Project Design Report

Future Greenhouse/Plant Nursery Automation

Students:
David Craft and Joshua Blazek

Emails:
craft@pdx.edu and blazek@pdx.edu

Professor Roy Kravitz

ECE 558 - Winter 2022
Version 1.0 – 02/28/2022

Table of Contents

1.0 Overview

1.1 Problem Statement:

Growing plants from seed is a great way to start gardening earlier in the season, but this can be a time consuming endeavor with a high chance of failure for even experienced gardeners. Growing from seed lets you select your own varieties, benefits the environment by reducing the carbon footprint of your food, it is sustainable, and economical.

The goal with seed starting is to have your seedlings ready to go outside when the weather is favorable, which generally requires starting the seeds six to eight weeks before the last frost. Each plant has unique seed-starting requirements, and can be fussy and difficult to start in the dead of winter. This project is designed to ease the burden of starting a garden from seed and streamline the process as a hands-off automated approach. The design is also able to be scaled up from seed nursery to an automated greenhouse.

1.2 Key project features and areas of focus:

- **Control** - An Android application connects to the automated garden via MQTT to maintain various environmental controls.
- **Monitor** - Android application will be used for Real-time feedback of the garden's various sensors to monitor the plants and environment.
- **Water wisely** - Automated drip irrigation by monitoring a moisture sensor. It's important to keep soil consistently moist, but avoid overwatering, which promotes diseases. Maintain consistent moisture. Once seedlings are growing, reduce watering so soil partially dries.
- **Keep soil warm** - Timed scheduler for heat mat. Seeds need warm soil to germinate. They germinate slower, or not at all, in soils that are too cool. Most seeds will germinate at around 78°F.
- **Give seedlings enough light** - Timed scheduler for light source. Not enough light leads to leggy, tall seedlings that will struggle once transplanted outdoors. Ideally, seedlings need 14-16 hours of direct light per day for healthiest growth.
- **Circulate the air** - Humidity sensor fan control. Circulating air helps prevent disease and encourages the development of strong stems.
- **Inspect for disease** - RPi camera/monitoring with disease image recognition using TensorFlow and app alert. The moist environment needed for the plants is also a good environment for the formation of powdery mildew.

1.3 Android:

The Android application primarily is used to remotely monitor and a control for the smart greenhouse/nursery. The app will have access to sensors that provide reading for temperature, humidity, soil moisture, light, and a camera designed to detect powdery mildew. The application provides settings to control the heat mat and lighting schedules, and is able to activate the water pump and fan. The application connects to a Raspberry Pi that is controlling the garden using the Paho MQTT client using the HiveMQ broker.

1.4 Raspberry Pi and other hardware:

The Raspberry Pi runs a Python script to connect to and control the various functions of the garden and will utilize the Paho MQTT client library to connect to the Android device via HiveMQ. The other primary hardware used for this project is listed in the following table.

Hardware	**Power**	**Description**
Raspberry Pi 3 B or later	5V	SBC selected to control and monitor the garden
Adafruit AHT20	5V	Temperature and humidity sensor, trigger to activate fan
Fan	12V	Fan turns on when temperature or humidity is too high
Soil Humidity Sensor	5V	Check soil moisture, trigger to activate water pump
Small water pump	5V	Used to irrigate the soil trays with drip irrigation.
Water Level Sensor	5V	Measure water level in tank for drip irrigation feed
PCF8591 A/D Converter	3.3V	Analog to digital converter for getting analog signals into the RPi
Photoresistor	3.3V	Monitors light, trigger to turn on relay for lights
4x Relay Module	3.3V	Used to turn on higher voltage devices
Heat Mat	120V	Warm soil, activated by relay switch
Light(LED or fluorescent)	120V	Maintain ~16/8 on/off light cycle, activated by relay switch
Raspberry Pi Camera	-	Used for imaging and tensorflow script to check for mildew

Table 1: Project Hardware

1.5 Other tools:

- Firmware: Python
- Development environment: VSCode, Pycharm, Android Studio
- MQTT Broker: HiveMQ
- Image analysis: Tensorflow
- Various other hardware for physical build(pliers, hole punch, drip irrigation, containers, wires)

1.6 Project Goals

We were able to accomplish many of our stated goals. Number 8 is yellow as half met because this was primarily a stand-in in case we were not able to connect any higher voltage devices. We did connect LEDs to check sensor signals before attaching the larger devices. There is still an LED indicator connected to the light sensor, but #8 was more of a control panel indicator light idea which was removed from the system during the design process.

Must:

1. **Soil moisture sensor and water pump activation**
2. **Temperature/Humidity sensor with fan activation**
3. **Camera with tensorflow to monitor plants**
4. **Android application with added sensors and parameter settings for hardware**

Should:

5. **Add photoresistor to monitor luminosity**
6. **Add 5V relay switch**
7. **Code block for light and heat mat timer**
8. **LEDs to indicate when various hardware is/should be on**

May:

9. **Connect higher voltage lights and heat mat to 5V relay switch**

~~10. Android: database to save data points, create modular controls for plant groupings~~

11. Additional sensors: **water level sensor for supply**, ~~Servo for vent cover, Barometer, gas sensor~~

2.0 Design Details

2.1 System Architecture

The goal of the future garden project is to automate as much of the process for starting plants from seed as possible. We worked to accomplish this by connecting a number of environmental sensors to a Raspberry Pi that will act on those sensor readings for plant management. There are six primary inputs into the system: soil humidity sensor, water level sensor, AHT20 temperature sensor, AHT20 humidity sensor, light (photoresistor) sensor and image input from a camera. Our outputs primarily flow through the 4x relay switch to control our lights, fan, water pump, and heat mat. We designed override controls to work from an Android device by MQTT publish/subscribe protocol via the HiveMQ broker.

2.1.1 Block Diagram:

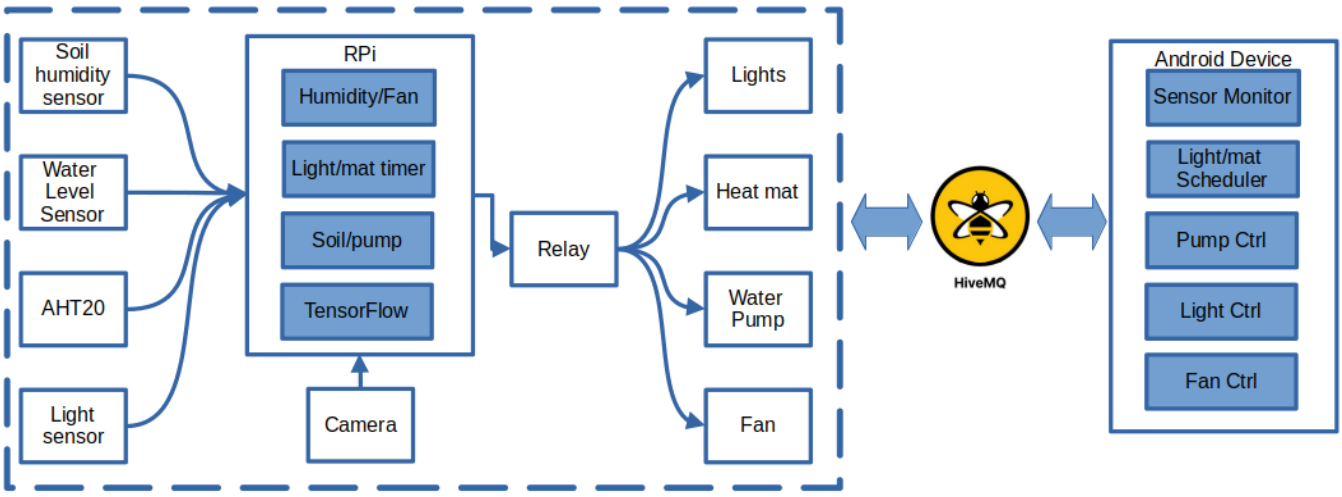

Figure 2: Block Diagram of Future Garden Design

The Raspberry Pi does most of the heavy lifting in our design and should be able to happily manage the plants without intervention beyond occasionally filling the water reservoir (future design could connect a main with a solenoid switch). The Pi will turn on the fan at specific humidity, turn on and off the lights on timed intervals, add water when the soil gets dry, and tell you if it sees any mildew forming. The current build will pump water into the drip irrigation tank for drip irrigation until the water level sensor is tripped and turn off the pump to not overflow the tank or over water the plants.

2.1.2 RPi Physical Build:

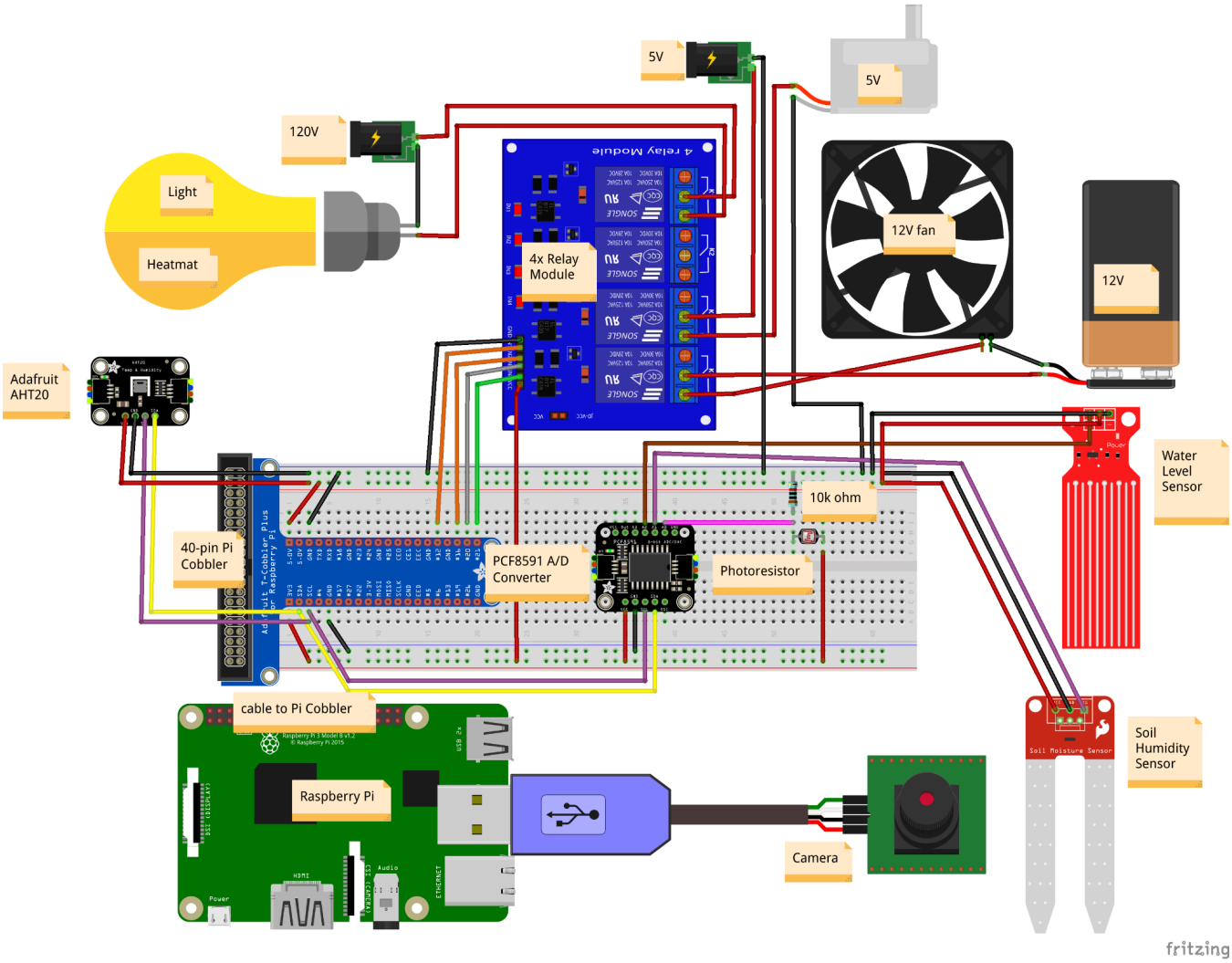

Figure 3: Physical Build Layout

The above image is the basic layout of the physical build for our Raspberry Pi. Since half of our sensors required 5V and the other half 3.3V, we dedicated one rail of our breadboard to each voltage. There are two I2C devices and they have separate SDA/SCL lines running from the Raspberry Pi, one connected to the Adafruit AHT20 temperature and humidity sensor and the other to our PCF8591 A/D converter. Our remaining sensors have analog signal outputs and are run through three of the four A/D inputs of the PCF8591: photoresistor is connected to port A0, water level sensor to A1, and soil humidity sensor to A2. GPIOs 12, 16, 20, and 21 are connected to our 4x relay module to power our higher voltage devices. Even though the water pump is 5V, it is powered externally through the relay due to the possibility of current overdraw. The GPIOs are toggled on and off by input signals from the sensors or manually activated from our android application.

3.0 Theory of Operation

3.1 System Communication

Our system operation is decentralized in nature. Our Raspberry Pi base station can be replicated to control numerous greenhouses across different locations, this was demonstrated by our ability to control our separate garden setups at our build locations. We chose HiveMQ as the MQTT broker and connected using the PAHO MQTT package on the RaspberryPi and the Android device. All the messages are sent over WiFi using the MQTT messaging protocol. Each Pi will operate autonomously and provide MQTT message data though a wifi connection to the HiveMQ broker. This data can be monitored on the smartphone application and commands can be sent to the Pi controlled gardens to override their normal operations.

3.2 Raspberry Pi Firmware and Sensor Communication

When firmware initializes it connects to the HiveMQ broker service and subscribes to the list of topics that it will receive from the android device. Next we enter our loop. The first thing the program does is check for new MQTT messages, if there is a message it will parse the message to determine what action needs to be taken. In the case there are no messages we then check if there are any scheduled tasks, if the time has passed to turn on/off the circuit for the lights and heat mat it will happen at this time. After the messages and scheduler are checked we begin reading sensors.

The temp/humidity sensor is checked first. The temp/humidity sensor's main function is to control the fan and provide stats for the monitor. If an existing command flag exits the user turned on the fan and just the stats are provided. When there is no flag set, we check to see if the relative humidity is over 70%, if it is we turn on the fan to circulate the air until our humidity is lower.

Before we check our soil humidity sensor we check to see if our pump is already running, either by user or from a previous sensor command, by checking the status of the relay switch. If the pump is running we check the water level sensor in the drip irrigation tank. If the water level sensor is tripped the pump will be turned off regardless of the user command. This acts both as a safety precaution for water overflow and to prevent over watering the plants.

Next we read our soil moisture sensor. If our soil moisture level is less than 50 our soil is drying out, we then check to see if the pump is already or the drip irrigation tank is full of water. If this is the case we just return the reading, otherwise a signal is sent to turn the pump on.

We then get a reading from our light sensor. Currently this just provides information about the lighting conditions and turns on an LED when it gets dark. We initially had it also turn on the main

lights, but realized that this could upset the scheduled lighting and the feature was removed from the final design.

With our last input we check our camera image and look for any powdery mildew using TensorFlow which is explained in a later section. If any disease is identified an alert message is generated. Lastly all sensor data and peripheral device statuses are packaged/published to HiveMQ and the loop repeats.

3.2.1 Flowchart:

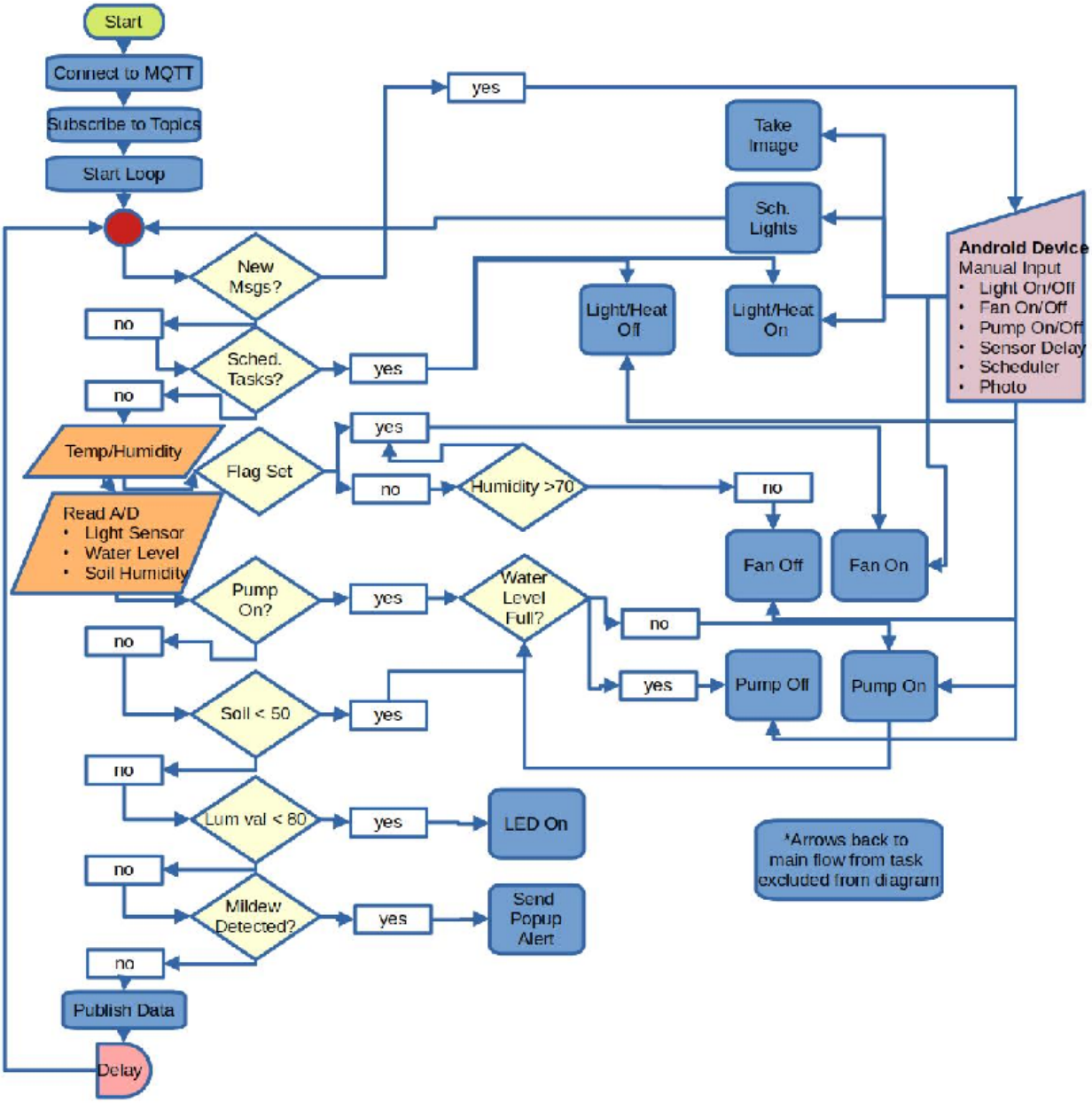

Figure 4: Design Flowchart

3.3 Raspberry Pi TensorFlow

One of our main goals was to be able to notify the user of disease on our plants. This was no small feat and required many MANY hours of computer time to build out models that would work on the small footprint of the Raspberry Pi. To talk about this we'll touch a bit on what's going on in this picture.

Figure 5: Object Detection

The idea behind the Convolution Neural Network is to take something - in our case an image - and send through a variety of layers - extracting features through convolution, pooling or sub sampling those features together and eventually - fully connecting back to classify the data with a probability. In this image here the output is a set of 4 probabilities where each probability is 1 of 4 categories with a deer getting a probability of 94%.

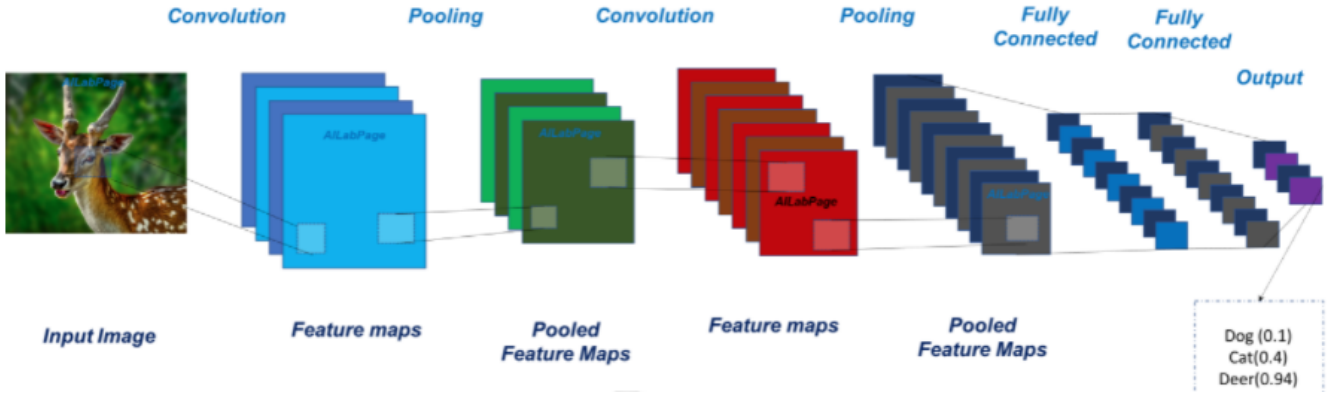

Figure 6: Flow for CNN

We needed a dataset to train our model. So we painstakingly sifted through images of ostriches and mildewy plants to get 200 total images. For object detection to work best you want to tell the neural network where in the image your object is. Especially if there are multiple objects in the image. A common example of this problem is cats and dogs. Most pictures of dogs are taken outside and

most pictures of cats are taken inside. If you train a model without annotating it will learn that a picture with green in it is probably a dog regardless of what animal is in the picture. So you annotate the picture. We used a program called label image for this.

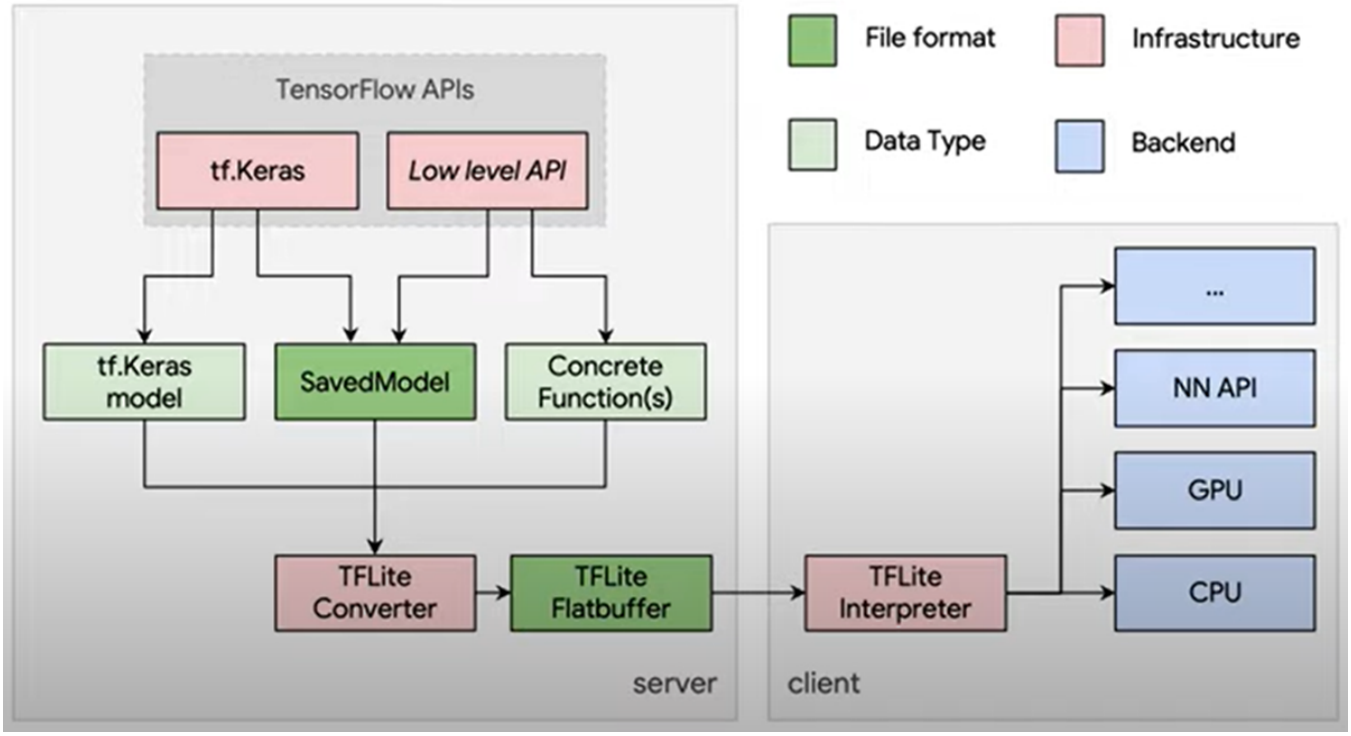

Figure 7: TensorFlow Flowchart to TFLite

Our first step was to see if image classification was possible. It took a few hours but using Google's cloud computers we were able to build out an image classification model using EfficientNet which was first published in 2016 at the International Conference on Machine Learning. This family of models is built to scale down the convolution network into a tflite model. A model built especially for light weight duty such as that on a Raspberry Pi. After running my model through the TFlite converter it was 8.4x smaller and 6.1x faster.

It's not perfect but it consistently gives good data for our project. After catching an ostrich or mildew, it will publish to MQTT a message with the relevant data and it will also publish the captured image to be viewed on the Android device.

3.4 Android Application Operation

The Android application is pretty straightforward to control. Once the application is open the connect button is used to establish the connection to HiveMQ. A series of toast will advise of success/failure and then will subscribe to a series of topics to get information from the Raspberry Pi. The top half of the screen *fig.6* (1) displays our sensor readings to monitor the garden. In the rest of the display (2)-(9) are our controls:

(2) - Turn the fan on and off.

(3) - Turn the pump on and off (override if our drip irrigation container is full).

(4) - Turn the lights and heat mat on and off.

(5) - Button to schedule timer for the lights to turn on and off - > popup (9).

(6) - Slider to change the interval between sensor reads.

(7) - Connect to the MQTT broker service.

(8) - Disconnect from MQTT.

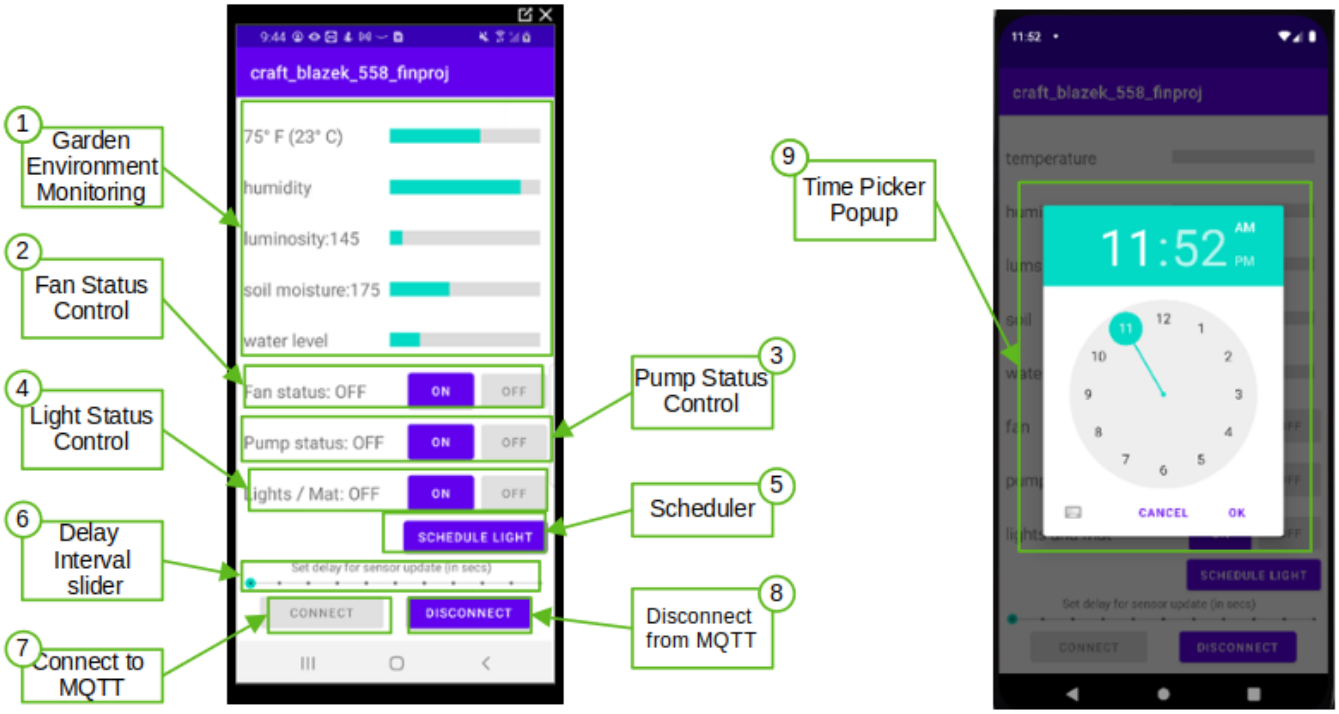

Figure 6: Android Control Interface

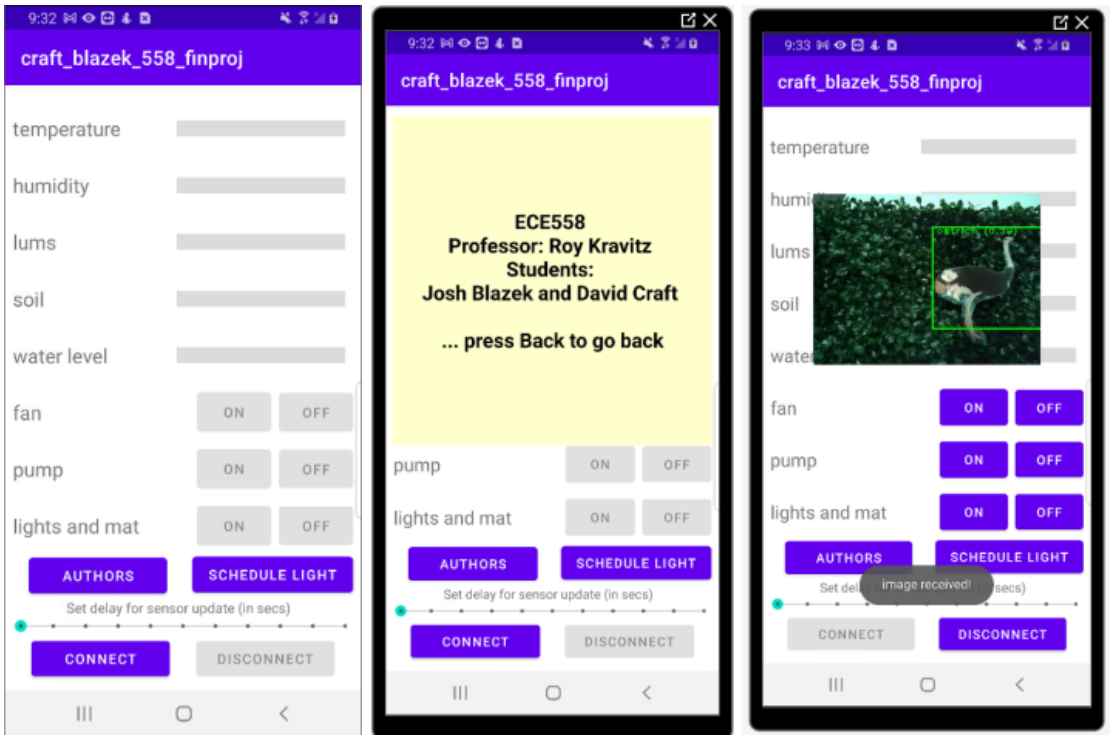

Figure 7: Demonstration of Fragment Use and Image through MQTT

4.0 Results

We were able to demonstrate the activation of the heat mat, the lighting, and the humidity fan control through the Android application. We were also able to demonstrate autonomous function by the various sensors activating the fan when it was too humid, pumping water when the soil was dry, turning off the pump by the water level sensor. The scheduled timer also works as it should, however, currently has a set interval coded for when to turn off. We were also able to demonstrate our mildew detection by spreading Baking Soda on leafy greens as a stand-in and show that an alert message was sent to the Android device.

Overall the project was successful and met all of our primary goals and requirements. Given additional time the phone application could use some extra features. Our Raspberry Pi currently publishes images of the garden, but we ran out of time to get the application to display them. Since demo night we were almost able to add an additional feature to our project which actually displays the images that are published by the Raspberry Pi through MQTT, we were able to get a bitmap popup, but had difficulty integrating it into a fragment.

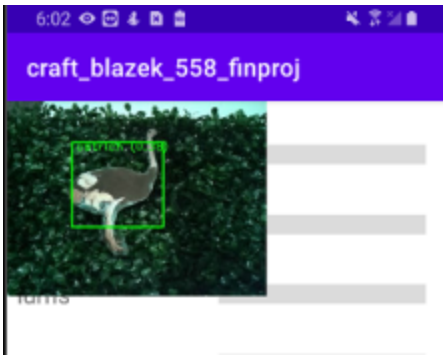

Figure 8: Image Received from the Garden

As mentioned the scheduler currently only has a start time available, most seed starts need about the same amount of light hours but not all so adding the stop time would be a good thing to add in the future. We could also add controls to change the desired relative humidity level, and options to switch between gardens. We are currently subscribing and publishing to topics for both garden setups at the same time, it would be better if the topics were toggled for specific gardens.

The physical build could also be improved upon. Currently the water reservoir needs to be refilled periodically, it would be better to have a main water line with a solenoid switch to activate the water. The downside of this approach is that it makes the unit less mobile since it needs to be connected so would be better used for larger setups.

5.0 Setup Project/Software

To set up the project on the Raspberry Pi: copy the files in the folder named ‘craft_blazek_558_finproj_RPi’ onto a Pi running the Raspberry Pi OS. Install any packages that are missing from your device. Follow the instructions in ‘TensorFlow HowTo Notes.docx’ to setup TensorFlow on the Raspberry Pi. Follow the diagram in section 2 of this document for the physical build. Once complete you should be able to run the ‘client.py’ file to start the firmware running.

The Android application is much more straightforward. Load the project in the ‘craft_blazek_558_finproj_android’ folder in android studio and start the application with your android device connected to your computer. The app should load and be able to connect to your Raspberry Pi. This will publish and subscribe to the topics previously designated on both devices and will need to be changed if you want to use different topics or duplicated if you have numerous garden setups.

6.0 Contribution

6.1 Project Delegation

David Craft - Hardware build for the soil moisture sensor, water pump, temperature/ humidity sensor, fan and relay switch to control the lights and heat mat. David also worked on the development of the controlling firmware to run on the Raspberry Pi and on the android application to monitor and control the garden.

Josh Blazek - Worked on the camera / tensorflow functionality that will run on the Raspberry Pi to detect powdery mildew and mold growth on the plants or soil. Josh also worked on the development of the android application to add fragments. Josh also created the promotion video for our project.

6.2 List of Deliverables:

- Project proposal
- Narrated Video/Demo
- PDF of demo presentation slides
- Final Design Report
- Source Code from Github
    - Rpi firmware
    - Android Studio project

7.0 Document Revision

02/27/2022 - Version 0.5 - Initial document
02/28/2022 - Version 1.0 - Final document

8.0 References

1. MQTT broker. The Public MQTT Broker by HiveMQ - Check out our MQTT Demo. (n.d.). Retrieved March 3, 2022, from https://www.hivemq.com/public-mqtt-broker/
2. Documentation; android developers. Android Developers. (n.d.). Retrieved March 3, 2022, from https://developer.android.com/docs
3. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017, April 17). MobileNets: Efficient convolutional neural networks for Mobile

Vision Applications. arXiv.org. Retrieved March 17, 2022, from https://doi.org/10.48550/arXiv.1704.04861

4. Tan, M., & Le, Q. V. (2020, September 11). EfficientNet: Rethinking model scaling for Convolutional Neural Networks. arXiv.org. Retrieved March 17, 2022, from https://doi.org/10.48550/arXiv.1905.11946
5. Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015, February 21). Microsoft Coco: Common Objects in Context. arXiv.org. Retrieved March 17, 2022, from https://doi.org/10.48550/arXiv.1405.0312
6. Tensorflow. GitHub. (n.d.). Retrieved March 15, 2022, from https://github.com/tensorflow