhouse does not require those extra capabilities since most of the data will be stored and processed using AWS IoT. It is also worth noting that some household plants that originate from tropical regions thrive in higher temperatures, so it is safer to use a board with a higher maximum operating temperature just in case the internal temperature of the greenhouse is set very high.

B. Irrigation Systems

While designing the automatic watering system, we compared different types of irrigation systems to choose the most effective one for our purpose. The 3 common types of irrigation that could be used within a greenhouse are drip irrigation, sprinkler irrigation, and sub-irrigation. Drip irrigation delivers water at or near the root of plants using a drip, spray or stream. Sprinkler irrigation utilizes overhead high-pressure sprinklers or guns to distribute water. Sub-irrigation distributes water across land by raising the water table through a system of pumping stations, canals, gates, or ditches. A simple drip irrigation system only requires a water pump and a soaker hose to build, while evaporation and runoff are minimized. Sprinkler irrigation is also simple to implement, since it only requires a hose connected to a sprinkler. However, there would be a need to carefully contain the water within the greenhouse since the greenhouse will be tested for use indoors. Sub-irrigation can control water flow more efficiently and precisely than a sprinkler, but would be the most complicated to implement, since it requires a system of pumping stations, canals, gates, or ditches to control the water table. Thus, we came to the conclusion that the drip irrigation system would be the most simple and effective way to water the plants within the greenhouse.

C. AWS vs. Other Cloud Solutions

As mentioned before, we are using AWS IoT in order to send/receive commands to/from our ESP32. Even though there are many other cloud solutions such as Microsoft Azure, we chose AWS EC2 for our cloud solution because we are aiming for quicker communication between hardware and software by completing all systems within AWS.

Similarly, while we have options such as MongoDB that can hold more complex data, we decided to use AWS DynamoDB as we will be able transfer our data quickly from/to AWS IoT when we try to send commands to the hardware. Moreover, all the data we need to store are simple values, therefore DynamoDB should be sufficient for our purpose.

D. Django vs. Other backend web frames

We chose Django over other web frameworks because it is a high-end Python web framework. Our CV analysis will also be written in Python. Once our CV analysis detects problems on plants, we need to organize this information and send them to the users as a notification. Therefore, it would be the best practice for us to use Django and keep our overall programming language the same by using Python for faster communication.

E. Twilio vs Nexmo SMS Messaging

Our SMS API, Twilio, can be also customized with Python. We also looked into other SMS APIs but most of them have limited or superabundant services. For example, Nexmo SMS Messaging, the one we were also looking at, has many features including calling and also global texting; however, these features are unnecessary for the project. The cost performance of Twilio also seems reasonable. The overall cost depends on how many messages we receive/send, so it can stay cheap and is perfect for our project as we will be sending only a couple messages per day.

F. Computer Vision vs. Machine Learning Model

Initially, we were planning on implementing a Machine Learning model to categorize growth stages and to compare healthy or diseased plant images to the greenhouse plants, but through further research we found that this process holds high risks and complicates visual plant growth analysis. Because every plant grows so differently even within a species of plants, there is much room for error and misleading classification with image-fed learning models. We would also have had to be reliant on common images from online for our

training data, but with most images online taken with specific lighting and grown in commercial amounts, such data would not be appropriate for the small scale system we've designed.

Alternatively, computer vision applications are much more common and foreseeable in smart planting systems. With the CV approach, not only is a growth stage classifier and defect detection possible, but also vine bending is much more viable. We could much more efficiently customize the analysis to work with the lighting system that the plants are under as well, as opposed to the online images we may have had to rely on for the learning model. The growth stage classifier could also be much easily implemented with a pixel per metric method, where the height of the plant in the image can be converted to real-life size by , rather than being trained with images. Because of the simplicity and safety of the CV approach compared to the learning model, we decided to take the former path.

G. Raspberry Pi IR-Cut Camera vs. Logitech C920

TABLE II. CAMERA COMPARISON

Camera	Price	Resolution	Strengths
RPi IR-Cut	$24.99	1080p/30fps	Day & Night Vision
Logitech C920	$69.99	1080p/30fps	RightLight Tech.

Initially, we had looked into the Logitech C920 camera, as it is a popular webcam for CV projects and it had RightLight technology, which adjusts the video quality to the lighting conditions. This would reduce the complications of trying to adjust the LED lights of the greenhouse for the pea shoots to be visible. Further, with constant image quality, we would not need to change the specific HSV color values for edge detection of the video. However, when we pivoted to adding a live stream monitoring system, we realized that we would need both day and night vision. The RPi IR-Cut camera was the only camera we could find which integrated both vision types in one camera, and we would've needed to add a filter or work with two cameras if we chose the Logitech C290. Because of the lower price, equal resolution, and the day and night camera, we continued with the RPi IR-Cut camera. As for the lighting conditions for high quality imaging, we will adjust the LED lights to accommodate for the RPi camera.

H. Raspberry Pi vs. ESP32 Integration for Computer Vision

Initially, we were planning on connecting the Raspberry Pi Camera to the ESP32 and uploading the CV application to the ESP32 since the ESP32 is cheaper and all the hardware would be connected to one board. However, the pins used to connect a RPi camera module to the ESP32 are not compatible. Further, there are no ESP32 camera modules that have both day and night vision, which is integral for the live stream. Because Sarah and Hiroko are working separately, we thought it would be best to work with hardware that suit their respective components that they were working on. Further, the RPi was available to be borrowed from the ECE department. Hence, we decided to implement the CV application on the RPi, since there is a larger community of CV projects made with Raspberry Pi Camera Modules, the Raspberry Pi ensures the processing power for constant live streaming, the RPi could be borrowed, and its compatible camera module comes with both day and night vision.

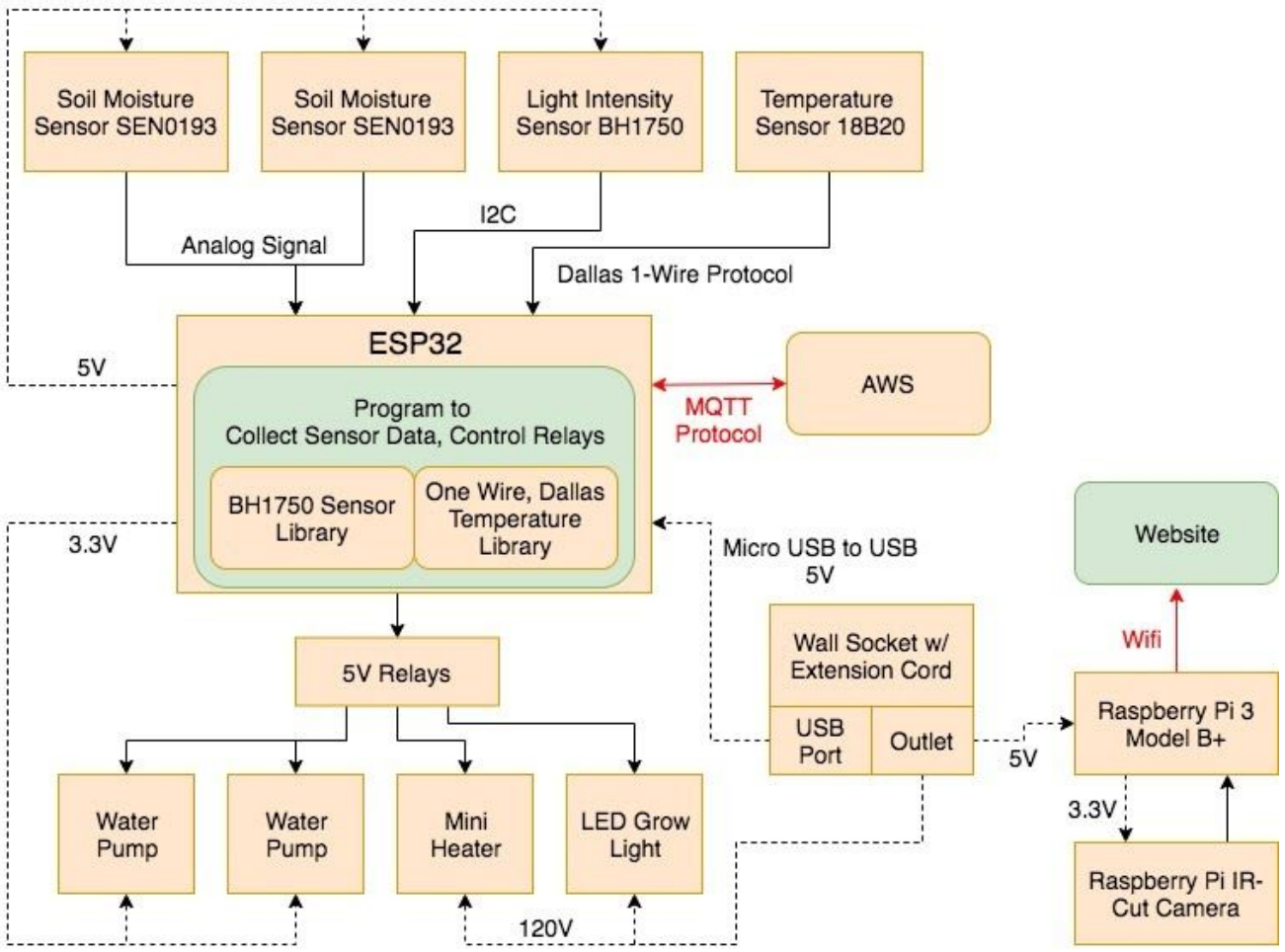


Fig. 3. Hardware Block Diagram

V. System Description

A. Hardware Subsystem

The hardware subsystem of the greenhouse revolves around the ESP32 board that gathers sensor data, sends sensor data to AWS, and controls the watering, heating, and lighting systems. Temperature is measured by the waterproof 18B20 temperature sensor, which uses the Dallas 1-Wire Protocol to send the data over to the ESP32 using 1 GPIO port. Light intensity is measured by the BH1750 photodetector, which communicates with the ESP32 via the I2C bus. In order to extract the sensor data from the input signals, the ESP32 will have the BH1750 sensor library, OneWire library, and the DallasTemperature library installed. As shown in Fig. 1, our greenhouse will have space for 2 trays of pea shoot plants, so there will be 2 capacitive soil moisture sensors SEN0193 that send analog signals to the ESP32, which has built-in ADCs to convert them into digital signals. All of the sensors except the temperature sensor is powered by the 5V output from the ESP32, while the temperature sensor is powered through the data line. Once the ESP32 has all of the input signals converted into temperature, soil moisture percentage, and light intensity values, the information is sent to AWS using the MQTT protocol, so that the software subsystem can analyze the data.

On the other hand, the ESP32 will receive commands from AWS to turn the water pumps, heater, and LED plant light on/off. The ESP32 is connected to 5V relays that are wired to an electrical outlet where the various appliances are plugged into, which gives the board the power to turn the appliances on and off. As shown in Fig. 1, the small space heater is placed at the bottom of the greenhouse shelf since the warm air will rise and create a convection current to maintain a constant temperature throughout the greenhouse. The special LED plant light is placed above the plants to provide light with photosynthetically active radiation (PAR), the range of light that can be used by plants to photosynthesise. 2 water pumps are placed inside a water tank to pump water through a tube to water the 2 separate plant trays.

In order to provide the computer vision subsystem with live images of the plant, a Raspberry Pi IR-cut camera connected to a Raspberry Pi will be placed inside of the greenhouse shelf. The details of these components will be explained later in the computer vision section, since the camera is separate from the other sensors and appliances in order for us to parallelize the workflow. Once the computer vision algorithm is complete towards the end of the semester, the camera and Raspberry Pi will be sent to Pittsburgh, and integrated into the physical greenhouse.

After the electric components are wired and placed within the greenhouse, we will use a clear acrylic display case to contain the sensors and microcontrollers to protect the parts from water. We may need to cut holes into the acrylic display case to let the temperature sensor and wires stick out, in which case we will put an order in with TechSpark once we get specific dimensions.

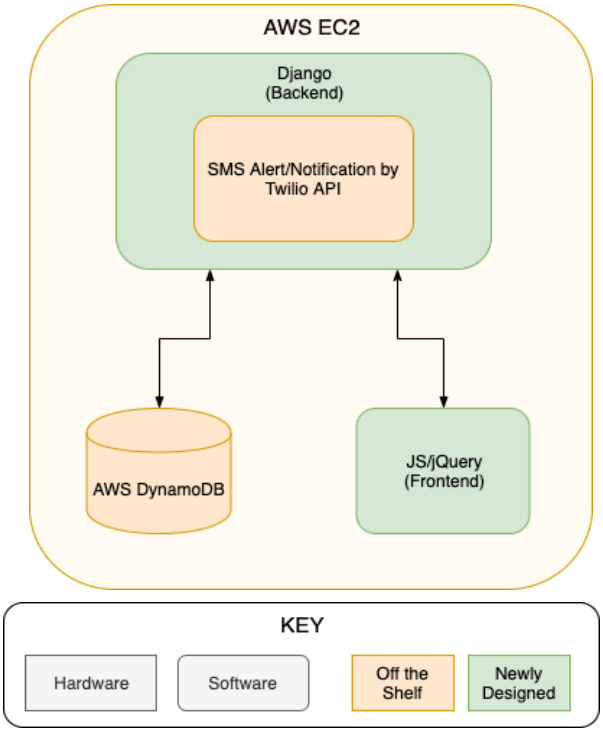


Fig. 4. Web Application Block diagram

B. *Web Application Subsystem*

As shown in Figure 4, we will be deploying our website with AWS EC2. Backend will be developed with Django while the frontend will be developed with Javascript. Within our backend, we are using Twilio for sending SMS alerts and notifications. The data used within our web application will be stored within AWS DynamoDB.

From Figure 3, the ESP32 will be sending data to AWS. It will be first stored in AWS DynamoDB, using AWS IoT. From here, our backend is retrieving this data from the database and renders these values on the website. Whenever the user modifies the parameters of the greenhouse, this data will be sent to AWS DynamoDB and then to AWS IoT so that our web application and hardware can communicate with each other. The sensor data will be further analyzed within the backend. We will use this analyzed data to send users notifications. For example, if the temperature sensor detects the greenhouse to be 60-70°F over the target temperature, our web application will warn the user. If it goes below our target temperature, we will automatically turn the heater on while also sending alerts. Once we hit the target temperature, we will automatically turn off the heater. Even when the user is not interacting with the website, AWS IoT is constantly sharing information from hardware to the web application. Hence, within the database, the current information of the greenhouse will be always updated.

From Figure 5, the data from CV analysis and live streaming system will be handled in a similar manner. By using our live stream script, the unprocessed video data will be directly rendered onto the website using the IP address of IR-Cut Camera. The data from the camera will also be analyzed within openCV. This analyzed data will hold information about growth stage and defects. The backend will receive this information and send alerts or notifications accordingly.

The website frontend will be pretty simple because all the important information can be handled within the backend. It will be a single page application so that the user can access from PC and also from their phones.

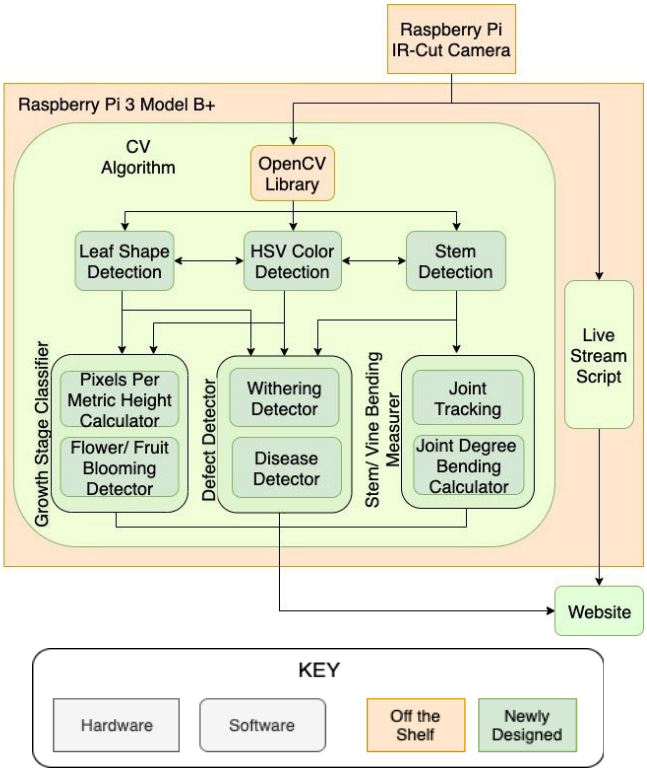


Fig. 5. Computer Vision Application Block Diagram

C. *Computer Vision Subsystem*

The computer vision application will be reliant on the OpenCV and NumPy libraries, specifically in the HSV Color Detection and Edge Detection functionalities. After extracting the outlines of the pea shoots, we will determine the different

parts to the plant such as the flowers, peas, stems, and leaves. There will be three types of detectors: the classification of three main growth stages (germination, flowering, and harvest), the detection of common diseases or withering, and the measurer for extreme vine or stem bend. The application will be integrated onto a Raspberry Pi 3 Model B+ and a real-time video will be captured through the Raspberry Pi IR-Cut Camera Module which includes both day and night vision for constant monitoring. The camera must be placed sideways to capture a side view of the pea shoots in order for the detectors to work properly.

The first parameter that will determine which growth stage the pea shoots are at is the height of the plant, which will be measured through the pixel per metric method. To initialize the method, it takes in the distance from the camera to an object along with the height of an object, and divides the pixels by the real height of the object. With this pixel per metric conversion scale, we multiply it by the height of the pixels captured in the camera to determine the real life plant height. The second parameter that would likely distinguish flowering from harvest is the detection of flower and fruits, which will be implemented by passing another layer of the HSV Color Detection onto the non-stem parts, since the leaves and flower shapes of pea shoots are similar.

The defect detector will analyze the leaves distinguished by the leaf shape detector and search for any spotting which is a common disease and insect bite pattern. For withering, we will look for yellowish or brownish colors.

For detecting extreme bending of stems or vines, we will distinguish the base stem of a pea shoot, and determine whether the rest of the stem strays significantly from the x-axis point of the base. Then, we find the starting point of the bend and if the bend is greater than 45 degrees from the straight part of the vine, we will categorize that as urgent.

The first time a growth stage is reached, the user will be alerted. When harvest is reached, a notification may be sent more than once so that the peas are picked and the plant is snipped to promote further growth or to prevent overgrowth. Defect and vine bending notifications may be sent out multiple times a day based on the severity of the plant health.

Further, there will be a live stream monitoring system that will be captured and run through the same Raspberry Pi and camera used for CV analysis. The live stream will be available through referencing its IP address and will be embedded into the website for viewing.

For further information and details, refer to the block diagram in Figure 5.

VI. Project Management

A. Schedule

Currently, we are on time with our schedule, as we are all working on our individually assigned components. Kanon built the website's login and registration page along with the greenhouse controls page with Django, and she is working on setting up DynamoDB. Hiroko is connecting the sensors to the ESP32 and will be adding the relays to the equipment. Sarah has implemented the HSV Color Detection and edge detection, and is applying that to the growth stage classifier. For specifics of the schedule, please refer to Appendix I: Schedule Chart.

As we researched more about greenhouses and plants, we made some changes to the schedule. After deciding on pea shoots as our testing subject, we updated the schedule to include when to start planting pea shoots for our tests. We gave the pea shoots a week head start before the testing period, because around a week is needed for their germination.

We also had to factor in when Sarah would be sending the Raspberry Pi camera and board to the greenhouse in Pittsburgh, so we would send it two weeks before the project is due so that Kanon and Hiroko have time to set it up and make adjustments to make the CV implementation and live stream work on the greenhouse environment.

B. Team Member Responsibilities

We have divided the work such that Hiroko is working on most of the hardware, specifically arranging the relays, setting up a feedback loop with the sensors, the MQTT protocol connection between the ESP32 and the website, and the assembling of the sensors, greenhouse, and equipment. Kanon is working on the cloud database setup with AWS EC2, integrating DynamoDB and Twilio with the website, managing the data received and sent between the hardwares and software, and creating a web platform with Django for users to interact with the greenhouse, and assisting in the hardware setup. Sarah will be focused on creating a remote CV application and is responsible for setting up her day and night vision RPi Camera with the RPi and implementing analysis on recognizing growth stages, defects such as withering, diseases, or deforms, and extreme vine curvature that requires plant staking. She will also complete a 24/7 live stream of the greenhouse and a notification system when attention is needed on the plants.

C. Budget

The Bill of Materials is located at the bottom of the report, in Appendix II: Budget and Parts List. With Sarah not being in the same location as Hiroko and Kanan, we decided that it would be best to work with several hardware instead of integrating all the hardware together, so Sarah's CV implementation will be integrated to the RPi and will work remotely from the ESP32 system. In the Bill of Materials, the first column contains the web development service and API credits and all the hardware components bought the first week after the project proposal. The second column contains the AWS services that have prices dependent on how much of the service we use, so we estimated how much we would use these web development tools and gave enough room in the remaining AWS credit for any changes in the future. We saved some room in our budget as well for the shipping cost of Sarah's hardware components to Hiroko and Kanon, and sending back the material borrowed from the ECE Department.

D. Risk Management

The risks we had in mind changed considerably between

when we pitched our idea and when we began implementing the components and designing the project. For the hardware, we are concerned about the response time being too low for the communication between the hardware, cloud, and software, the feedback loop appropriately adjusting conditions without overshooting or taking too much time, and placing hardware such that it will all fit in the greenhouse while being safely away from the water and organic material we are working with, and the instability of the connection between hardware and software. Plans for mitigating such challenges include setting a threshold for the frequency of the feedback loop updates and changes and putting the hardware in a container. For the software, we are aware of the potential issues with the latency of the live stream monitoring, the lag in the website, the proper automation of multiple plant types, overlapping leaves and plants that may hinder CV analysis of defects, and high measures of false negatives for classifying growth stages and defects. To mitigate this, we chose the RPi to run the CV application and live streaming script as it has the computing power to run both components and in the future we may add more cameras or moving cameras to get all angles of the plant.

Further into the project, more risks we had to consider were receiving the wrong data from the database, and to mitigate this issue we are thinking about either regaining the data before outputting the value to the website or to notify the website if drastic changes take place. We also hope that the night vision in the IR-Cut cameras will work properly, but in the case that it does not we will need to reconfigure the LED lights to be on a certain brightness for night vision to work. After testing the OpenCV module, we found it very important that the subject we are analyzing contrasts with the background or the unnecessary components of an image, so we are looking into making a monochrome backdrop in the greenhouse that provides the best contrast. Any unexpected issues that we may come across with the hardware, we have decided that our system will be able to detect and notify the user about it. We also found some difficulties in connecting the sensors properly without damaging them, so Hiroko signed up for TechSpark to receive some help from the TAs and professors on soldering the sensors to the ESP32.

VII. Related Work

Agricultural technology is one of the most rapidly developing fields, and we took some inspiration from the greenhouses that are already automated at large farms. The standard sensing systems for these greenhouses are LED lights, hydrometers, and thermometers, so we decided to include these components to our project and include a heater, irrigation system, and lighting system to adjust such parameters. There are many smaller projects online that use sensors compatible to the Raspberry Pi, but none of them seem to implement lighting, watering, and temperature altogether, so we are hoping to scale down the size of commercial greenhouses, combine all the sensors and equipment through an ESP32 instead, and create a smarter system with controls connected from a website and with CV analysis.

VIII. Summary

Our team has been successful in completing the deliverables on time, and creating detailed, thorough designs for our subsystems. We have enough room in our budget to attend to any unexpected issues and risk mitigation.

A. Future work

Beyond the end of the course, we hope to scale the types of plants the greenhouse accounts for, so ordering a larger greenhouse and more sensors and equipment are in the bigger picture. To accommodate for more plant types, we could add a web functionality where users can input the conditions necessary for each plant type or users can input the type of plant and we retrieve information about that plant through an official database on plant conditions.

Due to financial and technical constraints, we could not include the air cooler, but if the greenhouse were to be used in a more tropical environment, an air cooling system would be necessary. Depending on the future scale of the greenhouse, we may need more compact and customized containers for the hardware, especially if the greenhouse is subject to high levels of humidity and moisture. A customized PCB board could also simplify the wiring and complication that comes with breadboards.

Some of our members hope to use some of the systems in the greenhouse to grow plants in an urban setting, and depending on the living situation the system may be adjusted to run without an enclosed greenhouse.

References

[1] Aufranc, Jean-Luc. “Know the Differences between Raspberry Pi, Arduino, and ESP8266/ESP32.” *Embedded Systems News*, CNX Software, 24 Mar. 2020, www.cnx-software.com/2020/03/24/know-the-differences-between-raspberry-pi-arduino-and-esp8266-esp32/.

[2] Badgery-Parker, Jeremy. “Light in the Greenhouse.” NSW Agriculture, *Agnote*, Sept. 1999, https://www.dpi.nsw.gov.au/__data/assets/pdf_file/0007/119365/light-in-greenhouse.pdf.

[3] Datta, Sumon, et al. “Understanding Soil Water Content and Thresholds for Irrigation Management.” Oklahoma State University, Aug. 2018, https://extension.okstate.edu/fact-sheets/understanding-soil-water-content-and-thresholds-for-irrigation-management.html.

[4] Hattersley, Lucy, and Lucy is Editor of The MagPi. *Monitor Plant Growth with AI and OpenCV*. The MagPi Magazine, www.magpi.raspberrypi.org/articles/monitor-plant-growth-ai-opencv

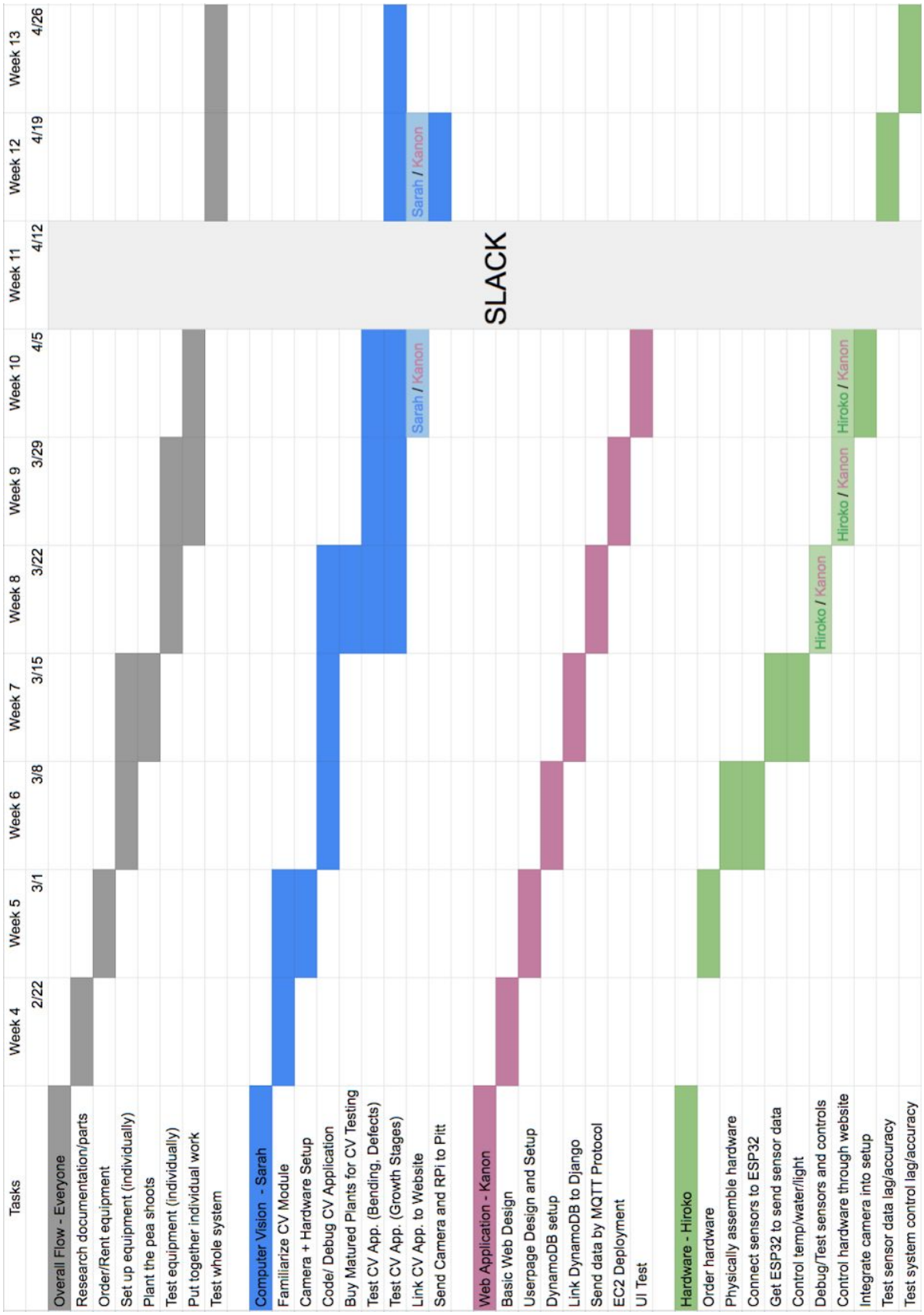
[5] Miles, Carol A, et al. “Pea Shoots.” Pacific Northwest Extension, http://pubs.cahnrs.wsu.edu/publications/wp-content/uploads/sites/2/publications/PNW567.pdf.

[6] Pennisi, Svoboda Vladimirova, and Robert Westerfield. “Care of Holiday and Gift Plants.” University of Georgia Extension, 1 June 2006, https://extension.uga.edu/publications/detail.html?number=B1318&title=Growing%20Indoor%20Plants%20with%20Success

[7] Rizza, Matteo. *Greenhouse Monitoring with Discovery Kit IoT and Android*. Hackster.io, 6 June 2019, www.hackster.io/matteo-rizza/greenhouse-monitoring-with-discovery-kit-iot-and-android-333430.

[8] “Types of Agricultural Water Use.” Centers for Disease Control and Prevention, www.cdc.gov/healthywater/other/agricultural/types.html.

IX. Appendix I: Schedule Chart



X. Appendix II: Budget and Parts List

Component	Price	Quantity	Total Price	Source
AWS Credit	$50.00			18-500 lab
Twilio Notification	$0.0075 / send	1000sends	$7.50	Twilio
Greenhouse Shelf	$38.99	1	$38.99	Amazon
LED Grow Light (2 ft)	$44.85	1	$44.85	Home Depot
Zip Ties	$5.49	1	$5.49	Amazon
Planter Tray w/ Soil (2 Pack)	$34.99	2	$69.98	Amazon
Pea Shoot Seeds	$15.99	2	$31.98	Amazon
Water Tank (2.5 gallon)	$9.06	1	$9.06	Amazon
Mini Heater	$19.99	1	$19.99	Amazon
ESP32 (3 boards)	$16.99	1	$16.99	Amazon
Irrigation DIY Kit	$30.99	1	$30.99	Amazon
Breadboard (3 boards)	$0.00	1	$0.00	18-500 lab
Temperature Sensor (18B20)	$12.49	1	$12.49	Amazon
Light Intensity Sensor (BH1750)	$6.69	1	$6.69	Amazon
Extension Cord (8 ft)	$25.99	1	$25.99	Amazon
Micro USB to USB (10 ft)	$13.99	1	$13.99	Amazon
Wire Cutter	$9.59	1	$9.59	Amazon
Outlet Box	$5.26	2	$10.52	Amazon
Duplex Receptacle	$1.29	2	$2.58	Amazon
Duplex Receptacle Wallplate	$2.98	2	$5.96	Amazon
NM/SE Clamp Type Connector	$3.99	2	$7.98	Amazon
Appliance Cord	$7.80	2	$15.60	Amazon
5V Relays	$8.49	1	$8.49	Amazon
Raspberry Pi IR-Cut Camera Day & Night Vision	$24.99	1	$24.99	Amazon
Raspberry Pi 3 Model B V1.2	$0.00	1	$0.00	18-500 lab
RF Transmitter and Receiver	$0.00	1	$0.00	18-500 lab
Raspberry Pi Power Adapter	$14.90	1	$14.90	Amazon
Shipping Label	$15.00	1	$15.00	18-500 lab
Budget			$600.00	
Total Cost			$450.59	
Remaining Balance			$149.41	

AWS Component	AWS Price	Quantity	AWS Total Price
AWS IoT Core (Connection)	$0.08 / 1 million minutes	129600 min	$0.01
AWS IoT Core (MQTT Protocol)	$1 / 1 million messages	1 million messages	$1.00
AWS EC2	$0.0843 / 1 hour	360 hours	$30.35
AWS DynamoDB	$1.5 / 1 million write+read	1 million write+read	$3.00
AWS Credit			$50.00
AWS Total Cost			$34.36
Remaining AWS Credit			$15.64

Tool	Purpose
Arduino IDE	Program ESP32
BH1750 Sensor Library	Read data from light intensity sensor
One Wire, Dallas Temperature Library	Read data from temperature sensor
AWS IoT Core	Communicate with ESP32
AWS EC2	Cloud solution
AWS DynamoDB	Database
Twilio	Notification system
OpenCV Library	Plant detection